Manuel de PureBasic
6.00

http://www.purebasic.com/

1er juillet 2022

Table des matières

Première partie

Thèmes généraux

Chapitre 1

Introduction

PureBasic est un langage de programmation de "haut niveau" basé sur les règles du langage BASIC. Il est identique à tout autre compilateur BASIC que vous auriez pu déjà utiliser. PureBasic a été créé pour être aussi accessible au débutant qu'à l'expert, il est donc très facile à apprendre. La compilation est très rapide, presque instantanée. Nous avons consacré beaucoup de temps et d'efforts pour vous proposer un langage rapide, fiable et convivial.
La syntaxe de PureBasic est simple, mais ses possibilités sont infinies grâce à certaines caractéristiques évoluées comme, entre autres, les pointeurs, structures, procédures, listes dynamiques, les maps, les interfaces, la programmation modulaire (modules), un assembleur en ligne, etc. Le programmeur expérimenté n'aura aucune difficulté à accéder aux structures du système d'exploitation et aux API's.
PureBasic est un langage portable qui fonctionne actuellement sur les ordinateurs PC dotés du système d'exploitation Windows, Linux ou Mac OS X. Cela signifie qu'un même programme peut être compilé sur chacune de ces machines et en exploiter toute la puissance. PureBasic est un compilateur. Il n'utilise donc pas de code intermédiaire ou de machine virtuelle, mais produit un code optimisé directement exécutable par la machine ou le système d'exploitation sur lequel il est compilé. Les bibliothèques externes sont écrites et optimisées manuellement en assembleur. Les commandes sont donc très rapides, souvent plus rapides que leurs équivalentes écrites en langage C/C++.
Pour information, les passionnés de machines anciennes trouveront une version en opensource pour AmigaOS (680x0 et PowerPC) ici

Caractéristiques techniques

- Supporte nativement les microprocesseurs x86 et x64
- 83 bibliothèques natives
- Des centaines de fonctions
- Définitions intégrées de tableaux, listes dynamiques, structures complexes, maps, pointeurs et variables
- Types supportés : Byte (8-bit), Word (16-bit), Long (32-bit), Quad (64-bit), Flottant (32 et 64-bit) et Caractère (8 ou 16-bit)
- Types définis par l'utilisateur (structures)
- Définition intégrée des chaines de caractères avec de nombreuses fonctions dédiées, y compris en mode unicode
- Puissant support des macros
- Support des constantes et valeurs octales et hexadécimales
- Réducteur d'expression (regroupement des constantes et valeurs explicites)
- Calcul arithmétique standard avec respect de la priorité des opérateurs et parenthèses : +, -, /, *, and, or, lsl, asl, lsr, asr
- Compilation extrêmement rapide
- Support des procédures pour une programmation structurée avec variables globales, statiques et locales
- Supporte tous les mots-clé évolués du langage Basic : If-Else-EndIf, Repeat-Until, For-Next, etc.
- Support de bibliothèques externes pour la manipulation d'objets Windows : Images, fenêtres, composants graphiques, DirectX, etc.

- Les bibliothèques externes sont également toutes écrites en langage assembleur optimisé pour plus de rapidité et de compacité
- Les API's de Windows (Linux et OSX) sont supportées et considérées comme des mots-clé du BASIC
- Langage assembleur intégré permettant d'insérer toute commande ou routine assembleur dans le corps du programme Basic
- Constantes et structures précompilées pour une compilation encore plus rapide
- Débogueur intégré pour suivre l'exécution d'un programme et corriger les erreurs plus facilement
- Options de compilation configurables
- Editeur dédié avec syntaxe automatiquement surlignée
- Système convivial, facile à installer, à comprendre et utiliser
- Entièrement en français
- SDK Visual C
- Compilation de programme fenêtré, console ou DLL
- Création de fenêtres wysiwyg intégré
- Création de fenêtres avec réorganisation automatique des gadgets (layout) avec la bibliothèque 'Dialog'
- 33 gadgets natifs
- 1 gadget spécialisé dans l'OpenGL
- 1 gadget scintilla
- Processus, thread, mutex, semaphore
- Drag'n drop
- DPI pour MS Windows@ LineBreak - QT and GTK3 pour Linux
- Sous système DirectX et OpenGL
- Import de bibliothèques statiques (lib) ou dynamiques (dll, so, etc)
- Dessin avec antialiasing avec la bibliothèque 'Vector'
- Fonction 3D avec le moteur 3D OGRE
- Pseudotypes : p-ascii, p-utf8, p-bstr, p-unicode, p-variant
- Array, listes, maps, Base de données (MySQL, SQLite, ODBC, PostgreSQL, Maria)
- json, xml
- Expression régulière
- http, ftp, mail, application server/client
, CGI et FastCGI
- Cypher
- OnError
- Compresseur/décompresseur : BriefLZ, JCALG1, LZMA, Tar, Zip
- ini file
- Imprimante, port série
- Runtime

~~Voici la liste exhaustives de toutes les bibliothèques que propose PureBasic :~~

```
  Fenêtres & systèmes :
    arrays
    cgi
    cipher
    clipboard
    console
    database
    date
    debugger
    desktop
    dialog
    dragdrop
    file
    filesystem
    font
    ftp
    gadget
    help
    http
    json
    library
```

```
    lists
    mail
    maps
    math
    memory
    menu
    network
    onerror
    packer
    preference
    printer
    process
    regularexpression
    requester
    runtimes
    scintilla
    serialport
    sort
    statusbar
    string
    system
    systray
    thread
    toolbar
    window
    xml

2D & Multimédia :
    2ddrawing
    audiocd
    font
    image
    imageplugin
    joystick
    keyboard
    mouse
    movie
    music
    screen
    sound
    soundplugin
    sprite
    vectordrawing

 Moteur 3D :
    billboard
    camera
    engine3d
    entity
    entityanimation
    gadget3d
    joint
    light
    material
    mesh
    node
    nodeanimation
    particle
    sound3d
```

```
    specialeffect
    spline
    staticgeometry
    terrain
    text3d
    texture
    vehicle
    vertexanimation
    window3d
```

Chapitre 2

Licence et Conditions

Ce programme est proposé sans aucunes garanties. Fantaisie Software ne peut en aucun cas être tenu responsable des dégâts occasionnés par PureBasic. Vous utilisez ce logiciel à vos risques et périls.
La version de démonstration de PureBasic peut être distribuée librement tant que le contenu de l'archive reste intacte. Il n'est permis de modifier, changer l'organisation de l'archive sans la permission écrite de Fantaisie Software.
La version complète de PureBasic ne doit en aucun cas être cédée à une personne autre que l'acheteur du logiciel.
Fantaisie Software détient tous les droits sur PureBasic et ses composants. Aucun module ne peut être utilisé dans une autre application sans l'autorisation de Fantaisie Software. Il est interdit d'encapsuler ou d'utiliser directement les commandes de haut niveau de PureBasic dans un autre langage de programmation que ce soit sous forme de bibliothèques dynamiques (DLL) ou statiques. Cette règle ne s'applique pas pour les utilisateurs possédant une licence de PureBasic.
PureBasic uses the Scintilla editing component and its related license can be consulted here .
PureBasic uses GCC compiler and its related license can be consulted here and here
The PureBasic uses FASM flat assembler and its related license can be consulted here
The PureBasic uses PellesC toolchain and its related license can be consulted here
PureBasic built-in libraries uses opensources components, the licenses can be consulted here and here

Chapitre 3

Configuration système requise

PureBasic fonctionne sur les versions 32 bits et 64 bits de Windows XP, Vista, Windows 7, Windows 8 et Windows 10, Linux (noyau 2.2 et ultérieur) et MacOS X (10.8.5 et ultérieur).
Si vous rencontrez un problème sur l'une des configurations ci-dessus, n'hésitez pas à nous en informer.

Windows

Afin d'utiliser les nombreuses fonctions graphiques 3D ainsi que les Sprites , il est nécessaire d'utiliser la version 9.0c de DirectX. Windows Vista et les versions ultérieures de Windows contiennent déjà cette version de DirectX, mais pour les versions antérieures de Windows, il faudra procéder à une installation manuelle.
Vous pouvez la télécharger et l'installer via le "DirectX End-User Runtime Web Installer" ici : http ://www.microsoft.com/en-us/download/details.aspx ?displaylang=en&id=35.
En français : https ://www.microsoft.com/fr-fr/download/details.aspx ?id=20265

Linux

Paquets nécessaires pour utiliser PureBasic sous Linux :

- sdl 1.2 devel (pour les commandes multimédias)
- gtk 3 (Pour les programmes GUI)
- libstdc++ devel
- gcc correctement installé
- iodbc et iodbc-devel doivent être installés pour utiliser les commandes Database (voir www.iodbc.org)
- libwebkit doit être installé pour utiliser WebGadget() .
- xine et xine-devel pour les commandes Movie Pour plus d'information, consulter les fichiers INSTALL et README.

MacOS X

Les outils de développement Apple (Apple Developer Tools) doivent être installés avant de pouvoir utiliser PureBasic. Ils sont fournis sur les CDs (ou DVD) d'installation de MacOS X ou disponible sur le site web d'Apple : http ://developer.apple.com/
Pensez à utiliser la dernière version des différents outils (par exemple XCode), adaptée à la bonne version de votre OS ! Les outils de ligne de commandes doivent être installés depuis XCode.

Chapitre 4

Installation

Pour installer PureBasic, cliquez sur l'icône d'installation et suivez les instructions. Cliquez ensuite sur l'icône PureBasic qui se trouve dans le menu démarrer ou sur le bureau.
Pour utiliser le compilateur en ligne de commande, il est conseillé de rajouter le répertoire "compilers\" de l'installation PureBasic à la variable d'environnement 'PATH'. Ainsi la commande 'pbcompiler' sera accessible à partir de n'importe quelle console.
Note : Pour éviter les conflits avec une ancienne version, installez toujours une nouvelle version de PureBasic dans un nouveau répertoire .

Chapitre 5

Obtenir PureBasic

PureBasic est un langage de programmation très performant à un prix raisonnable. En achetant PureBasic, vous encouragez son développement et vous participez à son évolution future. Les mises à jour sont gratuites et illimitées, ce qui implique que vous ne paierez PureBasic qu'une seule et unique fois, contrairement à de nombreux autres logiciels. Encore plus fort, vous avez droit à toutes les versions de PureBasic pour toutes les plateformes actuellement supportées ,Windows, Linux et MacOS (AmigaOS pour les versions antérieures). Merci encore pour votre confiance et votre support !

La version de démonstration est limitée de la manière suivante :

- Impossible de créer une DLL
- Impossible d'utiliser les fonctions externe (API de l'OS)
- Pas de kit de développement pour ajouter des libraries à PureBasic
- Nombre maximum de lignes pour un programme : 800

Prix : La version complète de PureBasic :

Prix pour la version complète : 79 Euros
Prix spéciaux pour les licences entreprises (499 euros) et licences éducations (199 euros pour une classe).

Paiement en ligne sécurisé

http ://www.purebasic.com
La version complète sera envoyée par e-mail (email contenant un lien à télécharger) dès réception de la commande.

Chapitre 6

Contacts

Veuillez envoyer vos suggestions, rapports de bugs, ou si vous voulez simplement nous contacter à une des adresses suivantes :

Frédéric 'AlphaSND' Laboureur

Fred 'AlphaSND' est le fondateur de Fantaisie Software et le programmeur principal de PureBasic. Toutes les suggestions, rapports de bugs etc... lui seront adressés à :

Courrier :

Fantaisie Software
10, rue de Lausanne
67640 Fegersheim
France

e-mail : fred@purebasic.com

André Beer

André se charge de la traduction et du support allemand de PureBasic (manuel et site web).
e-mail : andre@purebasic.com

Chapitre 7

Remerciements

Nous voulons remercier les nombreuses personnes qui nous ont aidés dans cet ambitieux projet. Il n'aurait pas pu prendre vie sans eux !

- Tous les utilisateurs enregistrés : Pour supporter activement le développement de ce logiciel... Encore merci !

Codeurs

- **Roger Beausoleil** : Le premier à croire vraiment en ce projet, et son aide précieuse dans le design initial de PureBasic.
- **Andre Beer** :Pour prendre le temps de traduire chaque nouvelle version en Allemand...
- **Francis G.Loch** : Pour avoir corrigé toutes les fautes d'orthographe dans le manuel anglais. Encore merci !
- **Steffen Haeuser** : Pour avoir donné de son temps lors de la création de la version Amiga PowerPC (PPC) de PureBasic.
- **Martin Konrad** : Un 'bug reporter' fantastique qui a mis à nus la plupart des bugs présents dans PureBasic Amiga.
- **James L. Boyd** : Pour avoir trouvé beaucoup de bugs dans la version Windows x86 et pour nous avoir donné une tonne de boulot supplémentaire :).
- **Les** : Pour avoir corrigé la version anglaise du site web et du guide de référence de PureBasic.
- **L'équipe NAsm** : Pour avoir créé un assembleur étonnant utilisé pour le développement des bibliothèques x86
- **Tomasz Grysztar** : Pour FAsm, un excellent assembleur x86 actuellement utilisé par PureBasic. http ://flatassembler.net/
- **Pelle Orinius** : Pour ses excellents outils (linker et compilateur de ressources de PellesC) qui sont actuellement utilisés par PureBasic.
- **Jacob Navia** :Pour son linker (Lcc32) qui était utilisé dans les versions antérieures de PureBasic
- **Frank Wille** : Pour nous avoir permis d'inclure ses deux assembleurs (PhxASS et pasm) dans les versions 680x0 et PowerPC de PureBasic. Enfin pour son aide en ce qui concerne la programmation assembleur PPC.
- **Danilo**, **Rings**, **Paul**, **Franco** and **MrVain/Secretly** : Pour les rapports de bugs et les suggestions interminables qui ont permis de stabiliser et d'améliorer PureBasic x86 d'une manière très rapide.
- **Sylvain B.** : Pour avoir traduit la première partie de la documentation française.
- **Francois Weil** : Pour avoir traduit le reste (une grande majorité) de la documentation française, et ce en moins d'une semaine ! Encore merci...
- **David 'Tinman' McMinn** et **Timo 'Freak' Harter** : Pour améliorer considérablement l'aide du PureBasic et ainsi sa découverte au plus grand nombre !
- **Danilo** : Pour avoir fait un travail énorme sur l'éditeur et même sur le jeu interne des commandes, sans

oublier ses excellentes suggestions quant à l'optimisation et la réduction des exécutables... Encore merci !
- **Marc Vitry** : Pour avoir partagé ses sources de sa bibliothèque 'SerialPort', qui ont servi de base à celle incluse dans le PureBasic
- **Stefan Moebius** : Pour avoir envoyé les sources de son sous-système DX9, qui ont servi de base au sous-système DX actuel
- **Comtois**, **G-Rom** et **T-Myke** : pour avoir amélioré drastiquement la partie 3D de PureBasic ! Ca compte !
- **Gaetan Dupont-Panon** : Pour le fabuleux nouvel éditeur visuel de fenêtres, qui fonctionne sour Windows, Linux et OS X !
- **Jean R. VIALE** : Pour avoir amélioré et corrigé de manière conséquente l'aide française, et pour continuer à le faire !

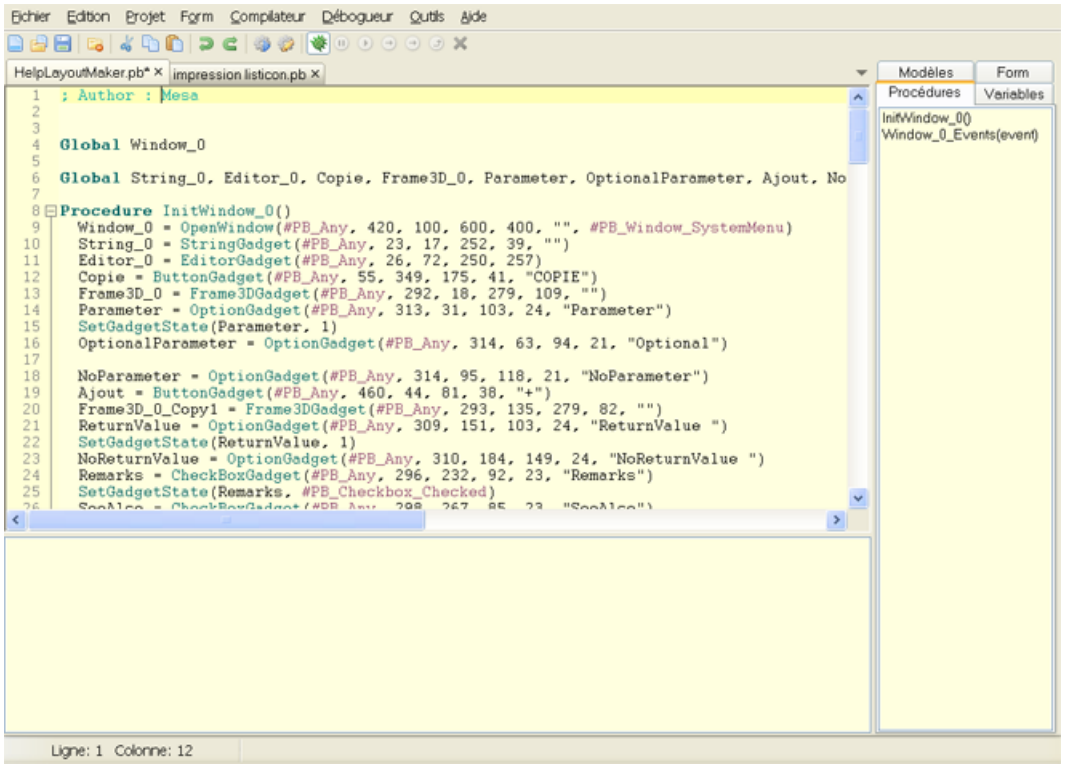
Deuxième partie

L'IDE de PureBasic

Chapitre 8

Débuter avec l'IDE

L'IDE PureBasic est un environnement complet qui permet la création et l'édition de fichiers sources, leur compilation, débogage ainsi que la création de l'exécutable final. Il sert d'interface au compilateur ainsi qu'au débogueur .
La fenêtre principale de l'IDE se divise en 3 parties :



La zone d'édition du code (en dessous de la barre d'outils)
Tous les codes sources sont affichés ici. Il est possible de passer de l'un vers l'autre à l'aide des onglets situés entre la barre d'outils et la zone d'édition.
La palette d'outils (par défaut située à droite de la zone d'édition)
La palette contient plusieurs outils rendant la programmation plus facile et plus productive. Les outils affichés ici peuvent être configurés, déplacés ou même enlevés si besoin est. Voir Configurer l'IDE pour plus de renseignements.
Le rapport d'activité (situé en dessous de la zone d'édition)
Les erreurs de compilation ainsi que les messages d'erreur du débogueur seront affichés dans cette partie. Il peut être visible ou caché séparément pour chaque fichier.

Le reste de l'interface se compose du menu principal et de la barre d'outils. Cette dernière est

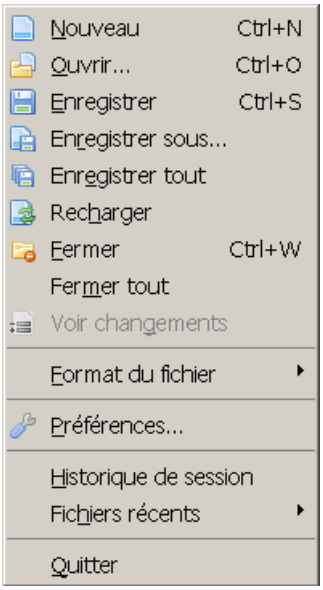
simplement un raccourci des fonctions disponibles dans le menu et peut être complètement paramétrée (chaque fonction du menu peut être enlevée, déplacée ou ajoutée à la barre d'outils). Pour savoir ce que fait un bouton, il suffit de laisser le pointeur de la souris dessus un court instant et une aide apparaîtra (et affichera la fonction du menu correspondante). Les commandes du menu sont expliquées dans une autre section.

Chapitre 9

Gestion des fichiers

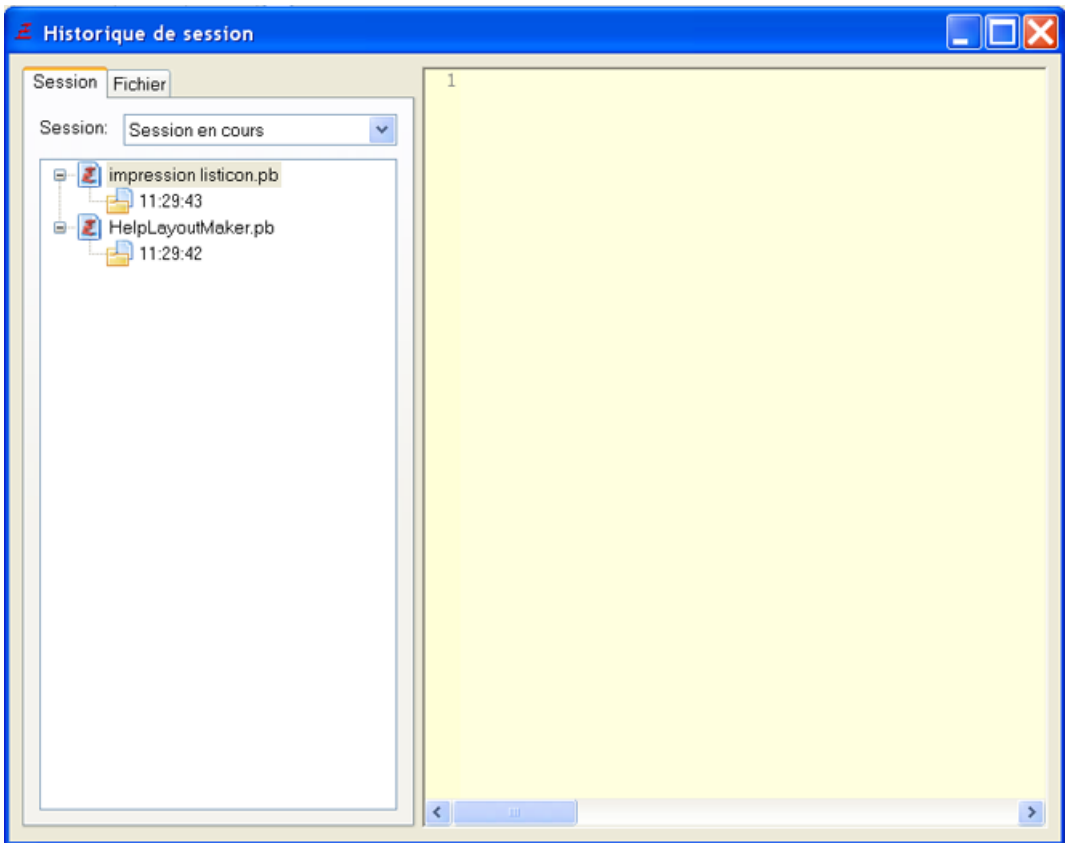
Le menu "Fichier" permet de réaliser les opérations de base tel que l'ouverture ou l'enregistrement des fichiers.
Il est possible d'éditer plusieurs codes sources simultanément, et de passer de l'un à l'autre grâce aux onglets situés sous la barre d'outils. Par défaut, les raccourcis claviers Ctrl+Tab et Ctrl+Shift+Tab permettent de passer au fichier suivant (ou précédent).
L'IDE permet l'édition de fichiers texte qui ne sont pas du code source. Dans ce mode "texte brut", les caractéristiques liées au code tels que la coloration synthaxique, la correction des cas, l'auto-completion automatique sont désactivées. Lors de l'enregistrement de ces fichiers, l'IDE n'ajoutera pas ses paramètres à la fin du fichier, même si cela est configuré pour les fichiers de code dans les préférences .
Un fichier qui contient du code ou non dépend de son extension. Les extensions de fichiers standard PureBasic (pb, pbi et pbf) sont reconnues comme des fichiers contenant du code. Plusieurs extensions de fichier peuvent être reconnues comme des fichiers de code en les configurant dans la section "Editeur" des Préférences .

Contenu du menu "Fichier"



Nouveau
Créé un nouveau code source vide.
Ouvrir
Ouvre un code source existant pour l'éditer.
Les fichiers de type 'texte' seront tous chargés dans la zone d'édition. Il est aussi possible d'ouvrir des fichiers binaires et dans ce cas ils seront ouverts par le visualisateur de fichier interne.

Enregistrer
Enregistre le source en cours d'édition sur le disque. S'il n'a pas encore été enregistré, son nom et son emplacement devront être définis à l'aide de la fenêtre de dialogue, sinon le code sera enregistré dans le fichier précédent.
Enregistrer sous
Enregistre le code source en cours d'édition à un emplacement différent. Une boîte de dialogue demandant le nom du nouveau fichier sera affichée. L'ancien code source restera inchangé.
Enregistrer tout
Enregistre tous les codes source ouverts.
Recharger
Recharge le fichier courant à partir du disque. Cela annulera tous les changements non sauvegardés.
Fermer
Ferme le code source en cours d'édition. Si c'était le dernier fichier, l'IDE affichera un nouveau fichier vide.
Fermer tout
Ferme tous les codes source en cours d'édition. L'IDE affichera un nouveau fichier vide.
Voir changements
Affiche les modifications qui ont été apportées au code source courant par rapport à la version qui existe sur le disque dur.
Format du fichier
Dans ce sous-menu, il est possible de sélectionner le format d'encodage ainsi que les terminaisons de lignes du fichier courant. L'IDE supporte les fichiers textes ASCII ou UTF-8. Les terminaisons de ligne supportées sont : Windows (CRLF), Linux/Unix (LF) et MacOSX (CR). Les paramètres par défaut appliqués lors de la création d'un nouveau fichier peuvent être changés dans les préférences de l'IDE.
Préférences
Cette fenêtre permet de contrôler le look et le comportement de l'IDE pour qu'il s'adapte aux besoins de chacun. Pour une description détaillée des fonctions disponibles, voir Configurer l'IDE .
Historique de session
L'historique de session est un outil puissant qui enregistre régulièrement les modifications apportées aux fichiers ouverts dans l'IDE, dans une base de données. Une session est créée lors du lancement de l'IDE, et est fermée lorsque l'IDE se ferme. Cette option est utile pour revenir à une version antérieure d'un fichier ou pour retrouver un fichier supprimé ou corrompu. C'est un peu comme un outil de sauvegarde des sources mais limité dans le temps (par défaut, un mois d'enregistrement). Il n'est pas destiné à remplacer un vrai système de gestion des sources comme SVN ou GIT. Le code source sera stocké sans chiffrement, donc si vous travaillez sur un code source sensible, assurez-vous d'avoir ce fichier de base de données dans un endroit sûr, ou désactiver cette fonction.
Pour configurer l'outil historique de la session, voir préférences .



Fichiers récents
Le sous-menu affiche la liste des dix derniers fichiers ouverts. Sélectionner un fichier dans le sous-menu l'ouvrira à nouveau.

Quitter
Cela permet bien entendu de quitter l'IDE. Si un ou plusieurs fichiers ne sont pas enregistrés, une confirmation pour chacun des fichiers sera demandée.

Chapitre 10

Edition du code

L'IDE du PureBasic se comporte comme un éditeur de texte classique en ce qui concerne la saisie de texte. Les touches de curseur, Page précédente, Page suivante, Début et Fin permettent de se déplacer dans le code. Ctrl+Début se rend au début du code et Ctrl+Fin se déplace à la fin.
Les raccourcis par défaut Ctrl+C (copier), Ctrl+X (couper) et Ctrl+V (coller) servent à l'édition. La touche Insérer change le mode d'édition pour déterminer si le texte doit etre remplacé ou ajouté (le mode par défaut est l'ajout). La touche Supprimer supprime une lettre à droite du curseur. Maintenir la touche Shift appuyée et utiliser les flèches permet de sélectionner du texte.
Les raccourcis Ctrl+Maj+Haut et Ctrl+Maj+Bas permettent de déplacer une ligne vers le haut ou vers le bas.
Le raccourci Ctrl+D permet de dupliquer une sélection.
Sous Windows, les raccourcis Ctrl+Plus et Ctrl+Moins permettent de zoomer et de dézoomer l'affichage de tous les onglets et Ctrl+0 rétablit le zoom par défaut. Les utilisateurs de Linux et OSX peuvent les créer eux-mêmes. Le raccourci Ctrl+molette de la souris ne zoom que l'onglet en cours.
Les raccourcis Ctrl+Tab et Ctrl+Maj+Tab permettent de se déplacer dans les onglets ouverts.
De nombreux raccourcis sont à votre disposition, comme Ctrl+Maj+U/L/X pour changer la casse, etc.
Rendez-vous dans les préférences , dans le menu Fichiers/Préférences/Général/Raccourcis.
Au delà des fonctions de base, l'IDE a de nombreuses fonctionnalités dédiées à la programmation PureBasic.

Indentation

Quand un retour à ligne est effectué, l'indentation (nombre d'espaces/tabulations en début de ligne) de la ligne courante et de la ligne suivante sera calculée automatiquement en fonction du mot-clé qui est présent sur la ligne. Un mode "block" est aussi disponible. Dans ce cas, la nouvelle ligne aura la même indentation que la ligne courante. Tout ceci est paramètrable dans les preférences .

Caractères de tabulation

Par défaut, l'IDE n'insère pas une vraie tabulation quand la touche Tab est utilisée. En cas d'échange de source, ceci permet d'éviter les problèmes d'affichage inhérents aux normes de tabulation utilisées.
A la place, deux caractères espaces sont insérés. Ce comportement peut être modifié dans les préférences de l'IDE (voir Configurer l'IDE).

Comportement particulier de la touche Tab

Quand la touche Tab est utilisée alors que la sélection est vide ou ne contient que quelques caractères, elle agit comme décrit ci-dessus (insertion d'un nombre d'espaces, ou d'une vraie tabulation selon le

réglage défini dans les préférences de l'IDE).
Quand la touche Tab est utilisée alors qu'une ou plusieurs lignes complètes sont sélectionnées, des espaces (ou des tabulations, en fonction de la configuration) sont insérés au début de chaque ligne. Cela permet d'indenter rapidement des blocs complets de code.
Quand la combinaison de touches Shift+Tab est utilisée alors que plusieurs lignes sont sélectionnées, les espaces/tabulations au début de chaque ligne sont supprimés afin de réduire l'indentation du bloc complet.

Retrait (Indentation) / Alignement des commentaires :

Similaire au comportement de l'onglet spécial ci-dessus, les raccourcis clavier Ctrl+E et Ctrl+Maj+E (CMD+E et CMD+Maj+E sur OSX) peuvent être utilisés pour modifier le retrait des commentaires sélectionnés. Cela aide à aligner les commentaires à la fin du code, pour le rendre encore plus lisible. Les raccourcis utilisés peuvent être configurés dans les préférences .

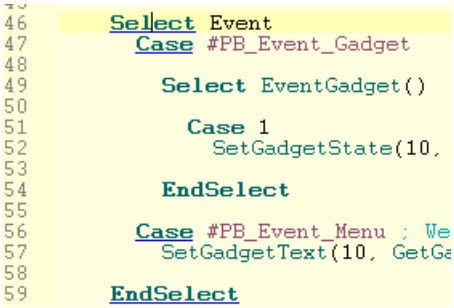
Sélection des blocs de code :

Le raccourci Ctrl+M (CMD+M sur OSX) peut être utilisé pour sélectionner le bloc de code qui contient la position du curseur (par exemple un bloc 'If', une boucle ou une procédure). Une utilisation répétée du raccourci sélectionne d'autres blocs de code environnant.
Le raccourci Ctrl+Maj+M (CMD+Maj+M sur OSX) a la fonction inverse et rétablit la sélection du bloc précédemment sélectionné avec Ctrl+M.
Les raccourcis utilisés peuvent être configurés dans les préférences .

Double-cliquer sur un mot

En principe, double-cliquer sur un mot le sélectionne. Néanmoins, dans quelques cas, le comportement est différent :
Maintenir la touche Ctrl appuyée et double-cliquer sur le nom d'une procédure , définie dans le même fichier, permet de se rendre directement à sa déclaration. Si la procédure se trouve dans un autre fichier, celui-ci doit déjà être ouvert dans l'IDE.
Double-cliquer sur un mot-clé IncludeFile ou XincludeFile ouvrira le fichier dans l'IDE (le nom du fichier doit être spécifié en entier, sans utiliser de constante par exemple).
De la même manière, double-cliquer sur le mot-clé IncludeBinary aura pour effet d'ouvrir le fichier dans le visualisateur de fichier intégré à l'IDE.

Marqueur de parenthèse et de mots-clés :



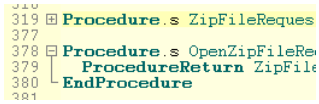
Quand le curseur est sur une parenthèse ouvrante ou fermante, l'IDE mettra en surbrillance l'autre parenthèse qui lui correspond. Si aucune parenthèse ne correspond (ce qui est une erreur de syntaxe en PureBasic), l'IDE affichera la parenthèse en rouge. Le même concept est appliqué aux mots-clés. Si le curseur est sur un mot-clé comme "If", l'IDE soulignera ce mot-clé et tous les autres mots-clés qui lui correspond, comme "Else", "ElseIf" ou "EndIf". Si il manque des mots-clés, il sera souligné en rouge. Le

menu ”Aller au mot-clé correspondant” décrit ci-dessous peut être utilisé pour aller rapidement d’un mot-clé à l’autre.
Le marqueur de parenthèse et de mots-clés est configurable dans les préférences .

Aide syntaxique dans la barre d’état

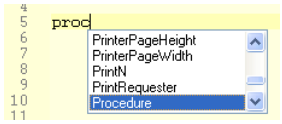
L’IDE affichera les paramètres nécessaires pour chaque fonction PureBasic qui est en cours de frappe. Cela rend plus facile la saisie, en montrant les paramètres qui sont indispensables. Cela fonctionne aussi pour les procédures , prototypes , interfaces ou fonctions importées , s’ils sont déclarés dans le même fichier source ou dans le même projet .

Options de pliage



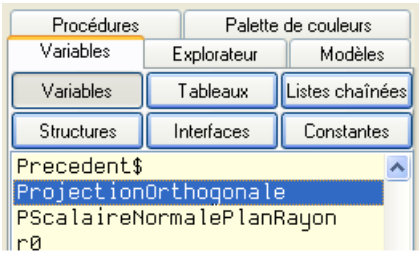
Quand un mot-clé de pliage est rencontré (Procedure / EndProcedure par défaut, d’autres peuvent être ajoutés) l’IDE marque la région délimitée par ces deux mots-clés par un [-] sur la ligne du premier mot et une ligne verticale jusqu’à la fin du bloc.
En cliquant sur le [-], il est possible de cacher (replier) le contenu du bloc pour garder une meilleure vue d’ensemble sur les codes sources conséquents. Le [-] se transformera alors en [+]. En cliquant de nouveau dessus, le code sera de nouveau affiché (déplié).
Note : Même si l’état des blocs est préservé lors de l’enregistrement du fichier, le fichier lui-même contient bien sûr toutes les lignes de codes. Cette option affecte seulement l’affichage du code source, pas son contenu.
Deux autres délimiteurs de replis sont activés par défaut : ”;{” et ”;}”. Comme ”;” marque le début d’un commentaire en PureBasic, ils seront ignorés par le compilateur. Néanmoins, ils permettent de placer des délimiteurs à des endroits arbitraires qui ne correspondent pas forcément à un mot-clé PureBasic.

Auto-complétion



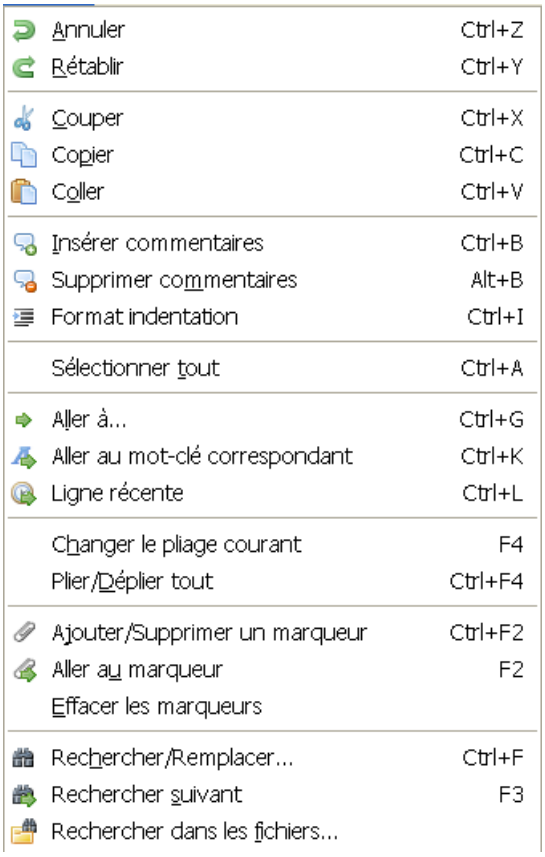
Pour ne pas avoir à se souvenir de tous les noms des commandes PureBasic, il y a l’auto-complétion qui peut rendre bien des services.
Après avoir tapé le début du nom de la commande, une liste des commandes possibles qui débute par ce mot s’affichera. Une liste de choix est aussi affichée après avoir tapé ”\” sur une variable liée à une structure ou une interface.
Une fois le mot sélectionné à l’aide des flèches il peut être inséré dans le code source en appuyant sur la touche Tab. Il est aussi possible de continuer à taper les lettres pendant que la liste est affichée, pour réduire le choix. Si plus aucune proposition n’est disponible, alors la boîte se fermera automatiquement. La touche Echappement permet de fermer la liste à tout moment. La liste se ferme aussi automatiquement si un bouton souris est utilisé dans l’IDE.
Note : Pour configurer le contenu qui sera affiché dans la liste d’auto-complétion, ainsi que ses nombreuses fonctionnalités, voir Configurer l’IDE . Par exemple, il est possible de désactiver l’apparition automatique de la liste (Ctrl+Espace sera alors necessaire pour l’afficher).

Panneau d'outils



La plupart des outils qui rendent la navigation et l'édition de code sources plus aisés peuvent être ajoutés au panneau d'outils situé sur le côté de la zone d'édition. Pour avoir un aperçu des outils disponibles et pour les configurer, voir Les outils intégrés .

Le menu édition



Les rubriques suivantes expliquent les commandes disponibles dans le menu édition. A noter que bon nombre de ces commandes sont aussi présentes dans le menu contextuel de la zone d'édition.

Annuler

Annule la dernière action effectuée dans la zone d'édition. Il est possible d'annuler plusieurs actions à la suite.

Rétablir

Refait la dernière action annulée par la commande "Annuler".

Couper

Copie le texte sélectionné dans le presse-papier et le supprime du code.

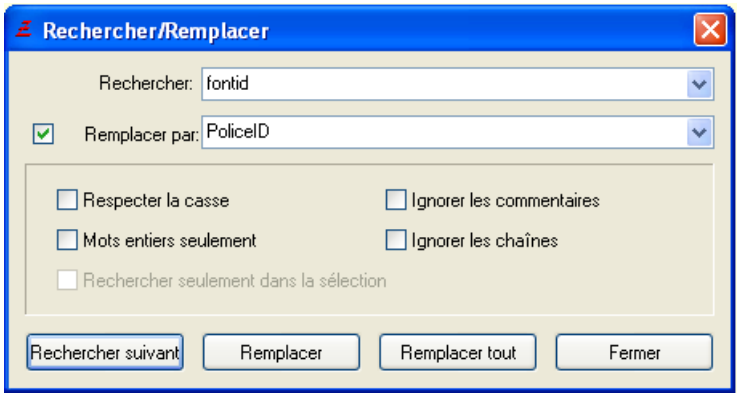
Copier

Copie le texte sélectionné dans le presse-papier.

Coller

Insère le contenu du presse-papier dans le code, à la position du curseur. Si une partie du code était sélectionnée, elle sera alors remplacée par le contenu du presse-papier.

Insérer commentaires
Insère un commentaire (";") au début de chaque ligne sélectionnée. Cela permet de commenter rapidement un nombre conséquent de lignes.
Supprimer commentaires
Enlève le commentaire (";") du début de chaque ligne sélectionnée. Cela permet de dé-commenter rapidement un nombre conséquent de lignes. C'est la fonction inverse de "Insérer commentaire", mais elle fonctionne aussi sur des commentaires entrés manuellement.
Format et indentation
Reformate le code source sélectionné pour aligner le code en fonction des règles définies dans les préférences .
Sélectionner tout
Sélectionne le code source en entier.
Aller à
Permet de se rendre directement à une ligne donnée.
Aller au mot-clé correspondant
Si le curseur est actuellement sur un mot-clé comme "If", le curseur ira directement sur le mot-clé correspondant (dans ce cas "EndIf").
Ligne récente
L'IDE garde une trace des lignes qui sont visualisées. Par exemple, si la position du curseur est changée par la commande "Aller à" ou en cliquant sur l'outil Navigateur de procédures , cette fonction permet de revenir rapidement à la ligne précédente. L'IDE retient 20 positions différentes.
A noter que l'enregistrement ne se fait que si le déplacement est conséquent (ex : pas si les touches haut/bas sont utilisées).
Changer le pliage courant
Change l'état du point de repli associé à la ligne courante (le déplier ou le replier le cas échéant).
Plier/Déplier tout
Change l'état de tous les points de replis dans le code source, très utile pour cacher toutes les procédures d'un code source (ou pour ré-afficher le code complet en une seule fois).
Ajouter/Supprimer un marqueur
Cette commande permet d'ajouter un marqueur sur la ligne courante (ou de le retirer si il y en avait déjà un). Un marqueur permet de retrouver rapidement une ligne de code. Il est identifié par une petite flèche horizontale (verte, par défaut) à côté de la ligne concernée. Il est possible de passer de marqueur en marqueur en utilisant la commande "Marqueur suivant".
Note : Pour ajouter/retirer un marqueur à l'aide de la souris, il faut maintenir la touche Ctrl appuyée pendant le clic sur la bordure qui affiche les marqueurs (la colonne de numérotation des lignes n'est pas prise en compte).
Aller au marqueur
Se rend au premier marqueur disponible, situé en dessous de la ligne en cours d'édition. Si il n'y a plus de marqueur en dessous de cette ligne, la recherche reprend au début du code source. En utilisant le raccourci "Marqueur suivant" (F2, par défaut) plusieurs fois, il est possible de parcourir tous les marqueurs du code très rapidement.
Effacer les marqueurs
Retire tous les marqueurs du code source en cours d'édition.
Rechercher/Remplacer



Cette fenêtre permet de rechercher un mot (ou une suite de mots) dans le code source et de les remplacer par d'autres mots si nécessaire.

Le bouton "Chercher suivant" démarre la recherche. La recherche peut continuer une fois un terme trouvé en utilisant la commande du menu "Rechercher à nouveau" (raccourci F3 par défaut).
Pour rendre la recherche plus pointue, les options suivantes sont disponibles :
Respecter la casse : Seul le texte qui est strictement identique au mot recherché sera retenu (les minuscules ne sont pas identiques aux majuscules).
Mots entiers seulement : Recherche seulement les termes qui forment un mot. Les termes qui contiennent le mot recherché ne seront pas retenus.
Ignorer les commentaires : Tous les commentaires PureBasic sont exclus de la recherche.
Ignorer les chaînes : Tous les contenus des strings (entre " ") sont ignorés.
Rechercher seulement dans la sélection : recherche seulement dans le texte sélectionné. C'est une option très utile quand elle est utilisée en conjonction avec la fonction "Remplacer tout". Dans ce cas, le remplacement ne s'opèrera uniquement que dans la sélection.
En activant la case à cocher "Remplacer par", la fonction remplacer devient disponible. "Chercher suivant" cherchera toujours l'occurrence suivante, et chaque clic sur le bouton "Remplacer" remplacera le texte trouvé par le contenu du champ "Remplacer par".
En cliquant sur "Remplacer tout", tous les termes correspondants à la recherche situés en dessous du curseur seront remplacés (sauf si l'option "Rechercher seulement dans la sélection" est activée).

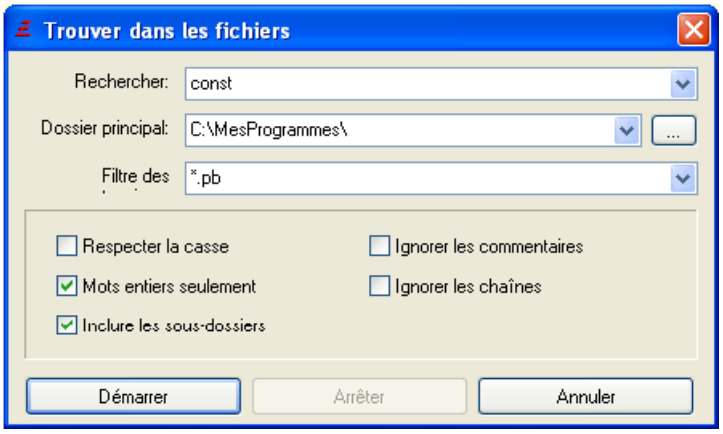
Rechercher suivant

Continue la recherche correspondant à la dernière recherche entamée dans la fenêtre "Chercher/Remplacer".

Rechercher précédent

Continue la recherche en remontant à la recherche précédente entamée dans la fenêtre "Chercher/Remplacer".

Rechercher dans les fichiers



Cette fonction permet de rechercher parmi tous les fichiers d'un répertoire donné (et optionnellement dans ses sous-répertoires).
Il suffit de spécifier le mot (ou la séquence de mots) à rechercher ainsi que le répertoire de base (champ "Répertoire"). Il est possible d'indiquer des filtres pour inclure uniquement les fichiers voulus (ex : *.pb). Plusieurs filtres peuvent être spécifiés, en les séparant par une virgule "," (un filtre vide ou *.* recherche dans tous les fichiers). Des options similaires à la fenêtre "Rechercher/Remplacer" sont disponibles pour rendre la recherche plus pointue.
La case à cocher "Inclure les sous-répertoires" permet de rechercher dans tous les sous-répertoires (d'une manière récursive) du répertoire de base.
Quand la recherche débute, une fenêtre indépendante sera ouverte pour afficher les résultats de la recherche en indiquant le fichier, le numéro de la ligne ainsi que le contenu de la ligne.
En double-cliquant sur une des lignes, le fichier sera automatiquement ouvert dans l'IDE à la ligne concernée.

Chapitre 11

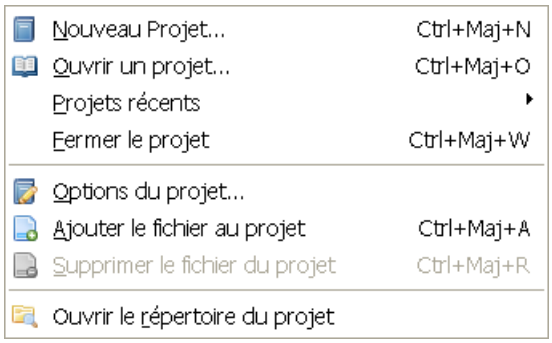
Gestion de projets

L'IDE contient des fonctionnalités avancées pour gérer de larges projets. Ces fonctionnalités sont complètement facultatives, les programmes peuvent être créés et compilés sans recourir à la création d'un projet. Cependant, dès qu'un programme atteint une certaine taille et qu'il se décompose en de nombreux fichiers, il sera probablement plus facile de le gérer au sein d'un projet.

Aperçu de la gestion de projets

Un projet regroupe tous les codes sources et les autres fichiers nécessaires au programme à un seul endroit, avec un accès rapide à tous les fichiers à l'aide de l'outil de projet . Les fichiers sources inclus dans le projet peuvent être scannés automatiquement pour l'auto-complétion même s'ils ne sont pas ouverts dans l'IDE. Donc les fonctions, constantes, variables etc. du projet entier peuvent être utilisés dans l'auto-complétion. Le projet enregistre aussi les fichiers ouverts lors sa fermeture, donc l'espace de travail sera retabli à l'identique lors de la réouverture du projet.
De plus, le projet regroupe toutes les options de compilation à un seul endroit (dans le fichier du projet) et permet même de créer plusieurs cibles de compilation par projet. Une cible de compilation est un ensemble d'options permettant de générer plusieurs programmes distincts, ou plusieurs versions différentes du même programme. Toutes les cibles peuvent être compilées en une seule étape, ce qui permet de gagner en productivité.
Pour compiler un projet à partir d'un script ou d'un makefile, l'IDE accepte une option en ligne de commande pour compiler un projet sans ouvrir d'interface. Voir la section Commutateurs de la ligne de commande pour plus de détails.
Tous les noms de fichiers et les chemins d'un projet sont stockés de manière relative à son fichier de configuration, donc il est facile de déplacer ou de partager un projet, tout en gardant la structure du répertoire intacte.

Le menu Projet

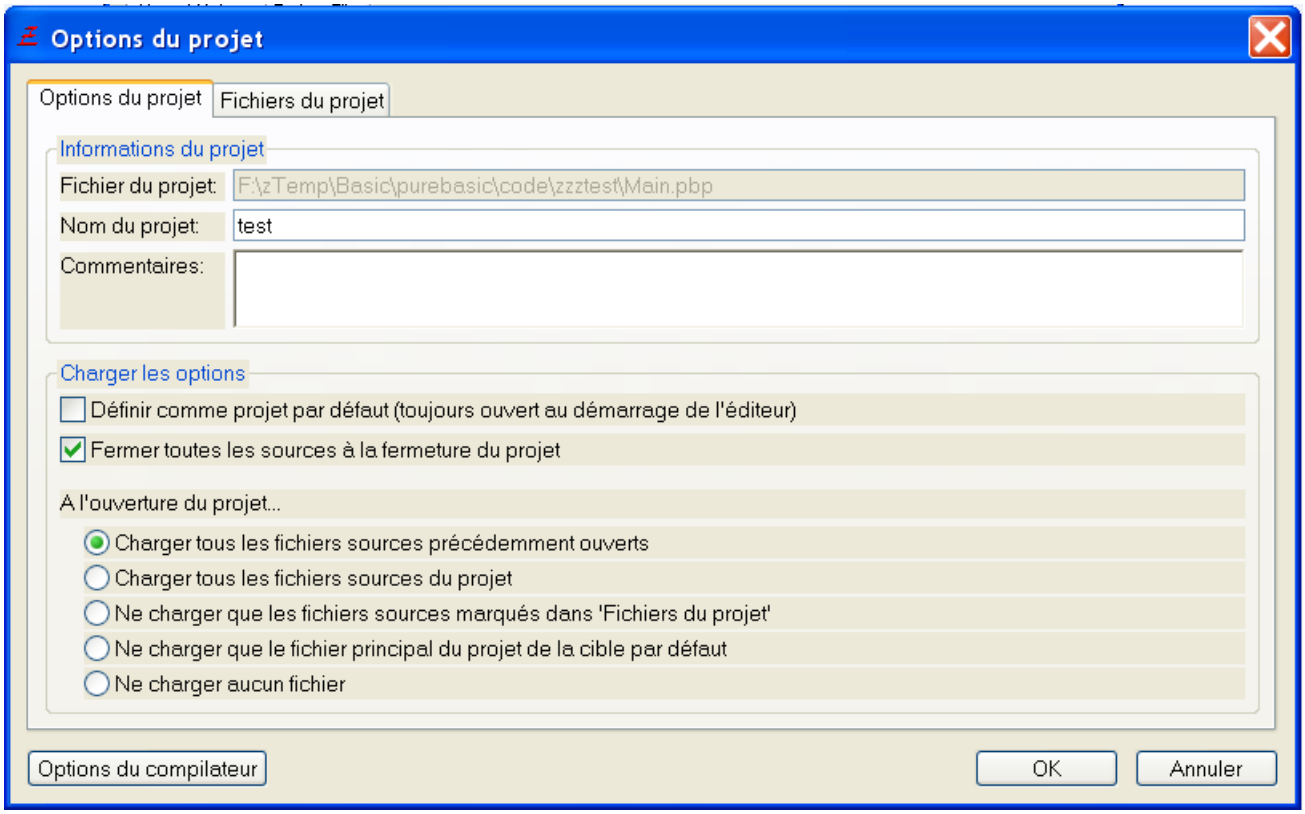


Nouveau projet

Crée un nouveau projet. Si un autre projet était déjà ouvert, il sera fermé. La fenêtre d'options sera ouverte où le nom du projet devra être saisi. La configuration du projet se fera à partir de cette fenêtre.
Ouvrir un projet
Ouvre un projet existant. Si un autre projet était déjà ouvert, il sera fermé. Les fichiers qui étaient déjà ouverts dans ce projet seront automatiquement ré-ouverts (si cette option est activée).
Projets récents
Ce sous-menu affiche la liste des projets les plus récemment ouverts.
Fermer le projet
Ferme le projet actuellement ouvert. La configuration du projet sera enregistrée et les fichiers ouverts associés au projets seront fermés (si cette option est activée).
Options du projet
Ouvre la fenêtre d'options propre au projet (voir ci-dessous pour plus d'informations)
Ajouter le fichier au projet
Ajoute le fichier courant dans le projet courant. Les fichiers qui appartiennent à un projet ont le symbole ">" devant leur nom dans les onglets.
Retirer le fichier du projet
Retire le fichier courant du projet courant.
Ouvrir le répertoire du projet
Ouvre le répertoire qui contient le fichier projet dans l'explorateur de fichier du système.

La fenêtre d'options du projet

La fenêtre d'options du projet est le centre de configuration du projet. Les paramètres relatifs au projet et aux fichiers gérés par le projet sont accessibles ici.



Les paramètres suivants peuvent être modifiés dans l'onglet "Options du projet" :
Nom de fichier du projet
Affiche le nom de fichier du projet. Il ne peut être défini seulement pendant la création du projet.
Nom du projet
Nom du projet. Ce nom sera affiché dans la barre de titre de l'IDE et dans le menu "Projets récents".
Commentaires
Ce champ permet d'ajouter des informations complémentaires sur le projet. Elles seront affichées dans

l'onglet d'information du projet.

Définir comme projet par défaut

Le projet par défaut sera ouvert automatiquement à chaque démarrage de l'IDE. Un seul projet peut être le projet par défaut. S'il n'y a pas de projet par défaut, l'IDE chargera le dernier projet qui était ouvert lors de sa fermeture (s'il y en avait un).

Fermer tous les fichiers lors de la fermeture du projet

Permet de fermer tous les fichiers qui appartiennent au projet lorsque ce dernier est fermé.

A l'ouverture du projet...

Charger tous les fichiers sources précédemment ouverts

Quand le projet est ouvert, tous les fichiers sources précédemment ouverts seront chargés.

Charger tous les fichiers sources du projet

Quand le projet est ouvert, tous les fichiers sources du projet seront chargés.

Ne charger que les fichiers sources marqués dans 'Fichiers du projet'

Quand le projet est ouvert, seuls les fichiers sources marqués dans 'Fichier du projet' seront chargés. De cette manière, il sera possible d'avoir toujours la même configuration lorsque qu'un projet est ouvert.

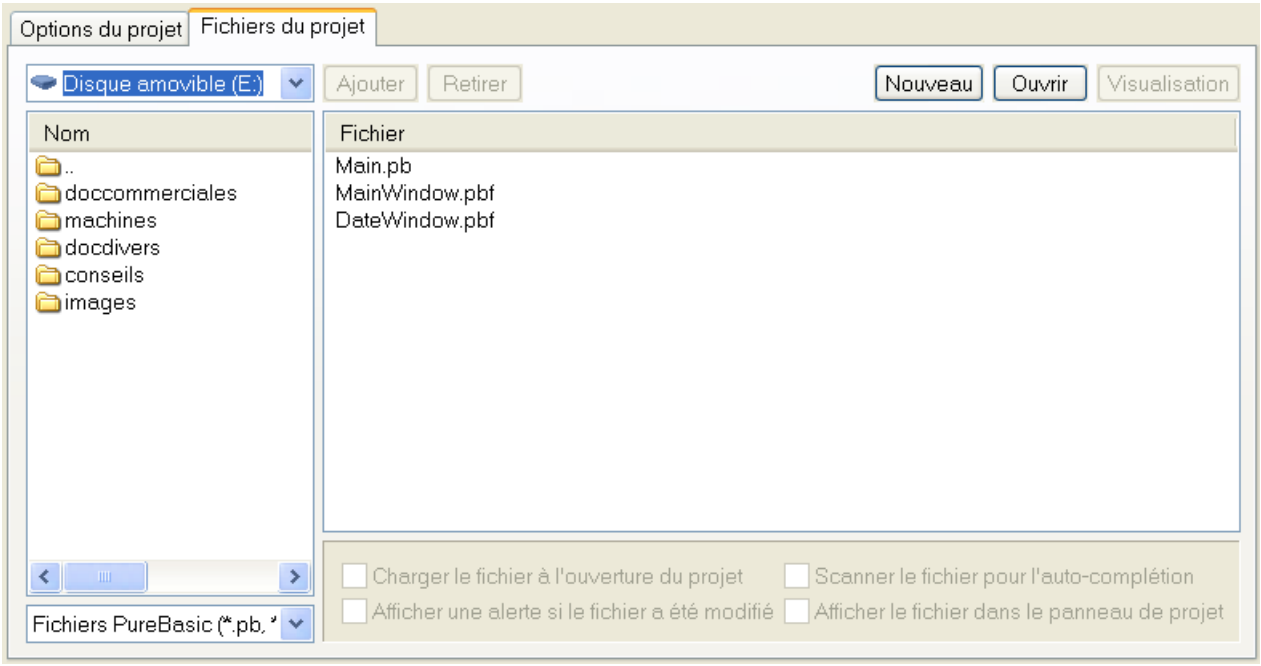
Ne charger que le fichier principal du projet de la cible par défaut

Quand le projet est ouvert, seul le fichier principal de la cible par défaut sera chargé.

Ne charger aucun fichier

Aucun fichier source ne sera chargé lors de l'ouverture du projet.

L'onglet "Fichiers du projet" affiche la liste des fichiers qui compose le projet et permet de changer leurs paramètres. L'explorateur sur la partie gauche permet de rajouter facilement des fichiers au projet.



Les boutons en haut de l'onglet ont les fonctions suivantes :

Ajouter

Ajouter le(s) fichier(s) sélectionné(s) de l'explorateur dans le projet.

Retirer

Retire du projet les fichiers sélectionnés dans la liste.

Nouveau

Ouvre une fenêtre permettant de choisir le nom du nouveau fichier à créer. Ce fichier sera créé, ajouté au projet et ouvert dans l'IDE.

Ouvrir

Ouvre une fenêtre permettant de choisir un fichier existant à ajouter au projet. Le fichier sera automatiquement ouvert dans l'IDE.

Voir

Ouvre le(s) fichier(s) sélectionné(s) de la liste dans l'IDE, ou dans le visualisateur de fichiers si ce sont des fichiers binaires.

Les cases à cocher en dessous de la liste des fichiers représentent les options disponibles pour chaque fichier. Elles peuvent être changées pour un seul ou plusieurs fichiers à la fois. Voici leurs descriptions :

Changer le fichier à l'ouverture du projet
Indique à l'IDE de charger ce fichier lorsque que le projet est ouvert et que l'option "Ne charger que les fichiers sources marqués dans 'Fichiers projet"' est activée dans l'onglet "Options du projet".

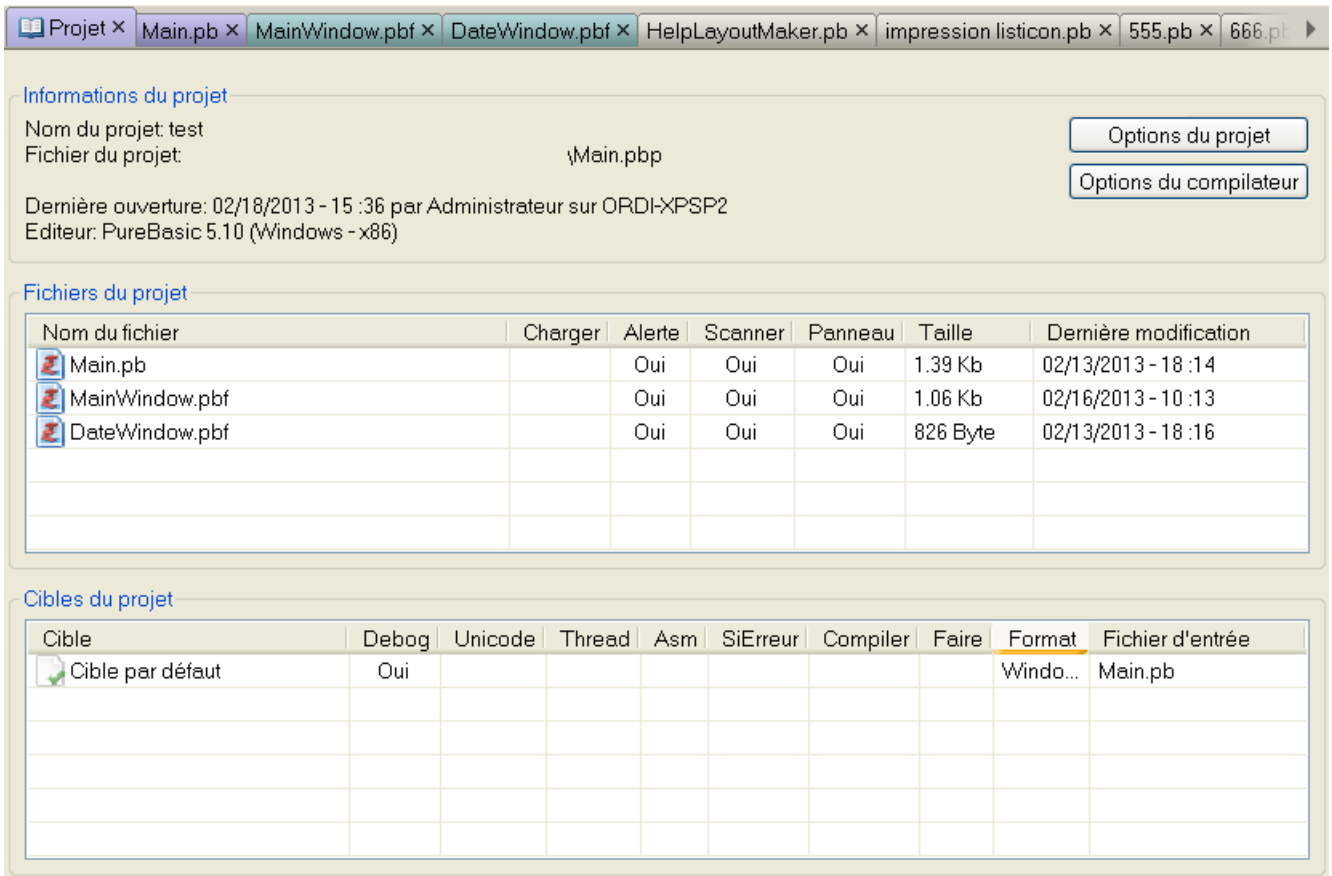
Afficher une alerte si le fichier a été modifié
Quand le projet est fermé, l'IDE calcule un checksum de tous les fichiers qui ont cette option activée, et affichera une alerte si le fichier a été modifié en dehors de l'IDE. Cela permet de savoir qu'une modification externe a eu lieu lorsque l'on partage des fichiers avec différents projets. Cette option ne devrait pas être activée pour les fichiers binaires imposants pour garder une bonne rapidité d'ouverture et de sauvegarde du projet.

Scanner le fichier pour l'auto-complétion
Active le scan du fichier pour la liste d'auto-complétion, même si ce dernier n'est pas chargé dans l'IDE. Cette option est activée par défaut pour tous les fichiers non-binaires. Il faudra le désactiver pour tous les fichiers qui ne contiennent pas de code source, ainsi que pour les fichiers qui n'ont pas leurs places dans la liste d'auto-complétion.

Afficher le fichier dans le panneau de projet
Affichera le fichier dans le panneau de projet, qui se trouve à droite de l'éditeur. Si le projet a beaucoup de fichiers, il peut être utile d'en cacher quelques-un pour garder un accès rapide aux fichiers principaux.

Résumé du projet

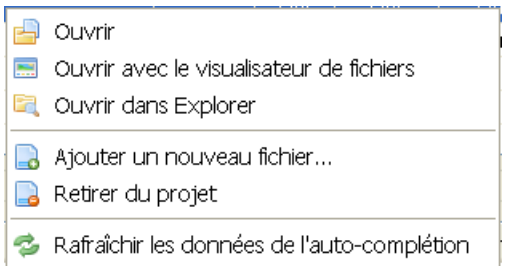
Quand un projet est ouvert, le premier onglet des fichiers affiche un résumé du projet et de ses fichiers.



Informations du projet
Cette section affiche les informations générales du projet, comme le nom du projet, ses commentaires ou quand le fichier a été ouvert pour la dernière fois.

Fichiers du projet

Cette section affiche tous les fichiers du projet et leurs paramètres. Un double-clic sur un de ces fichiers l'ouvre automatiquement dans l'IDE. Un clic-droit fait apparaître un menu contextuel avec davantage d'options :



Ouvrir - Ouvre le fichier dans l'IDE.
Ouvrir avec le visualisateur de fichiers - Ouvre le fichier dans le visualisateur intégré de l'IDE.
Ouvrir avec l'explorateur - Ouvre le fichier dans l'explorateur du système d'exploitation.
Ajouter un nouveau fichier - Ajoute un nouveau fichier au projet.
Retirer du projet - Retire le(s) fichier(s) sélectionné(s) du projet.
Rafraîchir les données de l'auto-complétion - Rescanne tous les fichiers du projet pour actualiser les données de l'auto-complétion.

Cibles du projet
Cette section affiche toutes les cibles du projet et quelques un de leurs paramètres. Un double-clic sur l'une des cibles l'ouvre dans les options de compilation . Un clic-droit affiche un menu contextuel avec davantage d'options :

Editer la cible - Ouvre la cible dans la fenêtre options de compilation.
Définir comme défaut - Défini cette cible comme celle par défaut.
Inclure dans 'Créer toutes les cibles' - Inclut cette cible dans 'Créer toutes les cibles'.

La panneau projet

Un outil intégré permet d'accéder rapidement aux fichiers du projet. Pour plus d'informations voir la section outils intégrés .

Chapitre 12

Compilation d'un programme

La compilation d'un programme est facile, il suffit de sélectionner 'Compiler/Executer' (raccourcis F5 par défaut) et le code sera compilé et exécuté.
Les "Options de compilation" permettent de paramétrer le processus de compilation. Les choix effectués sont associés au fichier ou au projet courant et sont persistants, même quand ils sont fermés. Par défaut ces paramètres sont écrits à la fin du fichier, sous forme de commentaires (invisibles quand le fichier est chargé par l'IDE). Il est possible de changer cet emplacement si, par exemple, un outil de gestion de versions tel que CVS est utilisé (pour éviter les conflits entre les utilisateurs qui auront chacun leurs propres paramètres en fin de fichier).
Si une erreur est rencontrée lors de la compilation, un message sera affiché en indiquant la ligne incriminée. Ce message sera aussi ajouté dans le rapport d'activité.

Le menu "Compilateur"

Compiler/Exécuter
Lance la compilation du code source en cours d'édition en tenant compte de ses options de compilation. Le programme est créé dans un répertoire temporaire, mais il sera exécuté comme si il avait été lancé à partir du répertoire du code source (donc charger un fichier de ce répertoire fonctionnera).
Le code source n'a pas besoin d'être enregistré pour être compilé (mais les fichiers qu'il inclut doivent être enregistrés).
La commande "Compiler/Exécuter" utilise le paramétrage du débogueur (actif ou inactif), qu'il soit défini à partir du menu ou des options de compilations (les deux reviennent au même).
Exécuter
Exécute le dernier programme compilé une nouvelle fois. Tout changement concernant le débogueur (actif ou inactif) ne sera pas pris en compte par cette commande.
Vérification de la syntaxe

Vérifie la syntaxe du code.

Compiler avec le Débogueur

Lance la compilation de la même manière que la commande "Compiler/Exécuter", en forçant l'utilisation du débogueur. Cela permet de tester rapidement une compilation avec le débogueur lorsqu'il est normalement désactivé.

Compiler sans le Débogueur

Lance la compilation de la même manière que la commande "Compiler avec le débogueur", en désactivant le débogueur.

Redémarrer le compilateur (Windows uniquement)

Relance le compilateur et réinitialise toutes les bibliothèques et résidents chargées (sous Windows, le compilateur se lance lors du démarrage de l'IDE et se met en attente pour une plus grande rapidité). Cela a pour effet de mettre à jour la liste des fonctions, structures, interfaces et constantes reconnues par l'IDE. Cette commande permet de charger une nouvelle bibliothèque utilisateur nouvellement installée sans redémarrer l'IDE. De plus, pour les développeurs de bibliothèques, cela facilite le test en évitant de relancer l'IDE.

Options du Compilateur...

Ouvre la fenêtre d'options, qui permet d'ajuster les paramètres concernant la compilation du code source en cours d'édition.

Créer un éxecutable...

Ouvre une boîte de dialogue de sauvegarde, demandant le nom de l'exécutable à créer. Si le format exécutable est réglé sur DLL, il créera une DLL sous Windows, un objet partagé sous Linux et une dylib sous OS X. Lorsque vous créez un fichier exécutable sous OS X, en ajoutant '. App 'au nom de l'exécutable créera un fichier exécutable fourni avec la structure de répertoires nécessaires, y compris l'icône. Si aucun '. App' n'est défini, il va créer un exécutable standard de type console.

Cible par défaut

Quand un projet est ouvert, ce sous-menu permet de changer rapidement de cible par défaut. La cible par défaut est celle qui sera lancée automatiquement par le menu "Compiler/Exécuter.

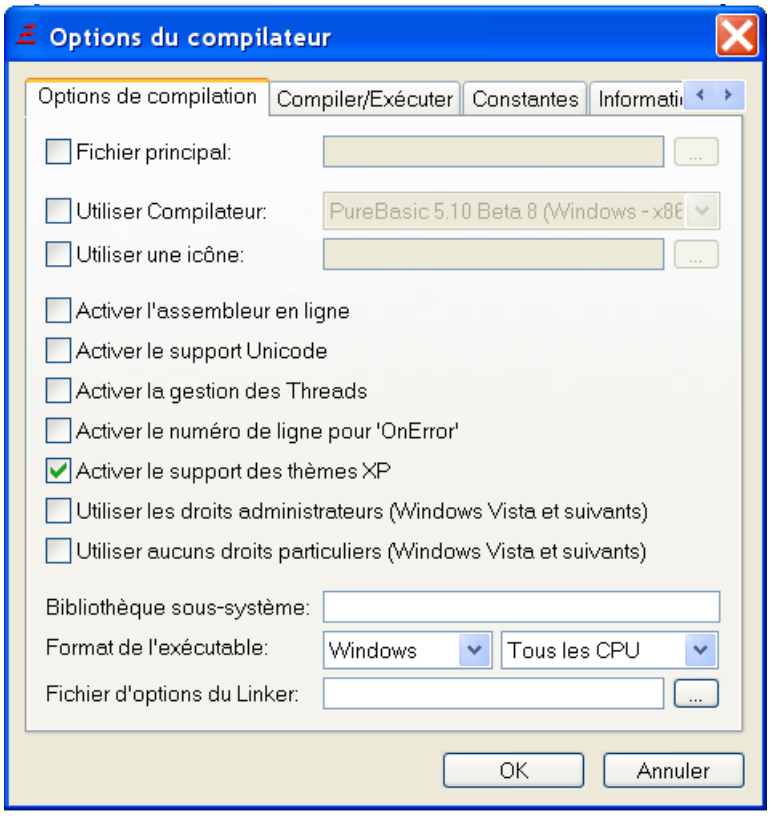
Créer une cible

Quand un projet est ouvert, ce sous-menu montre toutes les cibles disponibles et permet d'en compiler directement une d'entre elles.

Créer toutes les cibles

Quand un projet est ouvert, ce menu compile toutes les cibles qui ont l'option idoine activée dans les options de compilation. Une fenêtre s'ouvre permettant de suivre l'état d'avancement des compilations.

La fenêtre d'options de compilation pour les fichiers hors projets



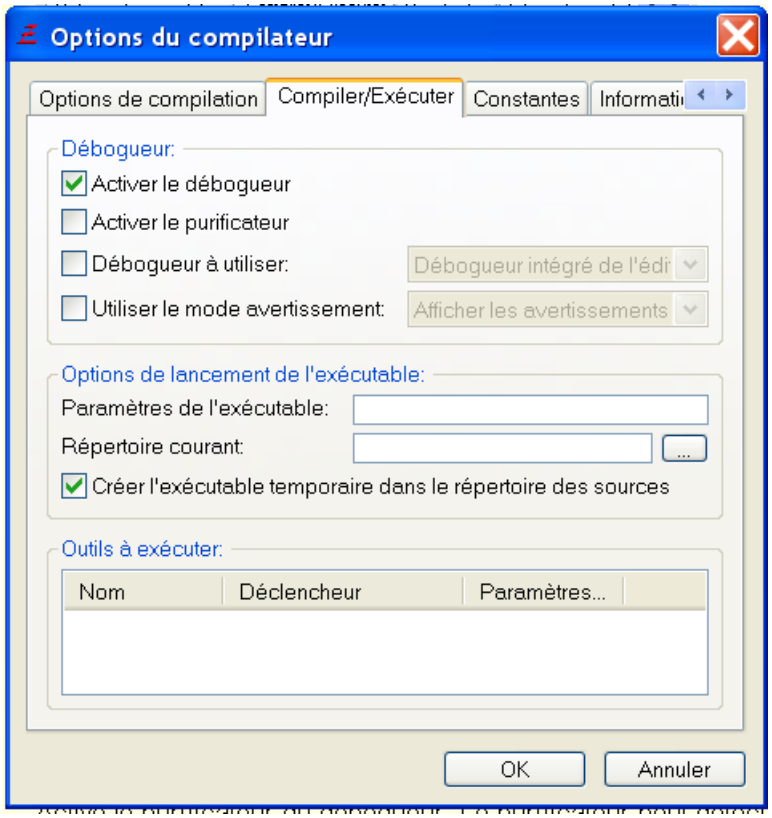
Fichier principal
En activant cette option, il est possible de définir un fichier principal pour la compilation du fichier actuel. Ce sera ce fichier qui sera alors utilisé lors de la compilation et non le fichier en cours d'édition. Note : quand cette option est utilisée, le fichier en cours d'édition doit être enregistré avant de lancer la compilation, sinon les modifications ne seront pas prises en compte (car les fichiers inclus sont chargés à partir du disque). La plupart des options de compilation seront héritées du fichier principal, seuls les paramètres concernant le débogueur et la ligne de commande seront propres au fichier actuel.
Utiliser Compilateur
Cette option permet de sélectionner un compilateur différent de la version actuelle de PureBasic. Cela facilite la compilation d'un programme sous différentes achitectures (x86 ou x64) sans avoir à redémarrer une nouvelle instance de l'IDE. Les compilateurs additionnels sont paramétrables dans les préférences . Si la version du compilateur est identique à celle de l'IDE mais que le processeur géré est différent, le débogueur intégré peut quand même être utilisé pour déboguer cet exécutable. Par exemple, une exécutable compilé avec la version x86 du compilateur peut être débogué avec l'IDE x64 et vice versa. Si la version du compilateur et de l'IDE ne correspondent pas, alors le débogueur indépendant sera utilisé pour éviter des problèmes potentiels.
Utiliser une icône (Uniquement pour Windows et MacOS X)
Il est possible de spécifier une icône qui sera associée à l'exécutable et affichée dans l'Explorateur Windows (ainsi que dans le coin supérieur gauche des fenêtres du programme et dans la barre des tâches).
Windows : L'icône doit être au format ICO (icônes Windows).
MacOS X : Le fichier icône doit être au format ICNS (icônes Macintosh). Pour créer un tel fichier, il faut 4 fichiers PNG de dimensions 128x128, 48x48, 32x32 et 16x16 de l'image qui sera utilisée pour l'icône. Ensuite l'outil "Icon Composer" fourni avec les outils de développement Apple servira à créer le fichier final (il devrait se trouver dans le dossier : /Developer/Applications/Utilities/). Il sera peut-être nécessaire de redémarrer l'explorateur (Finder) pour l'afficher.
Activer l'optimiseur de code
Active l'optimisation du code C.
Activer la colorisation des mots clés assembleur
Active la colorisation des mots clés de l'assembleur. Voir la section Assembleur en ligne x86 pour plus d'informations.

Activer la gestion des Thread
Informe le compilateur qu'il doit utiliser les routines multi-threadées lors de la création de l'exécutable (voir la bibliothèque Thread pour plus d'information).
Ce mode permet aussi au débogueur d'afficher correctement les informations si des threads sont utilisés. Sans cette option, le débogueur peut afficher un numéro de ligne erroné si une erreur intervient à l'intérieur d'un thread.
Activer le numéro de ligne pour OnError (Uniquement pour Windows)
Ajoute des informations dans l'exécutable pour pouvoir identifier la ligne qui a provoqué une erreur (à utiliser en combinaison avec les commandes de la bibliothèque OnError). Cette option influe légèrement sur les performances du programme.
Activer le support des thèmes (Uniquement pour Windows)
Ajoute un fichier permettant la gestion des thèmes Windows (fenêtres et gadgets skinées), lorsque le programme est exécuté sur Windows XP, Windows Vista, Windows 7, Windows 8 (fortement recommandé) ou Windows 10.
Utiliser les droits administrateurs (Windows Vista et suivants) (Windows seulement)
L'exécutable créé sera toujours lancé avec les droits d'administration (i.e vous devez être logué en tant qu'administrateur système) sous Windows Vista et suivants. (il ne s'exécutera pas si le mot de passe de l'administrateur n'est pas donné). Cette option sera définie pour des programmes qui ont besoin d'un accès à des dossiers ou à des zones de registres à accès restreint afin d'obtenir un accès complet.
Si cette option est activée, le débogueur indépendant sera automatiquement sélectionné pendant la phase de débogage, ainsi le programme pourra être testé sous le mode administrateur.
Note : cette option n'a aucun effet quand le programme est lancé sous d'autres versions de windows.
Utiliser aucun droits particuliers (Windows Vista et suivants) (Windows seulement)
Cette option désactive la «virtualisation» de cet exécutable sur Windows Vista et suivants. La Virtualisation provoque la redirection des fichiers et les accès au Registre vers un dossier utilisateur si l'utilisateur n'a pas les droits nécessaires à l'opération. (ce qui permet de conserver la compatibilité avec d'anciens programmes)
Notez que cette redirection se fait sans en aviser l'utilisateur ce qui peut conduire à une certaine confusion si il essaye de trouver les fichiers sauvegardés sur le système de fichiers. Pour cette raison, il est recommandé de désactiver cette fonctionnalité si le logiciel est conforme avec les règles d'accès aux fichiers ou du registre de Vista.
Note : cette option n'a aucun effet quand le programme est lancé sous d'autres versions de windows. Elle ne peut pas être combinée avec l'option "Mode Administrateur" ci-dessus.
Activer le facteur d'échelle d'affichage DPI (Windows)
Cette option active la détection DPI lors de la création d'un exécutable. Cela signifie que l'interface graphique créée dans PureBasic sera automatiquement redimensionnée si le DPI de l'écran est supérieur à 100%. Ce processus se fait généralement sans problèmes, mais certains cas doivent être résolus de façon spécifique, tels que les gadgets qui manipulent des pixels (ImageGadget, CanvasGadget, etc.).
bibliothèque Sous-système
Il est possible d'utiliser différents sous-systèmes lors de la compilation. Plus d'un sous-système peut être spécifié, séparés par une virgule. Pour plus d'informations, consultez sous-systèmes .
Format de l'exécutable
Permet de déterminer le type de l'exécutable qui sera généré :
Windows : format par défaut sous Windows, convient pour tout type d'applications.
Console : un exécutable avec une console par défaut. Il est toujours possible de créer des fenêtres, boutons etc. avec ce type d'exécutable, mais une console sera toujours ouverte quand le programme est exécuté. Si le programme est exécuté à partir d'une ligne de commande, il utilisera ce terminal pour les sorties textes, là où un programme du type "Windows" nécessite la création d'une nouvelle console (à l'aide d'OpenConsole()). Si un programme peut avoir son entrée ou sa sortie redirigée (au travers de 'pipes'), cette option doit être utilisée.
DLL Partagé : crée une DLL Windows. Voir "Créer une DLL" pour plus d'informations.
Note : Quand un code source de type DLL est lancé avec "Compiler/Exécuter", il sera traité comme un programme classique, pour permettre de tester facilement les fonctions de la DLL. La DLL sera réellement créée en appellant la commande "Créer un exécutable".
Optimisations CPU (à coté de "Format de l'exécutable")
Cette option permet d'inclure des fonctions spécialement optimisées pour un type de processeur, lorsqu'elles sont disponibles.
Tout CPU : Les fonctions génériques qui fonctionnent sur tous les processeurs seront utilisées.
CPU Dynamique : Les fonctions génériques ainsi que les fonctions spécifiques à tous les CPU supportés

seront intégrées à l'exécutable. Le choix des fonctions à utiliser se fera au démarrage de l'exécutable, en détectant dynamiquement le processeur sur lequel il est lancé. Cela crée des exécutables plus gros, mais qui seront aussi rapides que possible.
Les autres options : Les fonctions spécifiques à un CPU seront utilisées (ainsi que les fonctions génériques pour celle qui n'ont pas de version spécifique). L'exécutable ne fonctionnera que sur le processeur choisi. Note : Pour le moment, aucune fonction de PureBasic n'intègre de version spécifique, mais quelques bibliothèques utilisateurs en proposent.
Fichier d'options du Linker
Un fichier texte peut être spécifié ici avec de nouvelles options de ligne de commande à transmettre à l'éditeur de liens lors de la création de l'exécutable. Le fichier doit contenir une option par ligne.

Compiler/Exécuter

Cette section contient des options qui affectent la façon dont l'exécutable est exécuté à partir de l'IDE pendant les tests. Sauf pour l'option 'outils', ils n'ont aucun effet quand le menu "Créer un exécutable" est utilisé.

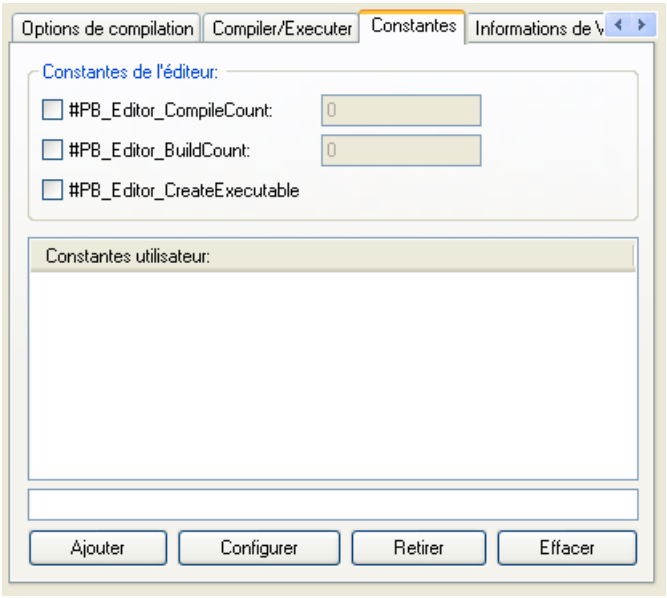


Activer le Débogueur
Définit l'état du débogueur (on/off) pour le code source en cours, ou si l'option du fichier principal est utilisé, pour ce fichier aussi. On peut utiliser directement le menu du débogueur
Activer le Purifier
Active le purificateur du débogueur. Le purificateur peut détecter certains types d'erreurs de programmation comme écrire au delà des capacités d'un tampon en mémoire. Voir outils de débogage interne pour plus de détails.
Débogueur à utiliser
Permet de choisir un type de débogueur différent pour le fichier en cours uniquement. Si cette option est désactivée, le débogueur par défaut est utilisé, c'est celui spécifié dans les préférences .
Utiliser le mode avertissement
Permet d'ignorer, d'afficher ou de traiter les avertissements.
Paramètre de l'éxecutable
La chaîne donnée ici sera passée en tant que ligne de commande pour le programme lorsqu'il est exécuté à partir de l'IDE.

Répertoire courant
Le répertoire spécifié ici sera défini comme le répertoire courant pour le programme lorsqu'il est exécuté à partir de l'IDE.
Créer l'exécutable temporaire dans le répertoire source
Avec cette option activée, le fichier exécutable temporaire sera placé à l'intérieur du répertoire source. Cela peut être utile si le programme dépend de fichiers à l'intérieur du répertoire source. Avec cette option désactivée, l'exécutable est créé dans le répertoire temporaire de système.
Outils à exécuter
Des outils externes peuvent être activés. La colonne "Paramètres globaux" indique si l'outil est activé ou désactivé dans la configuration des outils . Un outil ne sera exécuté pour la source que s'il est à la fois activé globalement et pour la source en cours.
Remarque : Pour qu'un outil puisse être énuméré ici, il doit avoir l'option "Activer l'outil sur une base per-source" dans la configuration des outils et être exécuté par un déclencheur qui est associé à un fichier source. (c.-à-pas par le menu de démarrage ou de l'éditeur par exemple)

Constantes

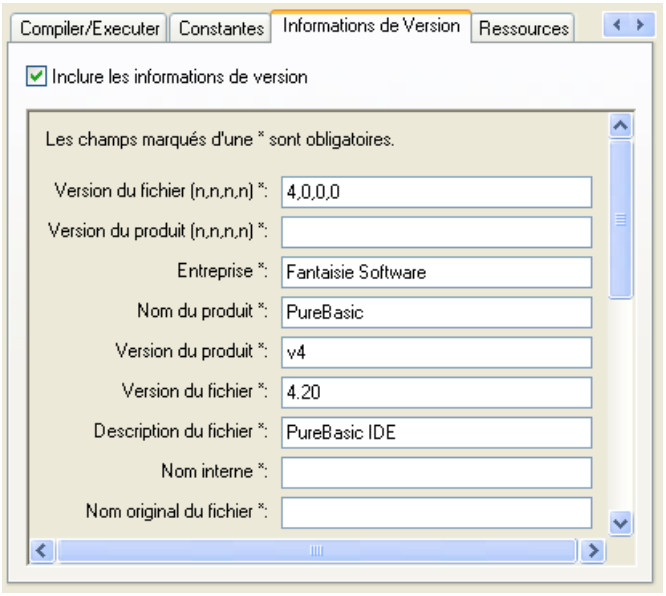
Dans cette section, un ensemble de constantes spéciales pour l'éditeur aussi bien que des constantes personnalisées peuvent être définies pour ce qui sera prédéfini au moment de la compilation du code source.



#PB_Editor_CompileCount
Si cette constante est activée, elle contiendra le nombre de fois que le code a été compilé en choisissant aussi bien "Compiler/Exécuter" que "Créer un exécutable" dans le menu. Le compteur peut être édité manuellement dans le champ de saisie.
#PB_Editor_BuildCount
Si cette constante est activée, elle contiendra le nombre de fois que le code a été compilé en choisissant "Créer un exécutable" dans le menu. Le compteur peut être édité manuellement dans le champ de saisie.
#PB_Editor_CreateExecutable
Si cette constante est activée, elle aura la valeur 1 si le code est compilé en choisissant "Créer un exécutable" dans le menu ou 0 si "Compiler/Exécuter" est utilisé.
Constantes utilisateurs
Les constantes personnalisées peuvent être définies et facilement activé / désactivé avec les cases à cocher. Les constantes devraient être ajoutées à mesure qu'elles apparaissent dans le code source. Cela fournit un moyen pour activer / désactiver certaines fonctionnalités dans un programme et on peut activer / désactiver ces fonctionnalités avec CompilerIf/CompilerEndIf .
Dans la définition de ces constantes, les variables d'environnement peuvent être utilisées dans un style "bash" avec un "$" devant. La variable d'environnement sera remplacée par la définition avant de compiler le source.

Exemple : #Creator=$USERNAME
Ici,le $USERNAME sera remplacé par le nom de l'utilisateur connecté sur les systèmes Windows. Si une variable d'environnement n'existe pas, elle sera remplacée par une chaîne vide.
Note : Pour tester dans le code source si une constante est définie ou non, la fonction Defined() dans fonction du compilateur peut être utilisée.

Informations de version

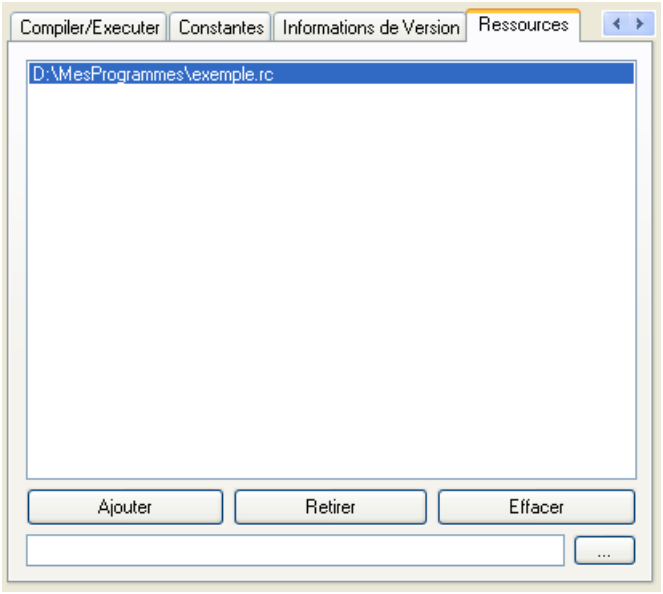


Pour Windows seulement : Si cette fonction est activée, une 'ressource' sera ajoutée à l'exécutable final contenant les informations de version du programme. Il est possible de les consulter en utilisant l'explorateur Windows et de sélectionner 'Propriétés' dans le menu contextuel. D'autres outils peuvent utiliser ces informations, comme par exemple les installateurs de programmes.
Les champs marqués avec une '*' sont indispensables, sinon les informations ne seront pas affichées correctement sur toutes les versions de Windows.
Les deux premiers champs doivent contenir 4 nombres séparés par des virgules. Tous les autres champs peuvent contenir n'importe quel texte. Dans les 3 boîtes vides, il est possible de définir ses propres champs à inclure dans les informations de version.
Dans tous les champs texte, il est possible d'intégrer des tags spéciaux qui seront remplacés par leur valeur associée lors de la compilation :
%OS : remplacé par la version de Windows utilisée pour compiler le programme.
%SOURCE : remplacé par le nom du fichier source (sans son chemin).
%EXECUTABLE : remplacé par le nom de l'exécutable créé (fonctionne uniquement quand "Créer un exécutable" est utilisé).
%COMPILECOUNT : remplacé par la valeur de la constante #PB_Editor_CompileCount.
%BUILDCOUNT : remplacé par la valeur de la constante #PB_Editor_BuildCount.
De plus, il possible d'utiliser n'importe quels tags supportés par la commande FormatDate(). Ils seront alors remplacés par leur valeur par rapport à la date de la compilation (ex : %yy correspondra à l'année de la compilation).
Définition des 3 champs du bas :
OS du fichier
Spécifie l'OS pour lequel ce programme est compilé (utiliser VOS_DOS ou VOS_WINDOWS16 n'a pas vraiment de sens pour un programme PureBasic, ils sont présents uniquement pour avoir une liste exhaustive).
Type du fichier
Définit le type de l'exécutable (ici seuls VFT_UNKNOWN, VFT_APP et VFT_DLL ont un sens pour les programmes PureBasic).
Langue
Spécifie la langue dans laquelle sont décrites ces informations de version.
Les valeurs des champs sont accessibles lors de la compilation du programme de l'IDE en utilisant les

constantes suivantes (même ordre) :

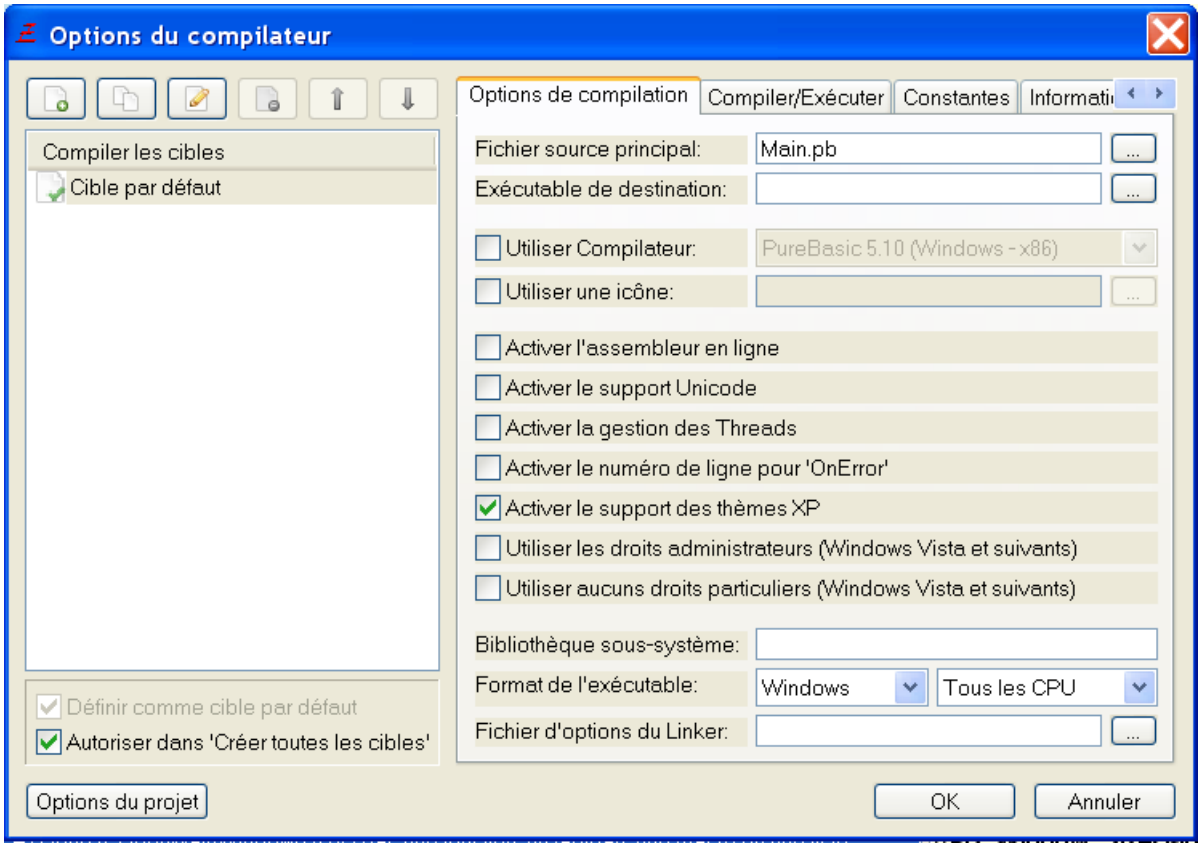
```
#PB_Editor_FileVersionNumeric
#PB_Editor_ProductVersionNumeric
#PB_Editor_CompanyName
#PB_Editor_ProductName
#PB_Editor_ProductVersion
#PB_Editor_FileVersion
#PB_Editor_FileDescription
#PB_Editor_InternalName
#PB_Editor_OriginalFilename
#PB_Editor_LegalCopyright
#PB_Editor_LegalTrademarks
#PB_Editor_PrivateBuild
#PB_Editor_SpecialBuild
#PB_Editor_Email
#PB_Editor_Website
```

Ressources



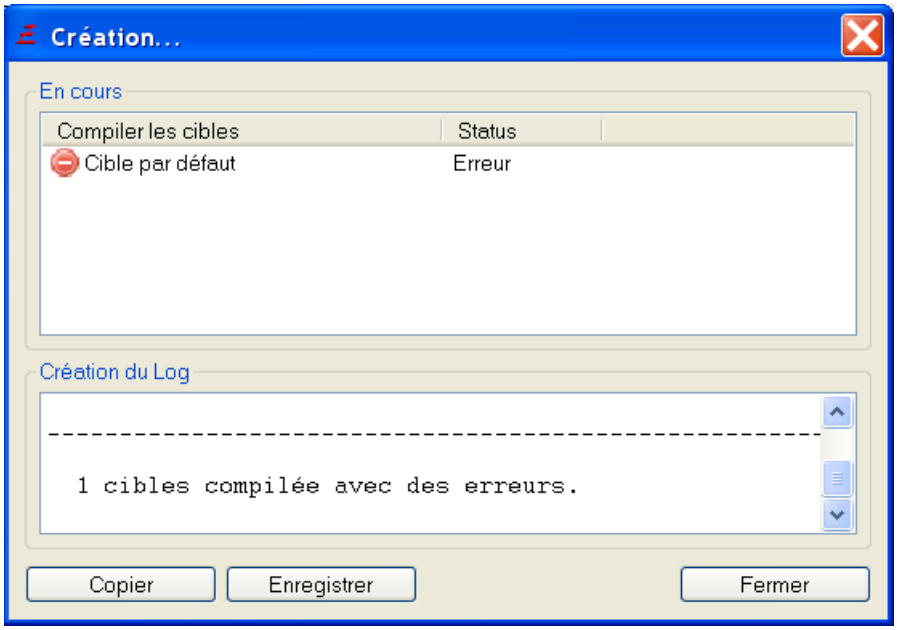
Pour Windows seulement : Permet d'inclure autant de scripts de ressources (fichiers *.rc) que nécessaire. Ils seront alors compilés et ajoutés à l'exécutable, et accessibles dans le programme via les commandes API (étant donné que les ressources sont spécifiques à Windows, PureBasic ne les supporte pas de manière implicite, pour plus de renseignements consultez la MSDN). Pour créer facilement ces scripts, il est conseillé d'utiliser un éditeur de ressources tel que celui présent dans 'PellesC'.

La fenêtre d'options de compilation pour projets



La fenêtre "options de compilation pour projets" permet de définir plusieurs cibles de compilation. Chaque cible est un ensemble de paramètres de compilation avec un fichier source principal et un exécutable. La partie gauche de la fenêtre affiche la liste de toutes les cibles disponibles. La barre d'outils juste au dessus permet de créer, effacer, copier, éditer ou déplacer une cible.
La cible par défaut sera celle qui sera exécutée quand le menu "Compilateur/Exécuter" est sélectionné. Elle peut être rapidement changée par la case à cocher "Définir comme cible par défaut" ou à partir du menu "Compilateur". L'option "Autoriser dans 'Compiler toutes les cibles"' indique si cette cible doit être compilée quand le menu "Compiler toutes les cibles" est utilisé.
La partie droite de la fenêtre est quasiment la même qu'en mode hors projet, et reflète les paramètres de compilation de la cible actuellement selectionnée. Les seules différences sont les champs "Fichier source d'entrée" et "Exécutable de destination" qui doivent être renseignés pour chaque cible, car sans ces informations, il n'est pas possible de compiler la cible automatiquement.
En mode projet, les informations relatives à chaque cible sont enregistrées dans le fichier projet, et pas dans les fichiers sources. Les informations qui sont propres à chaque fichier (comme l'état du pliage) sont toujours enregistrées pour chaque fichier à l'endroit indiqué dans les Preferences .

La fenêtre de compilation des cibles



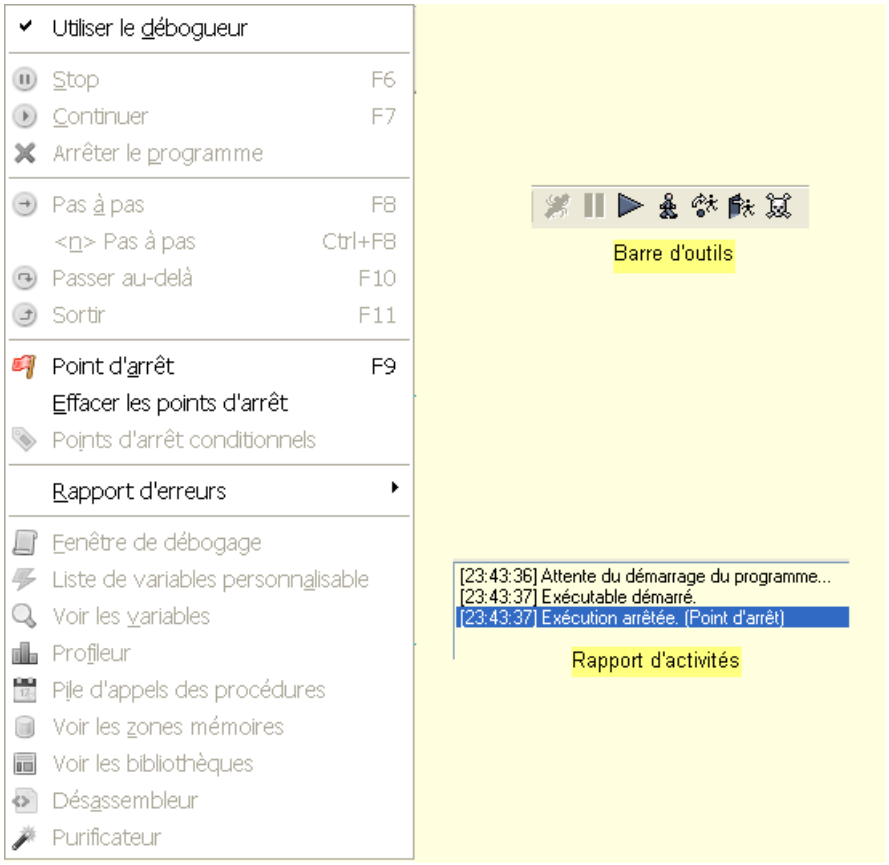
Quand le menu "Créer toutes les cibles" est sélectionné pour le projet en cours, toutes les cibles qui ont l'option idoine activée seront compilées dans l'ordre où elles apparaissent dans la liste des cibles. La fenêtre de progression montre la cible et l'ensemble des cibles ainsi que leurs statuts. Quand le processus de compilation est fini, le log peut être copié dans le presse-papier ou enregistré sur le disque.

Chapitre 13

Utiliser le débogueur

PureBasic est fourni avec un puissant débogueur pour trouver rapidement les erreurs et les bugs des programmes. Il permet de contrôler le flux d'exécution du programme, d'observer le contenu des variables, tableaux et listes, d'afficher des informations de débuggage etc. Il supporte aussi des fonctions avancées tels que l'affichage du contenu des registres (assembleur), voir le contenu de la pile ou d'une zone de mémoire quelconque. De plus il permet de faire du déboguage à distance, via le réseau.
Pour activer le débogueur, il suffit de choisir "Utiliser le débogueur" dans menu "Débogueur" (ou de l'activer dans les "Options de compilation"). En choisissant le menu "Compiler avec le débogueur", le débogueur sera activé seulement pour une compilation.
Le débogueur du PureBasic se présente sous 3 formes :
- Un débogueur intégré directement dans l'IDE, pour une utilisation facile et rapide. C'est ce débogueur qui propose le plus de fonctionnalités.
- Un débogueur indépendant, qui est utile dans plusieurs cas spéciaux (par exemple si le programme est déjà en cours de débuggage et qu'il doit être exécuté une nouvelle fois) ou pour être utilisé par un éditeur de code tierce. Il propose quasiment toutes les fonctionnalités du débogueur intégré, mais parce qu'il est séparé de l'IDE, la rapidité des commandes est légèrement diminuée.
De plus il permet de déboguer un programme à distance, via le réseau.
Un débogueur en ligne de commande uniquement. Le but premier de ce débogueur est de pouvoir tester et développer un programme PureBasic sur un système dépourvu d'environnement graphique (comme un serveur linux), et/ou développer à distance via SSH.
Le type de débogueur à utiliser est sélectionnable dans les préférences.
Bien entendu, quand un programme utilise le débogueur, il fonctionne bien plus lentement. Cela ne devrait pas poser de problèmes majeurs étant donné que la rapidité est toujours acceptable, et que ce mode est utilisé uniquement pendant le développement.
Si quelques parties du programme ont déjà été testées et nécessitent une rapidité maximale même pendant les phases de déboguages, il est possible d'utiliser les directives de compilation DisableDebugger et EnableDebugger autour de ces parties.

Le débogueur intégré



Toutes les commandes relatives au débogueur lorsqu'un programme est en cours d'exécution sont disponibles à partir du menu "Débogueur" (ou de la barre d'outils et des raccourcis claviers).
Tant que le programme est en cours de débuggage, tous les fichiers sources qui sont en rapport avec ce programme sont verrouillés en lecture seule jusqu'à la fin de son exécution. Ceci est fait pour assurer que le code qui sera affiché lors du 'pas à pas' ou lors d'une erreur sera correct (et qu'il n'a pas été édité entre temps sans avoir été recompilé).
A noter qu'un programme ne peut être débogué qu'une seule fois par le débogueur intégré. Si le même programme est de nouveau compilé pour être debuggé, le débogueur indépendant sera utilisé. Par contre il est possible de déboguer plusieurs programmes différents simultanément avec le débogueur intégré.
Astuce
Sous Windows, le menu "Débogueur" est aussi ajouté au menu système de la fenêtre principale de l'IDE (le menu qui apparaît lors d'un click sur l'icône située dans le coin supérieur gauche de la fenêtre). Cela permet aussi d'accéder à ce menu à partir de la barre des tâches, en cliquant avec le bouton de droite de la souris sur l'icône correspondant à l'IDE.

Controle de l'exécution

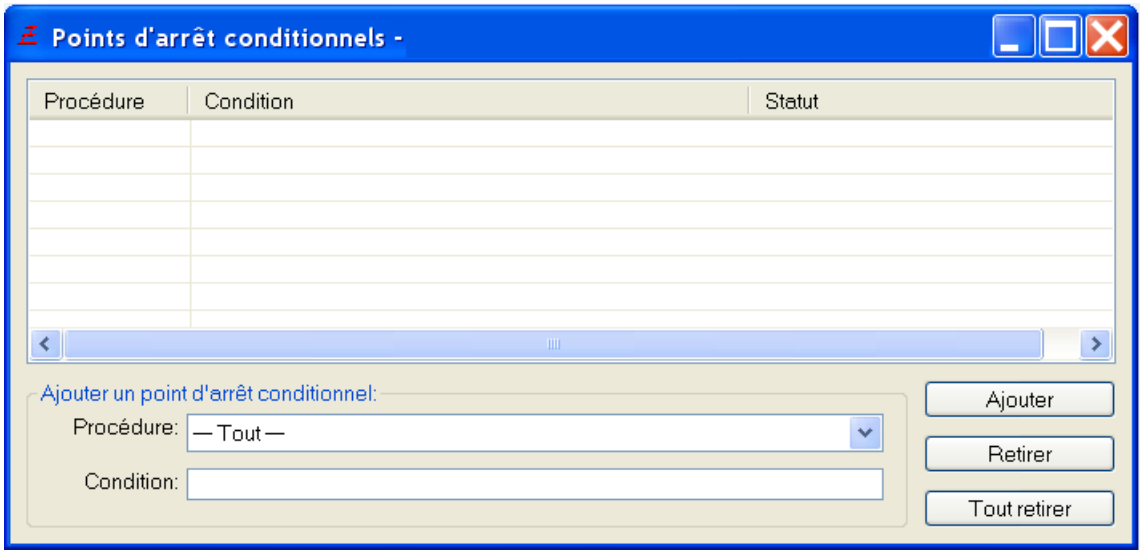
Ces fonctions permettent le contrôle sur le déroulement du programme en cours de débuggage. Le programme peut être stoppé, exécuté pas à pas (ligne par ligne), examiné (voir le contenu des variables à cet instant etc.). Quand le programme est stoppé, la ligne qui va être exécutée est marquée (par défaut en bleu clair) dans le code source correspondant.
L'état du programme est indiqué dans la barre d'état de l'IDE et dans le rapport d'activité.
Commandes du menu permettant le contrôle du programme :
Stop
Stoppe l'exécution du programme et affiche la ligne qui va être exécutée.
Continue
Reprend l'exécution du programme, de manière normale.
Tuer le programme

Force le programme à se terminer et ferme toutes les fenêtres de débuggage associées à ce programme.
Pas
Exécute la ligne du programme actuellement affichée et stoppe de nouveau l'exécution.
Pas <n>
Exécute le nombre de lignes indiqué et stoppe l'exécution du programme.
Passer au-delà
Exécute la ligne du programme actuellement affichée et stoppe de nouveau l'exécution, comme un 'Pas' normal. La différence survient si la ligne contient un ou plusieurs appels à des procédures . Dans ce cas, les procédures seront toutes exécutées, sans arrêt, contrairement au mode 'Pas' qui rentre dans chaque procédure. Ceci permet de passer rapidement sur des procédures qui sont connues comme étant correctes.
Sortir
Exécute le reste du code de la procédure en cours d'exécution et s'arrête à sa sortie. Si la ligne courante n'est pas dans une procédure, un 'Pas' normal sera fait.

Points d'arrêt (ligne)

Les points d'arrêt sont une autre manière de contrôler l'exécution d'un programme. Avec la commande "Point d'arrêt" du menu, la ligne courante sera considérée comme un point d'arrêt (ou retire le point d'arrêt si cette ligne en était un). Quand le programme atteint un point d'arrêt, alors l'exécution est stoppée sur cette ligne (la ligne n'a pas encore été exécutée). A noter que si le point se trouve sur une ligne ou aucun code n'est exécutable (commentaire, ligne vide, structure etc.), le programme s'arrêtera sur la prochaine ligne exécutable rencontrée.
Un fois que l'exécution du programme a été interrompue par un point d'arrêt, il est possible d'utiliser n'importe quelle commande du "Contrôle de l'exécution" pour continuer à déboguer le programme.
Les points d'arrêts peuvent être ajoutés ou retirés de manière dynamique pendant l'édition ou pendant l'exécution du programme. La commande "Effacer les points d'arrêt" permettent d'enlever tous les points d'arrêt du fichier source en cours d'édition.
Note : Pour ajouter/retirer un point d'arrêt à l'aide de la souris, il faut maintenir la touche 'Alt' appuyée pendant le clic sur la bordure qui affiche les points d'arrêts (la colonne de numérotation des lignes n'est pas prise en compte).

Points d'arrêts (conditionnel)



En plus des points d'arrêts classiques, le débogueur permet d'arrêter l'exécution du programme si une condition donnée est remplie. Par exemple, il est possible de mettre une condition sur la valeur d'une variable et d'arrêter le programme quand elle atteint une certaine limite. La condition prend la forme d'une expression PureBasic évaluable en vrai ou faux. Tout ce que l'on peut mettre après un If, y compris les opérateur logiques tels que And, Or ou Not est accepté. La plupart des fonctions des

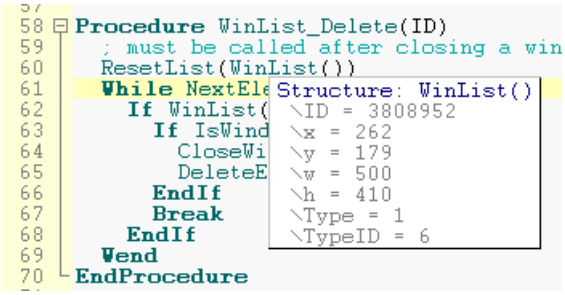
bibliothèques Math , Memory et String ainsi que toutes les fonctions de validation de la forme IsXxx() and the XxxID sont disponibles.
Exemples de conditions :

```
1  MaVariable$ <> "Salut" Or Compteur < 0  ; arrête l'execution si
     'MaVariable$' n'est plus égale à "Salut" ou si 'Compteur' devient
     inférieur à zéro
2  PeekL(*UneAdresse+500) <> 0             ; arrête l'exécution la
     valeur de type long contenu à l'adresse données n'est plus égale à
     zéro
```

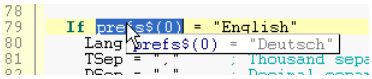
Un point d'arrêt conditionnel peut être ajouté via l'entrée "Point d'arrêt conditionnel" du menu "Debogueur". Il peut se limiter à une procedure particulière, ou il peut être ajouté pour tout le code source. L'entrée "Principal" dans la selection des procédures indique que le point d'arrêt conditionnel devra être évalué seulement en dehors des procédures.
La colonne 'status' affiche le résultat de tous les points d'arrêt conditionnels après leur dernière évaluation. Cela peut être 'vrai', 'faux' ou 'error' si la condition n'est pas une expression valide. Dès qu'une condition est vraie, alors l'exécution du programme s'arrête. Cette condition est automatiquement enlevée de la liste, donc si le programme continue, il ne s'arrêtera pas immédiatement.
Note : La vérification des points d'arrêt conditionnels ralentit l'exécution du programme car les expressions doivent être évaluées à chaque ligne de code. Il vaut mieux les déclarer uniquement quand c'est vraiment nécessaire, pour garder une exécution rapide du programme. Le fait de limiter un point d'arrêt conditionnel à une procédure permet aussi de limiter l'impact sur la rapidité d'exécution, car l'expression n'est évaluée que lorsque la procédure est appelée.

Variables en cours d'exécution

La valeur d'une variable peut être très rapidement vue pendant que le programme est en cours d'exécution en plaçant le curseur de la souris dessus un bref instant dans le code source. Si la variable est actuellement dans la portée et peut être affichée, sa valeur sera affichée dans une info-bulle.



Les expressions plus complexes (par exemple les champs de tableau array) peuvent être consultées en les sélectionnant avec la souris et en plaçant le curseur de la souris sur la sélection.



Des outils du Debogueur offrent également un certain nombre de façons d'examiner le contenu des variables , tableaux ou des listes .

Erreurs dans le programme

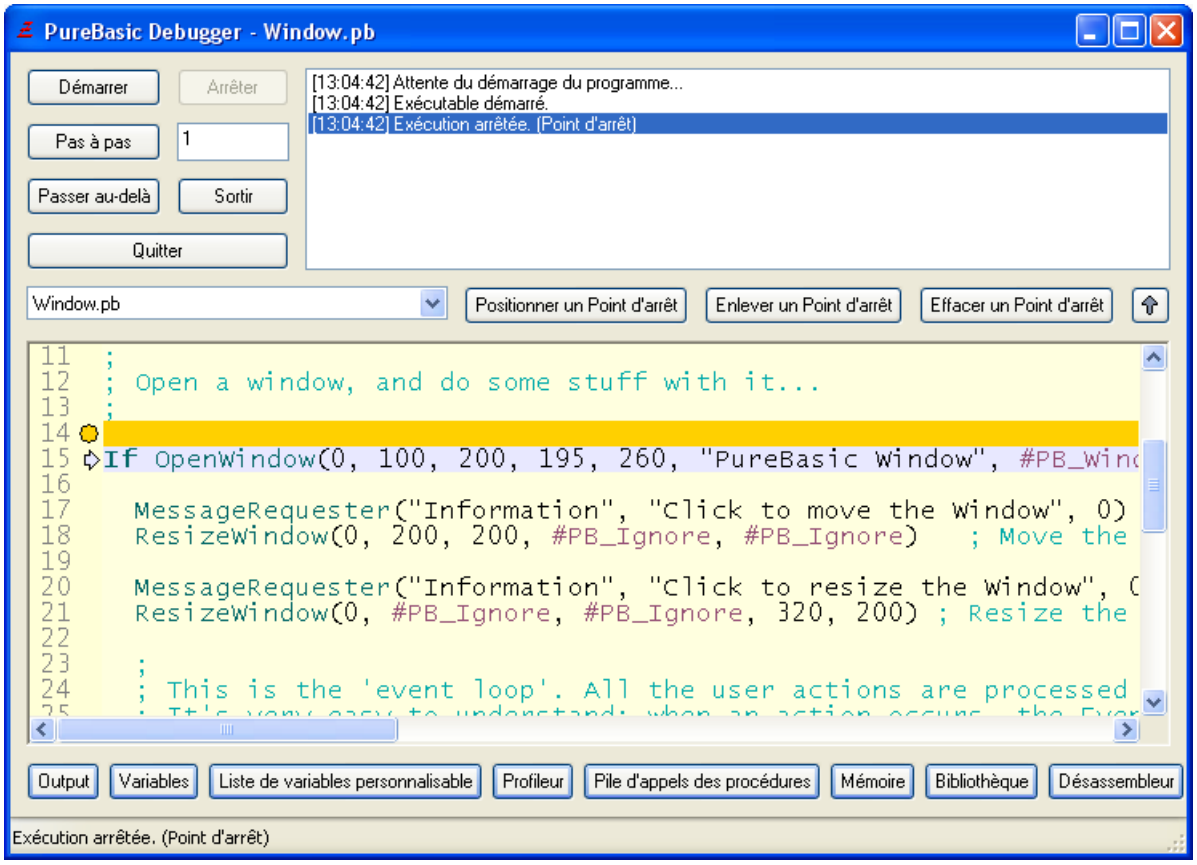
Si le débogueur rencontre une erreur dans le programme en cours de débuggage, il arrêtera l'exécution et marquera la ligne qui contient l'erreur (par défaut en rouge) et affichera le message d'erreur dans la barre d'état et le rapport d'activité.
A ce moment, il est toujours possible de regarder le contenu des variables, de la mémoire et l'historique des appels des procédures, mais les autres fonctionnalités telles que l'affichage des registres ou l'état de

la pile ne sont plus disponibles.
Si l'erreur est considérée comme fatale (comme un accès interdit à une zone mémoire ou une division par 0) il ne sera pas possible de continuer l'exécution du programme. Par contre, si l'erreur est reportée par une commande PureBasic, il est possible de continuer tout de même l'exécution du programme, mais dans ce cas d'autres erreurs anormales peuvent apparaître.
Après une erreur (même fatale), la commande "Tuer le programme" doit etre utilisée pour finir l'exécution du programme et reprendre l'édition du code source. Le programme n'est pas automatiquement tué après une erreur pour permettre au développeur d'utiliser les fonctionnalités du débogueur (comme examiner le contenu des variables) pour essayer de détecter la cause de l'erreur.
Note : il est possible de paramétrer le débogueur pour qu'il termine automatiquement l'exécution du programme en cas d'erreur (voir Paramétrer l'IDE).

Le rapport d'activité

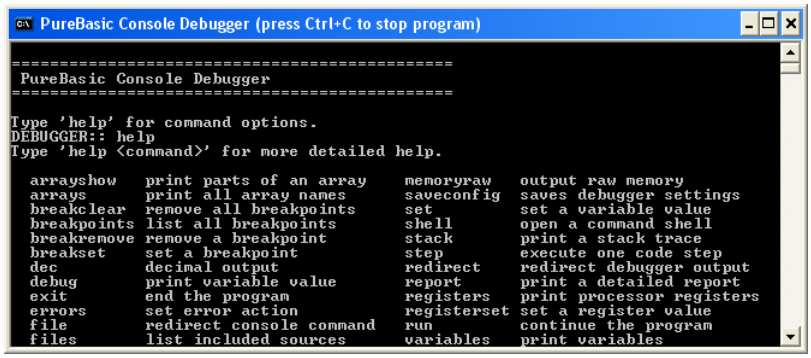
Le rapport d'activité est utilisé pour l'affichage des erreurs de compilation, ainsi que des messages survenant durant le débuggage. Les messages sont toujours affichés dans le rapport d'activité du fichier concerné, donc si une erreur survient dans un fichier inclus, ce dernier sera affiché et un message ajouté dans son rapport d'activité.
Le sous-menu "Rapport d'activité" du menu "Débogueur" contient les commandes pour sa gestion :
Afficher/Cacher
Affiche ou cache le rapport d'activité pour le fichier en cours d'édition.
Effacer
Efface toutes les lignes du rapport d'activité du fichier en cours d'édition.
Copier
Copie le contenu du rapport d'activité dans le presse-papier.
Effacer les erreurs
Une fois que le programme a été tué, les erreurs affichées dans le code sources restent afin de repérer facilement les lignes qui ont causé des problèmes et de les corriger. Cette commande permet d'enlever le surlignage sur les lignes d'erreur. Il est aussi possible de configurer l'IDE pour ôter automatiquement le surlignage des erreurs lorsque le programme se termine (voir Paramétrer l'IDE)

Le débogueur indépendant



Le débogueur indépendant est très similaire à celui intégré à l'IDE et sera seulement expliqué brièvement :
Sur la fenêtre du débogueur sont présents les boutons permettant le contrôle sur l'exécution du programme comme décrit plus haut. Le bouton "Pas" avance d'autant de lignes que défini dans le champ à sa droite. Quand le débogueur est fermé à l'aide du bouton "Quitter" ou en cliquant sur le bouton de fermeture de la fenêtre, le programme en cours de débuggage est immédiatement tué.
La zone de rapport d'activité peut être cachée à l'aide du bouton 'flèche vers le haut' pour rendre la fenêtre de débuggage plus petite.
La zone d'affichage du code source est utilisée pour montrer la ligne en cours d'exécution ainsi que les erreurs ou les points d'arrêt. La liste déroulante située au dessus permet de sélectionner les différents fichiers composant le programme. Les boutons "Mettre point d'arrêt", "Enlever point d'arrêt" et "Effacer points d'arrêt" permettent de gérer dynamiquement les points d'arrêt dans le fichier source affiché. Dans le code, existe également la fonction de survol (du débogueur intégré) pour visualiser rapidement le contenu d'une variable.
Les outils du débogueur peuvent être affichés à l'aides des boutons situés dans la partie inférieure de la fenêtre. L'utilisation de ces outils est la même qu'avec le débogueur intégré.
Note : Le débogueur indépendant n'a pas de configuration spéciale. Il utilise les paramètres du débogueur et de coloration syntaxique de l'IDE. Donc si un éditeur de code tierce est utilisé pour le développement, il convient de régler ces paramètres à l'aide de l'IDE.
Exécution du débogueur indépendant à partir de la ligne de commande :
Pour exécuter un programme compilé en ligne de commande avec le débogueur activé (option /DEBUGGER (Windows) ou -d (Linux/MacOS X)), le débogueur doit être invoqué comme suit :
pbdebugger <fichier exécutable> <paramètres ligne de commande pour l'exécutable>
Si le programme est exécuté immédiatement après une compilation par ligne de commande, le débogueur indépendant est automatiquement utilisé.

Le débogueur en ligne de commande



Le débogueur en ligne commande ne fait pas partie de l'IDE, il ne sera donc pas expliqué en détail dans cette section.
Lorsque le programme est en cours d'exécution, la combination Ctrl+C dans le terminal permet de le stopper et d'invoquer le débogueur. La commande "help" permet d'avoir un apercu des commandes disponibles et "help <commandname>" affichera une aide sur l'utilisation d'une commande. Sous Windows le débogueur en ligne de commande est utilisé uniquement si l'option /CONSOLE est spécifiée.

Déboguer un programme multi-threadé :

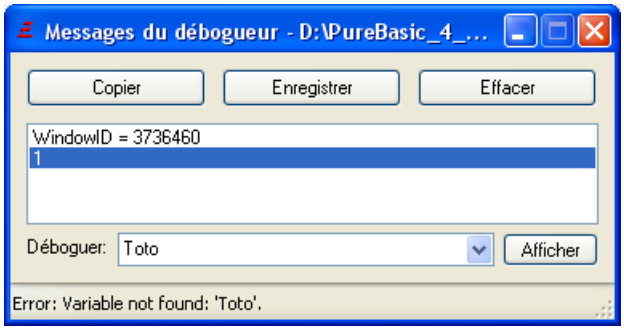
Pour utiliser le débogueur avec un proramme qui utilise des threads , l'option 'Créer un exécutable multi-threadé' doit être activé dans les Options de compilation , sinon les informations affichées par le débogueur concernant les numéros de lignes, les erreurs, les variables locales peuvent être erronées, à cause des threads.
Les limitations suivantes doivent être prises en compte lors du déboguage d'un programme multi-threadé :
Quand le programme est en cours d'exécution, la visionneuse de variables, l'affichage de la pile ou le débogueur assembleur afficheront uniquement les informations du thread principal. Quand un programme est sur arrêt, ils afficheront les informations sur le thread courant. Donc, pour examiner les variables locales d'un thread, l'exécution doit être arrêtée dans ce thread (en utilisant un point d'arrêt ou CallDebugger). Les commandes 'Pas à pas' du débogueur s'appliquent uniquement au thread où l'exécution est arrêtée.
Si une erreur survient, l'exécution est stoppée dans ce thread, donc toute information affichée par la visionneuse de variables ou l'affichage de la pile sera relative à ce thread.
La 'Liste de visualisation' des variables affiche seulement les variables locales du thread principal.
Quand le débogueur s'arrête dans un thread, l'exécution de tous les autres threads est suspendue.

Chapitre 14

Utiliser les outils de débogage

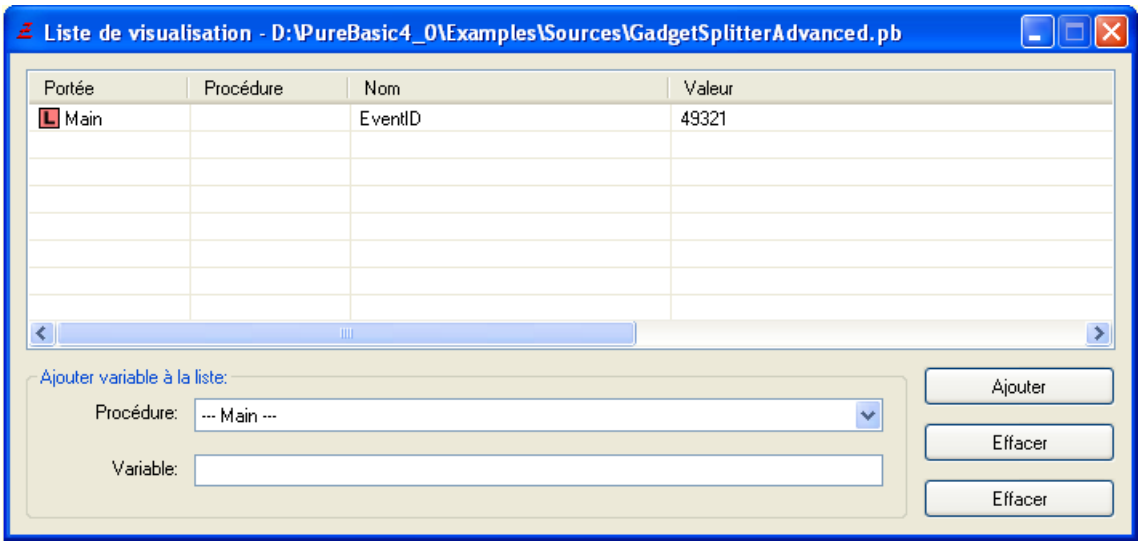
Ces outils proposent de nombreuses fonctionnalités pour inspecter le programme en cours de déboguage. Ils ne peuvent pas être utilisés lorsque que le code source est en train d'être édité. Ils sont tous disponibles à partir du débogueur intégré ou du débogueur indépendant. Le débogueur en ligne de commande propose aussi bon nombre de ces fonctionnalités mais uniquement à l'aide de commandes, rendant leurs utilisations moins facile.
Certains de ces outils permettent d'afficher le contenu des variables. Voici comment est présenté l'affichage pour ces données :
Scope
Le scope d'une variable est la zone dans laquelle elle est valide. Un variable peut être globale , locale , partagée , statique ou threadé , en fonction de sa déclaration dans le code source. Le terme 'byref' ("par référence", c'est à dire en utilisant l'adresse) est utilisé pour indiquer un tableau ou une liste qui a été passé en paramètre d'une procédure.
Type des variables
Le type des variables est indiqué à l'aide d'une icône colorée, pour faciliter leur reconnaissance :
B : Byte
A : Ascii
W : Word
U : Unicode
L : Long
I : Integer
Q : Quad
F : Float
D : Double
S : String (Texte)
Sn : Fixed String (Texte)
Une structure est soit indiquée à l'aide d'un point, ou d'une flèche. Si c'est une flèche, alors le contenu de la structure peut être affiché en double-cliquant sur la ligne. L'icône est alors remplacée par une flèche vers le bas. Une structure représentée par un point ne peut pas être consultée (souvent car il s'agit d'un pointeur vers une structure).
Les tableaux dynamiques à l'intérieur de structures sont représentés avec leurs dimensions. Les listes et les maps à l'intérieur de structures sont représentées avec leur taille et de leur élément courant (s'il existe).

La fenêtre de débug



C'est dans cette fenêtre qu'apparaîtront les résultats de la commande "Debug". Cette commande est un moyen rapide et simple pour afficher des messages destinés au débuggage du programme.
Cette fenêtre sera affichée automatiquement la première fois qu'un message sera émis par le programme. Si elle est fermée, elle ne sera plus affichée automatiquement pour les messages suivants, mais ils seront tout de même enregistrés. Il est possible de copier le contenu de cette fenêtre dans le presse-papier ou de l'enregistrer dans un fichier. Un bouton est aussi présent pour effacer tous les messages précédents.
Le champ en bas de la fenêtre vous permet de saisir une expression qui sera évaluée, et le résultat sera affiché dans la fenêtre du débogueur. Ceci permet de contrôler rapidement la valeur d'une variable ou le champ d'un tableau sans avoir à lancer un des outils de débogage. Pour afficher le résultat de l'expression , appuyez sur la touche [entrée] ou cliquez sur le bouton "Afficher". Si l'expression ne peut pas être évaluée, un message d'erreur est affiché dans la barre d'état.
L'expression peut être n'importe quelle expression PB valide, à l'exception des expressions logiques ou contenant des mots clefs . Elle admet les variables , tableaux , listes , constantes et aussi quelques commandes des bibliothèques Math , Memory et String .

La fenêtre de surveillance



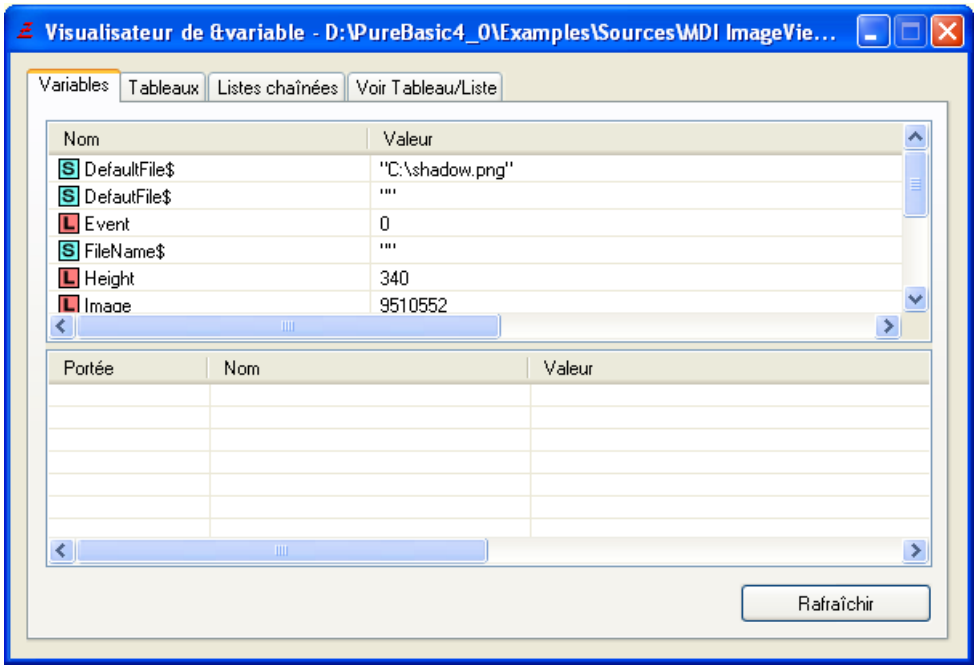
Elle est utilisée pour surveiller en temps réel les valeurs des variables. Il n'est possible d'afficher que des variables de type basique (pas de structures complètes), cependant ces variables peuvent faire partie d'une structure. Les champs de type tableau, liste ou map faisant partie d'une structure ne peuvent pas être affichés dans la liste de surveillance.
Pour ajouter une variable, il faut choisir sa procédure (si c'est une variable locale) ou "— Principal —" si c'est une variable globale, une composante d'un tableau ou d'une liste. Il suffit d'entrer ensuite le nom de la variable dans le champ correspondant et d'appuyer sur "Ajouter".
Exemples :

```
MaVariable$                          - ajoute une variable de type
 string
```

```
Tableau(1, 5)                          ajoute une case d'un tableau
Structure\SousChamp[5]\Valeur          ajoute un champ d'une structure
MaListeChainee()\SousChampStructure    ajoute un champ d'une liste
```

Il est aussi possible d'ajouter une nouvelle variable à la liste de surveillance à partir du "Visualisateur de variables", en cliquant avec le bouton droit de la souris sur une variable et en sélectionnant "Surveiller". Dans la liste, les valeurs des variables surveillées seront affichées. Si la valeur est affichée avec "—", cela veut dire qu'elle n'est pas valide à cet endroit du programme (par exemple une variable locale en dehors de la procédure concernée ou une liste sans élément).
Les variables surveillées sont persistantes entre les sessions de débuggage, et même sauvegardées avec les options de compilation, il n'est donc pas nécessaire de les saisir à chaque fois.

Le visualisateur de variables



Il permet d'examiner les variables, tableaux , listes et maps présents dans le programme. Des les onglets, la partie supérieure montre les éléments globaux and threadés et la partie inférieure les éléments locaux , partagés et statiques .
Le bouton "Actualiser" permet de visualiser les données actuelles du programme en cours d'exécution. Si le programme est stoppé ou en mode pas à pas, les données sont actualisées automatiquement.
Sous Windows, le contenu du visualisateur de variable peut-être trié par nom, scope ou valeur en cliquant sur l'entête de la colonne appropriée.
L'onglet 'Variables'
Cet onglet affiche toutes les variables du programme. En faisant un click-droit sur la variable, il est possible de l'ajouter à la liste de surveillance.
L'onglet 'Tableaux'
Cet onglet affiche tous les tableaux du programme ainsi que leurs dimensions (-1 indique que Dim n'a pas encore été appelé pour ce tableau). En faisant un click-droit sur le tableau, son contenu peut être visualisé dans l'onglet "Voir Tableau/Liste/Map".
L'onglet 'Listes'
Cet onglet affiche toutes les listes du programme, le nombre d'éléments ("-" indique que NewList n'a pas encore été appelé), ainsi que l'index actuel de l'élément courant pour chaque liste ("-" indique qu'il n'y a pas d'élément courant). En faisant un click-droit sur la liste, son contenu peut être visualisé dans l'onglet "Voir Tableau/Liste/Map".
L'onglet 'Maps'
Cet onglet affiche toutes les map du programme, le nombre d'éléments ("-" indique que NewMap n'a pas encore été appelé), ainsi que la clef de l'élément courant pour chaque map ("-" indique qu'il n'y a pas

d'élément courant). En faisant un click-droit sur la map, son contenu peut être visualisé dans l'onglet "Voir Tableau/Liste/Map". **L'onglet 'Voir Tableau/Liste/Map'**
Cet onglet permet d'afficher les éléments d'un tableau, d'une liste ou d'une map, y compris ceux déclarés dans une structure. Pour ce faire, il faut spécifier le nom du tableau, de la liste chaînée ou de la map en incluant les parenthèses "()" à la fin, choisir le type d'éléments à afficher et appuyer sur "Afficher". A noter que les valeurs ne sont pas automatiquement actualisées quand le programme est en mode pas à pas.
"Afficher tout" montre tous les éléments. "Afficher éléments non-nuls" affiche seulement les éléments qui ne sont pas égaux à 0. Cela permet d'afficher des gros tableaux/listes plus facilement si seulement quelques éléments sont utilisés. Une structure est considérée comme "nulle" si tous ses champs sont à 0.
"Afficher partiellement" permet d'afficher seulement une partie du tableau, de la liste ou de la map. Pour les tableaux, l'intervalle peut être spécifié séparément pour chaque dimension en utilisant une virgule. Si une dimension n'est pas du tout spécifiée, tous ses éléments seront affichés.
Voici quelques exemples d'intervalles :

```
"1-2, 2-5, 2" : la première dimension entre 1 et 2, la deuxième entre
 2 et 5, la troisième à 2.
"1, 2-5"      : la première dimension à 1, la deuxième entre 2 et 5
"1, , 5"      : la première dimension à 1, tous les éléments de la
 deuxième dimension, la troisième à 5.
```

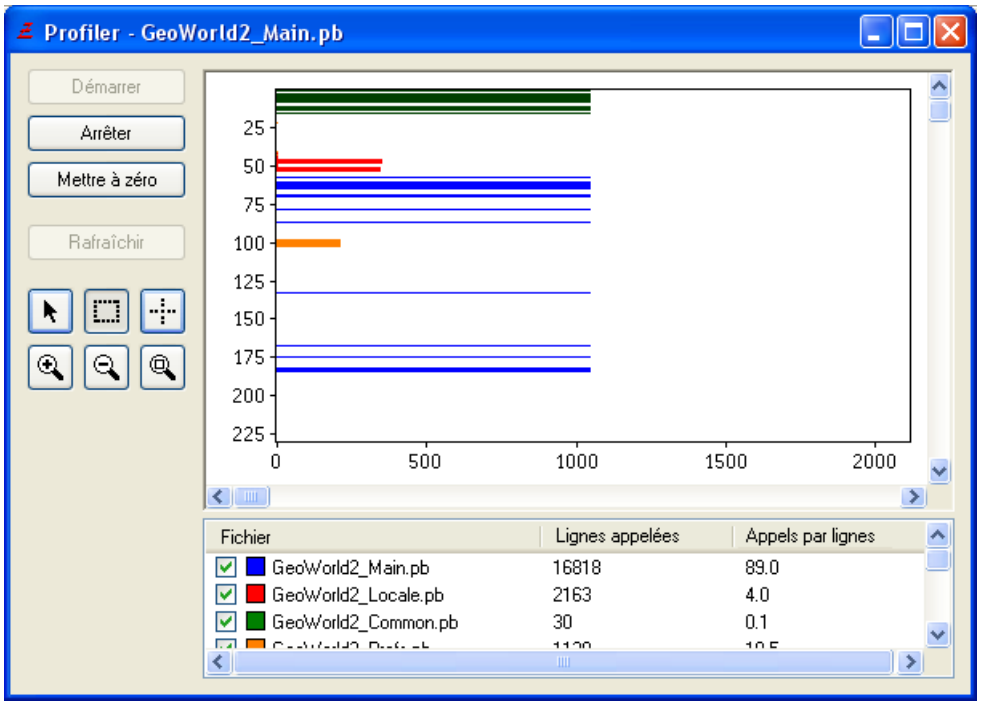
Pour les listes, "Afficher partiellement" accepte une valeur simple ou un intervalle d'éléments (le premier élément a l'index 0).

```
"0"     : le premier élément
"1-3"   : du 2e au 4e élément
```

L'affichage des maps, "Afficher partiellement" est utilisable pour filtrer les clefs des éléments à afficher. Il doit contenir le masque de la clef qui servira de filtre (sans les trémas). Un " ?" représente n'importe quel caractère, un "*" représente n'importe quel nombre de caractères. Voici quelques exemples de masque :

```
"hat" : affichera seulement l'élément avec la clef "hat".
"?at" : affichera tous les éléments de 3 lettres se finissant pas
 "at", comme "bat", "hat" etc.
"h*t" : affichera tous les élémentes qui commencent pas "h" et se
 terminent par "t".
```

Le Profileur



Le profileur est un outil qui permet de compter le nombre d'exécutions de chaque ligne de code. Cela permet d'identifier quelles sont les parties les plus utilisées et donc où les optimisations auront le plus d'effet. Le profileur aide aussi à identifier un problème où une portion du code est exécutée trop souvent à cause d'une erreur.

Enregistrement des données

L'enregistrement est contrôlé par les boutons placés dans la fenêtre du profileur : 'Démarrer', 'Arrêter' et 'Mettre à zéro' (pour tout initialiser). Le bouton 'Rafraîchir' permet de mettre à jour le graphe pendant l'exécution du programme. Les données sont automatiquement mises à jour lorsque le programme est stoppé ou en mode pas à pas. Par défaut le profiler enregistre les données dès le début du programme mais cette option peut-être changée dans les Préférences .

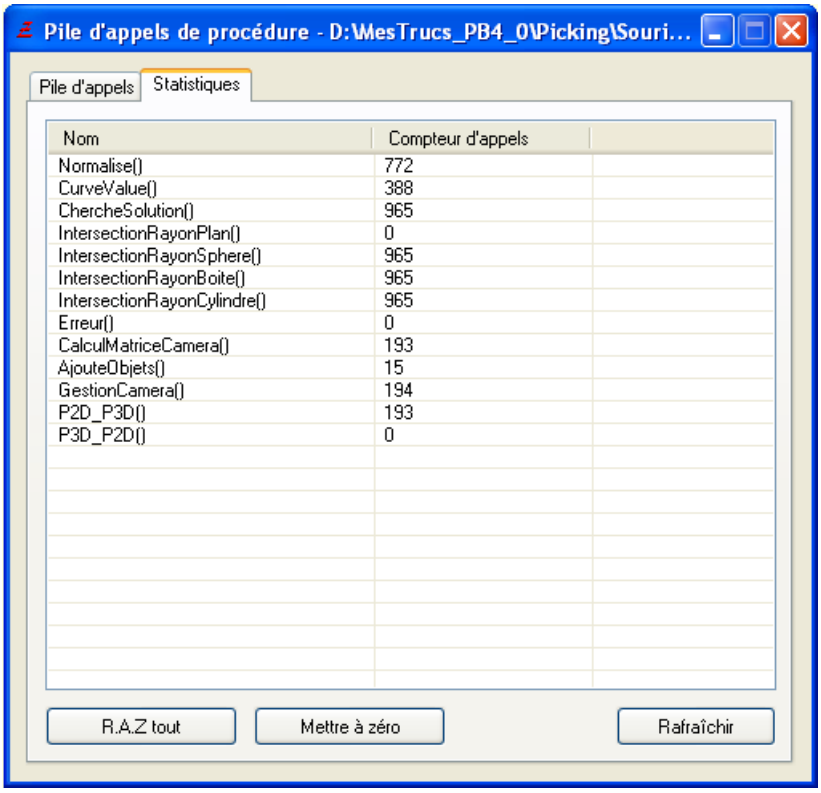
Affichage des données

L'enregistrement est affiché sous forme de graphe, avec comme ordonnées les numéros de lignes et en abscisses le nombre d'exécutions. Si le programme est réparti sur plusieurs fichiers source, la liste des fichiers est présentée en dessous du graphe. Pour afficher un fichier il suffit de le sélectionner ou de cocher la case à cocher, vous pourrez ainsi afficher les résultats de plusieurs fichiers pour mieux les comparer. Un clic droit sur un des fichiers vous permet de changer sa couleur d'affichage dans le graphe.

Utilisation de la souris dans le graphe

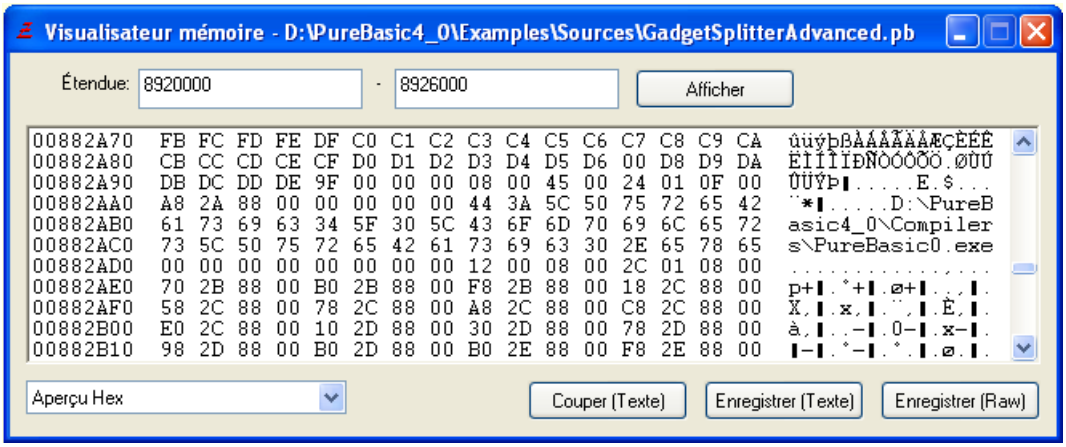
Un clic droit dans le graphe affiche un menu flottant qui autorise à zoomer, montrer la ligne de code sélectionnée dans l'IDE ou le code dans le débogueur. Vous pouvez aussi utiliser les boutons à gauche :
- Flèche : Un clic gauche maintenu permet de faire défiler le graphe.
- Carré : Un clic gauche maintenu permet de sélectionner une zone où faire un zoom.
- Croix : Tant que ce bouton est actif, une croix est affichée et vous aide à lire les numéros de ligne et leur nombre d'appels.
- Zoom : Ces trois boutons permettent de zoomer et d'afficher toutes les lignes.

L'historique des procédures



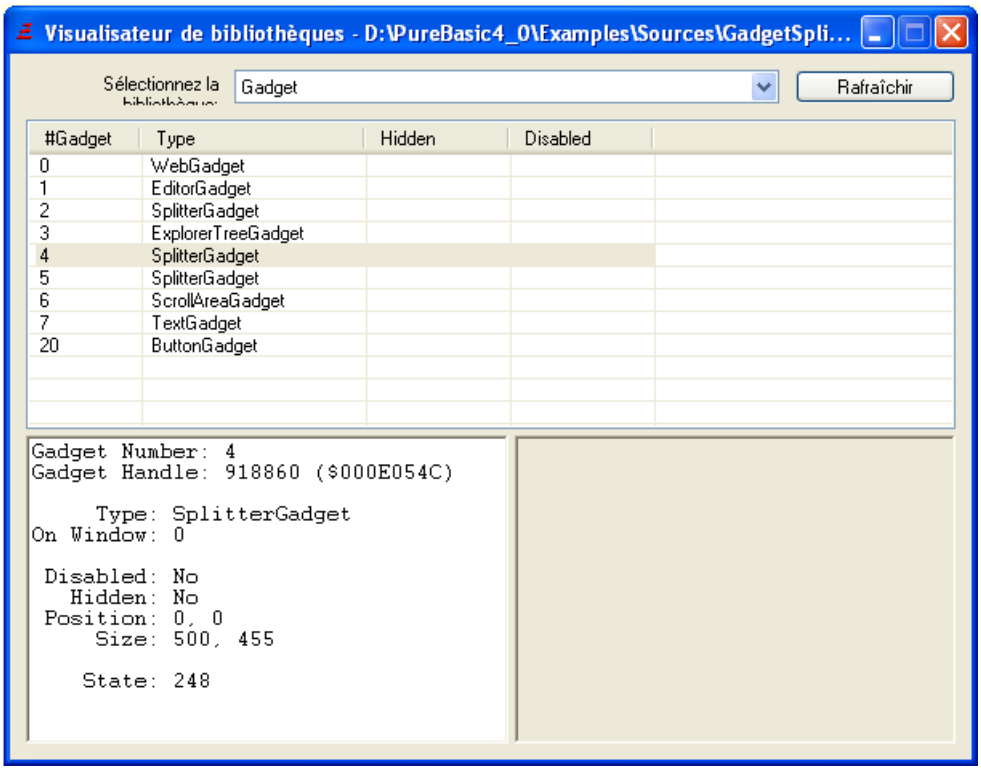
Cet outil montre les procédures qui ont été appelées pour en arriver à la position actuelle du programme. Chaque élément de la liste représente une procédure, la ligne et le fichier où elle est déclarée ainsi que ses arguments qui ont été utilisés lors de son appel. En cliquant sur le bouton "Variables", il est possible de voir les variables de cette procédure.
Cela permet de tracer facilement, à partir de quelle partie du code une procédure a été appelée.
L'historique des procédures ne se met à jour automatiquement que lorsque le programme est arrêté, ou en mode pas à pas. Quand le programme est en cours d'exécution, il est nécessaire d'appuyer sur le bouton "Actualiser" pour mettre à jour la liste.
L'onglet "Statistiques" montre le nombre de fois qu'une procédure a été appelée. Il est possible de réinitialiser un compteur en sélectionnant la procédure puis en appuyant sur "Réinitialiser". De même, en cliquant sur "Réinitialiser tout", tous les compteurs seront remis à 0. Comme pour l'historique des procédures, la mise à jour des compteurs n'est pas automatique quand le programme n'est pas arrêté ou en mode pas à pas, il convient d'utiliser le bouton "Actualiser".

Le visualisateur de mémoire



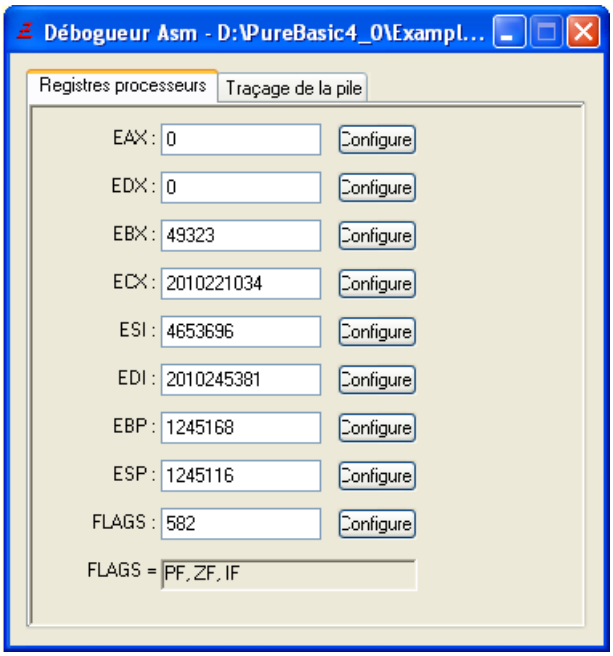
Il permet d'afficher le contenu d'une zone de mémoire arbitraire de votre programme. Une fois les limites inférieures et supérieures renseignées, cliquez sur "Afficher". (vous pouvez utiliser une valeur décimale, un nombre hexadécimal précédé par le caractère '$' ou n'importe quelle autre expression valide, incluant une variable ou un pointeur provenant du code). Si le contenu du second champ commence par le signe '+', alors il est considéré comme relatif au premier champ. Si la zone de mémoire est valide, elle sera affichée. Si la zone complète ou seulement une portion est invalide, alors un message d'erreur sera affiché. Exemple : "*Buffer + 10" to "+30" affichera les 30 octets dans la mémoire en commençant 10 octets après l'adresse pointée par *Buffer.
La façon de présenter le contenu de la mémoire peut être défini grâce à la liste déroulante située en bas à gauche. Les modes suivants sont disponibles :
Aperçu Hex
La mémoire sera affichée en hexadécimal à la manière des visualisateurs hexadécimaux classiques, avec l'adresse de la mémoire à gauche, suivie par le contenu en hexadécimal et la représentation sous forme de caractères dans la colonne de droite.
Tableau d'octets/de caractères/ de mots/ de long/ de quad/ de float/ de double
La mémoire sera montrée sous forme de tableau en fonction du type choisi. Il est possible de configurer ce tableau en mode colonne simple ou multi-colonne dans les préférences (voir Configurer l'IDE) Les données seront affichées en décimal, octal ou hexadécimal via le bouton d'à côté.
Texte ascii, unicode ou utf-8
Affiche la zone mémoire sous forme de texte, avec tous les caractères de contrôles affichés entre [] (par exemple : "[NULL]" pour le caractère 0). Un retour à la ligne est ajouté après les caractères 'nouvelle ligne' ([LF],[CR]) et [NULL] pour améliorer la lisibilité de la sortie. La zone mémoire peut être interprétée comme une chaîne Ascii, Unicode ou Utf8.
Data
La case à cocher **Data**, vous permet d'exporter le tableau en Data Section lors de la copie ou de l'enregistrement.
Il est aussi possible d'exporter la zone de mémoire affichée :
Copier (Texte) : Copie la zone de mémoire affichée dans le presse-papier.
Enregistrer (Texte) : Enregistre la zone de mémoire affichée dans un fichier.
Enregistrer (Raw-Binaire) : Enregistre la zone de mémoire affichée dans un fichier sous forme binaire.

Le visualisateur de bibliothèque



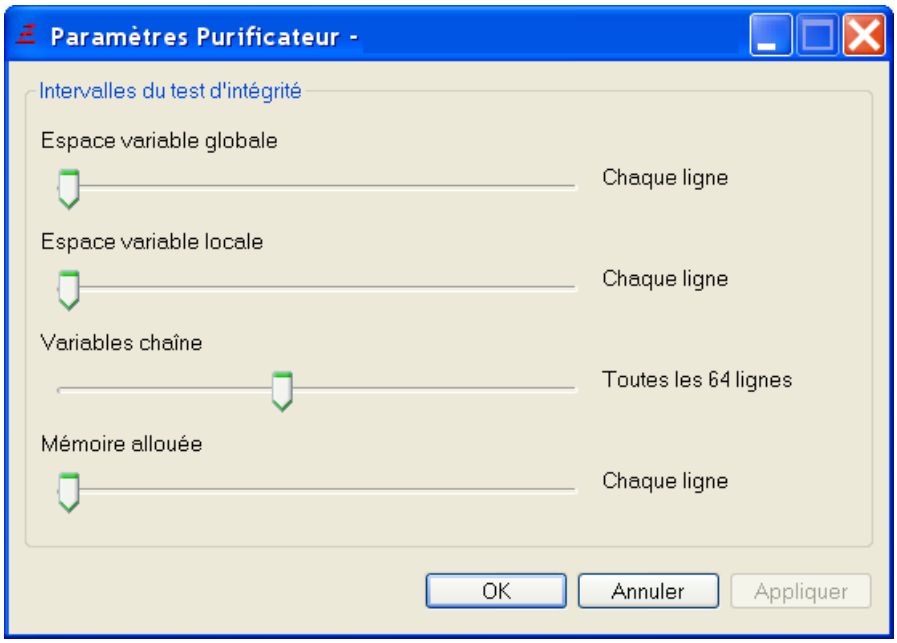
Le visualisateur de bibliothèque donne des informations à propos des objets PureBasic qui ont été créés avec les bibliothèques qui supportent cette fonctionnalité. Par exemple, elle permet de vérifier rapidement les images actuellement chargées dans le programme, ou quels gadgets ont été créés.
Une fois que le programme est démarré, la liste déroulante en haut de la fenêtre peut être utilisée pour sélectionner la bibliothèque à examiner. La liste de tous les objets de cette bibliothèque apparaîtra et en cliquant sur un objet, des informations relatives à cet objet seront affichées. Certaines bibliothèques affichent même un aperçu de l'objet (sprite, image par exemple).
Si la liste déroulante affiche "Aucune information", cela veut dire que l'exécutable n'a utilisé aucune bibliothèque qui utilise cette fonctionnalité.
Pour l'instant, les bibliothèques suivantes sont supportées :
Thread
Gadget
Window
File
Image
Sprite
XML

Le débogueur assembleur



Le débogueur assembleur est utile pour les développeurs expérimentés qui veulent pouvoir examiner le contenu des registres CPU ou de la pile (surtout quand de l'assembleur en ligne est utilisé).
La fenêtre des registres est uniquement disponible quand le programme est stoppé, ou en pas à pas. Il est possible de changer le contenu des registres en modifiant les valeurs des champs puis en cliquant sur "Changer".
L'analyse de la pile affiche le contenu de la pile du programme par rapport au registre ESP. Si la position actuelle de la pile n'est pas alignée sur une valeur multiple de 4, il n'est pas possible de la décoder correctement, donc elle sera affichée sous forme hexadécimale.
Si le pointeur est correctement aligné, le contenu de la pile est affiché avec des commentaires par rapport aux valeurs rencontrées (détails des valeurs passées en paramètres des fonctions PureBasic etc.)
L'affichage du contenu de la pile est actualisé automatiquement quand le programme est stoppé ou en mode pas à pas. Il est possible de désactiver cette mise à jour automatisée dans les préférences de l'IDE, dans ce cas un bouton "Actualiser" sera disponible.
Note : Le débogueur assembleur n'est pas pour l'instant disponible sur MacOS X.

Le Purificateur



Le purificateur permet de détecter des erreurs d'écriture dans des zones de mémoire interdites (dépassement de tampon). La plupart de ces erreurs entrainent des crashs ou des comportements inattendus, souvent très difficiles et laborieux à localiser, car ils sont aléatoires.
Pour fonctionner, le purificateur a besoin de données particulières générées par le compilateur, c'est pourquoi il est nécessaire de cocher la case "Activer le purificateur" dans les options de compilation .
Le purificateur détecte les erreurs d'écriture grâce à des valeurs uniques autour des variables locales et globales, des chaînes de caractères et des blocs de mémoire alloués dynamiquement. Ces valeurs uniques sont vérifiées régulièrement, et si une de ces valeurs a changé, alors il y a eu une écriture interdite et une erreur est affichée. Ces vérifications ralentissent considérablement l'exécution du programme, particulièrement pour les gros programmes, c'est pourquoi il est possible de régler l'intervalle des vérifications dans la fenêtre du purificateur :

Variables globales
Définit l'intervalle (en nombre de lignes exécutées) pour la vérification de l'intégrité des variables globales.

Variables locales
Définit l'intervalle (en nombre de lignes exécutées) pour la vérification de l'intégrité des variables locales.

Chaînes de caractères
Définit l'intervalle (en nombre de lignes exécutées) pour la vérification de l'intégrité des chaînes de caractères.

Blocs de données dynamiques
Définit l'intervalle (en nombre de lignes exécutées) pour la vérification de l'intégrité des blocs de données alloués dynamiquement avec AllocateMemory() .

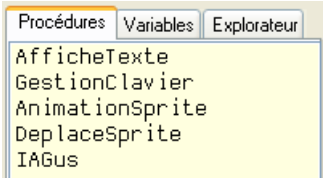
Chapitre 15

Les outils intégrés

L'IDE PureBasic intègre un grand nombre d'outils intégrés, pour rendre la programmation aisée et productive. La plupart d'entre eux peuvent être affichés dans une fenêtre séparée (accessibles alors par le menu) ou dans la palette d'outils située sur le côté de la zone d'édition.
Pour plus d'informations quant à la configuration de ces outils et comment ils sont affichés, voir Configurer l'IDE .

Outils disponibles pour la palette.

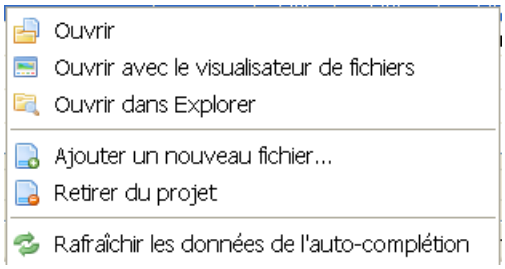
Navigateur de procédures



Cet outil affiche la liste de toutes les procédures déclarées dans le fichier source en cours d'édition. En cliquant sur un élément de cette liste, le curseur changera immédiatement pour aller à la déclaration de cette procédure.
Les macros seront identifiées avec un signe "+" avant le nom.
Il est aussi possible de mettre des commentaires particuliers dans le code qui sera alors aussi affiché dans le navigateur de procédures. Ils ont la forme suivante : ";- <description>". Le ';' démarre le commentaire et le '-' qui le suit immédiatement définit ce type de commentaire. La 'description' sera alors affichée dans la liste et un clic sur cet élément changera la position du curseur pour cette ligne. Ce type de commentaire se distingue dans la liste par l'ajout du caractère '>' devant la description.
La liste des procédures peut être triée, et peut aussi afficher les paramètres de la procédure/macro. Pour ces options, voir Configuration de l'IDE .
Projet

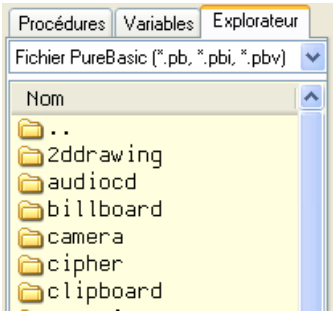
L'outil projet affiche un arbre de tous les fichiers du projet actuellement chargés. Un double-clic sur un

fichier l'ouvrira dans l'IDE. Cela permet un accès rapide à tous les fichiers du projet. Un clic-droit sur un fichier ouvre un menu contextuel qui propose davantage d'options :



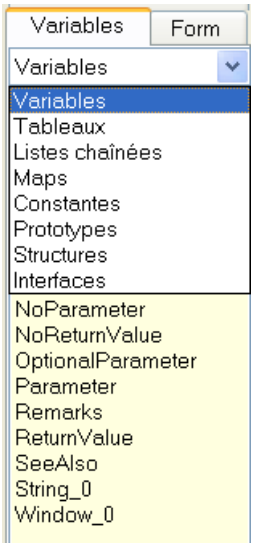
Ouvrir - Ouvre le fichier dans l'IDE.
Ouvrir avec le visualisateur de fichiers - Ouvre le fichier dans le visualisateur intégré de l'IDE.
Ouvrir avec l'explorateur - Ouvre le fichier dans l'explorateur du système d'exploitation.
Ajouter un nouveau fichier - Ajoute un nouveau fichier au projet.
Retirer du projet - Retire le(s) fichier(s) sélectionné(s) du projet.
Rafraîchir les données de l'auto-complétion - Rescanne tous les fichiers du projet pour actualiser les données de l'auto-complétion.

Explorer



L'outil Explorer affiche une liste de fichiers et de répertoires à partir de laquelle il est possible d'ouvrir rapidement n'importe quel type de fichier, en double-cliquant dessus. Les fichiers PureBasic (*.pb, *.pbi, *.pbp, *.pbf) seront chargés directement dans la zone d'édition et les fichiers reconnus par l'éditeur (textes et binaires) seront ouverts par le visualisateur interne de fichiers.

Visualisateur de variables

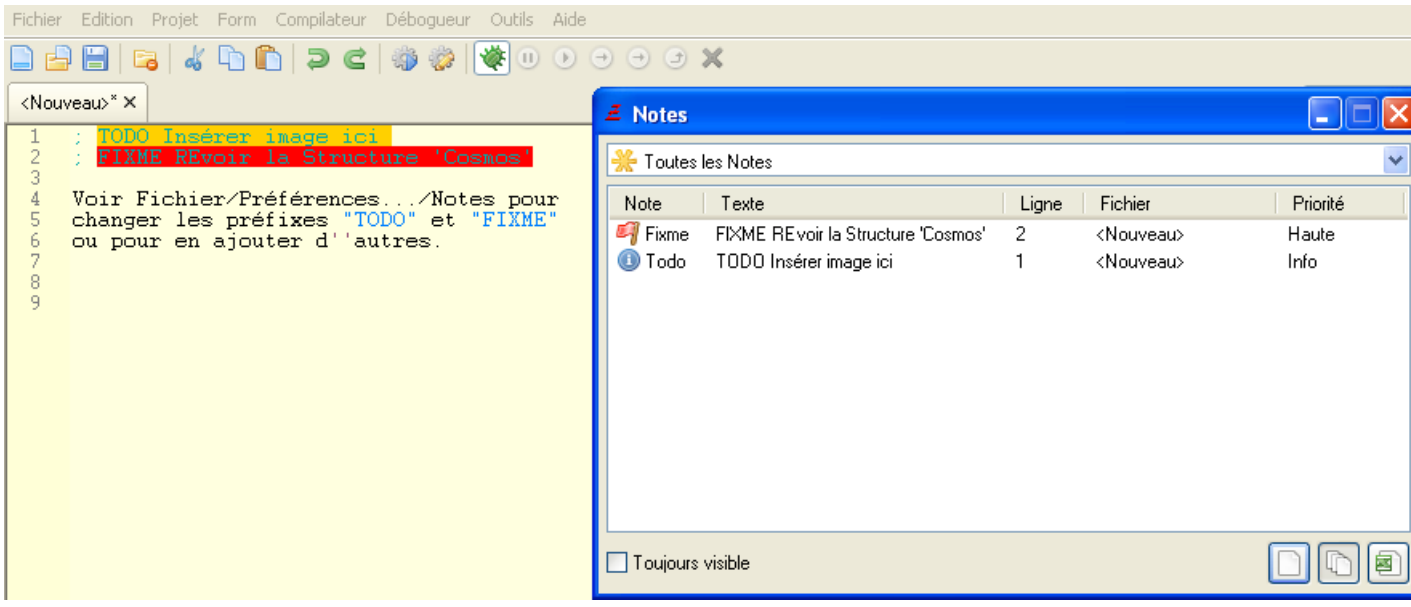


Le visualisateur de variables peut en fait afficher les variables, tableaux, listes, constantes, structures et interfaces définis dans le source en cours d'édition, ou dans tous les fichiers ouverts. La configuration de ce qui doit être affiché se fait dans les préférences .
Note : l'affichage des variables est un peu limité pour l'instant. Seules les variables qui sont déclarées par Define , Global , Shared , Protected ou Static seront reconnues.

Modèles de codes

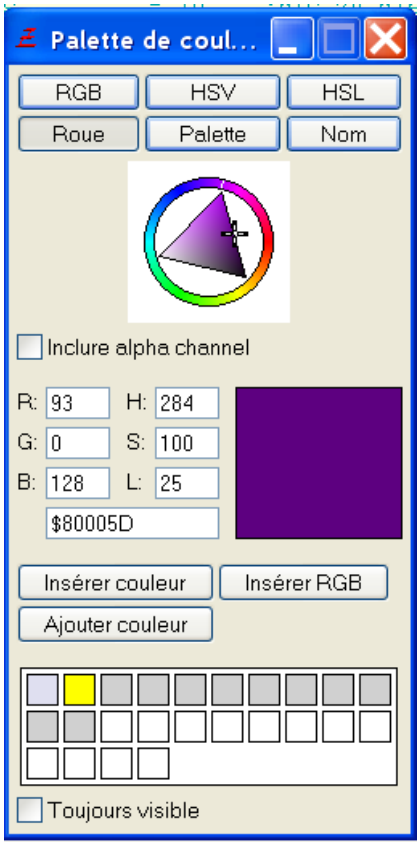
Cet outil permet d'organiser de manière hiérarchisée une liste de petits bouts de code qui sont souvent utilisés. Ils peuvent être insérés rapidement à n'importe quel endroit du fichier en cours d'édition en double-cliquant sur le code voulu.

Explorateur de Notes



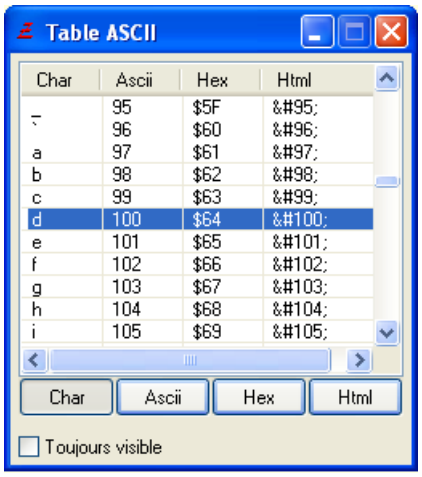
L'outil Explorateur de Notes recueille les commentaires du code source qui correspondent au format défini et les répertorie par ordre de priorité. Il peut être utilisé pour suivre les zones du code source qui doivent encore être travaillées.
Chaque problème affiché correspond à un commentaire dans le code. Un double-clic sur la Note affiche la ligne de code. Les Notes peuvent être affichées pour le fichier en cours, ou pour plusieurs fichiers (tous les fichiers ouverts, ou tous les fichiers qui appartiennent aux projet courant). La liste peut également être exportée au format CSV.
Pour configurer la liste des Notes recueillies, consultez la section "Notes" dans les Préférences .

Choix de couleur



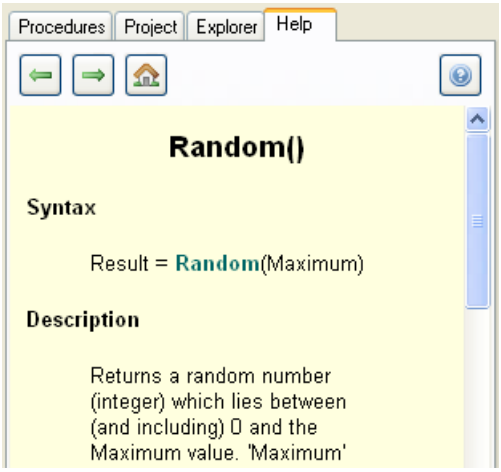
Le sélecteur de couleur vous aide à trouver la valeur de la couleur parfaite pour n'importe quelle tâche. Les méthodes suivantes de choix de couleur sont :
RGB : Sélectionnez une couleur en choisissant les intensités de rouge, de vert et de bleu.
HSV : Sélectionnez une couleur en choisissant la teinte, la saturation et la valeur.
HSL : Sélectionnez une couleur en choisissant la teinte, la saturation et la luminosité.
Roue : Sélectionnez une couleur en utilisant le modèle HSV dans la roue de couleur.
Palette : Sélectionnez une couleur dans une palette prédéfinie.
Nom : Sélectionnez une couleur dans une palette par son nom.
La sélection des couleurs comprend une composante alpha, si l'option "Inclure le canal alpha" est activée. Les composantes individuelles (intensités rouge / vert / bleu ou teinte / saturation / luminosité), ainsi que la représentation hexadécimale de la couleur actuelle peuvent être vues et modifiées dans les champs de texte.
Le bouton "Insérer Couleur" insère la valeur hexadécimale de la couleur courante dans le code source. Le bouton "Insérer RGB" insère la couleur comme le ferait un appel à la fonction RGB() ou RGBA() dans le code. Le bouton "Ajouter couleur" permet d'ajouter la couleur actuelle dans la zone d'historique. En cliquant sur une couleur dans cette zone, permet à cette couleur de devenir la couleur courante.

Table des caractères



La table affiche une liste des 256 premiers caractères unicode avec leurs correspondances en décimal, hexadécimal et HTML. En double-cliquant sur une ligne, ce caractère sera inséré dans le code source. Les boutons en dessous de la liste permettent de choisir le format dans lequel le caractère sera inséré.

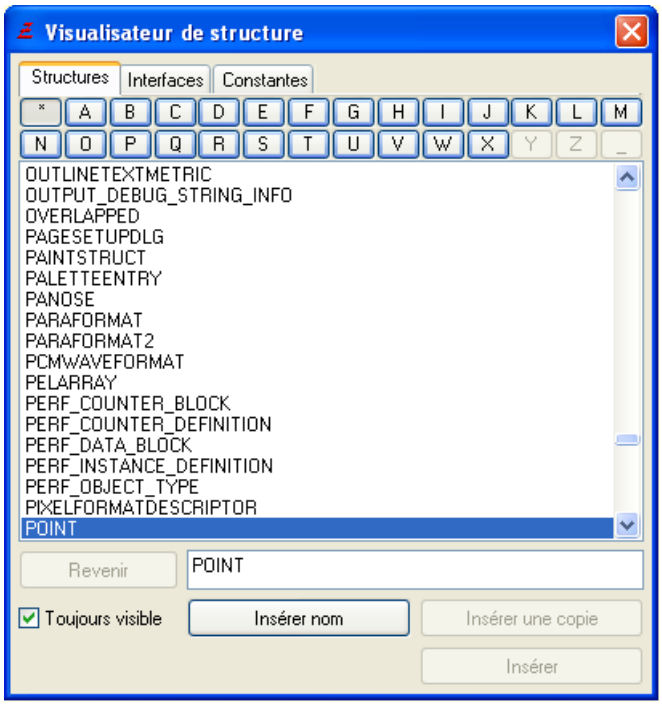
Outil Aide



L'outil d'aide est une visionneuse alternative pour le guide de référence . Il peut être utilisé pour afficher le manuel de PureBasic à côté du code, que ce soit avec le raccourci F1 ou non. Voir préférences .

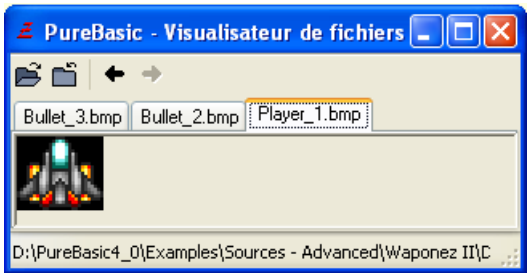
Les autres outils intégrés

Visualisateur de structures



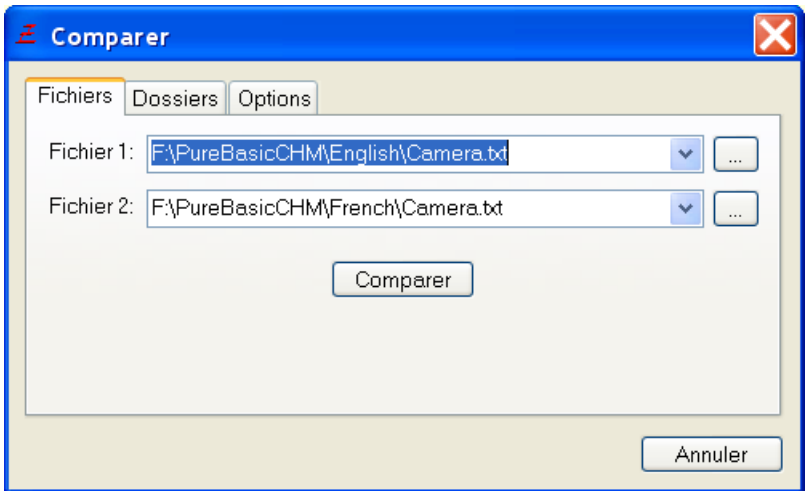
Cet outil permet de voir toutes les structures, interfaces et constantes qui sont prédéfinies dans PureBasic. Double-cliquer sur une structure ou une interface affichera la déclaration (le contenu) de l'élément. Il est possible de filtrer l'affichage en choisissant une lettre dans les boutons affichés au dessus de la liste.
Le bouton "Retour" revient à l'affichage précédant le double-clic.
Le bouton "Insérer nom" insère uniquement le nom de l'élément sélectionné.
Le bouton "Insérer copie" insère une copie de la déclaration de l'élément sélectionné.
Le bouton "Insérer" permet d'entrer le nom d'une variable et d'insérer tous les champs de la structure ou interface sélectionnée (en utilisant cette variable comme base).

Visualisateur de fichiers

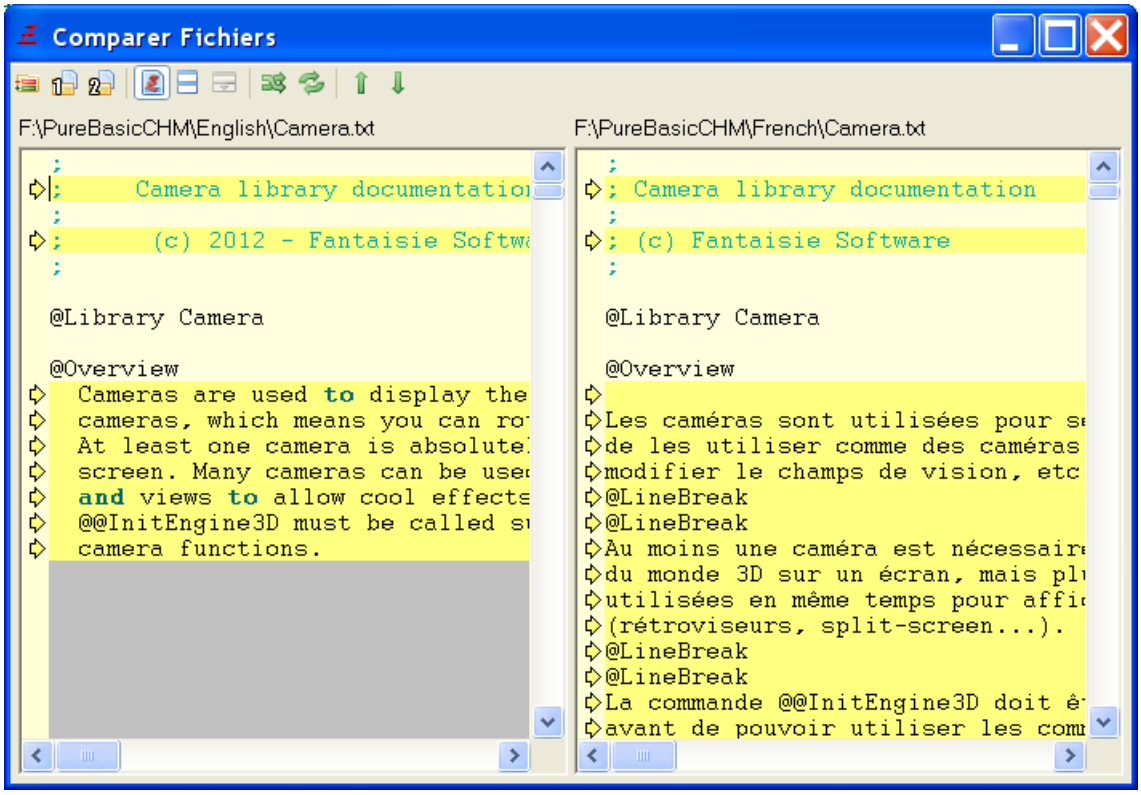


Le visualisateur de fichiers permet d'afficher plusieurs types de fichiers : textes, images et pages web (Windows seulement). Si un type de fichiers n'est pas reconnu, il sera affiché sous forme hexadécimale.
Le bouton "Ouvrir" ouvre un nouveau fichier et le bouton "X" ferme le fichier courant. Les flèches permettent de naviguer parmi les fichiers ouverts.
A noter qu'il est aussi possible d'ouvrir des fichiers dans le visualisateur interne en double-cliquant sur des fichiers binaires dans l'outil 'Explorer' ou sur le mot clef IncludeBinary dans la zone d'édition.

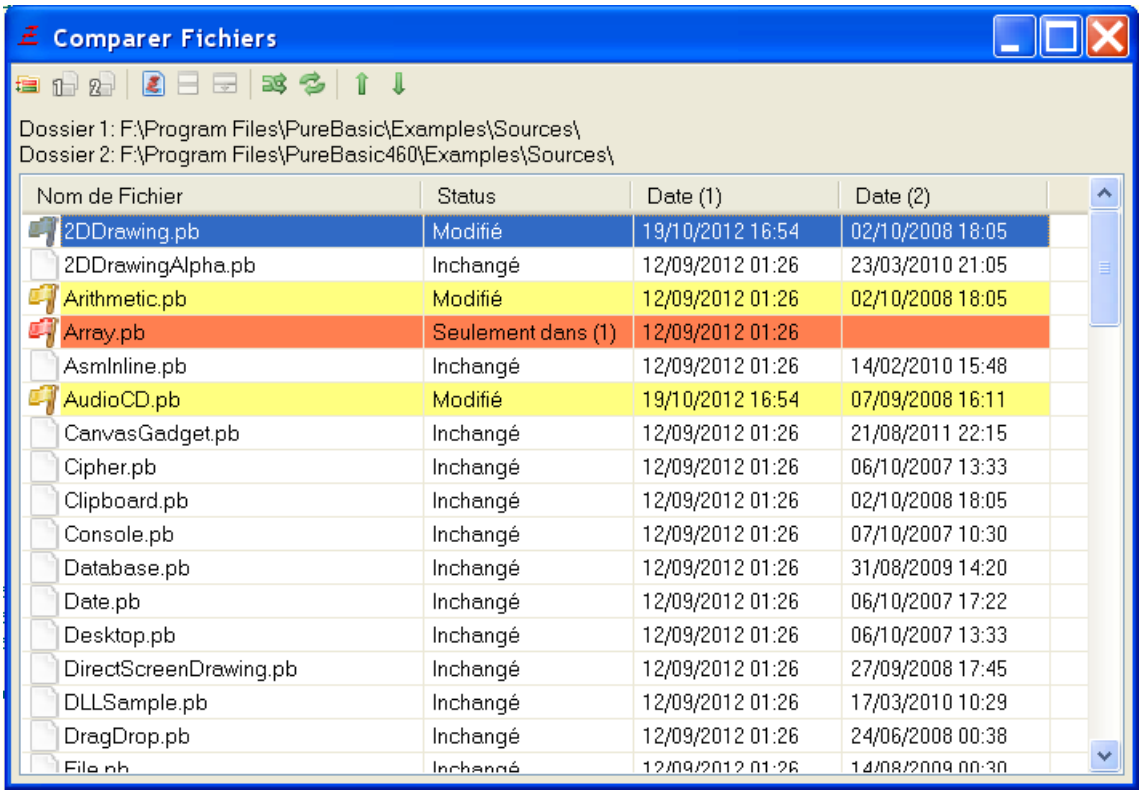
Comparer Fichiers/Dossiers



Cet outil permet de comparer deux fichiers (texte) ou des répertoires et de mettre en évidence leurs différences. L'onglet "Options" peut être utilisé pour ignorer des différences telles que les espaces ou les changements majuscules/minuscules.



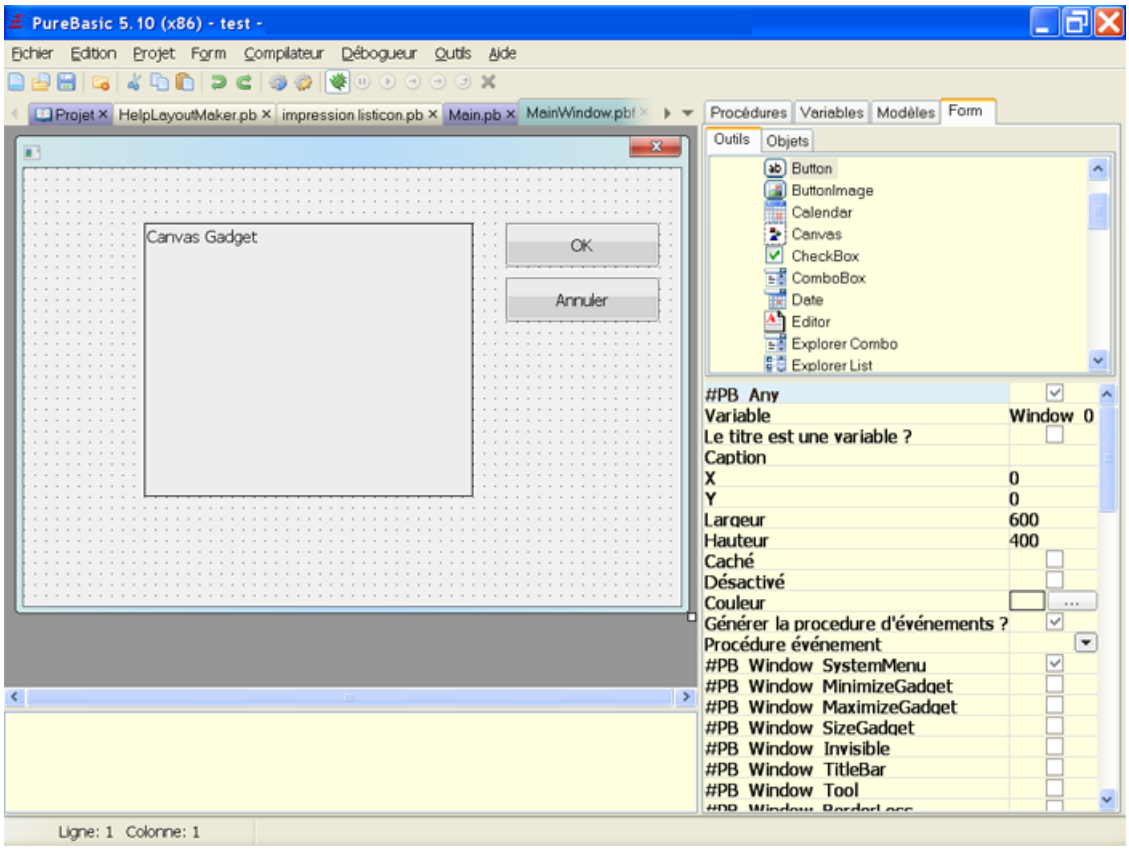
Les fichiers sont affichés côte à côte avec les différences marquées de la manière suivante : Les lignes affichées en rouge ont été enlevées dans le fichier, les lignes indiquées en vert ont été ajoutées dans le fichier et les lignes indiquées en jaune ont été modifiées.



Dans la comparaison de répertoires, le contenu des deux répertoires est examiné (avec la possibilité de filtrer la recherche par extension de fichier et d'inclure les sous-répertoires) et les fichiers sont marqués d'une manière similaire : Dossiers en rouge n'existent pas dans le deuxième répertoire, les fichiers en vert sont nouveaux dans le deuxième répertoire et fichiers en jaune ont été modifiés. Un double-clic sur un fichier modifié montre les modifications qui ont été apportées.

Autes outils du menu Outils

Concepteur de fenêtre

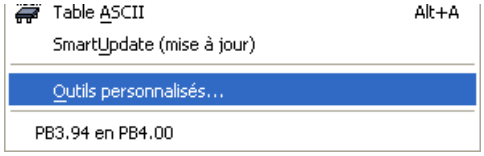


Le concepteur de fenêtre (ou de formulaire) peut être utilisé pour concevoir l'interface utilisateur de votre application. Pour plus d'informations, reportez-vous au chapitre concepteur de fenêtre .

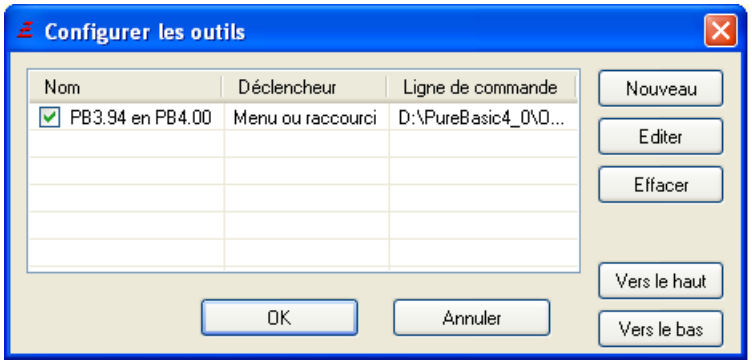
Chapitre 16

Les outils externes

L'IDE du PureBasic vous permet de configurer des programmes externes pour qu'ils soient appelés directement à partir d'un menu, de raccourcis, de la barre d'outils ou d'évènements "déclencheurs" spéciaux (ex : ouverture de fichier, fermeture etc..). Ceci permet un accès aisé à ces outils tout en programmant.
Vous pouvez également écrire vos propres petits outils en PureBasic pour effectuer des actions spéciales sur le code source que vous êtes en train d'éditer, afin d'automatiser les tâches récurrentes (ex : réorganisation du code, statistiques, etc..). De plus, vous pouvez configurer des visualisateurs de fichiers externes pour remplacer le visualisateur de fichier intégré à l'IDE pour des types particuliers de fichier ou pour tous les fichiers.



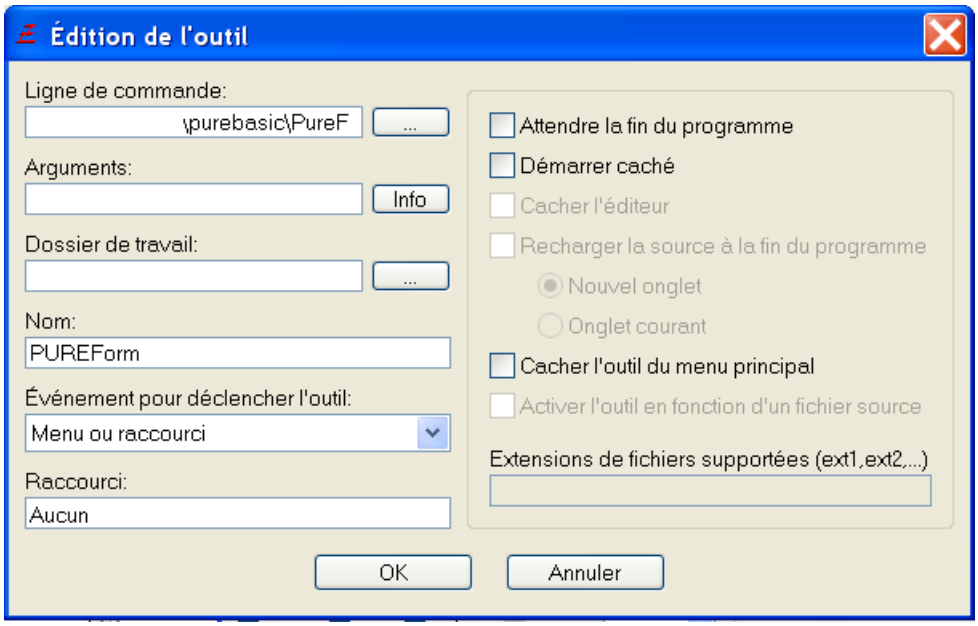
La commande "Outils personnalisés" du menu "Outils" ouvre la fenêtre de gestion des outils externes. La liste affiche tous les outils déjà configurés par ordre d'apparition dans le menu "Outils" (les outils ayant l'option 'caché' figurent aussi). Vous pouvez ajouter (bouton Nouveau) et supprimer (bouton Effacer) des outils, ou modifier leur ordre de priorité en cliquant sur les boutons "Vers le haut" / "Vers le bas" après avoir sélectionné un élément.



Chaque outil peut être rapidement activé ou désactivé à partir de la fenêtre "Outils personnalisés" du menu "Outils", en cochant ou décochant la case à gauche du nom de l'outil.

Configurer un outil

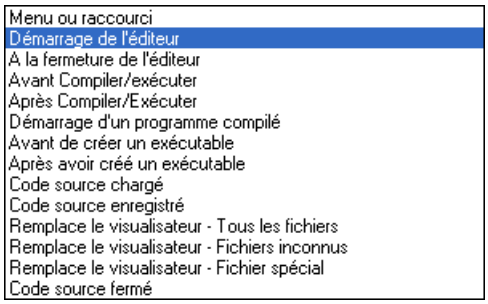
Les seuls éléments obligatoires dans la configuration sont la ligne de commande du programme à exécuter et son nom dans la liste du menu "Outils". Tout le reste est optionnel.



Ligne de commande
Contient le chemin et le nom du programme à exécuter.
Arguments
Contient les arguments qui seront passés au programme. Vous pouvez écrire des options fixes, tout comme des mots-clefs spéciaux qui seront remplacés lors de l'exécution du programme) :
%PATH : sera remplacé par le chemin du fichier source en cours d'édition. Il sera vide si le code source n'a pas encore été enregistré.
%FILE : sera remplacé par le nom du fichier source en cours d'édition. Il sera vide si le code source n'a pas encore été enregistré. Si l'outil est destiné à remplacer le visualisateur de fichier intégré, ce mot-clef représentera le fichier à ouvrir.
%TEMPFILE : quand cette option est spécifiée, le fichier source en cours d'édition est enregistré dans un fichier temporaire dont le nom est inséré ici. Ce fichier est créé uniquement pour l'outil et peut être librement modifié ou effacé.
%COMPILEFILE : ce mot-clef est uniquement valide pour les déclencheurs de compilation (voir ci-dessous). Il sera remplacé par le nom du fichier temporaire qui est envoyé au compilateur pour effectuer la compilation. En modifiant ce fichier, il est possible de changer ce qui sera effectivement compilé.
%EXECUTABLE : sera remplacé par le nom de l'exécutable créé par la dernière commande "Créer un exécutable". Pour le déclencheur "Après Compiler/Exécuter", il sera remplacé par le nom du fichier exécutable temporaire créé par le compilateur.
%CURSOR : sera remplacé par la position actuelle du curseur dans le code en cours d'édition, sous la forme : LIGNExCOLONNE.
%SELECTION : sera remplacé par la sélection actuelle sous la forme : LIGNEDEBUTxCOLONNEDEBUTxLIGNEFINxCOLONNEFIN. Il peut être utilisé en conjonction avec %TEMPFILE, si vous voulez que votre outil effectue des actions basées sur le texte sélectionné.
%WORD : sera remplacé par le mot actuellement situé sous le curseur.
%PROJECT : le chemin complet vers le répertoire contenant le fichier du projet si un projet est ouvert.
%HOME : le chemin complet vers le répertoire PureBasic. Note : pour chaque mot-clef désignant un fichier ou un chemin, il est généralement conseillé de les placer entre "" (ex : "%TEMPFILE") pour s'assurer que les chemins contenant des espaces seront correctement transmis à l'outil. Ces mots-clefs ainsi qu'une brève description peuvent aussi être consultés en cliquant sur le bouton "Info" à droite du champ Arguments.
Dossier de travail
Permet de choisir un répertoire que l'outil utilisera lors de son lancement. Si aucun répertoire n'est indiqué, l'outil utilisera le répertoire du code source en cours d'édition.
Nom
Permet d'attribuer un nom à l'outil. Ce nom s'affichera dans la liste des outils, et si l' outil n'est pas caché, dans le menu "Outils".

Evènement pour déclencher l'outil

Ici vous pouvez choisir quand l'outil devra être lancé. Plusieurs outils peuvent avoir le même déclencheur, ils seront tous exécutés quand cet évènement déclencheur se produira. L'ordre de leur exécution dépend de l'ordre dans lequel ils apparaissent dans la liste des outils.



Menu ou raccourci
L'outil ne sera pas lancé automatiquement. Il sera exécuté à partir d'un raccourci clavier ou d'un menu. Note : pour exécuter un programme à partir de la barre d'outils, il faut lui ajouter un bouton dans Fichier/Préférences/Général/Barre d'outils (voir Configurer l'IDE pour plus d'informations).
Si ce déclencheur est choisi, l'option "Raccourci" devient active et permet d'entrer un raccourci qui lancera cet outil.
Démarrage de l'éditeur
L'outil sera exécuté immédiatement après le démarrage complet de l'IDE.
A la fermeture de l'éditeur
L'outil sera exécuté juste avant que l'IDE ne quitte. Notez que tous les fichiers sources auront déjà été fermés.
Avant Compiler/Exécuter
L'outil sera exécuté juste avant la compilation du fichier source. En utilisant le mot-clef %COMPILEFILE, il est possible de modifier le code à compiler. Ceci permet par exemple de créer un petit pré-processeur pour le code source. Notez que vous devriez activer l'option "Attendre la fin du programme" si vous voulez que les modifications soient prises en compte par le compilateur.
Après Compiler/Exécuter
L'outil sera exécuté juste après la compilation, mais avant que le programme ne soit executé. En utilisant le mot-clef %EXECUTABLE, il est possible de récupérer le nom du fichier qui vient d'être généré. Les modifications sont autorisées, mais si le fichier est effacé, alors une erreur surviendra lorsque l'IDE essaiera d'exécuter le fichier.
Démarrage d'un programme compilé
L'outil sera lancé lorsque la commande "Exécuter" du menu "Compilateur" est activée. L'outil est lancé avant que le programme ne soit exécuté. Le mot-clef %EXECUTABLE est valide ici aussi.
Avant de créer un exécutable
Cet évènement est identique à "Avant Compiler/Exécuter", mais il est déclenché juste avant la création de l'exécutable final.
Après avoir créé un exécutable
L'outil sera lancé une fois que l'exécutable final aura été créé. Le mot-clef %EXECUTABLE peut servir à récupérer le nom de l'exécutable créé et ainsi effectuer des actions dessus (ex : pour le compresser).
Code source chargé
L'outil se lancera à chaque fois qu'un code source sera chargé dans l'IDE. Les mots-clefs %FILE et %PATH sont toujours valides, car le fichier est forcément chargé à partir d'un média (disque, réseau etc..).
Code source enregistré
L'outil se lancera à chaque fois qu'un code source sera enregistré par l'IDE. Les mots-clefs %FILE et %PATH sont toujours valides, car le fichier vient d'être enregistré sur un média (disque, réseau etc.).
Code source fermé
L'outil se lancera à chaque fois qu'un code source sera sur le point d'être fermé. A ce stade le fichier est toujours là, donc vous pouvez obtenir son contenu avec le mot-clef %TEMPFILE. Le mot-clef %FILE sera vide si le fichier n'a jamais été enregistré.
Remplace le visualisateur - Tous les fichiers
L'outil remplacera complètement le visualisateur de fichier intégré. Si on essaie d'ouvrir un fichier qui ne

peut être édité dans l'IDE, l'IDE va d'abord essayer les outils externes ayant comme déclencheur ce type de fichier particulier, et si aucun ne correspond, alors le fichier sera géré par cet outil. Utilisez le mot-clef %FILE pour récupérer le nom du fichier qui doit être ouvert.
Note : un seul outil peut être associé à ce déclencheur. Tous les autres outils associés à ce déclencheur seront ignorés.

Remplace le visualisateur - Fichiers inconnus

Cet outil remplacera le visualisateur hexadécimal intégré, qui est normalement utilisé pour ouvrir les fichiers de types inconnus. Il sera exécuté, lorsque l'extension du fichier est inconnue pour l'IDE, et si aucun autre outil externe n'a été configuré pour gérer ce fichier (si un outil est configuré avec le déclencheur "Remplace le visualisateur - Tous les fichiers", alors cet outil ne sera jamais appelé).
Note : un seul outil peut être associé à ce déclencheur.

Remplace le visualisateur Fichier spécial

Ce déclencheur permet à l'outil de gérer des extensions de fichiers spécifiques. Il a une plus haute priorité que les évènements "Remplace le visualisateur - Tous les fichiers", "Remplace le visualisateur Fichiers inconnus" et plus haute également que le visualisateur de fichier intégré. Indiquez les extensions que l'outil doit gérer dans le champ prévu à cet effet, sur la droite. Plusieurs extensions peuvent être attribuées.
Une utilisation courante de ce déclencheur est, par exemple, la configuration d'un programme comme Acrobat Reader pour gérer les fichiers ayant l'extension "pdf". Ce qui permet d'ouvrir facilement ces fichiers à partir de l'Explorateur, du visualisateur de fichier intégré ou en double-cliquant sur le mot-clef "IncludeBinary" dans le source.

Ouvrir un fichier avec une extension spéciale

Cela sera déclenché pour des extensions de fichiers spécifiques. Il a une priorité plus élevée que les déclencheurs "Ouvrir un fichier binaire non PureBasic" ou "Ouvrir un fichier texte non PureBasic". Spécifie les extensions que l'outil doit gérer dans la zone d'édition à droite. Plusieurs extensions peuvent être données.
Note : Les déclencheurs d'ouverture de fichier sont actifs lorsque vous ouvrez un fichier via le menu Fichier/Ouvrir et également lorsque vous glisser-déposer un fichier dans l'IDE. Ils ont une priorité plus élevée pour les tâches Fichier/Ouvrir. Ce n'est que s'il n'y a pas d'outil actif que les déclencheurs de la visionneuse de fichiers seront gérés.

Ouvrir un fichier binaire non PureBasic

Celui-ci sera déclenché pour les fichiers binaires qui ne font pas partie de PureBasic (plus ou moins)
Note : Un seul outil à la fois peut avoir ce déclencheur. Tous les autres outils avec ce déclencheur seront ignorés.

Ouvrir un fichier texte non PureBasic

Celui-ci sera déclenché pour les fichiers texte qui ne font pas partie de PureBasic
Note : Un seul outil à la fois peut utiliser ce déclencheur.

Autres options sur le côté droit

Attendre la fin du programme

L'IDE sera complètement bloqué jusqu'à ce que l'outil termine son exécution. Cette option est requise si l'outil doit modifier un fichier source qui doit être rechargé par la suite, ou passé au compilateur par les déclencheurs de compilation.

Démarrer caché

L'outil sera lancé en mode invisible. Ne pas utiliser cette option si le programme nécessite une interaction avec l'utilisateur, car il ne pourra plus être fermé.

Cacher l'éditeur

Uniquement disponible si l'option "Attendre la fin du programme" est activée. Cache l'éditeur pendant l'exécution de l'outil.

Recharger la source à la fin du programme

Uniquement disponible si l'option "Attendre la fin du programme" est activée, et quand le mot-clef %FILE ou %TEMPFILE est utilisé dans le champ "Arguments".
Une fois que l'outil a terminé son exécution, l'IDE rechargera le fichier source dans l'éditeur. Il est possible de choisir si le fichier doit être rechargé dans l'onglet actuel ou dans un nouvel onglet.

Cacher l'outil du menu principal

N'affiche pas l'outil dans la liste du menu "Outils" de l'IDE. C'est utile pour les outils qui doivent uniquement être exécutés à partir de déclencheurs spéciaux, mais pas à partir du menu.

Activer l'outil en fonction d'un fichier source
Les outils avec cet option seront inscrits dans la liste "Exécuter des outils" dans les options du compilateur , et exécutés uniquement pour les sources où il est permis. Notez que lors de la désactivation de l'outil dans la fenêtre "outils de configuration", il sera désactivé au niveau global et ne pourra être lancé depuis une source, même s'il est activé ici.
Cette option n'est valable qu'avec :
- Avant Compiler/Exécuter - Après Compiler/Exécuter - Exécuter Programme compilé - Avant créer Exécutable - Après créer Exécutable - Code source chargé - Code source enregistré - Code source fermé
Extensions des fichiers supportées (ext1,ext2,...)
Uniquement disponible pour le déclencheur "Remplace le visualisateur - Fichier spécial". Permet de saisir la liste des extensions gérées.

Astuces pour l'écriture de vos propres outils de traitement

L'IDE fournit des informations supplémentaires pour les outils sous la forme de variables d'environnement. Elles peuvent être facilement lues par l'outil à l'aide des commandes de la bibliothèque Process .
Voici une liste des variables fournies. Notez que celles qui donnent des informations sur le fichier en cours d'édition ne sont pas accessibles pour les outils exécutés au démarrage ou à la fermeture de l'IDE.

```
  PB_TOOL_IDE         - Chemin complet et nom de l'exécutable de l'IDE
  PB_TOOL_Compiler    - Chemin complet et nom de l'exécutable du
   compilateur
  PB_TOOL_Preferences - Chemin complet et nom du fichier des
   préférences de l'IDE
  PB_TOOL_Project     - Chemin complet et le nom du projet actuellement
   ouvert (le cas échéant)
  PB_TOOL_Language    - Langue actuellement utilisée dans l'IDE
  PB_TOOL_FileList    - Liste de tous les fichiers ouverts dans l'IDE,
   séparés par un Chr(10)

  PB_TOOL_Debugger    - Ces variables fournissent les paramètres
   correspondant à ceux de la fenêtre d'options de compilation

  PB_TOOL_InlineASM     pour le source en cours. Elles sont à 1 si
   l'option
  PB_TOOL_Unicode       est activée, à 0 sinon.
  PB_TOOL_Thread
  PB_TOOL_XPSkin
  PB_TOOL_OnError

  PB_TOOL_SubSystem   - Contenu du champ Bibliothèque Sous-système
 dans les options de compilation
  PB_TOOL_Executable  - Identique au mot-clef %COMPILEFILE pour la
   ligne de commande
  PB_TOOL_Cursor      - Identique au mot-clef %CURSOR      pour la
   ligne de commande
  PB_TOOL_Selection   - Identique au mot-clef %SELECTION   pour la
   ligne de commande
  PB_TOOL_Word        - Identique au mot-clef %WORD        pour la
   ligne de commande

  PB_TOOL_MainWindow  - Identifiant système (Handle) de la fenêtre
   principale de l'IDE
  PB_TOOL_Scintilla   - Identifiant système (Handle) du composant
   d'édition de code Scintilla pour le source en cours
```

Quand les mots-clefs %TEMPFILE ou %COMPILEFILE sont utilisés, l'IDE ajoute les options de

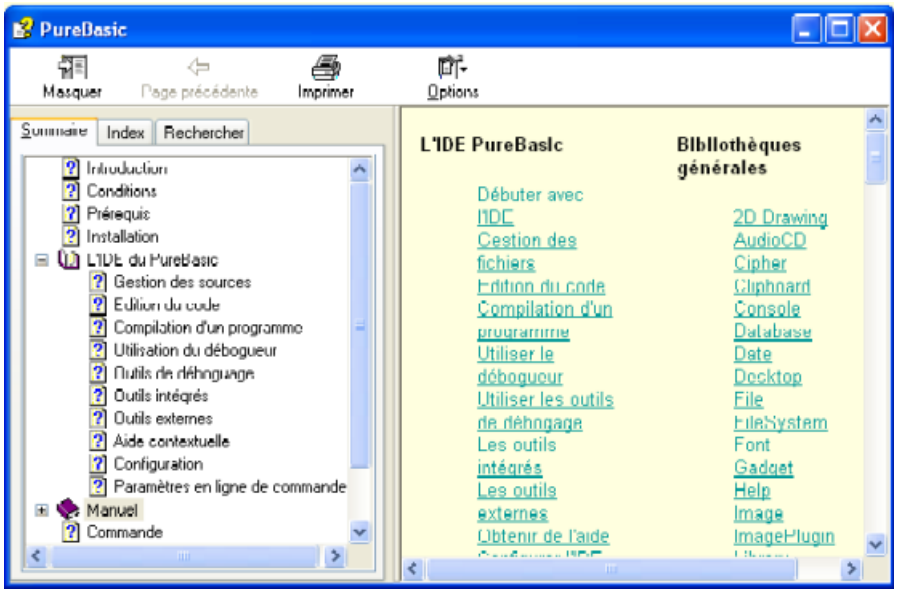
compilation à la fin du fichier temporaire (sous forme de commentaires), même si l'utilisateur a choisi de ne pas sauver ces options en enregistrant un code source.
Ceci permet à l'outil de lire les options de compilation spécifiques à ce fichier, et de les prendre en compte dans les actions qu'il va effectuer.

Chapitre 17

Obtenir de l'aide

L'IDE du Purebasic permet l'accès immédiat à l'aide de PureBasic, ainsi qu'aux autres fichiers d'aides disponibles sur la plateforme de développement sans quitter l'éditeur.

Accès rapide à l'aide de référence



En sélectionnant la commande "Aide..." du menu "Aide" ou en utilisant son raccourci clavier (par défaut F1) quand le curseur est sur un mot-clef ou une fonction PureBasic, l'aide sera ouverte à la page correspondante.
Si le mot à la position du curseur n'a pas de rubrique dédiée dans l'aide, la page de référence sera affichée.

Accès rapide à l'aide sur l'API Windows

Il y a deux méthodes pour accéder rapidement à l'aide des fonctions de l'API Windows supportées par PureBasic :
Le Microsoft Plateform SDK Ce SDK regroupe toute l'aide actuellement disponible pour le développement d'un programme sous Windows. Il contient des informations sur toutes les fonctions API, ainsi que des petits tutoriaux et des discussions techniques sur des points particuliers. De ce fait, il est plutôt conséquent (jusqu'à 400 Mo en fonction des composants installés).
Pour l'intégration avec l'IDE, il est possible d'installer l'édition de "Fevrier 2003" ou de "Windows Server 2003 SP1".

Le SDK peut être téléchargé à partir du lien suivant :
https ://www.microsoft.com/en-us/download/details.aspx ?id=15656
A noter qu'il peut aussi être commandé par CD. Par ailleurs, il est aussi fourni dans la plupart des logiciels de développement de Microsoft (comme Visual Studio).
Le fichier d'aide Win32.hlp Il y a une autre alternative bien plus petite au SDK complet (7.5 Mo). Ce fichier d'aide est très ancien (écrit pour Windows 95), donc il ne contient pas d'informations concernant les API introduites depuis là. Néanmoins, il contient les informations essentielles à propos des API les plus utilisées, qui sont toujours d'actualité car elles n'ont quasiment pas évolué. Ce fichier est recommandé seulement pour une utilisation occasionnelle des API (et que le SDK n'est pas disponible).
Le fichier peut être téléchargé ici :
http ://www.purebasic.com/download/WindowsHelp.zip
Pour l'utiliser à partir de l'IDE, il suffit de créer le sous-répertoire "Help" dans le dossier "PureBasic" et d'y copier le fichier "Win32.hlp".

Accéder à d'autres fichiers d'aide

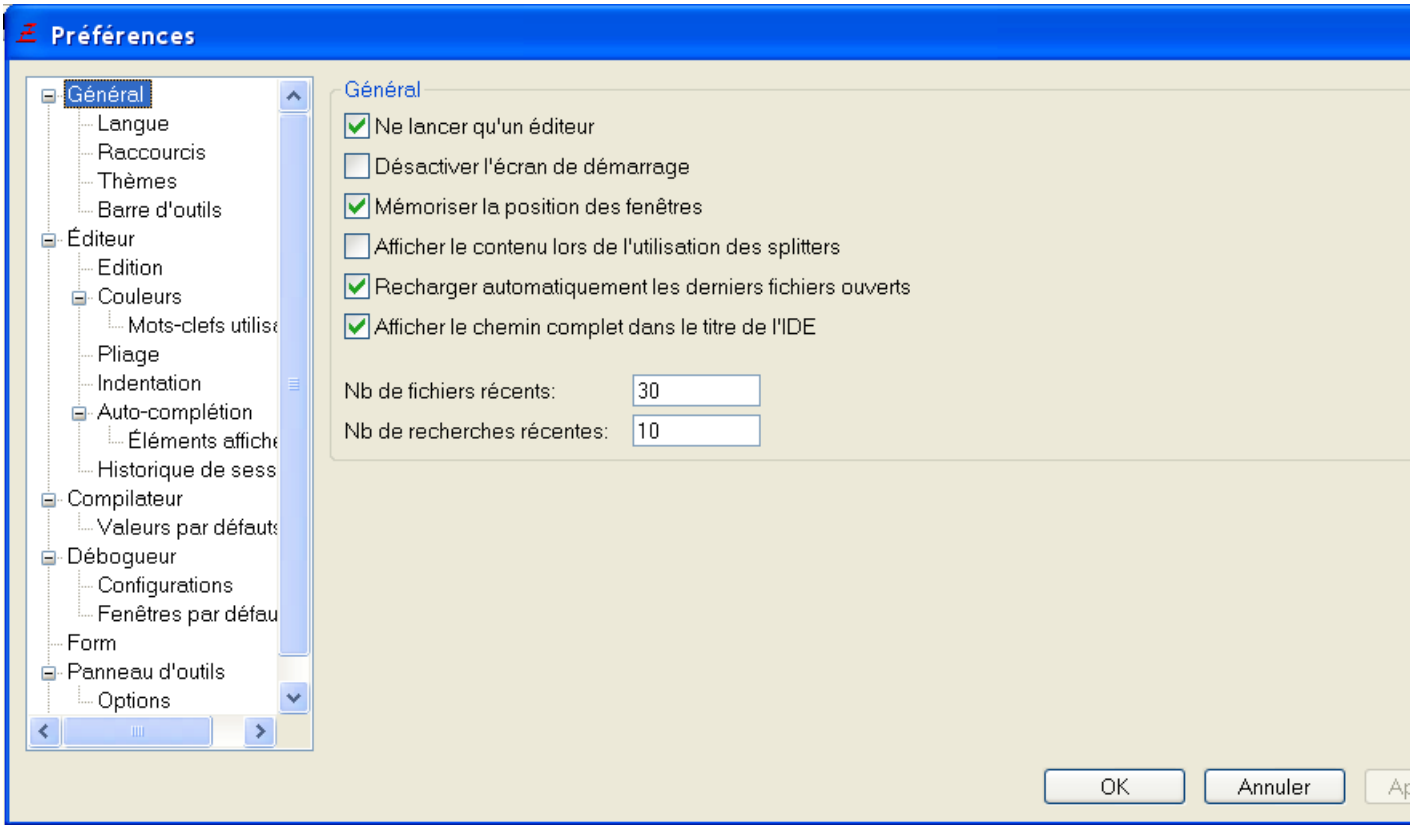
Pour accéder à d'autres fichiers d'aide à partir de l'IDE, il faudra créer le sous-répertoire "Help" dans le dossier "PureBasic" puis les copier dedans. Ces fichiers apparaîtront alors dans le sous-menu "Aide externe" du menu "Aide" (et dans le menu contextuel de la zone d'édition). Les fichiers .chm et .hlp seront affichés par les visualisateurs Microsoft. L'IDE ouvrira les fichiers d'aide dans le visualisateur interne. Les fichiers tels que les textes seront donc directement consultables. Pour les autres types, il faudra utiliser le menu Configuration des outils pour paramétrer un outil externe qui gérera ce type de fichier. L'aide sera alors affichée en utilisant cet outil.
Par exemple, si des fichiers d'aide sont en pdf, il suffit de paramétrer un outil externe qui ouvrira les pdf et mettre les fichiers pdf dans le sous-répertoire "Help" du dossier "PureBasic". Lorsqu'un de ces fichiers d'aide sera sélectionné dans le sous-menu "Aides externes", il sera ouvert en utilisant cet outil.

Chapitre 18

Configurer l'IDE

L'IDE du PureBasic propose un nombre très conséquent d'options pour qu'il puisse s'adapter aux habitudes des programmeurs qui l'utiliseront. Ces paramètres sont regroupés dans la fenêtre de la commande Préférences du menu "Fichier", et la description de chacune est décrite dans ce document. Tout changement ne sera effectif uniquement lorsque le bouton "OK" ou "Appliquer" sera utilisé.

Général



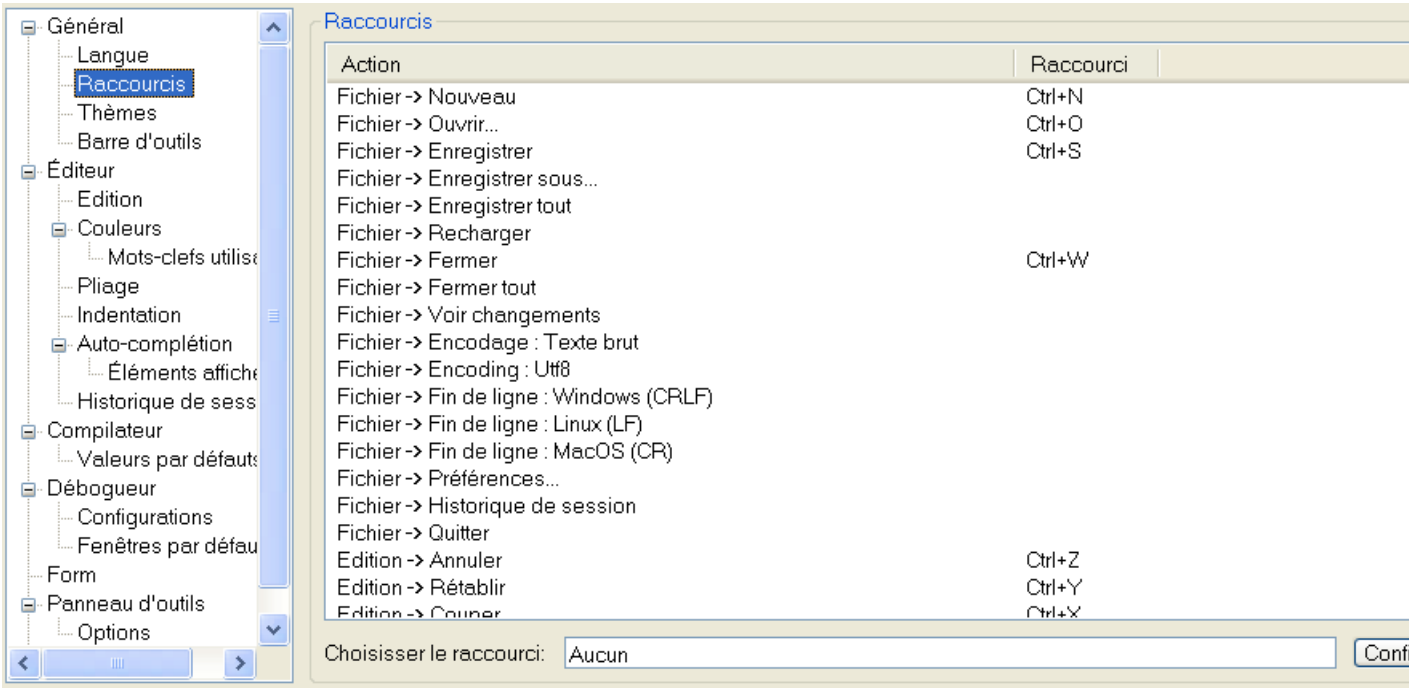
Cette section regroupe les options qui influent sur le comportement général de l'IDE.
Ne lancer qu'un éditeur
Si cette option est activée, l'IDE ne pourra être exécuté qu'une seule fois, et toute exécution ultérieure activera l'IDE déjà ouvert. Par exemple cliquer sur un fichier .pb dans l'explorateur ouvrira le fichier dans un nouvel onglet au lieu de lancer une nouvelle instance de l'IDE.
Désactiver l'écran de démarrage
Désactive l'écran "PureBasic" qui est affiché lors du démarrage de l'IDE.

Mémoriser la position des fenêtres
Enregistre la position et la taille des fenêtres de l'IDE lorsqu'elles sont fermées. Pour avoir les fenêtres toujours ouvertes avec les mêmes dimensions, il faudra les placer à l'endroit désiré, activer cette option, puis quitter l'IDE (pour enregistrer les options). Au prochain démarrage, désactiver cette option et les fenêtres utiliseront la taille et la position précédemment enregistrées.
Afficher le contenu lors de l'utilisation des splitters
Si l'ordinateur de développement est rapide, cette option peut être activée, sinon déplacer les splitters peut entrainer un scintillement désagréable.
Recharger automatiquement les derniers fichiers ouverts
Recharge automatiquement tous les fichiers qui étaient ouverts lorsque l'IDE a quitté pour la dernière fois.
Afficher le chemin complet dans le titre de l'IDE
Si cette option est activée, le titre de la fenêtre principale de l'IDE affichera le chemin complet du fichier en cours d'édition, sinon seul le nom de fichier sera affiché.
Activer le mode sombre
Le thème 'Dark Mode' est utilisé.
Liste des fichiers récents
Nombre de fichiers récents à afficher dans le sous-menu "Fichiers récents" du menu "Fichier".
Recherches récentes
Détermine le nombre de mots récents qui seront mémorisés dans les commandes "Rechercher/Remplacer" et "Rechercher dans les fichiers".
Vérifier les mises à jour
Indique combien de fois l'IDE doit vérifier sur le serveur de purebasic.com la disponibilité des nouvelles mises à jour. Une vérification de mise à jour peut également être effectuée manuellement à tout moment à partir du menu "Aide".
Vérifier les versions finales (releases)
Indique quelles sont les types de version qui provoqueront une notification si elles sont disponibles.

Général - Langue

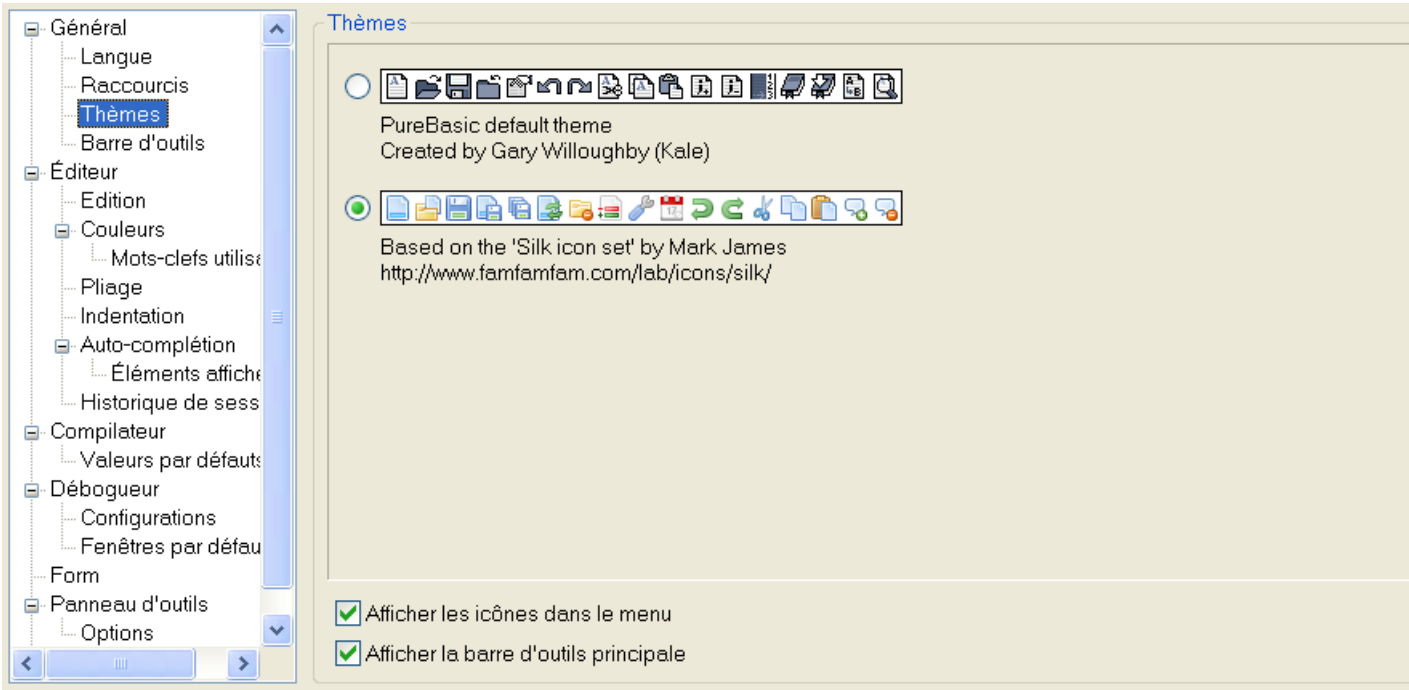
Permet de changer la langue utilisée par l'IDE. La liste déroulante affiche les langues disponibles et des informations relatives à la traduction (ex : la personne qui a fait la traduction et la date).

Général - Raccourcis claviers



Dans cette section il est possible de paramétrer absolument tous les raccourcis pour les commandes de l'IDE. Créer un nouveau raccourci, selectionner le champ de saisie du raccourci, taper directement le raccourci souhaité sur le clavier puis cliquer sur "Appliquer".
A noter que les touches Tab et Shift+Tab sont réservées pour l'indentation des blocs de code et ne peuvent pas être modifiées. De plus certaines combinaisons de touches peuvent avoir un sens particulier pour l'OS et ne pourront pas être utilisées.

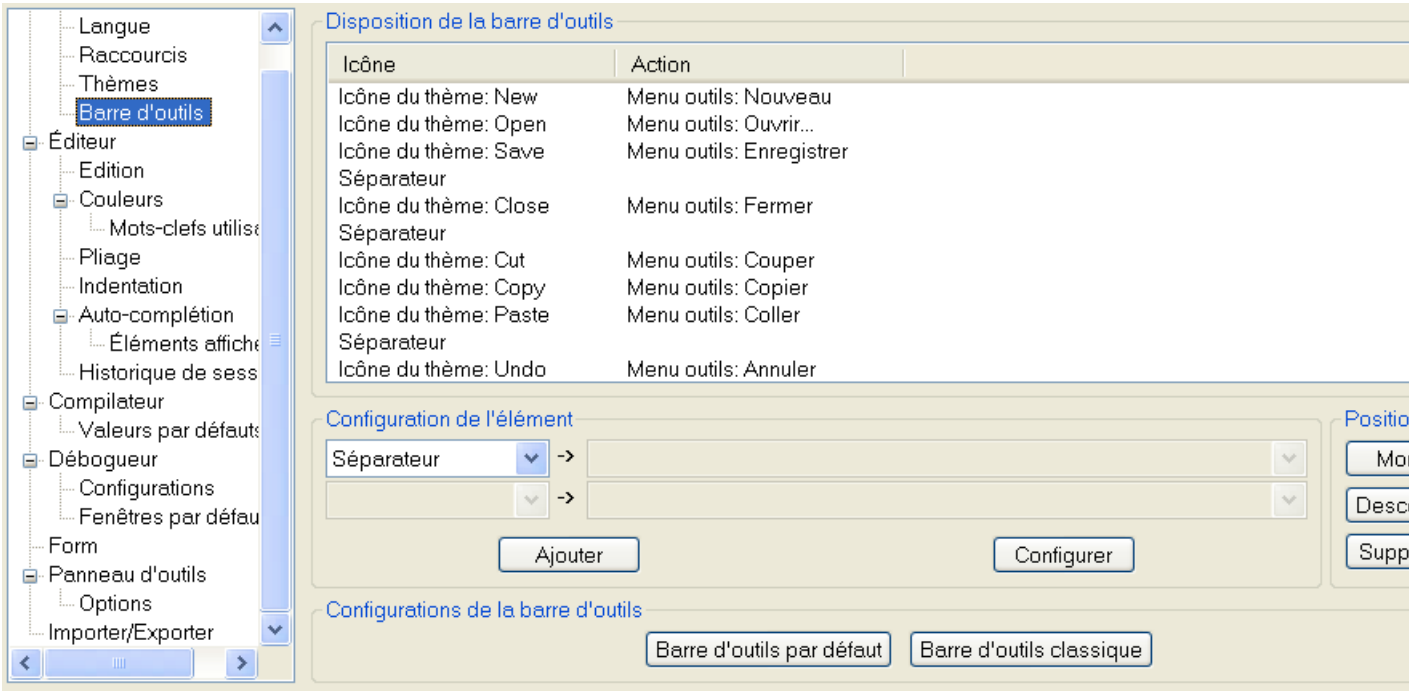
Général - Thèmes



Cette section permet de visualiser les thèmes disponibles pour l'IDE et d'en sélectionner un. Par défaut, deux thèmes sont à disposition.
Davantage de thèmes peuvent être facilement ajoutés en créant un fichier zip contenant les images (au

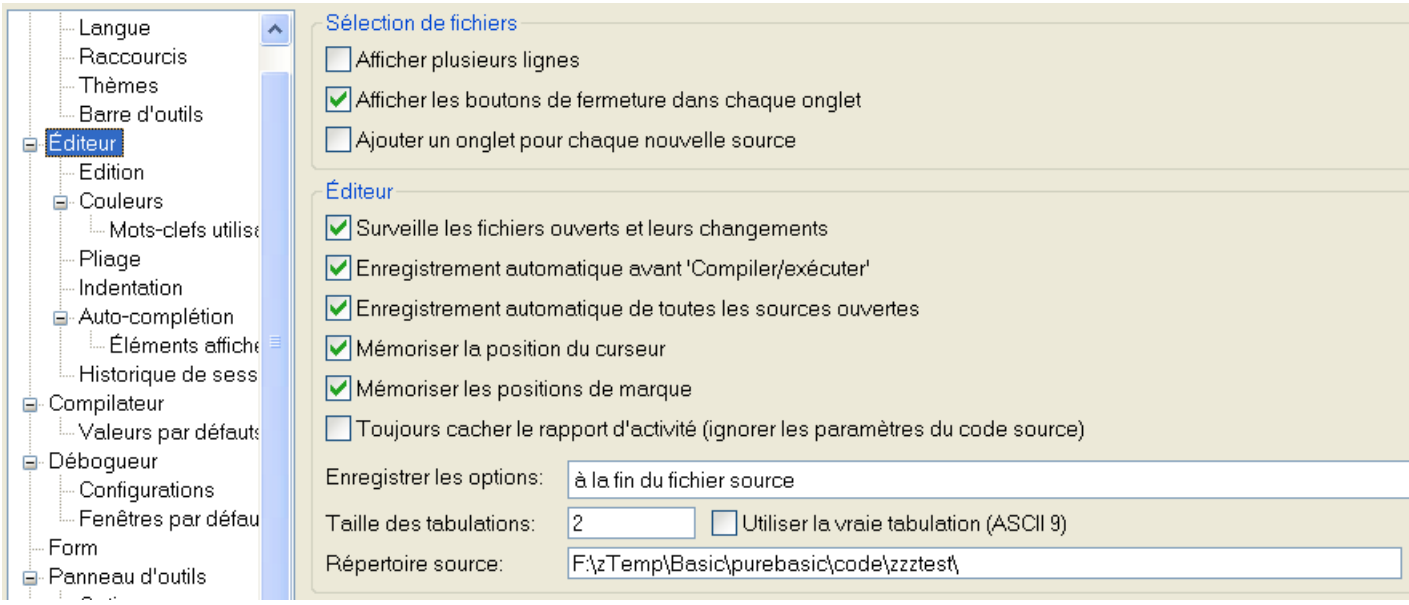
format PNG) et un fichier "Theme.prefs" qui décrit le thème. Le fichier zip doit ensuite être copié dans le répertoire "Themes" de PureBasic pour être reconnu par l'IDE. Le fichier "SilkTheme.zip" peut être utilisé comme base pour créer un nouveau thème.
Afficher les icônes dans le menu
Permet d'afficher ou de cacher les icônes dans les menus de l'IDE.
Afficher la barre d'outils principale
Permet d'afficher ou de cacher la barre d'outils principale, pour gagner un peu de place dans la fenêtre d'édition.

Général - Barre d'outils



La barre d'outils est entièrement paramètrable. Pour changer la position des icônes, utiliser les boutons de la section "Position". Pour modifier un bouton ou en ajouter un nouveau, utiliser les boutons de la section "Configuration de l'élément" (les nouveaux éléments sont toujours ajoutés en fin de liste).
Types des éléments :
Séparateur : une barre verticale de séparation.
Espacement : un espace vide de la taille d'une icône.
Icône standard : permet de choisir une icône système (fournie par l'OS) dans la liste déroulante.
Icône de l'IDE : permet de choisir une icône propre à l'IDE dans la liste déroulante.
Fichier icône : permet de choisir une icône à partir d'un fichier spécifié dans le champ à droite (les fichiers PNG sont acceptés sur toutes les plateformes, ainsi que les fichiers icône sous Windows).
Si le type de l'élément n'est pas un "Séparateur" ni un "Espacement", alors il est possible de lui associer une action lorsque le bouton sera pressé :
Elément d'un menu : exécutera la commande correspondante à l'élément du menu spécifié dans la liste déroulante.
Lancer un outil : exécutera l'outil externe spécifié dans la liste déroulante.
La section "Configurations de la barre d'outils" contient les deux configurations de base de la barre d'outils qu'il est possible de choisir et de modifier à convenance.

Editeur



Paramètres qui influent sur l'édition des codes sources

Surveille les fichiers ouverts et leurs changements

Surveille toutes les modifications qui sont apportées aux fichiers sur le disque alors qu'ils sont édités dans l'IDE.

Si des modifications sont apportées par d'autres programmes, un avertissement s'affiche avec le choix de recharger le fichier à partir du disque.

Enregistrement automatique avant 'Compiler/Exécuter'

Enregistre le code source en cours d'édition avant chaque 'Compiler/Exécuter'. A noter que les fichiers inclus ne sont jamais enregistrés automatiquement.

Enregistrement automatique avant 'Créer un exécutable'

Enregistre le code source en cours d'édition avant de créer un fichier exécutable.

Enregistrement automatique de toutes les sources ouvertes

Enregistre toutes les sources et pas seulement la source en cours avec l'une des options de sauvegarde automatique .

Mémoriser la position du curseur

Enregistre la position du curseur ainsi que l'état de tous les replis pour le fichier en cours d'édition.

Mémoriser les positions des marques

Enregistre la position des marqueurs pour le fichier en cours d'édition.

Toujours cacher le rapport d'activité

Le journal d'erreurs est montré ou caché en fonction de la configuration de chaque code source. Cette option permet de s'assurer que le journal d'erreurs ne sera jamais affiché si l'utilisateur le souhaite. La partie du menu concernant le journal d'erreur est aussi supprimée.

Enregistrer les options

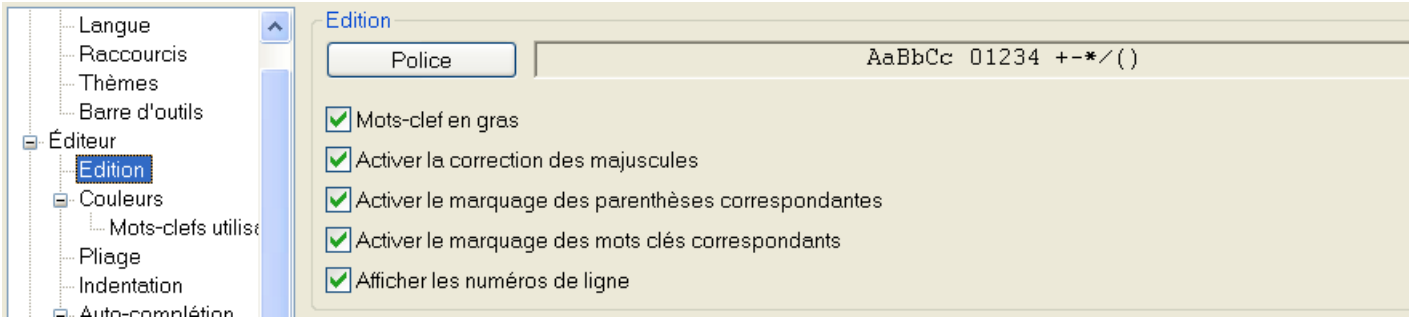
Permet de déterminer où les options propres à chaque fichier doivent être enregistrées :

> à la fin du fichier source
> Enregistre les options dans un bloc de commentaire à la fin du fichier. Lorsqu'un fichier contenant ce type de commentaires est ouvert par l'IDE, ils resteront invisibles.
> dans un fichier <nomdufichier>.pb.cfg
> Crée un fichier .pb.cfg pour chaque fichier enregistré qui contient ces informations.
> dans un fichier project.cfg pour chaque répertoire
> Crée un fichier nommé project.cfg dans chaque répertoire ou des fichiers PureBasic sont enregistrés. Ce fichier contiendra les options de tous les fichiers du répertoire.
> nulle part
> Les options ne sont pas enregistrées du tout. Quand les fichiers seront ouverts, ils utiliseront toujours les valeurs par défaut.

Valeur d'une tabulation

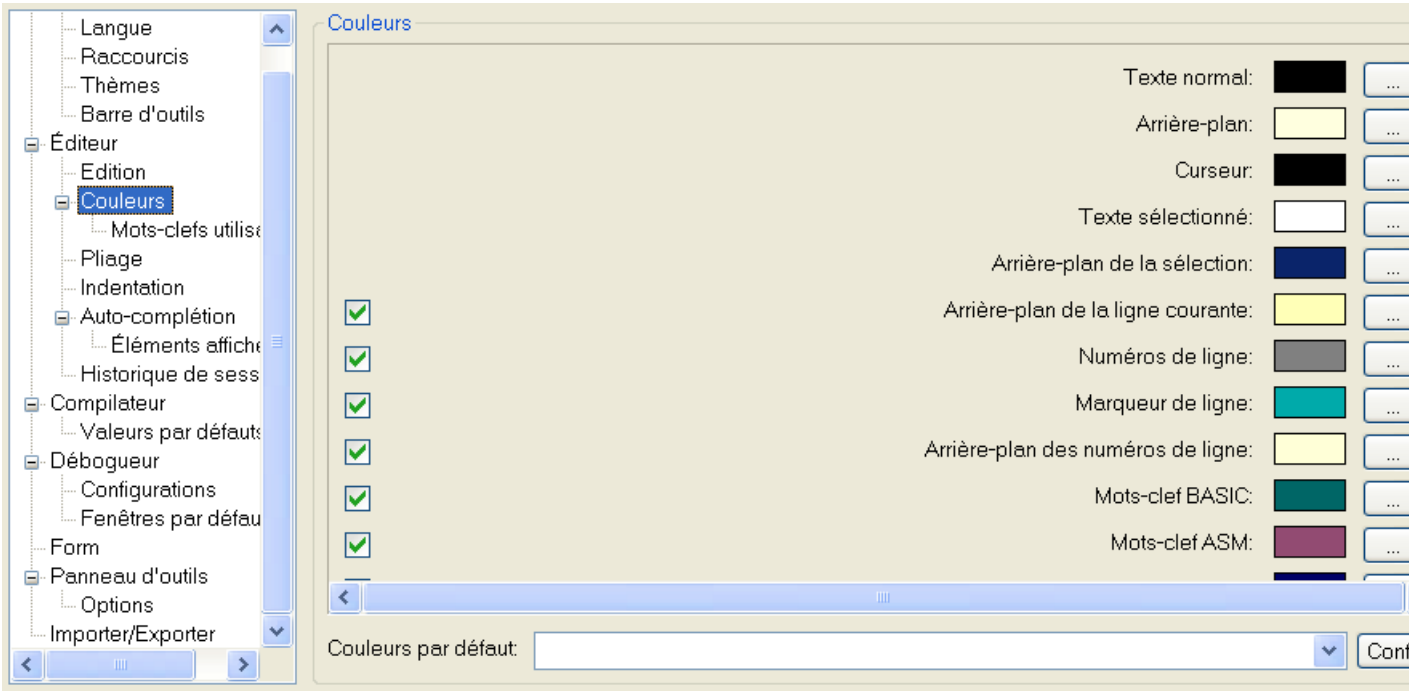
Permet de spécifier le nombre d'espaces qui seront insérés à chaque tabulation.
Utiliser les vraies tabulations (Ascii 9)
Si cette option est activée, les tabulations utiliseront le caractère 'tab' au lieu des espaces. A noter que si cette option est activée, le champ "Valeur d'une tabulation" définit la taille de la tabulation affichée.
Répertoire par défaut
Définit le répertoire par défaut qui sera utilisé par les commandes "Ouvrir" et "Enregistrer" si aucun fichier n'est ouvert, ou que le fichier en cours d'édition n'a pas encore été enregistré. Il est conseillé de mettre le répertoire où sont principalement stockés les codes sources.
Les extensions de fichier de code
L'IDE détecte les fichiers de code par leur extension (pb, pbi ou pbf par défaut). Les fichiers sans code sont modifiés dans un mode "texte brut" dans lequel les caractéristiques liées au code sont désactivées. Ce paramètre oblige l'IDE à reconnaître de nouvelles extensions de fichiers comme des fichiers de code. Le champ peut contenir une liste d'extensions séparées par des virgules (ex : "pbx, xyz").

Editeur - Edition



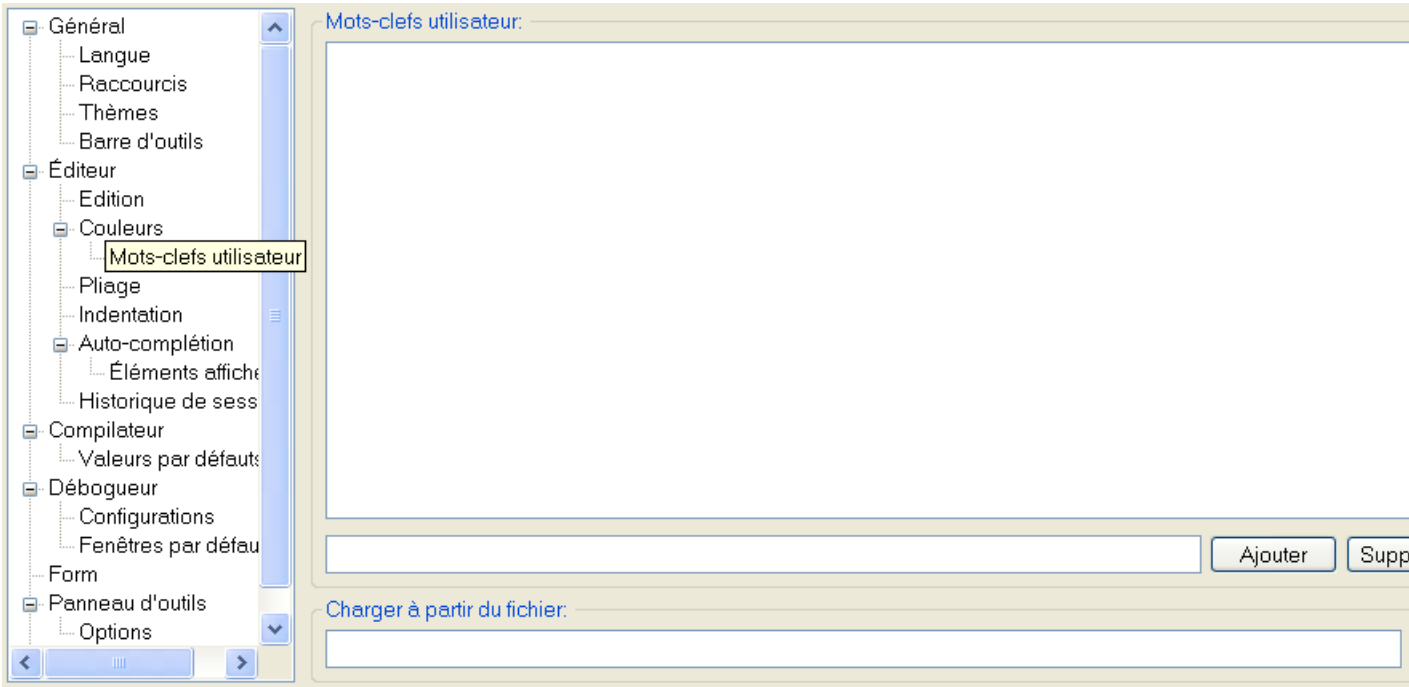
Utiliser "Sélectionner une police" pour changer la police de caractères utilisée pour l'affichage du code source. Pour assurer une bonne lisibilité du code, la police doit être de taille fixe (c'est à dire que tous les caractères ont la même largeur), si possible même pour les lettres en gras.
Activer la coloration syntaxique
Active ou désactive la coloration des codes sources. Le désactiver rend l'IDE légèrement plus rapide, mais bien moins convivial.
Mot clefs en gras
Si la police de caractères n'affiche pas les lettres en gras avec la même taille que les lettres normales, il est conseillé de désactiver cette option.
Activer la correction des majuscules
Corrige automatiquement les mots clefs PureBasic, les fonctions PureBasic et API ainsi que les constantes prédéfinies pour qu'ils soient exactement comme définis initialement (ex : 'openwindow()' sera automatiquement changé en 'OpenWindow()').
Activer le marquage des parenthèses correspondantes
Active le marquage des parenthèses lorsque le curseur se trouve sur une parenthèse. Cela permet de vérifier rapidement s'il y a le bon nombre de parenthèses, et de savoir à quelle parenthèse fermante correspond la parenthèse ouvrante (et vice-versa).
Activer le marquage des mots-clés correspondant
Permet de souligner tous les mots-clés en rapport avec le mot-clé actuellement sous le curseur.
Afficher les numéros de lignes
Affiche ou cache la numérotation des lignes située à gauche de la zone d'édition.

Editeur - Couleurs



Cette partie permet de modifier les couleurs utilisées par la coloration syntaxique ainsi que les marqueurs du débogueur. Des configurations prédéfinies sont disponibles dans la liste déroulante en bas de la fenêtre (une fois appliquées, elles sont modifiables). Chaque couleur peut être désactivée en utilisant les cases à cocher.
Note : Le modèle de couleur 'Accessibility' a (indépendamment des couleurs à fort contraste) un réglage spécial pour utiliser toujours la couleur système pour la sélection dans l'éditeur. Ceci permet aux applications qui permettent de lire un écran de mieux détecter le texte sélectionné.

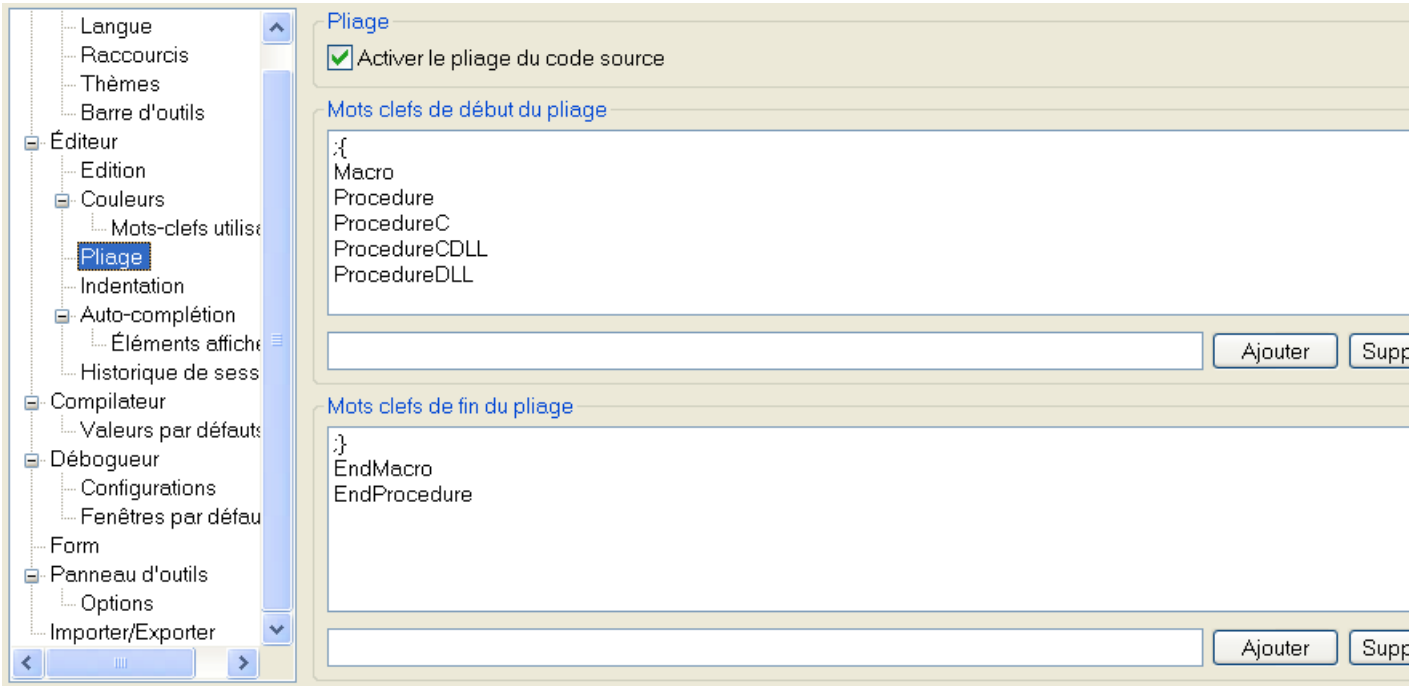
Editeur - Couleurs - Mots Clés personnalisés



Dans cette section, une liste de mots-clés personnalisés peut être définie. Ces mots-clés peuvent avoir une

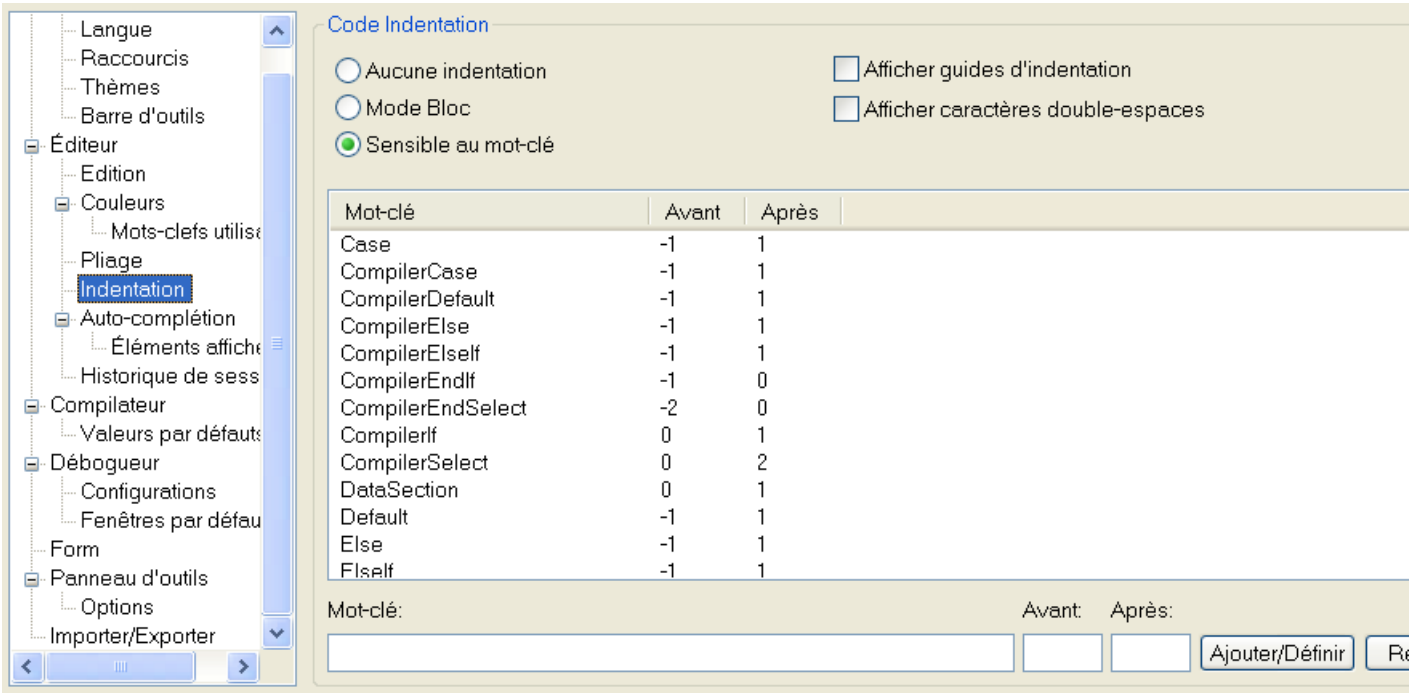
couleur spéciale qui leur est attribuée dans les options de couleur et l'IDE l'appliquera si cette fonction est activée. Cela permet d'appliquer une couleur particulière aux mots-clés par des outils-préprocesseurs ou des macros, ou tout simplement pour avoir des mots-clés PB colorés différemment.
Notez que ces mots-clés priment alors cela permet de changer la correction de couleur ou la casse même pour les mots-clés PureBasic.
Les mots-clés peuvent être entrés directement dans les préférences ou spécifiés dans un fichier texte avec un mot-clé par ligne. (ou les deux)

Editeur - Pliage



Il est possible d'ajouter des mots clefs pour marquer le début et la fin d'un bloc repliable dans le code source. Un nombre illimité de mots clefs peuvent être saisis. Les mots qui sont dans les commentaires sont ignorés sauf si le mot clef employé a pour initiale le symbole commentaire (comme le mot clef par défaut " ;{").
Un mot clef ne peut pas contenir d'espace.

Editeur - Indentation



Ici il est possible de paramétrer comment l'éditeur gère l'indentation du code quand la touche "Entrée" est appuyée.

Aucune indentation

Appuyer sur la touche "Entrée" place toujours le curseur au début de la ligne suivante.

Mode bloc

La nouvelle ligne aura la même indentation que la ligne courante.

Sensible au mot-clé

Appuyer sur la touche "Entrée" corrigera l'indentation de la ligne courante et de la nouvelle ligne, en fonction du mot-clé présent sur ces lignes. Les règles pour effectuer cette correction sont spécifiées dans la liste des mots-clés. Ces règles s'appliquent aussi lors d'un "Reformatter sélection" du menu édition .

Afficher les indentations

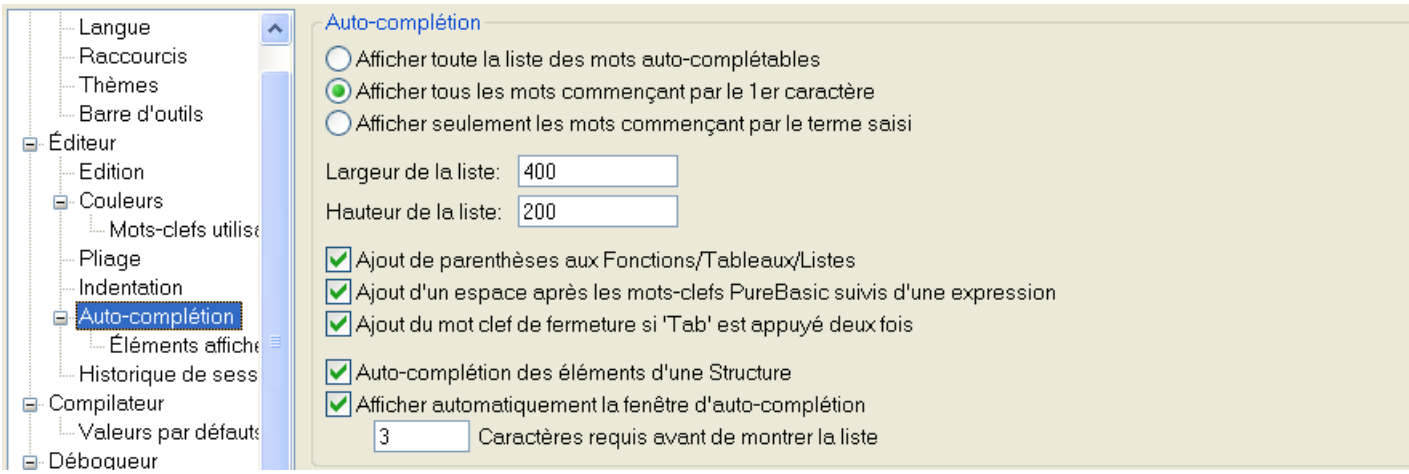
Affiche des lignes verticales pour visualiser les indentations de chaque ligne. Cela montre plus rapidement les lignes ayant le même niveau d'indentation.

Afficher les espaces

Affiche les caractères d'espacement sous forme de points (espaces) ou flèches (tabulations).

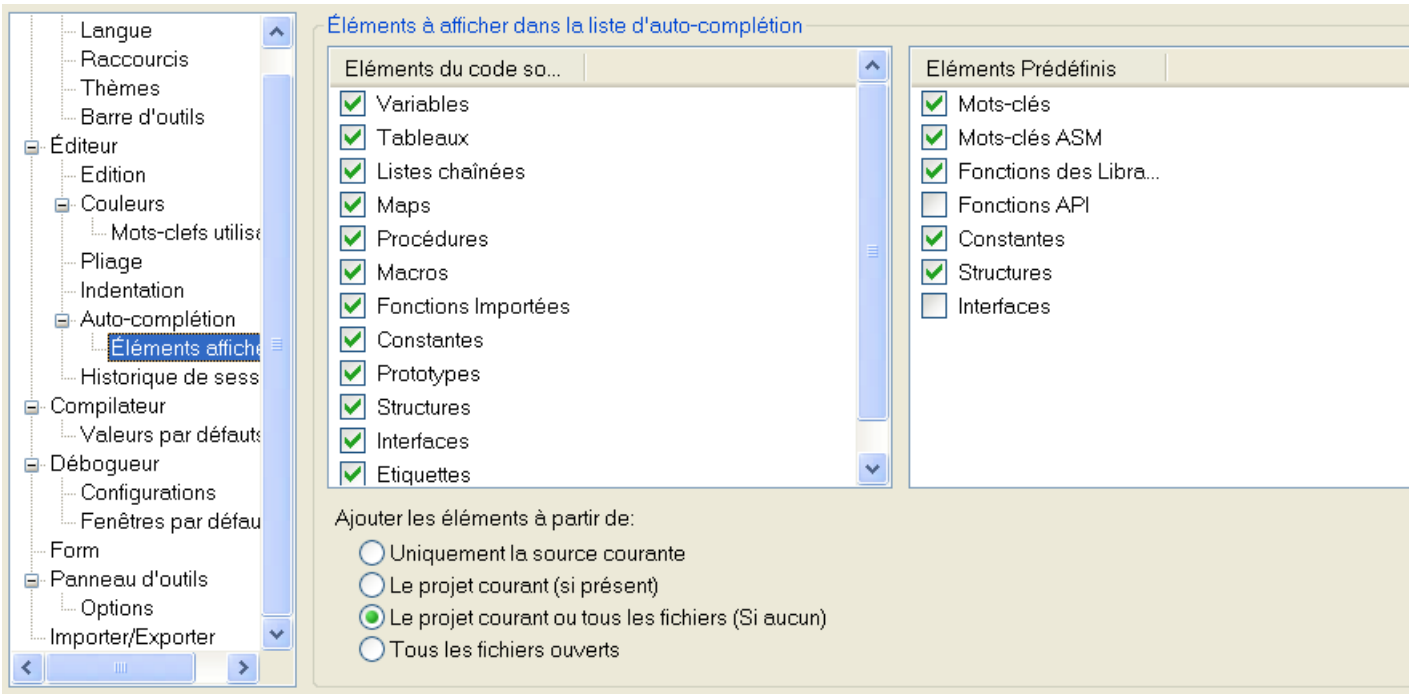
La liste des mots-clés contient les mot-clés qui influencent l'indentation. Le paramètre "Avant" indique le changement d'indentation sur le ligne qui contient le mot-clé, tandis que le paramètre "Après" spécfie le changement à appliquer sur la ligne suivante.

Editeur - Auto-complétion



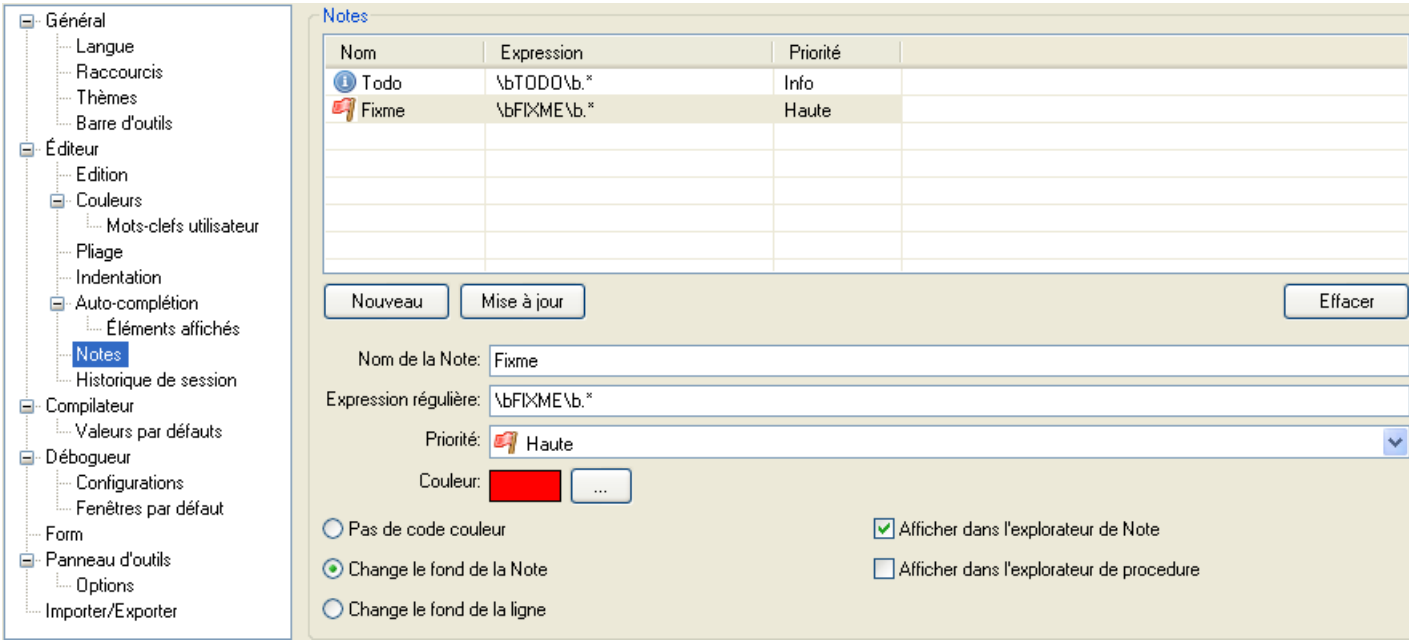
Afficher toute la liste des mots auto-complétables
Affiche toujours toute la liste des mots qui sont gérés par l'auto-complétion, mais choisit tout le même le mot le plus proche.
Afficher tous les mots commencant par le 1er caractère
Affiche uniquement les mots qui commencent par le premier caractère saisi. Le mot le plus proche est tout de même sélectionné.
Afficher seulement les mots commençant par le terme saisi
Affiche seulement les mots commençant par le terme saisi. Si aucun mot ne correspond, la liste ne sera pas affichée du tout.
Largeur de la liste / Hauteur de la liste
La dimension de la liste d'auto-complétion peut être définie ici (en pixels). Note : Ce sont les valeurs maximum. La fenêtre de la liste peut être plus petite s'il y a très peu d'éléments à afficher.
Ajout des parenthèses aux fonctions/tableaux/listes
Ajoute automatiquement la parenthèse ouvrante "(" après chaque fonction, tableau ou liste insérée par l'auto-complétion. Les fonctions sans paramètres et les listes chainées auront automatiquement les doubles parenthèses "()" ajoutées.
Ajout d'un espace après les mots-clefs PureBasic suivis d'une expression
Quand un mot clef PureBasic ne peut pas être utilisé tout seul (tel que If, For, While etc.) et nécessite une expression, un espace sera automatiquement ajouté lors de l'auto-complétion.
Ajout du mot clef de fermeture si Tabulation/Entrée est appuyée deux fois
Si la touche 'Entrée' ou 'Tabulation' est appuyée deux fois, le mot clef de fermeture sera automatiquement inséré (ex : "EndSelect" pour "Select", "EndIf " pour "If") après le mot clef lui-même (et après le curseur, dont il possible de continuer l'édition immédiatement après l'auto-complétion).
Auto-complétion des éléments d'une structure
Affiche automatiquement la liste d'auto-complétion quand le curseur est juste après une variable associée à une structure ou à une interface et que la touche "\" est entrée. La liste contiendra tous les champs qui composent la structure ou l'interface. Si cette option est désactivée, la liste peut toujours être affichée manuellement en utilisant le raccourci clavier associé (par défaut Ctrl+Espace).
Afficher automatiquement la fenêtre de complétion des fonctions
Affiche automatiquement la liste d'auto-complétion après qu'un nombre de caractères aient été entrés et qu'un ou plusieurs mots peuvent correspondre. Si cette option est désactivée, la liste peut toujours être affichée manuellement en utilisant le raccourci clavier associé (par défaut Ctrl+Espace).
Caractères requis avant l'affichage de la liste
Permet le configurer le nombre minimum de caractères saisis avant que la liste ne soit automatiquement affichée.

Editeur - Auto-complétion - Eléments affichés



Permet d'ajuster avec précision quels éléments seront affichés dans la liste d'auto-complétion.
Eléments du code source
Eléments définis dans le code source actuel, ou dans les autres sources ouverts (voir ci-dessous).
Eléments prédéfinis
Eléments qui sont prédéfinis par PureBasic, comme les mots-clés, les fonctions ou les constantes.
Ajouter les éléments : du code source courant seulement
Les éléments ajoutés dans la liste de complétion sont seulement récupérés du code source courant.
Ajouter les éléments : du projet courant (si présent)
Les éléments ajoutés dans la liste de complétion sont récupérés à partir du projet courant, s'il y en a un. Les autres codes sources du projet n'ont pas à être obligatoirement ouverts dans l'IDE pour que ce soit fonctionnel.
Ajouter les éléments : du projet courant ou de tous les fichiers (si aucun)
Les éléments ajoutés dans la liste de complétion sont récupérés à partir du projet courant, s'il y en a un. Si le code source courant n'appartient pas au projet courant, tous les éléments des codes sources ouverts seront pris en compte.
Ajouter les éléments : de tous les fichiers ouverts
Les éléments ajoutés dans la liste de complétion sont récupérés à partir de tous les fichiers ouverts dans l'IDE.

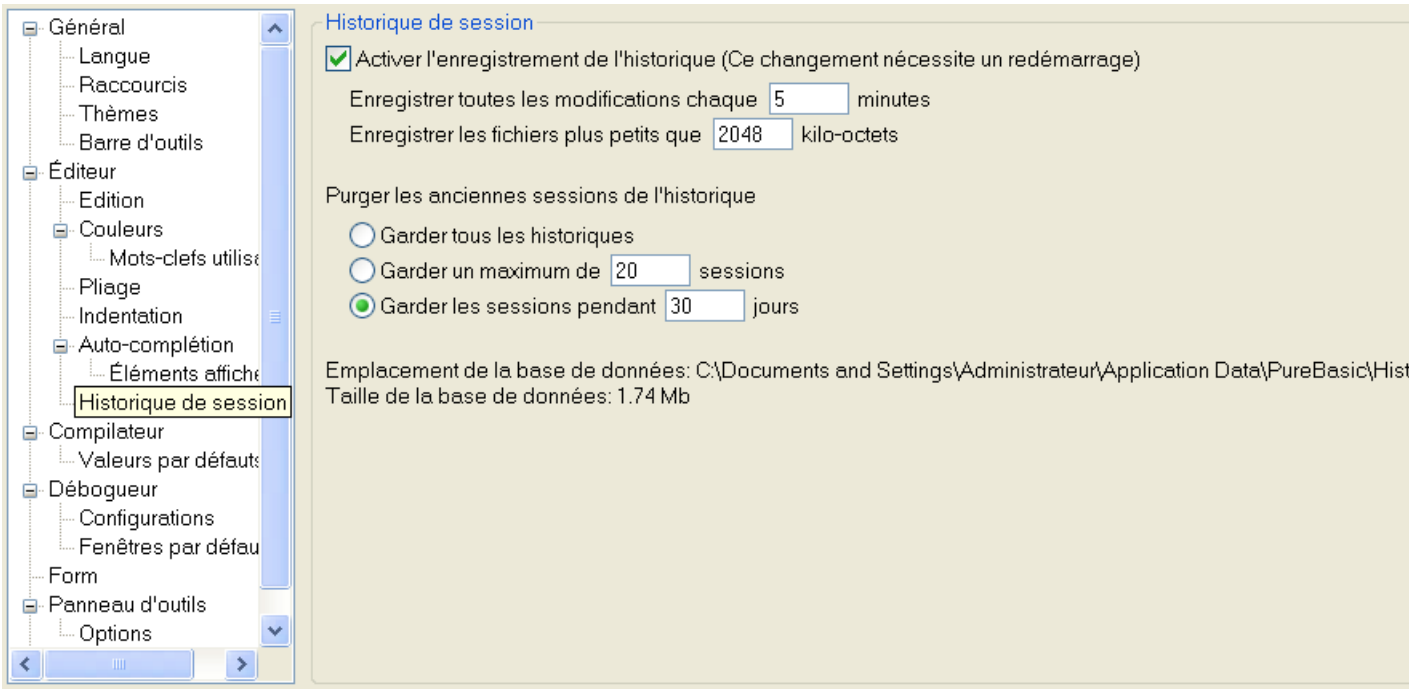
Editeur - Notes



Permet de configurer des marqueurs de 'Note' au niveau des commentaires du code source. Ces marqueurs peuvent être affichés dans les Notes du menu Outils ou dans l'outil de Explorateur de Procedure, et ils peuvent être marqués dans le code source avec un fond coloré.
La définition d'une Note :

Note
Un nom pour le type de Note.
Expression régulière
Une expression régulière définissant le motif de la Note. Cette expression régulière est appliquée à tous les commentaires dans le code source. Chaque match de l'expression est considéré en fonction du type de Note.
Priorité
A chaque type de Note est affecté une priorité. La priorité peut être utilisée pour ordonner et filtrer les Notes affichées.
Couleur
La couleur utilisée pour marquer la Note dans le code source (si activé). La couleur sera utilisée pour marquer le fond, soit du texte de la Note elle-même, soit de la ligne de code complète en fonction de l'option de coloration.
Afficher l'outil Note
Si activé, toutes les Notes trouvées sont répertoriées dans l'outil Note. Cette option peut être désactivée pour provoquer le marquage d'une Note avec une couleur de fond spéciale, si vous le souhaitez.
Afficher dans l'explorateur de procédure
Si activé, toutes les Notes trouvées sont présentées comme une entrée dans l'outil explorateur de procédure.

Editeur - Historique de Session



Permet de configurer la façon dont l'historique de la session enregistre les modifications.

Activer l'enregistrement de l'historique

Active ou désactive l'enregistrement de l'historique de la session. Lorsqu'il est activé, toutes les modifications apportées à un fichier seront enregistrées en tâche de fond, dans une base de données. Une session est créée lors du lancement de l'IDE, et est fermée lorsque l'IDE se ferme. Cette option est utile pour revenir à une version antérieure d'un fichier ou pour retrouver un fichier supprimé ou corrompu. C'est un peu comme un outil de sauvegarde des fichiers sources très puissant mais limité dans le temps (par défaut, un mois d'enregistrement). Il n'est pas destiné à remplacer un système de gestion des sources comme SVN ou GIT. Le code source sera stocké sans chiffrement, donc si vous travaillez sur code source sensible, assurez-vous d'avoir ce fichier de base de données dans un endroit sûr, ou de désactiver cette fonction. Il est possible de définir l'emplacement de la base de données en utilisant un commutateur en ligne de commande .

Enregistrer toutes les modifications chaque X minutes

Modifier l'intervalle entre chaque enregistrement automatique (lors de l'édition). Un fichier sera automatiquement enregistré lors de son enregistrement ou de sa fermeture.

Enregistrer les fichiers plus petits que X kilo-octets

Modifier la taille maximale (en kilo-octets) des fichiers en cours d'enregistrement. Ceci permet d'exclure les très gros fichiers qui pourraient rendre la base de données trop importante.

Garder tous les historiques

Gardez tout les historiques, la base de données n'est jamais purgée. Elle grossit toujours et devra donc être surveillée.

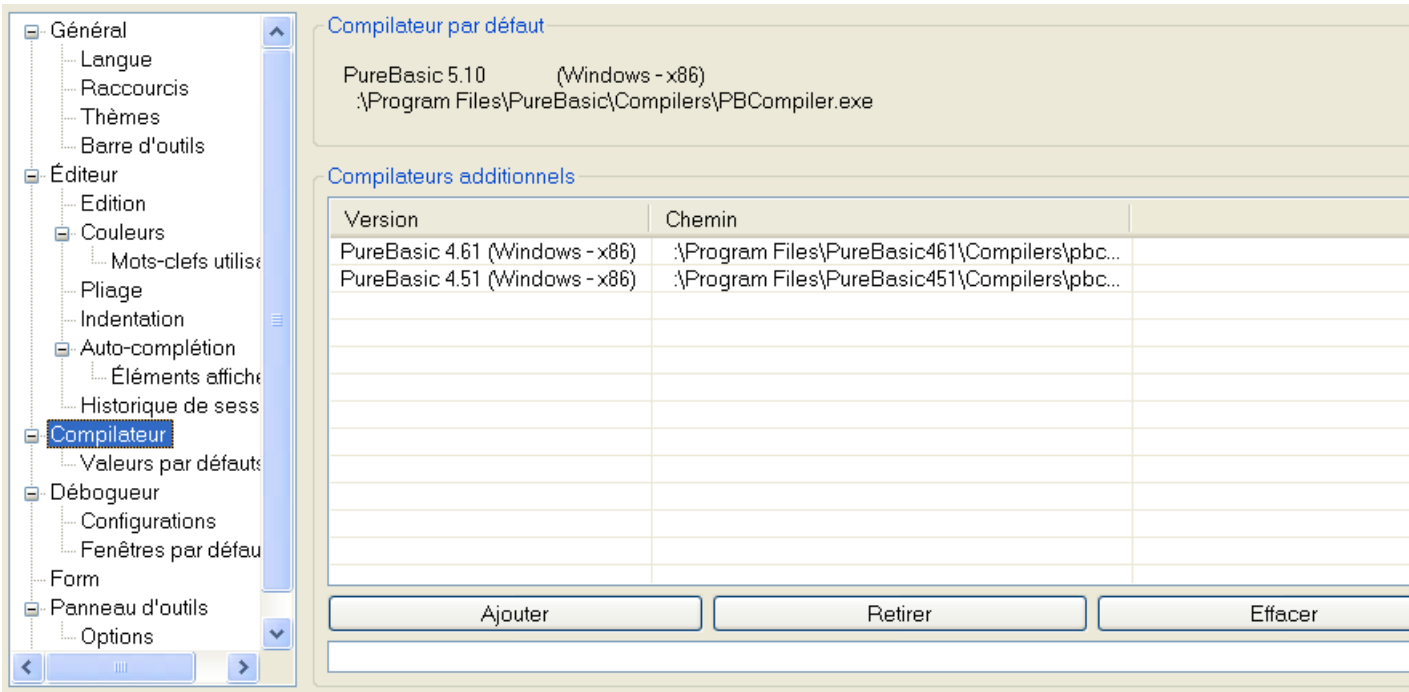
Garder un maximum de X sessions

Après avoir atteint le nombre maximal de sessions, la session la plus ancienne sera supprimée de la base de données.

Garder les sessions pendant X jours

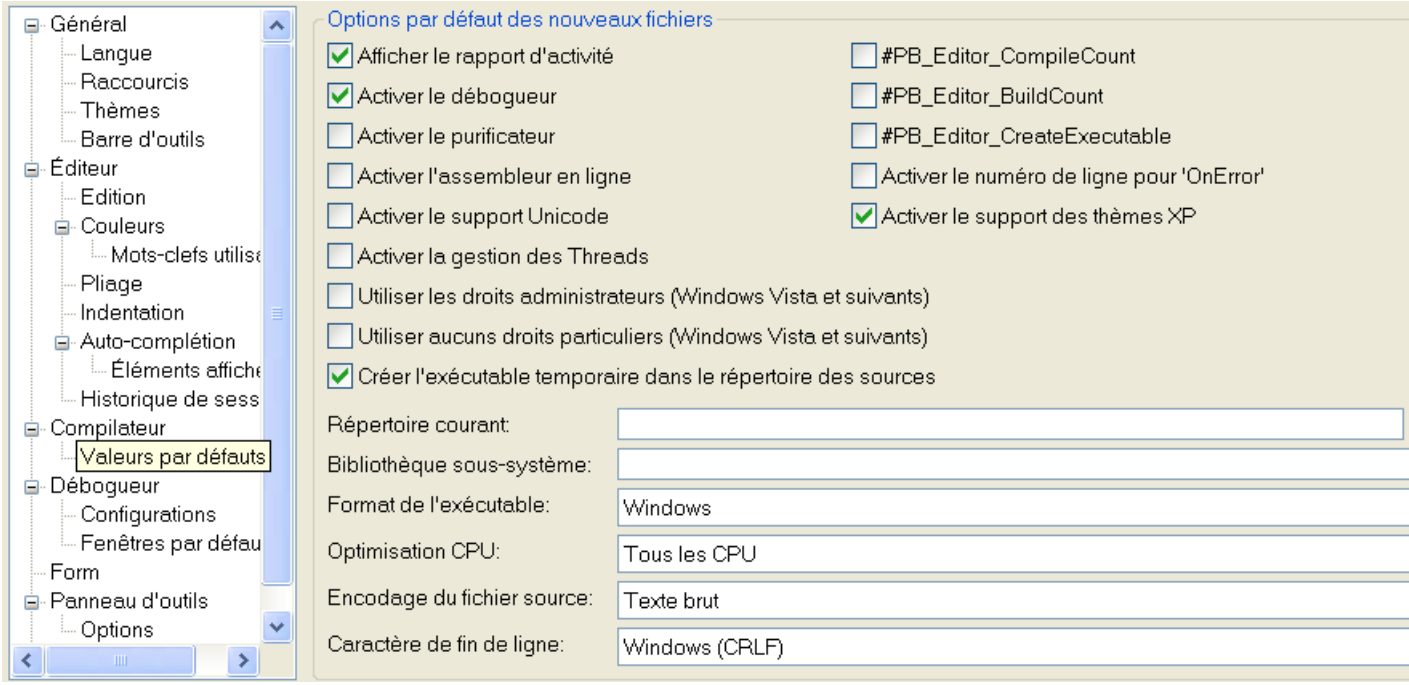
Après avoir atteint le nombre maximum de jours, la session sera supprimée de la base de données.

Compilateur



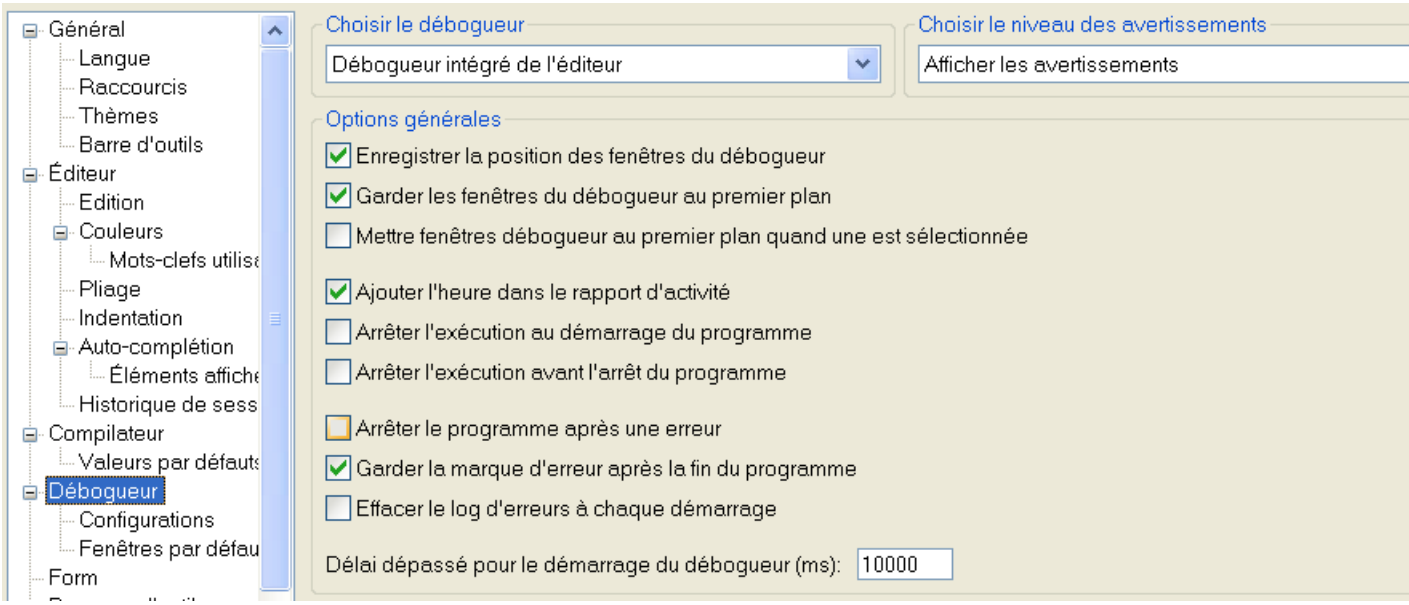
Cette section permet de configurer un compilateur additionnel qui sera ensuite disponible pour les options compilations . Ainsi, il est possible d'échanger rapidement le compilateur (x86 vers x64 par exemple) ou même d'utiliser d'anciens compilateurs directement dans le nouvel IDE.
N'importe quel compilateur à partir de la version 4.10 peut être ajouté ici. Le type de processeur supporté n'a pas à être le même que celui de l'IDE, tant que le système d'exploitation est identique. La liste affiche la version du compilateur et son chemin.
Les informations utilisées par l'IDE (pour la coloration syntaxique, l'autocomplétion, le visualisateur de structures) proviennent toujours du compilateur par défaut. Les compilateurs additionnels ne servent que pour la compilation.

Compilateur - Par Défauts



Cette page permet de régler les options par défaut qui seront utilisées lors de la création d'un nouveau fichier source à l'aide de l'IDE.
Pour une explication des paramètres, voir les options compilations .

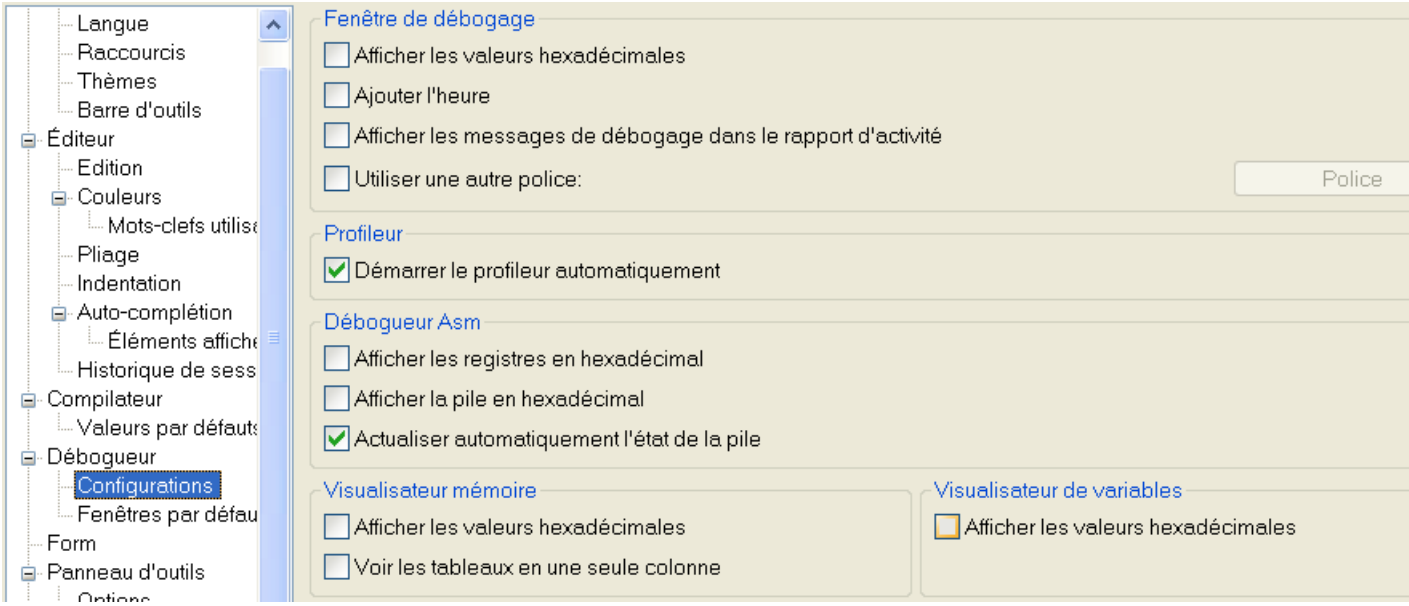
Débogueur



Permet de paramétrer le comportement du débogueur intégré ou indépendant. Le débogueur en ligne de commande ne se configure que par l'intermédiaire du terminal.
Type de débogueur
Sélectionne le type du débogueur à utiliser lorsqu'un programme est compilé dans l'IDE.
Enregistrer la position des fenêtres du débogueur
Même fonction que l'option "Mémoriser la position des fenêtres" de la section "Général", mais appliquée aux fenêtres du débogueur.
Garder les fenêtres du débogueur en premier plan
Toutes les fenêtres de débuggage seront toujours au premier plan, même devant les autres applications. Cela peut faciliter le débuggage des applications prenant toute la surface du bureau.
Mettre les fenêtres du débogueur au premier plan quand une est sélectionnée
Avec cette option activée, lorsqu'une fenêtre de débuggage est activée à l'aide de la souris ou du clavier, les autres fenêtres ouvertes sont automatiquement ramenées au premier plan.
Ajouter l'heure dans le rapport d'activité
Ajoute l'heure de chaque évènement dans le rapport d'activité.
Arrêter l'exécution au démarrage du programme
Chaque programme sera stoppé au début de son exécution, pour donner la possibilité de dérouler le programme pas à pas (sans avoir besoin de mettre un point d'arrêt au début de chaque source).
Arrêter l'exécution avant l'arrêt du programme
Chaque programme sera stoppé juste avant de quitter réellement. Cela permet par exemple d'utiliser les outils de débuggage pour examiner les variables ou la mémoire avant que le programme ne quitte.
Arrêter le programme après une erreur
Si le programme rencontre une erreur, il sera alors terminé automatiquement et toutes les fenêtres du débogueur seront fermées. Cela permet d'éditer le code immédiatement après une erreur, mais ce n'est plus possible d'examiner le programme pour essayer de déterminer la cause de l'erreur.
Garder les marqueurs d'erreurs après la fin du programme
N'efface pas les marqueurs d'erreurs quand le programme se termine. Cela permet de voir plus clairement ou l'erreur est survenue lors du retour à l'édition du code. Les marqueurs peuvent ensuite être enlevés à l'aide de la commande "Effacer les marqueurs d'erreurs" du sous-menu "Rapport d'activité".
Effacer le rapport d'activité à chaque exécution
Efface le rapport d'activité à chaque fois qu'un programme est exécuté. Cela assure que le rapport ne deviendra pas trop conséquent (cette option est aussi disponible quand le débogueur en ligne de

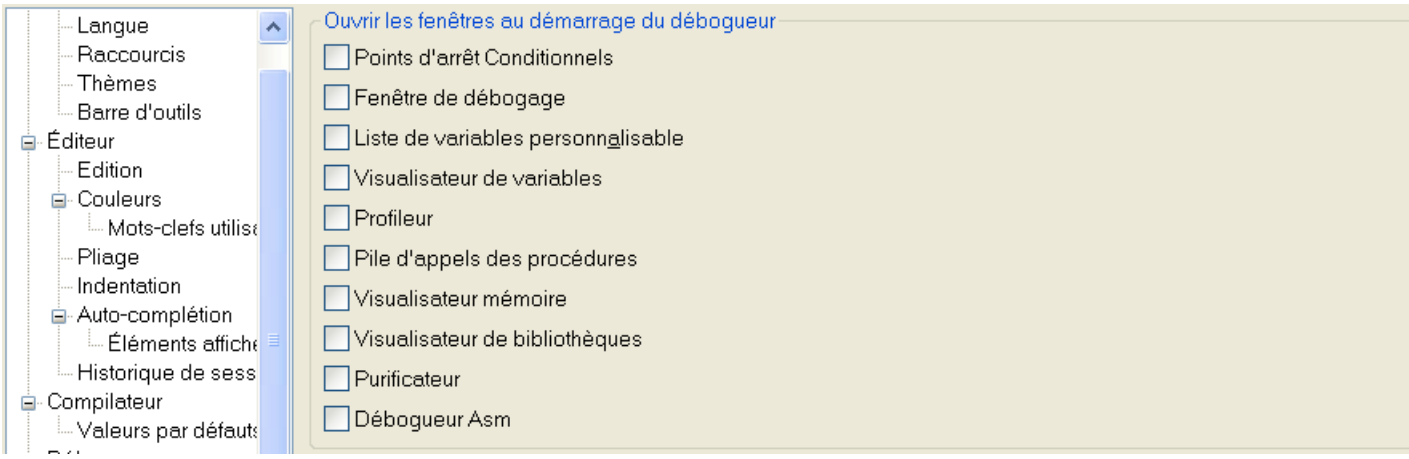
commande est sélectionné).
Delai maximum pour le lancement du débogueur
Indique le temps maximum, en millisecondes, qui sera donné au programme pour s'initialiser. Ce delai évite que le débogueur ne bloque indéfiniment si l'exécutable à déboguer ne peut pas se lancer, pour une raison ou pour une autre.

Débogueur - Configuration des outils



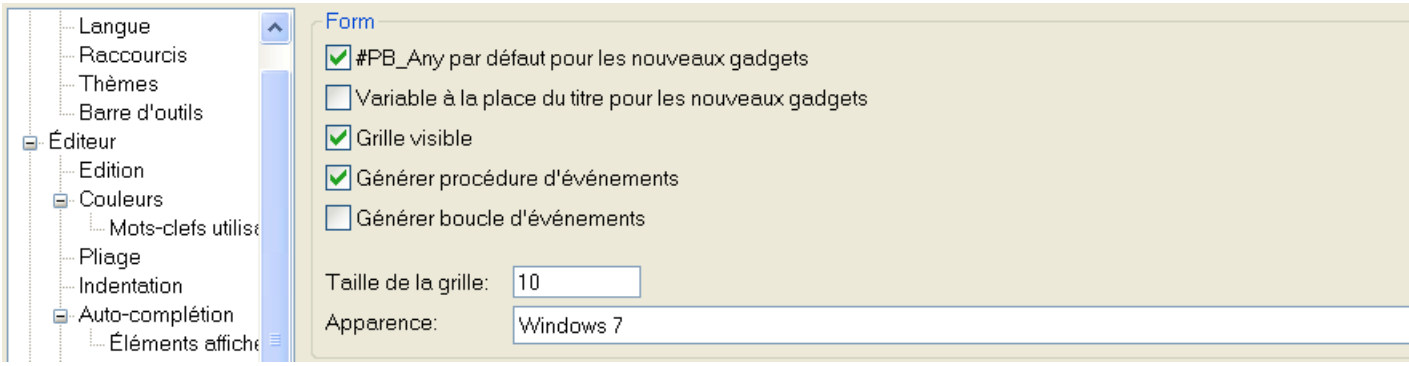
Cela permet de contrôler l'affichage et le comportement des outils intégrés au débogueur. Les options "Afficher les valeurs en hexadecimal" change l'affichage des octets, word ou long du format décimal (base 10) au format hexadécimal (base 16).
Fenêtre de débogage Ajouter l'heure
Ajoute l'heure à chaque ligne affichée dans la fenêtre de débogage.
Fenêtre de débogage - Affiche les messages de débogage dans le rapport d'activité
Avec cette option activée, une commande Debug dans le code n'ouvrira pas la fenêtre du débogueur , mais affichera les messages de débogage dans le rapport d'activité .
Fenêtre de débogage Utilise une autre police
Une police personnalisée peut être sélectionnée pour l'affichage dans la fenêtre de débogage. Ce qui permet de choisir une petite police pour afficher plus de données ou de choisir une police avec une taille proportionnelle quand c'est nécessaire.
Profileur - Lancer le profileur au démarrage du programme
Détermine si le profileur commencera l'enregistrement des données au démarrage du programme.
Débogueur ASM Actualiser automatiquement l'état de la pile
Met à jour automatiquement l'état de la pile à chaque "Pas" ou "Stop" effectué. Si cette option est désactivée, un bouton "Actualiser" sera affiché dans la fenêtre ASM.
Observateur de mémoire Voir les tableaux en une seule colonne
Si la zone de mémoire est affichée en mode "tableau", cette option indique si ce tableau sera multi-colonnes (avec 16 octets par colonne) ou seulement avec une colonne.

Débogueur - Fenêtres par défaut



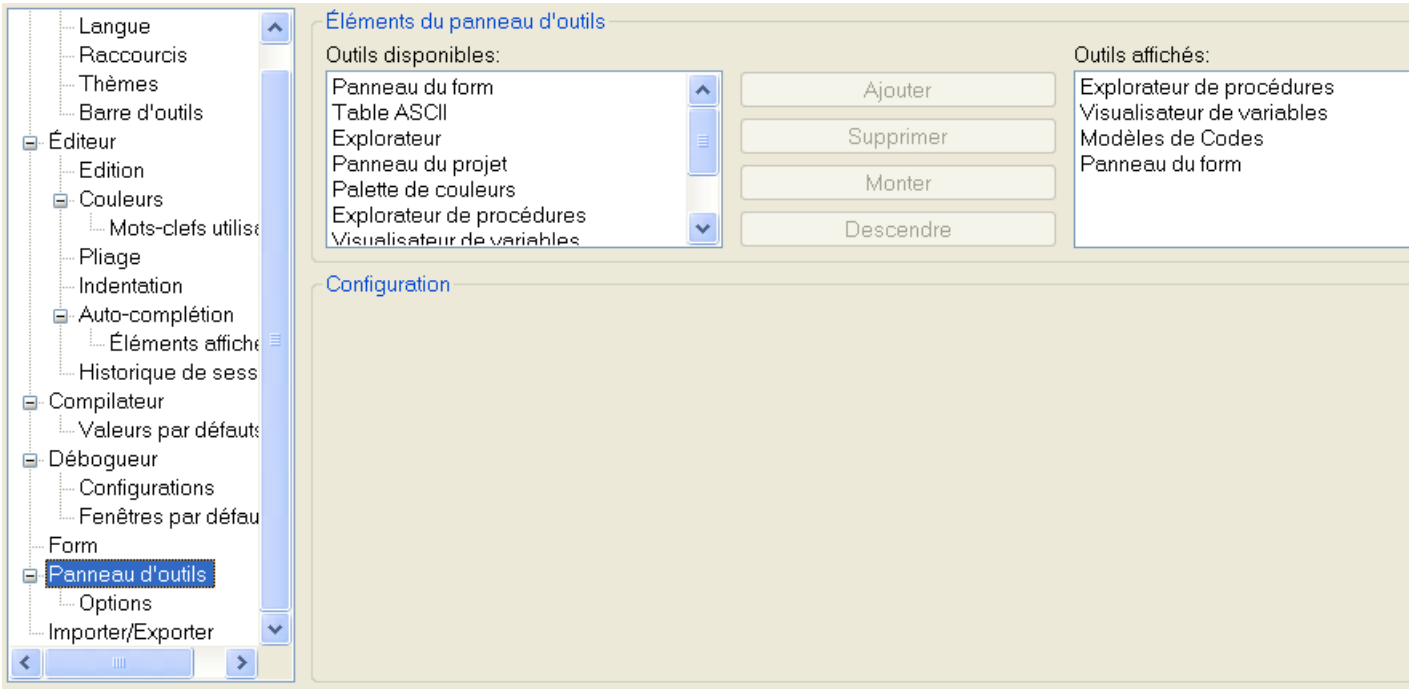
Les fenêtres cochées sur cette page seront ouvertes automatiquement à chaque fois qu'un programme sera lancé avec le débogueur.

Form



Permet de personnaliser le comportement du concepteur de fenêtre (Form) intégré. **#PB_Any par défaut pour les nouveaux gadgets**
Si elle est activée, la création du nouveau gadget utilisera '#PB_Any' au lieu d'une numéro.
Variable à la place du titre pour les nouveaux gadgets
Si elle est activée, le nouveau gadget utilisera une variable au lieu d'un texte de titre prédéfini. Cela peut être utile pour localiser facilement la fenêtre.
Grille visible
Si elle est activée, la grille sera visible sur le concepteur de fenêtres, afin de faciliter l'alignement des gadgets.
Générer procédure d'événements
Si elle est activée, une procédure d'événement sera générée automatiquement (et à mise jour).
Générer boucle d'événements
Si elle est activée, une boucle d'événement de base est générée pour le form.
Taille de la grille
L'espace entre les deux points de la grille, en pixels.
Apparence
Habillage à utiliser pour le concepteur de forms.

Panneau d'outils



Gère la configuration des outils internes qui peuvent être affichés dans le panneau latéral. Chaque outil qui est dans la liste des "Outils affichés" sera affiché dans la palette d'outils. Les outils qui ne sont pas présents dans cette liste sont accessibles à partir du menu "Outils". Ils seront ouverts dans une fenêtre indépendante.
Il est préférable de ne mettre que les outils qui sont utilisés très fréquemment dans la palette d'outils, et de mettre le plus utilisé en premier, car c'est celui qui est affiché le premier lorsque l'IDE démarre.
En sélectionnant un outil dans l'une des deux listes, un panneau de configuration spécifique à cet outil (si il y en a un) apparaîtra dans la partie "Configuration".
Les outils suivants peuvent être configurés :

Exploreur

Il est possible de choisir entre un affichage en mode liste ou en mode hierarchique (arbre) et de mémoriser ou non le dernier répertoire affiché (s'il n'est pas mémorisé, alors le "Répertoire par défaut" défini dans les options "Général" sera utilisé)

Navigateur de procédures

"Trier les procédures par nom" : trie la liste des procédures par ordre alphabétique (par défaut, les procédures apparaissent dans l'ordre de leurs déclarations)
"Grouper les marqueurs" : groupe les marqueurs (";-") entre eux.
"Afficher les arguments des procédures" : Affiche le nom de la procédure et tous ses paramètres.

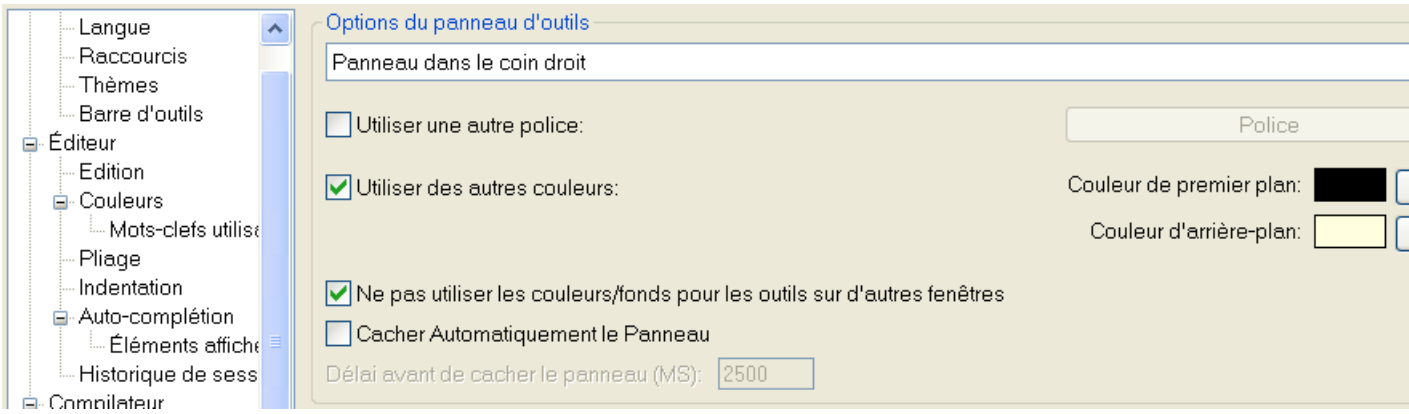
Visualisateur de variables

L'option "Afficher les éléments de tous les fichiers ouverts" détermine si la liste doit contenir les éléments du fichier en cours d'édition, ou de tous les fichiers actuellement ouverts par l'IDE.
De plus, il est possible de sélectionner le type d'éléments qui seront affichés dans le visualisateur de variables.

Outils Aide

Appuyer sur F1 ouvre l'aide dans le panneau : Spécifie s'il faut ouvrir l'outil d'aide au lieu du visualisateur d'aide lorsque F1 est pressé.

Panneau d'outils - Options



L'apparence de la palette d'outil en elle même peut être configurée dans cette section : le côté qui sera utilisé pour afficher la palette (à droite ou à gauche de la zone d'édition), la police de caractères, ainsi que la couleur de fond et d'avant plan utilisée par les outils affichés. La police et les couleurs optionnelles peuvent être désactivées pour utiliser les valeurs par défaut de l'OS.

Ignorer la couleur/police dans les fenêtres indépendantes

Si cette option est activée, les options de couleur/police ne seront appliquées que si l'outil est affiché dans la palette d'outils (ceux ouverts via le menu "Outils" utiliseront la couleur/police par défaut).

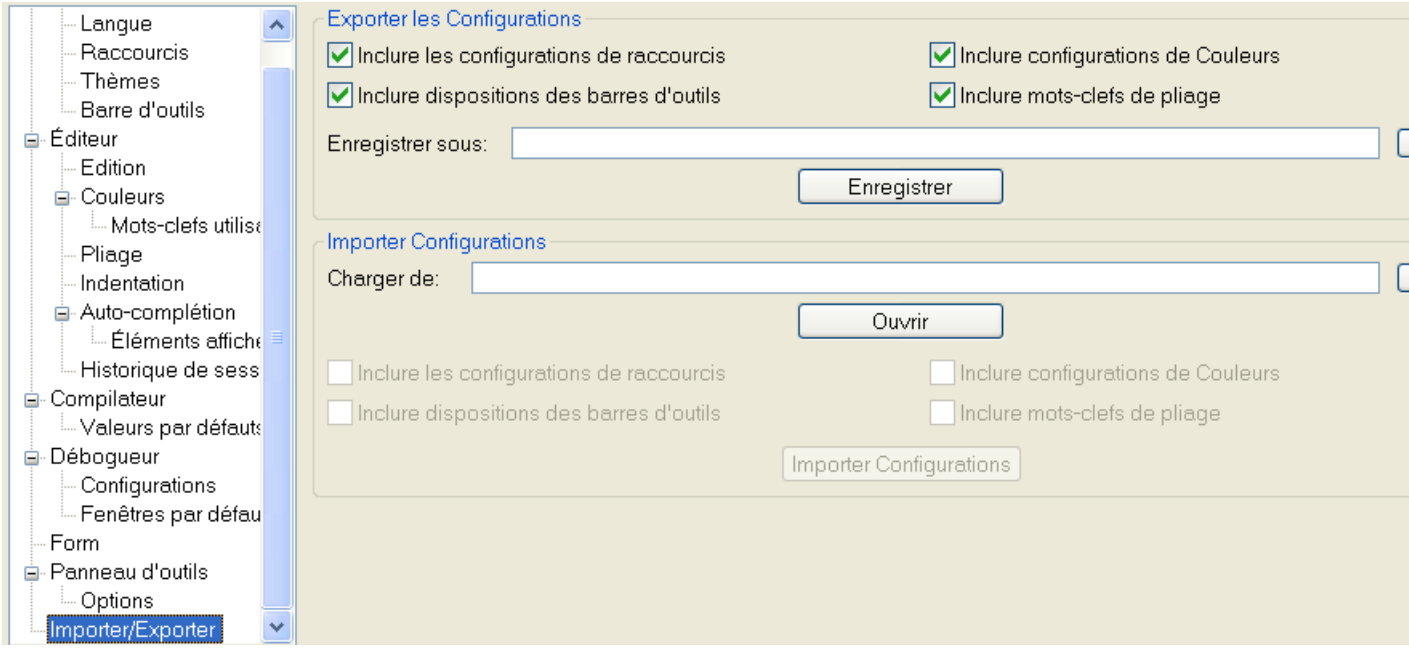
Cacher automatiquement la palette d'outils

Pour gagner un peu de place, la palette d'outils peut se cacher automatiquement si la souris ne se trouve pas au dessus d'elle. Pour la faire réapparaître, il suffit de déplacer la souris sur la zone située sur le côté de la zone d'édition.

Délai en millisecondes avant de cacher la palette

Défini un délai (en ms), après lequel la palette d'outils sera cachée si la souris n'est plus au dessus d'elle.

Importer/Exporter



Il est possible d'importer et d'exporter la configuration de l'IDE dans un format indépendant, et de l'importer dans un autre IDE (sous un autre OS par exemple, ou pour le partager avec d'autres utilisateurs).

Pour réaliser une exportation, il faut sélectionner le type des paramètres à exporter, saisir un nom de fichier et cliquer sur le bouton "Enregistrer"

Pour importer des paramètres, il suffit de choisir le nom du fichier et de cliquer sur ”Ouvrir”. Les paramètres disponibles seront alors affichés, il faudra décocher les paramètres indésirables puis cliquer sur ”Importer”.
Pour que les nouveaux paramètres prennent effet, il faudra appuyer sur le bouton ”Appliquer”.
Note : il est possible d'importer les fichiers 'styles' de l'éditeur jaPBe, mais seuls les paramètres concernant les couleurs seront importés.

Chapitre 19

Commutateurs de la ligne de commande

L'IDE de PureBasic permet de modifier les chemins et les fichiers utilisés lors de son lancement. Cela permet de créer plusieurs raccourcis qui vont lancer l'IDE avec une configuration particulière pour chaque utilisateur, ou pour chaque projet.
Il existe également des options pour la compilation des projets PureBasic directement en ligne de commande. Construire un projet à partir de la ligne de commande comporte les mêmes actions, comme le choix de la 'Cible' ou "Créer toutes les cibles" du menu Compilateur .
Options générales :

```
/VERSION                  affiche la version de l'IDE
/HELP ou /?               affiche une description des arguments de la
 ligne de commande

/P <fichier Préférences>             charge (et enregistre) la
 configuration à partir du fichier spécifié.
/T <fichier Modèle>                  charge (et enregistre) les
 modèles de code à partir du fichier spécifié.
/A <fichier outils>                  charge (et enregistre) la
 configuration des outils externes à partir du fichier spécifié.
/S <Chemin Source>                   change l'option "Répertoire
 principal" des préférences.
/E <Chemin Explorateur>              démarre l'outil 'Explorateur'
 avec le chemin spécifié.
/L <Numéro ligne>                    déplace le curseur à la ligne
 indiquée (seulement dans le dernier fichier chargé).
/H <Base de données de l'Historique> le fichier à utiliser pour la
 base de données de l'historique de la session.
/NOEXT                               désactive l'association
 automatique des fichiers '.pb' dans la base de registre.
/LOCAL                               place tous les fichiers
 préférences dans le répertoire de PureBasic au lieu du répertoire
 personnel de l'utilisateur.
/PORTABLE                            combinaison de /LOCAL et de
 /NOEXT
```

Options pour compiler les projets :

```
/BUILD <file>             spécifie le fichier de projet à construire
/TARGET <target>          spécifie la cible à construire (la valeur
 par défaut est de construire toutes les cibles)
/QUIET                    cache tous les messages lors de la
 construction sauf les erreurs
/READONLY                 ne met pas à jour le fichier de projet après
 la compilation (ni nouvelle date d'accès et ni nouveaux compteurs de
```

```
    construction )
```

Par défaut, les fichiers paramètres utilisés avec /P /T et /A se trouvent dans le répertoire %APPDATA%\PureBasic\.

```
  Le paramètre /NOEXT est particulièrement utile lorque plusieurs
   versions de PureBasic
  cohabitent sur un même système (pour tester les versions bêta par
   exemple). Ainsi les
  fichiers .pb seront toujours associés à la même version de PureBasic.
\\



  Le paramètre /PORTABLE peut être utilisé afin de garder l'ensemble de
   la configuration
  dans le répertoire local pour copier facilement PureBasic sur
   différents ordinateurs
  (ou le faire fonctionner depuis une clé USB par exemple).
```

Exemple :

```
1   PureBasic.exe Example.pb /PORTABLE
```

Il est aussi possible d'ouvrir des fichiers en ligne de commande, par exemple la commande : PureBasic.exe Exemple.pb, ouvrira le fichier Exemple.pb dans l'éditeur de PureBasic. Il est même possible de spécifier un masque en entrée, par exemple "*.pb" chargera tous les fichiers sources du répertoire.

Troisième partie

La langue

Chapitre 20

Travailler avec différentes bases numériques

(Note : Ces exemples utilisent le symbole ^ pour signifier 'élevé à la puissance de' - ceci est seulement une convention dans ce document, PureBasic n'a pas actuellement d'opérateur d'élévation à la puissance ! Utilisez plutôt la commande PureBasic Pow() de la bibliothèque "Math".)

Introduction

Une base numérique est une manière de représenter les nombres, en utilisant une certaine quantité de symboles possibles par chiffre. La plus courante que vous devez connaître est la base 10 (c-a-d décimale), car il y a 10 chiffres utilisés (0 à 9), et c'est celle que la plupart des humains peuvent manipuler le plus facilement.
Le but de cette page est d'expliquer différentes bases numériques et comment vous pouvez travailler avec elles. En effet les ordinateurs travaillent en binaire (base 2) et il y a certains cas où il est avantageux de le comprendre (par exemple, lors de l'utilisation d'opérateurs logiques, masques de bits, etc).

Vue d'ensemble des bases numériques

Le système Décimal

Pensez à un nombre décimal, et réfléchissez à sa représentation en colonnes. Prenons par exemple le nombre 1234. Séparé en colonnes, nous avons :

```
1   2   3   4
```

Réfléchissez à ce que sont les en-têtes de chaque colonne. Nous savons qu'il y a les unités, les dizaines, les centaines et les milliers, présentés ainsi :

```
1000  100   10    1
   1    2    3    4
```

Nous pouvons voir que le nombre 1234 est constitué de

```
   1*1000=1000
+ 2* 100= 200
+ 3*  10=  30
+ 4*   1=   4
Total   =1234
```

Si nous regardons également les en-têtes de colonne, vous verrez qu'à chaque fois que vous bougez d'une colonne vers la gauche nous multiplions par 10, qui justement s'avère être la base numérique. Chaque fois que vous bougez d'une colonne vers la droite, vous divisez par 10. Les en-têtes de colonnes peuvent être appelées les poids, puisque pour obtenir tout le nombre nous devons multiplier les chiffres de chaque colonne par le poids. Nous pouvons exprimer les poids en utilisant des index. Par exemple 10 ^2 signifie '10 élevé à la puissance de 2' ou 1*10*10 (=100). De même, 10 ^4 signifie 1*10*10*10*10 (=10000). Remarquez dans ces exemples, que peu importe la valeur de l'index, il correspond au nombre de fois que nous multiplions le nombre qui doit être élevé. 10 ^0 signifie 1 (puisque nous multiplions par 10 aucune fois). Utiliser des nombres négatifs montre que nous devons diviser, par exemple 10 ^-2 signifie 1/10/10 (=0.01). Les valeurs de l'index prennent plus de sens quand nous attribuons un numéro à chaque colonne - vous verrez souvent des choses comme 'bit 0' qui signifie en fait 'chiffre binaire en colonne 0'.

```
Dans cet exemple, ^ signifie élevé à la puissance de, donc 10^2
 signifie 10 élevé à la puissance de 2.
Numéro de colonne          3     2     1     0
Poids (index)           10^3  10^2  10^1  10^0
Poids (valeur réelle)   1000   100    10     1
Nombre exemple (1234)      1     2     3     4
```

Précédemment nous avons vu comment convertir le nombre 1234 dans son équivalent décimal. Conversion peu justifiée, car il était déjà en décimal mais nous pouvons en tirer la méthode générale - voici donc comment convertir n'importe quel nombre en sa valeur décimale :

```
B = Valeur de la base numérique

1) Séparer le nombre, peu importe la base, en colonnes. Par
   exemple, si nous avions la valeur 'abcde' dans notre base
 numérique fictive 'B',
   les colonnes seraient : a   b   c   d   e

2) Multiplier chaque symbole par le poids de chaque colonne (le poids
 étant
   calculé par 'B' élevé à la puissance du numéro de la colonne):
     a * B^4 = a * B * B * B * B
     b * B^3 = b * B * B * B
     c * B^2 = c * B * B
     d * B^1 = d * B
     e * B^0 = e

3) Calculer la somme de toutes ces valeurs. En écrivant toutes ces
 valeurs dans leur
   équivalent décimal pendant les calculs, il devient beaucoup plus
 facile de voir le
   résultat and de faire le calcul (si nous convertissons en décimal).
```

Convertir dans la direction opposée (de décimal vers la base numérique 'B') s'effectue en utilisant la division au lieu de la multiplication :

```
1) Commencer par le nombre décimal que vous voulez convertir (e.g.
 1234).

2) Diviser par la base numérique ciblée ('B') et prendre note du
 quotient et du
   reste.

3) Diviser le quotient de (2) par la base numérique ciblée ('B') et
 prendre note du
   quotient et du reste.

4) Continuer à diviser comme ça jusqu'à ce que vous obteniez un
 quotient de 0.
```

```
5) Votre nombre dans la base numérique ciblée est constitué des
 restes écrits dans l'ordre
   du dernier calculé au premier calculé. Par exemple, votre nombre
 serait constitué
   des restes des étapes dans cet ordre 432.
```

Des exemples plus particuliers seront donnés dans les paragraphes concernant les bases numériques particulières.

Système binaire

Tout dans un ordinateur est stocké en binaire (base 2, avec les symboles définis '0' ou '1') mais le travail avec des nombres binaires suit les mêmes règles qu'en décimal. Chaque symbole dans un nombre binaire est appelé bit, abréviation de BInary digiT (chiffre binaire). Généralement, vous travaillerez avec des bytes (octets, 8-bit), words (mots, 16-bit) ou longwords (mots longs, 32-bit) car ce sont les tailles par défaut des types intégrés à PureBasic. Les poids pour un byte (octet) sont les suivants :

```
(^ signifie 'élevé à la puissance de', la base numérique est 2 pour
 le binaire)
Bit/numéro de colonne      7     6     5     4     3     2     1     0
Poids (index)            2^7   2^6   2^5   2^4   2^3   2^2   2^1   2^0
Poids (valeur réelle)    128    64    32    16     8     4     2     1
```

Donc, par exemple, si nous avions le nombre 00110011 (Base 2), nous pourrions établir sa valeur comme ceci :

```
  0 * 128
+ 0 *  64
+ 1 *  32
+ 1 *  16
+ 0 *   8
+ 0 *   4
+ 1 *   2
+ 1 *   1
=      51
```

Un exemple de conversion inversée serait d'écrire la valeur 69 en binaire. Nous ferions comme ceci :

```
69 / 2 = 34 r 1     ^
34 / 2 = 17 r 0    /|\
17 / 2 =  8 r 1     |
 8 / 2 =  4 r 0     |    Lire les restes dans cette direction
 4 / 2 =  2 r 0     |
 2 / 2 =  1 r 0     |
 1 / 2 =  0 r 1     |
(Arrêt ici car le quotient de la dernière division était 0)

Lire les restes de bas en haut pour obtenir la valeur en binaire =
  1000101
```

Une autre chose à considérer lorsque l'on travaille avec des nombres binaires est la représentation des nombres négatifs. Comme nous le faisons quotidiennement, nous pourrions simplement mettre un symbole négatif devant le nombre décimal. Nous ne pouvons pas faire cela en binaire, mais il y a un moyen (après tout, PureBasic travaille principalement avec des nombres signés, et doit donc être capable de gérer les nombres négatifs). Cette méthode est appelée 'complément à deux' et indépendamment de toutes les bonnes fonctionnalités que cette méthode possède (et qui ne seront pas expliquées ici, pour épargner une certaine confusion) la manière la plus simple de la comprendre, est de se dire que le poids

du bit le plus significatif (Most Significant bit / le MSb est le numéro du bit avec la plus grande valeur. Dans le cas d'un octet (byte), ce serait le bit 7) est réellement une valeur négative. Donc pour un système de complément à deux, les poids des bits sont modifés en :

```
(^ signifie 'élevé à la puissance de', la base numérique est 2 pour
 le binaire)
Bit/numéro de colonne      7     6     5     4     3     2     1     0
Poids (index)           -2^7   2^6   2^5   2^4   2^3   2^2   2^1   2^0
Poids (valeur réelle)   -128    64    32    16     8     4     2     1
```

et vous pourrez faire la conversion de binaire vers décimal exactement de la même manière que ci-dessus, mais en utilisant le nouveau jeu de poids. Par exemple, le nombre 10000000 (Base 2) est -128, et 10101010 (Base 2) est -86.
Pour convertir un nombre binaire positif en négatif et vice versa, vous inversez tous les bits et ensuite ajoutez 1. Par exemple, 00100010 passerait en négatif en inversant -> 11011101 et additionant 1 -> 11011110.
Ceci facilite la conversion de décimal à binaire, car vous pouvez convertir les nombres décimaux négatifs comme leurs équivalents positifs (en utilisant la méthode ci-dessus) et ensuite rendre le nombre binaire négatif à la fin.
Les nombre binaires dans PureBasic sont précédés du symbol 'pourcentage', et évidement tous les bits du nombre doivent être un '0' ou un '1'. Par exemple, vous pouvez utiliser la valeur %110011 dans PureBasic pour signifier 51. Notez que vous n'avez pas besoin de mettre les '0' de tête (ce nombre serait vraiment %00110011) mais cela peut faciliter la lisibilité de votre source si vous écrivez l'ensemble des bits.

Le système Hexadécimal

L'hexadecimal (pour base 16, symboles '0'-'9' suivis de 'A'-'F') est une base numérique qui est plus généralement employée en informatique, car il s'agit probablement de la base numérique non décimale la plus facile à comprendre pour les humains, et vous n'avez pas à écrire de longues chaines de symboles pour vos nombres (comme c'est le cas avec le binaire).
Les mathématiques hexadécimales suivent les mêmes règles qu'en décimal, bien que vous ayez maintenant 16 symboles par colonne jusqu'à ce que vous ayez besoin d'une retenue. La conversion entre l'hexadécimal et le décimal suit les mêmes principes qu'entre le binaire et le décimal, sauf que les poids sont des multiples de 16 et les divisions sont faites par 16 au lieu de 2 :

```
Numéro de colonne          3      2      1      0
Poids (index)           16^3   16^2   16^1   16^0
Poids (valeur réelle)   4096    256     16      1
```

Convertir la valeur hexadécimale BEEF (Base 16) en décimal serait fait comme ceci :

```
  B * 4096 = 11 * 4096
+ E *  256 = 14 *  256
+ E *   16 = 14 *   16
+ F *    1 = 15 *    1
=               48879
```

Et convertir la valeur 666 en hexadécimal serait fait comme ceci :

```
666 / 16 = 41 r 10    ^
 41 / 16 =  2 r  9   /|\    Lire les chiffres dans cette direction,
 se souvenir de convertir
  2 / 16 =  0 r  2    |     en chiffres hexadecimaux où nécessaire
(Arrêt ici car le quotient de la dernière division était 0)
La valeur hexadécimale de 666 est 29A
```

La chose vraiment intéressante avec l'hexadécimal c'est qu'il permet également de convertir vers le binaire très facilement. Chaque chiffre hexadécimal représente 4 bits, pour convertir entre hexadécimal et binaire, vous devez juste convertir chaque chiffre hexadécimal en 4 bits ou chaque groupe de 4 bits vers

un chiffre hexadécimal (en se rappelant que 4 est un diviseur de toutes les tailles communes de nombres binaires dans un CPU). Par exemple :

```
Nombre hexadécimal      5         9         D         F            4E
Valeur binaire       0101      1001      1101      1111        01001110
```

Lorsque vous utilisez des valeurs hexadécimales dans PureBasic, vous mettez un symbole 'dollar' devant le nombre, par exemple $BEEF.

Les conversions Hexadécimal <-> Binaire

Il est aisé de convertir un nombre hexadécimal en un nombre binaire et inversement, en utilisant les 'nibles'. Un nible est un groupe de 4 bits, appelé demi octet ou quartet. Un quartet contenant 4 bits, il peut donc prendre seize (2 ^4) valeurs différentes et correspond donc à un chiffre en hexadécimal. Deux chiffres hexadécimaux formant un octet, ce dernier est souvent représenté par deux nibbles.
Par exemple :

```
Numéro de colonne          3       2       1       0
Poids (index)             2^3     2^2     2^1     2^0
Poids (valeur réelle)       8       4       2       1
                           -------------------------
                  $7        0       1       1       1  : 0*8 + 1*4 + 1*2
 + 1*1 = 7
                  $C        1       1       0       0  : 1*8 + 1*4 = 12 =
 $C
```

Comment s'écrit $F2 en binaire ?

```
Poids (valeur réelle)       8   4   2   1 |  8   4   2   1
                           -------------|-------------
                  $F2       1   1   1   1    0   0   1   0 : 1*8 + 1*4 + 1*2
 + 1*1 = 15 = $F
                                                          et 0*8 + 0*4 + 1*2
 + 0*1 =  2 = $2
```

Facile non !

Chapitre 21

Break : Continue

Syntax

```
Break [Niveau]
```

Description

Break permet de quitter à n'importe quel moment une ou plusieurs des boucles suivantes : Repeat , For , ForEach et While .

Arguments

Niveau (optionnel) Indique le nombre de boucles qui doivent être abrégées.
Sans paramètre, Break quitte la boucle en cours.

Valeur de retour

Aucune.

Exemple : Boucle simple

```
1  For k=0 To 10
2    If k=5
3      Break  ; Quitte immédiatement la boucle For/Next
4    EndIf
5    Debug k
6  Next
```

Exemple : Boucles imbriquées

```
1  For k=0 To 10
2    Counter = 0
3    Repeat
4      If k=5
5        Break 2 ; Quitte immédiatement les boucles Repeat/Until et
   For/Next
```

```
      EndIf
      Counter+1
    Until Counter > 1
    Debug k
  Next
```

Syntax

```
Continue
```

Description

Continue permet de passer directement à la prochaine itération dans l'une des boucles suivantes : Repeat , For , ForEach et While .

Arguments

Aucun.

Valeur de retour

Aucune.

Exemple

```
For k=0 To 10
  If k=5
    Continue  ; N'affichera pas 'Debug 5' car la fin de cette itération sera ignorée
  EndIf
  Debug k
Next
```

Chapitre 22

Utiliser le compilateur en ligne de commande

Introduction

Le compilateur est situé dans le sous-dossier 'compilers' du dossier PureBasic. La manière la plus simple d'y accéder est d'ajouter ce répertoire à la variable PATH, ce qui vous donnera un accès permanent à toutes les commandes de ce répertoire. Chaque commutateur de commande qui commence par '/' ne concerne que Windows.
Il existe deux compilateurs pour PureBasic :
- 'pbcompiler' qui génère du code assembleur (x86 et x64).
- 'pbcompilerc' (ou 'pbcompiler' sur une plate-forme qui ne prend pas en charge le backend d'assemblage) qui génère du code C.
Le compilateur backend C est disponible sur toutes les plateformes prises en charge, tandis que le compilateur backend ASM n'est pris en charge que sur Windows (x86, x64), Linux (x86, x64) et OS X (x64).

Paramètres du compilateur (multi-plateforme)

-h, –help, / ? : affiche une aide simplifiée sur le compilateur.
-c, –commented, /COMMENTED : crée un fichier de sortie '.asm' commenté en même temps que l'exécutable. Ce fichier peut être ré-assemblé ultérieurement après l'avoir modifié selon vos besoins. Cette option est à destination des programmeurs avancés.
-d, –debugger, /DEBUGGER : active le débogueur.
-pf, –purifier, /PURIFIER : active le purifier. Le debugger doit être activé pour faire fonctionner le purifier.
-o, –output, /OUTPUT "Fichier" : crée un fichier exécutable ou une DLL indépendant ayant pour nom 'Fichier' et enregistré dans le chemin désiré. Sur MacOS X, il est possible de créer un ensemble d'applications en ajoutant ".app" dans le nom de l'exécutable. De cette façon, toute la structure de répertoire sera créé automatiquement.
-r, –resident, /RESIDENT "Fichier" : crée un fichier résident ayant pour nom 'Fichier'.
-i, –import, /IMPORT "Fichier" : crée un fichier d'importation au nom de fichier donné. Seulement un seul bloc Import/EndImport autorisé dans le fichier. Les fonctions importées seront chargées automatiquement pour tous les projets de PureBasic.
-l, –linenumbering, /LINENUMBERING : ajoute des informations concernant les lignes et les fichiers sources à l'exécutable, qui peut le ralentir considérablement. Ce qui permet à la bibliothèque OnError d'indiquer le fichier et le numéro de ligne en cas d'erreur.
-q, –quiet, /QUIET : désactive toutes les éditions de texte inutiles. Très pratique lorsque vous utilisez un éditeur tiers. -sb, –standby, /STANDBY : charge le compilateur en mémoire, en attente de commandes externes (éditeurs, scripts...). Plus d'informations sur l'utilisation de ce paramètre est disponible dans le

fichier 'CompilerInterface.txt' du répertoire 'SDK' de PureBasic.
-ir, –ignoreresident, /IGNORERESIDENT "Fichier" : ne charge pas le fichier résident au démarrage du compilateur. C'est utile pour mettre à jour un fichier résident déjà existant.
-co, –constant, /CONSTANT Nom=Valeur : crée la constante spécifiée avec la valeur indiquée.
Exemple : 'pbcompiler test.pb /CONSTANT MaConstante=10'. La constante #MaConstante sera créée automatiquement avec la valeur 10 avant de commencer la compilation.
-t, –thread, /THREAD : utilise les routines thread-safe pour toutes la gestion des chaînes de caractères et pour toutes les commandes.
-s, –subsystem, /SUBSYSTEM "Nom" : utilise le sous-système spécifié pour remplacer des commandes internes. Pour plus d'informations, voir sous-systèmes . -k, –check, /CHECK : vérifie la syntaxe uniquement, ne créer pas ni ne lance l'exécutable.
-z, –optimizer, /OPTIMIZER : Active l'optimiseur de code.
-pp, –preprocess, /PREPROCESS "Fichier" : Prétraite le code source et écrit la sortie dans le "Fichier" spécifié.
Le fichier traité est un fichier unique avec toutes les macro déployées, les directives de compilation sont prises en compte et le multiligne intégré. Cela permet une analyse plus facile d'un fichier source PureBasic, car tout est déployé et disponible dans un format de fichier plat 'flat'. Les commentaires ne sont pas inclus par défaut, sauf avec l'option /COMMENTED. Le fichier prétraité peut être recompilé comme n'importe quel autre fichier source PureBasic pour créer l'exécutable final.
-g, –language, /LANGUAGE \"language\" : utilise la langue spécifiée pour les messages d'erreur du compilateur.
-v, –version, /VERSION : Affiche la version du compilateur.
Exemples :

```
CLI> pbcompiler codesource.pb
```

Le compilateur compile le code 'codesource.pb' et l'exécute.

```
CLI> pbcompiler codesource.pb /DEBUGGER
```

Le compilateur compile le code 'codesource.pb' et l'exécute avec le débogueur actif.

Paramètres du compilateur spécifiques à Windows

/ICON "Icone.ico" : ajoute l'icône spécifiée dans l'exécutable.
/CONSOLE : le fichier de sortie est au format Console. Le format par défaut est Win32.
/DLL : le fichier de sortie est une DLL .
/XP : ajoute le support des thèmes ('skins') pour Windows XP.
/REASM : réassemble le fichier PureBasic.asm file en fichier exécutable. Ceci permet d'utiliser la fonction /COMMENTED, de modifier le fichier de sortie asm et de recréer un exécutable.
/MMX, /3DNOW, /SSE or /SSE2 : génère un exécutable spécifique à un type de processeur. Si une routine est disponible dans une version optimisée pour le type de processeur choisi, elle sera utilisée. Du coup, l'exécutable ne fonctionnera correctement que sur ce type de processeur.
/DYNAMICCPU : génère un exécutable qui contient toutes les versions des routines spécialement optimisées pour un type de processeur. Quand le programme se lance, il déterminera dynamiquement le type du processeur et choisira les routines les plus adaptées. Cela produit un exécutable plus gros mais plus rapide.
/RESOURCE "Fichier" : ajoute un fichier de resource Windows (.rc) à l'exécutable ou la DLL. Ce fichier ne doit pas être un fichier resource compilé, mais un fichier texte contenant des directives. Il est possible de ne specifier qu'un seul fichier resource, mais il peut en inclure d'autre si besoin est, à l'aide des directives adéquates.
/LINKER "FichierCommandes" : spécifie un fichier qui contient des commandes à passer directement au linker. Sur Windows, PureBasic utilise le linker de PellesC (polink), plus d'informations à propos des options possible sont disponible dans l'aide de PellesC.
Les deux options de compilation suivantes sont nécessaires pour créer des programmes fonctionnant sur Microsoft Vista ou au-dessus (Windows 7/8/10). Ces deux options sont en rapport avec le 'manifeste' inclus, donc elles sont ignorées avec les versions plus anciennes de Windows. Si aucune de ces deux options ne s ont utilisées, l'exe s'exécutera en tant qu'utilisateur normal, mais avec la virtualisation activée (redirection du registre et des fichiers). Il est recommandé d'utiliser le commutateur /USER pour

désactiver la virtualisation pour tous les programmes qui ont les privilèges d'utilisateur standard, car c'est seulement destinés à la compatibilité des programmes plus anciens. Ces options sont également disponibles dans le menu Compilateur/Options du compilateur de l'IDE.
/ADMINISTRATOR : oblige le programme à demander des privilèges d'administrateur au démarrage. Le programme ne fonctionnera pas sans ça. Cette option est nécessaire. Note : Vous pouvez également déboguer les programmes avec cette option, mais seulement avec la version autonome du débogueur (car il doit fonctionner avec des droits élevés lui aussi).
/USER : le programme sera exécuté avec les droit de l'utilisateur qui l'a lancé. Virtualisation pour l'exe est désactivée.
/DPIAWARE : Activer le support DPI sur l'exécutable.
Exemples :

```
 CLI> pbcompiler "C:\Project\Source\DLLSource.pb" /EXE
"C:\Projet\projet.dll" /DLL
```

Le compilateur compile le code 'DLLSource.pb' (ici avec le chemin complet) et crée la DLL "projet.dll" à l'emplacement indiqué.

Paramètres du compilateur spécifiques à Linux

-so ou –sharedobject "fichier" : Créer une bibliothèque dynamique (objet partagé).
-h ou –help : affiche une aide simplifiée sur le compilateur.
-mmx, -3dnow, -sse ou -sse2 : génère un exécutable spécifique à un type de processeur. Si une routine est disponible dans une version optimisée pour le type de processeur choisi, elle sera utilisée. Du coup, l'exécutable ne fonctionnera correctement que sur ce type de processeur.
-dc ou –dynamiccpu : génère un exécutable qui contient toutes les versions des routines spécialement optimisées pour un type de processeur. Quand le programme se lance, il déterminera dynamiquement le type du processeur et choisira les routines les plus adaptées. Cela produit un exécutable plus gros mais plus rapide.

Paramètres du compilateur spécifiques à OSX

-dl ou –dylib "Fichier" : crée une bibliothèque dynamique (dylib).
-n ou –icon "Icone.icns" : ajoute une icon à l'application.
-f ou –front : met le processus à l'avant plan lors de son lancement.
-ibp ou –ignorebundlepath : ne pas utiliser le chemin du bundle comme répertoire courant.

Chapitre 23

Les directives du compilateur

Syntax

```
CompilerIf  <expression constante>
  ...
[CompilerElseIf]
  ...
[CompilerElse]
  ...
CompilerEndIf
```

Description

Si <expression constante> est vrai alors le code inclus dans la structure 'If' sera compilé, sinon il sera totalement ignoré.
Cette directive permet de construire des programmes multi-plateformes en personnalisant des parties du code source en fonctions de chaque système d'exploitation.

Exemple

```
1  CompilerIf #PB_Compiler_OS = #PB_OS_Linux
2    ; code spécifique Linux..
3  CompilerElseIf #PB_Compiler_OS = #PB_OS_Windows
4    ; code spécifique pour Windows
5  CompilerEndIf
```

Syntax

```
CompilerSelect <constante numérique>
  CompilerCase <constante numérique>
    ...
  [CompilerElse]
    ...
  [CompilerDefault]
    ...
CompilerEndSelect
```

Description

Fonctionne comme un Select : EndSelect classique en opérant une compilation conditionnelle.
Cette directive permet de construire des programmes multi-plateformes en personnalisant des parties du code source en fonctions de chaque système d'exploitation.

```
1  CompilerSelect #PB_Compiler_OS
2    CompilerCase #PB_OS_MacOS
3      ; du code spécifique à Mac OS X
4    CompilerCase #PB_OS_Linux
5      ; du code spécifique à Linux
6  CompilerEndSelect
```

Syntax

```
CompilerError <message>
CompilerWarning <message>
```

Description

Génère une erreur ou un avertissement du compilateur, comme si c'était une erreur de syntaxe et affiche le message spécifié. Ceci peut être utile pour faire des routines spécifiques et indiquer qu'elles ne sont pas disponibles pour un OS particulier.
Note : Alors que CompilerWarning affiche des avertissements "seulement", pendant que le processus de compilation se poursuit, la commande CompilerError arrête le processus de compilation.

Exemple : Génère une erreur

```
1  CompilerIf #PB_Compiler_OS = #PB_OS_Linux
2    CompilerError "Linux n'est pas supporté, désolé."
3  CompilerElse
4    CompilerError "OS Supporté, vous pouvez me mettre en commentaire."
5  CompilerEndIf
```

Exemple : Génère un avertissement

```
1  CompilerIf #PB_Compiler_OS = #PB_OS_Linux
2    CompilerWarning "Linux n'est pas supporté, désolé."
3  CompilerElse
4    CompilerWarning "OS Supporté, vous pouvez me mettre en commentaire."
5  CompilerEndIf
```

Syntax

```
EnableExplicit
DisableExplicit
```

Description

Active ou désactive le mode explicite. Quand il est actif, toutes les variables qui ne sont pas explicitement déclarées avec Define , Global , Protected ou Static ne seront pas acceptées par le compilateur et génèreront une erreur. Cela peut aider à éviter les erreurs de frappe.

Exemple

```
EnableExplicit

Define a

a = 20 ; Ok, car déclaré avec 'Define'
b = 10 ; Erreur
```

Syntax

```
EnableASM
DisableASM
```

Description

Active ou désactive l'assembleur en ligne. Quand il est actif, toutes les commandes assembleur sont incluses directement dans le code source, pour plus d'informations référez vous à la page L'assembleur en ligne .

Exemple

```
; Exemple assembleur x86
;
Test = 10

EnableASM
  MOV dword [v_Test],20
DisableASM

Debug Test ; Affichera 20
```

Constantes prédéfinies

Le compilateur PureBasic déclare automatiquement plusieurs constantes avant chaque compilation pour donner plus d'informations sur l'environnement de développement :

```
 #PB_Compiler_OS: Permet de savoir sur quelle plateforme est exécuté
le compilateur
 #PB_OS_Windows : Le compilateur est exécuté sur Windows
 #PB_OS_Linux   : Le compilateur est exécuté sur Linux
 #PB_OS_MacOS   : Le compilateur est exécuté sur Mac OS X

#PB_Compiler_Processor : Détermine le type de processeur pour lequel
 le programme est créé. Il peut s'agir de l'un des éléments suivants:
 #PB_Processor_x86     : Architecture de processeur x86 (aussi
appelé IA-32 ou x86-32)
 #PB_Processor_x64     : Architecture de processeur x86-64 (aussi
appelé x64, AMD64 ou Intel64)
 #PB_Processor_arm32   : Architecture de processeur  arm32
 #PB_Processor_arm64   : Architecture de processeur arm64 (appelé M1
sur les ordinateurs Apple)
```

```
  #PB_Compiler_Backend : Détermine quel type de compilateur est
   actuellement utilisé. Il peut s'agir de l'un des éléments suivants :
    #PB_Backend_Asm    : Le compilateur assembleur génére le code.
    #PB_Backend_C      : Le compilateur C génére le code.

  #PB_Compiler_ExecutableFormat : Détermine le format de l'exécutable.
   Il peut être l'un des suivants:
    #PB_Compiler_Executable : Exécutable standard
    #PB_Compiler_Console    : Exécutable console (Uniquement sur
   Windows, les autres OS utiliseront le format exécutable standard)
    #PB_Compiler_DLL        : DLL (dynlib sur MacOS X et objet partagé
   sur Linux)


  #PB_Compiler_Date     : La date actuelle, au moment de la
   compilation, au format date
PureBasic.
  #PB_Compiler_File     : Chemin complet et nom du fichier en cours de
   compilation, utile pour debogguer
.
  #PB_Compiler_FilePath : Chemin complet du fichier en cours de
   compilation, utile pour debogguer
.
  #PB_Compiler_Line     : Numéro de la ligne du fichier en cours de
   compilation, utile pour debogguer
.
  #PB_Compiler_Procedure: Nom de la procédure actuelle, si la ligne est
   à l'intérieur d'une procédure
.
  #PB_Compiler_Module   : Nom du module courant, si la ligne est à
   l'intérieur d'un module
.
  #PB_Compiler_Version  : Version du compilateur, nombre entier sous la
   forme '420' pour la version 4.20.
  #PB_Compiler_Home     : Chemin complet du répertoire PureBasic, utile
   pour localiser des fichiers inclus

  #PB_Compiler_Debugger : Egal à 1 si le débogueur
est activé, égal 0 sinon. Quand un exécutable est
                          créé, le débogueur est toujours désactivé
   (cette constante sera égale à 0).
  #PB_Compiler_Thread   : Egal à 1 si l'exécutable est compilé en mode
   'multi-threadé', égal à 0 sinon.
  #PB_Compiler_Unicode  : Egal à 1 si l'exécutable est compilé en mode
   unicode
, égal à 0 sinon.
  #PB_Compiler_LineNumbering : Egal à 1 si l'exécutable est compilé
   avec l'option 'Activer le numéro de ligne pour OnError
', égal à 0 sinon.
  #PB_Compiler_InlineAssembly: Egal à 1 si l'exécutable est compilé
   avec l'option 'Activer l'assembleur en ligne
', égal à 0 sinon.
  #PB_Compiler_EnableExplicit: Egal à 1 si l'exécutable est compilé en
   mode 'explicit', égal à 0 sinon.
  #PB_Compiler_IsMainFile    : Egal à 1 si l'exécutable en cours de
   compilation est le fichier principal, égal à 0 sinon.
  #PB_Compiler_IsIncludeFile : Egal à 1 si l'exécutable en cours de
   compilation a été inclus par un autre fichier, égal à 0 sinon.
```

```
#PB_Compiler_32Bit    : Mis à 1 si le compilateur génère du code 32
 bits, mis à 0 sinon.
#PB_Compiler_64Bit    : Mis à 1 si le compilateur génère du code 64
 bits, mis à 0 sinon.
#PB_Compiler_Optimizer: Mis à 1 si le compilateur génère du code
 optimisé, mis à 0 sinon.
```

Chapitre 24

Les fonctions du compilateur

Syntax

```
Bool(<expression booléenne>)
```

Description

Bool permet d'évaluer une expression booléenne en dehors des opérateurs conditionnels réguliers comme If, While, Until, etc.

Arguments

expression booléenne L'expression booléenne à tester.

Valeur de retour

Renvoie #True si l'expression booléenne est vraie, #False sinon.

Exemple

```
1  Salut$ = "Salut"
2  LeMonde$ = "Le Monde"
3
4  Debug Bool(Salut$ = "Salut")  ; Affichera 1
5  Debug Bool(Salut$ <> "Salut" Or LeMonde$ = "Le Monde") ; Affichera 1
```

Exemple

```
1  Procedure Chiffre(char.c)
2    ProcedureReturn Bool(char >= '0' And char <= '9')
3  EndProcedure
4
5  Debug Chiffre('0')
6  Debug Chiffre('1')
7  Debug Chiffre('a')
8  Debug Chiffre('z')
```

Syntax

```
Resultat = Defined(Nom, Type)
```

Description

Defined détermine si un objet tel qu'une structure , interface , variables etc. est déjà défini dans le programme.

Arguments

Nom Le nom de l'objet.
Le paramètre 'Nom' doit être spécifié sans aucune forme de décoration (sans le '#' pour une constante , sans les '()' pour un tableau , une liste , une map ou une procédure).

Type Peut prendre une des valeurs suivantes :

```
#PB_Constant
#PB_Variable
#PB_Array
#PB_List
#PB_Map
#PB_Structure
#PB_Interface
#PB_Procedure
#PB_Function
#PB_OSFunction
#PB_Label
#PB_Prototype
#PB_Module
#PB_Enumeration
```

Valeur de retour

Aucune.

Exemple

```
1   #PureConstante = 10
2
3   CompilerIf Defined(PureConstante, #PB_Constant)
4     Debug "La constante 'PureConstante' est déjà déclarée"
5   CompilerEndIf
6
7   Test = 25
8
9   CompilerIf Defined(Test, #PB_Variable)
10    Debug "La variable 'Test' est déjà déclarée"
11  CompilerEndIf
```

Syntax

```
Resultat = Subsystem(<expression texte constante>)
```

Description

Subsystem permet de savoir si un sous-système est utilisé pour le programme en cours de compilation.

Arguments

expression texte constante Le nom de sous-système.
Peut prendre une des valeurs suivantes : Sensible à la casse.

```
OpenGL
DirectX11
```

Valeur de retour

Renvoie une valeur ni nulle si le sous système est utilisé, zéro sinon.

Exemple

```
1  CompilerIf Subsystem("OpenGL")
2    Debug "Compilation avec le sous-système OpenGL"
3  CompilerEndIf
```

Syntax

```
Resultat = OffsetOf(Structure\Champ)
Resultat = OffsetOf(Interface\Fonction())
```

Description

La commande OffsetOf permet de déterminer la position en mémoire d'un champ d'une structure ou la position en mémoire d'une fonction dans le cas d'une interface (soit IndexDeLaFunction*SizeOf(Integer)).

Arguments

Structure\Champ ou Interface\Fonction() Le champ de la structure ou la fonction de l'interface.

Valeur de retour

Renvoie l'index du champ ou de la fonction, zéro sinon.

Exemple

```
1  Structure Personne
2    Nom.s
3    Prenom.s
4    Age.w
5  EndStructure
```

```
6
7   Debug OffsetOf(Personne\Age) ; Affichera 8 car une 'string' occupe 4
     octets en mémoire
8                                ;(16 avec un compilateur 64 bits, car
     une 'string' occupe 8 octets en mémoire)
9
10  Interface ITest
11    Creer()
12    Detruire(Options)
13  EndInterface
14
15  Debug OffsetOf(ITest\Detruire()) ; Affichera 4
```

Syntax

```
Resultat = SizeOf(Type)
```

Description

La commande SizeOf permet de renvoyer la taille en octets que prendra une structure , les types de base tels que les 'word', les 'float', etc.), une interface ou même une variable (Si une variable et une structure ont le même nom, la structure aura la priorité sur la variable.).
Comme SizeOf est une fonction du compilateur, elle ne fonctionne pas avec les tableaux, les Listes ou les Maps. Utilisez ArraySize() , ListSize() ou MapSize() à la place.
En tant que fonction du compilateur, SizeOf(x) est affectée à une constante et ne nécessite pas d'affectation à une autre variable si elle se trouve à l'intérieur d'une boucle ou d'une procédure appelée à plusieurs reprise.

Arguments

Type Le type de l'objet.

Valeur de retour

La taille de l'objet en mémoire, en octets

Remarques

C'est très utile dans de nombreux cas, notamment lors de l'utilisation des commandes API.
Note : Une variable de type caractère (CHARACTER) (.c) est unicode et occupe 2 octets et une variable de type ASCII (.a) n'occupe qu'1 octet.

Exemple : 1

```
1   VariableCaractere.c = '!'
2   Debug SizeOf(VariableCaractere) ; affiche 2, c'est à dire 2 octets
3   Debug SizeOf(CHARACTER)         ; affiche 2, c'est à dire 2 octets
4
5   VariableAscii.a = '!'
6   Debug SizeOf(VariableAscii)     ; affiche 1, c'est à dire 1 octets
7   Debug SizeOf(ASCII)             ; affiche 1, c'est à dire 1 octets
```

Exemple : 2

```
1   Structure Personne
2     Nom.s
3     Prenom.s
4     Age.w
5   EndStructure
6
7   Debug "La taille d'une personne est "+Str(Sizeof(Personne))+" octets"
      ; Affichera 10 (4+4+2)
8
9   Jean.Personne\Nom = "Jean"
10
11  Debug SizeOf(Jean) ; Affichera 10 aussi
```

Syntax

```
Resultat = TypeOf(Objet)
```

Description

TypeOf permet de déterminer le type d'une variable , ou d'un champ de structure .

Arguments

Objet L'objet à utiliser.

Valeur de retour

Le type de l'objet.
Le type peut être une des valeurs suivantes :

```
#PB_Byte
#PB_Word
#PB_Long
#PB_String
#PB_Structure
#PB_Float
#PB_Character
#PB_Double
#PB_Quad
#PB_List
#PB_Array
#PB_Integer
#PB_Map
#PB_Ascii
#PB_Unicode
#PB_Interface
```

Exemple

```
Structure Personne
  Nom.s
  Prenom.s
  Age.w
EndStructure

If TypeOf(Personne\Age) = #PB_Word
  Debug "Age est un 'Word'"
EndIf

Surface.f
If TypeOf(Surface) = #PB_Float
  Debug "Surface est un 'Float'"
EndIf
```

Syntax

```
InitializeStructure(*Memoire, Structure)
```

Description

Initialise un objet structuré en mémoire.
En fait, cette fonction initialise les membres d'une structure . Ces membres sont de type Array , List ou Map mais les autres types ne sont pas affectés (.s .l, .i, etc.).

Arguments

***Memoire** L'adresse mémoire à utiliser.

Structure Le nom de la structure qui doit être utilisé pour effectuer l'initialisation.
Il n'y a pas de contrôle pour s'assurer que la zone mémoire corresponde à la structure.

Valeur de retour

Aucune.

Remarques

Attention : plusieurs appels à InitializeStructure crée une fuite de mémoire parce que les anciens membres de la structure ne sont pas libérés de la mémoire. ClearStructure doit être appelée avant d'appeler une nouvelle fois InitializeStructure.
Cette fonction est pour les utilisateurs avancés et doit être utilisée avec précaution.
Pour allouer une structure dynamique, utiliser AllocateStructure() .

Exemple

```
Structure Personne
  Prenom$
  Age.l
  List Amis.s()
EndStructure
```

```
6
7   *Etudiant.Personne = AllocateMemory(SizeOf(Personne))
8   InitializeStructure(*Etudiant, Personne)
9
10  ; Maintenant, la liste est prête à l'emploi
11  ;
12  AddElement(*Etudiant\Amis())
13  *Etudiant\Amis() = "John"
14
15  AddElement(*Etudiant\Amis())
16  *Etudiant\Amis() = "Yann"
17
18  ; Affichage du contenu de la liste
19  ;
20  ForEach *Etudiant\Amis()
21    Debug *Etudiant\Amis()
22  Next
```

Syntax

```
CopyStructure(*Source, *Destination, Structure)
```

Description

Copie une structure en mémoire vers une autre.

Arguments

***Source** L'adresse mémoire contenant la structure à copier.

***Destination** L'adresse mémoire de la copie.

Structure Le nom de la structure qui doit être utilisé pour effectuer la copie.

Valeur de retour

Aucune.

Remarques

C'est particulièrement utile lors de l'utilisation de mémoire dynamique avec les pointeurs. Chaque champ de la structure sera dupliqué, y compris les tableaux dynamiques, les listes ou les maps . La structure de destination sera automatiquement effacée avant de faire la copie, il n'est pas nécessaire d'appeler ClearStructure avant CopyStructure.
Attention : La destination doit être une zone de mémoire de structure valide, ou une zone mémoire effacée. Si la zone de mémoire n'est pas effacée, cela pourrait provoquer un crash, car des valeurs aléatoires seront utilisées.
Il n'y a pas de contrôle pour s'assurer que les deux zones mémoires sont bien du type 'Structure', donc il est impératif de manipuler cette commande avec précaution.

Exemple

```
Structure Personne
  Prenom$
  Nom$
  Map Amis$()
  Age.l
EndStructure

Etudiant.Personne\Prenom$ = "Paul"
Etudiant\Nom$ = "Morito"
Etudiant\Amis$("Tom") = "Jones"
Etudiant\Amis$("Jim") = "Doe"

CopyStructure(@Etudiant, @EtudiantCopy.Personne, Personne)

Debug EtudiantCopy\Prenom$
Debug EtudiantCopy\Nom$
Debug EtudiantCopy\Amis$("Tom")
Debug EtudiantCopy\Amis$("Jim")
```

Syntax

```
Resultat = ClearStructure(*Memoire, Structure)
```

Description

Vide la zone mémoire structurée et met la valeur de tous les champs à zéro.

Arguments

***Memoire** L'adresse mémoire contenant la structure à effacer.

Structure Le nom de la structure qui sera utilisée pour effectuer le nettoyage.

Valeur de retour

Aucune.

Remarques

C'est particulièrement utile quand la structure contient des chaînes de caractères, des tableaux , des listes ou des maps qui ont été alloués en interne par PureBasic. Tous les champs seront mis à zéro, même les types natifs comme long, integer, etc.
Il n'y a pas de contrôle pour s'assurer que la zone mémoire est bien du type 'Structure' spécifié, donc il est impératif de manipuler cette commande avec précaution. Cette fonction est réservée aux utilisateurs avancés.

Exemple

```
Structure Personne
  Prenom$
  Nom$
```

```
    Age.l
  EndStructure

  Etudiant.Personne\Prenom$ = "Paul"
  Etudiant\Nom$ = "Morito"
  Etudiant\Age = 10

  ClearStructure(@Etudiant, Personne)

  ; Affichera des chaines vide, car la structure entiere a ete videe.
   Tous les autres champs ont ete remis a zero
  ;
  Debug Etudiant\Prenom$
  Debug Etudiant\Nom$
  Debug Etudiant\Age
```

Syntax

```
ResetStructure(*Memoire, Structure)
```

Description

ResetStructure, vide la zone mémoire structurée et l'initialise pour être prête à être utilisée.

Arguments

***Memoire** L'adresse mémoire contenant la structure à réinitialiser.

Structure Le nom de la structure utilisée.

Valeur de retour

Aucune.

Remarques

C'est particulièrement utile quand la structure contient des chaînes de caractères, des tableaux , des listes ou des maps qui ont été alloués en interne par PureBasic. Tous les champs seront mis à zéro, même les types natifs comme long, integer, etc.
Il n'y a pas de contrôle pour s'assurer que la zone mémoire est bien du type 'Structure' spécifié, donc il est impératif de manipuler cette commande avec précaution. Cette fonction est réservée aux utilisateurs avancés.

Exemple

```
  Structure Personne
    Map Amis.s()
  EndStructure

  Henri.Personne\Amis("1") = "Paul"

  ResetStructure(@Henri, Personne)
```

```
8
9   ; Affiche une chaîne vide car l'ensemble de la structure a été
      réinitialisée.
10  ; La map est encore utilisable, mais vide.
11  ;
12  Debug Henri\Amis("1")
```

Chapitre 25

Data

Introduction

PureBasic autorise l'utilisation de données pour stocker des blocs d'informations prédéfinies dans votre programme. Ceci est utile pour disposer de valeurs par défaut (messages textuels par exemple) ou dans un jeu pour définir le cheminement prédéfini d'un sprite.
DataSection doit être placé en tête du bloc de données. Tous les labels et composants data sont stockés dans cette partie data dont l'accès est plus rapide que la zone de code. Data est utilisé pour entrer des données. EndDataSection doit être spécifié si du code à exécuter est situé après. Il est intéressant de pouvoir placer plusieurs zones de données à différents endroits du code source. Les données peuvent être chargées à l'aide des commandes Restore et Read.
Ces fonctions ne doivent pas être utilisées dans un thread.

Syntax

```
DataSection
```

Description

Début d'une zone de données.

Exemple

```
1  DataSection
2    DonneesNumeriques:
3      Data.i 100, 200, -250, -452, 145
4
5    DonneesTexte:
6      Data.s "Bonjour", "Qu'est-ce", "que ", "c'est ?"
7  EndDataSection
```

Syntax

```
EndDataSection
```

Description

Fin d'une zone de données.

Exemple

```
1  DataSection
2    DonneesNumeriques:
3      Data.l 100, 200, -250, -452, 145
4
5    DonneesTexte:
6      Data.s "Bonjour", "Qu'est-ce", "que ", "c'est ?"
7  EndDataSection
```

Syntax

```
Data.NomType
```

Description

Définit les données. Le type peut être choisi parmi les types natifs (integer, long, word, byte, ascii, unicode, float, double, quad, character, string). Un nombre quelconque de données peut être placé sur une même ligne, chacune étant séparée par une virgule.

Exemple

```
1  DataSection
2    Divers:
3  Data.l 100, 200, -250, -452, 145
4  Data.s "Bonjour", "Qu'est", "ce que ", "c'est ?"
5  EndDataSection
```

Pour les programmeurs chevronnés : il est aussi possible de mettre l'adresse d'une procédure ou d'un label avec Data si le type 'entier' (integer .i) est utilisé. Sur un système 32 bits les adresses seront stockées sur 4 octets et sur 8 octets sur un système 64 bits.
Par exemple, des tables de fonctions virtuelles peuvent être créées facilement.

Exemple

```
1  Procedure Max(Nombre, Nombre2)
2  EndProcedure
3
4  Etiquette:
5
6  DataSection
7    Divers:
8    Data.i ?Etiquette, @Max()
9  EndDataSection
```

Exemple

```
Interface MonObjet
   FaireCeci()
   FaireCela()
EndInterface

Procedure Ceci(*Self)
   MessageRequester("MonObjet", "Ceci")
EndProcedure

Procedure Cela(*Self)
   MessageRequester("MonObjet", "Cela")
EndProcedure

m.MonObjet = ?VTable

m\FaireCeci()
m\FaireCela()


DataSection
   VTable:
      Data.i ?Procedures
   Procedures:
      Data.i @Ceci(), @Cela()
EndDataSection
```

Syntax

```
Restore label
```

Description

Ce mot clef permet de placer un indicateur de début de zone pour la commande Read. Le label utilisé par cette commande doit être déclaré dans le bloc DataSection, car il sera déplacé lors de la création de l'exécutable.

Exemple

```
Restore DonneesTexte
Read.s MonPremierTexte$
Read.s MonSecondTexte$

Restore DonneesNumeriques
Read.l a
Read.l b

Debug MonPremierTexte$
Debug a

End

DataSection

```

```
16      DonneesNumeriques:
17        Data.l 100, 200, -250, -452, 145
18
19      DonneesTexte:
20        Data.s "Bonjour", "Qu'est-ce", "que ", "c'est ?"
21    EndDataSection
```

Syntax

```
Read[.<type>] <variable>
```

Description

Lit la donnée suivante et par défaut, la donnée suivante est la première donnée déclarée.
Le pointeur peut être modifié en utilisant la commande Restore.
<type> est le type de données à lire et s'il n'est pas spécifié le type par défaut qui est 'Integer' sera utilisé. Il est toutefois conseillé d'utiliser le type ad-hoc afin d'éviter le message d'erreur qui apparaîtra quand vous lirez des datas de type 'String' et pour éviter une confusion de type 'Integer' qui est un type 'Long' pour les compilateurs 32 bits et qui est un type 'Quad' pour les compilateurs 64 bits.

Exemple

```
1   Restore DonneesTexte
2   Read.s MonPremierTexte$
3
4   Restore DonneesNumeriques
5   Read a    ; Attention, le compilateur lira un 'Integer' (Un 'Long' en
     32 bits ou un 'Quad' en 64 bits)
6   Read.q b
7   Read c    ; Attention, le compilateur lira un 'Quad' si c'est un
     compilateur 64 bits même si les données sont 'Long' !
8   Read.l d
9
10  Debug MonPremierTexte$; Affiche Hello
11  Debug a   ; Affiche 100
12  Debug b   ; Affiche 11111111111111111
13  Debug c   ; Attention, l'affichage dépend du compilateur !  : 200 en
     32 bits ou 1288490189000 en 64 bits
14  Debug d   ; Idem : 300  en 32 bits ou 400 en 64 bits
15
16  End
17
18  DataSection
19
20    DonneesNumeriques:
21      Data.i 100
22      Data.q 11111111111111111
23      Data.l 200, 300, 400
24
25    DonneesTexte:
26      Data.s "Bonjour", "Qu'est-ce", "que ", "c'est ?"
27  EndDataSection
```

Chapitre 26

Commandes de débogage

Introduction

La description complète des fonctionnalités du débogueur se trouve dans les chapitres Utilisation du débogueur et Utiliser les outils de déboguage .
Une bibliothèque Debugger est également disponible pour controler le comportement du débogueur à partir du code source.

Syntax

```
CallDebugger
```

Description

Appel du ”débogueur” et arrêt immédiat du programme au point courant du code.

Syntax

```
Debug <expression> [, NiveauDebug]
```

Description

Affiche la fenêtre DebugOutput et le résultat correspondant. L'expression peut être toute expression valide en PureBasic, de forme numérique ou chaine. Un point important est que toute commande Debug et les expressions associées sont totalement ignorées (non compilées) si le débogueur est désactivé. Cela singifie qu'il n'est pas nécessaire de passer les instruction Debug en commentaires lors de la création de l'exécutable final tout en ayant la possibilité de tracer facilement l'exécution du programme pour le développeur.
'NiveauDebug' est le niveau de priorité des messages du débogueur. Tous les messages (avec un niveau non spécifié) sont affichés automatiquement. Lorsqu'un niveau est spécifié alors le message correspondant ne sera affiché que si le niveau de debug courant est égal ou supérieur au niveau associé au message. Cela permet de réaliser une traçabilité hiérarchisée en affichant des informations de plus en plus précises en fonction de la valeur 'NiveauDebug' utilisée.

Syntax

```
DebugLevel <expression constante >
```

Description

Fixe le niveau courant pour les messages 'Debug'.
Note : Le niveau est fixé au moment de la compilation, ce qui signifie que vous devez mettre la commande DebugLevel avant les commandes debug pour être sûr qu'elles seront bien toutes affectées.

Syntax

```
DisableDebugger
```

Description

Interromps l'utilisation du débogueur sur les lignes du code qui suivent cette commande.

Syntax

```
EnableDebugger
```

Description

Active l'utilisation du débogueur sur les lignes de code qui suivent cette commande (lorsque le débogueur a été préalablement interrompu par la commande DisableDebugger).

Chapitre 27

Define

Syntax

```
Define.<type> [<variable> [= <expression>], <variable> [=
    <expression>], ...]
```

Description

Permet d'assigner le même type de données à une série de variables .
Sans ce mot clé, les variables sont crées avec le type par défaut de PureBasic qui est le type entier INTEGER. Pour rappel le type INTEGER vaut :
4 octets (avec un compilateur 32 bits) allant de -2147483648 à +2147483647
8 octets (avec un compilateur 64 bits) allant de -9223372036854775808 à +9223372036854775807

Exemple

```
1   Define.b a, b = 10, c = b*2, d ; Ces 4 variables sont de type octets
    (byte .b)
```

Define est d'une grande souplesse car il permet aussi d'assigner un type à une variable en particulier à l'intérieur d'une série.

Exemple

```
1   Define.q a, b.w, c, d  ; a, c, et d sont des Quad (.q) alors que b
    est un Word (.w)
2
3   Debug SizeOf(a) ; Affichera 8
4   Debug SizeOf(b) ; Affichera 2
5   Debug SizeOf(c) ; Affichera 8
6   Debug SizeOf(d) ; Affichera 8
```

Define peut également être utilisé avec les tableaux , les listes et les maps .

Syntax

```
Define <variable>.<type> [= <expression>] [, <variable>.<type> [=
    <expression>], ...]
```

Description

Autre possibilité pour la déclaration des variables avec Define.

Exemple

```
1  Define MonChar.c ; Caractère
2  Define MonLong.l ; Double Mots
3  Define MonWord.w ; Mot
4
5  Debug SizeOf(MonChar)   ; Affichera 2 (à cause du mode unicode)
6  Debug SizeOf(MonLong)   ; Affichera 4
7  Debug SizeOf(MonWord)   ; Affichera 2
```

Chapitre 28

Dim

Syntax

```
Dim nom.<type>(<expression>, [<expression>], ...)
```

Description

Dim est utilisé pour créer un nouveau tableau.
Un tableau peut être composé d'éléments de type basique connus sous PureBasic, incluant les et les types définis par l'utilisateur.
Attention : Les éléments d'un tableau doivent tous être de même type.
Une fois le tableau créé, il peut être redimensionné avec ReDim .
Pour afficher toutes les commandes utilisées pour gérer les tableaux, reportez-vous à la bibliothèque Array .
Les tableaux sont alloués dynamiquement ce qui signifie que leur dimensionnement peut se faire à partir d'une variable ou d'une expression.
Attention, la numérotation des éléments dans PureBasic (comme dans les autres BASIC) commence à 0.
Par exemple, le tableau Dim(10) aura 11 éléments allant de 0 à 10.
Ce comportement est différent pour les tableaux statiques spécifiques aux structures . Les tableaux statiques utilisent des crochets "[]", par exemple TabStatic[2] ne comporte que 2 éléments de 0 à 1 et les fonctions de la bibliothèques Array ne fonctionnent pas avec eux.
Les tableaux sont toujours locaux par défaut, donc pour accéder à partir d'une procedure à un tableau défini dans le code source principal du programme, l'utilisation de Global ou Shared est requise. Il est également possible de passer un tableau en paramètre d'une procédure à l'aide du mot clef Array. Il sera passé "par référence" (ce qui signifie que le tableau ne sera pas copié, et les fonctions dans la procédure manipulerons le tableau original).
Pour effacer le contenu complet d'un tableau et libérer la mémoire qu'il occupe, utilisez FreeArray() .
Si Dim est utilisé sur un tableau existant, il réinitialise son contenu à zéro.
Utilisez la commande Swap pour permuter le contenu de tableaux rapidement.
Note : La vérification des accès à un tableau est effectuée uniquement quand le débogueur est activé.

Exemple

```
1  Dim MonTableau(41)
2  MonTableau(0) = 1
3  MonTableau(1) = 2
```

Exemple : Tableau à dimensions multiples

```
Dim TableauMultiple.b(NbColonnes, NbLignes)
TableauMultiple(10, 20) = 10
TableauMultiple(20, 30) = 20
```

Exemple : Tableau en paramètre d'une procédure

```
Procedure fill(Array Nombres(1), Longueur)  ; Le 1 représente le nombre de dimensions du tableau
  For i = 0 To Longueur
    Nombres(i) = i
  Next i
EndProcedure

Dim Nombres(10)
fill(Nombres(), 10)  ; Le tableau Nombres() est passé en paramètre

Debug Nombres(5)
Debug Nombres(10)
```

Syntax

```
ReDim nom.<type>(<expression>, [<expression>], ...)
```

Description

ReDim permet de redimensionner un tableau déjà déclaré tout en préservant son contenu. La nouvelle taille peut être plus grande ou plus petite, mais le nombre de dimensions ne peut pas être modifié.
Si ReDim est utilisé sur un tableau à plusieurs dimensions, seule la dernière dimension peut être changée.

Exemple : Une dimension

```
Dim MonTableau.l(1) ; nous avons 2 éléments
MonTableau(0) = 1
MonTableau(1) = 2

ReDim MonTableau(4) ; Maintenant nous avons 5 éléments
MonTableau(2) = 3

For k = 0 To 2
  Debug MonTableau(k)
Next
```

Exemple : Plusieurs dimensions

```
Dim MonTableau.l(1,1,1) ; 3 dimensions et 2 éléments par dimension

ReDim MonTableau(1,1,4) ; OUI seule la dernière dimension peut être changée.
;ReDim MonTableau(4,1,1) ; NON
;ReDim MonTableau(1,4,1) ; NON
MonTableau(1,1,4) = 3
```

Chapitre 29

Construire une DLL

Introduction

PureBasic permet de créer des DLL Microsoft Windows (DLL : Dynamic Linked Library), des objets partagés (.so) sous Linux, et des bibliothèques dynamiques (.dylib) sous MacOS X. Le code d'une DLL est de même nature que le code PureBasic excepté qu'aucun code ne peut exister en dehors d'une procédure. Tout le code est intégré dans des procédures, sauf la déclaration des variables globales et des structures.
L'avantage d'une DLL est de pouvoir partager du code déjà compilé avec un programme tiers. Pour cela il faut déclarer la DLL avec le mot clef ProcedureDLL au lieu de Procedure . Si la procédure partagée doit être au format 'CDecl' (ce qui n'est pas le cas pour les DLL standards de Windows), le mot clef ProcedureCDLL doit être utilisé.
De même Declare doit être remplacé par DeclareDLL ou DeclareCDLL.
Avant de compiler a DLL, il est necessaire de sélectionner 'Shared DLL' comme format de sortie dans le menu 'Compiler\Option' de l'éditeur PureBasic (ou d'utiliser le commutateur /DLL dans la ligne de commande). Une fois compilé, une DLL apparait avec le même nom que le fichier PureBasic.

Exemple

```
1   ; Créer et enregistrer ce fichier sous le nom 'PureBasic.pb'
2   ; La DLL 'PureBasic.dll' sera créée.
3
4   ProcedureDLL MaFonction()
5     MessageRequester("Bonjour", "Voici une DLL PureBasic !", 0)
6   EndProcedure
7
8   ; Créer un deuxième fichier PureBasic avec le code suivant:
9   ; Voici le programme client qui utilise la DLL
10  ; Compiler et executer ce programme dans le dossier
11  ; contenant la dll 'PureBasic.dll'.
12
13  If OpenLibrary(0, "PureBasic.dll")
14    CallFunction(0, "MaFonction")
15    CloseLibrary(0)
16  EndIf
```

Pour les programmeurs avancés, il existe 4 procédures spéciales qui sont appelées automatiquement par l'OS lorsque l'un des évènements suivants surviennent :
- une DLL est attachée à un nouveau process
- une DLL est détachée d'un process
- une DLL est attachée à un nouveau thread

- une DLL est détachée d'un thread
Pour gérer cela, il est possible de déclarer 4 procédures spéciales nommées : AttachProcess(Instance), DetachProcess(Instance), AttachThread(Instance) et DetachThread(Instance). Cela signifie que ces 4 noms sont réservés et ne peuvent être utilisés par le programmeur pour d'autres usages.
Notes à propos de la création des DLL's :
- La déclaration des tableaux avec Dim , des listes avec NewList ou des maps avec NewMap doit toujours être faite dans la procédure 'AttachProcess()'.
- Ne pas écrire de code en dehors des procédures. La seule exception est la déclaration des variables et des structures.
- Les routines d'initialisation DirectX ne doivent pas être dans la procédure 'AttachProcess()'.
A propos du renvoi de chaîne de caractères (string) par une DLL :
Pour qu'une fonction puisse renvoyer une string, elle doit être déclarée en global .

Exemple

```
1  Global ReturnString$
2
3  ProcedureDLL.s MaFonction(var.s)
4    ReturnString$ = var + " test"
5    ProcedureReturn ReturnString$
6  EndProcedure
```

Si la chaîne de caractères n'est pas déclarée en global alors elle sera locale à la procédure et donc libérée à la fin de la procédure. Elle ne pourra pas être utilisée par le programme appelant la fonction.
Quand CallFunction() ou une fonction similaire de type CallXXX est utilisée dans un programme pour appeler une fonction dans une DLL, le programme reçoit un pointeur vers une chaîne de caractères, qui pourra être lu avec PeekS() .

Exemple

```
1  String.s = PeekS(CallFunction(0, "NomFonction", Parametre1,
     Parametre2))
```

Vous trouverez ci dessous un exemple complet :

Chapitre 30

Enumérations

Syntax

```
Enumeration [nom] [<constante> [Step <constante>]]
  #Constante1
  #Constante2 [= <constante>]
  #Constante3
  ...
EndEnumeration
```

```
EnumerationBinary [nom] [<constante>]
  #Constante1
  #Constante2 [= <constante>]
  #Constante3
  ...
EndEnumeration
```

Description

Les énumérations sont très pratiques pour déclarer rapidement une série de constantes sans s'occuper de leur valeur numérique. La première constante de l'énumération prendra la valeur 0, la constante suivante prendra la valeur 1 etc. Il est possible de changer la valeur de départ de l'énumération et d'ajuster la valeur utilisée pour l'incrémentation de chaque constante. Si nécessaire, il est possible d'affecter directement une valeur numérique à une constante (grâce à l'opérateur '=') et les constantes suivantes utiliseront cette nouvelle valeur comme valeur de base. Comme les énumerations n'acceptent que les nombres entiers, les nombres flottants seront arroundis à l'entier le plus proche.
Un 'nom' peut être configuré pour identifier une énumération et permettre de l'interrompre puis de la poursuivre plus tard. C'est utile pour regrouper des objets dans une même énumération tout en les déclarant dans différents endroits du code.
Pour les utilisateurs avancés seulement : La constante réservée #PB_Compiler_EnumerationValue stocke la prochaine valeur qui sera utilisée par l'énumération. Cela peut être utile pour connaître la valeur de la dernière énumération ou pour chaîner plusieurs énumérations.
EnumerationBinary peut être utilisé pour créer des énumérations appropriées pour les options (flags). La première valeur de l'élément est 1.

Exemple : Enumération simple

```
1  Enumeration
2    #GadgetInfo ; égale à 0
```

```
    #GadgetText ; égale à 1
    #GadgetOK   ; égale à 2
  EndEnumeration
```

Exemple : Enumération avec un pas déterminé

```
  Enumeration 20 Step 3
    #GadgetInfo ; égale à 20
    #GadgetText ; égale à 23
    #GadgetOK   ; égale à 26
  EndEnumeration
```

Exemple : Enumération avec un changement dynamique

```
  Enumeration
    #GadgetInfo       ; égale à 0
    #GadgetText = 15 ; égale à 15
    #GadgetOK         ; égale à 16
  EndEnumeration
```

Exemple : Enumérations nommées

```
  Enumeration Gadget
    #GadgetInfo ; égale à 0
    #GadgetText ; égale à 1
    #GadgetOK   ; égale à 2
  EndEnumeration

  Enumeration Window
    #FirstWindow  ; égale à 0
    #SecondWindow ; égale à 1
  EndEnumeration

  Enumeration Gadget
    #GadgetCancel ; égale à 3
    #GadgetImage  ; égale à 4
    #GadgetSound  ; égale à 5
  EndEnumeration
```

Exemple : Obtenir la prochaine valeur d'énumération

```
  Enumeration
    #GadgetInfo ; égale à 0
    #GadgetText ; égale à 1
    #GadgetOK   ; égale à 2
  EndEnumeration

  Debug "La prochaine valeur d'énumération est : " +
   #PB_Compiler_EnumerationValue ; affiche 3
```

Exemple : Enumération binaire

```
1  EnumerationBinary
2    #Flags1 ; égale à 1
3    #Flags2 ; égale à 2
4    #Flags3 ; égale à 4
5    #Flags4 ; égale à 8
6    #Flags5 ; égale à 16
7  EndEnumeration
```

Chapitre 31

For : Next

Syntax

```
For <variable> = <expression1> To <expression2> [Step <constante>]
  ...
Next [<variable>]
```

Description

La fonction For : Next est utilisée pour produire une boucle dans le programme, avec les paramètres définis. A chaque cycle, <variable> est incrémentée de 1 (ou d'une valeur correspondant au pas indiqué dans Step <constante>). Le première valeur de <variable> est <expression1>. La boucle est interrompue dès que la valeur de <variable> est supérieure la valeur de <expression2>.
La commande Break permet de quitter à n'importe quel moment une ou plusieurs boucles. la commande Continue permet de passer directement à la prochaine itération de la boucle.
La boucle For : Next fonctionne uniquement avec des valeurs entières, aussi bien pour les expressions que pour la constante Step. La constante Step peut aussi être négative.

Exemple

```
1  For k = 0 To 10
2    Debug k
3  Next
```

Dans cet exemple, le programme bouclera 11 fois (de 0 à 10) et sortira.

Exemple

```
1  For k = 10 To 1 Step -1
2    Debug k
3  Next
```

Dans cet exemple, le programme bouclera 10 fois (de 10 à 1) et sortira.

Exemple

```
a = 2
b = 3
For k = a+2 To b+7 Step 2
  Debug k
Next k
```

Ici, le programme bouclera 4 fois avant de sortir (k est augmentée de 2 à chaque cycle et prend donc successivement les valeurs 4, 6, 8 et 10=. La lettre k après le mot clef Next indique que ce Next ferme la boucle "For k". Si un autre nom de variable est utilisé, le compilateur génèrera une erreur. Cela est utile pour la lisibilité d'un code comprenant des boucles imbriquées.

Exemple

```
For x=0 To 319
  For y=0 To 199
    Plot(x,y)
  Next y
Next x
```

Note : Gardez à l'esprit que la valeur de <expression2> peut également être changée dans la boucle For : Next, ce qui peut engendrer une boucle sans fin si la valeur n'est pas correcte.

Chapitre 32

ForEach : Next

Syntax

```
ForEach Liste() Ou Map()
  ...
Next [Liste() Ou Map()]
```

Description

ForEach énumère tous les élèments d'une liste ou d'une map . Si la liste ou la map est vide, ForEach : Next quitte immédiatement, sans entrer dans la boucle.
Lors du l'utilisation en conjonction avec une liste : comme la boucle se termine seulement lorsque le dernier élément de la liste est atteint (en terme de position), il est tout à fait possible de supprimer ou d' ajouter des éléments durant un cycle de boucle. De même il est permis de changer l'élément courant avec ChangeCurrentElement() . Après l'un de ces changements, le prochain cycle de boucle continue avec l'élément qui suit l'élément courant.
Il est possible de quitter une boucle ForEach : Next à tout moment à l'aide de la commande Break . La commande Continue permet de passer directement à l'itération suivante, sans exécuter la fin du code contenu dans la boucle.

Exemple : Liste

```
1   NewList Nombre()
2
3   AddElement(Nombre())
4   Nombre() = 10
5
6   AddElement(Nombre())
7   Nombre() = 20
8
9   AddElement(Nombre())
10  Nombre() = 30
11
12  ForEach Nombre()
13    Debug Nombre() ; Affichera 10, 20 et 30
14  Next
```

Exemple : Map

```
NewMap Pays.s()

Pays("US") = "United States"
Pays("FR") = "France"
Pays("DE") = "Allemagne"

ForEach Pays()
  Debug Pays()
Next
```

Chapitre 33

Règles de syntaxe générales

PureBasic a défini des règles qui ne changent jamais,à savoir :

; ; ; ; ; ; ; ; ;> Commentaires

Exemple

```
1  If a = 10 ; Ceci est un commentaire pour indiquer quelque
     chose.
```

> Mots clé (Keywords)

Tous les **mots clés** sont utilisés pour des choses générales à l'intérieur de PureBasic, comme la création de tableaux (Dim) ou des listes (NewList), ou le contrôle du déroulement du programme (If : Else : EndIf). Ils ne sont pas suivis par les parenthèses '()', qui sont généralement utilisées par PureBasic pour les **fonctions**.

Exemple

```
1  If a = 1       ; If, Else et EndIf sont des mots clés;
   contrairement à 'a'
2     ...         ; qui est une variable utilisée dans une
   expression ('a = 1').
3  Else
4     ...
5  EndIf
```

Les **mots clés** sont régulièrement décrits dans les chapitres sur le côté gauche de la page d'index du manuel de référence.

> Fonctions

Toutes les fonctions doivent avoir un nom suivi d'un '(' à défaut de quoi elle ne sera pas considérée comme une fonction. Cela est vrai y compris lorsque la fonction ne prend aucun paramètre.

Exemple

```
1  WindowEvent() ; est une fonction.
2  WindowEvent   ; est une variable.
```

> Constantes

Toutes les constantes ont un nom précédé par un #.
Elles ne peuvent être déclarées qu'une fois et garde toujours leur valeur prédéfinie. (Le compilateur remplace tous les noms des constantes avec leur valeur correspondante lors de la compilation de l'exécutable.)

Exemple

```
1  #Hello = 10 ; est une constante.
2  Hello  = 10 ; est une variable.
```

> Texte littéral

Les chaînes littérales sont encadrées par le caractère ". Les séquences d'échappement sont prises en charge en utilisant le caractère * * avant la chaîne littérale.
Les séquences d'échappement autorisés sont :

```
\a: bip                    Chr(7)
\b: retour arrière         Chr(8)
\f: saut de page           Chr(12)
\n: retour à la ligne      Chr(10)
\r: retour chariot         Chr(13)
\t: tabulation horizontal  Chr(9)
\v: tabulation vertical    Chr(11)
\": double quote           Chr(34)
\\: antislash              Chr(92)
```

Il y a deux constantes spéciales pour les chaînes :

```
#Empty$: représente une chaîne vide (exactement la même chose que "")
#Null$ : représente une chaîne nulle. Ceci peut être utilisé avec les
 fonctions
         des APIs nécessitant un pointeur NULL en guise de chaîne ou
 à une chaîne vraiment libre.
```

Attention : Sous Windows, \t ne fonctionne pas avec les fonctions graphiques des bibliothèques 2DDrawing et VectorDrawing.

Exemple

```
1  a$ =  "Salut le monde"  ; Texte standard
2  b$ = ~"Echappe\nMoi !"  ; Texte avec une séquence
    d'échappement
```

> Labels

Tous les labels (étiquettes) doivent être suivis par un ' :'. Les noms de label ne peuvent pas contenir d'opérateurs (+,-,...) ou de caractères spéciaux (é,à,ß,ä,ö,ü,...).
Dans une procédure , l'étiquette sera disponible uniquement dans cette procédure.

Exemple

```
1  Je_suis_un_label:
```

> Modules

Les modules utilisent un double deux-points comme cela **' : :'** ce qui permet au programmeur d'accéder à un élément d'un module depuis l'extérieur de ce module. Le nom du module étant précisé, il est suivi du séparateur ' : :'. Donc attention à ne pas confondre le ' : :' avec le simple' :' d'une étiquette (label).

Exemple

```
1  Debug Voitures::NbVoitures ; Affiche le contenu de la
   variable 'NbVoitures' du module 'Voitures'
```

> Expressions

On appelle expression toute séquence de code qui peut être évaluée. Une expression peut regrouper toute variable, constante ou fonction d'un même type. Lorsque vous utilisez des nombres dans une expression, vous pouvez utiliser le symbole $ en tête pour préciser qu'il s'agit d'une valeur hexadécimale ou un % pour signifier une valeur binaire. Sans l'un ou l'autre de ces deux symboles, la valeur sera toujours considérée comme décimale. Les chaînes de caractères doivent être délimitées par des guillemets.

Exemple

```
1  a = a + 1 + (12 * 3)
2  a = a + WindowHeight() + b/2 + #MaConstante
3
4  If a <> 12 + 2
5    b + 2 >= c + 3
6  EndIf
7
8  a.s = b.s + "ceci est une chaine" + c.s
9  a$ = b$ + "ceci est une autre chaine" + c$
10
11 foo = foo + $69 / %1001 ; Utilisation de nombres hexadécimal
    et binaire
```

> Regroupement des commandes

Il est possible de placer un nombre quelconque de commandes sur la même ligne en les séparant par :.

Exemple

```
1  If Variable=0 : Debug "Ok" : Else : Debug "Erreur" : EndIf
```

> Texte multiligne

Si une ligne de code contient une expression de grande taille, elle peut être divisée en plusieurs lignes. Une ligne découpée doit se terminer avec l'un des opérateurs suivants : plus (+), virgule (,), ou (||), And, Or, Xor.

Exemple

```
1  Texte$ = "Très très très très long texte" + #LF$ +
2           "un autre long texte" + #LF$ +
3           "et la fin du long texte."
4
5  MessageRequester("Bonjour c'est un titre très long.",
6                   "Et un très long message, afin que nous
    puissions utiliser le multiligne" + #LF$ + Texte$,
7                   #PB_MessageRequester_Ok)
```

> Glossaire

Les mots suivants utilisés dans ce manuel ont toujours le même sens :
<variable> : une variable basic.
<expression> : une expression comme commenté ci-dessus.
<constant> : une constante numérique.
<label> : un label de programme.
<type> : tout type, (standard ou structuré).

> Autres

- Dans ce manuel, tous les sujets sont listés en ordre alphabétique pour réduire tout temps de recherche.
- La **valeur renvoyée** par les commandes est le plus souvent un Integer . Dans le cas contraire, le type de la valeur est indiqué dans la description (ligne de syntaxe) de la commande.
- Dans la documentation de PureBasic, les termes "commandes" et "fonctions" ont le même sens, independamment du fait que la fonction retourne une valeur ou non.

Chapitre 34

Global

Syntax

```
Global[.<type>] <variable[.<type>]> [= <expression>] [, ...]
```

Description

Une variable déclarée Global n'importe où dans le code est visible et utilisable n'importe où dans le code à l'exception du cas suivant : Les modules n'ont pas accès aux variables déclarées globales en dehors de ce module.
Global peut aussi être utilisé avec les tableaux , les listes et les maps .
Les instructions Protected et Static permettent de déclarer une variable locale dans une procédure qui a le même nom qu'une variable globale, sans risque de conflit.

Exemple : Avec des variables

```
1  Global a.l, b.b, c, d = 20
2
3  Procedure Change()
4    Debug a  ; Affiche 10 car la variable 'a' est globale
5  EndProcedure
6
7  a = 10
8  Change()
```

Exemple : Avec un tableau

```
1  Global Dim Tableau(2)
2
3  Procedure Change()
4    Debug Tableau(0)  ; Affiche 10 car le tableau 'Tableau()' est global
5  EndProcedure
6
7  Tableau(0) = 10
8  Change()
```

Exemple : Complexe avec un module : Tous les cas de figure

```
Global Var_GlobaleHorsModule = 12
Debug  Var_GlobaleHorsModule ; Affiche 12

DeclareModule Ferrari
  Global Var_GlobaleModule = 308
  #FerrariName$ = "458 Italia"
  Debug #FerrariName$ ; Affiche "458 Italia"

  ; Debug Var_GlobaleHorsModule ; Affiche une erreur car le compilateur veut créer une var globale qui exste déjà
  Debug Var_GlobaleModule       ; Affiche 308

  Declare CreateFerrari()
EndDeclareModule


; Tous les éléments de cette section seront privés. Tous les noms peuvent être utilisés sans conflit
;
; ------------------------------------------------------------------------------
Module Ferrari
  Debug Var_GlobaleHorsModule  ; Affiche 0  <== Regarder !
  Debug Var_GlobaleModule      ; Affiche 308

  Global Initialized = 205
  Debug Initialized ; Affiche 205

  Procedure Init()
    Debug Var_GlobaleHorsModule ; Affiche 0
    Debug Var_GlobaleModule     ; Affiche 308
    Debug "InitFerrari()"
  EndProcedure

  Procedure CreateFerrari()    ; Procédure publique (car déclarée dans la section 'DeclareModule')
    Init()
    Debug "CreateFerrari()"
    Debug Var_GlobaleHorsModule ; Affiche 0
    Debug Var_GlobaleModule     ; Affiche 308
  EndProcedure

EndModule


; Code principal
;
; ------------------------------------------------------------------------------
Procedure Init()               ; Procédure d'initialisation principale, n'entre pas en conflit avec la procédure
                               ; Init() du Module Ferrari

  Debug "Procédure init() du code principal."
  Debug Var_GlobaleHorsModule ; Affiche 12
  Debug Var_GlobaleModule     ; Affiche 0
EndProcedure

Init()
```

```
52
53  Ferrari::CreateFerrari()
54  Debug Ferrari::#FerrariName$ ; Affiche 458 Italia
55  Debug Var_GlobaleHorsModule  ; Affiche 12
56  ; Debug Var_GlobaleModule    ; Affiche une erreur car le compilateur
     veut créer une var globale qui exste déjà
57
58
59  UseModule Ferrari            ; Maintenant, importer tous les éléments
     publics directement dans le programme principal
60
61  CreateFerrari()
62  Debug #FerrariName$          ; Affiche 458 Italia
63  Debug Var_GlobaleHorsModule  ; Affiche 12
64  Debug Var_GlobaleModule      ; Affiche 308  <== Regarder !
65
66  UnuseModule Ferrari
67  ; Debug #FerrariName$        ; Affiche une erreur, introuvable
68  Debug Var_GlobaleHorsModule  ; Affiche 12
69  Debug Var_GlobaleModule      ; Affiche 0  <== Regarder !
```

Chapitre 35

Gosub : Return

Syntax

```
Gosub MonLabel

MonLabel:
  ...
Return
```

Description

Gosub signifie 'Go to sub routine', en français : 'Aller au sous-programme'. Un label doit être spécifié après Gosub pour que l'exécution du programme se poursuive à la position du label et jusqu'à ce qu'un Return soit rencontré. Lorsque le Return est atteint, le programme revient à l'instruction suivant le Gosub. Gosub est pratique pour construire rapidement un code structuré.
Les procédures sont une autre alternative pour la conception rapide d'un programme structuré. Gosub peut être seulement utilisé dans la partie principale du programme, pas dans les procedures .

Exemple

```
1   a = 1
2   b = 2
3   Gosub OperationComplexe
4   Debug a
5   End
6
7   OperationComplexe:
8     a = b*2+a*3+(a+b)
9     a = a+a*a
10  Return
```

Syntax

```
FakeReturn
```

Description

Lorsque vous souhaitez sauter d'un sous-programme à une autre partie du code extérieur au sous-programme (en utilisant une commande Goto), vous pouvez utiliser un FakeReturn qui simule un

Return sans l'effectuer réellement. Si vous n'utilisez pas ce dispositif, votre programme génèrera une erreur. Cette commande ne devrait pas avoir d'utilité car un programme bien conçu ne devrait pas utiliser de Goto . Toutefois, dans certains cas où la performance est critique, cela peut aider le programmeur.

Exemple

```
1  Gosub SousProgramme1
2
3  SousProgramme1:
4    ...
5    If a = 10
6      FakeReturn
7      Goto ProgrammePrincipal
8    EndIf
9  Return
```

Chapitre 36

Numéros et Identifiants (Handles)

Les numéros

Tous les objets crées sont identifiés par un numéro arbitraire (qui n'est pas l'identifiant de l'objet, comme défini ci-dessous). Dans ce manuel, les numéros s'écrivent sous la forme #Numéro (par exemple, chaque gadget créé a un numéro #Gadget).
Les numéros que vous leurs attribuez n'ont pas besoin d'être des constantes, mais ils doivent être uniques pour chaque objet de votre programme (une image peut avoir le même numéro qu'un gadget, parce que ce sont des types d'objets différents). Ces numéros sont utilisés pour accéder ultérieurement à ces objets dans votre programme.
Par exemple, les fonctions de gestion des évènements renvoient des numéros :

```
1  EventGadget()
2  EventMenu()
3  EventWindow()
```

Les identifiants

Tous les objets ont également un numéro unique qui leur est attribué par le système. Ces numéros uniques sont appelés identifiants. Parfois, une fonction PureBasic n'a pas besoin du numéro comme paramètre, mais de l'identifiant. Dans ce manuel, ce genre de besoin est indiqué, à l'aide du suffixe ID.

Exemple

```
1  ImageGadget(#Gadget, x, y, Largeur, Hauteur, ImageID [, Options])
2  ; #Gadget doit être le numéro que vous voulez attribuer au Gadget
3  ; ImageID doit être l'IDentifiant (numéro unique) de l'image.
```

Pour obtenir l'identifiant d'un objet, il y a des fonctions spéciales telles que :

```
1  FontID()
2  GadgetID()
3  ImageID()
4  ThreadID()
5  WindowID()
```

La plupart des fonctions qui créent ces objets renvoient cet identifiant comme résultat, si tout s'est bien passé. Ceci est seulement le cas si #PB_Any n'a pas été utilisé pour créer l'objet. Si #PB_Any est

utilisé, ces fonctions renvoient le numéro de l'objet qui a été attribué par PB pour elles, mais pas l'identifiant.

Exemple

```
1   GadgetHandle = ImageGadget(...)
```

Chapitre 37

If : Else : EndIf

Syntax

```
If <expression>
  ...
[ElseIf <expression>]
  ...
[Else]
  ...
EndIf
```

Description

La structure If est utilisée pour effectuer des tests et/ou changer le déroulement du programme selon le résultat (vrai ou faux) du test. ElseIf est utilisé pour produire un nombre quelconque de tests additionnels si le premier n'a pas eu un résultat vrai. La commande optionnelle Else est utilisée pour exécuter une séquence de code si tous les tests précédents de la structure ont échoué. Les structures If peuvent être imbriquées sans limite de profondeur.
Les court-circuits sont pris en charge, ce qui signifie que si un test est vrai, tous les tests suivants seront ignorés.

Exemple : Test simple

```
1  a = 5
2  If a = 10
3    Debug "a = 10"
4  Else
5    Debug "a <> 10"
6  EndIf
```

Exemple : Test multiple

```
1  a = 10
2  ; b = 15
3  c = 20
4
5  If (a = 10) And (b >= 10) Or (c = 20)
6    If b = 15
7      Debug "b = 15"
```

```
    Else
      PrintN("Autre possibilité")
    EndIf
  Else
    PrintN("Erreur de test")
  EndIf
```

Exemple : Court-circuit

```
Procedure AfficherSalut()
  Debug "Salut"
  ProcedureReturn 1
EndProcedure

a = 10
If a = 10 Or AfficherSalut() = 1 ; a est égal à 10, alors le deuxième test est totalement ignoré
  Debug "Succès"
Else
  Debug "Erreur"
EndIf
```

Chapitre 38

Import : EndImport

Syntax

```
Import "NomFichier"
  NomFonction.<type>(<parametre>, [, <parametre> [= ValeurDefaut]...])
   [As "NomSymbole"]
  ...
  NomVariable.<type> [As "NomSymbole"]
EndImport
```

Description

Pour les programmeurs chevronnés. Import : EndImport permet de déclarer facilement des fonctions et des variables externes à partir d'un fichier bibliothèque (.lib) ou objet (.obj).
Une fois déclarées, les fonctions importées sont directement disponibles dans le programme, comme n'importe quelle autre commande. Le compilateur ne vérifie pas si la fonction existe réellement dans le fichier importé, donc si une erreur survient, elle sera reportée par le linker.
Les fonctions importées remplacent avantageusement les commandes OpenLibrary() /CallFunction() : la vérification du type, le nombre de paramètres sont validés par le compilateur. De plus, contrairement à CallFunction(), une fonction importée peut gérer les types 'double', 'float' et 'quad' sans aucun problème.
Les paramètres en fin de fonction peuvent avoir une valeur par défaut (une expression constante est requise). Les paramètres ayant une valeur par défaut pourront être omis lors de l'appel de la fonction, la valeur par défaut de chaque paramètre manquant sera utilisée.
Par défaut, le symbole de la fonction importée est décoré suivant le schéma suivant : _NomFonction@tailleargument. Cela devrait fonctionner pour la plupart des fonctions qui utilise la convention standard d'appel (stdcall). Par contre, si la bibliothèque est écrite en C, et que les fonctions exportées ne sont pas stdcall, ImportC devra être utilisé à la place de Import. Dans ce cas, par défaut le symbole est préfixé comme ceci : _NomFonction.
Les pseudotypes peuvent être utilisés pour les paramètres, mais pas pour le type de la valeur de retour.

Remarques

En 64 bits, il n'y a qu'une seule convention d'appel, alors ImportC fait la même chose que Import.

Exemple

```
1  Import "User32.lib"
2
```

```
    ; Pas besoin d'utiliser 'As' car PureBasic préfixe la fonction correctement
    ; Nous définissons également le paramètre 'Options' comme facultatif, avec une valeur par défaut de 0 (quand il est omis)
    ;
    MessageBoxA(Fenetre.i, Corps$, Titre$, Options.i = 0)

    ; Cette fois PureBasic ne peut pas se débrouiller tout seul, car le
    ; nom de la fonction n'est pas le même que celui utilisé par le symbole
    ;
    BoiteDeMessage(Fenetre.i, Corps$, Titre$, Options.i) As "_MessageBoxA@16"

EndImport

MessageBoxA(0, "Salut", "le Monde") ; Nous ne précisons pas les options
BoiteDeMessage(0, "Salut", "le Monde 2", 0)
```

Exemple : Avec les pseudotypes

```
Import "User32.lib"

    ; Nous utilisons le pseudotype 'p-unicode' pour les paramètres chaîne, car
    ; MessageBoxW() est une fonction unicode seulement. Le compilateur va
    ; automatiquement convertir les chaînes en unicode quand nécessaire.
    ;
    MessageBoxW(Fenetre.l, Corps.p-unicode, Titre.p-unicode, Options.l = 0)

EndImport

;
MessageBoxW(0, "Salut", "le Monde")
```

Chapitre 39

Les fonctions 'Include'

Syntax

```
IncludeFile  "NomFichier"
XIncludeFile "NomFichier"
```

Description

IncludeFile permet d'inclure tout fichier source spécifié à l'endroit où la commande est située dans le code même si XIncludeFile a été appelé avant.
Cette commande est utile, si vous voulez séparer votre code source en plusieurs fichiers, afin de réutiliser un même code dans différents projets.

Exemple

```
1  IncludeFile "Sources\myfile.pb" ; Ce fichier sera inséré à cet
     endroit du code.
```

Syntax

```
XIncludeFile "NomFichier"
```

Description

XIncludeFile est similaire à IncludeFile, mais en évitant toutefois d'inclure le même fichier plusieurs fois.
Cette commande est utile, si vous voulez séparer votre code source en plusieurs fichiers, afin de réutiliser un même code dans différents projets.

Exemple

```
1  XIncludeFile "Sources\monfichier.pb" ; monfichier.pb sera inséré.
2  XIncludeFile "Sources\monfichier.pb" ; monfichier.pb sera ignoré dans
     tout appel supplémentaire.
```

Syntax

```
IncludeBinary "NomFichier"
```

Description

IncludeBinary inclut le fichier nommé à l'endroit où la commande est placée.

Exemple

```
1   IncludeBinary "Data\map.data"
```

Syntax

```
IncludePath "Chemin"
```

Description

IncludePath spécifie le chemin d'accès par défaut pour tous les fichiers inclus appelés après la commande. Très pratique pour l'inclusion de nombreux fichiers provenant d'un même répertoire.

Exemple

```
1   IncludePath  "Sources\Data"
2   IncludeFile  "Sprite.pb"
3   XIncludeFile "Music.pb"
```

Chapitre 40

L'assembleur en ligne x86

PureBasic permet d'inclure toute commande assembleur x86 (y compris les instructions MMX et FPU) directement dans le code source comme dans un vrai source assembleur. Vous pouvez également utiliser directement les variables ou pointeurs avec les instructions assembleur et pouvez intégrer plusieurs commandes assembleur sur une même ligne.
Sous Windows et Linux, PureBasic utilise **fasm** (http ://flatassembler.net), alors pour plus d'information à propos de la syntaxe, lire le guide fasm.
Sous OS X, PureBasic utilise **yasm** (http ://yasm.tortall.net/), alors pour plus d'information à propos de la syntaxe, lire le guide yasm.
Pour utiliser l'assembleur en ligne, utiliser les directives du compilateur EnableASM ou DisableASM .
Il est possible d'activer la coloration syntaxique ASM dans l'IDE avec "Activer la coloration des mots clés assembleur" compiler option .

Règles

Vous devez respecter plusieurs règles précises pour inclure de l'assembleur dans du code PureBasic :
- Les variables et les pointeurs doivent être déclarés **préalablement** à leur utilisation dans un contexte assembleur.
- Concernant les étiquettes (labels) : Les labels référencés en assembleur sont toujours en minuscule. Par exemple, le label 'ETIquette :' sera transformé en 'l_etiquette' dans le code assembleur.
Lorsque vous faites référence à un label purebasic dans le corps du programme (en dehors des procedures et des modules) avec des fonctions assembleurs, vous devez précéder son nom par un L minuscule et d'un caractère de soulignement comme ceci **'l_'**.
Si le label est défini dans une procédure , alors son préfixe est **'ll_nomprocedure_'**, en minuscules (ll comme local label) mais il n'est accessible qu'à l'intérieur de la procédure.
Pour accéder à un label dans un module avec des fonctions assembleurs, vous devez ajouter devant le label le préfixe **'nomdumodule.l_nomprocedure_'** écrit tout en minuscule.
Et enfin un label défini dans une procédure elle-même dans un module s'écrira **'nomdumodule.ll_nomprocedure_'** - Pour info, les listes commencent par 't_nomlist', les map par 'm_nommap' et les tableaux par 'a_nomtableau' est toujours en minuscule.

Exemple

```
1  DeclareModule MonModule
2    LabelDeclareModule: ;Son nom assembleur est
     monmodule.l_labeldeclaremodule:
3    Declare Init()
4  EndDeclareModule
5
6  Module MonModule
```

```
7     Procedure Init()
8       LabelModuleProcedure: ; Son nom assembleur est
      monmodule.ll_init_labelmoduleprocedure:
9       Debug "InitFerrari()"
10    EndProcedure
11
12    LabelModule1: ;Son nom assembleur est monmodule.l_labelmodule1:
13  EndModule
14
15   Procedure Test (*Pointer, Variable)
16    TokiSTART:  ;Son nom assembleur est ll_test_tokistart:
17
18    ! MOV dword [p.p_Pointer], 20
19    ! MOV dword [p.v_Variable], 30
20    Debug *Pointer  ;Son nom assembleur est p.p_Pointer
21    Debug Variable  ;Son nom assembleur est p.v_Variable
22  EndProcedure
23
24  VAR=1                         ;Son nom assembleur est v_VAR
25  *Pointt=AllocateMemory(10)  ;Son nom assembleur est p_Pointt
26
27  MonModule::Init()
28  Test(0,0)
29
30  Label1: ;Son nom assembleur est l_label1:
31  !jmp l_labelend ; Une instruction en assembler doit suivre les règles
     ci-dessus. Ici c'est l_nomelabel
32  ;...
33  LabelEnd: ;Son nom assembleur est l_labelend:
```

- Les erreurs dans une section asm ne sont pas reportées par PureBasic mais par Fasm. Vérifiez votre code si une telle erreur survient.
- Avec l'assembleur en ligne activé, vous ne pouvez pas utiliser les mots clés ASM pour les étiquettes (label).
- **Sur les processeurs x86, les registres volatiles sont : eax, ecx, edx, xmm0, xmm1, xmm2 et xmm3.** Tous les autres doivent être préservés.
- **Sur les processeurs x64, les registres volatiles sont : rax, rcx, rdx, r8, r9, xmm0, xmm1, xmm2 et xmm3.** Tous les autres doivent être préservés.
- Windows : un fichier aide-ASM peut être téléchargé ici. Si vous déplacez le fichier 'ASM.HLP' dans le dossier 'Help/' de PureBasic, vous aurez accès à l'aide sur les mots clés assembleur en appuyant sur F1. (Note : Ne fonctionne que si l'option 'assembleur en ligne' est activé).

Quand on utilise l'assembleur dans une procédure , il est utile de connaître les points suivants :
- Pour renvoyer directement la valeur du registre 'eax' (ou 'rax' sur x64) comme valeur de retour, il suffit d'utiliser ProcedureReturn, sans paramètre. Le contenu du registre 'eax' (ou 'rax' sur x64) restera inchangé et sera utilisé comme valeur de retour.

Exemple

```
1   Procedure.l MonTest()
2     !MOV eax, 45
3     ProcedureReturn  ; La valeur de retour sera 45
4   EndProcedure
5   Debug MonTest()
```

- Les variables locales en PureBasic sont directement indexées par rapport au registre de la **pile** (ESP) ce qui implique qu'un changement de la valeur de ce registre par une instruction assembleur (tel que PUSH, POP etc..) implique que la référence vers la variable ne sera plus correcte.

- Il est possible de passer directement une ligne complète à l'assembleur sans aucune modification en utilisant le caractère **'!'** en début de ligne. Ceci permet d'avoir un accès total aux fonctionnalités de l'assembleur. Pour faciliter l'accès aux variables locales, une notation a été mise en place : **'p.v_NomVariable'** pour une variable standard et **'p.p_NomPointeur'** pour un pointeur.

Exemple

```
1  Procedure Test(*Pointer, Variable)
2    ! MOV dword [p.p_Pointer], 20
3    ! MOV dword [p.v_Variable], 30
4    Debug *Pointer
5    Debug Variable
6  EndProcedure
7
8  Test(0, 0)
```

Chapitre 41

Interfaces

Syntax

```
Interface <nom> [Extends <nom>]
  <Methode[.<type>]()>
...
EndInterface
```

Description

Les Interfaces sont utilisées pour accéder facilement aux modules 'Orientés Objets' tels que les bibliothèques COM (Component Object Model) ou DirectX. Ce type de bibliothèques sont de plus en plus courantes sous Windows et les interfaces permettent une exploitation de ces fonctions sans impacts de performances. Les interfaces jettent aussi les bases pour une 'Programmation Orientée Object' (OOP en anglais) avec PureBasic mais de solides connaissances sont nécessaires pour en tirer parti (les interfaces n'ont pas été conçues pour ajouter une couche objet à PureBasic mais plutôt pour accéder facilement à des objets dejà conçus). La plupart des interfaces utilisées sous Windows sont déjà incluses dans les fichiers résidents 'Interfaces.res' et 'InterfaceDX.res', ce qui rend leur utilisation immédiate.
Le paramètre optionnel Extends permet d'étendre une interface sans avoir à dupliquer ses fonctions (ces 'fonctions' sont aussi communément appelées 'méthodes' dans les autres langages objet tels que C++ ou Java). Toutes les fonctions contenues dans l'interface étendue seront disponibles dans la nouvelle interface. C'est utile pour un héritage simple d'objets.
les tableaux (Arrays) peuvent être passés en paramètres à l'aide du mot clé Array, Les listes en utilisant le mot clé List et les maps en utilisant le mot clé Map.
Un type de retour peut être défini dans la déclaration de l'interface en ajoutant le type après la méthode.
La commande SizeOf peut être utilisée avec les interfaces pour déterminer la taille d'une interface et la commande OffsetOf peut être utilisée pour déterminer l'index d'une fonction dans une interface.
Les pseudotypes peuvent être utilisés pour les paramètres des fonctions de l'interface, mais pas pour le type de retour.
Note : Les concepts objets sont principalement appropriés pour les développeurs expérimentés et il n'est pas nécessaire de les maîtriser ou même les comprendre pour réaliser des applications ou jeux professionnels.

Exemple : Appel à une fonction objet

```
1  ; Nous allons considérer que vous voulez accéder à un objet externe
     (à l'intérieur d'une DLL par exemple).
2  ; Premièrement, déclarez son interface.
3  ;
4  Interface MonObjet
5    Deplacer(x,y)
```

```
    DeplacerF(x.f,y.f)
    Detruire()
  EndInterface

  ; Si 'CreationObjet()' est la fonction qui crée l'objet, à partir de la DLL,
  ; dont l'interface vient d'être définie...
  ; Alors créons le premier objet.
  ;
  Objet1.MonObjet = CreationObjet()

  ; Et le deuxième.
  ;
  Objet2.MonObjet = CreationObjet()

  ; Ensuite, les fonctions qui viennent d'être définies, peuvent être utilisées,
  ; afin d'agir sur l'objet désiré.

  Objet1\Deplacer(10, 20)
  Objet1\Detruire()

  Objet2\DeplacerF(10.5, 20.1)
  Objet2\Detruire()
```

Exemple : Utilisation de 'Extends'

```
  ; Définition d'une interface générique 'Cube'
  ;
  Interface Cube
    EnvoyerPosition()
    DefinirPosition(x)
    EnvoyerLargeur()
    DefinirLargeur(Largeur)
  EndInterface

  Interface CubeColorer Extends Cube
    EnvoyerCouleur()
    DefinirCouleur(Couleur)
  EndInterface

  Interface CubeTexturer Extends Cube
    EnvoyerTexture()
    DefinirTexture(TextureID)
  EndInterface

  ; Nous avons maintenant 3 interfaces pour chaque objet:
  ;
  ; - 'Cube' a les fonctions Envoyer/DefinirPosition() et Envoyer/DefinirLargeur()
  ; - 'CubeColorer' a les fonctions Envoyer/DefinirPosition(), Envoyer/DefinirLargeur() et Envoyer/DefinirCouleur()
  ; - 'CubeTexturer' a les fonctions Envoyer/DefinirPosition(),Envoyer/DefinirLargeur() et Envoyer/DefinirTexture()
  ;
```

Chapitre 42

Licenses for the PureBasic applications (without using 3D engine)

```
This program makes use of the following components:

----------------------------------------------------------------------
Component: MD5
----------------------------------------------------------------------

Copyright (C) 1991-2, RSA Data Security, Inc. Created 1991. All
rights reserved.

License to copy and use this software is granted provided that it
is identified as the "RSA Data Security, Inc. MD5 Message-Digest
Algorithm" in all material mentioning or referencing this software
or this function.

License is also granted to make and use derivative works provided
that such works are identified as "derived from the RSA Data
Security, Inc. MD5 Message-Digest Algorithm" in all material
mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either
the merchantability of this software or the suitability of this
software for any particular purpose. It is provided "as is"
without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this
documentation and/or software.

----------------------------------------------------------------------
Component: AES
----------------------------------------------------------------------

Optimized ANSI C code for the Rijndael cipher (now AES)

@authorVincent Rijmen <vincent.rijmen@esat.kuleuven.ac.be>
@authorAntoon Bosselaers <antoon.bosselaers@esat.kuleuven.ac.be>
@authorPaulo Barreto <paulo.barreto@terra.com.br>

This code is hereby placed in the public domain.
```

THIS SOFTWARE IS PROVIDED BY THE AUTHORS ''AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; *LOSS OF USE, DATA, OR PROFITS; OR* BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Component: SHA1

SHA-1 in C
By Steve Reid <steve@edmweb.com>
100% Public Domain

Component: zlib

Copyright (C) 1995-2012 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; *you must not* claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly — jloup@gzip.org
Mark Adler — madler@alumni.caltech.edu

Component: libpq

Portions Copyright (c) 1996-2011, PostgreSQL Global Development Group
Portions Copyright (c) 1994, The Regents of the University of California

Permission to use, copy, modify, and distribute this software and its documentation for any purpose, without fee, and without a written agreement
is hereby granted, provided that the above copyright notice and this paragraph and the following two paragraphs appear in all copies.

```
IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY
   FOR
DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES,
   INCLUDING
LOST PROFITS, ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS
DOCUMENTATION, EVEN IF THE UNIVERSITY OF CALIFORNIA HAS BEEN ADVISED OF
   THE
POSSIBILITY OF SUCH DAMAGE.

THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES,
INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED HEREUNDER
   IS
ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO
   OBLIGATIONS TO
PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

--------------------------------------------------------------------
Component: sqlite3
--------------------------------------------------------------------

The author disclaims copyright to this source code.  In place of
a legal notice, here is a blessing:

    May you do good and not evil.
    May you find forgiveness for yourself and forgive others.
    May you share freely, never taking more than you give.

--------------------------------------------------------------------
Component: libjpeg
--------------------------------------------------------------------

The authors make NO WARRANTY or representation, either express or
   implied,
with respect to this software, its quality, accuracy, merchantability,
   or
fitness for a particular purpose.  This software is provided "AS IS",
   and you,
its user, assume the entire risk as to its quality and accuracy.

This software is copyright (C) 1991-2012, Thomas G. Lane, Guido
   Vollbeding.
All Rights Reserved except as specified below.

Permission is hereby granted to use, copy, modify, and distribute this
software (or portions thereof) for any purpose, without fee, subject to
   these
conditions:
(1) If any part of the source code for this software is distributed,
   then this
README file must be included, with this copyright and no-warranty notice
unaltered; and any additions, deletions, or changes to the original
   files
must be clearly indicated in accompanying documentation.
(2) If only executable code is distributed, then the accompanying
documentation must state that "this software is based in part on the
   work of
the Independent JPEG Group".
```

(3) Permission for use of this software is granted only if the user accepts
full responsibility for any undesirable consequences; the authors accept
NO LIABILITY for damages of any kind.

These conditions apply to any software derived from or based on the IJG code,
not just to the unmodified library. If you use our work, you ought to
acknowledge us.

Permission is NOT granted for the use of any IJG author's name or company name
in advertising or publicity relating to this software or products derived from
it. This software may be referred to only as "the Independent JPEG Group's
software".

We specifically permit and encourage the use of this software as the basis of
commercial products, provided that all warranty or liability claims are
assumed by the product vendor.

Component: libpng
---

libpng versions 1.2.6, August 15, 2004, through 1.5.12, July 11, 2012, are
Copyright (c) 2004, 2006-2012 Glenn Randers-Pehrson, and are
distributed according to the same disclaimer and license as libpng-1.2.5
with the following individual added to the list of Contributing Authors:

   Cosmin Truta

libpng versions 1.0.7, July 1, 2000, through 1.2.5, October 3, 2002, are
Copyright (c) 2000-2002 Glenn Randers-Pehrson, and are
distributed according to the same disclaimer and license as libpng-1.0.6
with the following individuals added to the list of Contributing Authors:

   Simon-Pierre Cadieux
   Eric S. Raymond
   Gilles Vollant

and with the following additions to the disclaimer:

   There is no warranty against interference with your enjoyment of the
   library or against infringement. There is no warranty that our
   efforts or the library will fulfill any of your particular purposes
   or needs. This library is provided with all faults, and the entire
   risk of satisfactory quality, performance, accuracy, and effort is with
   the user.

libpng versions 0.97, January 1998, through 1.0.6, March 20, 2000, are
Copyright (c) 1998, 1999, 2000 Glenn Randers-Pehrson, and are
distributed according to the same disclaimer and license as libpng-0.96,
with the following individuals added to the list of Contributing

```
   Authors:

   Tom Lane
   Glenn Randers-Pehrson
   Willem van Schaik

libpng versions 0.89, June 1996, through 0.96, May 1997, are
Copyright (c) 1996, 1997 Andreas Dilger
Distributed according to the same disclaimer and license as libpng-0.88,
with the following individuals added to the list of Contributing
   Authors:

   John Bowler
   Kevin Bracey
   Sam Bushell
   Magnus Holmgren
   Greg Roelofs
   Tom Tanner

libpng versions 0.5, May 1995, through 0.88, January 1996, are
Copyright (c) 1995, 1996 Guy Eric Schalnat, Group 42, Inc.

For the purposes of this copyright and license, "Contributing Authors"
is defined as the following set of individuals:

   Andreas Dilger
   Dave Martindale
   Guy Eric Schalnat
   Paul Schmidt
   Tim Wegner

The PNG Reference Library is supplied "AS IS".  The Contributing Authors
and Group 42, Inc. disclaim all warranties, expressed or implied,
including, without limitation, the warranties of merchantability and of
fitness for any purpose.  The Contributing Authors and Group 42, Inc.
assume no liability for direct, indirect, incidental, special,
   exemplary,
or consequential damages, which may result from the use of the PNG
Reference Library, even if advised of the possibility of such damage.

Permission is hereby granted to use, copy, modify, and distribute this
source code, or portions hereof, for any purpose, without fee, subject
to the following restrictions:

  1. The origin of this source code must not be misrepresented.

  2. Altered versions must be plainly marked as such and must not
     be misrepresented as being the original source.

  3. This Copyright notice may not be removed or altered from
     any source or altered source distribution.

The Contributing Authors and Group 42, Inc. specifically permit, without
fee, and encourage the use of this source code as a component to
supporting the PNG file format in commercial products.  If you use this
source code in a product, acknowledgment is not required but would be
appreciated.

------------------------------------------------------------------------
```

Component: OpenJPEG

Copyright (c) 2002-2007, Communications and Remote Sensing Laboratory,
   Universite catholique de Louvain (UCL), Belgium
Copyright (c) 2002-2007, Professor Benoit Macq
Copyright (c) 2001-2003, David Janssens
Copyright (c) 2002-2003, Yannick Verschueren
Copyright (c) 2003-2007, Francois-Olivier Devaux and Antonin Descampe
Copyright (c) 2005, Herve Drolon, FreeImage Team
Copyright (c) 2006-2007, Parvatha Elangovan
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ‘AS
   IS’
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
   THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
   PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS
   BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
   THE
POSSIBILITY OF SUCH DAMAGE.

Component: libtiff

Copyright (c) 1988-1997 Sam Leffler
Copyright (c) 1991-1997 Silicon Graphics, Inc.

Permission to use, copy, modify, distribute, and sell this software and
its documentation for any purpose is hereby granted without fee,
   provided
that (i) the above copyright notices and this permission notice appear
   in
all copies of the software and related documentation, and (ii) the
   names of
Sam Leffler and Silicon Graphics may not be used in any advertising or
publicity relating to the software without the specific, prior written
permission of Sam Leffler and Silicon Graphics.

THE SOFTWARE IS PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND,
EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY

```
WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

IN NO EVENT SHALL SAM LEFFLER OR SILICON GRAPHICS BE LIABLE FOR
ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND,
OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY
   OF
LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE
OF THIS SOFTWARE.

-----------------------------------------------------------------------
Component: libmodplug (Module)
-----------------------------------------------------------------------

This source code is public domain.

-----------------------------------------------------------------------
Component: udis86 (OnError)
-----------------------------------------------------------------------

Copyright (c) 2002-2009 Vivek Thampi
All rights reserved.

Redistribution and use in source and binary forms, with or without
   modification,
are permitted provided that the following conditions are met:

    * Redistributions of source code must retain the above copyright
   notice,
      this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
   notice,
      this list of conditions and the following disclaimer in the
   documentation
      and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
   IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
   DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
   SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
   AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
   OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

-----------------------------------------------------------------------
Component: brieflz
-----------------------------------------------------------------------

Copyright (c) 2002-2004 by Joergen Ibsen / Jibz
```

```
All Rights Reserved

http://www.ibsensoftware.com/

This software is provided 'as-is', without any express or implied
warranty. In no event will the authors be held liable for any damages
arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:

 1. The origin of this software must not be misrepresented; you must
    not claim that you wrote the original software. If you use this
    software in a product, an acknowledgment in the product
    documentation would be appreciated but is not required.

 2. Altered source versions must be plainly marked as such, and must
    not be misrepresented as being the original software.

 3. This notice may not be removed or altered from any source
    distribution.

-------------------------------------------------------------------
Component: jcalg1
-------------------------------------------------------------------

This software is provided as-is, without warranty of ANY KIND, either
   expressed
or implied, including but not limited to the implied warranties of
   merchantability
and/or fitness for a particular purpose. The author shall NOT be held
   liable for
ANY damage to you, your computer, or to anyone or anything else, that
   may result
from its use, or misuse. Basically, you use it at YOUR OWN RISK.

-------------------------------------------------------------------
Component: lzma
-------------------------------------------------------------------

LZMA SDK is written and placed in the public domain by Igor Pavlov.

Some code in LZMA SDK is based on public domain code from another
   developers:
  1) PPMd var.H (2001): Dmitry Shkarin
  2) SHA-256: Wei Dai (Crypto++ library)

-------------------------------------------------------------------
Component: libzip
-------------------------------------------------------------------

Copyright (C) 1999-2008 Dieter Baron and Thomas Klausner
The authors can be contacted at <libzip@nih.at>

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
```

notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The names of the authors may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHORS ‘‘AS IS’’ AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Component: pcre

PCRE LICENCE

PCRE is a library of functions to support regular expressions whose syntax
and semantics are as close as possible to those of the Perl 5 language.

Release 8 of PCRE is distributed under the terms of the "BSD" licence, as
specified below. The documentation for PCRE, supplied in the "doc"
directory, is distributed under the same terms as the software itself. The data
in the testdata directory is not copyrighted and is in the public domain.

The basic library functions are written in C and are freestanding. Also
included in the distribution is a set of C++ wrapper functions, and a
just-in-time compiler that can be used to optimize pattern matching. These
are both optional features that can be omitted when the library is built.

THE BASIC LIBRARY FUNCTIONS

Written by: Philip Hazel
Email local part: ph10
Email domain: cam.ac.uk

University of Cambridge Computing Service,
Cambridge, England.

Copyright (c) 1997-2020 University of Cambridge

```
All rights reserved.


PCRE JUST-IN-TIME COMPILATION SUPPORT
-------------------------------------

Written by: Zoltan Herczeg
Email local part: hzmester
Email domain: freemail.hu

Copyright(c) 2010-2020 Zoltan Herczeg
All rights reserved.


STACK-LESS JUST-IN-TIME COMPILER
--------------------------------

Written by: Zoltan Herczeg
Email local part: hzmester
Email domain: freemail.hu

Copyright(c) 2009-2020 Zoltan Herczeg
All rights reserved.


THE C++ WRAPPER FUNCTIONS
-------------------------

Contributed by: Google Inc.

Copyright (c) 2007-2012, Google Inc.
All rights reserved.


THE "BSD" LICENCE
-----------------

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
   met:

* Redistributions of source code must retain the above copyright notice,
this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.

* Neither the name of the University of Cambridge nor the name of Google
Inc. nor the names of their contributors may be used to endorse or
promote products derived from this software without specific prior
written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
   IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
   THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
   PURPOSE
```

ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.

Component: scintilla

License for Scintilla and SciTE

Copyright 1998-2003 by Neil Hodgson <neilh@scintilla.org>

All Rights Reserved

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation.

NEIL HODGSON DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL NEIL HODGSON BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Component: expat

Copyright (c) 1998, 1999, 2000 Thai Open Source Software Center Ltd
and Clark Cooper
Copyright (c) 2001, 2002, 2003, 2004, 2005, 2006 Expat maintainers.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,

TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Component: libogg

Copyright (c) 2002, Xiph.org Foundation

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

- Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.

- Neither the name of the Xiph.org Foundation nor the names of its
contributors may be used to endorse or promote products derived from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE FOUNDATION
OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Component: libvorbis

Copyright (c) 2002-2004 Xiph.org Foundation

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

- Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.

- Neither the name of the Xiph.org Foundation nor the names of its
contributors may be used to endorse or promote products derived from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT

```
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE FOUNDATION
OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

-----------------------------------------------------------------------
Component: neuquant
-----------------------------------------------------------------------

NeuQuant Neural-Net Quantization Algorithm Interface
-----------------------------------------------------

Copyright (c) 1994 Anthony Dekker

NEUQUANT Neural-Net quantization algorithm by Anthony Dekker, 1994.
See "Kohonen neural networks for optimal colour quantization"
in "Network: Computation in Neural Systems" Vol. 5 (1994) pp 351-367.
for a discussion of the algorithm.
See also  http://members.ozemail.com.au/~dekker/NEUQUANT.HTML

Any party obtaining a copy of these files from the author, directly or
indirectly, is granted, free of charge, a full and unrestricted
   irrevocable,
world-wide, paid up, royalty-free, nonexclusive right and license to
   deal
in this software and documentation files (the "Software"), including
   without
limitation the rights to use, copy, modify, merge, publish, distribute,
   sublicense,
and/or sell copies of the Software, and to permit persons who receive
copies from any such party to do so, with the only requirement being
that this copyright notice remain intact.


Modified to quantize 32bit RGBA images for the pngnq program.
Also modified to accept a numebr of colors arguement.
Copyright (c) Stuart Coyle 2004-2006



Rewritten by Kornel Lesinski (2009)
Euclidean distance, color matching dependent on alpha channel
and with gamma correction. code refreshed for modern
   compilers/architectures:
ANSI C, floats, removed pointer tricks and used arrays and structs.
```

Chapitre 43

Licenses for the 3D engine integrated with PureBasic

```
This program makes use of the following components:

------------------------------------------------------------------------
Component: OGRE
------------------------------------------------------------------------

OGRE (www.ogre3d.org) is made available under the MIT License.

Copyright (c) 2000-2012 Torus Knot Software Ltd

Permission is hereby granted, free of charge, to any person obtaining a
    copy
of this software and associated documentation files (the "Software"),
    to deal
in the Software without restriction, including without limitation the
    rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or
    sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included
    in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
    IN
THE SOFTWARE.

------------------------------------------------------------------------
Component: CEGUI
```

Copyright (C) 2004 - 2006 Paul D Turner & The CEGUI Development Team

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Component: bullet

Copyright (c) 2003-2010 Erwin Coumans
http://continuousphysics.com/Bullet/

This software is provided 'as-is', without any express or implied warranty.
In no event will the authors be held liable for any damages arising from the use of this software.
Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely,
subject to the following restrictions:

1. The origin of this software must not be misrepresented; *you must not claim*
that you wrote the original software. If you use this software in a product, an
acknowledgment in the product documentation would be appreciated but is not
required.

2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

3. This notice may not be removed or altered from any source distribution.

Component: FreeImage

FreeImage Public License - Version 1.0

1. Definitions.

1.1. "Contributor" means each entity that creates or contributes to the creation
of Modifications.

1.2. "Contributor Version" means the combination of the Original Code, prior
Modifications used by a Contributor, and the Modifications made by that
particular Contributor.

1.3. "Covered Code" means the Original Code or Modifications or the combination
of the Original Code and Modifications, in each case including portions thereof.

1.4. "Electronic Distribution Mechanism" means a mechanism generally accepted in
the software development community for the electronic transfer of data.

1.5. "Executable" means Covered Code in any form other than Source Code.

1.6. "Initial Developer" means the individual or entity identified as the
Initial Developer in the Source Code notice required by Exhibit A.

1.7. "Larger Work" means a work which combines Covered Code or portions thereof
with code not governed by the terms of this License.

1.8. "License" means this document.

1.9. "Modifications" means any addition to or deletion from the substance or
structure of either the Original Code or any previous Modifications. When
Covered Code is released as a series of files, a Modification is:

A. Any addition to or deletion from the contents of a file containing Original
Code or previous Modifications.

B. Any new file that contains any part of the Original Code or previous
Modifications.

1.10. "Original Code" means Source Code of computer software code which is
described in the Source Code notice required by Exhibit A as Original Code, and
which, at the time of its release under this License is not already Covered Code
governed by this License.

1.11. "Source Code" means the preferred form of the Covered Code for making
modifications to it, including all modules it contains, plus any associated

interface definition files, scripts used to control compilation and installation
of an Executable, or a list of source code differential comparisons against
either the Original Code or another well known, available Covered Code of the
Contributor's choice. The Source Code can be in a compressed or archival form,
provided the appropriate decompression or de-archiving software is widely
available for no charge.

1.12. "You" means an individual or a legal entity exercising rights under, and
complying with all of the terms of, this License or a future version of this
License issued under Section 6.1. For legal entities, "You" includes any entity
which controls, is controlled by, or is under common control with You. For
purposes of this definition, "control" means (a) the power, direct or indirect,
to cause the direction or management of such entity, whether by contract or
otherwise, or (b) ownership of fifty percent (50%) or more of the outstanding
shares or beneficial ownership of such entity.

2. Source Code License.

2.1. The Initial Developer Grant. The Initial Developer hereby grants You a
world-wide, royalty-free, non-exclusive license, subject to third party
intellectual property claims:

(a) to use, reproduce, modify, display, perform, sublicense and distribute the
Original Code (or portions thereof) with or without Modifications, or as part of
a Larger Work; and

(b) under patents now or hereafter owned or controlled by Initial Developer, to
make, have made, use and sell ("Utilize") the Original Code (or portions
thereof), but solely to the extent that any such patent is reasonably necessary
to enable You to Utilize the Original Code (or portions thereof) and not to any
greater extent that may be necessary to Utilize further Modifications or
combinations.

2.2. Contributor Grant. Each Contributor hereby grants You a world-wide,
royalty-free, non-exclusive license, subject to third party intellectual
property claims:

(a) to use, reproduce, modify, display, perform, sublicense and distribute the
Modifications created by such Contributor (or portions thereof) either on an
unmodified basis, with other Modifications, as Covered Code or as part of a
Larger Work; and

(b) under patents now or hereafter owned or controlled by Contributor, to
Utilize the Contributor Version (or portions thereof), but solely to the extent
that any such patent is reasonably necessary to enable You to Utilize the
Contributor Version (or portions thereof), and not to any greater extent that
may be necessary to Utilize further Modifications or combinations.

3. Distribution Obligations.

3.1. Application of License. The Modifications which You create or to which You
contribute are governed by the terms of this License, including without
limitation Section 2.2. The Source Code version of Covered Code may be
distributed only under the terms of this License or a future version of this
License released under Section 6.1, and You must include a copy of this License
with every copy of the Source Code You distribute. You may not offer or impose
any terms on any Source Code version that alters or restricts the applicable
version of this License or the recipients' rights hereunder. However, You may
include an additional document offering the additional rights described in
Section 3.5.

3.2. Availability of Source Code. Any Modification which You create or to which
You contribute must be made available in Source Code form under the terms of
this License either on the same media as an Executable version or via an
accepted Electronic Distribution Mechanism to anyone to whom you made an
Executable version available; *and if made available via Electronic Distribution*
Mechanism, must remain available for at least twelve (12) months after the date
it initially became available, or at least six (6) months after a subsequent
version of that particular Modification has been made available to such
recipients. You are responsible for ensuring that the Source Code version

```
remains available even if the Electronic Distribution Mechanism is
    maintained by
a third party.

3.3. Description of Modifications. You must cause all Covered Code to
    which  you
contribute to contain  a file documenting  the changes You  made to
    create  that
Covered Code and the date of any change. You must include a prominent
    statement
that the  Modification is  derived, directly  or indirectly,  from
    Original Code
provided  by  the  Initial  Developer and  including  the  name  of the
     Initial
Developer in (a) the Source Code, and (b) in any notice in an
    Executable version
or related documentation in  which You describe the  origin or
    ownership of  the
Covered Code.

3.4. Intellectual Property Matters

(a)  Third  Party  Claims.  If  You  have  knowledge  that  a  party
    claims  an
intellectual  property  right  in  particular  functionality  or  code
    (or  its
utilization under this License),  you must include a  text file with
    the  source
code distribution titled "LEGAL" which describes the claim and the
    party  making
the claim in sufficient  detail that a recipient  will know whom to
    contact. If
you  obtain  such  knowledge  after  You  make  Your  Modification
    available as
described in Section 3.2, You shall promptly modify the LEGAL file in
    all copies
You make  available thereafter  and shall  take other  steps (such  as
    notifying
appropriate mailing lists or  newsgroups) reasonably calculated to
    inform those
who received the Covered Code that new knowledge has been obtained.

(b)  Contributor  APIs.  If  Your  Modification  is  an  application
    programming
interface  and You  own or  control patents  which are  reasonably
    necessary  to
implement that API, you must also include this information in the LEGAL
    file.

3.5. Required Notices. You must duplicate  the notice in Exhibit A in
    each file
of the Source Code, and this  License in any documentation for the
    Source Code,
where You describe recipients' rights  relating to Covered Code. If
    You created
one or  more Modification(s),  You may  add your  name as  a
    Contributor  to the
notice described in  Exhibit A. If  it is not  possible to put  such
    notice in a
```

particular Source Code file due to its structure, then you must include such
notice in a location (such as a relevant directory file) where a user would be
likely to look for such a notice. You may choose to offer, and to charge a fee
for, warranty, support, indemnity or liability obligations to one or more
recipients of Covered Code. However, You may do so only on Your own behalf, and
not on behalf of the Initial Developer or any Contributor. You must make it
absolutely clear than any such warranty, support, indemnity or liability
obligation is offered by You alone, and You hereby agree to indemnify the
Initial Developer and every Contributor for any liability incurred by the
Initial Developer or such Contributor as a result of warranty, support,
indemnity or liability terms You offer.

3.6. Distribution of Executable Versions. You may distribute Covered Code in
Executable form only if the requirements of Section 3.1-3.5 have been met for
that Covered Code, and if You include a notice stating that the Source Code
version of the Covered Code is available under the terms of this License,
including a description of how and where You have fulfilled the obligations of
Section 3.2. The notice must be conspicuously included in any notice in an
Executable version, related documentation or collateral in which You describe
recipients' rights relating to the Covered Code. You may distribute the
Executable version of Covered Code under a license of Your choice, which may
contain terms different from this License, provided that You are in compliance
with the terms of this License and that the license for the Executable version
does not attempt to limit or alter the recipient's rights in the Source Code
version from the rights set forth in this License. If You distribute the
Executable version under a different license You must make it absolutely clear
that any terms which differ from this License are offered by You alone, not by
the Initial Developer or any Contributor. You hereby agree to indemnify the
Initial Developer and every Contributor for any liability incurred by the
Initial Developer or such Contributor as a result of any such terms You offer.

3.7. Larger Works. You may create a Larger Work by combining Covered Code with
other code not governed by the terms of this License and distribute the Larger
Work as a single product. In such a case, You must make sure the requirements of
this License are fulfilled for the Covered Code.

4. Inability to Comply Due to Statute or Regulation.

If it is impossible for You to comply with any of the terms of this License with
respect to some or all of the Covered Code due to statute or regulation then You
must: (a) comply with the terms of this License to the maximum extent possible;
and (b) describe the limitations and the code they affect. Such description must
be included in the LEGAL file described in Section 3.4 and must be included with
all distributions of the Source Code. Except to the extent prohibited by statute
or regulation, such description must be sufficiently detailed for a recipient of
ordinary skill to be able to understand it.

5. Application of this License.

This License applies to code to which the Initial Developer has attached the
notice in Exhibit A, and to related Covered Code.

6. Versions of the License.

6.1. New Versions. Floris van den Berg may publish revised and/or new versions
of the License from time to time. Each version will be given a distinguishing
version number.

6.2. Effect of New Versions. Once Covered Code has been published under a
particular version of the License, You may always continue to use it under the
terms of that version. You may also choose to use such Covered Code under the
terms of any subsequent version of the License published by Floris van den Berg
No one other than Floris van den Berg has the right to modify the terms
applicable to Covered Code created under this License.

6.3. Derivative Works. If you create or use a modified version of this License
(which you may only do in order to apply it to code which is not already Covered
Code governed by this License), you must (a) rename Your license so that the
phrases "FreeImage", ‘FreeImage Public License", "FIPL", or any

confusingly
similar phrase do not appear anywhere in your license and (b) otherwise make it
clear that your version of the license contains terms which differ from the
FreeImage Public License. (Filling in the name of the Initial Developer,
Original Code or Contributor in the notice described in Exhibit A shall not of
themselves be deemed to be modifications of this License.)

7. DISCLAIMER OF WARRANTY.

COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS" BASIS, WITHOUT
WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT
LIMITATION, WARRANTIES THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE,
FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE
QUALITY AND PERFORMANCE OF THE COVERED CODE IS WITH YOU. SHOULD ANY COVERED CODE
PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER
CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION.
THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO
USE OF ANY COVERED CODE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

8. TERMINATION.

This License and the rights granted hereunder will terminate automatically if
You fail to comply with terms herein and fail to cure such breach within 30 days
of becoming aware of the breach. All sublicenses to the Covered Code which are
properly granted shall survive any termination of this License. Provisions
which, by their nature, must remain in effect beyond the termination of this
License shall survive.

9. LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING
NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL THE INITIAL DEVELOPER, ANY OTHER
CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED CODE, OR ANY SUPPLIER OF ANY OF SUCH
PARTIES, BE LIABLE TO YOU OR ANY OTHER PERSON FOR ANY INDIRECT, SPECIAL,
INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT
LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER

FAILURE OR
MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH
PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS
LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL
INJURY RESULTING FROM SUCH PARTY'S NEGLIGENCE TO THE EXTENT APPLICABLE LAW
PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR
LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THAT EXCLUSION AND
LIMITATION MAY NOT APPLY TO YOU.

10. U.S. GOVERNMENT END USERS.

The Covered Code is a "commercial item," as that term is defined in 48 C.F.R.
2.101 (Oct. 1995), consisting of "commercial computer software" and "commercial
computer software documentation," as such terms are used in 48 C.F.R. 12.212
(Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through
227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Code with
only those rights set forth herein.

11. MISCELLANEOUS.

This License represents the complete agreement concerning subject matter hereof.
If any provision of this License is held to be unenforceable, such provision
shall be reformed only to the extent necessary to make it enforceable. This
License shall be governed by Dutch law provisions (except to the extent
applicable law, if any, provides otherwise), excluding its conflict-of-law
provisions. With respect to disputes in which at least one party is a citizen
of, or an entity chartered or registered to do business in, the The Netherlands:
(a) unless otherwise agreed in writing, all disputes relating to this License
(excepting any dispute relating to intellectual property rights) shall be
subject to final and binding arbitration, with the losing party paying all costs
of arbitration; *(b) any arbitration relating to this Agreement shall be held in*
Almelo, The Netherlands; *and (c) any litigation relating to this Agreement shall*
be subject to the jurisdiction of the court of Almelo, The Netherlands with the
losing party responsible for costs, including without limitation, court costs

and reasonable attorneys fees and expenses. Any law or regulation which provides
that the language of a contract shall be construed against the drafter shall not
apply to this License.

12. RESPONSIBILITY FOR CLAIMS.

Except in cases where another Contributor has failed to comply with Section 3.4,
You are responsible for damages arising, directly or indirectly, out of Your
utilization of rights under this License, based on the number of copies of
Covered Code you made available, the revenues you received from utilizing such
rights, and other relevant factors. You agree to work with affected parties to
distribute responsibility on an equitable basis.

EXHIBIT A.

"The contents of this file are subject to the FreeImage Public License Version
1.0 (the "License"); you may not use this file except in compliance with the
License. You may obtain a copy of the License at
http://home.wxs.nl/~flvdberg/freeimage-license.txt

Software distributed under the License is distributed on an "AS IS" basis,
WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the
specific language governing rights and limitations under the License.

Component: FreeType

The FreeType Project LICENSE

2006-Jan-27

Copyright 1996-2002, 2006 by
David Turner, Robert Wilhelm, and Werner Lemberg

Introduction
============

The FreeType Project is distributed in several archive packages;
some of them may contain, in addition to the FreeType font engine,
various tools and contributions which rely on, or relate to, the
FreeType Project.

This license applies to all files found in such packages, and

which do not fall under their own explicit license. The license affects thus the FreeType font engine, the test programs, documentation and makefiles, at the very least.

This license was inspired by the BSD, Artistic, and IJG (Independent JPEG Group) licenses, which all encourage inclusion and use of free software in commercial and freeware products alike. As a consequence, its main points are that:

- We don't promise that this software works. However, we will be interested in any kind of bug reports. ('as is' distribution)
- You can use this software for whatever you want, in parts or full form, without having to pay us. ('royalty-free' usage)
- You may not pretend that you wrote this software. If you use it, or only parts of it, in a program, you must acknowledge somewhere in your documentation that you have used the FreeType code. ('credits')

We specifically permit and encourage the inclusion of this software, with or without modifications, in commercial products. We disclaim all warranties covering The FreeType Project and assume no liability related to The FreeType Project.

Finally, many people asked us for a preferred form for a credit/disclaimer to use in compliance with this license. We thus encourage you to use the following text:

```
"""
 Portions of this software are copyright © <year> The FreeType
 Project (www.freetype.org).  All rights reserved.
"""
```

Please replace <year> with the value from the FreeType version you actually use.

Legal Terms

0. Definitions

Throughout this license, the terms 'package', 'FreeType Project', and 'FreeType archive' refer to the set of files originally distributed by the authors (David Turner, Robert Wilhelm, and Werner Lemberg) as the 'FreeType Project', be they named as alpha, beta or final release.

'You' refers to the licensee, or person using the project, where 'using' is a generic term including compiling the project's source code as well as linking it to form a 'program' or 'executable'. This program is referred to as 'a program using the FreeType engine'.

This license applies to all files distributed in the original FreeType Project, including all source code, binaries and

documentation, unless otherwise stated in the file in its original, unmodified form as distributed in the original archive. If you are unsure whether or not a particular file is covered by this license, you must contact us to verify this.

The FreeType Project is copyright (C) 1996-2000 by David Turner, Robert Wilhelm, and Werner Lemberg. All rights reserved except as specified below.

1. No Warranty

THE FREETYPE PROJECT IS PROVIDED 'AS IS' WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL ANY OF THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY DAMAGES CAUSED BY THE USE OR THE INABILITY TO USE, OF THE FREETYPE PROJECT.

2. Redistribution

This license grants a worldwide, royalty-free, perpetual and irrevocable right and license to use, execute, perform, compile, display, copy, create derivative works of, distribute and sublicense the FreeType Project (in both source and object code forms) and derivative works thereof for any purpose; and to authorize others to exercise some or all of the rights granted herein, subject to the following conditions:

- Redistribution of source code must retain this license file ('FTL.TXT') unaltered; any additions, deletions or changes to the original files must be clearly indicated in accompanying documentation. The copyright notices of the unaltered, original files must be preserved in all copies of source files.

- Redistribution in binary form must provide a disclaimer that states that the software is based in part of the work of the FreeType Team, in the distribution documentation. We also encourage you to put an URL to the FreeType web page in your documentation, though this isn't mandatory.

These conditions apply to any software derived from or based on the FreeType Project, not just the unmodified files. If you use our work, you must acknowledge us. However, no fee need be paid to us.

3. Advertising

Neither the FreeType authors and contributors nor you shall use the name of the other for commercial, advertising, or promotional purposes without specific prior written permission.

We suggest, but do not require, that you use one or more of the following phrases to refer to this software in your documentation or advertising materials: 'FreeType Project', 'FreeType Engine', 'FreeType library', or 'FreeType Distribution'.

As you have not signed this license, you are not required to accept it. However, as the FreeType Project is copyrighted material, only this license, or another one contracted with the authors, grants you the right to use, distribute, and modify it. Therefore, by using, distributing, or modifying the FreeType Project, you indicate that you understand and accept all the terms of this license.

4. Contacts

There are two mailing lists related to FreeType:

- freetype@nongnu.org

  Discusses general use and applications of FreeType, as well as future and wanted additions to the library and distribution. If you are looking for support, start in this list if you haven't found anything to help you in the documentation.

- freetype-devel@nongnu.org

  Discusses bugs, as well as engine internals, design issues, specific licenses, porting, etc.

Our home page can be found at

http://www.freetype.org

Component: libogg

Copyright (c) 2002, Xiph.org Foundation

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- Neither the name of the Xiph.org Foundation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE FOUNDATION OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY

THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Component: libvorbis

Copyright (c) 2002-2004 Xiph.org Foundation

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

- Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.

- Neither the name of the Xiph.org Foundation nor the names of its
contributors may be used to endorse or promote products derived from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE FOUNDATION
OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Component: zlib

Copyright (C) 1995-2012 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied
warranty. In no event will the authors be held liable for any damages
arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not
   claim that you wrote the original software. If you use this software
   in a product, an acknowledgment in the product documentation would be
   appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not
   be
   misrepresented as being the original software.
3. This notice may not be removed or altered from any source

```
   distribution.

Jean-loup Gailly        Mark Adler
jloup@gzip.org          madler@alumni.caltech.edu

------------------------------------------------------------------------
Component: pcre
------------------------------------------------------------------------

PCRE LICENCE
------------

PCRE is a library of functions to support regular expressions whose
   syntax
and semantics are as close as possible to those of the Perl 5 language.

Release 8 of PCRE is distributed under the terms of the "BSD" licence,
   as
specified below. The documentation for PCRE, supplied in the "doc"
directory, is distributed under the same terms as the software itself.
   The data
in the testdata directory is not copyrighted and is in the public
   domain.

The basic library functions are written in C and are freestanding. Also
included in the distribution is a set of C++ wrapper functions, and a
just-in-time compiler that can be used to optimize pattern matching.
   These
are both optional features that can be omitted when the library is
   built.


THE BASIC LIBRARY FUNCTIONS
---------------------------

Written by: Philip Hazel
Email local part: ph10
Email domain: cam.ac.uk

University of Cambridge Computing Service,
Cambridge, England.

Copyright (c) 1997-2020 University of Cambridge
All rights reserved.


PCRE JUST-IN-TIME COMPILATION SUPPORT
-------------------------------------

Written by: Zoltan Herczeg
Email local part: hzmester
Email domain: freemail.hu

Copyright(c) 2010-2020 Zoltan Herczeg
All rights reserved.


STACK-LESS JUST-IN-TIME COMPILER
--------------------------------
```

Written by: Zoltan Herczeg
Email local part: hzmester
Email domain: freemail.hu

THE C++ WRAPPER FUNCTIONS

Contributed by: Google Inc.

THE "BSD" LICENCE

Component: MeshMagick

```
Permission is hereby granted, free of charge, to any person obtaining a
    copy
of this software and associated documentation files (the "Software"),
    to deal
in the Software without restriction, including without limitation the
    rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or
    sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included
    in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
    IN
THE SOFTWARE.

-------------------------------------------------------------------------
Component: OgreBullet
-------------------------------------------------------------------------

Copyright 2007 Paul "Tuan Kuranes" Cheyrou-Lagrèze.

This file is part of OgreBullet an integration layer between the OGRE 3D
graphics engine and the Bullet physic library.

Permission is hereby granted, free of charge, to any person obtaining a
    copy
of this software and associated documentation files (the "Software"),
    to deal
in the Software without restriction, including without limitation the
    rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or
    sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included
    in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
```

```
   FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
   IN
THE SOFTWARE.

-----------------------------------------------------------------------
Component: OgreProcedural
-----------------------------------------------------------------------

This source file is part of ogre-procedural

For the latest info, see http://code.google.com/p/ogre-procedural/

Copyright (c) 2010 Michael Broutin

Permission is hereby granted, free of charge, to any person obtaining a
   copy
of this software and associated documentation files (the "Software"),
   to deal
in the Software without restriction, including without limitation the
   rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or
   sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included
   in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
   OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
   THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
   FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
   IN
THE SOFTWARE.

-----------------------------------------------------------------------
Components:
  - OpenAL
  - OgreAL
  - zziplib
  - Hydrax
-----------------------------------------------------------------------

      GNU LIBRARY GENERAL PUBLIC LICENSE
           Version 2, June 1991

 Copyright (C) 1991 Free Software Foundation, Inc.
                    675 Mass Ave, Cambridge, MA 02139, USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

[This is the first released version of the library GPL.  It is
```

numbered 2 because it goes with version 2 of the ordinary GPL.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Library General Public License, applies to some specially designated Free Software Foundation software, and to any other libraries whose authors decide to use it. You can use it for your libraries, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library, or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link a program with the library, you must provide complete object files to the recipients so that they can relink them with the library, after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

Our method of protecting your rights has two steps: (1) copyright the library, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the library.

Also, for each distributor's protection, we want to make certain that everyone understands that there is no warranty for this free library. If the library is modified by someone else and passed on, we want its recipients to know that what they have is not the original version, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that companies distributing free software will individually obtain patent licenses, thus in effect transforming the program into proprietary software. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

Most GNU software, including some libraries, is covered by the ordinary
GNU General Public License, which was designed for utility programs. This
license, the GNU Library General Public License, applies to certain designated libraries. This license is quite different from the ordinary one; be sure to read it in full, and don't assume that anything in it is

the same as in the ordinary license.

  The reason we have a separate public license for some libraries is that
they blur the distinction we usually make between modifying or adding to a
program and simply using it. Linking a program with a library, without
changing the library, is in some sense simply using the library, and is
analogous to running a utility program or application program.
  However, in
a textual and legal sense, the linked executable is a combined work, a
derivative of the original library, and the ordinary General Public License
treats it as such.

  Because of this blurred distinction, using the ordinary General
Public License for libraries did not effectively promote software
sharing, because most developers did not use the libraries. We
concluded that weaker conditions might promote sharing better.

  However, unrestricted linking of non-free programs would deprive the
users of those programs of all benefit from the free status of the
libraries themselves. This Library General Public License is intended to
permit developers of non-free programs to use free libraries, while
preserving your freedom as a user of such programs to change the free
libraries that are incorporated in them. (We have not seen how to achieve
this as regards changes in header files, but we have achieved it as regards
changes in the actual functions of the Library.) The hope is that this
will lead to faster development of free libraries.

  The precise terms and conditions for copying, distribution and
modification follow. Pay close attention to the difference between a
"work based on the library" and a "work that uses the library". The
former contains code derived from the library, while the latter only
works together with the library.

  Note that it is possible for a library to be covered by the ordinary
General Public License rather than by this special one.

      GNU LIBRARY GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

  0. This License Agreement applies to any software library which
contains a notice placed by the copyright holder or other authorized
party saying it may be distributed under the terms of this Library
General Public License (also called "this License"). Each licensee is
addressed as "you".

  A "library" means a collection of software functions and/or data
prepared so as to be conveniently linked with application programs
(which use some of those functions and data) to form executables.

  The "Library", below, refers to any such software library or work
which has been distributed under these terms. A "work based on the
Library" means either the Library or any derivative work under
copyright law: that is to say, a work containing the Library or a

portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If

identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License.

Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also compile or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

c) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

d) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues),

conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any
particular circumstance, the balance of the section is intended to apply,
and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add
an explicit geographical distribution limitation excluding those countries,
so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Library General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our

decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

```
    You should have received a copy of the GNU Library General Public
    License along with this library; if not, write to the Free
    Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the library, if
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the
  library 'Frob' (a library for tweaking knobs) written by James Random
   Hacker.

  <signature of Ty Coon>, 1 April 1990
  Ty Coon, President of Vice

That's all there is to it!
```

Chapitre 44

Les macros

Syntax

```
Macro <nom> [(Parametre [, ...])]
  ...
EndMacro
```

Description

Le système de macro est une fonctionnalité très puissante, principalement utile pour les programmeurs chevronnés. Une macro est un bout de code quelconque (un mot clef, une ligne, plusieurs lignes) qui peut être inséré à n'importe quel endroit dans le code source en indiquant le nom de la macro. En ce sens, une macro est différente d'une procedure , car les procédures ne dupliquent pas leur code à chaque appel.
La déclaration Macro : EndMacro doit être effectuée avant le premier appel à la macro. Comme les macros seront complètement remplacées par le code correspondant au moment de la compilation, elles ne sont pas locales à une procédure.
Une macro ne peut pas avoir un type de retour, ou des paramètres typés, car cela n'a pas de sens.
Quand une macro a des paramètres, ils sont remplacés dans le code de la macro par l'expression littérale qui a été passée lors de l'appel. Aucune évaluation n'est faite à ce stade, ce qui est très important à comprendre : l'évaluation de la ligne ne commence uniquement lorsque toutes les macros trouvées sur cette ligne ont été traitées.
Les macros sont divisées en deux catégories : les simples (sans paramètres) et les complexes (avec paramètres, obligation d'utiliser les parenthèses pour les appeler). Quand aucun paramètre n'est spécifié, il est possible de remplacer n'importe quel mot avec un autre mot (ou une autre expression). Les macros ne peuvent pas être appelées de manière récursive.

Exemple : Macro simple

```
1
2   Macro MonNot
3     Not
4   EndMacro
5
6   a = 0
7   If MonNot a    ; Ici la ligne sera remplacée par : 'If Not a'
8     Debug "Ok"
9   EndIf
```

En utilisant les paramètres, il est possible de créer des macros très flexibles. Le caractère de concaténation '#' est disponible pour créer des nouveaux mots ou labels en combinant le code de la

macro et l'expression passée en paramètre (les espaces ne sont pas acceptées entre chaque mots devant être concaténés). Il est aussi possible de définir des valeurs par défaut pour chaque paramètre, pour qu'ils puissent être omis lors de l'appel de la macro.

Exemple : Macro avec paramètres

```
1  Macro BoiteDeMessageEnMajuscule(Titre, Corps)
2    MessageRequester(Titre, UCase(Corps), 0)
3  EndMacro
4
5  Texte$ = "le Monde"
6  BoiteDeMessageEnMajuscule("Salut", "-"+Texte$+"-") ; Ici la ligne
    sera remplacée comme ça :
7                                                    ;
    MessageRequester("Salut", UCase("-"+Texte$+"-"), 0)
```

Exemple : Macro avec paramètre par défaut

```
1  Macro BoiteDeMessageEnMajuscule(Titre, Corps = "Ah, aucun corps
    spécifié")
2    MessageRequester(Titre, UCase(Corps), 0)
3  EndMacro
4
5  BoiteDeMessageEnMajuscule("Salut") ; Ici la ligne sera remplacée
    comme ça :
6                                     ; MessageRequester("Salut",
    UCase("Ah, aucun corps spécifié"), 0)
```

Exemple : Macro et concaténation

```
1  Macro XCase(Type, Texte) ; renvoie le texte dans la casse du type
    spécifié
2    Type#Case(Texte)         ; Type U => MAJUSCULES
3  EndMacro                  ; Type L => minuscules
4
5  Debug XCase(U, "Salut") ; macro remplacée par UCase("Salut")
6  Debug XCase(L, "Salut") ; macro remplacée par LCase("Salut")
```

Exemple : Macro complexe sur plusieurs lignes

```
1  Macro Guillemet
2    "
3  EndMacro
4
5  Macro Assertion(Expression)
6    CompilerIf #PB_Compiler_Debugger  ; active uniquement l'assertion
    lorsque le débogueur est actif
7      If Expression
8        Debug "Assertion (Ligne " + #PB_Compiler_Line + ") : " +
    Guillemet#Expression#Guillemet
9      EndIf
10   CompilerEndIf
```

```
  EndMacro

  Assertion(10 <> 10) ; N'affichera rien
  Assertion(10 <> 15) ; Devrait afficher l'assertion
```

Syntax

```
UndefineMacro <nom>
```

Description

UndefineMacro permet d'annuler une macro définie précédemment, et de la redéfinir d'une manière différente.
Une fois que la macro a été annulée, elle n'est plus disponible.

Exemple : annulation de macro

```
  Macro Test
    Debug "1"
  EndMacro

  Test ; Appel de la macro

  UndefineMacro Test ; Annule la définition du macro, elle n'existe
   plus.

  Macro Test ; Maintenant, nous pouvons redéfinir la macro.
    Debug "2"
  EndMacro

  Test ; Appel de la macro
```

Syntax

```
MacroExpandedCount
```

Description

MacroExpandedCount permet d'obtenir le nombre de fois que la macro a été utilisée/appel. Il peut être utile de générer des identifiants uniques dans la même macro pour chaque utilisation (comme un label, le nom de la procédure, etc.)

Exemple : Comptage

```
  Macro Test
    Debug MacroExpandedCount
  EndMacro

  Test ; Appel de la macro
  Test ; Appel de la macro
  Test ; Appel de la macro
```

Chapitre 45

Pointeurs et accès mémoire

; ; ; ; ; ; ; ; ;Pointeurs

Exemple

```
  *MonEcran.Ecran = OpenScreen(0,320,200,8,0)
  mouseX = *MonEcran\SourisX ; La structure Ecran devant
   contenir un champ SourisX
\end{quote}
```

Il existe seulement trois méthodes valides pour fixer la valeur d'un pointeur :- Obtenir le résultat par une fonction (voir l'exemple ci-dessous)- Copier la valeur d'un autre pointeur- Trouver l'adresse d'une variable, procédure ou label (voir ci-dessous)Note : A l'inverse du C/C++, en PureBasic l' '*' fait **partie intégrante** du nom de la variable. Aussi ptr et *ptr sont deux variables bien distinctes.ptr est une variable (régulière) contenant une valeur, *ptr est une autre variable de type pointeur contenant une adresse.Pointeurs et taille mémoireComme un pointeur reçoit uniquement une adresse mémoire comme valeur, sa taille en mémoire sera celle qui permettra de représenter une adresse du processeur :- Sur un processeur 32-bit, les adresses sont représentées sur 32-bit, par conséquent un pointeur prendra 32-bit en mémoire (soit 4 octets comme une variable de type long').- Sur les processeurs 64-bit, les adresses sont représentées sur 64-bit, ce qui implique qu'un pointeur prendra 64-bit en mémoire (soit 8 octets comme une variable de type quad').C'est pour cette raison qu'un pointeur est une variable dite de type pointeur car son encombrement en mémoire sera lié à la capacité d'adressage mémoire du processeur.Il en découle qu'affecter un type natif à un pointeur (*Pointeur.l, *Pointeur.b) n'a aucun sens puisque l'encombrement mémoire d'un pointeur est imposé par celui d'une adresse et non par celui d'un type.Note :- A chaque fois qu'une adresse mémoire doit être stockée dans une variable, il faudrait le faire par l'intermédiaire d'un pointeur. Ceci garanti que l'adresse sera correctement représentée lors de la compilation du code que ce soit par un processeur 32-bit comme par un processeur 64-bit par exemple.Pointeurs et structuresAttacher une structure à un pointeur (par exemple *MonPoint.Point) permet d'accéder au contenu mémoire de chaque membre de la structure avec le caractère \ .

Exemple : Pointeurs et variables structurées

```
  Define Point1.Point, Point2.Point
  *PointCourant.Point = @Point1 ; Déclare le pointeur,
   l'associe à une structure et l'initialise avec
   l'adresse de Point1
  *PointCourant\x = 10          ; Assigne la valeur 10 à
   Point1\x
```

```
  *PointCourant.Point = @Point2 ; Récupère l'adresse de Point2
  *PointCourant\x = 20          ; Assigne la valeur 20 à Point2\x
  Debug Point1\x
  Debug Point2\x
\end{quote}
```

Pointeurs et tableaux

Attacher un tableau à un pointeur permet d'accéder au contenu mémoire de chaque cellules à travers son adresse.

Exemple : Pointeurs et tableaux

```
  Define Point1.Point, Point2.Point ; 2 Variables de type 'point' (type prédéfini dans PureBasic)
  Dim *Points.Point(1) ; Un tableau de 2 pointeurs et le tableau est de type 'Point'
  *Points(0) = @Point1 ; Le premier pointeur contient l'adresse de la variable Point1
  *Points(1) = @Point2 ; Le second pointeur contient l'adresse de la variable Point2

  *Points(0)\x = 10 ; Modification de la variable Point1 à travers le pointeur
  *Points(1)\x = 20 ; Idem avec Point2

  Debug Point1\x
  Debug Point2\x
\end{quote}
```

Les pointeurs permettent donc de se déplacer, de lire et d'écrire facilement en mémoire. De plus ils permettent aux programmeurs d'accéder à de grandes quantités de données sans coût supplémentaire suite à une duplication de ces données. Copier un pointeur est beaucoup plus rapide. Les pointeurs sont également disponibles dans les structures, pour plus d'informations, consultez le chapitre sur les structures .

Pointeurs et chaînes de caractères

Toutes les variables ont une taille fixe en mémoire (2 octets pour un Word, 4 octets pour un Long, etc) hormis les chaînes de caractères dont la longueur peut changer, ce qui fait qu'elles sont gérées différemment.
Ainsi, les champs d'une structure faisant référence à une chaîne de caractères stockent l'adresse mémoire où réside la chaîne de caractères et non la chaîne elle-même : ce sont des pointeurs vers des chaînes de caractères.

Exemple

```
  Texte$ = "Bonjour"
  *Texte = @Texte$ ;*Texte a pour valeur l'adresse où réside la chaîne de caractères en mémoire
  *Pointeur.String = @*Texte ; *Pointeur pointe sur *Texte
  Debug *Pointeur\s ; Lit la chaîne de caractères qui réside à l'adresse écrite en *Pointeur (c-a-d @Texte$)
\end{quote}
```

Arithmétiques des pointeurs
Il est possible d'effectuer des opérations arithmétiques sur les pointeurs en s'aidant de la commande SizeOf() .

Exemple

```
  Dim Tableau.Point(1) ; tableau de points

  *Pointeur.Point = @Tableau() ; Récupère l'adresse du tableau
  *Pointeur\x = 10 ; Modifie l'élément 0 du tableau
  *Pointeur\y = 15

  *Pointeur + SizeOf(Point) ; Pointe sur l'élément suivant

  *Pointeur\x = 7 ; Modifie l'élément 1 du tableau
  *Pointeur\y = 9

  ;Affiche le résultat
  For i = 0 To 1
    Debug Tableau(i)\x
    Debug Tableau(i)\y
  Next i
\end{quote}
```

Adresses des variables : '@'

Pour obtenir l'adresse d'une variable dans votre code, utilisez le symbole @. La raison la plus fréquente d'utiliser ce système est le transfert d'une variable de type structure à une procédure . Il faut passer un pointeur à la procédure car il est impossible de passer directement la structure comme argument.

Exemple

```
  Structure astruct
    a.w
    b.l
    c.w
  EndStructure

  Procedure SetB(*monpointeur.astruct)
    *monpointeur\b = 69
  EndProcedure

  Define.astruct mavariable
  mavariable\b = 0
  SetB( @mavariable )
  Debug mavariable\b
\end{quote}
```

Adresses des chaînes littérales

Pour obtenir l'adresse d'une chaîne de caractères littérales, utilisez le symbole @ devant elle. Les chaînes constantes sont aussi supportées.

Exemple

```
1   *Chaine = @"Test"
2   Debug PeekC(*Chaine) ; Affiche 84,
     qui est la valeur de la lettre T
     majuscule 'T'
3 \end{quote}
```

Adresses des procédures : '@'

En principe seuls les programmeurs avancés ont à connaître l'adresse d'une procédure. La raison la plus fréquente est de devoir négocier des échanges de bas niveau avec le système d'exploitation. Certains systèmes autorisent la mise en place de callbacks ou points d'ancrage (hooks) permettant au système d'exploitation de dialoguer avec le programme en étendant ainsi les capacités du système d'exploitation. L'adresse d'une procédure est accessible d'une manière similaire à une variable .

Exemple

```
1   Procedure WindowCB(WindowID.i,
     Message.i, wParam.i, lParam.i)
2     ; C'est ici que le
     traitement de votre callback
     sera effectué
3   EndProcedure
4
5   ; Un callback spécifique pour
     Windows permet de traiter les
     évènements sur les fenêtres
6   SetWindowCallback( @WindowCB()
     )
7 \end{quote}
```

Adresses des labels : '?'

Il peut également être utile de connaître l'adresse d'un label dans votre programme. Cela peut être le cas pour accéder au code ou aux données placées à cet endroit ou toute autre bonne raison qui peut vous venir à l'esprit. Pour trouver l'adresse d'un label dans votre programme, placez un '?' devant le nom du label.

Exemple

```
  Debug "Taille du fichier de données = " + Str(?endofmydata - ?mydata)

  DataSection
    mydata:
      IncludeBinary "somefile.bin"
    endofmydata:
\end{quote}
```

Chapitre 46

Guide de migration

Introduction

PureBasic est un langage de programmation moderne qui évolue rapidement. Il suit les changements technologiques et apporte une mise à jour de son jeu de fonction, pour le programmeur. Cela implique parfois de modifier ou de réécrire une partie du language. Alors nous essayons de faire ces modifications à minima et ce guide de migration aidera à mettre à jour vos codes sources d'une version vers une autre.
Si la stabilité est prioritaire sur les nouveautés, nous vous recommandons fortement de garder la version 'LTS' (Long Term Support/Support à Long Terme) de PureBasic, qui est présentée au public tous les 2 ans, et dont la maintenance active permet la corrections des bugs périodiquement.

Dernière version

migration de 5.50 vers 5.60
migration de 5.40 vers 5.50
migration de 5.30 vers 5.40

Version LTS

migration de 5.60 vers 5.72 LTS
migration de 5.20 LTS vers 5.40 LTS

Chapitre 47

Migration de PureBasic 5.20 LTS vers 5.40 LTS

Bibliothèque Billboard

AddBillboard() : code changé

```
; Ancien
AddBillboard(Billboard, BillboardGroup, x, y, z)

; Nouveau
Resultat = AddBillboard(BillboardGroup, x, y, z)
```

Bibliothèque Cipher

ExamineMD5Fingerprint() : code changé

```
; Ancien
ExamineMD5Fingerprint(#FingerPrint)

; Nouveau
UseMD5FingerPrint()
StartFingerprint(#FingerPrint, #PB_Cipher_MD5)
```

ExamineSHA1Fingerprint() : code changé

```
; Ancien
ExamineSHA1Fingerprint(#FingerPrint)

; Nouveau
UseSHA1FingerPrint()
StartFingerprint(#FingerPrint, #PB_Cipher_SHA1)
```

MD5FileFingerprint() : code changé

```
; Ancien
Resultat$ =
 MD5FileFingerprint(Fichier$)

; Nouveau
UseMD5FingerPrint()
Resultat$ =
 FileFingerprint(Fichier$,
 #PB_Cipher_MD5)
```

MD5Fingerprint() : code changé

```
; Ancien
Resultat$ =
 MD5Fingerprint(*Memoire,
 Taille)

; Nouveau
UseMD5FingerPrint()
Resultat$ =
 Fingerprint(*Memoire,
 Taille, #PB_Cipher_MD5)
```

SHA1FileFingerprint() : code changé

```
; Ancien
Resultat$ =
 SHA1FileFingerprint(Fichier$)

; Nouveau
UseSHA1FingerPrint()
Resultat$ =
 FileFingerprint(Fichier$,
 #PB_Cipher_SHA1)
```

SHA1Fingerprint() : code changé

```
; Ancien
Resultat$ =
 SHA1Fingerprint(*Memoire,
 Taille)

; Nouveau
UseSHA1FingerPrint()
Resultat$ =
 Fingerprint(*Memoire,
 Taille, #PB_Cipher_SHA1)
```

CRC32FileFingerprint() : code changé

```
; Ancien
Resultat =
 CRC32FileFingerprint(Fichier$)

; Nouveau
UseCRC32FingerPrint()
Resultat.l =
 Val("$"+FileFingerprint(Fichier$,
 #PB_Cipher_CRC32))
```

CRC32Fingerprint() : code changé

```
; Ancien
Resultat =
 CRC32Fingerprint(*Memoire,
 Taille)

; Nouveau
UseCRC32FingerPrint()
Resultat.l =
 Val("$"+Fingerprint(*Memoire,
 Taille, #PB_Cipher_CRC32))
```

NextFingerprint() : renommé

```
; Ancien
NextFingerprint(#FingerPrint,
 *Memoire, Taille)

; Nouveau
AddFingerprintBuffer(#FingerPrint,
 *Memoire, Taille)
```

Bibliothèque Mail

SendMail() : code changé si le paramètre 'Asynchronous' est utilisé

```
; Ancien
SendMail(#Mail, Smtp$,
 Port, 1)

; Nouveau
SendMail(#Mail, Smtp$,
 Port,
 #PB_Mail_Asynchronous)
```

Bibliothèque Packer

RemovePackFile() : supprimé
PackerEntrySize() :
#PB_Packer_CompressedSize supprimé pour les archives ZIP et 7z

Bibliothèque XML

CreateXMLNode() : code changé

```
; Ancien
Resultat =
 CreateXMLNode(NoeudParent...)
SetXMLNodeName(Noeud, Nom$)

; Nouveau
Resultat =
 CreateXMLNode(NoeudParent,
 Nom$...)
```

Bibliothèque Screen

AvailableScreenMemory() Supprimée car la nouvelle API ne le prend plus en charge. Elle renvoyait toujour un zéro de toute façon.

Bibliothèque Window

#PB_Event_SizeWindow et #PB_Event_MoveWindow ne sont plus en temps réel, utiliser BindEvent() () pour avoir l'action en temps réel.

Engine3D library

WorldCollisionAppliedImpulse() renvoie maintenant un float qui est l'impulsion appliquée.
GetX/Y/Z() ne sont plus supportées.

Divers

- Les étiquettes(labels) en DataSection dans les procédures sont maintenant des étiquettes (labels) locales.
- Les préfixes des étiquettes(labels) locales en ASM ont été changées de "l_" en "ll_", afin d'éviter tout problème avec les étiquettes principales.
- La constante #PB_LinkedList a été renommée en #PB_List pour une meilleure cohérence.

Chapitre 48

Migration de PureBasic 5.30 vers 5.40

Bibliothèque Cipher

ExamineMD5Fingerprint() : code changé

```
  ; Ancien
  ExamineMD5Fingerprint(#FingerPrint)

  ; Nouveau
  UseMD5FingerPrint()
  StartFingerprint(#FingerPrint,
   #PB_Cipher_MD5)
```

ExamineSHA1Fingerprint() : code changé

```
  ; Ancien
  ExamineSHA1Fingerprint(#FingerPrint)

  ; Nouveau
  UseSHA1FingerPrint()
  StartFingerprint(#FingerPrint,
   #PB_Cipher_SHA1)
```

MD5FileFingerprint() : code changé

```
  ; Ancien
  Resultat$ =
   MD5FileFingerprint(Fichier$)

  ; Nouveau
  UseMD5FingerPrint()
  Resultat$ =
   FileFingerprint(Fichier$,
   #PB_Cipher_MD5)
```

MD5Fingerprint() : code changé

```
  ; Ancien
  Resultat$ =
   MD5Fingerprint(*Memoire,
   Taille)

```

```
4   ; Nouveau
5   UseMD5FingerPrint()
6   Resultat$ =
     Fingerprint(*Memoire,
     Taille, #PB_Cipher_MD5)
```

SHA1FileFingerprint() : code changé

```
1   ; Ancien
2   Resultat$ =
     SHA1FileFingerprint(Fichier$)
3
4   ; Nouveau
5   UseSHA1FingerPrint()
6   Resultat$ =
     FileFingerprint(Fichier$,
     #PB_Cipher_SHA1)
```

SHA1Fingerprint() : code changé

```
1   ; Ancien
2   Resultat$ =
     SHA1Fingerprint(*Memoire,
     Taille)
3
4   ; Nouveau
5   UseSHA1FingerPrint()
6   Resultat$ =
     Fingerprint(*Memoire,
     Taille, #PB_Cipher_SHA1)
```

CRC32FileFingerprint() : code changé

```
1   ; Ancien
2   Resultat =
     CRC32FileFingerprint(Fichier$)
3
4   ; Nouveau
5   UseCRC32FingerPrint()
6   Resultat.l =
     Val("$"+FileFingerprint(Fichier$,
     #PB_Cipher_CRC32))
```

CRC32Fingerprint() : code changé

```
1   ; Ancien
2   Resultat =
     CRC32Fingerprint(*Memoire,
     Taille)
3
4   ; Nouveau
5   UseCRC32FingerPrint()
6   Resultat.l =
     Val("$"+Fingerprint(*Memoire,
     Taille, #PB_Cipher_CRC32))
```

NextFingerprint() : renommé

```
1   ; Ancien
2   NextFingerprint(#FingerPrint,
     *Memoire, Taille)
3
```

```
4   ; Nouveau
5   AddFingerprintBuffer(#FingerPrint,
     *Memoire, Taille)
```

Bibliothèque Mail

SendMail() : code changé si le paramètre 'Asynchronous' a été utilisé

```
1   ; Ancien
2   SendMail(#Mail, Smtp$,
     Port, 1)
3
4   ; Nouveau
5   SendMail(#Mail, Smtp$,
     Port,
     #PB_Mail_Asynchronous)
```

Bibliothèque Packer

RemovePackFile() : supprimé
PackerEntrySize() :
#PB_Packer_CompressedSize supprimé pour les archives ZIP et 7z

Bibliothèque Screen

AvailableScreenMemory() Supprimée car la nouvelle API ne le prend plus en charge. Elle renvoyait toujour un zéro de toute façon.

Engine3D library

WorldCollisionAppliedImpulse() renvoie maintenant un float qui est l'impulsion appliquée.
GetX/Y/Z() ne sont plus supportées.

Chapitre 49

Migration de PureBasic 5.40 vers 5.50

Bibliothèque Particule

ParticleVelocity() : code changé pour supporter la vélocité en cours.

```
1  ; Ancien
2  ParticleVelocity(#EmetteurParticule,
     Minimum, Maximum)
3
4  ; Nouveau
5  ParticleVelocity(#EmetteurParticule,
     Mode, Valeur)
```

Divers

La manipulation interne des chaînes de caractères est maintenant uniquement unicode.
Le mode ASCII n'est plus prise en charge de façon interne. La fonction *Resultat = Ascii(Chaine$) permet la manipulation de chaînes de caractères ASCII par l'utilisateur.

Chapitre 50

Migration de PureBasic 5.50 vers 5.60

Cipher library

Base64Encoder : Fonction renommée

```
1  ; Ancien
2  Base64Encoder()
3
4  ; Nouveau
5  Base64EncoderBuffer()
```

Base64Decoder : Fonction renommée

```
1  ; Ancien
2  Base64Decoder()
3
4  ; Nouveau
5  Base64DecoderBuffer()
```

Autres

La syntaxe autonome 'Define.b', pour changer le type par défaut des variables non typées, est désormais interdite et lèvera une 'Erreur de syntaxe'. Il suffit de supprimer cette instruction de vos code tout simplement (assurez-vous de typer vos variables non typées convenablement). Remarque : La syntaxe 'Define.b a, b, c' est toujours pris en charge.

Chapitre 51

Migration de PureBasic 5.60 to 5.72 LTS

Bibliothèque Math

Sign() : renvoie maintenant un entier au lieu d'un flottant.

```
1  ; Ancien
2  Var.d(f) = Sign(X)
3
4  ; Nouveau
5  Var.i = Sign(X)
```

Bibliothèque Window

PostEvent() : renvoie maintenant un résultat car il pouvait échouer dans certains cas extrêmes.

Divers

Sous MS WIndows, la police de caractères par défaut est passée de MS Shell Dlg à Segoe UI taille 9.

Chapitre 52

Module

Syntax

```
DeclareModule <nom>
  ...
EndDeclareModule

Module <nom>
  ...
EndModule

UseModule <nom>
UnuseModule <nom>
```

Description

Les modules sont un moyen facile d'isoler une morceau de code à partir d'un code principal, permettant ainsi la réutilisation d'un code et son partage sans risque de conflit de nom. Dans d'autres langages de programmation, les modules sont connus sous le nom : "espaces de noms". Un module doit avoir une section DeclareModule (qui est l'interface publique) et une section Module associée (qui est son implémentation). Seuls les éléments déclarés dans la section DeclareModule seront accessibles depuis l'extérieur du module. Tout le code de la section Module sera maintenu privé à ce module. Les éléments du code principal, comme les procédures, les variables, etc, ne seront pas accessibles à l'intérieur du module, même s'ils sont déclarés en global . Un module peut être considéré comme une boîte noire, une feuille de code vide où les noms des éléments ne peuvent pas entrer en conflit avec les éléments de même nom du code principal. Il est plus facile d'écrire du code, comme par exemple, on peut utiliser des noms simples qui peuvent être réutilisées au sein de chaque module sans risque de conflit.
Les éléments acceptés dans la section DeclareModule peuvent être les suivants : procédures (seule la déclaration des

procédures est autorisée), structures, macros, variables, constantes, énumérations, tableaux, listes, maps et les étiquettes (labels).
Pour accéder à un élément d'un module depuis l'extérieur, le nom du module doit être précisé suivie du séparateur **' : :'**. En spécifiant explicitement le nom du module, l'élément est disponible partout dans le code source, même dans un autre module. Tous les éléments de la section DeclareModule peuvent être automatiquement importés dans un autre module ou dans le code principal à l'aide de UseModule. Dans ce cas, si un nom de module est en conflit, les éléments du module ne seront pas importés et une erreur du compilateur sera levée. UnuseModule retire les éléments du module. UseModule n'est pas obligatoire pour accéder à un élément d'un module, mais le nom du module doit être spécifié.
Pour partager des informations entre les modules, un module commun peut être créé et utilisé par tous les modules qui en ont besoin. C'est la façon normale pour disposer de données globales disponibles pour tous les modules.
Toutes les commandes PureBasic, les structures et les constantes sont des éléments par défaut disponibles dans les modules. Par conséquent les éléments de modules ne peuvent pas avoir le même nom que les commandes internes de PureBasic, les structures ou les constantes.
Tout les codes à l'intérieur des sections DeclareModule et des sections Module sont exécutés comme n'importe quel autres codes lorsque le flux du programme atteint le module.
Si les mots clés Define, EnableExplicit, EnableASM sont utilisés dans un module, ils n'ont pas d'effet en dehors de ce module.
Quand les états Define, EnableExplicit, EnableASM sont utilisés à l'intérieur d'un module, ils n'ont pas d'effet en dehors du module, ni depuis l'extérieur du module.
Note : Les modules ne sont pas obligatoires dans PureBasic mais sont recommandés lors de la réalisation de grands projets. Ils aident à la maintenance du code, même si c'est légèrement plus verbeux. Avoir une section DeclareModule rend le module plus ou moins auto-documenté pour une réutilisation ou un partage.

Exemple

```
1
```

```
2   ; Tous les éléments de cette section seront disponibles de l'extérieur
3   ;
    ------------------------------------------------------------
4   DeclareModule Ferrari
5
6     #FerrariName$ = "458 Italia" ; Constante publique (accessible depuis l'extérieur du module)
7     Declare CreateFerrari()
          ; La Procédure sera publique (accessible depuis l'extérieur du module)
8
9   EndDeclareModule
10
11
12  ; Tous les éléments de cette section seront privés. Tous les noms peuvent être utilisés sans conflit
13  ;
    ------------------------------------------------------------
14  Module Ferrari
15
16    Global Initialized = #False
17
18    Procedure Init()
              ; Procedure privée Init() (inaccessible depuis l'extérieur du module)
19      If Initialized = #False
20        Initialized = #True
21        Debug "InitFerrari()"
22      EndIf
23    EndProcedure
24
25    Procedure CreateFerrari()
        ; Procédure publique (car déclarée dans la section 'DeclareModule')
26      Init()
27      Debug "CreateFerrari()"
28    EndProcedure
29
30  EndModule
31
32
33  ; Code principal
34  ;
    ------------------------------------------------------------
```

```
35  Procedure Init()
                ; Procédure d'initialisation principale, n'entre pas en conflit avec la procédure
36                                          ;
    Init() du Module Ferrari
37
38    Debug "Procédure init() du code principal."
39
40  EndProcedure
41
42  Init()
43
44  Ferrari::CreateFerrari()
45  Debug Ferrari::#FerrariName$
46
47  Debug "------------------------------"
48
49  UseModule Ferrari
              ; Maintenant, importer tous les éléments publics directement dans le programme principal
50
51  CreateFerrari()
52  Debug #FerrariName$
```

Exemple : Avec un module commun

```
1
2   ; Le module commun, qui sera utilisé par d'autres modules afin de partager des données
3   ;-------------------------------------------------------------------------------
4   DeclareModule Voitures
5     Global NbVoitures = 0
6   EndDeclareModule
7
8   Module Voitures
9   EndModule
10
11
12  ; Premier module de voiture
13  ;--------------------------
14  DeclareModule Ferrari
15  EndDeclareModule
16
17  Module Ferrari
18
19    UseModule Voitures
20    NbVoitures+1
21
```

```
22  EndModule
23
24
25  ; Second module de voiture
26  ; ------------------------
27  DeclareModule Porsche
28  EndDeclareModule
29
30  Module Porsche
31
32    UseModule Voitures
33    NbVoitures+1
34
35  EndModule
36
37
38  ; Code principal
39  ;
      ------------------------------------------------------------------------
40
41  Debug Voitures::NbVoitures
```

Chapitre 53

NewList

Syntax

```
NewList nom.<type>()
```

Description

NewList permet de déclarer une nouvelle liste. Chaque élément de la liste est alloué dynamiquement. Il n'y a pas de limite, ce qui permet d'en créer autant que nécessaire. Une liste peut avoir tout type standard ou structuré valide. Pour connaître toutes les commandes disponibles pour la gestion des listes, consultez la bibliothèque List .
Les listes sont toujours locales par défaut, donc pour accéder à partir d'une procédure à une liste définie dans le code source principal du programme, l'utilisation de Global ou Shared est requise. Il est également possible de passer une liste en paramètre d'une procédure à l'aide du mot-clef List.
Utilisez la commande Swap pour permuter rapidement un élément d'une liste avec une variable, ou un élément d'un tableau ou un élément d'une autre liste.

Exemple : Liste simple

```
1   NewList MaListe()
2
3   AddElement(MaListe())
4   MaListe() = 10
5
6   AddElement(MaListe())
7   MaListe() = 20
8
9   AddElement(MaListe())
10  MaListe() = 30
11
12  ForEach MaListe()
13    Debug MaListe()
14  Next
```

Exemple : Liste en paramètre d'une procédure

```
1   NewList Test()
2
3   AddElement(Test())
4   Test() = 1
5   AddElement(Test())
6   Test() = 2
7
8   Procedure DebugList(c, List
     ParameterList())
9
10    AddElement(ParameterList())
11    ParameterList() = 3
12
13    ForEach ParameterList()
14      MessageRequester("Liste",
     Str(ParameterList()))
15    Next
16
17  EndProcedure
18
19  DebugList(10, Test())
```

Chapitre 54

NewMap

Syntax

```
NewMap nom.<type>([Slots])
```

Description

NewMap permet de déclarer une nouvelle map, aussi connue sous le nom de table de hachage ou dictionnaire. Elle permet un accès rapide à un élément basé sur une clef. Chaque clef dans la map est unique, ce qui signifie qu'il n'est pas possible d'avoir deux éléments distincts avec la même clef. Il n'y a pas de limite sur le nombre d'élément que peut contenir une map. Une map peut être déclarée avec n'importe quel type basic ou structuré . Pour consulter toutes les commandes nécessaires à la manipulation des maps, voir la bibliothèque Map .
Quand une nouvelle clef est utilisée lors d'une affectation, un nouvel élément est automatiquement ajouté à la map. Si un autre élément avec la même clef était déjà présent dans la map, il sera remplacé par le nouveau. Une fois qu'un élément a été créé ou accédé, il devient l' élément courant de la map, et les accès à cet élément peuvent ensuite s'effectuer sans avoir à spécifier de clef. C'est très utile lors de l'utilisation d'une map structurée, car la recherche de l'élément ne sera plus nécéssaire pour accéder aux différents champs.
Les maps sont toujours locales par défaut, donc pour accéder à partir d'une procédure à une map définie dans le code source principal du programme, l'utilisation de Global ou Shared est requise. Il est également possible de passer une map en paramètre d'une procédure à l'aide du mot-clef Map.
Utilisez la commande Swap pour permuter rapidement un élément d'une map avec une variable, un élément d'un tableau ou un élément de la map.

Le paramètre optionnel 'Slots' défini le nombre de slots interne qui sera utilisé par la map pour effectuer le stockage des éléments. Plus il y a de slots en interne, plus l'accès à un élément sera rapide, mais plus la consommation mémoire sera importante. C'est un compromis dépendant du nombre d'éléments que la map contiendra au maximum et de la rapidité nécessaire à l'accès d'un élément. La valeur par défaut est 512. Ce paramètre n'a pas d'influence sur le nombre d'éléments que la map peut contenir.

Exemple : Map simple

```
1   NewMap Pays.s()
2
3   Pays("DE") = "Allemagne"
4   Pays("FR") = "France"
5   Pays("UK") = "United
     Kingdom"
6
7   Debug Pays("FR")
8
9   ForEach Pays()
10    Debug Pays()
11  Next
```

Exemple : Map en paramètre d'une procédure

```
1   NewMap Pays.s()
2
3   Pays("DE") = "Allemagne"
4   Pays("FR") = "France"
5   Pays("UK") = "United
     Kingdom"
6
7   Procedure DebugMap(Map
     ParameterMap.s())
8
9     ParameterMap("US") =
     "United States"
10
11    ForEach ParameterMap()
12      Debug ParameterMap()
13    Next
14
15  EndProcedure
16
17  DebugMap(Pays())
```

Exemple : Map structurée

```
1   Structure Voiture
2     Poids.l
3     Vitesse.l
4     Prix.l
5   EndStructure
6
7   NewMap Voitures.Voiture()
8
9   ; Ici, nous utilisons
     l'élément courant après
     l'insertion du nouvel
     élément
10  ;
11  Voitures("Ferrari
     F40")\Poids = 1000
12  Voitures()\Vitesse = 320
13  Voitures()\Prix = 500000
14
15  Voitures("Lamborghini
     Gallardo")\Poids = 1200
16  Voitures()\Vitesse = 340
17  Voitures()\Prix = 700000
18
19  ForEach Voitures()
20    Debug "Nom de la Voiture:
     "+MapKey(Voitures())
21    Debug "Poids:
     "+Str(Voitures()\Poids)
22  Next
```

Chapitre 55

Autres commandes

Syntax

```
Goto <label>
```

Description

Cette commande permet de transférer directement l'exécution du programme à l'emplacement d'un label. Soyez attentif en utilisant Goto car une mauvaise utilisation peut provoquer une fin anormale du programme...
Note : Pour sortir d'une boucle en toute sécurité, vous devez toujours utiliser Break à la place de Goto et ne l'utilisez pas dans un bloc Select/EndSelect, à moins que vous ayez les aptitudes nécessaire pour gérer la pile vous-même.

Syntax

```
End [CodeDeSortie]
```

Description

Termine et quitte le programme de manière correcte à n'importe quel endroit du code source. Le paramètre optionnel 'CodeDeSortie' peut être utilisé pour renvoyer un code d'erreur différent de 0 (valeur par défaut) lorsque le programme se termine (souvent utilisé par les programmes en mode console).
Le 'CodeDeSortie' peut-être récupéré dans un autre programme à l'aide de la commande ProgramExitCode() .

Syntax

```
Swap <expression>,
   <expression>
```

Description

Permute la valeur des deux expressions, de manière très rapide. Chaque <expression> doit être soit une variable , un élément (structuré ou non) de tableau , de liste ou de map et être d'un des types natifs : long (.l), quad (.q), string, etc.

Exemple : Permute des chaînes

```
1  Salut$ = "Salut"
2  Monde$ = "Monde"
3
4  Swap Salut$, Monde$
5
6  Debug Salut$+" "+Monde$
```

Exemple : Permute des tableaux à dimensions multiples

```
1  Dim Tableau1(5,5)
2  Dim Tableau2(5,5)
3  Tableau1(2,2) = 10      ;
     Initialise les tableaux
4  Tableau2(3,3) = 20
5
6  Debug Tableau1(2,2)     ;
     Affichera 10
7  Debug Tableau2(3,3)     ;
     Affichera 20
8
9  Swap Tableau1(2,2) ,
     Tableau2(3,3)     ;
     Permutation de 2 éléments
     entre 2 tableaux différents
10
11 Debug "Contenu des tableaux
     après permutation:"
12 Debug Tableau1(2,2)     ;
     Affichera 20
13 Debug Tableau2(3,3)     ;
     Affichera 10
```

Exemple : Permute différents éléments

```
1  Define a, b
2  Dim Tableau(4, 4)
3  NewList Liste1.Point()
4
5  ;-Initialise les variables,
     le tableau et la liste
6  a = 11
7  b = 12
8
```

```
  Tableau(1,1) = 21

  ;Ajoute un élément à la liste 1
  AddElement(Liste1())
  Liste1()\x = 31
  Liste1()\y = 32

  ;Permute un élément de la liste avec une variable
  Swap Liste1()\x, a
  ;Permute un élément de la liste avec un élément du tableau
  Swap Liste1()\y, Tableau(1,1)


  ;Affiche le résultat
  Debug a                 ; Affichera 31
  Debug b                 ; Affichera 12
  Debug Tableau(1,1)      ; Affichera 32
  Debug Liste1()\x        ; Affichera 11
  Debug Liste1()\y        ; Affichera 21
```

Chapitre 56

Procedures

Syntax

```
Procedure[.<type>]
    nom(<variable1[.<type>]>
    [, <variable2[.<type>]>,
    ...])
  ...
  [ProcedureReturn valeur]
EndProcedure
```

Description

Une procédure est une partie du code indépendante du programme principal. Elle peut avoir des paramètres et ses propres variables . En PureBasic, les procédures sont récursives et peuvent donc s'appeler elles-mêmes. Lors de chaque appel à la procédure, les variables locales sont automatiquement initialisées avec la valeur nulle.
Pour accéder aux variables du programme principal, ils faut les partager au préalable en utilisant les mots clefs Shared ou Global (voir aussi : Protected et Static).
Il est possible d'utiliser des paramètres ayant une valeur par défaut à condition que cette expression soit constante et placée en fin de liste des paramètres. Les paramètres ayant une valeur par défaut pourront être omis lors de l'appel de la procédure, la valeur par défaut de chaque paramètre manquant sera utilisée.
Par exemple : Procedure Calcul(num1, num2=0)
Calcul (2, 3)
Calcul (2)
Une procédure peut également recevoir en paramètre des listes , des maps et des tableaux à l'aide des mot-clefs List, Map et Array
Important : Pour les tableaux vous devrez indiquer le nombre de dimensions.
Par exemple : Procedure Calcul(Array num(3))

Ici num() est un tableau à 3 dimensions !
Une procédure peut renvoyer une valeur de retour. Pour cela, il faut deux choses. Premièrement, il faut définir le type de donnée de retour après Procedure qui peut être 'Integer', 'Long', 'Float', 'String' ou autre puis utiliser le mot clef ProcedureReturn pour déclencher le retour. Attention ProcedureReturn sort immédiatement d'une procédure, même si l'appel est placé à l'intérieur d'une boucle. Toutefois ProcedureReturn ne peut pas être utilisé pour renvoyer un tableau , une liste ou une map . Pour cela, passer le tableau, la liste ou la map en tant que paramètre dans la procédure.
Si aucune valeur n'est spécifiée pour ProcedureReturn, la valeur renvoyée sera indéterminée (voir assembleur en ligne pour plus d'information).
Les procédures peuvent également être appelées de manière asynchrone en utilisant les threads .
Les procédures peuvent également contenir une DataSection locale.
Note : Pour déclarer une procedure partagée dans une DLL, voir ProcedureDLL ou ProcedureCDLL .
Pour renvoyer une chaîne de caractères depuis une DLL, voir DLLs .
Note : Pour les programmeurs chevronnés, ProcedureC est disponible pour déclarer la procédure en utilisant la convention d'appel 'cdecl' au lieu de 'stdcall'.

Exemple : procédure qui renvoie une valeur numérique

```
1   Procedure Maximum(nb1, nb2)
2     If nb1>nb2
3       Resultat = nb1
4     Else
5       Resultat = nb2
6     EndIf
7
8     ProcedureReturn Resultat
9   EndProcedure
10
11  Resultat = Maximum(15,30)
12  Debug Resultat
13  End
```

Exemple : Procedure qui renvoie une chaîne de caractères

```
1   Procedure.s
     Attacher(Texte1$, Texte2$)
```

```
    ProcedureReturn Texte1$ + " " + Texte2$
EndProcedure

Resultat$ = Attacher("Coder avec ","PureBasic"))
Debug Resultat$
```

Exemple : Procédure avec un paramètre par défaut

```
Procedure a(a, b, c=2)
  Debug c
EndProcedure

a(10, 12) ; ou
a(10, 12, 15)
```

Exemple : Listes en paramètre

```
NewList Test.Point()

AddElement(Test())
Test()\x = 1
AddElement(Test())
Test()\x = 2

Procedure DebugList(c.l, List ParametreListe.Point())

  AddElement(ParametreListe())
  ParametreListe()\x = 3

  ForEach ParametreListe()
    MessageRequester("Liste", Str(ParametreListe()\x))
  Next

EndProcedure

DebugList(10, Test())
```

Exemple : Tableau en paramètre

```
Dim Tableau.Point(10, 15)

Tableau(0,0)\x = 1
Tableau(1,0)\x = 2

Procedure Test(c.l, Array ParametreTableau.Point(2))
  ; Attention, ici le tableau comporte 2 dimensions
```

```
7
8       ParametreTableau(1, 2)\x
        = 3
9       ParametreTableau(2, 2)\x
        = 4
10
11  EndProcedure
12
13  Test(10, Tableau())
14
15  MessageRequester("Tableau",
     Str(Tableau(1, 2)\x))
```

Exemple : Structure en paramètre

```
1   Structure Truc
2     a.l
3     b.l[2]            ; Tableau
      statique (Standard C) avec
      2 valeurs b[0] et b[1],
      non redimensionnable
4     Array c.l(3,3)   ; Tableau
      dynamique avec 16 valeurs
      de c(0,0) à c(3,3),
      redimensionnable avec
      ReDim()
5   EndStructure
6
7   MaVar.Truc
8
9   Procedure
      MaProcedure(*blabla.Truc)
10    *blabla\a = 5
11    *blabla\b[0] = 1
12    *blabla\b[1] = 2
13    *blabla\c(3,3) = 33
14  EndProcedure
15
16  MaProcedure(@MaVar)
17  Debug MaVar\a
18  Debug MaVar\b[0]
19  Debug MaVar\b[1]
20  Debug MaVar\c(3,3)
```

Exemple : Appeler une fonction par son nom

```
1     Prototype Function()
2
3   Runtime Procedure
      Function1()
4       Debug "J'appelle la
      Fonction1 par son nom";
5   EndProcedure
6
```

```
7   Procedure
     LaunchProcedure(Name.s)
8       Protected
     ProcedureName.Function =
     GetRuntimeInteger(Name +
     "()")
9       ProcedureName()
10  EndProcedure
11
12  LaunchProcedure("Function1")
     ; Affiche "J'appelle la
     Fonction1 par son nom"
```

Exemple : ProcedureC

```
1   ImportC ""
2     qsort(*base, num, taille,
     *ProcedureComparer)
3   EndImport
4
5   Dim valeurs.s(5)
6   valeurs(0) = "40"
7   valeurs(1) = "10"
8   valeurs(2) = "100"
9   valeurs(3) = "90"
10  valeurs(4) = "20"
11  valeurs(5) = "25"
12
13  ProcedureC.i
     Comparer(*a.String,
     *b.String)
14    ProcedureReturn Val(*a\s)
     - Val(*b\s)
15  EndProcedure
16
17  qsort(@valeurs(),
     ArraySize(valeurs()) + 1,
     SizeOf(String),
     @Comparer())
18
19  For n = 0 To 5
20    Debug valeurs(n)
21  Next n
```

Syntax

```
Declare[.<type>]
    nom(<variable1[.<type>]>
    [, <variable2[.<type>] [=
    ValeurParDefaut]>, ...])
```

Description

Dans certains cas, une procédure peut appeler une autre procédure qui n'a pas été déclarée avant sa propre définition. Ce cas peut se produire et provoquer une erreur de

compilation. Declare permet de traiter ce cas particulier en déclarant seulement l'en-tête de la procédure. Il est essentiel que les attributs de la fonction Declare et la déclaration réelle de la procédure soient identiques (type et ValeurParDefaut compris).
Note : Pour déclarer une procedure partagée dans une DLL voir DeclareDLL ou DeclareCDLL .
Note : Pour les programmeurs chevronnés, DeclareC est disponible pour déclarer la procédure en utilisant la convention d'appel 'cdecl' au lieu de 'stdcall'.

Exemple

```
1   Declare Maximum(Valeur1,
     Valeur2)
2
3   Procedure Traitement()
4     Resultat = Maximum(10, 2)
          ; A cet instant
     Maximum() n'est pas connu
     du compilateur.
5     ProcedureReturn Resultat
6   EndProcedure
7
8   Procedure Maximum(Valeur1,
     Valeur2)
9     If Valeur1 > Valeur2
10      Resultat = Valeur1
11    Else
12      Resultat = Valeur2
13    EndIf
14
15    ProcedureReturn Resultat
16  EndProcedure
17
18  Debug Traitement()
```

Chapitre 57

Protected

Syntax

```
Protected[.<type>]
    <variable[.<type>]> [=
    <expression>] [, ...]
```

Description

Protected permet de créer une variable locale dans une procédure. Elle supplante l'éventuelle variable globale du même nom pendant toute la procédure (contrairement à une variable locale classique non protégée). Une valeur par défaut peut être assignée à la variable. Protected peut aussi être utilisé avec les tableaux , les listes et les maps . La valeur de la variable locale sera réinitialisée à chaque appel de la procédure. Pour éviter cela, Static permet de déclarer une variable locale indépendante des variables globales tout en gardant sa valeur au fil des appels de la procédure.

Exemple : Avec une variable

```
1  Global a
2  a = 10
3
4  Procedure Change()
5    Protected a
6    a = 20
7  EndProcedure
8
9  Debug a ; Affichera 10 car
    la variable a été protégée.
```

Exemple : Avec un tableau

```
1  Global Dim Tableau(2)
2  Tableau(0) = 10
3
4  Procedure Change()
```

```
5       Protected Dim Tableau(2)
        ; Ce tableau est protégé,
        il sera local.
6       Tableau(0) = 20
7   EndProcedure
8
9   Change()
10  Debug Tableau(0) ;
        Affichera 10 car le
        tableau a été protégé.
```

Chapitre 58

Prototypes

Syntax

```
Prototype.<type>
    nom(<parametre>, [,
    <parametre> [=
    ValeurDefaut]...])
```

Description

Pour les programmeurs chevronnés. Un Prototype permet la déclaration d'un type particulier qui servira à appeler une fonction. Cela permet de faire facilement des pointeurs de fonctions, car ce type peut être affecté à une variable.
Cette fonctionnalité peut remplacer CallFunction() car elle présente quelques avantages : vérification du type de paramètre, du nombre de paramètres. Contrairement à CallFunction() , le prototype peut gérer les paramètres de types 'double', 'float' et 'quad' sans aucun problème. GetFunction() permet d'obtenir facilement le pointeur d'une fonction dans une bibliothèque.
Les paramètres en fin de prototype peuvent avoir une valeur par défaut (une expression constante est requise). Les paramètres ayant une valeur par défaut pourront être omis lors de l'appel du prototype, la valeur par défaut de chaque paramètre manquant sera utilisée.
Par défaut, la fonction utilisera la convention d'appel 'stdcall' sur x86, ou 'fastcall' sur x64. Si le pointeur de fonction appelle une fonction C utilisant la convention d'appel 'cdecl', PrototypeC est fortement conseillé.
Les pseudotypes peuvent être utilisés pour les paramètres, mais pas pour le type de retour.

Exemple

```
1  ;MessageRequester("Exemple","Prototype...")
    ; Décommenter cette ligne
    sous Windows XP
2  Prototype.i
    ProtoMessageBox(Fenetre.i,
    Corps$, Titre$, Options.i
    = 0)
3
4  If OpenLibrary(0,
    "User32.dll")
5
6     ; 'MsgBox' est une
    variable de type
    'ProtoMessageBox'
7     ;
8     MsgBox.ProtoMessageBox =
    GetFunction(0,
    "MessageBoxW")
9
10    MsgBox(0, "Hello",
    "World") ; Les options
    peuvent être omises
11 EndIf
```

Exemple : Avec des pseudotypes

```
1  ; Nous spécifions le
    pseudotype 'p-unicode'pour
    les 2 paramètres de type
    string
2  ; (texte à afficher et
    titre) car l'api
    MessageBoxW est une
    fonction unicode. Le
    compilateur
3  ; convertira
    automatiquement les
    chaînes ascii en unicode
    pour les besoins de la
    fonction
4  ;
5  ;MessageRequester("Exemple","Prototype...")
    ; Décommenter cette ligne
    sous Windows XP
6  Prototype.i
    ProtoMessageBoxW(Fenetre.i,
    Corps.p-unicode,
    Titre.p-unicode, Options.i
    = 0)
7
8  If OpenLibrary(0,
    "User32.dll")
9
10    ; 'MsgBox' est une
    variable de type
    'ProtoMessageBoxW'
11    ;
```

```
12      MsgBox.ProtoMessageBoxW =
        GetFunction(0,
        "MessageBoxW")
13
14      MsgBox(0, "Hello",
        "World") ; Les options
        peuvent être omises
15    EndIf
```

Chapitre 59

Pseudotypes

Description

Pour les programmeurs chevronnés. Les pseudotypes sont destinés à faciliter la programmation quand l'utilisation de bibliothèques externes nécessitant des types non supportés par PureBasic sont requises. Plusieurs types prédéfinis sont disponibles et permettent une conversion à la volée des types PureBasic vers ces types. Etant donné que ce ne sont pas des types normaux, leur notation est volontairement différente : un préfixe 'p-' (pour 'pseudo') fait partie du nom du type. Les pseudotypes disponibles sont :

```
  p-ascii: se comporte comme
   un type 'string', mais la
   chaîne de caractères sera
   toujours convertie en
           ascii lors de
   l'appel de la fonction,
   même si le programme est
   compilé en mode unicode
.
           C'est très utile
   pour appeler les fonctions
   d'une bibliothèque qui ne
   supporte pas
           l'unicode, dans un
   programme unicode.

  p-utf8: se comporte comme
   un type 'string', mais la
   chaîne de caractères sera
   toujours convertie en
          utf-8 lors de
   l'appel de la fonction.
   C'est très utile pour
   appeler les fonctions
   d'une bibliothèque qui ne
   supporte que
          l'utf-8 comme
   format de chaîne de
   caractères.
```

```
p-bstr: se comporte comme
 un type 'string', mais la
 chaîne de caractères sera
 toujours convertie en
        bstr lors de
 l'appel de la fonction.
 C'est très utile pour
 appeler les fonctions
 d'une bibliothèque qui ont
 besoin
        des chaînes de
 caractères bstr, comme les
 composants COM.

p-unicode: se comporte
 comme un type 'string',
 mais la chaîne de
 caractères sera toujours
 convertie en
          unicode lors de
 l'appel de la fonction,
 même si le programme est
 compilé en mode ascii.
          C'est très utile
 pour appeler les fonctions
 d'une bibliothèque qui ne
 supporte que
          l'unicode, dans
 un programme ascii.

p-variant: se comporte
 comme un type numérique,
 en ajustant l'appel de la
 fonction pour utiliser
 correctement
          un paramètre de
 type 'VARIANT'. C'est très
 utile pour appeler les
 fonctions d'une
 bibliothèque qui ont besoin
          des paramètres
 de type 'VARIANT', comme
 les composants COM.
```

Les pseudotypes peuvent être utilisés uniquement avec les prototypes , les interfaces et les fonctions importées . Ils ne font la conversion que si c'est nécessaire.

Exemple

```
1  ;MessageRequester("Exemple","Pseudotype...")
    ; Décommenter cette ligne
    sous Windows XP
2  Import "User32.lib"
3
4    ; Nous spécifions le
    pseudotype 'p-ascii'pour
```

```
    les 2 paramètres de type
    string
5   ; (texte à afficher et
    titre) car l'api
    MessageBoxA est une
    fonction ascii. Le
    compilateur
6   ; convertira
    automatiquement les
    chaînes ascii en ascii
    pour les besoins de la
    fonction
7   ;
8   MessageBoxA(Window.l,
    Corps.p-ascii,
    Titre.p-ascii, Options.i =
    0)
9
10  EndImport
11
12  ; L'exemple suivant
    fonctionnera correctement
    même si les chaînes
    internes PureBasic sont en
    unicode
13  ; car le compilateur se
    chargera automatiquement
    de la conversion ASCII.
14  ;
15  MessageBoxA(0, "Hello",
    "World")
```

Chapitre 60

Les objets PureBasic

Introduction

L'objectif de ce chapitre est d'assimiler la création et la manipulation des objets en PureBasic. Pour cette présentation nous allons utiliser l'objet Image , mais la même logique s'applique à tous les autres objets PureBasic. Quand nous voulons créer une image, nous avons deux possibilités : la méthode indexée et la méthode dynamique.

I. Les objets indexés

La manière indexée (statique) permet de référencer un objet à l'aide d'une valeur numérique que nous déterminons. Elle doit être comprise entre 0 et un nombre maximum qui va dépendre du type de l'objet (en principe entre 5000 et 64000). Ainsi, si vous utilisez la valeur 0 pour votre premier objet et la valeur 1000 pour votre deuxième objet, il y aura 1001 index de disponibles et 999 seront inutilisés, ce qui n'est pas très efficace (gâchis de mémoire vive). C'est pour cette raison qu'il faut autant que possible utiliser une indexation sequentielle, qui commence à 0. Si vous avez besoin d'une méthode plus flexible, il vous faudra probablement utiliser la méthode dynamique, décrite dans la section II. La méthode indexée offre plusieurs avantages :

- Manipulation plus facile, pas besoin de variables ni de tableaux .
- Manipulation 'Groupée' sans avoir à utiliser de tableaux supplémentaires.
- Utilisation des objets dans les procédures sans avoir à déclarer de globales (si on utilise une constante ou un nombre).

- Destruction automatique des objets précédents quand un index est réutilisé.

Les Enumérations sont fortement recommandées si vous souhaitez utiliser des constantes séquentielles pour identifier vos objets.

Exemple

```
1  CreateImage(0, 640, 480) ; Crée une image à l'index n°0 dans l'indexation des images
2  ResizeImage(0, 320, 240) ; Redimensionne l'image n°0
```

Exemple

```
1  CreateImage(2, 640, 480) ; Crée une image à l'index n°2 dans l'indexation des images
2  ResizeImage(2, 320, 240) ; Redimensionne l'image n°2
3  CreateImage(2, 800, 800) ; Crée une nouvelle image à l'emplacement n°2. L'ancienne image n°2 est désallouée automatiquement
```

Exemple

```
1  For k = 0 To 9
2    CreateImage(k, 640, 480) ; Crée 10 images, aux emplacements 0 à 9
3    ResizeImage(k, 320, 240) ; Recrée une nouvelle image à l'emplacement n°2. L'ancienne image n°2 est désallouée automatiquement
4  Next
```

Exemple

```
1  #ImageBackground = 0
2  #ImageButton     = 1
3
4  CreateImage(#ImageBackground, 640, 480) ; Crée une image à l'emplacement #ImageBackground (0)
5  ResizeImage(#ImageBackground, 320, 240) ; Redimensionne l'image 0 (ImageBackground)
```

```
6   CreateImage(#ImageButton
      , 800, 800) ; Crée une
      image (n°1)
```

II. Les objets dynamiques

Quelquefois, les objets indexés ne sont pas pratiques pour gérer des situations où l'on ne connaît pas à l'avance le nombre d'objets nécessaire à un instant donné. Pour cela PureBasic permet de créer très facilement des objets dynamiques. Les deux méthodes (indexée et dynamique) peuvent être utilisées en même temps sans risque de conflit. Pour créer un objet dynamique, il suffit de spécifier la constante #PB_Any à la place du numéro et un numéro dynamique sera retourné comme résultat de la fonction. Cette manière de gérer les objets se marie bien avec les listes , qui sont aussi un moyen dynamique de gérer des données.

Exemple

```
1   DynamicImage1 =
      CreateImage(#PB_Any, 640,
      480) ; Crée une image
      dynamiquement
2   ResizeImage(DynamicImage1,
      320, 240) ; Redimensionne
      l'image
```

Code complet d'exemple de gestion dynamique d'objets avec une liste chainée :

Présentation des différents objets PureBasic

Différents objets PureBasic (Windows, gadgets, sprites, etc) peuvent utiliser la même énumération de numéros d'objet et pas d'autres. Ainsi, chacun des objets suivants peuvent être énumérés en commençant à 0 (ou autre valeur) car PureBasic les gère par leur type :

- Database
- Dialog
- Entity
- File
- FTP
- Gadget (ScintillaGadget() inclus)
- Gadget3D
- Image
- Library
- Light
- Mail

- Material
- Menu (sauf les MenuItem() qui ne sont pas des objets)
- Mesh
- Movie
- Music
- Network
- Node
- Particle
- RegularExpression
- SerialPort
- Sound
- Sound3D
- Sprite
- StatusBar
- Texture
- ToolBar
- Window
- Window3D
- XML

Chapitre 61

Repeat : Until

Syntax

```
Repeat
  ...
Until <expression> [ou
   Forever]
```

Description

Cette fonction boucle jusqu'à ce que <expression> soit vrai. Si une boucle infinie est requise il est préférable d'utiliser le mot clef Forever à la place de Until.

Exemple

```
1  a=0
2  Repeat
3    a=a+1
4    Debug a
5  Until a>100
```

Ceci produira une boucle jusqu'à ce que "a" prenne une valeur strictement supérieure à 100, (il y aura donc 101 cycles).

Chapitre 62

Résidents

Description

Les résidents sont des fichiers pré-compilés qui sont chargés lors du démarrage du compilateur. Ils se trouvent dans le dossier des 'residents' dans le chemin d'installation de PureBasic. Un fichier résident doit avoir l'extension '.res' et peut contenir les éléments suivants : Structures , interfaces , prototypes , macros et constantes . Il ne peut pas contenir un code dynamique ni des procédures .
Lorsqu'un résident est chargé, tout son contenu est disponible pour le programme en cours de compilation. C'est pourquoi toutes les constantes intégrées comme #PB_Event_CloseWindow sont disponibles, elles sont dans le fichier 'PureBasic.res.
Toutes les structures et les constantes de l'API sont également dans un fichier résident. L'utilisation des résidents est un bon moyen pour stocker les macros, les structures et les constantes communes afin qu'elles soient disponibles pour tous les programmes.
Lors de la distribution d'une bibliothèque utilisateur, c'est aussi une bonne solution pour fournir les constantes et les structures nécessaires.
Pour créer un nouveau résident, le compilateur en ligne de commande doit être utilisé, car il n'y a pas d'option pour le faire à partir de l'IDE. Il est souvent nécessaire d'utiliser /IGNORERESIDENT et /CREATERESIDENT en même temps afin d'éviter des erreurs, car la version précédente du résident est chargée avant la création de la nouvelle.
Les résidents aident grandement à avoir une compilation et un démarrage du compilateur plus rapide, car toutes les informations sont stockées au format binaire. C'est beaucoup plus rapide que

l'analyse d'un fichier d'inclusion à chaque compilation.

Chapitre 63

Runtime

Syntax

```
Runtime Variable
Runtime #Constante
Runtime declaration
    Procedure()
Runtime declaration
    Enumeration
```

Description

Pour les programmeurs chevronnés.
Runtime est utilisé pour créer une liste dite ”runtime” (”en cours d’exécution”) pour avoir accès aux objets du programme en cours, comme les variables, les constantes et les procédures. Une fois compilé, un programme n’a plus d’étiquette (labels) ni de variable, ni de constantes ou de noms de procédure car tout est converti en code binaire. Runtime force le compilateur à ajouter une référence supplémentaire à chaque objet qui sera alors disponible à travers la bibliothèque Runtime . Les objets peuvent être manipulés à l’aide de leur référence qui est une chaîne de caractère, même lorsque le programme est compilé.
Pour illustrer l’utilisation d’un Runtime, jetez un coup d’oeil à la bibliothèque Dialog qui l’utilise pour accéder aux procédures d’événement associées à un gadget . Ici, le nom de la procédure à utiliser par le gestionnaire d’événements est spécifié dans le fichier XML (qui est un fichier texte), puis la bibliothèque Dialog utilise la fonction GetRuntimeInteger() pour retrouver l’adresse de la procédure pendant l’exécution du programme. Il n’est pas nécessaire de recompiler le programme pour la changer.
Une autre utilisation serait d’ajouter un petit langage de script en temps réel pour le programme, ce qui permet de modifier facilement les variables exposées, en utilisant les valeurs des constantes à

l'exécution. Alors que cela pourrait être fait manuellement par la construction d'une Map d'objets, le mot clé Runtime permet de le faire d'une manière standard et unifiée.

Exemple : Procedure

```
1  Runtime Procedure OnEvent()
2    Debug "OnEvent"
3  EndProcedure
4
5  Debug
     GetRuntimeInteger("OnEvent()")
     ; Affichera l'adresse de
     la procédure
```

Exemple : Enumeration

```
1  Runtime Enumeration
2    #Constante1 = 10
3    #Constante2
4    #Constante3
5  EndEnumeration
6
7  Debug
     GetRuntimeInteger("#Constante1")
8  Debug
     GetRuntimeInteger("#Constante2")
9  Debug
     GetRuntimeInteger("#Constante3")
```

Exemple : Variable

```
1  Define a = 20
2  Runtime a
3
4  Debug GetRuntimeInteger("a")
5  SetRuntimeInteger("a", 30)
6
7  Debug a ; La variable a été
     modifiée
```

Exemple : Appeler une fonction par son nom

```
1  Prototype Fonction()
2
3  Runtime Procedure
     Fonction1()
4      Debug "J'appelle la
     Fonction1 par son nom";
5  EndProcedure
6
7  Runtime Procedure
     Fonction2()
```

```
8       Debug "J'appelle la Fonction2 par son nom"
9   EndProcedure
10
11  Procedure LancerProcedure(Nom.s)
12      Protected ProcedureNom.Fonction = GetRuntimeInteger(Nom + "()")
13      ProcedureNom()
14  EndProcedure
15
16  LancerProcedure("Fonction1") ; Affiche "J'appelle la Fonction1 par son nom"
17  LancerProcedure("Fonction2") ; Affiche "J'appelle la Fonction2 par son nom"
```

Chapitre 64

Select : EndSelect

Syntax

```
Select <expression1>
  Case <expression> [,
   <expression> [<expression
   numerique> To <expression
   numerique>]]
     ...
  [Case <expression>]
     ...
  [Default]
     ...
EndSelect
```

Description

Select permet d'opérer des choix rapides. Le programme exécute <expression1> et retient la valeur en mémoire. Cette valeur est ensuite comparée à chacune des valeurs "Case <expression>" et s'il y a égalité, le code du bloc Case est exécuté pour quitter ensuite la structure Select. Case supporte les valeurs multiples ainsi que les intervalles à l'aide du mot-clef To (seulement pour les intervalles numériques). Si aucune des valeurs Case n'est vraie, alors le code du bloc Default, (s'il est spécifié) est exécuté. Note : Select accepte les nombres à virgules (float) comme <expression1>, mais ils seront arrondis à l'entier inférieur (les comparaisons ne se font que sur des nombres entiers).

Exemple

```
1   Valeur = 2
2
3   Select Valeur
4     Case 1
5       Debug "Valeur = 1"
```

```
6
7       Case 2
8         Debug "Valeur = 2"
9
10      Case 20
11        Debug "Valeur = 20"
12
13      Default
14        Debug "Je ne sais pas"
15    EndSelect
```

Exemple : Cas multiples et intervalles

```
1   Valeur = 2
2
3   Select Valeur
4     Case 1, 2, 3
5       Debug "Valeur est 1, 2
      ou 3"
6
7     Case 10 To 20, 30, 40 To
      50
8       Debug "Valeur est entre
      10 et 20, égale à 30 ou
      entre 40 et 50"
9
10    Default
11      Debug "Je ne sais pas"
12
13  EndSelect
```

Chapitre 65

Utiliser plusieurs versions de PureBasic avec Windows

Introduction

Il est possible d'installer plusieurs versions de PureBasic sur votre disque dur. C'est utile pour finir un projet avec une ancienne version de PureBasic, tout en commençant le développement d'un nouveau projet avec une version plus récente de PureBasic.

Comment procéder

Créez différents répertoires comme par exemple "PureBasic_v3.94" et "PureBasic_v5" et installez les versions de PureBasic correspondant à chaque répertoire.
Quand un des fichiers "PureBasic.exe" est démarré, il associe tous les fichiers ".pb" à la version de PureBasic correspondante. Aussi quand un code source est chargé en double cliquant sur son fichier, c'est la version de PureBasic associée en cours qui sera démarrée. PureBasic ne changera rien, qui pourrait affecter les autres versions de PureBasic dans les autres répertoires.
Pour éviter l'association automatique des fichiers ".pb" au démarrage de l'IDE, un raccourci peut être créé depuis PureBasic.exe avec "/NOEXT" comme paramètre. La ligne de commande optionnelle depuis l'IDE est décrite ici .
Note : Depuis PureBasic 4.10, les paramètres de l'IDE ne sont plus enregistrés dans le répertoire 'PureBasic' mais dans le répertoire %APPDATA%\PureBasic. Pour utiliser le même fichier de configuration avec différentes versions de Purebasic,

utilisez les interrupteurs /P /T et /A. En outre, le commutateur /PORTABLE met tous les fichiers dans le répertoire PureBasic et désactive la création de l'extension '.pb'.

Chapitre 66

Shared

Syntax

```
Shared <variable> [, ...]
```

Description

Shared permet de rendre une variable , un tableau , une liste ou une map non global accessible depuis une procédure . Quand Shared est utilisé avec un tableau, une liste ou une map, seul le nom suivi de '()' doit être spécifié.

Exemple : Avec une variable

```
1  a = 10
2
3  Procedure Change()
4    Shared a
5    a = 20
6  EndProcedure
7
8
9  Change()
10 Debug a   ; Affichera 20,
    car la variable est
    partagée.
```

Exemple : Avec un tableau et une liste

```
1  Dim Array(2)
2  NewList List()
3  AddElement(List())
4
5  Procedure Change()
6    Shared Array(), List()
7    Array(0) = 1
8    List() = 2
9  EndProcedure
10
11 Change()
```

```
12 | Debug Array(0)   ; Affichera 1, car le tableau est partagé.
13 | Debug List()     ; Affichera 2, car la liste est partagée.
```

Chapitre 67

Static

Syntax

```
Static[.<type>]
    <variable[.<type>]> [=
    <constant expression>] [,
    ...]
```

Description

Static permet de créer des variables locales persistantes dans une procedure . Les variables statiques sont prioritaires sur les variables globales , ce qui implique qu'une variable globale sera ignorée dans une procédure si une variable statique portant le même nom est déjà déclarée. La valeur de la variable statique n'est pas réinitialisée à chaque appel de la procédure : c'est donc un bon moyen pour avoir une variable globale affectée à une seule procédure. Static peut aussi être utilisé avec les tableaux , les listes et les maps . Lors de la déclaration d'un tableau static, ses paramètres doivent être une valeur constante.

Exemple : Avec une variable

```
1   Global a
2   a = 10
3
4   Procedure Change()
5     Static a
6     a+1
7     Debug "Dans la Procédure:
      "+Str(a) ; Affichera 1, 2,
      3 car la variable
      s'incrémente à chaque
      appel de la procédure.
8   EndProcedure
9
10  Change()
11  Change()
12  Change()
```

```
13  Debug a ; Affichera 10, car une variable 'static' n'affecte pas une variable 'global'.
```

Exemple : Avec un tableau

```
1   Global Dim Tableau(2)
2   Tableau(0) = 10
3
4   Procedure Change()
5     Static Dim Tableau(2)
6     Tableau(0)+1
7     Debug "Dans la Procédure: "+Str(Tableau(0)) ; Affichera 1, 2, 3 car la valeur du champ du tableau s'incrémente à chaque appel de la procédure.
8   EndProcedure
9
10  Change()
11  Change()
12  Change()
13  Debug Tableau(0) ; Affichera 10, car un tableau 'static' n'affecte pas un tableau 'global'.
```

Exemple : Avec plusieurs procedures

```
1   Procedure Foo()
2     Static x = 100  ; La déclaration et l'affectation sont effectuées une seule fois, au lancement du programme.
3
4     Debug x
5     x + 1
6   EndProcedure
7
8   Foo() ; Affiche 100
9   Foo() ; Affiche 101
10  Foo() ; Affiche 102
11
12  Debug "---"
13
14  Procedure Bar()
15    Static x  ; La déclaration est effectuée une seule fois, au lancement du programme.
16    x = 100   ; L'affectation est effectuée à chaque lancement de la Procedure.
```

```
17
18      Debug x
19      x + 1
20    EndProcedure
21
22    Bar() ; Affiche 100
23    Bar() ; Affiche 100
24    Bar() ; Affiche 100
```

Chapitre 68

Structures

Syntax

```
Structure <nom> [Extends
    <name>] [Align <expression
    numérique constante>]
  ...
EndStructure
```

Description

Structure est utile pour définir un type utilisateur et accéder à des zones mémoires du système d'exploitation par exemple. Les structures peuvent être utilisées pour rendre l'accès à des grands fichiers plus facilement. Cela peut être plus efficace dans la mesure où vous pouvez regrouper dans un même objet des informations communes. On accède aux structures avec le caractère \ . Les structures peuvent s'imbriquer. Les tableaux statiques sont acceptés dans une structure.
Les champs de structure doivent avoir un type explicite parmi tous les Types basiques gérés par PureBasic, à savoir Byte (.b), Ascii (.a), Caractère (.c) , Word (.w) , Unicode (.u), Long (.l), Integer (.i), Float (.f), Quad (.q), Double (.d)), String (.s) et String Fixe (.s{Longueur}).
Les objets dynamiques tel que les tableaux, listes et maps sont aussi supportés dans les structures et sont automatiquement initialisés quand l'objet utilisant la structure est déclaré. Pour définir un tel champ, utiliser les mot-clés suivant : Array, List et Map.
Généralement, les structures sont utilisées en association avec une variable, un tableau , une liste , ou une map . Toutefois, les utilisateurs avancés pourront allouer une structure en mémoire avec AllocateStructure() et la libérer avec FreeStructure() . Il est aussi possible d'initialiser une structure en mémoire avec InitializeStructure() , de la copier avec

CopyStructure() , de la vider avec ClearStructure() et de la reinitialiser avec ResetStructure()
Il est possible de copier une structure complète en utilisant l'opérateur égal ("=") entre deux éléments de même type.
Le paramètre optionnel Extends permet d'étendre une structure existante avec de nouveaux champs. Tous les champs se trouvant dans la structure étendue se retrouveront en tête de la nouvelle structure. C'est très utile pour faire un héritage simple de structures.
Pour les utilisateurs avancés seulement. Le paramètre Align permet d'ajuster l'alignement entre chaque champ de la structure. L'alignement par défaut est de 1, ce qui signifie pas d'alignement. Par exemple, si l'alignement est fixé à 4, chaque champs sera aligné sur 4 octets. Cela peut aider à améliorer les performances lors de l'accès aux champs de la structure, mais cela peut utiliser plus de mémoire, car un certain espace entre chaque champs sera perdu. La valeur spéciale #PB_Structure_AlignC peut être utilisée pour aligner la structure telle qu'elle se ferait en langage C, utile lors de l'importation structures C utilisées avec des fonctions API.

```
  - SizeOf
permet de connaître la
   taille en octets d'une
   structure
  - OffsetOf
peut être utilisé pour
   rechercher l'index du
   champ indiqué.
```

Note : Un **Tableau statique** dans une structure ne se comporte pas de la même façon qu'un tableau défini avec la commande Dim . Ceci pour être conforme au format de structures en C/C++ (pour permettre un portage direct des structures de l'API). Ce qui signifie que a**[2]** assignera un tableau de 0 à 1 (deux éléments) alors que Dim a**(2)** assignera un tableau de 0 à 2 (trois éléments). Et Les fonctions de la bibliothèque Array ne peuvent pas être utilisées avec ce type de tableaux.
Lorsque vous utilisez des pointeurs dans les structures, L'étoile '*' doit être omise lors de l'utilisation du champ, une fois de plus pour faciliter le portage de code API. Cela peut être considéré comme une bizarrerie (et pour être honnête, ça l'est) mais c'est comme ça depuis le début de PureBasic et beaucoup, beaucoup de sources sont écrites

de cette façon et cela restera inchangé. Quand beaucoup de champs doivent être remplis en une fois, il est conseillé d'utiliser With : EndWith pour réduire la quantité de code à saisir et améliorer sa lisibilité.

Exemple

```
1  Structure Personne
2    Nom.s
3    Prenom.s
4    Age.w
5  EndStructure
6
7  Dim MesAmis.Personne(100)
8
9  ; Ici la position '0' du
    tableau MesAmis()
10 ; contiendra une personne
    et ses informations
    personnelles
11
12 MesAmis(0)\Nom = "Durand"
13 MesAmis(0)\Prenom = "Michel"
14 MesAmis(0)\Age = 32
```

Exemple : Structure plus complexe (Tableau statique imbriqué)

```
1    Structure Fenetre
2     *FenetreSuivante.Fenetre
     ; Pointe vers un autre
     objet fenêtre
3     x.w
4     y.w
5     Nom.s[10]   ; 10 noms
     possibles
6  EndStructure
```

Exemple : Structure étendue

```
1  Structure MonPoint
2    x.l
3    y.l
4  EndStructure
5
6  Structure MonPointEnCouleur
    Extends MonPoint
7    couleur.l
8  EndStructure
9
10 ColoredPoint.MonPointEnCouleur\x
    = 10
11 ColoredPoint.MonPointEnCouleur\y
    = 20
```

```
ColoredPoint.MonPointEnCouleur\couleur = RGB(255, 0, 0)
```

Exemple : Copie de structure

```
Structure MonPoint
  x.l
  y.l
EndStructure

PointGauche.MonPoint\x = 10
PointGauche\y = 20

PointDroit.MonPoint = PointGauche

Debug PointDroit\x
Debug PointDroit\y
```

Exemple : Objet Dynamique

```
Structure Personne
  Nom$
  Age.l
  List Amis$()
EndStructure

Jean.Personne
Jean\Nom$ = "Jean"
Jean\Age   = 23

; Ajoutons des amis à Jean
;
AddElement(Jean\Amis$())
Jean\Amis$() = "Jim"

AddElement(Jean\Amis$())
Jean\Amis$() = "Monica"

ForEach Jean\Amis$()
  Debug Jean\Amis$()
Next
```

Exemple : Tableau statique, dynamique et Structure en argument de procédure

```
Structure Truc
  a.l
  b.l[2]          ; Tableau statique (Standard C) avec 2 valeurs b[0] et b[1], non redimensionnable
```

```
4     Array c.l(3,3)  ; Tableau
      dynamique avec 16 valeurs
      de c(0,0) à c(3,3),
      redimensionnable avec
      ReDim()
5   EndStructure
6
7   MaVar.Truc
8
9   Procedure
      MaProcedure(*blabla.Truc)
10    *blabla\a = 5
11    *blabla\b[0] = 1
12    *blabla\b[1] = 2
13    *blabla\c(3,3) = 33
14  EndProcedure
15
16  MaProcedure(@MaVar)
17  Debug MaVar\a
18  Debug MaVar\b[0]
19  Debug MaVar\b[1]
20  Debug MaVar\c(3,3)
21
22  ;Debug MaVar\c(0,10) ;
      Erreur index hors limite
23  ReDim MaVar\c(3,10)  ;
      Attention, seule la
      dernière dimension peut
      être redimensionnée !
24  Debug  MaVar\c(0,10)
```

Exemple : Structure de structure(s)

```
1   Structure pointF
2     x.f
3     y.f
4   EndStructure
5
6   Structure Champs
7     Champs1.q
8     Champs2.s{6}
9     Champs3.s
10    Array Tab.pointF(3)
11  EndStructure
12
13  Define MaVar.Champs
14
15  MaVar\Tab(3)\x = 34.67
```

Exemple : Alignement Mémoire

```
1   Structure Type Align 4
2     Byte.b
3     Word.w
4     Long.l
```

```
    Float.f
  EndStructure

  Debug OffsetOf(Type\Byte)
    ; Affiche 0
  Debug OffsetOf(Type\Word)
    ; Affiche 4
  Debug OffsetOf(Type\Long)
    ; Affiche 8
  Debug OffsetOf(Type\Float)
    ; Affiche 12
```

Exemple : Pointers

```
  Structure Personne
    *Next.Personne ; Ici, le '*' est obligatoire pour déclarer un pointeur
    Nom$
    Age.b
  EndStructure

  Timo.Personne\Nom$ = "Timo"
  Timo\Age = 25

  Fred.Personne\Nom$ = "Fred"
  Fred\Age = 25

  Timo\Next = @Fred ; Lorsque vous utilisez le pointeur, le '*' est omis

  Debug Timo\Next\Nom$ ; Affichera 'Fred'
```

Syntax

```
StructureUnion
  Field1.Type
  Field2.Type
  ...
EndStructureUnion
```

Description

StructureUnion est prévu pour les programmeurs avancés qui souhaitent économiser de la mémoire en partageant certains champs à l'intérieur d'une même structure. Il s'agit d'un équivalent du mot clef 'union' en C/C++.
Note : Chaque champ dans la déclaration StructureUnion peut être d'un type différent.

Exemple

```
Structure Type
  Nom$
  StructureUnion
    Long.l      ; Chaque champ (Long, Float et Byte) est placé à la
    Float.f     ; même adresse mémoire.
    String.b    ;
  EndStructureUnion
EndStructure
```

Exemple : Exemple extended (gestion des dates)

```
Structure date
  jour.s{2}
  pk1.s{1}
  mois.s{2}
  pk2.s{1}
  an.s{4}
EndStructure

Structure date2
  StructureUnion
    s.s{10}
    d.date
  EndStructureUnion
EndStructure

Dim d1.date2(5)

d1(0)\s = "05.04.2008"
d1(1)\s = "07.05.2009"

Debug d1(0)\d\jour
Debug d1(0)\d\mois
Debug d1(0)\d\an

Debug d1(1)\d\jour
Debug d1(1)\d\mois
Debug d1(1)\d\an

d2.date2\s = "15.11.2010"

Debug d2\d\jour
Debug d2\d\mois
Debug d2\d\an
```

Chapitre 69

Sous-systèmes

Introduction

Les commandes intégrées de PureBasic s'appuient sur les bibliothèques disponibles de chaque système d'exploitation. Parfois, il est possible d'atteindre un même résultat de différentes façons. Pour ce faire, PureBasic offre la possibilité de changer de bibliothèque, sans changer une ligne de code source. Par exemple, sous Windows, DirectX peut être utiisé en utilisant le sous-système 'DirectX' de PureBasic, qui lui, utilisera les fonctions DirectX pour rendre les sprites par exemple, en lieu et place de OpenGL (sous-système par défaut). Pour activer un sous-système, son nom doit être défini dans l'IDE avec le menu Options du compilateur , ou par l'intermédiaire du commutateur /SUBSYSTEM en /ligne de commande . Il s'agit d'une option de compilation, ce qui signifie qu'un exécutable ne peut pas incorporer plus d'un sous-système à la fois. Si le support de multiple sous-systèmes est nécessaire (par exemple l'envoi d'une version OpenGL et DirectX d'un jeu), deux exécutables doivent être créés.
Les sous-systèmes disponibles sont situés dans le dossier 'subsystems' de PureBasic. Lorsqu'un sous-système est spécifié, tous les résidents ou les bibliothèques trouvées dans ce dossier auront préséance sur les bibliothèques par défaut et les résidents. N'importe quel nombre de sous-systèmes différents peuvent être spécifiés (pour autant que cela n'affecte pas les mêmes bibliothèques).
La fonction du compilateur Subsystem peut être utilisée pour détecter si un sous-système spécifique est utilisé pour la compilation.

Sous-systèmes disponibles

Voici une liste des sous-systèmes disponibles, et les bibliothèques concernées :
Windows

```
DirectX9: Utiliser DirectX9
 au lieu d'OpenGL.
 Bibliothèques concernées:
        - Sprite
        - Screen
        - Note: Le moteur
 3D n'est plus disponible
 car il utilise OpenGL
DirectX11: Utiliser DirectX
 11  au lieu d'OpenGL.
 Bibliothèques concernées:
        - Sprite
        - Screen
        - Note: Le moteur
 3D n'est plus disponible
 car il utilise OpenGL
```

Linux

```
gtk2: Bibliothèques
 affectées:

        - 2D Drawing
        - AudioCD
        - Clipboard
        - Desktop
        - Drag & Drop
        - Font
        - Gadget
        - Image
        - Menu
        - Movie
        - Printer
        - Requester
        - Scintilla
        - StatusBar
        - SysTray
        - Toolbar
        - Window

qt: Bibliothèques affectées:
        - 2D Drawing
        - AudioCD
        - Clipboard
        - Desktop
        - Drag & Drop
        - Font
        - Gadget
        - Image
        - Menu
        - Movie
        - Printer
        - Requester
        - Scintilla
        - StatusBar
```

```
- SysTray
- Toolbar
- Window
```

MacOS X

```
Rien
```

Chapitre 70

Threaded

Syntax

```
Threaded[.<type>]
    <variable[.<type>]> [=
    <expression constante>] [,
    ...]
```

Description

Threaded permet de créer une variable , un tableau sauf les tableaux multi-dimensionnels), une liste ou une map qui sera persistant pour chaque thread . C'est à dire que chaque thread aura sa propre version de l'objet. C'est uniquement utile lors de l'écriture de programmes multi-threadés.
Si un type est spécifié après le mot-clef Threaded, le type par défaut pour cette déclaration est modifié.
Chaque variable peut avoir une valeur par défaut assignée, mais cette valeur doit être une constante. Les variables threadées sont initialisées au lancement du premier thread. Cela implique que si la variable est définie est assignée à une valeur en même temps alors elle est définie pour tous les threads. Voir exemple 2. Lors de la déclaration d'un tableau threadé, les paramètres de dimensionnement doivent être des valeurs constantes.
Un objet threadé ne peut pas être déclaré dans une procédure, et sa portée est toujours globale.

Exemple : 1 Avec une variable

```
1  Threaded Compteur
2
3  Compteur = 128
4
5  Procedure Thread(Parametre)
6
```

```
7     Debug Compteur ;
      Affichera zero, car ce
      thread n'a pas encore
      utilisé cette variable
8     Compteur = 256
9     Debug Compteur ;
      Affichera 256
10
11  EndProcedure
12
13  Thread =
      CreateThread(@Thread(), 0)
14  WaitThread(Thread) ;
      Attente de la fin
      d'exécution du thread.
15
16  Debug Compteur ; Affichera
      128, meme si 'Compteur' a
      ete change dans le thread
```

Exemple : 2 Tous les threads

```
1   Threaded Compteur = 128
2
3   Procedure Thread(Parametre)
4
5     Debug Compteur ;
      Affichera 128, car quand
      on lance un programme, on
      lance aussi un thread
6     Compteur = 256
7     Debug Compteur ;
      Affichera 256
8
9   EndProcedure
10
11  Thread =
      CreateThread(@Thread(), 0)
12  WaitThread(Thread) ;
      Attente de la fin
      d'exécution du thread.
13
14  Debug Compteur ; Affichera
      128, meme si 'Compteur' a
      ete change dans le thread
```

Chapitre 71

Unicode

Introduction

'Unicode' est un terme utilisé pour désigner un jeu de caractères étendu destiné à afficher du texte dans de nombreux langages différents (y compris les langages non latins, avec beaucoup de symboles différents tel que le Japonais, le Coréen etc.). En unicode, chaque symbole a sa place propre, et il n'est pas nécessaire d'avoir une table de caractères par langue. Pour simplifier, l'unicode peut être vu comme une très grande table ascii, qui n'a pas 256 caractères mais 65536. Donc, pour stocker un caractère, 2 octets seront nécessaires en mémoire (c'est important de noter cette différence, notamment lors de l'utilisation de pointeurs vers des chaînes de caractères).
Voici quelques liens pour se faire une meilleure idée de l'unicode (lecture vivement conseillée) :
Information générale sur l'unicode (Français)
Information générale sur l'unicode (Anglais)
L'unicode et les BOM

Unicode et Windows

PureBasic utilise l'encodage UCS2 qui est aussi le format utilisé de manière interne par l'API unicode MS Windows, donc aucune conversion n'est nécessaire lorsqu'un fonction est appelée. Si le programme utilise des fonctions API, PureBasic utilisera automatiquement la version unicode de la fonction si elle est disponible (par exemple MessageBox_() pointera vers MessageBoxW()). Il en va de même pour toutes les structures et les constantes API supportées par PureBasic.

UTF-8

L'UTF-8 est une autre façon d'encoder une chaîne de caractères unicode. Contrairement à UCS2 qui prend toujours 2 octets par caractère, UTF-8 utilise un encodage variable pour chaque caractère (jusqu'à 4 octets pour représenter un caractère). Le point fort de l'UTF-8 est qu'il ne contient pas de caractère nul lors de son encodage, donc le texte peut être édité facilement. De plus, les caractères ASCII de 1 à 127 sont conservés (ils ne sont pas modifiés lors de l'encodage) ce qui rend le texte à peu près lisible pour les langues latines. Son point faible est le côté variable de ses caractères, ce qui nécessite des routines de gestion de texte plus lentes.
PureBasic utilise l'UTF-8 comme encodage par défaut pour l'écriture et la lecture des chaînes de caractères dans les fichiers (bibliothèques Fichier et Préférence), pour que les fichiers soient complètement indépendants de la plateforme.
Le compilateur du PureBasic gère aussi les fichiers sources ascii et UTF-8 (les fichiers UTF-8 doivent avoir une entête BOM). Les deux types de fichiers peuvent être combinés dans un programme sans problème : un fichier source ASCII peut inclure un fichier source UTF-8 et vice-versa, tant que les variable .c et .s créées par l'utilisateur (c'est-à-dire "x" ou 'x') ne contiennent que des caractères avec un code <= 127 dans les fichiers ASCII.
Lors du développement d'un programme, il est recommandé de mettre l'IDE en mode UTF-8, pour que tous les fichiers sources soient enregistrés en unicode.

Chapitre 72

Variables, Types et Opérateurs

Déclaration des variables

La déclaration d'une variable en PureBasic se fait en la nommant. Les variables n'ont pas besoin d'être explicitement déclarées et peuvent être utilisées "à la volée".
Vous pouvez également spécifier le type que vous souhaitez pour cette variable.
Cependant, par défaut, une variable qui est déclarée sans indiquer son type de façon explicite sera considérée comme étant de type INTEGER.
Le mot clef Define peut être utilisé pour déclarer plusieurs variables sur une même ligne.

Exemple

```
1  vitesse.q ; Déclare une
     variable 'vitesse' du type
     quad (.q).
2  c.l = a*d.w ; 'd' est
     déclarée ici au milieu
     d'une expression et elle
     est de type word (.w) !
3  Define.b a0, b0 = 10, c0 =
     b0*2, d0
```

Note : Les noms de variables ne peuvent pas commencer par un chiffre (0, 1, etc.), ne peuvent pas contenir de caractères spéciaux(é, à, ß, ä, ö, ü, etc.) ni d'opérateurs (+, -, etc.).
Note : Si le contenu d'une variable ne change pas tout au long de l'exécution du programme (utilisation d'une valeur fixe), il est préférable d'utiliser une constante .

Types basiques

PureBasic permet de définir des variables de plusieurs types comme les entiers, des

nombres à virgule, des caractères (char) et
des chaînes de caractères aussi.
Voici la liste des types natifs supportés :

```
   Nom         | Extension
  | Taille en mémoire
                      | Plage
-------------+--------------+--------------------------------
Byte          |     .b
  | 1 octet

  | -128 à +127
Ascii         |     .a
  | 1 octet

  | 0 à +255
Caractère     |     .c
  | 2 octets

  | 0 à +65535
Word          |     .w
  | 2 octets

  | -32768 à +32767
Unicode       |     .u
  | 2 octets

  | 0 à +65535
Long          |     .l
  | 4 octets

  | -2147483648 à +2147483647
Integer       |     .i
  | 4 octets (avec
  compilateur 32-bit) |
  -2147483648 à +2147483647
Integer       |     .i
  | 8 octets (avec
  compilateur 64-bit) |
  -9223372036854775808 à
  +9223372036854775807
Float         |     .f
  | 4 octets

  | illimité (voir
  ci-dessous)
Quad          |     .q
  | 8 octets

  | -9223372036854775808 à
  +9223372036854775807
Double        |     .d
  | 8 octets

  | illimité (voir
  ci-dessous)
String        |     .s
  | longueur string+1
                      | illimité
```

```
String Fixe   | .s{Longueur}
 | longueur string
                        |
 illimité
```

> **Types non-signés (.a, .u et .c) :** Purebasic offre des types non-signés pour les variables de type 'byte' et 'word' au travers des types 'ascii' (.a) et 'unicode' (.u).
Le type 'character'(.c) est 'word' non-signé en mode unicode .

> **Notation des variables de type chaîne de caractères ($ et .s) :** Généralement une chaîne de caractère se définie avec le type '.s' mais il est possible d'utiliser '**$**' comme dernier caractère d'une variable pour indiquer qu'il s'agit d'une chaîne de caractères. De cette façon vous pouvez utiliser 'a$' et 'a.s' qui sont alors deux variables différentes.
Vous devrez conserver le '$' à la fin de la variable a$ contrairement au '.s' de la variable a.s qui n'est nécessaire que pour sa déclaration.

```
1  a.s = "Une chaîne"
2  a$ = "Une autre chaîne"
3  Debug a    ; Affichera "Une
     chaîne"
4  Debug a$   ; Affichera "Une
     autre chaîne"
```

> **Notation scientifique exponentielle (Ae+-B) :** Les nombres à virgules (Float ou Double) peuvent être écrits sous la forme exponentielle :

```
1  valeur.d = 123.5e-20
2  Debug valeur    ; affichera
     0.00000000000000001235
```

Opérateurs

Les opérateurs peuvent être intégrés aux expressions pour combiner les variables, constantes et tout ce qui est nécessaire.
La table ci-dessous montre tous les opérateurs utilisables en PureBasic.
LHS = Left Hand Side ou partie gauche de l'équation.
RHS = Right Hand Side ou partie droite de l'équation.

Opérateur = (Egal)

Peut être utilisé suivant deux acceptions. La première est pour l'affectation de la variable LHS à la valeur résultat de l'expression RHS. La seconde signification est l'utilisation dans une expression de

comparaison entre LHS et RHS. Si le résultat de LHS est identique au résultat de RHS la valeur 'vrai' sera retournée sinon se sera la valeur 'faux'.

Exemple

```
a=b+c ; Affecte la valeur de "b+c" à la variable "a"
If abc=def ; Teste si les valeurs de abc et def sont identiques et utilise le résultat dans la commande If
```

Opérateur + (Plus)

Donne le résultat de la valeur de l'expression RHS ajoutée à la valeur de l'expression LHS. Si le résultat de cet opérateur n'est pas utilisé et qu'il y a une variable LHS, alors la valeur de l'expression RHS sera directement ajoutée à la valeur LHS.

Exemple

```
nombre=mavaleur+2 ; Ajoute la valeur 2 à "mavaleur" et utilise le résultat avec l'opérateur =
variable+expression ; La valeur de "expression" est directement ajoutée à "variable"
```

Opérateur - (Moins)

Soustrait la valeur de l'expression RHS de la valeur de l'expression LHS. S'il n'y a pas d'expression LHS l'opérateur prend la valeur négative de la valeur RHS. Si le résultat de l'opérateur n'est pas utilisé et qu'il n'y a pas de variable LHS, alors la valeur de RHS est directement soustraite à la valeur de la variable LHS. Cet opérateur ne peut être utilisé avec les variables de type chaîne.

Exemple

```
var=#MaConstante-chose; Soustrait la valeur de "chose" de "#MyConstant" et utilise le résultat avec l'opérateur égal.
```

```
2 uneautre=uneautre+ -var ;
   Calcule la valeur négative
   de "var" et utilise le
   résultat avec l'opérateur
   +.
3 variable-expression ; La
   valeur "expression" est
   directement soustraite à
   "variable"
```

Opérateur * (Multiplication)

Multiplie la valeur de l'expression LHS par la valeur de RHS. Si le résultat de l'opérateur n'est pas utilisé et qu'il y a une variable LHS, alors la valeur de la variable est directement multipliée par la valeur de l'expression RHS. Cet opérateur ne peut être utilisé dans une variable de type chaine.

Exemple

```
1 total=prix*quantite ;
   Multiplie la valeur de
   "prix" par la valeur de
   "quantite" et utilise le
   résultat avec l'opérateur =
2 variable*expression ;
   "variable" est multiplé
   directement par la valeur
   de "expression"
```

Opérateur / (Division)

Divise la valeur de l'expression LHS par la valeur de l'expression RHS. Si le résultat de l'opérateur n'est pas utilisé et qu'il y a une variable LHS, alors la valeur de la variable est directement divisée par la valeur de l'expression RHS. Cet opérateur ne peut être utilisé dans les variables de type chaine.

Exemple

```
1 quantite=total/prix ;
   Divise la valeur "total"
   par la valeur "prix" et
   utilise le résultat avec
   l'opérateur =
2 variable/expression ;
   "variable" est directement
   divisé par la valeur
   "expression"
```

Opérateur & (AND est un ET logique (binaire))

Il vous faut être familiarisé avec les nombres binaires pour utiliser cet opérateur. Le résultat de cet opérateur est le résultat d'un ET logique entre les valeurs des expressions LHS et RHS, bit à bit. La valeur de chaque bit résultant est fixée comme indiqué dans la table ci-dessous. De plus, si le résultat de l'opérateur n'est pas utilisé et qu'il y a une variable LHS, alors le résultat sera directement stocké dans cette variable. Cet opérateur ne peut être utilisé avec une variable de type chaîne.

```
LHS | RHS |Résultat
------------------
 0  |  0  |    0
 0  |  1  |    0
 1  |  0  |    0
 1  |  1  |    1
```

Exemple

```
1 ; La représentation binaire des valeurs est utilisée pour une présentation plus claire et lisible
2 a.w = %1000 & %0101 ; Le résultat sera 0
3 b.w = %1100 & %1010 ; Le résutlat sera %1000
4 bits = a & b ; Effectue un ET bit à bit entre a et b et utilise le résultat avec l'opérateur =
5 a & b ; Effectue un ET bit à bit entre a et b et place le résultat directement dans la variable "a"
```

Opérateur || (OR est un OU logique (binaire))

Vous devez être familiarisé avec les nombres binaires pour utiliser cet opérateur. Le résultat de cet opérateur est le résultat d'un OU logique entre les valeurs des expressions LHS et RHS bit à bit. La valeur de chaque bit résultant est fixée comme indiqué dans la table ci-dessous. De plus, si le résultat de l'opérateur n'est pas utilisé et qu'il et qu'il y a une variable LHS alors le résultat est directement stocké dans cette variable. Cet opérateur ne peut être utilisé avec une variable de type chaîne.

```
LHS | RHS |Résultat
------------------
 0  |  0  |    0
 0  |  1  |    1
 1  |  0  |    1
 1  |  1  |    1
```

Exemple

```
; La représentation binaire des valeurs est utilisée pour une présentation claire et lisible
a.w = %1000 | %0101 ; Le résultat sera %1101
b.w = %1100 | %1010 ; Le résultat sera %1110
bits = a | b ; Effectue un OU bit à bit entre a et b et utilise le résultat avec l'opérateur =
a | b ; Effectue un OU bit à bit entre a et b et place le résultat directement dans la variable "a"
```

Opérateur ! (XOR est un OU exclusif logique (binaire))

Vous devez être familiarisé avec les nombres binaires pour utiliser cet opérateur. Le résultat de cet opérateur est le résultat d'un OU Exclusif entre les valeurs LHS et RHS bit à bit. La valeur de chaque bit résultant est fixée comme indiqué dans la table ci-dessous. De plus, si le résultat de l'opérateur n'est pas utilisé et qu'il y a une variable LHS alors le résultat est directement stocké dans cette variable. Cet opérateur ne peut être utilisé avec une variable de type chaîne.

```
LHS | RHS |Résultat
------------------
 0  |  0  |    0
 0  |  1  |    1
 1  |  0  |    1
 1  |  1  |    0
```

Exemple

```
; La représentation binaire des valeurs est utilisée pour une présentation claire et lisible
```

```
2 | a.w = %1000 ! %0101 ; Le
    résultat sera %1101
3 | b.w = %1100 ! %1010 ; Le
    résultat sera %0110
4 | bits = a ! b ; Effectue un
    OU Exclusif bit à bit
    entre a et b et utilise le
    résultat avec l'opérateur =
5 | a ! b ; Effectue un OU
    Exclusif bit à bit entre a
    et b et place le résultat
    directement dans "a"
```

Opérateur * * (NON inversion logique (binaire))

Vous devez être familiarisé avec les nombres binaires pour utiliser cet opérateur. Le résultat de cet opérateur est une inversion bit à bit de la valeur RHS. La valeur de chaque bit est fixée comme indiqué dans la table ci-dessous. Cet opérateur ne peut être utilisé avec une variable de type chaîne.

```
RHS |Résultat
----------
 0  |    1
 1  |    0
```

Exemple

```
1 | ; La représentation binaire
    des valeurs est utilisée
    pour une présentation
    claire et lisible
2 | a.w = ~%1000 ; Le résultat
    sera %0111
3 | b.w = ~%1010 ; Le résultat
    sera %0101
```

() (Parenthèses)

Vous pouvez utiliser les parenthèses pour forcer l'évaluation prioritaire d'une partie d'une expression ou modifier l'ordre d'évaluation.

Exemple

```
1 | a = (5 + 6) * 3 ; Le
    résultat est 33 car 5+6
    est évalué en premier
2 | b = 4 * (2 - (3 - 4)) ; Le
    résultat est 12 car 3-4
    est évalué en premier,
    ensuite 2-résultat puis la
    mutiplication pour finir
```

< (Inférieur à)

Utilisé pour comparer les valeurs des expressions LHS et RHS. Si la valeur de LHS est plus petite que la valeur de RHS cet opérateur rend un résultat vrai, sinon le résultat est faux.

> (Supérieur à)

Utilisé pour comparer les valeurs des expressions LHS et RHS. Si la valeur de LHS est plus grande que la valeur de RHS cet opérateur rend un résultat vrai, sinon le résultat est faux.

<=, =< (Inférieur ou égal à)

Utilisé pour comparer les valeurs des expressions LHS et RHS. Si la valeur de LHS est plus petite ou égale à la valeur de RHS cet opérateur rend un résultat vrai, sinon le résultat est faux.

>=, => (Supérieur ou égal à)

Utilisé pour comparer les valeurs des expressions LHS et RHS. Si la valeur de LHS est plus grande ou égale à la valeur de RHS cet opérateur rend un résultat vrai, sinon le résultat est faux.

<> (Différent)

Utilisé pour comparer les valeurs des expressions LHS et RHS. Si la valeur de LHS est différente de la valeur de RHS cet opérateur rend un résultat vrai, sinon le résultat est faux.

And (ET logique)

Peut être utilisé pour combiner les résultats vrais ou faux des opérateurs de comparaison en donnant un résultat fixé comme indiqué dans la table ci-dessous.

LHS	RHS	Résultat
faux	faux	faux
faux	vrai	faux
vrai	faux	faux
vrai	vrai	vrai

Or (OU logique)

Peut être utilisé pour combiner les résultats vrais ou faux des opérateurs de comparaison en donnant un résultat fixé comme indiqué dans la table ci-dessous.

```
  LHS   |   RHS   | Résultat
------------------------
  faux  |   faux  |  faux
  faux  |   vrai  |  vrai
  vrai  |   faux  |  vrai
  vrai  |   vrai  |  vrai
```

Operator XOr (OU exclusif logique)

Peut être utilisé pour combiner les résultats vrais ou faux des opérateurs de comparaison en donnant un résultat fixé comme indiqué dans la table ci-dessous.

```
  LHS  |   RHS  | Résultat
-----------------------
 faux  | faux   | faux
 faux  | vrai   | vrai
 vrai  | faux   | vrai
 vrai  | vrai   | faux
```

Operator Not (NON logique)

Le résultat de cet opérateur sera la négation de l'expression RHS. Cet opérateur ne fonctionne pas avec les strings.

```
  RHS  | Résultat
----------------
 faux  | vrai
 vrai  | faux
```

Opérateur « (Décalage à gauche)

Décale vers la gauche les bits du nombre LHS de RHS places. Décaler les bits vers la gauche revient à faire une multiplication par un multiple de 2. Il est conseillé de bien comprendre les opérations binaires avant d'utiliser cet opérateur.

Exemple

```
1  a=%1011 << 1 ; La valeur de
     'a' sera %10110. (en
     decimal: %1011=11 et
     %10110=22)
2  b=%111 << 4 ; La valeur de
     'b' sera %1110000. (en
     decimal: %111=7 et
     %1110000=112)
3  c.l=$80000000 << 1 ; La
     valeur de 'c' sera 0. Les
     bits supérieurs sont
     perdus car ils dépassent
     la capacité du type.
```

Opérateur » (Décalage à droite)

Décale vers la droite les bits du nombre LHS de RHS places. Décaler les bits vers la droite revient à faire une division par un multiple de 2. Il est conseillé de bien comprendre les opérations binaires avant d'utiliser cet opérateur.

Exemple

```
1 d=16 >> 1 ; La valeur de
    'd' sera 8. (en binaire:
    16=%10000 et 8=%1000)
2 e.w=%10101010 >> 4 ; La
    valeur de 'e' sera %1010.
    (en décimal: %10101010=170
    et %1010=10).
3 f.b=-128 >> 1 ; La valeur
    de 'f' sera -64.
    -128=%10000000,
    -64=%11000000. Lors du
    décalage, le bit le plus
    fort reste (conservation
    du signe).
```

Opérateur % (Modulo)

Calcule le reste de la division entière de RHS par LHS.

Exemple

```
1 a=16 % 2 ; La valeur sera 0
    car 16/2 = 8 (aucun reste)
2 b=17 % 2 ; La valeur sera 1
    car 17/2 = 8*2+1 (reste 1)
```

Opérateurs raccourcis

Tous les opérateurs mathématiques peuvent être utilisés sous une forme abrégée.

Exemple

```
1 Valeur + 1   ; Equivaut à :
    Valeur = Valeur + 1
2 Valeur * 2   ; Equivaut à :
    Valeur = Valeur * 2
3 Valeur << 1  ; Equivaut à :
    Valeur = Valeur << 1
```

Note : Cela peut conduire à des résultats "imprévus" dans quelques rares cas, si l'assignement est modifiée avant l'affection.

Exemple

```
1  Dim MonTableau(10)
2  MonTableau(Random(10)) + 1
    ; Equivaut à:
    MonTableau(Random(10)) =
    MonTableau(Random(10)) +
    1, mais ici Random() ne
    renverra pas la même
    valeur à chaque appel.
```

Priorité des opérateurs

```
Niveau de Priorité |
 Opérateurs
--------------------+------------------------------
     8 (haut)       |
 ~, - (ici - est le négatif)
     7              |     <<,
 >>, %, !
     6              |
 |, &
     5              |
 *, /
     4              |
 +, - (ici - est la
 soustraction)
     3              | >, >=,
 =>, <, <=, =<, =, <>
     2              |
 Not
     1 (bas)        |  And,
 Or, XOr
```

Types structurés

Les types structurés peuvent être définis avec les options propres aux structures. Voyez le chapitre structures pour plus d'informations.

Types Pointeur

Les pointeurs sont déclarés avec un '*' devant le nom de la variable. Plus d'informations peuvent être trouvées dans le chapitre pointeurs .

Informations concernant les nombres flottants

Un nombre flottant est stocké de telle manière que la 'virgule flotte' autour de la partie réelle. De la sorte, il est possible d'avoir des nombres dont la valeur peut être aussi bien grande que petite. Toutefois vous ne pouvez pas stocker de grands nombres

avec une précision aussi élevée que des petits nombres.
Une autre limitation concernant les nombres flottants est qu'ils restent concrètement représentés sous une forme binaire. Ainsi, ils ne peuvent être restitués qu'à partir de multiples et de divisions en base 2. Celà est important pour comprendre que la représentation décimale lors de l'affichage ou du calcul n'est pas tout à fait identique à ce que l'on peut attendre dans une représentation humaine. Représenter 0.5 ou 0.125 est simple car ce sont des divisions parfaites de 2, cela est plus complexe pour des nombres comme 0.11 ou 0.10999999. L'affichage approché de la valeur est toujours correct à un nombre limité de décimales, mais ne soyez pas surpris si au-delà le nombre affiché s'écarte de la valeur que vous attendez !
Ces remarques s'appliquent aux nombres flottants traités par ordinateur d'une manière générale et non spécifiquement à Purebasic.
Comme leur nom l'indique, les 'doubles' sont des flottants 'double-precision' (64-bit) comparativement aux flottants 'simple-precision' que sont les floats (32-bit). Donc, pour avoir plus de précision dans la manipulation des nombres à virgule, il est préférable d'utiliser les 'doubles'.

> Float : De +- 1.175494e-38 à +- 3.402823e+38
> Double : De +- ; 2.2250738585072013e-308 à +- 1.7976931348623157e+308

Pour plus d'information sur le format 'IEEE 754', consulter l'article Wikipedia.

Exemple

```
1    a.f=0.1
2    b.f=0.5
3    c.f=0.9
4    Debug a ; Affiche
     0.10000000149012
5    Debug b ; Affiche 0.5
6    Debug c ; Affiche
     0.89999997615814
```

Chapitre 73

AddPathSegment Suite

Traduction FR de la page Standard SVG Tiny

Voir aussi

PathSegments() , AddPathSegments()

Chapitre 74

While : Wend

Syntax

```
While <expression>
  ...
Wend
```

Description

While produit une boucle jusqu'à ce que l'expression devienne fausse. Il est à noter qu'avec le test accompagnant While, si la première tentative échoue, le programme n'entrera pas dans la boucle et évitera donc cette section de code contrairement à une boucle Repeat qui est toujours exécutée au moins une fois (car le test est effectué en sortie de code conditionnel).

Exemple

```
1  b = 0
2  a = 10
3  While a = 10
4    b = b+1
5    If b=10
6      a=11
7    EndIf
8  Wend
```

Ce programme boucle jusqu'à ce que la valeur "a" devienne différente de 10. La boucle sera donc exécutée 10 fois.

Chapitre 75

Gestion des messages Windows

Introduction

Les messages de votre programme seront transférés par Windows dans une file d'attente, qui est traitée uniquement si vous le désirez. Windows transmet un millier de messages à votre programme sans avertissement direct.
Par exemple si vous changez le statut d'un gadget (que ce soit en ajoutant une saisie ou en changeant l'image d'un ImageGadget), un message est envoyé à la file d'attente de votre programme.
Il y a deux possibilités de récupérer et de traiter les messages Windows dans PureBasic : WaitWindowEvent() and WindowEvent() . La différence est, que WaitWindowEvent() attend qu'un message arrive alors que WindowEvent() permet de continuer à travailler. Les messages dans la file d'attente ne seront cependant traités qu'après l'appel à WindowEvent() ou WaitWindowEvent().

Spécificités de WindowEvent()

La commande WindowEvent() n'attend pas, qu'un message arrive, mais vérifie seulement s'il y en a un dans la file d'attente. Si Oui, le message est traité et WindowEvent() renvoie le numéro du message. Si aucun message n'est dans la file, alors zéro (0) est renvoyé.

Exemple

```
1   While WindowEvent() : Wend
```

Fait en sorte, que WindowEvent() soit appelée tant qu'elle ne renvoie pas 0, c'est à dire jusqu'à ce que tous les messages de la file d'attente soient traités
Insérer un simple 'WindowEvent()' après un SetGadgetState() ne suffit pas pour traiter ce message en particulier. Premièrement il peut toujours y avoir d'autres messages dans la file d'attente, qui sont arrivés auparavant, et deuxièmement Windows envoie également un nombre conséquent d'autres messages, dont nous n'avons que faire... mais qui néanmoins sont dans la file d'attente.
Un simple appel à

```
1  WindowEvent()
```

ne suffit pas, le programme peut fonctionner correctement dans certaines circonstances sur une version de Windows, mais pas sur une autre version. Les différentes versions de Windows ont un fonctionnement interne tellement spécifique, qu'une version envoie seulement 1 message mais une autre version peut envoyer 5 messages pour le même cas de figure. .
A cause de cela, on utilise toujours pour la mise à jour :

```
1  While WindowEvent() : Wend
```

Bien sûr il y a aussi l'alternative

```
1  Repeat : Until
     WindowEvent() = 0
```

possible, qui n'est cependant pas très courante.
La méthode While WindowEvent() : Wend est fréquemment utilisée avec la commande Delay() , lorsqu'une une boucle est insérée AVANT le Delay(), par exemple pour attendre que la mise à jour d'un ImageGadget soit effective, après avoir changé une image avec SetGadgetState().

Chapitre 76

With : EndWith

Syntax

```
With <expression>
  ...
EndWith
```

Description

Le bloc With : EndWith permet de réduire la quantité de code à saisir et améliore sa lisibilité quand beaucoup de champs d'une structure doivent être renseignés. L'<expression> spécifiée sera automatiquement insérée avant chaque caractère anti-slash '\' qui suit un espace ou un opérateur. C'est une directive de compilateur qui fonctionne de la même manière qu'une macro : la ligne complète est recréée lors de la compilation, puis elle est traitée. Les blocs With : EndWith ne peuvent pas être imbriqués, car cela pourrait générer des bugs difficiles à résoudre.

Exemple

```
1   Structure Personne
2     Nom$
3     Age.l
4     Taille.l
5   EndStructure
6
7   Ami.Personne
8
9   With Ami
10    \Nom$ = "Yann"
11    \Age   = 30
12    \Taille  = 196
13
14    Debug \Taille+\Taille
15  EndWith
```

Exemple

```
Structure Corps
  Poids.l
  Couleur.l
  Texture.l
EndStructure

Structure Personne
  Nom$
  Age.l
  Corps.Corps[10]
EndStructure

Ami.Personne

For k = 0 To 9
  With Ami\Corps[k]
    \Poids = 50
    \Couleur  = 30
    \Texture = \Couleur*k

    Debug \Texture
  EndWith
Next
```

Quatrième partie

Les bibliothèques

Chapitre 77

2DDrawing

Généralités

La bibliothèque 2DDrawing contient toutes les commandes permettant de dessiner en deux dimensions sur une fenêtre, un écran ou même une imprimante. Vous pourrez ainsi tracer un trait, un rectangle, un cercle, une ellipse, créer des dégradés de formes diverses et écrire du texte.
Les dessins commencent toujours après avoir appelé la fonction StartDrawing() et se terminent lorsque StopDrawing() est appelé.

OS Supportés

Tous

77.1 Red

Syntaxe

```
Resultat = Red(Couleur)
```

Description

Renvoie la valeur de la composante rouge d'une couleur.

Arguments

Couleur La valeur de la couleur.
Cela peut être une valeur RGB 24 bits ou une valeur RGBA 32 bits.

Valeur de retour

Renvoie la valeur de la composante rouge.
Le résultat sera compris entre 0 et 255.

Remarques

RGB() permet de combiner les valeurs rouge, verte et bleue en une couleur 24 bits. RGBA() permet de combiner les valeurs rouge, verte, bleue et alpha en une couleur 32 bits.
Ces fonctions sont utiles lors des opérations de dessin 2D .

Exemple

```
1  ; Charge une image
2  If LoadImage(0,
    #PB_Compiler_Home +
    "Examples\Sources\Data\Geebee2.bmp")
3    StartDrawing(ImageOutput(0))
4    Couleur=Point(10,10)   ;
    Couleur aux coordonnées
    10, 10
5    Debug Red(Couleur)     ;
    Affiche la composante
    rouge de la couleur
6    Debug Green(Couleur)   ;
    Affiche la composante
    verte de la couleur
7    Debug Blue(Couleur)    ;
    Affiche la composante
    bleue de la couleur
8    StopDrawing()
9  EndIf
```

Voir aussi

Green() , Blue() , Alpha() , RGB() , RGBA()

OS Supportés

Tous

77.2 Green

Syntaxe

```
Resultat = Green(Couleur)
```

Description

Renvoie la valeur de la composante verte d'une couleur.

Arguments

Couleur La valeur de la couleur.
Cela peut être une valeur RGB 24 bits ou une valeur RGBA 32 bits.

Valeur de retour

Renvoie la valeur de la composante verte.
Le résultat sera compris entre 0 et 255.

Remarques

RGB() permet de combiner les valeurs rouge, verte et bleue en une couleur 24 bits.
RGBA() permet de combiner les valeurs rouge, verte, bleue et alpha en une couleur 32 bits.
Ces fonctions sont utiles lors des opérations de dessin 2D .

Exemple

```
1  ; Charge une image
2  If LoadImage(0,
    #PB_Compiler_Home +
    "Examples\Sources\Data\Geebee2.bmp")
3    StartDrawing(ImageOutput(0))
4    Couleur=Point(10,10)   ;
    Couleur aux coordonnées
    10, 10
5    Debug Red(Couleur)     ;
    Affiche la composante
    rouge de la couleur
6    Debug Green(Couleur)   ;
    Affiche la composante
    verte de la couleur
7    Debug Blue(Couleur)    ;
    Affiche la composante
    bleue de la couleur
8    StopDrawing()
9  EndIf
```

Voir aussi

Red() , Blue() , Alpha() , RGB() , RGBA()

OS Supportés

Tous

77.3 Blue

Syntaxe

```
Resultat = Blue(Couleur)
```

Description

Renvoie la valeur de la composante bleue d'une couleur.

Arguments

Couleur La valeur de la couleur.
Cela peut être une valeur RGB 24 bits ou une valeur RGBA 32 bits.

Valeur de retour

Renvoie la valeur de la composante bleue.
Le résultat sera compris entre 0 et 255.

Remarques

RGB() permet de combiner les valeurs rouge, verte et bleue en une couleur 24 bits.
RGBA() permet de combiner les valeurs rouge, verte, bleue et alpha en une couleur 32 bits.
Ces fonctions sont utiles lors des opérations de dessin 2D .

Exemple

```
1  ; Charge une image
2  If LoadImage(0,
    #PB_Compiler_Home +
    "Examples\Sources\Data\Geebee2.bmp")
3    StartDrawing(ImageOutput(0))
4    Couleur=Point(10,10)   ;
    Couleur aux coordonnées
    10, 10
5    Debug Red(Couleur)     ;
    Affiche la composante
    rouge de la couleur
6    Debug Green(Couleur)   ;
    Affiche la composante
    verte de la couleur
7    Debug Blue(Couleur)    ;
    Affiche la composante
    bleue de la couleur
8    StopDrawing()
9  EndIf
```

Voir aussi

Red() , Green() , Alpha() , RGB() , RGBA()

OS Supportés

Tous

77.4 Alpha

Syntaxe

```
Resultat = Alpha(Couleur)
```

Description

Renvoie la valeur de la composante alpha (transparence) d'une couleur.

Arguments

Couleur La valeur de la couleur.
Cela ne peut être qu'une valeur RGBA 32 bits.

Valeur de retour

Renvoie la valeur de la composante alpha. Une valeur de 0 signifie une complète transparence et une valeur de 255 signifie opacité totale.
Le résultat sera compris entre 0 et 255.

Remarques

RGBA() permet de combiner les valeurs rouge, verte, bleue et alpha en une couleur 32 bits.
Ces fonctions sont utiles lors des opérations de dessin 2D .

Exemple

```
1  If OpenWindow(0, 0, 0, 200, 200, "Alpha", #PB_Window_SystemMenu|#PB_Window_ScreenCentered)
2   If CreateImage(0, 800, 600, 32) And StartDrawing(ImageOutput(0))
    ; création d'une image fond noir par défaut
3    DrawingMode(#PB_2DDrawing_AlphaChannel)
    ; le Canal alpha (transparence) sera modifié
4    Box(0, 0, 200, 200, $00000000)          ; le noir est complètement transparent
5
6    DrawingMode(#PB_2DDrawing_AlphaBlend)
      ; le dessin sera fusionné avec le fond en utilisant le canal alpha pour gérer la transparence de chaque pixel
7    Circle( 75,  75, 50, RGBA(255,   0,   0, 64))
    ; Cercle rouge très transparent
8    Circle(125,  75, 50, RGBA(  0, 255,   0, 128))
    ; Cercle vert moyennement transparent
```

```
    Circle(100, 125, 50,
    RGBA(  0,   0, 255, 192))
    ; Cercle bleu peu
    transparent
    DrawText(20, 180,
    "Cliquer sur le dessin")
        ; Texte sans
    transparence
    StopDrawing()
  EndIf

  ImageGadget(0, 0, 0, 0, 0,
   ImageID(0))  ; Affichage
   du dessin dans un gadget

   Repeat
     Event =
   WaitWindowEvent(0)
     If  Event =
   #PB_Event_Gadget And
   EventType() =
   #PB_EventType_LeftClick
       x = WindowMouseX(0)
       y = WindowMouseY(0)
       If Bool(x>0  And y>0)
         StartDrawing(ImageOutput(0))
         DrawingMode(#PB_2DDrawing_AlphaBlend)
         Couleur = Point(x,y)
         Transparence =
   Alpha(Couleur)
         Debug "La
   composante Transparence
   vaut : " +
   Str(Transparence)
         StopDrawing()
       EndIf
     EndIf
   Until Event =
   #PB_Event_CloseWindow
  EndIf
```

Voir aussi

Red() , Green() , Blue() , RGBA()

OS Supportés

Tous

77.5 RGB

Syntaxe

```
Resultat = RGB(Rouge, Vert,
   Bleu)
```

Description

Renvoie la valeur de la couleur 24 bits correspondant aux valeurs Rouge, Verte et Bleue.

Arguments

Rouge, Vert, Bleu La valeur des composantes rouge, vert et bleu de la couleur.
Chaque valeur doit être comprise entre 0 et 255.

Valeur de retour

Renvoie la valeur de la couleur.

Remarques

Pour extraire la valeur d'une des composantes 'Rouge', 'Verte' ou 'Bleue' à partir de la valeur d'une couleur 24 Bits, utilisez les commandes suivantes Red() , Green() et Blue() .
Ces fonctions sont utiles pour effectuer des opérations de dessin .
Un tableau représentant les couleurs les plus communes est disponible ici .

Exemple

```
1  Debug RGB(0, 0, 0)    ; 0
2  Debug RGB(255, 0, 0)  ; 255
3  Debug RGB(0, 1, 0)    ; 256
4  Debug RGB(0, 255, 0)  ; 65280
5  Debug RGB(0, 0, 255)  ; 16
     711 680
6  Debug RGB(255, 255, 255) ;
     16 777 215 : Voici les 16
     millions de couleurs...
7  Debug HEX(RGB(255, 255,
     255)) ; FFFFFF
```

Voir aussi

Red() , Green() , Blue() , RGBA()

OS Supportés

Tous

77.6 RGBA

Syntaxe

```
Resultat = RGBA(Rouge, Vert,
    Bleu, Alpha)
```

Description

Renvoie la valeur de la couleur 32 bits correspondant aux valeurs Rouge, Verte, Bleue et Alpha.

Arguments

Rouge, Vert, Bleu La valeur des composantes rouge, verte et bleue de la couleur.
Chaque valeur doit être comprise entre 0 et 255.

Alpha La composante alpha (transparence) de la couleur.
La valeur doit être comprise entre 0 et 255.
Une valeur de 0 signifie une transparence complète et une valeur de 255 signifie une opacité totale.

Valeur de retour

Renvoie la valeur de la couleur.

Remarques

Resultat varie de 0 à 4 294 967 295 teintes. Il est donc conseillé d'utiliser un 'quad', (Resultat.q) et de mettre à zéro les octets inutilisés. En effet, sur un système d'exploitation 32 Bits, Resultat est un integer de type Long (par défaut) dont la plage d'utilisation va de - 2 147 483 648 à + 2 147 483 647, alors comparer deux couleurs est hasardeux.
Utiliser les commandes suivantes Red() , Green() , Blue() et Alpha() pour extraire la valeur d'une des composantes 'Rouge', 'Verte', 'Bleue' ou 'Alpha'.
Un tableau représentant les couleurs les plus communes est disponible ici . Ces fonctions sont utiles lors des opérations de dessin 2D .

Exemple

```
1  Couleur.q = RGBA(255, 255,
     255, 255) ; Blanc
     totalement opaque
2  Debug LSet(Hex(Couleur,
     #PB_Quad), 16, "0")
3  Couleur = Couleur  &
     $FFFFFFFF  ; mise à zéro
     des octets inutilisés,
```

```
                                    ;
    utile pour la comparaison
    de couleur
Debug LSet(Hex(Couleur,
    #PB_Quad), 16, "0")
```

Exemple

```
Debug RGBA(0, 0, 0, 0)
    ; Noir totalement
    transparent
Debug RGBA(0, 0, 0, 128)
    ; Noir à moitié transparent
Debug RGBA(0, 0, 0, 255)
    ; Noir totalement opaque
Debug RGBA(255, 0, 0, 255)
    ; Rouge totalement opaque
Debug RGBA(0, 0, 255, 0)
    ; Bleu totalement
    transparent
Debug RGBA(255, 255, 255,
    255) ; Blanc totalement
    opaque
Debug HEX(RGBA(255, 255,
    255, 255)) ; Affiche
    FFFFFFFFFFFFFFFF
```

Exemple : Couleur 24 bits vers couleur 32 bits

```
Alpha = 255 ; de 0 à 255,
    255 = aucune transparence
    (Opacité = 100 %)

; Pour les couleurs, il est
    conseillé d'utiliser des
    variables de type Quad
    (Voir remarques)
Couleur24.q  =
    ColorRequester()

Couleur32.q =
    RGBA(Red(Couleur24),
    Green(Couleur24),
    Blue(Couleur24), alpha)
Couleur32 = Couleur32 &
    $FFFFFFFF ; Mise à zéro
    des octets inutilisés

; Il est aussi possible de
    remplacer les deux lignes
    ci-dessus par:
; Couleur32 = Couleur24 |
    Alpha << 24

Debug "Rouge " +
    Red(Couleur32)
```

```
13    Debug "Vert   " +
        Green(Couleur32)
14    Debug "Bleu   " +
        Blue(Couleur32)
15    Debug "Opacité " +
        Alpha(Couleur32)
```

Voir aussi

Red() , Green() , Blue() , Alpha() , RGB()

OS Supportés

Tous

77.7 AlphaBlend

Syntaxe

```
Couleur =
    AlphaBlend(Couleur1,
    Couleur2)
```

Description

Renvoie une couleur 32 bits qui est le résultat du mélange de deux autres couleurs 32 bits.

Arguments

Couleur1 La couleur d'avant-plan qui sera mélangée à 'Couleur2'.

Couleur2 La couleur d'arrière-plan.

Valeur de retour

Renvoie la couleur mélangée.

Remarques

RGBA() peut être utilisé pour créer des couleurs 32 bits avec transparence alpha. Ces fonctions sont utiles lors des opérations de dessin 2D .

Exemple

```
1     If OpenWindow(0, 0, 0, 200,
        200, "AlphaBlend",
        #PB_Window_SystemMenu |
        #PB_Window_ScreenCentered)
2
3      If CreateImage(0, 200,
       200) And
       StartDrawing(ImageOutput(0))
```

```
4         DrawingMode(#PB_2DDrawing_Default)
5         ;DrawingMode(#PB_2DDrawing_AlphaBlend)
      ; Enlever le point virgule
      pour voir la différence.
6
7         T = 128 ; Transparence
8           Box(0   , 0   , 100,
      100, RGBA(255, 0, 0, T)) ;
      Affiche du rouge
9           Box(100, 0   , 200,
      100, RGBA(0   , 0, 0, T)) ;
      Affiche du noir
10          Box(0   , 100, 100,
      200, AlphaBlend(RGBA(0   ,
      0, 0, T), RGBA(255, 0, 0,
      T))) ; Affiche du marron
11          Box(100, 100, 200,
      200, AlphaBlend(RGBA(255,
      0, 0, T), RGBA(0   , 0, 0,
      T))) ; Affiche du bordeaux
12        StopDrawing()
13        ImageGadget(0, 0, 0,
      200, 200, ImageID(0))
14      EndIf
15
16      Repeat
17        Event =
      WaitWindowEvent()
18      Until Event =
      #PB_Event_CloseWindow
19    EndIf
20
21    ; Le changement de
      transparence donnera
      d'autres teintes.
22    ; Un autre mode comme
      DrawingMode(#PB_2DDrawing_Default)
23    ; donnera aussi d'autres
      couleurs
```

Voir aussi

RGBA()

OS Supportés

Tous

77.8 BackColor

Syntaxe

```
BackColor(Couleur)
```

Description

Définit la couleur d'arrière plan.

Arguments

Couleur La couleur d'arrière plan à utiliser pour le dessin et le texte.
Les fonctions RGB() ou RGBA() peuvent être utilisées pour définir facilement une couleur.
C'est le mode de dessin qui indique si la transparence (canal alpha) est prise en compte.
Un tableau avec les couleurs les plus courantes est disponible ici .

Valeur de retour

Aucune.

Exemple

```
1   If OpenWindow(0, 0, 0, 200, 200, "BackColor", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2     If CreateImage(0, 200, 200) And StartDrawing(ImageOutput(0))
3       BackColor(RGB(Random(255), Random(255), Random(255)))
4       FrontColor(RGB(Random(255), Random(255), Random(255)))
5         Box(0, 0, 100, 100)
6         DrawText(50, 50, "PureBasic")
7       BackColor(RGB(Random(255), Random(255), Random(255)))
8       FrontColor(RGB(Random(255), Random(255), Random(255)))
9         Box(100, 100, 100, 100)
10        DrawText(50, 100, "PureBasic")
11      StopDrawing()
12      ImageGadget(0, 0, 0, 200, 200, ImageID(0))
13    EndIf
14
15    Repeat
16      Event = WaitWindowEvent()
17    Until Event = #PB_Event_CloseWindow
18  EndIf
```

Voir aussi

FrontColor() , RGB() , RGBA() , DrawingMode()

OS Supportés

Tous

77.9 Box

Syntaxe

```
Box(X, Y, Largeur, Hauteur [,
    Couleur])
```

Description

Dessine un rectangle sur la surface de dessin en cours.

Arguments

X, Y, Largeur, Hauteur Position et dimensions du rectangle.

Couleur (optionnel) La couleur du rectangle.
S'il n'est pas spécifié, la couleur fixée par la fonction FrontColor() sera utilisée.
Les fonctions RGB() ou RGBA() peuvent être utilisées pour définir facilement une couleur.

Valeur de retour

Aucune.

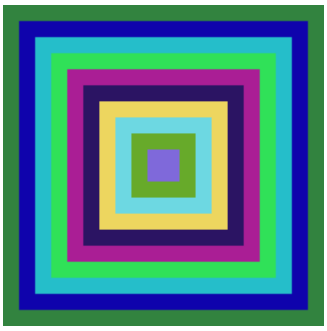
Remarques

Le mode de remplissage du rectangle est défini par la fonction DrawingMode() .

Exemple

```
1   If OpenWindow(0, 0, 0, 200,
     200, "Rectangles",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2     If CreateImage(0, 200,
     200) And
     StartDrawing(ImageOutput(0))
3       y = 0
4       For x = 0 To 95 Step 10
5         Box(x, y, 200-2*x,
     200-2*y, RGB(Random(255),
     Random(255), Random(255)))
6         y + 10          ; C'est
     équivalent à 'y = y + 10'
7       Next x
8       StopDrawing()
9       ImageGadget(0, 0, 0,
     200, 200, ImageID(0))
10    EndIf
```

```
11
12      Repeat
13        Event = WaitWindowEvent()
14      Until Event = #PB_Event_CloseWindow
15    EndIf
```



Voir aussi

RoundBox() , Line() , Circle() , Ellipse() FrontColor() , RGB() , RGBA() , DrawingMode()

OS Supportés

Tous

77.10 RoundBox

Syntaxe

```
RoundBox(X, Y, Largeur,
    Hauteur, ArrondiX,
    ArrondiY [, Couleur])
```

Description

Dessine un rectangle aux coins arrondis sur la surface de dessin en cours.

Arguments

X, Y, Largeur, Hauteur Position et dimensions du rectangle.

ArrondiX, ArrondiY Le rayon de l'arrondi des coins.

Couleur (optionnel) La couleur du rectangle.
S'il n'est pas spécifié, la couleur fixée par la fonction FrontColor() sera utilisée.
Les fonctions RGB() ou RGBA() peuvent être utilisées pour définir facilement une couleur.

Valeur de retour

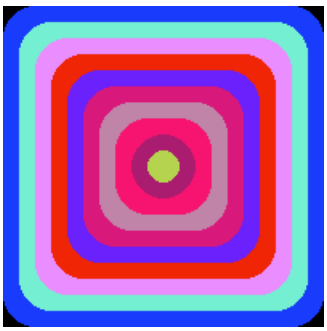
Aucune.

Remarques

Le mode de remplissage du rectangle est défini par la fonction DrawingMode() .

Exemple

```
If OpenWindow(0, 0, 0, 200, 200, "Rectangle arrondi", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  If CreateImage(0, 200, 200) And StartDrawing(ImageOutput(0))
    y = 0
    For x = 0 To 95 Step 10
      RoundBox(x, y, 200-2*x, 200-2*y, 20, 20, RGB(Random(255), Random(255), Random(255)))
      y + 10
    Next x
    StopDrawing()
    ImageGadget(0, 0, 0, 200, 200, ImageID(0))
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```



Voir aussi

Box() , Line() , Circle() , Ellipse() FrontColor() , RGB() , RGBA() , DrawingMode()

OS Supportés

Tous

77.11 Circle

Syntaxe

```
Circle(X, Y, Rayon [,
    Couleur])
```

Description

Dessine un cercle sur la surface de dessin en cours.

Arguments

X, Y Position du centre du cercle.

Rayon Le rayon du cercle.
Attention, le centre du cercle n'est pas inclus dans le rayon.

Couleur (optionnel) La couleur du cercle.
S'il n'est pas spécifié, la couleur fixée par la fonction FrontColor() sera utilisée.
Les fonctions RGB() ou RGBA() peuvent être utilisées pour définir facilement une couleur.

Valeur de retour

Aucune.

Remarques

Le mode de remplissage du rectangle est défini par la fonction DrawingMode() .

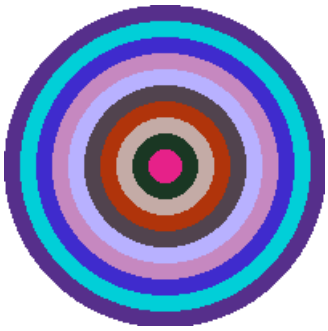
Valeur de retour

Aucune.

Exemple

```
1  If OpenWindow(0, 0, 0, 200,
     200, "Cercles",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2     If CreateImage(0, 200,
     200) And
     StartDrawing(ImageOutput(0))
3       Box(0, 0, 200, 200,
     RGB(255, 255, 255))
4       For Radius = 100 To 10
     Step -10
5         Circle(100, 100,
     Radius, RGB(Random(255),
     Random(255), Random(255)))
6       Next
7       StopDrawing()
```

```
8       ImageGadget(0, 0, 0,
    200, 200, ImageID(0))
9     EndIf
10
11    Repeat
12      Event =
    WaitWindowEvent()
13    Until Event =
    #PB_Event_CloseWindow
14  EndIf
```



Voir aussi

Box() , RoundBox() , Line() , Ellipse() FrontColor() , RGB() , RGBA() , DrawingMode()

OS Supportés

Tous

77.12 DrawImage

Syntaxe

```
DrawImage(ImageID, X, Y [,
    Largeur, Hauteur])
```

Description

Affiche une image sur la surface de dessin en cours.

Arguments

ImageID Le numéro d'identification de l'image.
L'ImageID peut être obtenu facilement en utilisant la fonction ImageID() de la bibliothèque Image.

X, Y La position du coin en haut et à gauche de l'image.

Largeur, Hauteur (optionnel) L'image sera redimensionnée en temps réel avant d'être affichée.
La taille de l'image originale n'est pas modifiée.

Valeur de retour

Aucune.

Remarques

L'image sera transparente si le mode d'affichage courant DrawingMode() est utilisé avec une des options d'alpha blending (transparence) sinon l'image est simplement copiée sur la surface de dessin en cours. Pour dessiner une image en utilisant la transparence, voir la commande DrawAlphaImage() .

Exemple

```
1   If OpenWindow(0, 0, 0, 200,
     200, "DrawImage",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2     If CreateImage(0, 200,
    200) And
    StartDrawing(ImageOutput(0))
3       DrawText(60,80,"CLIQUER
    !")
4       StopDrawing()
5       ImageGadget(0, 0, 0,
    200, 200, ImageID(0))
6     EndIf
7     If CreateImage(1, 100,
    50) And
    StartDrawing(ImageOutput(1))
8       DrawText(0,0,"PUREBASIC",
    $0000FF, $00FFFF)
9       StopDrawing()
10    EndIf
11    Repeat
12      Event =
    WaitWindowEvent()
13      If Event =
    #PB_Event_Gadget
14          StartDrawing(ImageOutput(0))
15          x=WindowMouseX(0)
16          y=WindowMouseY(0)
17          DrawImage(ImageID(1),x,y)
18          StopDrawing()
19          SetGadgetState(0,
    ImageID(0))
20      EndIf
21
22    Until Event =
    #PB_Event_CloseWindow
23  EndIf
```

OS Supportés

Tous

77.13 DrawAlphaImage

Syntaxe

```
DrawAlphaImage(ImageID, X, Y
    [, Transparence])
```

Description

Affiche une image transparente sur la surface de dessin en cours.

Arguments

ImageID Le numéro d'identification de l'image.
L'ImageID' peut être obtenu facilement en utilisant la fonction ImageID() de la bibliothèque Image.

X, Y La position du coin en haut et à gauche de l'image.

Transparence (optionnel) Le coefficient de transparence de l'image.
Sa valeur varie de 0 (complètement transparente) à 255 (complètement opaque). De fait, même les images qui n'ont pas de canal alpha peuvent être affichées de manière transparente.

Valeur de retour

Aucune.

Remarques

L'image sera fusionnée avec le fond du dessin en tenant compte de son canal alpha (même si le mode de dessin actuel n'est pas #PB_2DDrawing_AlphaBlend).
Cette commande fonctionne sur toutes les surfaces de dessin, même celles qui ne supportent pas les options de transparence (alpha blending) de DrawingMode() .
L'image sera affichée dans sa taille originale. ResizeImage() peut être utilisé pour changer la taille d'une image.
Cette commande ne peut pas être utilisée pour afficher une icône (chargée à partir d'un fichier '.ico').

Exemple

```
1  If OpenWindow(0, 0, 0, 200,
     200, "DrawImage",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
```

```
2     If CreateImage(0, 200,
      200, 32, RGB(255, 255,
      255)) And
      StartDrawing(ImageOutput(0))
3        DrawingMode(#PB_2DDrawing_Transparent)
4        DrawText(60,80,"CLIQUER
      !", RGBA(0, 255, 0, 255),
      RGBA(0, 0, 0, 0))
5        StopDrawing()
6        ImageGadget(0, 0, 0,
      200, 200, ImageID(0))
7     EndIf
8     If CreateImage(1, 100,
      50, 32,
      #PB_Image_Transparent )
      And
      StartDrawing(ImageOutput(1))
9        DrawingMode(#PB_2DDrawing_AlphaChannel)
      ; Le noir est transparent
10       Box(0, 0, 100, 50,
      $00000000)
11       DrawingMode(#PB_2DDrawing_AlphaBlend)
12       DrawText(0,0,"PUREBASIC",
      RGBA(255, 0, 0, 128),
      RGBA(0, 0, 0, 0))
13       StopDrawing()
14    EndIf
15    Repeat
16       Event =
      WaitWindowEvent()
17       If Event =
      #PB_Event_Gadget
18          StartDrawing(ImageOutput(0))
19          DrawingMode(#PB_2DDrawing_AlphaBlend)
20          x=WindowMouseX(0)
21          y=WindowMouseY(0)
22          DrawAlphaImage(ImageID(1),x,y)
23          StopDrawing()
24          SetGadgetState(0,
      ImageID(0))
25       EndIf
26
27    Until Event =
      #PB_Event_CloseWindow
28 EndIf
```

Voir aussi

DrawImage() , ImageID()

OS Supportés

Tous

77.14 DrawingBuffer

Syntaxe

```
*Resultat = DrawingBuffer()
```

Description

Renvoie le tampon (buffer) de dessin, ce qui permet la manipulation des pixels directement en mémoire.

Arguments

Aucun.

Valeur de retour

Renvoie le pointeur de données des pixels si l'accès direct aux pixels est possible ou zéro sinon.

Remarques

Cette fonction doit être appelée à nouveau si d'autres commandes de dessin de cette bibliothèque ont été utilisées depuis la dernière manipulation de pixels.
Une fois que StopDrawing() a été appelé, le tampon est invalidé et ne peut plus être utilisé.
Cette commande est destinée aux programmeurs chevronnés. Pour obtenir plus d'informations sur le buffer, les commandes suivantes peuvent être utilisées : DrawingBufferPixelFormat() et DrawingBufferPitch() .
Cela peut correspondre directement à la mémoire vidéo si la sortie est ScreenOutput() ou SpriteOutput() et permet des manipulations très rapide des pixels.
Avec ImageOutput() cette commande permet l'accès direct aux pixels de l'image.

Exemple

```
1  If OpenWindow(0, 0, 0, 200,
   200, "Buffer Image",
   #PB_Window_SystemMenu |
   #PB_Window_ScreenCentered)
2   If CreateImage(0, 200,
   200, 32) And
   StartDrawing(ImageOutput(0))
3     DrawText(60,80,"PUREBASIC",
   RGB(255, 255, 0))
4     Debug "Adresse mémoire
   : " + Str(DrawingBuffer())
5     Debug "Longueur réelle
   d'une ligne en octet : " +
   Str(DrawingBufferPitch())
```

```
6       Debug "Format de pixel
    : " +
    Str(DrawingBufferPixelFormat())
7       StopDrawing()
8       ImageGadget(0, 0, 0,
    200, 200, ImageID(0))
9     EndIf
10
11    Repeat
12      Event =
    WaitWindowEvent()
13    Until Event =
    #PB_Event_CloseWindow
14  EndIf
```

Voir aussi

DrawingBufferPixelFormat() , DrawingBufferPitch()

OS Supportés

Tous

77.15 DrawingBufferPitch

Syntaxe

```
Resultat =
    DrawingBufferPitch()
```

Description

Renvoie la longueur réelle d'une ligne du buffer courant, en octets.

Arguments

Aucun.

Valeur de retour

Renvoie la longueur en octets d'une ligne de la surface de dessin, y compris toutes les données annexes de chaque pixel.

Remarques

DrawingBuffer() doit être appelé avant d'utiliser cette fonction.

Exemple

```
1   If OpenWindow(0, 0, 0, 200,
     200, "Buffer Image",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
```

```
2     If CreateImage(0, 200,
      200, 32) And
      StartDrawing(ImageOutput(0))
3        DrawText(60,80,"PUREBASIC",
      RGB(255, 255, 0))
4        Debug "Adresse mémoire
      : " + Str(DrawingBuffer())
5        Debug "Longueur réelle
      d'une ligne en octet : " +
      Str(DrawingBufferPitch())
6        Debug "Format de pixel
      : " +
      Str(DrawingBufferPixelFormat())
7        StopDrawing()
8        ImageGadget(0, 0, 0,
      200, 200, ImageID(0))
9     EndIf
10
11    Repeat
12       Event =
      WaitWindowEvent()
13    Until Event =
      #PB_Event_CloseWindow
14  EndIf
```

Voir aussi

DrawingBuffer() ,
DrawingBufferPixelFormat()

OS Supportés

Tous

77.16 DrawingBufferPixelFormat

Syntaxe

```
Resultat =
    DrawingBufferPixelFormat()
```

Description

Renvoie le format de pixel.

Arguments

Aucun.

Valeur de retour

Peut être la combinaison des valeurs suivantes :

```
#PB_PixelFormat_8Bits
 : 1 octet par pixel,
 palettisé
```

```
#PB_PixelFormat_15Bits
 : 2 octets par pixel
#PB_PixelFormat_16Bits
 : 2 octets par pixel
#PB_PixelFormat_24Bits_RGB
 : 3 octets par pixel
 (RRGGBB)
#PB_PixelFormat_24Bits_BGR
 : 3 octets par pixel
 (BBGGRR)
#PB_PixelFormat_32Bits_RGB
 : 4 octets par pixel
 (RRGGBB)
#PB_PixelFormat_32Bits_BGR
 : 4 octets par pixel
 (BBGGRR)
#PB_PixelFormat_ReversedY
 : Les lignes sont
 inversées en hauteur (la
 dernière ligne est la
 première)
```

Remarques

DrawingBuffer() doit être appelé avant d'utiliser cette fonction.

Exemple

Les exemples suivants montrent comment gérer différents formats :

```
1   If
     DrawingBufferPixelFormat()
     =
     #PB_PixelFormat_32Bits_RGB
     | #PB_PixelFormat_ReversedY
2     ; RGB 32 bits en mode
     inversé
3   EndIf
4
5   If
     DrawingBufferPixelFormat()
     =
     #PB_PixelFormat_32Bits_RGB
6     ; RGB 32 bits
7   EndIf
8
9   If
     DrawingBufferPixelFormat()
     &
     #PB_PixelFormat_32Bits_RGB
10    ; RGB 32 bits (mode
     inversé ou non)
11  EndIf
```

Exemple

```
If OpenWindow(0, 0, 0, 200, 200, "Buffer Image", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  If CreateImage(0, 200, 200, 32) And StartDrawing(ImageOutput(0))
    DrawText(60,80,"PUREBASIC", RGB(255, 255, 0))
    Debug "Adresse mémoire : " + Str(DrawingBuffer())
    Debug "Longueur réelle d'une ligne en octet : " + Str(DrawingBufferPitch())
    Debug "Format de pixel : " + Str(DrawingBufferPixelFormat())
    StopDrawing()
    ImageGadget(0, 0, 0, 200, 200, ImageID(0))
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

StartDrawing() , DrawingBufferPitch()

OS Supportés

Tous

77.17 DrawingFont

Syntaxe

```
DrawingFont(PoliceID)
```

Description

Change la police de caractères de la surface de dessin en cours.

Arguments

PoliceID Le numéro d'identification de la police à utiliser.
Il peut être obtenu par la fonction FontID() de la bibliothèque 'Font'.

Remarques

Tous les textes affichés ensuite le seront avec cette nouvelle police.
Avant d'utiliser cette fonction, la police doit être chargée à l'aide de la fonction LoadFont() .

Exemple

```
1   If OpenWindow(0, 0, 0, 200,
     200, "DrawingFont",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2     Police = LoadFont(0,
     "Arial", 20,
     #PB_Font_Underline)
3     If CreateImage(0, 200,
     200) And
     StartDrawing(ImageOutput(0))
4       DrawText(50,30,"PUREBASIC",
     RGB(255, 255, 0))
5       DrawingFont(Police)
6       DrawText(5,80,"PUREBASIC",
     RGB(255, 255, 0))
7       StopDrawing()
8       ImageGadget(0, 0, 0,
     200, 200, ImageID(0))
9     EndIf
10
11    Repeat
12      Event =
     WaitWindowEvent()
13    Until Event =
     #PB_Event_CloseWindow
14  EndIf
```

Voir aussi

LoadFont() , FontID()

OS Supportés

Tous

77.18 DrawingMode

Syntaxe

```
DrawingMode(Mode)
```

Description

Change le mode d'affichage pour les textes et les dessins.

Arguments

Mode Peut être une combinaison des valeurs suivantes :

#PB_2DDrawing_Default

C'est le mode de dessin par défaut quand StartDrawing() est appelé. Le texte est affiché avec un fond solide et les formes géométriques sont pleines.
Si la surface de dessin gère un canal alpha, les opérations de dessin n'affecteront que les composantes couleurs et laisseront le canal alpha inchangé.

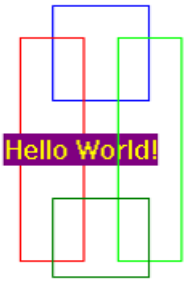
#PB_2DDrawing_Transparent

Si ce mode est activé alors le fond du texte affiché avec DrawText() sera transparent.

#PB_2DDrawing_XOr

Active le mode XOR. Toutes les couleurs des graphiques seront inversées par rapport à la couleur du fond. La couleur courante n'a plus d'importance et n'est pas prise en compte. Ce mode ne peut pas être combiné avec les mode alpha ci-dessous.
Note : Ce mode ne fonctionne pas avec la sortie PrinterOutput() .

#PB_2DDrawing_Outlined
Si ce mode est activé, seuls les contours des figures géométriques seront dessinés (les figures ne seront pas pleines). Ceci s'applique aux commandes telles que Circle() (), Box() , etc.



Note : Les modes suivants ne fonctionnent qu'avec ImageOutput() et CanvasOutput() . Ils sont ignorés pour toutes les autres sorties :
#PB_2DDrawing_AlphaBlend
Les opérations de dessin seront fusionnées avec le fond en utilisant le canal alpha pour gérer la transparence de chaque pixel. RGBA() peut être utilisé pour définir une couleur qui intègre un degré de transparence pour les commandes comme FrontColor() , Box() , DrawText() etc.

#PB_2DDrawing_AlphaClip
Les opérations de dessin seront fusionnées avec le fond en utilisant le canal alpha pour gérer la

transparence de chaque pixel, comme avec le mode #PB_2DDrawing_AlphaBlend mode, mais avec la différence que le fond agit comme un masque. Cela signifie que les zones du fond qui étaient transparentes avant la fusion le resteront après. Si la surface de dessin ne gère pas de canal alpha, alors ce mode agit exactement comme #PB_2DDrawing_AlphaBlend.

#PB_2DDrawing_AlphaChannel

Les opérations de dessin ne modifieront que les valeurs du canal alpha de la surface de dessin. Toutes les informations concernant les couleurs seront ignorées. Par exemple, dessiner un cercle avec une couleur RGBA (0, 0, 0, 0) fera un 'trou' en forme de cercle dans la surface, car cette zone sera alors complètement transparente. Si la surface de dessin n'a pas de canal alpha, comme le CanvasGadget, alors les opérations de dessin n'auront aucun effet dans ce mode.

#PB_2DDrawing_AllChannels

Les opérations de dessin modifieront les composantes couleurs ainsi que les valeurs du canal alpha de la surface de dessin. Il n'y a plus de mélange lors des opérations de dessin. Ce mode est équivalent à l'enchainement des modes #PB_2DDrawing_Default et de #PB_2DDrawing_AlphaChannel. Si la surface de dessin n'a pas de canal alpha, comme le CanvasGadget, alors ce mode est identique à #PB_2DDrawing_Default.

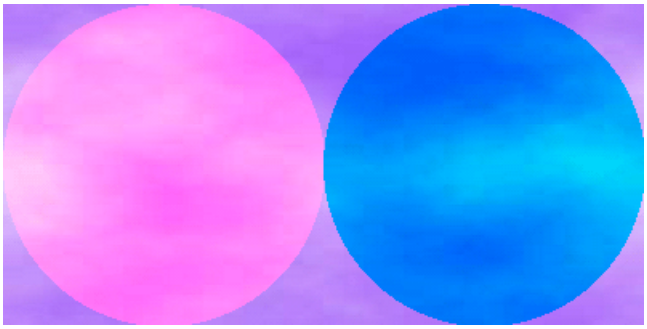
#PB_2DDrawing_Gradient

Ce mode permet de dessiner un dégradé à la place d'une couleur solide. La forme du dégradé peut être modifiée à l'aide des commandes telles que LinearGradient() , CircularGradient() etc. Les couleurs qui composeront le dégradé peuvent être définies avec GradientColor() .

Les paramètres de couleur spécifiés pour chaque opération de dessin seront ignorés dans ce mode. Il est possible de combiner ce mode avec les modes alpha ci-dessus pour avoir des dégradés semi-transparents.

#PB_2DDrawing_CustomFilter

Dans ce mode, la couleur de chaque pixel pourra être définie par la procédure spécifiée par CustomFilterCallback() . Cela permet d'implémenter des rendus complètement libres tout en utilisant des formes géométriques pour le dessin.



Valeur de retour

Aucune.

Remarques

Pour utiliser plusieurs modes en même temps, il suffit d'utiliser l'opérateur '||' (OR). Exemple pour des dessins détourés en mode 'xor' :

```
DrawingMode(#PB_2DDrawing_Outlined
  | #PB_2DDrawing_XOr)
```

Voir aussi

FrontColor() , BackColor()

OS Supportés

Tous

77.19 DrawRotatedText

Syntaxe

```
DrawRotatedText(X, Y, Texte$,
    Angle.f [, Couleur])
```

Description

Affiche une chaîne de caractères avec un angle donné sur la surface de dessin en cours.

Arguments

X, Y L'emplacement du coin en haut et à gauche du texte.
C'est aussi le centre de rotation du texte.

Texte$ Le texte à dessiner.

Angle.f L'angle de rotation en degrés (sens anti-horaire).

Couleur (optionnel) La couleur du texte.
Si ce paramètre n'est pas spécifié alors la couleur par défaut FrontColor() sera utilisée.
Cette couleur peut être au format RGB() ou RGBA() .
Le fond du texte est toujours transparent.

Valeur de retour

Aucune.

Exemple

```
1   If OpenWindow(0, 0, 0, 200,
     200, "DrawRotatedText
     Exemple",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2     If CreateImage(0, 200,
     200) And
     StartDrawing(ImageOutput(0))
3       Box(0, 0, 200, 200,
     RGB(255, 255, 255))
4       For Angle = 0 To 360
     Step 45
5         DrawRotatedText(100,
     100, "Hello World!",
     Angle, RGB(0, 0, 0))
6       Next Angle
7       StopDrawing()
8       ImageGadget(0, 0, 0,
     200, 200, ImageID(0))
9     EndIf
10
11    Repeat
```

```
12        Event =
      WaitWindowEvent()
13      Until Event =
      #PB_Event_CloseWindow
14    EndIf
```

Voir aussi

DrawText() , DrawingFont() , FrontColor()

OS Supportés

Tous

77.20 FillArea

Syntaxe

```
FillArea(X, Y, CouleurBord [,
    Couleur])
```

Description

Remplit une zone de dessin avec une couleur définie.

Arguments

X, Y Le remplissage commence à la position X, Y.

CouleurBord Le remplissage s'arrête quand il rencontre la couleur 'CouleurBord'.
Si défini avec la valeur -1 alors la zone est remplie avec la couleur trouvée en 'X, Y' jusqu'à ce qu'une couleur différente soit rencontrée.
Sur les images en 32 bits, le canal alpha est ignoré pour savoir si un pixel fait office de bordure ou non.

Couleur (optionnel) Couleur de remplissage.
Si le paramètre n'est pas précisé, la couleur définie par FrontColor() sera utilisée par défaut.

Les fonctions RGB() ou RGBA() peuvent être utilisées pour définir facilement une couleur.

Valeur de retour

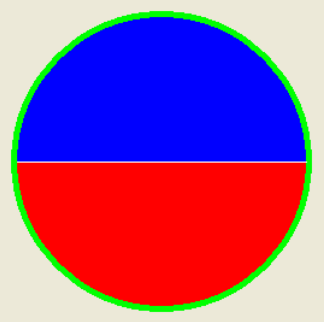
Aucune.

Remarques

Cette commande ne fonctionne pas avec PrinterOutput() .

Exemple

```
If OpenWindow(0, 0, 0, 300, 300, "FillArea", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
 If CreateImage(0, 300, 300) And StartDrawing(ImageOutput(0))
    Box(0, 0, 300, 300, RGB(255, 255, 255))

    Circle(150, 150, 125,$00FF00)
    Circle(150, 150, 120,$FF0000)
    LineXY(30, 150, 270, 150, $FFFFFF)
    FillArea(150, 155, -1, $0000FF) ; Remplacez -1 par $00FF00 et comparez le résultat.

    StopDrawing()
    ImageGadget(0, 0, 0, 300, 300, ImageID(0))
 EndIf

 Repeat
    Event = WaitWindowEvent()
 Until Event = #PB_Event_CloseWindow
EndIf
```



Voir aussi

FrontColor()

OS Supportés

Tous

77.21 GrabDrawingImage

Syntaxe

```
Resultat =
    GrabDrawingImage(#Image,
    X, Y, Largeur, Hauteur)
```

Description

Crée une nouvelle image en copiant une zone de la surface dessin en cours.

Arguments

#Image Le numéro de la nouvelle image. #PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur Position et simensions de la zone à copier. La nouvelle image aura les même dimensions.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé comme paramètre "#Image" alors le numéro de la nouvelle image est renvoyé dans 'Resultat'.

Remarques

Cette commande ne fonctionne pas avec PrinterOutput() .
Toute partie de la zone spécifiée qui se trouve en dehors de la surface de dessin sera indéfinie dans l'image créée. De même, si la sortie est WindowOutput() , toute partie de la fenêtre qui n'est pas visible à l'écran peut être indéfinie dans l'image créée.

Exemple

```
1   If OpenWindow(0, 0, 0, 410,
    100, "GrabDrawingImage",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
```

```
2     If CreateImage(0, 200,
    100) And
    StartDrawing(ImageOutput(0))
3       DrawText(50,30,"PUREBASIC",
    RGB(255, 255, 0))
4       GrabDrawingImage(1,0,0,100,100)
5       StopDrawing()
6       ImageGadget(0, 0, 0,
    200, 100, ImageID(0))
7       ImageGadget(1, 210, 0,
    200, 100, ImageID(1))
8     EndIf
9
10    Repeat
11      Event =
    WaitWindowEvent()
12    Until Event =
    #PB_Event_CloseWindow
13  EndIf
```

Voir aussi

GrabImage()

OS Supportés

Tous

77.22 StartDrawing

Syntaxe

```
Resultat =
    StartDrawing(OutputID)
```

Description

Change la surface de dessin par celle mentionnée par 'OutputID'. Après avoir utilisé cette fonction, toutes les commandes de dessin seront exécutées sur la nouvelle surface.

Arguments

OutputID Les dessins seront rendus directement sur :

```
WindowOutput()
: La fenêtre.
ScreenOutput()
: L'écran (utile pour les
  jeux).
SpriteOutput()
: Le sprite (utile pour
  les jeux).
ImageOutput()
```

```
    : L'image (voir aussi
    CreateImage()
)
  PrinterOutput()
 : L'imprimante.
  CanvasOutput()
  : Le CanvasGadget()
.
  TextureOutput()
 : La texture (pour les jeux
   3D).
```

Valeur de retour

Renvoie une valeur non nulle si le dessin est possible, zéro sinon.

Remarques

Lorsque tous les dessins sont terminés, la fonction StopDrawing() doit être appelée.
La couleur d'arrière plan est le noir (RGB(0,0,0)) et la couleur d'avant plan est le blanc (RGB(255,255,255)).
Si "Activer la gestion des Threads" est coché dans les options du compilateur alors chaque thread a sa propre surface de dessin, ce qui signifie que deux threads peuvent dessiner sur des surfaces de dessin différentes en même temps.

Exemple

```
1   If OpenWindow(0, 0, 0, 200,
     100, "StartDrawing",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2     If CreateImage(0, 200,
     100) And
     StartDrawing(ImageOutput(0))
3       DrawText(50,30,"PUREBASIC",
     RGB(255, 255, 0))
4       StopDrawing()
5       ImageGadget(0, 0, 0,
     200, 100, ImageID(0))
6     EndIf
7
8     Repeat
9       Event =
     WaitWindowEvent()
10    Until Event =
     #PB_Event_CloseWindow
11  EndIf
```

Voir aussi

StopDrawing()

OS Supportés

Tous

77.23 DrawText

Syntaxe

```
Resultat = DrawText(X, Y,
    Texte$ [, CouleurTexte [,
    CouleurFond]])
```

Description

Affiche une chaîne de caractères sur la surface de dessin en cours.

Arguments

X, Y Position du texte.

Texte$ Le texte à afficher.

CouleurTexte (optionnel) Couleur du texte.
Si le paramètre n'est pas précisé, la couleur définie par FrontColor() sera utilisée par défaut.
Les fonctions RGB() ou RGBA() peuvent être utilisées pour définir facilement une couleur.

CouleurFond (optionnel) Couleur de fond.
Si le paramètre n'est pas précisé, la couleur définie par BackColor() sera utilisée par défaut.
Les fonctions RGB() ou RGBA() peuvent être utilisées pour définir facilement une couleur.
Si le DrawingMode() courant utilise l'option #PB_2DDrawing_Transparent alors ce paramètre est ignoré et le fond est transparent.
Si DrawingMode() est configuré pour avoir un fond opaque et que le mode de dessin courant utilise le canal alpha alors le texte est d'abord mélangé au fond et ensuite appliqué sur la zone de dessin.

Valeur de retour

Renvoie la nouvelle position en X du curseur texte, juste après le dernier caractère affiché.

Exemple

```
1   If OpenWindow(0, 0, 0, 200,
    200, "DrawText Exemple",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2    If CreateImage(0, 200,
    200) And
    StartDrawing(ImageOutput(0))
3       DrawingMode(#PB_2DDrawing_Transparent)
4       Box(0, 0, 200, 200,
    RGB(255, 255, 255))
5       For i = 1 To 30
6         DrawText(Random(200),
    Random(200), "Hello
    World!", RGB(Random(255),
    Random(255), Random(255)))
7       Next i
8       StopDrawing()
9       ImageGadget(0, 0, 0,
    200, 200, ImageID(0))
10   EndIf
11
12   Repeat
13      Event =
    WaitWindowEvent()
14   Until Event =
    #PB_Event_CloseWindow
15  EndIf
```

Voir aussi

DrawRotatedText() , DrawingFont() , FrontColor() , BackColor()

OS Supportés

Tous

77.24 Ellipse

Syntaxe

```
Ellipse(X, Y, RayonX, RayonY
    [, Couleur])
```

Description

Dessine une ellipse sur la surface de dessin en cours.

Arguments

X, Y La position du centre de l'ellipse.

RayonX, RayonY Dimension de l'ellipse. RayonX est le demi grand axe de l'ellipse et RayonY est le demi petit axe de l'ellipse.
Attention, le centre l'ellipse qui est un pixel, n'est pas inclu dans ces valeurs.

Couleur (optionnel) Couleur de l'ellipse. Si ce paramètre n'est pas spécifié, la couleur fixée par la fonction FrontColor() sera utilisée.
Les fonctions RGB() ou RGBA() peuvent être utilisées pour définir facilement une couleur.
Le mode de remplissage est contrôlé par DrawingMode() .

Valeur de retour

Aucune.

Exemple

```
1   If OpenWindow(0, 0, 0, 200, 200, "Ellipse", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2     If CreateImage(0, 200, 200) And StartDrawing(ImageOutput(0))
3       Box(0, 0, 200, 200, RGB(255, 255, 255))
4       For radius=50 To 10 Step -10
5         Ellipse(100, 100, radius*2, radius, RGB(Random(255), Random(255), Random(255)))
6       Next radius
7       StopDrawing()
8       ImageGadget(0, 0, 0, 200, 200, ImageID(0))
9     EndIf
10
11    Repeat
12      Event = WaitWindowEvent()
13    Until Event = #PB_Event_CloseWindow
14  EndIf
```

Voir aussi

Box() , RoundBox() , Line() , Circle()
FrontColor() , RGB() , RGBA() ,
DrawingMode()

OS Supportés

Tous

77.25 FrontColor

Syntaxe

```
FrontColor(Couleur)
```

Description

Fixe la couleur d'avant plan.

Arguments

Couleur La nouvelle couleur d'avant plan à utiliser pour le dessin et le texte.
Les fonctions RGB() ou RGBA() peuvent être utilisées pour définir facilement une couleur.
L'utilisation de la transparence (canal alpha) dépend du mode de dessin .
Un tableau avec les couleurs les plus courantes est disponible ici .

Valeur de retour

Aucune.

Exemple

```
1  If OpenWindow(0, 0, 0, 200,
     200, "FrontColor",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2    If CreateImage(0, 200,
     200) And
     StartDrawing(ImageOutput(0))
```

```
3         BackColor(RGB(Random(255),
      Random(255), Random(255)))
4         FrontColor(RGB(Random(255),
      Random(255), Random(255)))
5           Box(0, 0, 100, 100)
6           DrawText(50, 50,
      "PureBasic")
7         BackColor(RGB(Random(255),
      Random(255), Random(255)))
8         FrontColor(RGB(Random(255),
      Random(255), Random(255)))
9           Box(100, 100, 100,
      100)
10          DrawText(50, 100,
      "PureBasic")
11        StopDrawing()
12        ImageGadget(0, 0, 0,
      200, 200, ImageID(0))
13      EndIf
14
15      Repeat
16        Event =
      WaitWindowEvent()
17      Until Event =
      #PB_Event_CloseWindow
18    EndIf
```

Voir aussi

BackColor() , RGB() , RGBA() , DrawingMode()

OS Supportés

Tous

77.26 Line

Syntaxe

```
Line(X, Y, Largeur, Hauteur
    [, Couleur])
```

Description

Trace une ligne sur la surface de dessin en cours.

Arguments

X, Y L'origine de la ligne.

Largeur, Hauteur Dimension de la ligne. Ces valeurs incluent le point d'origine 'X, Y'.
Une hauteur de '1' dessine une ligne horizontale tandis qu'une hauteur de '0' ne dessinera rien du tout.

Couleur (optionnel) Couleur de la ligne. Si ce paramètre n'est pas spécifié, la couleur fixée par la fonction FrontColor() sera utilisée.
Les fonctions RGB() ou RGBA() peuvent être utilisées pour définir facilement une couleur.

Valeur de retour

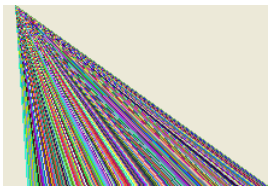
Aucune.

Remarques

Pour dessiner une ligne avec des coordonnées de départ et de fin de ligne, utiliser la fonction LineXY() .

Exemple

```
1   If OpenWindow(0, 0, 0, 200, 200, "Line", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2     If CreateImage(0, 200, 200) And StartDrawing(ImageOutput(0))
3       Box(0, 0, 200, 200, RGB(255, 255, 255))
4       For Width = 1 To 180 Step 5
5         Line(10, 10, Width, 180, RGB(Random(255), Random(255), Random(255)))
6       Next Width
7       StopDrawing()
8       ImageGadget(0, 0, 0, 200, 200, ImageID(0))
9     EndIf
10
11    Repeat
12      Event = WaitWindowEvent()
13    Until Event = #PB_Event_CloseWindow
14  EndIf
```



Voir aussi

LineXY() , Box() , RoundBox() , Ellipse() , Circle() , FrontColor() , RGB() , RGBA()

OS Supportés

Tous

77.27 LineXY

Syntaxe

```
LineXY(X1, Y1, X2, Y2 [,
    Couleur])
```

Description

Trace une ligne sur la surface de dessin en cours en fonction de ses coordonnées de début et de fin.

Arguments

X1, Y1 Position du point de départ.

X2, Y2 Position du point d'arrivée.

Couleur (optionnel) Couleur de la ligne. Si ce paramètre n'est pas spécifié, la couleur fixée par la fonction FrontColor() sera utilisée.
Les fonctions RGB() ou RGBA() peuvent être utilisées pour définir facilement une couleur.

Valeur de retour

Aucune.

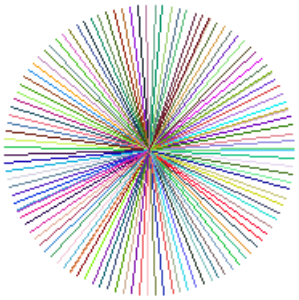
Remarques

Pour dessiner une ligne en fonction de ses dimensions, utiliser la fonction Line() .

Exemple

```
1   If OpenWindow(0, 0, 0, 200,
    200, "LineXY",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2    If CreateImage(0, 200,
    200) And
    StartDrawing(ImageOutput(0))
3      Box(0, 0, 200, 200,
    RGB(255, 255, 255))
4      For Angle = 0 To 360
    Step 3
5        LineXY(100, 100,
    100+Cos(Angle)*90,
    100+Sin(Angle)*90,
    RGB(Random(255),
    Random(255), Random(255)))
6      Next Angle
```

```
7         StopDrawing()
8         ImageGadget(0, 0, 0,
      200, 200, ImageID(0))
9       EndIf
10
11      Repeat
12        Event =
      WaitWindowEvent()
13      Until Event =
      #PB_Event_CloseWindow
14    EndIf
```



Voir aussi

Line() , Box() , RoundBox() , Ellipse() , Circle() , FrontColor() , RGB() , RGBA()

OS Supportés

Tous

77.28 Plot

Syntaxe

```
Plot(X, Y [, Couleur])
```

Description

Affiche un point sur la surface de dessin en cours.

Arguments

X, Y Position du point (pixel).
Les coordonnées X, Y doivent être obligatoirement à l'intérieur de la surface de dessin, car il n'y a pas de contrôle pour des raisons de rapidité. OutputWidth() et OutputHeight() peuvent être utilisés pour cela.
Cette commande n'est pas affectée par le clipping (écrêtage) imposé par ClipOutput() .

Couleur (optionnel) Couleur du point. Si ce paramètre n'est pas spécifié, la couleur fixée par la fonction FrontColor() sera utilisée.
Les fonctions RGB() ou RGBA() peuvent être utilisées pour définir facilement une couleur.

Valeur de retour

Aucune.

Exemple

```
1   If OpenWindow(0, 0, 0, 200, 200, "Plot", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2    If CreateImage(0, 200, 200) And StartDrawing(ImageOutput(0))
3       For x = 0 To 199
4         For y = 0 To 199
5           Plot(X, Y, RGB(Random(255), Random(255), Random(255)))
6         Next y
7       Next x
8       StopDrawing()
9       ImageGadget(0, 0, 0, 200, 200, ImageID(0))
10   EndIf
11
12   Repeat
13     Event = WaitWindowEvent()
14   Until Event = #PB_Event_CloseWindow
15  EndIf
```

Voir aussi

Point() , FrontColor()

OS Supportés

Tous

77.29 Point

Syntaxe

```
Resultat = Point(X, Y)
```

Description

Renvoie la couleur d'un point de la surface de dessin en cours.

Arguments

X, Y Position du pixel.
Les coordonnées X, Y doivent être obligatoirement à l'intérieur de la surface de dessin, car il n'y a pas de contrôle pour des raisons de rapidité. OutputWidth() et OutputHeight() peuvent être utilisés pour cela.
Cette commande n'est pas affectée par le clipping (écrêtage) imposé par ClipOutput() .

Valeur de retour

Renvoie la couleur du pixel.
Cette couleur contiendra une valeur alpha uniquement si la zone de dessin est en 32 bits et que le DrawingMode() est configuré sur un des modes alpha (transparence). Sinon la valeur alpha de la couleur sera égale à zéro.

Remarques

Cette commande ne fonctionne pas avec PrinterOutput() .

Exemple

```
1   If OpenWindow(0, 0, 0, 200, 200, "Point", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2    If CreateImage(0, 200, 200) And StartDrawing(ImageOutput(0))
3       Box(0, 0, 200, 200, RGB(255, 0, 0))
4       Debug Point(100, 100)
5       StopDrawing()
6       ImageGadget(0, 0, 0, 200, 200, ImageID(0))
7    EndIf
8
9    Repeat
10      Event = WaitWindowEvent()
11   Until Event = #PB_Event_CloseWindow
12  EndIf
```

Voir aussi

Plot() , Red() , Green() , Blue() , Alpha()

OS Supportés

Tous

77.30 StopDrawing

Syntaxe

```
StopDrawing()
```

Description

Lorsque tous les affichages graphiques ont été réalisés, cette fonction doit être appelée afin de terminer le dessin et libérer les ressources.

Arguments

Aucun.

Valeur de retour

Aucune.

Exemple

```
If OpenWindow(0, 0, 0, 200, 100, "StopDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  If CreateImage(0, 200, 100) And StartDrawing(ImageOutput(0))
    DrawText(50,30,"PUREBASIC", RGB(255, 255, 0))
    StopDrawing()
    ImageGadget(0, 0, 0, 200, 100, ImageID(0))
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

StartDrawing()

OS Supportés

Tous

77.31 TextHeight

Syntaxe

```
Resultat = TextHeight(Texte$)
```

Description

Renvoie la hauteur d'une chaîne de caractère sur la surface de dessin en cours.

Arguments

Texte$ Le texte à mesurer.

Valeur de retour

Renvoie la hauteur du texte donné en pixels avec la police de caractères actuelle.

Exemple

```
If OpenWindow(0, 0, 0, 200, 200, "Dimensions texte", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  If CreateImage(0, 200, 200) And StartDrawing(ImageOutput(0))
    DrawText(50, 80, "PUREBASIC", RGB(255, 255, 0))
    Debug "Longueur du texte en pixels : " + Str(TextWidth("PUREBASIC"))
    Debug "Hauteur du texte en pixels  : " + Str(TextHeight("PUREBASIC"))
    StopDrawing()
    ImageGadget(0, 0, 0, 200, 200, ImageID(0))
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

TextWidth() , DrawingFont()

OS Supportés

Tous

77.32 TextWidth

Syntaxe

```
Resultat = TextWidth(Texte$)
```

Description

Renvoie la longueur d'une chaîne de caractère sur la surface de dessin en cours.

Arguments

Texte$ Le texte à mesurer.

Valeur de retour

Renvoie la longueur du texte donné en pixels avec la police de caractères actuelle.

Exemple

```
If OpenWindow(0, 0, 0, 200, 200, "Dimensions texte", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  If CreateImage(0, 200, 200) And StartDrawing(ImageOutput(0))
    DrawText(50, 80, "PUREBASIC", RGB(255, 255, 0))
    Debug "Longueur du texte en pixels : " + Str(TextWidth("PUREBASIC"))
    Debug "Hauteur du texte en pixels  : " + Str(TextHeight("PUREBASIC"))
    StopDrawing()
    ImageGadget(0, 0, 0, 200, 200, ImageID(0))
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

TextHeight() , DrawingFont()

OS Supportés

Tous

77.33 OutputDepth

Syntaxe

```
Resultat = OutputDepth()
```

Description

Renvoie le niveau de profondeur de couleurs de la surface de dessin en cours.

Arguments

Aucun.

Valeur de retour

Renvoie la profondeur de couleur en bits par pixel.

Exemple

```
If OpenWindow(0, 0, 0, 200, 200, "Dimensions", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  If CreateImage(0, 200, 200) And StartDrawing(ImageOutput(0))
    DrawText(50, 80, "PUREBASIC", RGB(255, 255, 0))
    Debug "Longueur de la surface de dessin en pixels : " + Str(OutputWidth())
    Debug "Hauteur de la surface de dessin en pixels  : " + Str(OutputHeight())
    Debug "Profondeur de couleur de la surface de dessin en bits par pixels : " + Str(OutputDepth())
    StopDrawing()
    ImageGadget(0, 0, 0, 200, 200, ImageID(0))
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

OutputWidth() , OutputHeight()

OS Supportés

Tous

77.34 OutputWidth

Syntaxe

```
Resultat = OutputWidth()
```

Description

Renvoie la largeur de la surface de dessin en cours.

Arguments

Aucun.

Valeur de retour

Renvoie la largeur de la surface de dessin, en pixels.

Exemple

```
If OpenWindow(0, 0, 0, 200, 200, "Dimensions", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  If CreateImage(0, 200, 200) And StartDrawing(ImageOutput(0))
    DrawText(50, 80, "PUREBASIC", RGB(255, 255, 0))
    Debug "Longueur de la surface de dessin en pixels : " + Str(OutputWidth())
    Debug "Hauteur de la surface de dessin en pixels  : " + Str(OutputHeight())
    Debug "Profondeur de couleur de la surface de dessin en bits par pixels : " + Str(OutputDepth())
    StopDrawing()
    ImageGadget(0, 0, 0, 200, 200, ImageID(0))
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

OutputHeight() , OutputDepth()

OS Supportés

Tous

77.35 OutputHeight

Syntaxe

```
Resultat = OutputHeight()
```

Description

Renvoie la hauteur de la surface de dessin courante.

Arguments

Aucun.

Valeur de retour

Renvoie la largeur de la surface de dessin, en pixels.

Exemple

```
If OpenWindow(0, 0, 0, 200, 200, "Dimensions", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  If CreateImage(0, 200, 200) And StartDrawing(ImageOutput(0))
    DrawText(50, 80, "PUREBASIC", RGB(255, 255, 0))
    Debug "Longueur de la surface de dessin en pixels : " + Str(OutputWidth())
    Debug "Hauteur de la surface de dessin en pixels  : " + Str(OutputHeight())
    Debug "Profondeur de couleur de la surface de dessin en bits par pixels : " + Str(OutputDepth())
    StopDrawing()
    ImageGadget(0, 0, 0, 200, 200, ImageID(0))
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

OutputWidth() , OutputDepth()

OS Supportés

Tous

77.36 CustomFilterCallback

Syntaxe

```
CustomFilterCallback(@CallbackFiltre())
```

Description

Spécifie une procédure qui sera appelée pour chaque pixel affiché par une opération de dessin en mode #PB_2DDrawing_CustomFilter.

Arguments

@CallbackFiltre() L'adresse de la procédure de callback.
La procédure doit avoir la forme suivante :

```
1  Procedure CustomCallback(X, Y, CouleurSource, CouleurCible)
2    ;
3    ; Calcule la nouvelle Couleur à partir des entrées
4    ; et la renvoie avec ProcedureReturn
5    ;
6    ProcedureReturn Couleur
7  EndProcedure
```

La procédure sera appelée pour chaque pixel affiché par les commandes telles que Line() , Box() ou DrawText() .
'CouleurSource' représente la couleur donnée à l'opération de dessin, et 'CouleurCible' représente la couleur du pixel cible aux coordonnées X, Y de la surface de dessin.
Ces deux couleurs sont toujours en 32 bits (avec canal alpha) indépendamment de la résolution de la surface de dessin.
Les coordonnées X et Y sont toujours orientées par rapport au coin supérieur gauche de la surface de dessin.
Les coordonnées ne sont pas affectées par les appels à SetOrigin() ou à ClipOutput() .

Valeur de retour

Aucune.

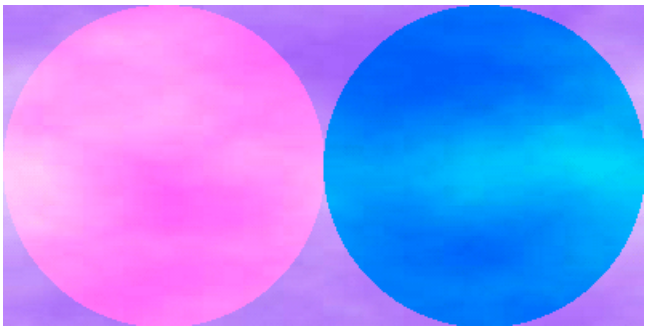
Remarques

Cette procédure sera appelée de nombreuses fois (pour chaque pixel à appeler), donc elle devra être la plus courte et la plus optimisée possible, sinon l'impact sur les performances de dessin sera conséquent.
Note : le mode de dessin pour #PB_2DDrawing_CustomFilter fonctionne uniquement avec ImageOutput() et CanvasOutput() .

Exemple

```
Procedure CallbackFiltre(X, Y, CouleurSource, CouleurCible)
  ; Ne modifie que la composante rouge de la Source
  ProcedureReturn RGBA(Red(CouleurSource), Green(CouleurCible), Blue(CouleurCible), Alpha(CouleurCible))
EndProcedure

UseJPEGImageDecoder()

If OpenWindow(0, 0, 0, 400, 200, "CustomFilterCallback", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  LoadImage(1, #PB_Compiler_Home + "Examples/3D/Data/Textures/clouds.jpg")

  If CreateImage(0, 400, 200) And StartDrawing(ImageOutput(0))
    DrawImage(ImageID(1), 0, 0, 400, 200)

    DrawingMode(#PB_2DDrawing_CustomFilter)
    CustomFilterCallback(@CallbackFiltre())
    Circle(100, 100, 100, $0000FF)
    Circle(300, 100, 100, $000000)

    StopDrawing()
    ImageGadget(0, 0, 0, 400, 200, ImageID(0))
  EndIf
```

```
22
23      Repeat
24        Event = WaitWindowEvent()
25      Until Event = #PB_Event_CloseWindow
26    EndIf
```



Voir aussi

DrawingMode() , CustomGradient()

OS Supportés

Tous

77.37 GradientColor

Syntaxe

```
GradientColor(Position.f, Couleur)
```

Description

Ajoute une couleur au dégradé de couleur.

Arguments

Position.f La position de la couleur dans le dégradé.
C'est une valeur de type float comprise entre 0.0 et 1.0.

Couleur La couleur à ajouter.
Les fonctions RGB() ou RGBA() peuvent être utilisées pour définir facilement une couleur.

Valeur de retour

Aucune.

Remarques

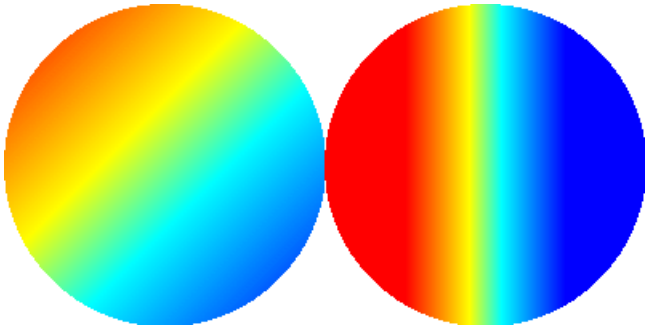
Par défaut, le dégradé de couleur va de la couleur de fond à la position 0.0 à la couleur d'avant plan à la position 1.0.
Avec cette commande, des couleurs intermédiaires peuvent être ajoutées, et les couleurs à 0.0 et 1.0 remplacées.
La commande ResetGradientColors() permet de revenir au dégradé par défaut, si nécessaire.
Les commandes suivantes peuvent être utilisées pour indiquer la forme du dégradé :
- LinearGradient()
- CircularGradient()
- EllipticalGradient()
- BoxedGradient()
- ConicalGradient()
- CustomGradient()

Note : Cette commande a un effet uniquement sur des surface de type ImageOutput() et CanvasOutput() . Le dégradé est dessiné uniquement si le mode #PB_2DDrawing_Gradient est activé avec DrawingMode() .

Exemple

```
1   If OpenWindow(0, 0, 0, 400, 200, "GradientColor", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2    If CreateImage(0, 400, 200) And StartDrawing(ImageOutput(0))
3       Box(0, 0, 400, 200, $FFFFFF)
4
5       DrawingMode(#PB_2DDrawing_Gradient)
6       BackColor($0000FF)
7       GradientColor(0.4, $00FFFF)
8       GradientColor(0.6, $FFFF00)
9       FrontColor($FF0000)
10
11      LinearGradient(0, 0, 200, 200)
12      Circle(100, 100, 100)
13      LinearGradient(350, 100, 250, 100)
14      Circle(300, 100, 100)
15
16      StopDrawing()
17      ImageGadget(0, 0, 0, 400, 200, ImageID(0))
18   EndIf
19
20   Repeat
```

```
21        Event =
      WaitWindowEvent()
22     Until Event =
      #PB_Event_CloseWindow
23    EndIf
```



Voir aussi

ResetGradientColors() , LinearGradient() , CircularGradient() , EllipticalGradient() , BoxedGradient() , ConicalGradient() , CustomGradient() , DrawingMode()

OS Supportés

Tous

77.38 ResetGradientColors

Syntaxe

```
ResetGradientColors()
```

Description

Retire toutes les couleurs personnalisées du dégradé et revient au dégradé par défaut qui part de couleur de fond jusqu'à la couleur d'avant plan .

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

La commande GradientColor() peut être utilisée pour ajouter des couleurs au dégradé.
Note : Cette commande a un effet uniquement sur des surfaces de type ImageOutput() et CanvasOutput() . Le

dégradé est dessiné uniquement si le mode #PB_2DDrawing_Gradient est activé avec DrawingMode() .

Exemple

```
If OpenWindow(0, 0, 0, 600, 200, "LinearGradient Exemple", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  If CreateImage(0, 600, 200) And StartDrawing(ImageOutput(0))
    Box(0, 0, 600, 200, $FFFFFF)

    DrawingMode(#PB_2DDrawing_Gradient)
    BackColor($00FFFF)
    FrontColor($FF0000)

    LinearGradient(0, 0, 200, 200)
    LineXY(0, 0, 200, 200)
    Circle(100, 100, 100)

    GradientColor(0.3, $0000FF)
    LinearGradient(200, 0, 400, 200)
    LineXY(200, 0, 400, 200)
    Circle(300, 100, 100)

    ResetGradientColors()
    LinearGradient(400, 0, 600, 200)
    LineXY(400, 0, 600, 200)
    Circle(500, 100, 100)
    StopDrawing()
    ImageGadget(0, 0, 0, 400, 200, ImageID(0))
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

GradientColor() , LinearGradient() , CircularGradient() , EllipticalGradient() , BoxedGradient() , ConicalGradient() , CustomGradient() , DrawingMode()

OS Supportés

Tous

77.39 LinearGradient

Syntaxe

```
LinearGradient(x1, y1, x2, y2)
```

Description

Crée un dégradé linéaire entre deux points.

Arguments

x1, y1 La position à laquelle est appliquée la couleur d'arrière plan .

x2, y2 La position à laquelle est appliquée la couleur d'avant plan .

Valeur de retour

Aucune.

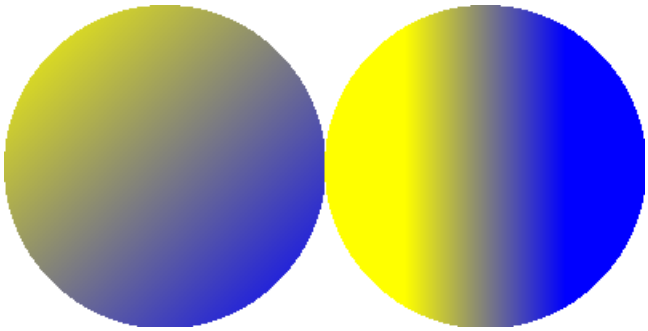
Remarques

il est possible d'ajouter des couleurs au dégradé avec GradientColor() .
Note : Cette commande a un effet uniquement sur des surfaces de type ImageOutput() et CanvasOutput() . Le dégradé est dessiné uniquement si le mode #PB_2DDrawing_Gradient est activé avec DrawingMode() .

Exemple

```
1   If OpenWindow(0, 0, 0, 400, 200, "LinearGradient", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2    If CreateImage(0, 400, 200) And StartDrawing(ImageOutput(0))
3       Box(0, 0, 400, 200, $FFFFFF)
4
5       DrawingMode(#PB_2DDrawing_Gradient)
6       BackColor($00FFFF)
7       FrontColor($FF0000)
8
9       LinearGradient(0, 0, 200, 200)
10      Circle(100, 100, 100)
11      LinearGradient(350, 100, 250, 100)
12      Circle(300, 100, 100)
```

```
13
14          StopDrawing()
15          ImageGadget(0, 0, 0,
        400, 200, ImageID(0))
16       EndIf
17
18       Repeat
19          Event =
        WaitWindowEvent()
20       Until Event =
        #PB_Event_CloseWindow
21    EndIf
```



Voir aussi

GradientColor() , ResetGradientColors() , CircularGradient() , EllipticalGradient() , BoxedGradient() , ConicalGradient() , CustomGradient() , DrawingMode()

OS Supportés

Tous

77.40 CircularGradient

Syntaxe

```
CircularGradient(X, Y, Rayon)
```

Description

Crée un dégradé circulaire autour d'un point.

Arguments

X, Y Position du point autour duquel est appliqué la couleur d'arrière plan .

Rayon Distance autour de 'X, Y' à laquelle est appliqué la couleur d'avant plan .

Valeur de retour

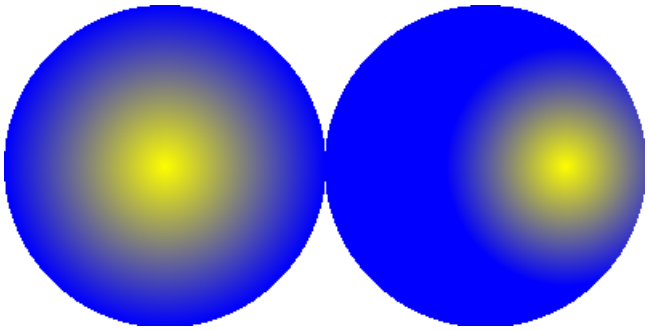
Aucune.

Remarques

Des couleurs additionnelles peuvent être ajoutées au dégradé avec GradientColor() .
Note : Cette commande a un effet uniquement sur des surfaces de type ImageOutput() et CanvasOutput() . Le dégradé est dessiné uniquement si le mode #PB_2DDrawing_Gradient est activé avec DrawingMode() .

Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "CircularGradient", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  If CreateImage(0, 400, 200) And StartDrawing(ImageOutput(0))
    Box(0, 0, 400, 200, $FFFFFF)

    DrawingMode(#PB_2DDrawing_Gradient)
    BackColor($00FFFF)
    FrontColor($FF0000)

    CircularGradient(100, 100, 100)
    Circle(100, 100, 100)
    CircularGradient(350, 100, 75)
    Circle(300, 100, 100)

    StopDrawing()
    ImageGadget(0, 0, 0, 400, 200, ImageID(0))
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```



Voir aussi

GradientColor() , ResetGradientColors() , LinearGradient() , EllipticalGradient() , BoxedGradient() , ConicalGradient() , CustomGradient() , DrawingMode()

OS Supportés

Tous

77.41 EllipticalGradient

Syntaxe

```
EllipticalGradient(X, Y,
    RayonX, RayonY)
```

Description

Crée un dégradé de forme elliptique autour d'un point.

Arguments

X, Y Position du point autour duquel est appliqué la couleur d'arrière plan .

RayonX, RayonY Distances en X et en Y autour de 'X, Y' auxquelles est appliqué la couleur d'avant plan .

Valeur de retour

Aucune.

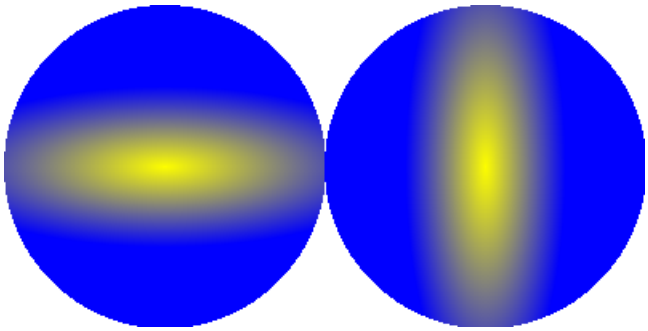
Remarques

Des couleurs additionnelles peuvent être ajoutées au dégradé avec GradientColor() .
Note : Cette commande a un effet uniquement sur des surfaces de type ImageOutput() et CanvasOutput() . Le dégradé est dessiné uniquement si le mode #PB_2DDrawing_Gradient est activé avec DrawingMode() .

Exemple

```
1  If OpenWindow(0, 0, 0, 400,
     200, "EllipticalGradient",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2    If CreateImage(0, 400,
     200) And
     StartDrawing(ImageOutput(0))
3       Box(0, 0, 400, 200,
     $FFFFFF)
4
```

```
5         DrawingMode(#PB_2DDrawing_Gradient)
6         BackColor($00FFFF)
7         FrontColor($FF0000)
8
9         EllipticalGradient(100,
      100, 150, 50)
10        Circle(100, 100, 100)
11        EllipticalGradient(300,
      100, 50, 150)
12        Circle(300, 100, 100)
13
14        StopDrawing()
15        ImageGadget(0, 0, 0,
      400, 200, ImageID(0))
16      EndIf
17
18      Repeat
19        Event =
      WaitWindowEvent()
20      Until Event =
      #PB_Event_CloseWindow
21    EndIf
```



Voir aussi

GradientColor() , ResetGradientColors() , LinearGradient() , CircularGradient() , BoxedGradient() , ConicalGradient() , CustomGradient() , DrawingMode()

OS Supportés

Tous

77.42 BoxedGradient

Syntaxe

```
BoxedGradient(X, Y, Largeur,
    Hauteur)
```

Description

Crée un dégradé de forme rectangulaire.

Arguments

X, Y, Largeur, Hauteur Position et taille du rectangle.
Le dégradé de couleur va de la couleur d'arrière plan au centre du rectangle à la couleur d'avant plan à son contour.

Valeur de retour

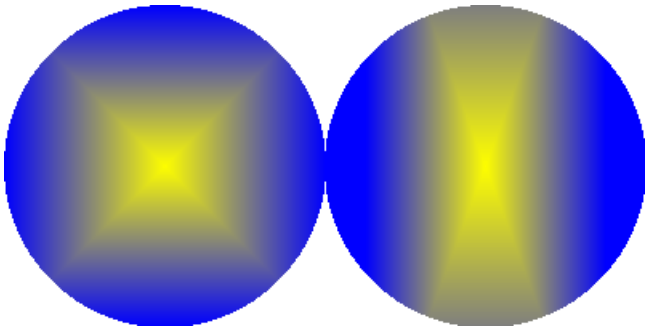
Aucune.

Remarques

Des couleurs additionnelles peuvent être ajoutées au dégradé avec GradientColor() .
Note : Cette commande a un effet uniquement sur des surfaces de type ImageOutput() et CanvasOutput() . Le dégradé est dessiné uniquement si le mode #PB_2DDrawing_Gradient est activé avec DrawingMode() .

Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "BoxedGradient", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
 If CreateImage(0, 400, 200) And StartDrawing(ImageOutput(0))
    Box(0, 0, 400, 200, $FFFFFF)

    DrawingMode(#PB_2DDrawing_Gradient)
    BackColor($00FFFF)
    FrontColor($FF0000)

    BoxedGradient(0, 0, 200, 200)
    Circle(100, 100, 100)
    BoxedGradient(225, -100, 150, 400)
    Circle(300, 100, 100)

    StopDrawing()
    ImageGadget(0, 0, 0, 400, 200, ImageID(0))
 EndIf

 Repeat
    Event = WaitWindowEvent()
 Until Event = #PB_Event_CloseWindow
EndIf
```



Voir aussi

GradientColor() , ResetGradientColors() , LinearGradient() , CircularGradient() , EllipticalGradient() , ConicalGradient() , CustomGradient() , DrawingMode()

OS Supportés

Tous

77.43 ConicalGradient

Syntaxe

```
ConicalGradient(X, Y, Angle.f)
```

Description

Crée un dégradé de forme conique autour d'un point.

Arguments

X, Y Position du centre du cône.

Angle.f L'angle de départ du dégradé en degré.
Le dégradé de couleur va de la couleur d'arrière plan à partir de l'angle spécifié, puis change jusqu'à la couleur d'avant plan dans le sens des aiguilles d'une montre en revenant à l'angle spécifié.

Valeur de retour

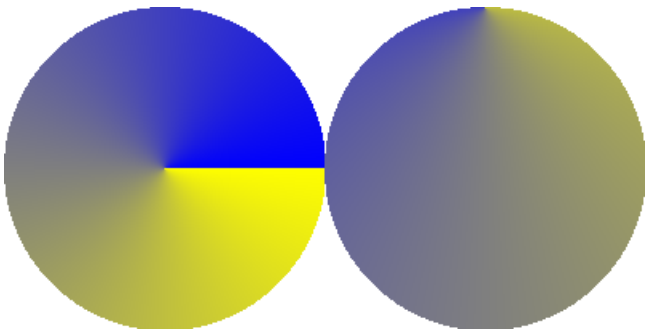
Aucune.

Remarques

Des couleurs additionnelles peuvent être ajoutées au dégradé avec GradientColor() .
Note : Cette commande a un effet uniquement sur des surfaces de type ImageOutput() et CanvasOutput() . Le dégradé est dessiné uniquement si le mode #PB_2DDrawing_Gradient est activé avec DrawingMode() .

Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "ConicalGradient", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
 If CreateImage(0, 400, 200) And StartDrawing(ImageOutput(0))
    Box(0, 0, 400, 200, $FFFFFF)

    DrawingMode(#PB_2DDrawing_Gradient)
    BackColor($00FFFF)
    FrontColor($FF0000)

    ConicalGradient(100, 100, 0.0)
    Circle(100, 100, 100)
    ConicalGradient(300, 0, 90.0)
    Circle(300, 100, 100)

    StopDrawing()
    ImageGadget(0, 0, 0, 400, 200, ImageID(0))
 EndIf

 Repeat
    Event = WaitWindowEvent()
 Until Event = #PB_Event_CloseWindow
EndIf
```



Voir aussi

GradientColor() , ResetGradientColors() , LinearGradient() , CircularGradient() , EllipticalGradient() , BoxedGradient() , CustomGradient() , DrawingMode()

OS Supportés

Tous

77.44 CustomGradient

Syntaxe

```
CustomGradient(@GradientCallback())
```

Description

Crée un dégradé avec une forme personnalisée.

Arguments

@GradientCallback() L'adresse de la procédure de callback.
La procédure doit avoir la forme suivante :

```
1  Procedure.f GradientCallback(X, Y)
2    ;
3    ; Renvoie une valeur entre 0.0 et 1.0 pour définir le dégradé à la position x,y
4    ;
5    ProcedureReturn 1.0
6  EndProcedure
```

La procédure sera appelée pour chaque pixel affiché par l'opération de dessin.
La procédure doit renvoyer une valeur comprise entre 0.0 and 1.0 (pas une valeur de couleur) pour définir la valeur du dégradé à une position donnée.
Les coordonnées X et Y sont toujours orientées par rapport au coin supérieur gauche de la surface de dessin.
Les coordonnées ne sont pas affectées par les appels à SetOrigin() ou à ClipOutput() .

Valeur de retour

Aucune.

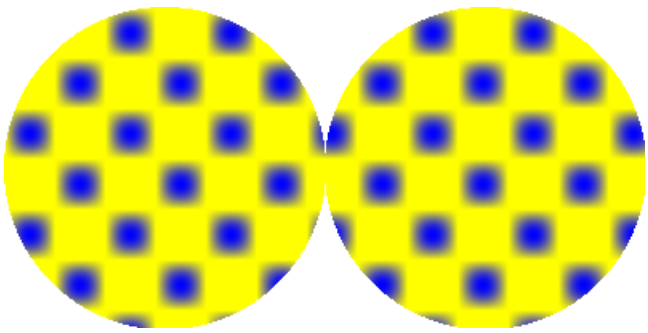
Remarques

Par défaut, la valeur 0.0 représente la couleur d'arrière plan et la valeur 1.0 représente la couleur d'avant plan .
Des couleurs additionnelles peuvent être ajoutées au dégradé avec GradientColor() .
Cette procédure sera appelée de nombreuse fois (pour chaque pixel à appeler), donc elle devra être la plus courte et la plus optimisée possible, sinon l'impact sur les performances de dessin sera conséquent.
Note : Cette commande a un effet uniquement sur des surfaces de type

ImageOutput() et CanvasOutput() . Le dégradé est dessiné uniquement si le mode #PB_2DDrawing_Gradient est activé avec DrawingMode() .

Exemple

```
Procedure.f GradientCallback(X, Y)
  ProcedureReturn Sin(x * 0.1) * Sin(y * 0.1) ; Valeur comprise entre 0 et 1
EndProcedure

If OpenWindow(0, 0, 0, 400, 200, "CustomGradient", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  If CreateImage(0, 400, 200) And StartDrawing(ImageOutput(0))
    Box(0, 0, 400, 200, $FFFFFF)

    DrawingMode(#PB_2DDrawing_Gradient)
    BackColor($00FFFF)
    FrontColor($FF0000)

    CustomGradient(@GradientCallback())
    Circle(100, 100, 100)
    Circle(300, 100, 100)

    StopDrawing()
    ImageGadget(0, 0, 0, 400, 200, ImageID(0))
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```



Voir aussi

GradientColor() , ResetGradientColors() , LinearGradient() , CircularGradient() , EllipticalGradient() , BoxedGradient() , ConicalGradient() , DrawingMode()

OS Supportés

Tous

77.45 SetOrigin

Syntaxe

```
SetOrigin(X, Y)
```

Description

Définit un décalage (offset) d'affichage que subira tout dessin.
Ceci définit l'emplacement des coordonnées (0,0) dans la surface de sortie pour toutes les commandes de dessin.
Par défaut, l'origine se trouve dans le coin supérieur gauche de la sortie.

Arguments

X, Y La nouvelle position de l'origine du dessin.
Il s'agit d'une position absolue et n'est pas affectée par un appel précédent à cette fonction.

Valeur de retour

Aucune.

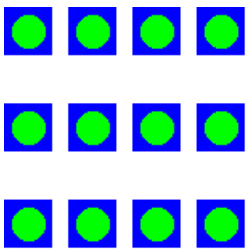
Remarques

Cette commande affecte l'emplacement du dessin pour l'utilisation de toutes les futures fonctions de dessin à l'exception de la fonction ClipOutput() et de la fonction SetOrigin() elle-même. En outre, les coordonnées reçues dans CustomGradient() ou dans CustomFilterCallback() sont toujours absolues, indépendamment de tout appel à cette fonction.

Exemple

```
1  If OpenWindow(0, 0, 0, 200,
     200, "SetOrigin",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
```

```
2     If CreateImage(0, 200,
      200, 24, $FFFFFF) And
      StartDrawing(ImageOutput(0))
3
4        ; Dessine la même
      figure à différents
      endroits en déplaçant
      l'origine dessin
5        For x = 0 To 120 Step 40
6          For y = 0 To 120 Step
      60
7            SetOrigin(X, Y)
8            Box(0, 0, 30, 30,
      $FF0000)
9            Circle(15, 15, 10,
      $00FF00)
10         Next y
11       Next x
12
13       StopDrawing()
14       ImageGadget(0, 0, 0,
      200, 200, ImageID(0))
15    EndIf
16
17    Repeat
18      Event =
      WaitWindowEvent()
19    Until Event =
      #PB_Event_CloseWindow
20  EndIf
```



Voir aussi

GetOriginX() , GetOriginY() , ClipOutput()

OS Supportés

Tous

77.46 GetOriginX

Syntaxe

```
Resultat = GetOriginX()
```

Description

Renvoie la coordonnée X de l'origine du dessin qui a été définie à l'aide de SetOrigin() .

Arguments

Aucun.

Valeur de retour

Renvoie la coordonnée X de l'origine du dessin.

Exemple

```
If OpenWindow(0, 0, 0, 200, 200, "GetOriginX", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
 If CreateImage(0, 200, 200, 24, $FFFFFF) And StartDrawing(ImageOutput(0))

   ; Dessine la même figure à différents endroits en déplaçant l'origine dessin
   For x = 0 To 120 Step 40
     For y = 0 To 120 Step 60
       SetOrigin(X, Y)
       Debug GetOriginX()
       Debug GetOriginY()
       Box(0, 0, 30, 30, $FF0000)
       Circle(15, 15, 10, $00FF00)
     Next y
   Next x

   StopDrawing()
   ImageGadget(0, 0, 0, 200, 200, ImageID(0))
 EndIf

 Repeat
   Event = WaitWindowEvent()
 Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

GetOriginY() , SetOrigin()

OS Supportés

Tous

77.47 GetOriginY

Syntaxe

```
Resultat = GetOriginY()
```

Description

Renvoie la coordonnée Y de l'origine du dessin qui a été définie à l'aide de SetOrigin() .

Arguments

Aucun.

Valeur de retour

Renvoie la coordonnée Y de l'origine du dessin.

Exemple

```
If OpenWindow(0, 0, 0, 200, 200, "GetOriginY", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
 If CreateImage(0, 200, 200, 24, $FFFFFF) And StartDrawing(ImageOutput(0))

   ; Dessine la même figure à différents endroits en déplaçant l'origine dessin
   For x = 0 To 120 Step 40
     For y = 0 To 120 Step 60
       SetOrigin(X, Y)
       Debug GetOriginX()
       Debug GetOriginY()
       Box(0, 0, 30, 30, $FF0000)
       Circle(15, 15, 10, $00FF00)
     Next y
   Next x

   StopDrawing()
   ImageGadget(0, 0, 0, 200, 200, ImageID(0))
 EndIf

 Repeat
```

```
20        Event =
      WaitWindowEvent()
21      Until Event =
      #PB_Event_CloseWindow
22    EndIf
```

Voir aussi

GetOriginX() , SetOrigin()

OS Supportés

Tous

77.48 ClipOutput

Syntaxe

```
ClipOutput(X, Y, Largeur,
    Hauteur)
```

Description

Définit un cadre d'affichage qui limite tous les dessins à ce cadre.
Tous les pixels établis en dehors de cette zone ne seront pas affichés.

Arguments

X, Y, Largeur, Hauteur La position et les dimensions de la zone de découpage. Les coordonnées (X, Y) sont toujours absolues et ne sont pas affectées par les appels à SetOrigin() .

Valeur de retour

Aucune.

Remarques

Cette commande n'a d'effet que sur les sorties de dessin créées par ImageOutput() ou CanvasOutput() .
L'origine du dessin n'est pas modifiée par un appel à cette fonction. Pour dessiner dans le coin supérieur gauche de la boîte de découpage, un appel séparé à SetOrigin() doit être fait si cela est souhaité.
Les fonctions Plot() et Point() n'effectuent pas de vérification des limites pour des raisons de performance et ne sont donc pas concernées par cette commande.

Exemple

```
If OpenWindow(0, 0, 0, 200, 200, "ClipOutput", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  If CreateImage(0, 200, 200, 24, $FFFFFF) And StartDrawing(ImageOutput(0))

    ClipOutput(50, 50, 100, 100) ; restriction de l'affichage du dessin à cette région
    Circle( 50,  50, 50, $0000FF)
    Circle( 50, 150, 50, $00FF00)
    Circle(150,  50, 50, $FF0000)
    Circle(150, 150, 50, $00FFFF)

    DrawingMode(#PB_2DDrawing_Outlined)
    Box(50, 50, 100, 100, $000000)

    StopDrawing()
    ImageGadget(0, 0, 0, 200, 200, ImageID(0))
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

UnclipOutput() , SetOrigin() , OutputWidth() , OutputHeight()

OS Supportés

Tous

77.49 UnclipOutput

Syntaxe

```
UnclipOutput()
```

Description

Retire tout écrêtage (clipping) imposé par la commande ClipOutput() .
Les commandes de dessin qui suivront, seront en mesure d'utiliser toute la surface de dessin à nouveau.

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Cette commande n'a d'effet que sur les dessins créés par ImageOutput() ou CanvasOutput() .

Exemple

```
1   If OpenWindow(0, 0, 0, 200, 200, "ClipOutput", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2    If CreateImage(0, 200, 200, 24, $FFFFFF) And StartDrawing(ImageOutput(0))
3       ClipOutput(50, 50, 100, 100) ; Avec restriction de l'affichage du dessin à cette région
4       Circle( 50,  50, 50, $0000FF)
5       Circle( 50, 150, 50, $00FF00)
6       Circle(150,  50, 50, $FF0000)
7       Circle(150, 150, 50, $00FFFF)
8       DrawText(55,90, "Cliquer ici !")
9       DrawingMode(#PB_2DDrawing_Outlined)
10      Box(50, 50, 100, 100, $000000)
11      StopDrawing()
12      ImageGadget(0, 0, 0, 200, 200, ImageID(0))
13   EndIf
14
```

```
15      Repeat
16        Event = WaitWindowEvent()
17        If Event = #PB_Event_Gadget
18          StartDrawing(ImageOutput(0))
19          UnclipOutput() ; Sans restriction de l'affichage du dessin à cette région
20          Circle( 50,  50, 50, $0000FF)
21          Circle( 50, 150, 50, $00FF00)
22          Circle(150,  50, 50, $FF0000)
23          Circle(150, 150, 50, $00FFFF)
24          DrawingMode(#PB_2DDrawing_Outlined)
25          Box(50, 50, 100, 100, $000000)
26          StopDrawing()
27          SetGadgetState(0,ImageID(0))
28        EndIf
29
30      Until Event = #PB_Event_CloseWindow
31    EndIf
```

Voir aussi

ClipOutput() , SetOrigin() , OutputWidth() , OutputHeight()

OS Supportés

Tous

Chapitre 78

Array

Généralités

Un tableau permet de stocker des éléments de manière indexée. Contrairement à une liste ou à une map , les éléments sont alloués de façon contiguë en mémoire. C'est pourquoi il n'est pas possible d'insérer ou de supprimer un élément. D'un autre côté, il permet d'accéder à n'importe quel élément quasiment instantanément.
Pour manipuler un tableau, il doit d'abord être déclaré avec le mot clé Dim et peuvent être redimensionnés avec ReDim .
Les tableaux peuvent être triés avec SortArray() ou SortStructuredArray() Il est également possible de réordonner les éléments d'un tableau dans un ordre aléatoire avec la fonction RandomizeArray() .
Note : Ces tableaux sont dit dynamiques car ils peuvent changer de taille. Ils existent toutefois des tableaux dit statiques, non redimensionnables et utilisés seulement dans les structures. Ces tableaux s'écrivent avec des crochets. Par exemple : TabStatic[2]. Voir ici . Les fonctions de cette bibliothèque Array ne peuvent pas être utilisées avec ce type de tableaux.

; ; ; ; ; Tri & divers

SortArray()
SortStructuredArray()
RandomizeArray()

OS Supportés

Tous

78.1 ArraySize

Syntaxe

```
Resultat =
    ArraySize(Tableau() [,
    Dimension])
```

Description

Renvoie la taille d'un tableau.

Arguments

Tableau() Le tableau à tester.

Dimension (optionnel) Pour les tableaux multidimensionnels, permet de renvoyer la taille d'une dimension donnée. La première dimension commence à 1.

Valeur de retour

Renvoie la taille de la dimension du tableau telle qu'elle a été spécifiée lors de sa déclaration avec Dim ou ReDim .
Si le tableau n'est pas encore déclaré ou si son allocation a échoué, il renverra -1.

Remarques

Attention, le nombre d'éléments est égale à la taille plus 1.
Par exemple : Dim a(2) contient 3 éléments de a(0) à a(2) pour une taille de 2.
Ne fonctionne pas avec les tableaux statiques déclarés dans les Structures .
Utilisez SizeOf à la place.

Exemple

```
1  Dim Tableau.l(10)
2  Debug ArraySize(Tableau())
     ; affichera '10'
3
4  Dim
     Tableau3Dimensions.l(10,
     20, 30)
5  Debug
     ArraySize(Tableau3Dimensions(),
     2) ; affichera '20'
6
7  Dim MultiArray2.l(2,2,2)
8    For n = 0 To
     ArraySize(MultiArray2(),2)
9  MultiArray2(0,n,0) = n+1
10 Next n
11 Debug MultiArray2(0,0,0) ;
     affichera '1'
12 Debug MultiArray2(0,1,0) ;
     affichera '2'
13 Debug MultiArray2(0,2,0) ;
     affichera '3' (Nous avons
     bien 3 éléments)
```

```
14   Debug
      ArraySize(MultiArray2(),2)
      ; affichera '2' (Et
      pourtant la taille est de
      2)
```

Exemple

```
1    Dim
      Test.q(99999999999999999)
2
3    If ArraySize(Test()) <> -1
4      Test(12345) = 123  ; tout
      se passe bien
5    Else
6      Debug "Le tableau
      'Test()' ne peut pas être
      initialisé."
7    EndIf
```

Exemple

```
1  Structure MaStructure
2    TabStatic.l[3]         ;
      Tableau statique,
      uniquement dans les
      structures
3                           ;
      (Standard C) avec 3
      valeurs de Ex\TabStatic[0]
      à Ex\TabStatic[2], non
      redimensionnable
4    Array TabDynamic.l(4) ;
      Tableau dynamique avec 5
      valeurs de TabDynamic.i(0)
      à TabDynamic.i(4),
      redimensionnable
5  EndStructure
6
7  Debug
      SizeOf(MaStructure\TabStatic)
      ; Affiche 12
8  Debug
      SizeOf(MaStructure\TabDynamic)
      ; Affiche 8
9
10 Ex.MaStructure
11 Debug
      ArraySize(Ex\TabDynamic())
      ; Affiche 4
```

Voir aussi

ListSize() , MapSize()

OS Supportés

Tous

78.2 CopyArray

Syntaxe

```
Resultat =
    CopyArray(TableauSource(),
    TableauDestination())
```

Description

Copie un tableau.

Arguments

TableauSource() Le tableau à copier.

TableauDestination() La copie du tableau source.
Tous les éléments qui se trouvaient auparavant dans ce tableau seront supprimés.
Ce tableau doit être du même type (natif ou structuré) et avoir le même nombre de dimensions que TableauSource().

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Exemple

```
1   Dim Nombres(5)
2   Dim NombresCopie(10)
3
4   Nombres(0) = 128
5   Nombres(5) = 256
6
7   Debug "Taille du tableau
      avant copie:
      "+Str(ArraySize(NombresCopie()))
      ; affichera 10
8
9   CopyArray(Nombres(),
      NombresCopie())
10
11  Debug "Taille du tableau
      après copie:
      "+Str(ArraySize(NombresCopie()))
      ; affichera 5
12  Debug NombresCopie(0)
13  Debug NombresCopie(5)
```

Voir aussi

CopyList() , CopyMap()

OS Supportés

Tous

78.3 FreeArray

Syntaxe

```
FreeArray(Tableau())
```

Description

Détruit un tableau et libère toutes les ressources associées.

Arguments

Tableau() Le tableau à libérer.

Valeur de retour

Aucune.

Remarques

Pour continuer à l'utiliser, Dim doit être appelé.

Voir aussi

FreeList() , FreeMap()

OS Supportés

Tous

Chapitre 79

AudioCD

Généralités

Les CDs Audio sont un bon moyen d'écouter de la musique en qualité Hi-Fi pendant que l'on joue ou travaille sur son ordinateur, sans que cela ne pénalise trop les performances de la machine. Cette bibliothèque propose toutes les fonctions nécessaires pour gérer facilement la lecture de CDs Audio dans un programme PureBasic.

OS Supportés

Tous

79.1 AudioCDLength

Syntaxe

```
Resultat = AudioCDLength()
```

Description

Renvoie la durée totale d'un CD-Audio en secondes.

Arguments

Aucun.

Valeur de retour

Renvoie la durée du CD, en secondes.

Voir aussi

AudioCDTrackLength() , UseAudioCD()

OS Supportés

Windows, MacOS X

79.2 AudioCDName

Syntaxe

```
Resultat\$ = AudioCDName()
```

Description

Renvoie le nom du lecteur de CD.

Arguments

Aucun.

Valeur de retour

Renvoie une chaîne de caractères contenant le nom du lecteur de CD (dépendant de l'O.S.). Par exemple, le nom retourné peut être 'D :\' dans un système Windows.

Voir aussi

UseAudioCD()

OS Supportés

Tous

79.3 AudioCDTrackLength

Syntaxe

```
Resultat =
    AudioCDTrackLength(NumeroPiste)
```

Description

Renvoie la longueur d'une piste audio.

Arguments

NumeroPiste Le numéro de la piste demandée.
La première piste est indexée à 1.

Valeur de retour

Renvoie la durée en secondes.

Voir aussi

AudioCDTracks() , UseAudioCD()

OS Supportés

Tous

79.4 AudioCDStatus

Syntaxe

```
Resultat = AudioCDStatus()
```

Description

Renvoie l'état actuel du lecteur de CD.

Arguments

Aucun.

Valeur de retour

Peut prendre les valeurs suivantes :

```
-1 : Lecteur de CD non prêt
 (vide ou avec tiroir
 ouvert)
 0 : Lecteur de CD arrêté
 (mais un CD est à
 l'intérieur et détecté)
>0 : Plage audio en cours
 de lecture.
```

Voir aussi

UseAudioCD()

OS Supportés

Tous

79.5 AudioCDTracks

Syntaxe

```
Resultat = AudioCDTracks()
```

Description

Renvoie le nombre total de pistes d'un CD.

Arguments

Aucun.

Valeur de retour

Renvoie le nombre de pistes du CD.

Voir aussi

UseAudioCD()

OS Supportés

Tous

79.6 AudioCDTrackSeconds

Syntaxe

```
Resultat =
    AudioCDTrackSeconds()
```

Description

Renvoie le nombre de secondes écoulées en lecture depuis le début de la piste.

Arguments

Aucun.

Valeur de retour

Renvoie le nombre de secondes écoulées depuis le début de la piste.

Voir aussi

AudioCDTrackLength() , AudioCDTracks() , UseAudioCD()

OS Supportés

Tous

79.7 EjectAudioCD

Syntaxe

```
EjectAudioCD(Etat)
```

Description

Ouvre ou ferme le tiroir du lecteur de CD.

Arguments

```
Etat    =1 : Le tiroir est
    ouvert.
   <>1 : Le tiroir est fermé.
```

Valeur de retour

Aucune.

Voir aussi

UseAudioCD()

OS Supportés

Windows, Linux

79.8 InitAudioCD

Syntaxe

```
Resultat = InitAudioCD()
```

Description

Tente d'initialiser les ressources nécessaires pour exécuter les fonctions CD audios.

Arguments

Aucun.

Valeur de retour

Renvoie le nombre de lecteurs de CD reconnus par le système en cas de succès ou zéro sinon.

Remarques

Cette fonction doit être appelée avant toute autre fonction de la bibliothèque CD-Audio.

Voir aussi

UseAudioCD()

OS Supportés

Tous

79.9 PauseAudioCD

Syntaxe

```
PauseAudioCD()
```

Description

Met sur pause la lecture d'un CD-Audio.

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

La lecture peut être reprise en appelant la fonction ResumeAudioCD() .

Voir aussi

UseAudioCD() , ResumeAudioCD()

OS Supportés

Tous

79.10 PlayAudioCD

Syntaxe

```
PlayAudioCD(PisteDebut,
    PisteFin)
```

Description

Commence la lecture du CD-Audio.

Arguments

PisteDebut La première piste lue porte le numéro 1.

PisteFin La dernière piste à jouer.

Valeur de retour

Aucune.

Voir aussi

AudioCDTracks() , UseAudioCD()

OS Supportés

Tous

79.11 ResumeAudioCD

Syntaxe

```
ResumeAudioCD()
```

Description

Reprend la lecture du CD après un pause demandée par la fonction PauseAudioCD() .

Arguments

Aucun.

Valeur de retour

Aucune.

Voir aussi

UseAudioCD()

OS Supportés

Tous

79.12 StopAudioCD

Syntaxe

```
StopAudioCD()
```

Description

Arrête la lecture du CD-Audio en cours.

Arguments

Aucun.

Valeur de retour

Aucune.

Voir aussi

UseAudioCD()

OS Supportés

Tous

79.13 UseAudioCD

Syntaxe

```
UseAudioCD(LecteurCD)
```

Description

Sélectionne le lecteur CD auquel les commandes AudioCD s'appliquent.

Arguments

LecteurCD Le lecteur à utiliser.
Le premier lecteur a le numéro 0.

Valeur de retour

Aucune.

Remarques

Il est possible de jouer plusieurs CD en même temps.
Comme la fonction InitAudioCD() nous permet de savoir combien de lecteurs de CD sont reconnus par le système, UseAudioCD() permet de changer le lecteur CD en cours.
La quantité d'unités disponibles dans le système est renvoyée par la fonction InitAudioCD()

Voir aussi

InitAudioCD()

OS Supportés

Tous

Chapitre 80

Billboard

Généralités

Les 'Billboards' sont des surfaces planes rectangulaires (composées de 2 triangles) qui sont toujours orientées face à la caméra . Ils sont très utiles pour afficher un grand nombre d' éléments tels que la pluie, la neige, des arbres, de la végétation etc.
La commande InitEngine3D() doit être appelée avec succès avant de pouvoir utiliser les commandes relatives aux billboards.
Les billboards sont gérés par groupe et chaque groupe contient ses propres billboards ayant une taille ou une position similaire afin d'accélérer la gestion d'un grand nombre d'objets.

OS Supportés

Tous

80.1 AddBillboard

Syntaxe

```
Resultat =
    AddBillboard(#BillboardGroup,
    X, Y, Z)
```

Description

Ajoute un billboard à un groupe de billboards.

Arguments

#BillboardGroup Numéro du groupe de billboards auquel le nouveau billboard sera attaché.

X, Y, Z Coordonnées du nouveau billboard relatives à la position du groupe de billboards.

Valeur de retour

Le nouvel indice du billboard.

Voir aussi

CreateBillboardGroup() , BillboardLocate() , BillboardX() , BillboardY() , BillboardZ() , MoveBillboard() , RemoveBillboard()

OS Supportés

Tous

80.2 BillboardGroupID

Syntaxe

```
Resultat =
    BillboardGroupID(#BillboardGroup)
```

Description

Renvoie l'identifiant unique d'un groupe de billboard.

Arguments

#BillboardGroup Le numéro du groupe de billboards à identifier.

Valeur de retour

Renvoie l'identifiant unique qui identifie #BillboardGroup.

Remarques

Cette fonction est très utile quand une fonction d'une autre bibliothèque nécessite l'identifiant d'un BillboardGroup.

Voir aussi

CreateBillboardGroup() , IsBillboardGroup()

OS Supportés

Tous

80.3 BillboardGroupMaterial

Syntaxe

```
BillboardGroupMaterial(#BillboardGroup,
    MatiereID)
```

Description

Affecte une matière à un groupe de billboards.

Arguments

#BillboardGroup Numéro du groupe de billboards.

MatiereID Numéro de la matière à assigner au groupe de billboards (et par conséquent aux billboards).
Peut être facilement récupérée à l'aide de la commande MaterialID() .
Cette matière sera utilisée par tous les billboards du groupe.
Un groupe de billboards ne peut avoir qu'une seule matière affectée à la fois.

Valeur de retour

Aucune.

Voir aussi

CreateBillboardGroup()

OS Supportés

Tous

80.4 BillboardGroupX

Syntaxe

```
Resultat =
    BillboardGroupX(#BillboardGroup
    [, Mode])
```

Description

Renvoie la position absolue en X du groupe de billboards.

Arguments

#BillboardGroup Numéro du groupe de billboards.

Mode (optionnel) Le mode d'obtention du vecteur de direction de la caméra.
Peut être l'une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 direction de la caméra
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 direction de la caméra
 par rapport à son parent.
```

Valeur de retour

La position absolue en X du groupe de billboards dans la scène 3D.

Voir aussi

CreateBillboardGroup() , BillboardGroupY() , BillboardGroupZ()

OS Supportés

Tous

80.5 BillboardGroupY

Syntaxe

```
Resultat =
    BillboardGroupY(#BillboardGroup
    [, Mode])
```

Description

Renvoie la position absolue en Y du groupe de billboards.

Arguments

#BillboardGroup Numéro du groupe de billboards.

Mode (optionnel) Le mode d'obtention du vecteur de direction de la caméra. Peut être l'une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 direction de la caméra
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 direction de la caméra
 par rapport à son parent.
```

Valeur de retour

La position absolue en Y du groupe de billboards dans la scène 3D.

Voir aussi

CreateBillboardGroup() , BillboardGroupX() , BillboardGroupZ()

OS Supportés

Tous

80.6 BillboardGroupZ

Syntaxe

```
Resultat =
    BillboardGroupZ(#BillboardGroup
    [, Mode])
```

Description

Renvoie la position absolue en Z du groupe de billboards.

Arguments

#BillboardGroup Numéro du groupe de billboards.

Mode (optionnel) Le mode d'obtention du vecteur de direction de la caméra. Peut être l'une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 direction de la caméra
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 direction de la caméra
 par rapport à son parent.
```

Valeur de retour

La position absolue en Z du groupe de billboards dans la scène 3D.

Voir aussi

CreateBillboardGroup() , BillboardGroupX() , BillboardGroupY()

OS Supportés

Tous

80.7 BillboardHeight

Syntaxe

```
Resultat =
    BillboardHeight(#Billboard,
    #BillboardGroup)
```

Description

Renvoie la hauteur d'un billboard au sein d'un groupe de billboards.

Arguments

#Billboard Numéro du billboard.

#BillboardGroup Numéro du groupe de billboards.

Valeur de retour

La hauteur du billboard exprimée dans la même unité que celle utilisée par le monde 3D.

Voir aussi

BillboardWidth() , ResizeBillboard()

OS Supportés

Tous

80.8 BillboardLocate

Syntaxe

```
BillboardLocate(#Billboard,
    #BillboardGroup, X, Y, Z)
```

Description

Déplace un billboard de manière absolue.

Arguments

#Billboard Numéro du billboard à déplacer.

#BillboardGroup Numéro du groupe de billboards qui contient le billboard à déplacer.

X, Y, Z La nouvelle position du billboard.

Valeur de retour

Aucune.

Remarques

La position réelle du billboard dans le monde 3D est relative à la position du groupe de billboards. Pour déplacer un billboard par rapport à sa position actuelle, utiliser la commande MoveBillboard() .

Voir aussi

MoveBillboard() , BillboardX() , BillboardY() , BillboardZ()

OS Supportés

Tous

80.9 BillboardWidth

Syntaxe

```
Resultat =
    BillboardWidth(#Billboard,
    #BillboardGroup)
```

Description

Renvoie la largeur d'un billboard au sein d'un groupe de billboards.

Arguments

#Billboard Numéro du billboard.

#BillboardGroup Numéro du groupe de billboards.

Valeur de retour

La largeur du billboard en unité monde.

Voir aussi

BillboardHeight() , ResizeBillboard()

OS Supportés

Tous

80.10 BillboardX

Syntaxe

```
Resultat =
    BillboardX(#Billboard,
    #BillboardGroup)
```

Description

Renvoie la position en X du billboard au sein d'un groupe de billboards.

Arguments

#Billboard Numéro du billboard.

#BillboardGroup Numéro du groupe de billboards.

Valeur de retour

La position en X du billboard dans le groupe de billboards.

Voir aussi

BillboardY() , BillboardZ() , BillboardLocate() , MoveBillboard()

OS Supportés

Tous

80.11 BillboardY

Syntaxe

```
Resultat =
    BillboardY(#Billboard,
    #BillboardGroup)
```

Description

Renvoie la position en Y du billboard au sein d'un groupe de billboards.

Arguments

#**Billboard** Numéro du billboard.

#**BillboardGroup** Numéro du groupe de billboards.

Valeur de retour

La position en Y du billboard dans le groupe de billboards.

Voir aussi

BillboardX() , BillboardZ() , BillboardLocate() , MoveBillboard()

OS Supportés

Tous

80.12 BillboardZ

Syntaxe

```
Resultat =
    BillboardZ(#Billboard,
    #BillboardGroup)
```

Description

Renvoie la position en Z du billboard au sein d'un groupe de billboards.

Arguments

#**Billboard** Numéro du billboard.

#**BillboardGroup** Numéro du groupe de billboards.

Valeur de retour

La position en Z du billboard dans le groupe de billboards.

Voir aussi

BillboardX() , BillboardY() , BillboardLocate() , MoveBillboard()

OS Supportés

Tous

80.13 ClearBillboards

Syntaxe

```
ClearBillboards(#BillboardGroup)
```

Description

Détruit tous les billboards contenus dans un groupe de billboards.

Arguments

#**BillboardGroup** Numéro du groupe de billboards.

Valeur de retour

Aucune.

Voir aussi

AddBillboard() , CountBillboards()

OS Supportés

Tous

80.14 CountBillboards

Syntaxe

```
Resultat =
    CountBillboards(#BillboardGroup)
```

Description

Renvoie le nombre de billboards contenus dans un groupe de billboards.

Arguments

#**BillboardGroup** Numéro du groupe de billboards.

Valeur de retour

Le nombre de billboards contenus dans le groupe spécifié.

Voir aussi

AddBillboard() , ClearBillboards()

OS Supportés

Tous

80.15 CreateBillboardGroup

Syntaxe

```
Resultat =
    CreateBillboardGroup(#BillboardGroup,
    MatiereID,
    LargeurBillboard,
    HauteurBillboard [, X, Y,
    Z [, MasqueVisibilité [,
    Type]]]
```

Description

Créé un nouveau groupe de billboards vide.

Arguments

#**BillboardGroup** Le numéro du nouveau BillboardGroup.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

MatiereID La matière spécifiée sera affectée au groupe et par conséquent utilisée par tous les billboards du groupe. La valeur 'MatiereID' peut être facilement récupérée à l'aide de la commande MaterialID() .

LargeurBillboard, HauteurBillboard Les dimensions par défaut (en unité monde) des futurs billboards.
Bien que la taille de chaque futur billboard pourra être definie indépendamment à l'aide de la commande ResizeBillboard() , il est important de garder à l'esprit que des problèmes de performances peuvent très rapidement apparaître si tous les billboards d'un même groupe n'ont pas la même taille.

X, Y, Z (optionnel) La position absolue du nouveau groupe de billboards dans le monde 3D.

MasqueVisibilité (optionnel) Un masque pour choisir sur quelle caméra le groupe de billboards doit être affiché. Si ce masque correspond au masque spécifié dans CreateCamera() alors l'entité sera visible par la caméra. Voir CreateEntity() pour construire des masques appropriés. Si ce paramètre est omis ou #PB_All alors le groupe de billboards sera visible par toutes les caméras.

Type (optionnel) Peut prendre l'une des valeurs suivantes :

```
#PB_Billboard_Point:
 Billboards de points
 standards, toujours de
 face et plein cadre et
 toujours debout (par
 défaut).
#PB_Billboard_Oriented:
 Billboards orientés
 autour d'un vecteur de
 direction partagé
 (utilisé comme axe Y) et
 ne tournent autour de
 cet axe que pour faire
 face à la caméra.
#PB_Billboard_SelfOriented:
 Billboards orientés
 autour de leur propre
 vecteur de direction (de
 leur propre axe Y) et ne
 tournent autour de cet
 axe que pour faire face
 à la caméra.
#PB_Billboard_Perpendicular:
 Billboards
 perpendiculaires à un
 vecteur de direction
 partagé (utilisé comme
 axe Z, face à l'écran)
 et les axes X, Y sont
 déterminés par un
 vecteur up commun.
#PB_Billboard_SelfPerpendicular:
 Billboards
 perpendiculaires à leur
 propre vecteur de
 direction (de leur
 propre axe Z, face à
 l'écran) et les axes X,
 Y sont déterminés par
 par un vecteur up commun.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Si le #BillboardGroup est déjà utilisé alors il sera automatiquement détruit (ainsi que tous ses billboards) et remplacé par le nouveau.

Voir aussi

FreeBillboardGroup() , HideBillboardGroup() , IsBillboardGroup() , BillboardGroupID() , AddBillboard() , BillboardGroupCommonDirection() , BillboardGroupCommonDirection()

OS Supportés

Tous

80.16 BillboardGroupCommonDirection

Syntaxe

```
BillboardGroupCommonDirection(#BillboardGroup,
    X, Y, Z)
```

Description

Change la direction commune d'affichage.

Arguments

#BillboardGroup Numéro du groupe de billboards.

X, Y, Z Le vecteur de direction commune (valeurs généralement entre -1,0 et 1,0, sinon elle sera automatiquement normalisée).

Valeur de retour

Aucune.

Voir aussi

CreateBillboardGroup()

OS Supportés

Tous

80.17 BillboardGroupCommonUpVector

Syntaxe

```
BillboardGroupCommonUpVector(#BillboardGroup,
    X, Y, Z)
```

Description

Modifie le vecteur up commun.

Arguments

#BillboardGroup Numéro du groupe de billboards.

X, Y, Z Le vecteur up commun (valeurs généralement entre -1,0 et 1,0, sinon elle sera automatiquement normalisée).

Valeur de retour

Aucune.

Voir aussi

CreateBillboardGroup()

OS Supportés

Tous

80.18 FreeBillboardGroup

Syntaxe

```
FreeBillboardGroup(#BillboardGroup)
```

Description

Détruit un groupe de billboards ainsi que tous les billboards qu'il contenait.

Arguments

#BillboardGroup Numéro du groupe de billboards à détruire.
Si @pb_allest spécifié, tous les autres groupes de billboards sont libérés.

Valeur de retour

Aucune.

Remarques

Un groupe détruit ne doit plus être utilisé par les commandes relatives aux groupes de billboards.
Tous les groupes de billboards restants sont automatiquement supprimés quand le programme se termine.

Voir aussi

CreateBillboardGroup() , HideBillboardGroup() , IsBillboardGroup() , BillboardGroupID() , AddBillboard()

OS Supportés

Tous

80.19 HideBillboardGroup

Syntaxe

```
HideBillboardGroup(#BillboardGroup,
    Etat)
```

Description

Cache ou affiche un groupe de billboards ainsi que tous les billboards qu'il contient.

Arguments

#**BillboardGroup** Numéro du groupe de billboards à afficher ou à cacher.

Etat Le nouvel état du groupe de billboards :

```
#True : Le groupe est
  caché.
#False: Le groupe est
  affiché.
```

Valeur de retour

Aucune.

Voir aussi

CreateBillboardGroup() , FreeBillboardGroup() , IsBillboardGroup() , BillboardGroupID() , AddBillboard()

OS Supportés

Tous

80.20 IsBillboardGroup

Syntaxe

```
Resultat =
    IsBillboardGroup(#BillboardGroup)
```

Description

Teste si un BillboardGroup est correctement initialisé.

Arguments

#BillboardGroup Numéro du groupe de billboards

Valeur de retour

Renvoie une valeur non nulle si l'objet est valide, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.

Voir aussi

CreateBillboardGroup() , FreeBillboardGroup() , HideBillboardGroup() , BillboardGroupID() , AddBillboard()

OS Supportés

Tous

80.21 MoveBillboard

Syntaxe

```
MoveBillboard(#Billboard,
    #BillboardGroup, X, Y, Z)
```

Description

Déplace un billboard à l'intérieur d'un groupe de manière relative à sa position actuelle. Pour effectuer un déplacement absolu, il convient d'utiliser la commande BillboardLocate() .

Arguments

#Billboard Numéro du billboard.

#BillboardGroup Numéro du groupe de billboards.

X, Y, Z Valeur du déplacement en X, Y et Z relatif à la position actuelle du billboard.

Valeur de retour

Aucune.

Voir aussi

BillboardLocate() , BillboardX() , BillboardY() , BillboardZ()

OS Supportés

Tous

80.22 MoveBillboardGroup

Syntaxe

```
MoveBillboardGroup(#BillboardGroup,
    X, Y, Z [, Mode])
```

Description

Déplace un groupe de billboards de manière relative à sa position actuelle.

Arguments

#BillboardGroup Numéro du groupe de billboards à déplacer.

X, Y, Z Nouvelle position du groupe de billboards.

Mode (optionnel) Peut être une des valeurs suivantes :

```
#PB_Relative: Déplacement
 relatif, à partir de la
 position actuelle du
 groupe de billboards
 (par défaut).
#PB_Absolute: Déplacement
 absolu à la position
 spécifiée.
```

combinée avec l'une des valeurs suivantes :

```
#PB_Local : Déplacement
 local.
```

```
#PB_Parent: Déplacement
 par rapport à la
 position du parent.
#PB_World : Déplacement
 par rapport au monde.
```

Valeur de retour

Aucune.

Voir aussi

CreateBillboardGroup() , FreeBillboardGroup() , HideBillboardGroup() , BillboardGroupID() , AddBillboard()

OS Supportés

Tous

80.23 RemoveBillboard

Syntaxe

```
RemoveBillboard(#Billboard,
    #BillboardGroup)
```

Description

Détruit un billboard d'un groupe de billboards.

Arguments

#Billboard Numéro du billboard à détruire.

#BillboardGroup Numéro du groupe de billboards qui contient le billboard à détruire.

Valeur de retour

Aucune.

Voir aussi

AddBillboard() , ClearBillboards() , CountBillboards()

OS Supportés

Tous

80.24 ResizeBillboard

Syntaxe

```
ResizeBillboard(#Billboard,
    #BillboardGroup, Largeur,
    Hauteur)
```

Description

Redimensionne un billboard.

Arguments

#Billboard Numéro du billboard à redimensionner.

#BillboardGroup Numéro du groupe de billboards qui contient le billboard à redimensionner.

Largeur, Hauteur Les nouvelles dimensions du billboard.
Eviter de redimensionner séparément chaque billboard d'un groupe car il peut y avoir de lourdes conséquences au niveau des perfomances si la taille utilisée n'est pas celle par défaut (spécifié par CreateBillboardGroup()).

Valeur de retour

Aucune.

Voir aussi

BillboardHeight() , BillboardWidth() , CreateBillboardGroup()

OS Supportés

Tous

80.25 RotateBillboardGroup

Syntaxe

```
RotateBillboardGroup(#BillboardGroup,
    X, Y, Z [, Mode])
```

Description

Effectue une rotation d'un groupe de billboards.

Arguments

#**BillboardGroup** Numéro du groupe de billboards qui contient le billboard à tourner.

X, Y, Z Angles x,y,z spécifiés en degrés de 0 à 360.

Mode (optionnel)

```
      PB_Absolute: Rotation
    absolue (par défaut).
  PB_Relative: Rotation
    relative basée sur la
    rotation précédente du
    #BillboardGroup.
```

Valeur de retour

Aucune.

Voir aussi

BillboardGroupMaterial() ,
BillboardGroupID()

OS Supportés

Tous

Chapitre 81

CGI

Généralités

CGI signifie ”Common Gateway Interface” (littéralement ”Interface de passerelle commune”) et généralement abrégée en ”CGI”. Elle permet de créer des applications côté serveur. Cette bibliothèque fournit toutes les commandes nécessaires pour recevoir les requêtes ou les fichiers, répondre et faire diverses autres opérations. CGI et le mode FastCGI sont pris en charge tous les deux.
L'article Wikipedia sur CGI fournit un bon point de départ pour les débutants à CGI (en anglais).
L'article Wikipedia sur CGI en français.

OS Supportés

Tous

81.1 CGICookieName

Syntaxe

```
Resultat\$ =
    CGICookieName(Index)
```

Description

Renvoie le nom du cookie spécifié.

Arguments

Index L'indice du cookie.
La première valeur de l'indice commence à 0.

Valeur de retour

Renvoie le nom du cookie spécifié.

Remarques

Pour obtenir le nombre de cookies disponibles, utiliser CountCGICookies() .

Exemple

```
If Not InitCGI() Or Not ReadCGI()
  End
EndIf

WriteCGIHeader(#PB_CGI_HeaderContentType, "text/html") ; Ecrit les en-têtes pour informer le navigateur du format du contenu
WriteCGIHeader(#PB_CGI_HeaderSetCookie, "mycookie=hello", #PB_CGI_LastHeader) ; Ecrit un cookie nommé 'mycookie'

WriteCGIString("<html><title>PureBasic - cookies</title><body>" +
                "NbCookies: " + CountCGICookies() + "<br><br>")

; Liste tous les cookies et affiche leur nom et leur valeur
;
For k = 0 To CountCGICookies()-1
  WriteCGIString(CGICookieName(k)+": " + CGICookieValue(CGICookieName(k)) + "<br>")
Next

WriteCGIString("</body></html>")
```

Voir aussi

CountCGICookies() , CGICookieValue()

OS Supportés

Tous

81.2 CGICookieValue

Syntaxe

```
Resultat$ = CGICookieValue(Nom$)
```

Description

Renvoie la valeur du cookie spécifié.

Arguments

Nom$ Le nom du cookie.
Le nom du cookie est sensible à la casse.

Valeur de retour

Renvoie la valeur du cookie spécifié.

Remarques

La fonction CGICookieName() peut être utilisée pour obtenir le nom d'un cookie spécifié.
Pour obtenir le nombre de cookies disponibles, utiliser CountCGICookies() .

Exemple

```
If Not InitCGI() Or Not ReadCGI()
  End
EndIf

WriteCGIHeader(#PB_CGI_HeaderContentType, "text/html") ; Ecrit les en-têtes pour informer le navigateur du format du contenu
WriteCGIHeader(#PB_CGI_HeaderSetCookie, "mycookie=hello", #PB_CGI_LastHeader) ; Ecrit un cookie nommé 'mycookie'

WriteCGIString("<html><title>PureBasic - cookies</title><body>" +
                "NbCookies: " + CountCGICookies() + "<br><br>")

; Liste tous les cookies et affiche leur nom et leur valeur
;
For k = 0 To CountCGICookies()-1
  WriteCGIString(CGICookieName(k)+": " + CGICookieValue(CGICookieName(k)) + "<br>")
Next

WriteCGIString("</body></html>")
```

Voir aussi

CountCGICookies() , CGICookieName()

OS Supportés

Tous

81.3 CountCGICookies

Syntaxe

```
Resultat = CountCGICookies()
```

Description

Renvoie le nombre de cookies disponibles.

Arguments

Aucun.

Valeur de retour

Renvoie le nombre de cookies disponibles.

Remarques

Les cookies sont des petits fichiers persistants stockés par le navigateur Web pour permettre de se rappeler un contexte et faciliter la navigation future lors du chargement de la même page plus tard.
S'il vous plaît noter que la législation européenne impose désormais d'informer les utilisateurs que les cookies ne sont pas utilisés pour recueillir des informations inutilement.
Le nom et la valeur des cookies peuvent être obtenus avec CGICookieName() et CGICookieValue() .

Exemple

```
1  If Not InitCGI() Or Not
     ReadCGI()
2    End
3  EndIf
4
5  WriteCGIHeader(#PB_CGI_HeaderContentType,
     "text/html") ; Ecrit les
     en-têtes pour informer le
     navigateur du format du
     contenu
6  WriteCGIHeader(#PB_CGI_HeaderSetCookie
     , "mycookie=hello",
     #PB_CGI_LastHeader) ;
     Ecrit un cookie nommé
     'mycookie'
```

```
7
8   WriteCGIString("<html><title>PureBasic
      - cookies</title><body>" +
9                  "NbCookies:
      " + CountCGICookies() +
      "<br><br>")
10
11  WriteCGIString("</body></html>")
```

Voir aussi

CGICookieName() , CGICookieValue()

OS Supportés

Tous

81.4 CountCGIParameters

Syntaxe

```
Resultat =
    CountCGIParameters()
```

Description

Renvoie le nombre de paramètres disponibles après une requête GET ou POST.

Arguments

Aucun.

Valeur de retour

Renvoie le nombre de paramètres disponibles après une requête GET ou POST.

Remarques

Des paramètres peuvent être obtenus avec CGIParameterName() , CGIParameterValue() et CGIParameterType() .

Exemple

```
1   If Not InitCGI() Or Not
      ReadCGI()
2     End
3   EndIf
4
```

```
5   WriteCGIHeader(#PB_CGI_HeaderContentType,
    "text/html",
    #PB_CGI_LastHeader) ;
    Ecrit les en-têtes pour
    informer le navigateur du
    format du contenu
6
7   WriteCGIString("<html><title>PureBasic
    -
    parameters</title><body>" +
8                  "NbParameters:
    " + CountCGIParameters() +
    "<br><br>")
9
10  WriteCGIString("</body></html>")
```

Voir aussi

CGIParameterName() ,
CGIParameterValue() ,
CGIParameterType() ,
CGIParameterData()

OS Supportés

Tous

81.5 CGIParameterName

Syntaxe

```
Resultat =
    CGIParameterName(Index)
```

Description

Renvoie le nom du paramètre spécifié.

Arguments

Index L'indice du paramètre.
La première valeur de l'indice commence à 0.

Valeur de retour

Renvoie le nom du paramètre spécifié.

Remarques

Pour obtenir le nombre de paramètres disponibles, utiliser CountCGIParameters()
.

Exemple

```
If Not InitCGI() Or Not ReadCGI()
  End
EndIf

WriteCGIHeader(#PB_CGI_HeaderContentType, "text/html", #PB_CGI_LastHeader) ; Ecrit les en-têtes pour informer le navigateur du format du contenu

WriteCGIString("<html><title>PureBasic - parametres</title><body>" + "NbParametres: " + CountCGIParameters() + "<br><br>")

; La liste de tous les paramètres et affiche leur nom
;
For k = 0 To CountCGIParameters()-1
  WriteCGIString(CGIParameterName(k)+"<br>")
Next

WriteCGIString("</body></html>")
```

Voir aussi

CountCGIParameters() ,
CGIParameterValue() ,
CGIParameterType()

OS Supportés

Tous

81.6 CGIParameterValue

Syntaxe

```
Resultat =
    CGIParameterValue(Nom$ [,
    Index])
```

Description

Renvoie la valeur du paramètre spécifié.

Arguments

Nom$ Le nom du paramètre.

Le nom du paramètre est sensible à la casse.
Ce paramètre sera ignoré si un "Index" est spécifié.

Index (optionnel) L'index du paramètre à chercher.
La première valeur de l'indice commence à 0.
Si spécifié, la valeur du paramètre 'Nom$' est ignorée (sauf si #PB_Ignore est utilisé).

Valeur de retour

Renvoie la valeur du paramètre spécifié.

Remarques

CGIParameterName() peut être utilisé pour obtenir le nom d'un paramètre spécifié.
Pour obtenir le nombre de paramètres disponibles, utiliser CountCGIParameters() .

Exemple

```
1   If Not InitCGI() Or Not
     ReadCGI()
2     End
3   EndIf
4
5   WriteCGIHeader(#PB_CGI_HeaderContentType,
     "text/html",
     #PB_CGI_LastHeader) ;
     Ecrit les en-têtes pour
     informer le navigateur du
     format du contenu
6
7   WriteCGIString("<html><title>PureBasic
     -
     parametres</title><body>" +
8                  "NbParametres:
     " + CountCGIParameters() +
     "<br><br>")
9
10  ; La liste des tous les
     paramètres et affiche leur
     nom et leur valeur
11  ;
12  For k = 0 To
     CountCGIParameters()-1
13    WriteCGIString(CGIParameterName(k)
     + ": " +
     CGIParameterValue("", k) +
     "<br>")
14  Next
15
16  WriteCGIString("</body></html>")
```

Voir aussi

CountCGIParameters() ,
CGIParameterName() ,
CGIParameterType()

OS Supportés

Tous

81.7 CGIParameterType

Syntaxe

```
Resultat =
    CGIParameterType(Nom$ [,
    Index])
```

Description

Renvoie le type du paramètre spécifié.

Arguments

Nom$ Le nom du paramètre.
Le nom du paramètre est sensible à la casse.

Index (optionnel) La première valeur de l'indice commence à 0.
Si spécifié, la valeur du paramètre 'Nom$' est ignorée (sauf si #PB_Ignore est utilisé).

Valeur de retour

Renvoie le type du paramètre spécifié.
Peut être l'une des valeurs suivantes :

```
  #PB_CGI_Text: le paramètre
    est une chaîne
  #PB_CGI_File: le paramètre
    est un fichier binaire.
                CGIParameterValue()
 renverra le nom du fichier
    original et
                CGIParameterData()
 récupèrera les données
    binaires.
```

Remarques

CGIParameterName() peut être utilisé pour obtenir le nom d'un paramètre spécifié.
Pour obtenir le nombre de paramètres disponibles, utiliser CountCGIParameters()
.

Exemple

```
If Not InitCGI() Or Not ReadCGI()
  End
EndIf

WriteCGIHeader(#PB_CGI_HeaderContentType, "text/html", #PB_CGI_LastHeader) ; Ecrit les en-têtes pour informer le navigateur du format du contenu

WriteCGIString("<html><title>PureBasic - parametres</title><body>" + "NbParametres: " + CountCGIParameters() + "<br><br>")

; La liste des tous les paramètres et affiche leur nom
;
For k = 0 To CountCGIParameters()-1
  If CGIParameterType("", k) = #PB_CGI_File
    WriteCGIString("[Fichier] "+CGIParameterName(k)+" (nom: "+CGIParameterValue("", k)+")<br>")
  Else
    WriteCGIString("[Chaine] "+CGIParameterName(k)+" (valeur: "+CGIParameterValue("", k)+")<br>")
  EndIf
Next

WriteCGIString("</body></html>")
```

Voir aussi

CGIParameterName() ,
CGIParameterValue() ,
CGIParameterData() ,
CGIParameterDataSize()

OS Supportés

Tous

81.8 CGIParameterData

Syntaxe

```
*Resultat =
    CGIParameterData(Nom$ [,
    Index])
```

Description

Renvoie l'adresse de la mémoire tampon du paramètre spécifié.

Arguments

Nom$ Le nom du paramètre.
Le nom du paramètre est sensible à la casse.

Index (optionnel) La première valeur de l'indice commence à 0.
Si spécifié, la valeur du paramètre 'Nom$' est ignorée (sauf si #PB_Ignore est utilisé).

Valeur de retour

Renvoie l'adresse de la mémoire tampon du paramètre spécifié.
Le type de paramètre doit être #PB_CGI_File.

Remarques

CGIParameterName() peut être utilisé pour obtenir le nom d'un paramètre spécifié.
Pour obtenir le nombre de paramètres disponibles, utiliser CountCGIParameters()
.
CGIParameterDataSize() peut être utilisé pour obtenir la taille de la mémoire tampon.

Exemple

```
1   If Not InitCGI() Or Not
     ReadCGI()
2     End
3   EndIf
4
5   WriteCGIHeader(#PB_CGI_HeaderContentType,
     "text/html",
     #PB_CGI_LastHeader) ;
     Ecrit les en-têtes pour
     informer le navigateur du
     format du contenu
6
7   WriteCGIString("<html><title>PureBasic
     -
     parametres</title><body>" +
```

```
8                   "NbParametres:
    " + CountCGIParameters() +
    "<br><br>")
9
10  ; La liste des tous les
    paramètres et affiche leur
    nom
11  ;
12  For k = 0 To
    CountCGIParameters()-1
13    If CGIParameterType("",
    k) = #PB_CGI_File
14      WriteCGIString("[Fichier]
    "+CGIParameterName(k)+"
    (nom:
    '"+CGIParameterValue("",
    k) +
15                                                          "'
    - taille: " +
    CGIParameterDataSize("",
    k) +
16                                                          "
    octets - *buffer: " +
    CGIParameterData("", k) +
    ")<br>")
17    EndIf
18  Next
19
20  WriteCGIString("</body></html>")
```

Voir aussi

CGIParameterName() ,
CGIParameterValue() ,
CGIParameterType() ,
CGIParameterDataSize()

OS Supportés

Tous

81.9 CGIParameterDataSize

Syntaxe

```
Resultat =
    CGIParameterDataSize(Nom$
    [, Index])
```

Description

Renvoie la taille des données du paramètre spécifié.

Arguments

Nom$ Le nom du paramètre.

Le nom du paramètre est sensible à la casse.

Index (optionnel) La première valeur de l'indice commence à 0.
Si spécifié, la valeur du paramètre 'Nom$' est ignorée (sauf si #PB_Ignore est utilisé).

Valeur de retour

Renvoie la taille des données (en octets) du paramètre spécifié.
Le type de paramètre doit être #PB_CGI_File.

Remarques

CGIParameterName() peut être utilisé pour obtenir le nom d'un paramètre spécifié.
Pour obtenir le nombre de paramètres disponibles, utiliser CountCGIParameters() .
CGIParameterDataSize() peut être utilisé pour obtenir la taille de la mémoire tampon.

Exemple

```
If Not InitCGI() Or Not ReadCGI()
  End
EndIf

WriteCGIHeader(#PB_CGI_HeaderContentType, "text/html", #PB_CGI_LastHeader) ; Ecrit les en-têtes pour informer le navigateur du format du contenu

WriteCGIString("<html><title>PureBasic - parametres</title><body>" + "NbParametres: " + CountCGIParameters() + "<br><br>")

; La liste des tous les paramètres et affiche leur nom
;
For k = 0 To CountCGIParameters()-1
  If CGIParameterType("", k) = #PB_CGI_File
    WriteCGIString("[Fichier] "+CGIParameterName(k)+" (nom: '"+CGIParameterValue("", k) +
```

```
15                                                          "'
    - taille: " +
    CGIParameterDataSize("",
    k) +
16                                                          "
    octets - *buffer: " +
    CGIParameterData("", k) +
    ")<br>")
17   EndIf
18  Next
19
20  WriteCGIString("</body></html>")
```

Voir aussi

CGIParameterName() ,
CGIParameterValue() ,
CGIParameterType() ,
CGIParameterData()

OS Supportés

Tous

81.10 CGIBuffer

Syntaxe

```
*Resultat = CGIBuffer()
```

Description

Pour les utilisateurs avancés. Renvoie l'adresse de la mémoire tampon de l'entrée brute CGI (utile seulement pour une requête de type POST).

Arguments

Aucun.

Valeur de retour

Renvoie l'adresse de la mémoire tampon de l'entrée CGI brute, ou zéro si une erreur est survenue.

Remarques

Il peut être utile de faire une analyse (parse) supplémentaire non prise en charge par cette bibliothèque tout en utilisant d'autres commandes. La taille de la mémoire tampon est la valeur renvoyée par ReadCGI() .

Exemple

```
If Not InitCGI()
  End
EndIf

BufferSize = ReadCGI()

WriteCGIHeader(#PB_CGI_HeaderContentType,
 "text/html",
 #PB_CGI_LastHeader) ;
 Ecrit les en-têtes pour
 informer le navigateur du
 format du contenu

WriteCGIString("<html><title>PureBasic
 - tampon
 brut(raw)</title><body>")
If CGIBuffer()
  WriteCGIString("le tampon
 brut contient: <br><pre>"
 + PeekS(CGIBuffer(),
 BufferSize, #PB_Ascii) +
 "</pre>")
EndIf

WriteCGIString("</body></html>")
```

Voir aussi

ReadCGI()

OS Supportés

Tous

81.11 CGIVariable

Syntaxe

```
Resultat\$ = CGIVariable(Nom$)
```

Description

Renvoie le contenu de la variable d'environnement CGI spécifiée.

Arguments

Nom$ Le nom de la variable à obtenir. Peut être une valeur personnalisée ou l'une des constantes prédéfinies suivantes :

```
#PB_CGI_AuthType
 : le procédé
 d'authentification
 utilisé par le
 navigateur Web
```

```
                                si
 un quelconque procédé
 d'authentification a été
 utilisé.
                                Il
 n'est pas défini, sauf
 si le script est protégé.

#PB_CGI_ContentLength
 : utilisé pour les
 scripts qui reçoivent
 des données de
                                formulaire
 utilisant la méthode
 POST. Cette variable
                                indique
 la longueur en octets du
 flux de données d'entrée
                                de
 CGI. Cela est nécessaire
 pour lire les données à
 partir
                                de
 l'entrée standard avec
 la méthode POST.

#PB_CGI_HeaderContentType
 : indique le type de
 support de données en
 cours de réception
                                de
 l'utilisateur.
                                il
 est utilisé pour les
 scripts appelés en
 utilisant la méthode
 POST.
#PB_CGI_DocumentRoot
 : le chemin de la racine
 de la page d'accueil
 HTML pour le serveur.
#PB_CGI_GatewayInterface
 : la version de CGI
 utilisé pour échanger
 les données entre le
 client et le serveur.
                                Généralement
 c'est CGI/1.1 pour le
 niveau de révision
 actuel.
#PB_CGI_PathInfo
 : informations de chemin
 supplémentaire ajouté à
 la fin de l'URL qui a
 accédé au programme de
 script côté serveur.
#PB_CGI_PathTranslated
 : une version traduite
 de la variable PATH_INFO
```

```
 par le serveur Web du
 virtuel vers le chemin
 physique.
#PB_CGI_QueryString
 : cette chaîne contient
 des informations à la
 fin du chemin de script
 côté serveur
                          qui
 a suivi un point
 d'interrogation.
                          Utilisé
 pour renvoyer les
 données si la méthode
 GET a été utilisé par un
 formulaire.
                          Il
 y a des restrictions de
 longueur à la
 QUERY_STRING.
#PB_CGI_RemoteAddr
 : l'adresse IP de
 l'ordinateur client.
#PB_CGI_RemoteHost
 : le nom de domaine
 complet de la machine du
 client faisant la
 requête HTTP.
                          Il
 peut ne pas être
 possible de déterminer
 ce nom puisque les noms
 de nombreux ordinateurs
 clients
                          ne
 sont pas enregistrées
 dans le système DNS.
#PB_CGI_RemoteIdent
 : la possibilité
 d'utiliser cette
 variable est limitée aux
 serveurs qui prennent
                          en
 charge la RFC 931. Cette
 variable peut contenir
 le nom d'utilisateur de
 la machine
                          cliente,
 mais il est destiné à
 être utilisé à des fins
 d'exploitation du
 fichier log,
                          quand
 il est disponible.
#PB_CGI_RemotePort
 : le port utilisé par
 les clients.
#PB_CGI_RemoteUser
 : si le script CGI a été
```

```
 protégé et l'utilisateur
 a été connecté pour
 accéder au script,
                              cette
 valeur contient le
 journal de l'utilisateur.
#PB_CGI_RequestURI
 : le chemin vers le
 fichier demandé par le
 client.
#PB_CGI_RequestMethod
 : décrit la méthode de
 requête utilisée par le
 navigateur qui est
 habituellement GET,
 POST, ou HEAD.
#PB_CGI_ScriptName
 : le chemin virtuel du
 script CGI en cours
 d'exécution.
#PB_CGI_ScriptFilename
 : le nom de fichier
 local du script en cours
 d'exécution.
#PB_CGI_ServerAdmin
 : l'adresse e-mail de
 l'administrateur du
 serveur.
#PB_CGI_ServerName
 : le nom du serveur,
 l'adresse IP ou le nom
 alias DNS présentée
 comme une URL
 d'auto-référencement.
                              Cela
 ne comprend pas
 l'identificateur de
 protocole tel que
 "HTTP", le nom de la
 machine, ou le numéro de
 port.
#PB_CGI_ServerPort
 : le numéro de port sur
 lequel les requêtes et
 les réponses HTTP sont
 envoyées.
#PB_CGI_ServerProtocol
 : cette valeur est
 généralement HTTP qui
 décrit le protocole
 utilisé entre les
 ordinateurs client et
 serveur.
#PB_CGI_ServerSignature
 : information sur le
 serveur en spécifiant le
 nom et la version du
 serveur Web et le port.
#PB_CGI_ServerSoftware
```

```
 : le nom et la version
 du serveur web.
#PB_CGI_HttpAccept
 : les types de supports
 de données que le
 navigateur client peut
 accepter.
                          Ces
 types de données sont
 séparées par des
 virgules.
#PB_CGI_HttpAcceptEncoding:
 le type d'encodage que
 le navigateur client
 accepte.
#PB_CGI_HttpAcceptLanguage:
 la langue que le
 navigateur client
 accepte.
#PB_CGI_HttpCookie
 : utilisé comme une
 variable d'environnement
 qui contient les cookies
 associés au domaine du
 serveur depuis le
 navigateur.
#PB_CGI_HttpForwarded
 : l'URL de la page
 transmise juste avant.
#PB_CGI_HttpHost
 : nom de l'hôte d'où
 viennent les requêtes
 HTTP.
#PB_CGI_HttpPragma
 : pragmas HTTP
#PB_CGI_HttpReferer
 : l'adresse de la page
 d'où la requête HTTP est
 originaire.
#PB_CGI_HttpUserAgent
 : le nom du navigateur
 Web du client qui a
 envoyé la requête.
```

Valeur de retour

Renvoie la valeur de la variable d'environnement CGI spécifiée.

Remarques

Lorsque le CGI est chargé, de nombreuses informations sont envoyées du serveur Web vers l'application CGI à travers les variables d'environnement.

Exemple

```
1   If Not InitCGI() Or Not
     ReadCGI()
2     End
3   EndIf
4
5   WriteCGIHeader(#PB_CGI_HeaderContentType,
     "text/html",
     #PB_CGI_LastHeader) ;
     Ecrit les en-têtes pour
     informer le navigateur du
     format du contenu
6
7   WriteCGIString("<html><title>PureBasic
     - variables</title><body>")
8
9   Procedure
     WriteCGIConstant(Constant$)
10    WriteCGIString(Constant$
     + ": " +
     CGIVariable(Constant$)+"<br>")
11  EndProcedure
12
13  WriteCGIConstant(#PB_CGI_AuthType)
14  WriteCGIConstant(#PB_CGI_ContentLength)
15  WriteCGIConstant(#PB_CGI_HeaderContentType)
16  WriteCGIConstant(#PB_CGI_DocumentRoot)
17  WriteCGIConstant(#PB_CGI_GatewayInterface)
18  WriteCGIConstant(#PB_CGI_PathInfo)
19  WriteCGIConstant(#PB_CGI_PathTranslated)
20  WriteCGIConstant(#PB_CGI_QueryString)
21  WriteCGIConstant(#PB_CGI_RemoteAddr)
22  WriteCGIConstant(#PB_CGI_RemoteHost)
23  WriteCGIConstant(#PB_CGI_RemoteIdent)
24  WriteCGIConstant(#PB_CGI_RemotePort)
25  WriteCGIConstant(#PB_CGI_RemoteUser)
26  WriteCGIConstant(#PB_CGI_RequestURI)
27  WriteCGIConstant(#PB_CGI_RequestMethod)
28  WriteCGIConstant(#PB_CGI_ScriptName)
29  WriteCGIConstant(#PB_CGI_ScriptFilename)
30  WriteCGIConstant(#PB_CGI_ServerAdmin)
31  WriteCGIConstant(#PB_CGI_ServerName)
32  WriteCGIConstant(#PB_CGI_ServerPort)
33  WriteCGIConstant(#PB_CGI_ServerProtocol)
34  WriteCGIConstant(#PB_CGI_ServerSignature)
35  WriteCGIConstant(#PB_CGI_ServerSoftware)
36  WriteCGIConstant(#PB_CGI_HttpAccept)
37  WriteCGIConstant(#PB_CGI_HttpAcceptEncoding)
38  WriteCGIConstant(#PB_CGI_HttpAcceptLanguage)
39  WriteCGIConstant(#PB_CGI_HttpCookie)
40  WriteCGIConstant(#PB_CGI_HttpForwarded)
41  WriteCGIConstant(#PB_CGI_HttpHost)
42  WriteCGIConstant(#PB_CGI_HttpPragma)
43  WriteCGIConstant(#PB_CGI_HttpReferer)
44  WriteCGIConstant(#PB_CGI_HttpUserAgent)
45
46  WriteCGIString("</body></html>")
```

Voir aussi

ReadCGI()

OS Supportés

Tous

81.12 FinishFastCGIRequest

Syntaxe

```
FinishFastCGIRequest()
```

Description

Termine la requête FastCGI en cours et libère toutes les ressources associées.

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle si une nouvelle requête a été traitée.

Remarques

Il n'est pas obligatoire d'utiliser cette commande, car la demande sera automatiquement terminée lorsque WaitFastCGIRequest() () est appelée de nouveau, ou quand le thread se termine. Cela peut être utile dans certains cas particuliers où les ressources sont faibles avant de faire un autre traitement.

Exemple

```
1  If Not InitCGI()
2    End
3  EndIf
4
5  If Not InitFastCGI(5600) ;
     Crée le programme FastCGI
     sur le port 5600
6    End
7  EndIf
8
9  While WaitFastCGIRequest()
10
11   If ReadCGI()
```

```
12        WriteCGIHeader(#PB_CGI_HeaderContentType,
      "text/html",
      #PB_CGI_LastHeader) ;
      Ecrit les en-têtes pour
      informer le navigateur du
      format du contenu
13
14        WriteCGIString("<html><title>PureBasic
      - FastCGI</title><body>" +
15                       "Hello
      depuis PureBasic FastCGI
      !<br>" +
16                       "Heure
      actuelle:
      <b>"+FormatDate("%hh:%ii",
      Date()) + "</b>" +
17                       "</body></html>")
18
19        FinishFastCGIRequest()
20
21        ; Traiter des choses
      ici...
22        ;
23        Delay(1000) ; Simuler
      un grand traitement
24
25      EndIf
26    Wend
```

Voir aussi

InitCGI() , InitFastCGI() , WaitFastCGIRequest()

OS Supportés

Tous

81.13 InitCGI

Syntaxe

```
Resultat =
    InitCGI([TailleRequeteMax])
```

Description

Initialise l'environnement CGI.

Arguments

TailleRequeteMax (optionnel) La taille maximale de la requête, en octets (la valeur par défaut est de 50 Mo).
Lors de l'envoi de grande quantité de données (comme les fichiers binaires), il pourrait être utile de spécifier une valeur plus grande.

Valeur de retour

Renvoie une valeur non nulle si l'environnement CGI a été correctement initialisé.

Remarques

Cette fonction doit être appelée avec succès avant d'utiliser d'autres commandes de cette bibliothèque.

Exemple

```
1  If Not InitCGI() Or Not
    ReadCGI()
2    End
3  EndIf
4
5  WriteCGIHeader(#PB_CGI_HeaderContentType,
    "text/html",
    #PB_CGI_LastHeader) ;
    Ecrit les en-têtes pour
    informer le navigateur du
    format du contenu
6
7  WriteCGIString("<html><title>PureBasic
    - variables</title><body>"
    +
8                 "Hello
    depuis PureBasic CGI !" +
9                 "</body></html>")
```

Voir aussi

ReadCGI()

OS Supportés

Tous

81.14 InitFastCGI

Syntaxe

```
Resultat =
    InitFastCGI(PortLocal)
```

Description

Initialise le support de FastCGI. Une fois appelées, toutes les commandes de CGI commutent automatiquement vers FastCGI. Cette bibliothèque supporte la gestion des threads FastCGI, à condition d'activer le mode "Activer la gestion des Threads" de PureBasic. FastCGI n'est prise en charge

uniquement qu'à travers un socket local. InitCGI() doit être appelée avant d'utiliser cette commande.
Contrairement à un programme CGI standard qui est lancé à chaque requête, le programme FastCGI reste en mémoire une fois lancé et peut gérer un certain nombre de demandes. Ce peut être très utile si l'initialisation de CGI prend du temps (comme la connexion à une base de données par exemple).

Arguments

PortLocal Le port local à utiliser.
Le serveur web doit être configuré pour utiliser ce port.

Valeur de retour

Renvoie une valeur non nulle si l'environnement FastCGI a été correctement initialisé.

Remarques

FastCGI peut être beaucoup plus facile à utiliser que CGI pour le développement, car le programme peut rester en mémoire et être déboggué comme une simple application PureBasic.
Pour configurer le support FastCGI sur Apache, vous devez activer les modules 'mod_proxy' et 'mod_proxy_fcgi', puis ajouter une déclaration 'ProxyPass' dans la configuration :

```
ProxyPass /myfastcgiapp/
 fcgi://localhost:5600/
```

Ici, l'url '/myfastcgiapp' va rediriger vers le programme FastCGI qui attend sur le port 5600.
Il est également possible d'exécuter le programme de FastCGI sur le serveur distant.

Exemple

```
1  If Not InitCGI()
2    End
3  EndIf
4
5  If Not InitFastCGI(5600) ;
    Crée le programme FastCGI
    sur le port 5600
6    End
7  EndIf
8
```

```
9   While WaitFastCGIRequest()
10
11    If ReadCGI()
12      WriteCGIHeader(#PB_CGI_HeaderContentType,
      "text/html",
      #PB_CGI_LastHeader) ;
      Ecrit les en-têtes pour
      informer le navigateur du
      format du contenu
13
14      WriteCGIString("<html><title>PureBasic
      - FastCGI</title><body>" +
15                     "Hello
      depuis PureBasic FastCGI
      !<br>" +
16                     "Temps
      actuel:
      <b>"+FormatDate("%hh:%ii",
      Date()) + "</b>" +
17                     "</body></html>")
18    EndIf
19
20  Wend
```

Voir aussi

InitCGI() , WaitFastCGIRequest()

OS Supportés

Tous

81.15 ReadCGI

Syntaxe

```
Resultat = ReadCGI()
```

Description

Lit la requête CGI.

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle si la requête CGI a été lue avec succès.
Si la requête était trop importante ou si une autre erreur survient alors zéro est renvoyé et le programme CGI doit être terminé.

Remarques

InitCGI() doit être appelé avec succès avant d'essayer le lire la requête CGI.

Exemple

```
If Not InitCGI() Or Not ReadCGI()
  End
EndIf

WriteCGIHeader(#PB_CGI_HeaderContentType, "text/html", #PB_CGI_LastHeader) ; Ecrit les en-têtes pour informer le navigateur du format du contenu

WriteCGIString("<html><title>PureBasic - variables</title><body>" +
               "Hello depuis PureBasic CGI !" +
               "</body></html>")
```

Voir aussi

InitCGI()

OS Supportés

Tous

81.16 WriteCGIFile

Syntaxe

```
Resultat = WriteCGIFile(Fichier$)
```

Description

Écrit un fichier entier sur la sortie CGI.

Arguments

Fichier$ Le fichier à écrire sur la sortie CGI.

Valeur de retour

Renvoie une valeur non nulle si le fichier a été écrit avec succès sur la sortie CGI.

Remarques

Lors de l'envoi des données binaires, l'en-tête "Content-Type" doit être défini en "application/octet-stream '.

Exemple

```
If Not InitCGI() Or Not
 ReadCGI()
  End
EndIf

WriteCGIHeader(#PB_CGI_HeaderContentType,
 "application/octet-stream")
WriteCGIHeader(#PB_CGI_HeaderContentDisposition,
 "attachment;
 filename=test.bmp",
 #PB_CGI_LastHeader)

WriteCGIFile(#PB_Compiler_Home
 +
 "examples/sources/data/PureBasic.bmp")
```

Voir aussi

InitCGI() , WriteCGIHeader()

OS Supportés

Tous

81.17 WriteCGIData

Syntaxe

```
Resultat =
    WriteCGIData(*Memoire,
    Taille)
```

Description

Écrit les données binaires sur la sortie CGI.

Arguments

***Memoire** Le tampon en mémoire à écrire.

Taille La taille (en octets) à écrire.

Valeur de retour

Renvoie une valeur non nulle si les données ont été écrites avec succès sur la sortie CGI.

Remarques

Lors de l'envoi des données binaires, l'en-tête "Content-Type" doit être défini en "application/octet-stream ".

Exemple

```
If Not InitCGI() Or Not ReadCGI()
  End
EndIf

WriteCGIHeader(#PB_CGI_HeaderContentType, "application/octet-stream")
WriteCGIHeader(#PB_CGI_HeaderContentDisposition, "attachment; filename=image.png", #PB_CGI_LastHeader)

If ReadFile(0, #PB_Compiler_Home + "examples/sources/data/world.png")
  Taille = Lof(0)
  *Memoire = AllocateMemory(Taille)
  ReadData(0, *Memoire, Taille) ; Lire l'intégralité du dossier dans le nouveau tampon mémoire

  WriteCGIData(*Memoire, Taille) ; Écrivez le tampon entier sur la sortie CGI

  CloseFile(0)
EndIf
```

Voir aussi

InitCGI() , WriteCGIHeader() , WriteCGIFile()

OS Supportés

Tous

81.18 WriteCGIHeader

Syntaxe

```
Resultat = WriteCGIHeader(EnTete$, Valeur$ [, Options])
```

Description

Ecrit un en-tête sur la sortie de CGI.
Les en-têtes doivent être écrits avant toutes autres données.

Arguments

EnTete$ L'en-tête à écrire.
Peut être une valeur personnelle ou l'une des valeurs suivantes :

```
#PB_CGI_HeaderContentLength
 : la longueur (en
 octets) du flux de
 sortie (implique des
 données binaires).
#PB_CGI_HeaderContentType
   : le type MIME du
 contenu du flux de
 sortie.
#PB_CGI_HeaderExpires
       : date et heure
 lorsque le document
 n'est plus valide et
 doit être rechargé par
 le navigateur.
#PB_CGI_HeaderLocation
      : redirection du
 serveur (ne peut pas
 être envoyé dans le
 cadre d'un en-tête
 complet).
#PB_CGI_HeaderPragma
        : mise en cache
 des documents ON/OFF.
#PB_CGI_HeaderStatus
        : état de la
 requête (ne peut pas
 être envoyé dans le
 cadre d'un en-tête
 complet).
#PB_CGI_HeaderContentDisposition:
 permet de spécifier un
 nom de fichier par
 défaut lors de l'envoi
 d'un fichier.
#PB_CGI_HeaderRefresh
       : le client
 recharge le document
 spécifié.
#PB_CGI_HeaderSetCookie
     : le client stocke
 les données spécifiées,
 utile pour garder la
 trace des données entre
 les requêtes.
```

Valeur$ La valeur d'en-tête à écrire.

Options (optionnel) Le codage de chaîne à utiliser.
Peut être l'une des valeurs suivantes :

```
#PB_Ascii (par défaut)
#PB_UTF8
```

En combinaison avec l'une des valeurs suivantes :

```
#PB_CGI_LastHeader : Ceci
 est le dernier en-tête
 écrit, ce qui signifie
 pas d'autres en-têtes
                      peuvent
 être envoyés. Cette
 option est obligatoire
 pour le dernier en-tête
 écrit.
```

Valeur de retour

Renvoie une valeur non nulle si l'en-tête a été écrit avec succès sur la sortie CGI.

Exemple

```
1  If Not InitCGI() Or Not
    ReadCGI()
2    End
3  EndIf
4
5  WriteCGIHeader(#PB_CGI_HeaderContentType,
    "text/html",
    #PB_CGI_LastHeader)
6
7  WriteCGIString("<html><title>PureBasic
    - test</title><body>" +
8                 "Hello
    depuis PureBasic CGI<br>" +
9                 "</body></html>")
```

Voir aussi

InitCGI() , WriteCGIFile() , WriteCGIStringN()

OS Supportés

Tous

81.19 WriteCGIString

Syntaxe

```
Resultat =
    WriteCGIString(Chaine$ [,
    Options])
```

Description

Ecrit une chaîne de caractères sur la sortie CGI.

Arguments

Chaine$ La chaîne de caractères à écrire.

Options (optionnel) L'encodage du texte.
Peut être l'une des valeurs suivantes :

```
#PB_UTF8 (par défaut)
#PB_Ascii
```

Valeur de retour

Renvoie une valeur non nulle si la chaîne a été écrite avec succès sur la sortie CGI.

Exemple

```
If Not InitCGI() Or Not ReadCGI()
  End
EndIf

WriteCGIHeader(#PB_CGI_HeaderContentType, "text/html", #PB_CGI_LastHeader)

WriteCGIString("<html><title>PureBasic - test</title><body>" +
               "Hello depuis PureBasic CGI<br>" +
               "Temps actuel: <b>"+FormatDate("%hh:%ii", Date()) + "</b>" +
               "</body></html>")
```

Voir aussi

InitCGI() , WriteCGIHeader() , WriteCGIStringN()

OS Supportés

Tous

81.20 WriteCGIStringN

Syntaxe

```
Resultat =
    WriteCGIStringN(Chaine$ [,
    Options])
```

Description

Ecrit une chaîne de caractères sur la sortie CGI incluant un retour charriot à la fin.

Arguments

Chaine$ La chaîne de caractères à écrire.

Options (optionnel) L'encodage du texte.
Peut être l'une des valeurs suivantes :

```
#PB_UTF8 (par défaut)
#PB_Ascii
```

Valeur de retour

Renvoie une valeur non nulle si la chaîne a été écrite avec succès sur la sortie CGI.

Exemple

```
If Not InitCGI() Or Not ReadCGI()
  End
EndIf

WriteCGIHeader(#PB_CGI_HeaderContentType, "text/html", #PB_CGI_LastHeader)

; L'utilisation du retour chariot permet une meilleure lisibilité de la page lorsque "Affiche la source de la page" est utilisé dans le navigateur.
;
WriteCGIStringN("<html><title>PureBasic - test</title><body>")
WriteCGIStringN("Hello depuis PureBasic CGI<br>")
WriteCGIStringN("Temps actuel: <b>"+FormatDate("%hh:%ii", Date()) + "</b>")
WriteCGIStringN("</body></html>")
```

Voir aussi

InitCGI() , WriteCGIHeader() , WriteCGIString()

OS Supportés

Tous

81.21 WaitFastCGIRequest

Syntaxe

```
Resultat =
    WaitFastCGIRequest()
```

Description

Attend une nouvelle requête entrante. Cette commande va arrêter l'exécution du programme jusqu'à ce qu'une nouvelle demande soit disponible.

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle si une nouvelle requête a été traitée.

Remarques

InitFastCGI() doit être appelé avec succès avant d'utiliser cette commande.

Exemple

```
1   If Not InitCGI()
2     End
3   EndIf
4
5   If Not InitFastCGI(5600) ; Crée le programme FastCGI sur le port 5600
6     End
7   EndIf
8
9   While WaitFastCGIRequest()
10
11    If ReadCGI()
12      WriteCGIHeader(#PB_CGI_HeaderContentType, "text/html", #PB_CGI_LastHeader) ; Ecrit les en-têtes pour informer le navigateur du format du contenu
13
14      WriteCGIString("<html><title>PureBasic - FastCGI</title><body>" +
15                     "Hello depuis PureBasic FastCGI !<br>" +
16                     "Temps actuel: <b>"+FormatDate("%hh:%ii", Date()) + "</b>" +
17                     "</body></html>")
18    EndIf
19  Wend
```

Voir aussi

InitCGI() , InitFastCGI() ,
FinishFastCGIRequest()

OS Supportés

Tous

Chapitre 82

Camera

Généralités

Les caméras sont utilisées pour visualiser un monde en 3D. Il est possible de les utiliser comme des caméras réelles. On peut déplacer une caméra, la faire pivoter, modifier le champ de vision, etc.
Au moins une caméra est nécessaire pour effectuer un rendu du monde 3D sur un écran, mais plusieurs caméras peuvent être utilisées en même temps pour afficher le monde sous des angles de vue différents, rétroviseurs, split-screen....
La commande InitEngine3D() doit être appelée avec succès avant de pouvoir utiliser les commandes relatives aux caméras.

OS Supportés

Tous

82.1 CameraBackColor

Syntaxe

```
CameraBackColor(#Camera,
    Couleur)
```

Description

Change la couleur de fond d'une caméra.

Arguments

#Camera La caméra à utiliser.

Couleur Nouvelle couleur de fond.
La fonction RGB() peut être utilisée pour obtenir une couleur valide.

Valeur de retour

Aucune.

Remarques

Par défaut, une caméra nouvellement créée a une couleur de fond noir.

Voir aussi

CreateCamera()

OS Supportés

Tous

82.2 CameraFollow

Syntaxe

```
CameraFollow(#Camera,
    ObjetID, Angle, Hauteur,
    Distance,
    PourcentageRotation,
    PourcentagePosition [,
    Mode])
```

Description

Suivre un objet.

Arguments

#Camera La caméra à utiliser.

ObjetID L'objet à suivre. Il peut être l'un des types suivants :

```
    - Entité                 :
      Utiliser EntityID()
  pour obtenir un identifiant
      valide.
    - Lumière                :
      Utiliser LightID()
  pour obtenir un identifiant
      valide.
    - Noeud                  :
      Utiliser NodeID()
  pour obtenir un identifiant
      valide.
    - Emetteur de Particles :
      Utiliser
      ParticleEmitterID()
  pour obtenir un identifiant
      valide.
    - Groupe de Billboards  :
      Utiliser
      BillboardGroupID()
  pour obtenir un identifiant
      valide.
    - Texte 3D               :
      Utiliser Text3DID()
  pour obtenir un identifiant
      valide.
```

Angle L'angle de la caméra par rapport à l'objet suivi.

Hauteur La hauteur absolue de la caméra.

Distance La distance de la caméra par rapport à l'objet suivi.

PourcentageRotation Valeur à appliquer lorsque la caméra tourne pour obtenir à nouveau le bon angle.
Les valeurs valides vont de 0 à 1.

PourcentagePosition Valeur à appliquer lorsque la caméra se déplace pour obtenir la position correcte à nouveau.
Les valeurs valides vont de 0 à 1.
Lorsque la valeur est 0, la caméra ne bouge pas. Lorsque la valeur est 1, l'appareil est réglé sur la position finale, sans interpolation.

Mode (optionnel) Peut être une des valeurs suivantes :

```
#True : La caméra regarde
 automatiquement l'objet
 (par défaut).
#False: La caméra ne
 regarde pas
 automatiquement l'objet.
```

Valeur de retour

Aucune.

Remarques

La poursuite de l'objet se fait de façon douce en utilisant l'interpolation.

Voir aussi

CreateCamera()

OS Supportés

Tous

82.3 CameraFOV

Syntaxe

```
CameraFOV(#Camera, Angle)
```

Description

Modifie le champ de vision d'une caméra (FOV : Field Of Vision) pour permettre une vue plus ou moins large de la scène.

Arguments

#**Camera** La caméra à utiliser.

Angle Le nouveau champ de vision en degrés.
Un angle supérieur à 90° donnera un effet de type grand-angle (ou fish-eye).
Un angle inférieur à 30° donnera un effet rétréci (téléscopique) à la scène.
Les valeurs habituelles sont comprises entre 45° et 60°.

Valeur de retour

Aucune.

Voir aussi

CameraRange()

OS Supportés

Tous

82.4 CameraID

Syntaxe

```
Resultat = CameraID(#Camera)
```

Description

Renvoie l'identifiant unique d'une caméra.

Arguments

#**Camera** La caméra à utiliser.

Valeur de retour

Renvoie le numéro de la caméra.

Remarques

Cette fonction est très utile quand une fonction d'une autre bibliothèque nécessite l'identifiant d'une caméra.

Voir aussi

CreateCamera() , IsCamera()

OS Supportés

Tous

82.5 CameraCustomParameter

Syntaxe

```
CameraCustomParameter(#Camera,
    ParametreIndex,
    Valeur1,Valeur2, Valeur3,
    Valeur4)
```

Description

Définit des valeurs personnalisées pour un paramètre d'un script shaders (soit GLSL soit HLSL) d'une caméra. .

Arguments

#**Camera** La caméra à utiliser.

ParametreIndex L'indice de paramètre dans le script shader.

Valeur1 La première valeur du paramètre.

Valeur2 La seconde valeur du paramètre (si le paramètre n'en n'accepte qu'une, cette valeur sera ignorée).

Valeur3 La troisième valeur du paramètre (si le paramètre n'en n'accepte que deux, cette valeur sera ignorée).

Valeur La quatrième valeur du paramètre (si le paramètre n'en n'accepte que trois, cette valeur sera ignorée).

Valeur de retour

Aucune.

OS Supportés

Tous

82.6 CheckObjectVisibility

Syntaxe

```
Resultat =
    CheckObjectVisibility(#Camera,
    ObjetID)
```

Description

Vérifie si un objet est visible par une caméra.

Arguments

#**Camera** La caméra à utiliser.

ObjetID L'objet à vérifier. Il peut s'agir de l'un des types suivants :

```
- Entité
  : Utiliser EntityID()
pour obtenir un
  identifiant valide.
- Lumière
  : Utiliser LightID()
pour obtenir un
  identifiant valide.
- Noeud
  : Utiliser NodeID()
pour obtenir un
  identifiant valide.
- Emmetteur de Particules
  : Utiliser
  ParticleEmitterID()
pour obtenir un
  identifiant valide.
- Groupe de Billboards
  : Utiliser
  BillboardGroupID()
pour obtenir un
  identifiant valide.
- Texte3D
  : Utiliser Text3DID()
pour obtenir un
  identifiant valide.
```

Valeur de retour

Renvoie #True si l'objet est visible, #False sinon.

Exemple

```
1   InitEngine3D()
2   InitSprite()
3   InitKeyboard()
4
5   OpenWindow(0, 0, 0, 800,
     600, "Visibilité d'un
     objet (Utilisez les
     flèches <--   -->  et
     Echap pour terminer)",
     #PB_Window_ScreenCentered)
6   OpenWindowedScreen(WindowID(0),0,
     0, 800, 600)
7   CreateCamera(0, 0, 0, 100,
     100)
8
9   CreateCube(0, 1)
10  CreateTexture(0, 100, 100)
11  If
     StartDrawing(TextureOutput(0))
12    DrawingMode(#PB_2DDrawing_Gradient)
13    CircularGradient(50, 50,
     48)
14    Circle(50, 50, 48)
15    StopDrawing()
```

```
16  EndIf
17
18  CreateMaterial(0,
     TextureID(0))
19  ScaleMaterial(0, 0.1, 0.1)
20  CreateEntity(0, MeshID(0),
     MaterialID(0), 0, 0, -2)
21
22  Repeat
23  Repeat : Event =
     WindowEvent() : Until
     Event = 0
24
25  RenderWorld()
26  RotateEntity(0, 0.1, -0.3,
     0.3, #PB_Relative)
27  ExamineKeyboard()
28
29  If
     KeyboardPushed(#PB_Key_Left)
30    MoveEntity(0, -0.1, 0,
     0,#PB_Relative)
31  ElseIf
     KeyboardPushed(#PB_Key_Right)
32    MoveEntity(0, 0.1, 0,
     0,#PB_Relative)
33  EndIf
34
35  If CheckObjectVisibility(0,
     EntityID(0)) = #False
36    Debug "Can't see the
     object anymore"
37  Else
38    Debug "I see it !"
39  EndIf
40
41  FlipBuffers()
42  Until
     KeyboardPushed(#PB_Key_Escape)
```

OS Supportés

Tous

82.7 CameraDirection

Syntaxe

```
CameraDirection(#Camera, X,
   Y, Z)
```

Description

Change l'axe d'une caméra.

Arguments

#**Camera** La caméra à utiliser.

X, Y, Z La nouvelle direction.
C'est un vecteur généralement compris entre -1.0 et 1.0, sinon il sera automatiquement normalisé.

Valeur de retour

Aucune.

Remarques

La position de la caméra reste inchangée.

Voir aussi

CameraDirectionX() , CameraDirectionY() , CameraDirectionZ()

OS Supportés

Tous

82.8 CameraDirectionX

Syntaxe

```
Resultat =
    CameraDirectionX(#Camera
    [, Mode])
```

Description

Renvoie la direction en X de la caméra.

Arguments

#Camera La caméra à utiliser.

Mode (optionnel) Le mode d'obtention du vecteur de direction de la caméra. Peut être l'une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 direction de la caméra
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 direction de la caméra
 par rapport à son parent.
```

Valeur de retour

Renvoie le vecteur de direction X de la caméra.
Cette valeur est généralement comprise entre -1.0 et 1.0, sinon il sera automatiquement normalisé.

Voir aussi

CameraDirection() , CameraDirectionY() , CameraDirectionZ()

OS Supportés

Tous

82.9 CameraDirectionY

Syntaxe

```
Resultat =
    CameraDirectionY(#Camera
    [, Mode])
```

Description

Renvoie la direction Y de la caméra.

Arguments

#**Camera** La caméra à utiliser.

Mode (optionnel) Le mode d'obtention du vecteur de direction de la caméra. Peut être l'une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 direction de la caméra
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 direction de la caméra
 par rapport à son parent.
```

Valeur de retour

Retourne le vecteur de direction Y de la caméra.
Cette valeur est généralement comprise entre -1.0 et 1.0, sinon il sera automatiquement normalisé.

Voir aussi

CameraDirection() , CameraDirectionX() , CameraDirectionZ()

OS Supportés

Tous

82.10 CameraDirectionZ

Syntaxe

```
Resultat =
    CameraDirectionZ(#Camera
    [, Mode])
```

Description

Renvoie la direction Z de la caméra.

Arguments

#**Camera** La caméra à utiliser.

Mode (optionnel) Le mode d'obtention du vecteur de direction de la caméra. Peut être l'une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 direction de la caméra
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 direction de la caméra
 par rapport à son parent.
```

Valeur de retour

Retourne le vecteur de direction Z de la caméra.
Cette valeur est généralement comprise entre -1.0 et 1.0, sinon il sera automatiquement normalisé.

Voir aussi

CameraDirection() , CameraDirectionX() , CameraDirectionY()

OS Supportés

Tous

82.11 CameraFixedYawAxis

Syntaxe

```
CameraFixedYawAxis(#Camera,
    Activer [, VecteurX,
    VecteurY, VecteurZ])
```

Description

Change l'axe fixe de lacet d'une caméra.

Arguments

#**Camera** La caméra à utiliser.

Activer Active ou désactive l'utilisation d'un axe de lacet personnalisé.

```
#True : Un nouveau vecteur
 doit être spécifié.
#False: La caméra fera un
 lacet autour de son
 propre axe Y.
```

VecteurX, VecteurY, VecteurZ (optionnel) Direction du nouvel axe de lacet. Généralement le vecteur a une valeur comprise entre -1.0 et 1.0, sinon il sera automatiquement normalisé). Le paramètre "Activer" doit être sur #True pour avoir un effet.

Valeur de retour

Aucune.

Remarques

Le comportement par défaut de la caméra est un lacet autour de son propre axe Y.

Voir aussi

CameraYaw()

OS Supportés

Tous

82.12 CameraLookAt

Syntaxe

```
CameraLookAt(#Camera, X, Y, Z)
```

Description

Spécifie le point, dans l'unité du monde qui fait face à une caméra.

Arguments

#**Camera** La caméra à utiliser.

X, Y, Z La caméra pointe vers la position 'X, Y, Z'.

Valeur de retour

Aucune.

OS Supportés

Tous

82.13 CameraProjectionMode

Syntaxe

```
CameraProjectionMode(#Camera,
    Mode [, Largeur, Hauteur]))
```

Description

Change le mode de projection utilisé par une caméra.

Arguments

#**Camera** La caméra à utiliser.

Mode Le paramètre 'Mode' représente la façon dont le monde sera projeté :

```
#PB_Camera_Perspective :
 Rend la scène en tenant
 compte de la perspective
#PB_Camera_Orthographic:
 Rend la scène sans
 perspective (plat, pas
 de profondeur)
```

Largeur (optionnel) Largeur de la zone d'affichage.

Hauteur (optionnel) Hauteur de la zone d'affichage.

Valeur de retour

Aucune.

Voir aussi

CameraProjectionX() , CameraProjectionY()

OS Supportés

Tous

82.14 CameraProjectionX

Syntaxe

```
Resultat =
    CameraProjectionX(#Camera,
    X, Y, Z)
```

Description

Renvoie la position X d'un point 3D vu par une caméra.

Arguments

#Camera La caméra à utiliser.

X, Y, Z Les coordonnées du point.

Valeur de retour

Renvoie la position X, en pixels, d'un point 3D vu par la caméra.
Si ce point est en dehors du champ de vision de la caméra alors 'Resultat' = -1.

Remarques

Cette fonction est très utile pour mapper des points 3D sur un écran 2D.

Voir aussi

CameraProjectionMode() , CameraProjectionY()

OS Supportés

Tous

82.15 CameraProjectionY

Syntaxe

```
Resultat =
    CameraProjectionY(#Camera,
    X, Y, Z)
```

Description

Renvoie la position Y d'un point 3D vu par une caméra.

Arguments

#Camera La caméra à utiliser.

X, Y, Z Les coordonnées du point.

Valeur de retour

Renvoie la position Y, en pixels, d'un point 3D vu par la caméra.
Si ce point est en dehors du champ de vision de la caméra alors 'Resultat' = -1.

Remarques

Cette fonction est très utile pour mapper des points 3D sur un écran 2D.

Voir aussi

CameraProjectionMode() ,
CameraProjectionX()

OS Supportés

Tous

82.16 CameraRange

Syntaxe

```
CameraRange(#Camera, Proche,
    Lointain)
```

Description

Modifie les valeurs du champ de vision d'une caméra.

Arguments

#**Camera** La caméra à utiliser.

Proche Indique la distance la plus proche de la caméra à partir de laquelle la scène sera rendue.

Lointain Indique la distance maximale jusqu'à laquelle la scène sera rendue.

Valeur de retour

Aucune.

Voir aussi

CameraLookAt()

OS Supportés

Tous

82.17 CameraRenderMode

Syntaxe

```
CameraRenderMode(#Camera,
    Mode)
```

Description

Change le mode de rendu d'une caméra.

Arguments

#**Camera** La caméra à utiliser.

Mode La manière dont le monde doit être rendu :

```
#PB_Camera_Plot      : Rend
 la scène en mode point
 (seuls les sommets des
 objets sont visibles)
#PB_Camera_Wireframe: Rend
 la scène en mode ligne
 (seuls les triangles des
 objets sont visibles)
#PB_Camera_Textured : Rend
 la scène avec un maximum
 de détails (textures,
 transparence etc..)
```

Valeur de retour

Aucune.

Remarques

Quand vous créez une nouvelle caméra avec CreateCamera() , le mode de rendu par défaut se fait avec un niveau de détails et de textures maximum.

Voir aussi

CameraProjectionMode()

OS Supportés

Tous

82.18 CameraReflection

Syntaxe

```
CameraReflection(#Camera,
   #CameraPrincipale,
   EntiteID)
```

Description

Définit une caméra comme une caméra-miroir.

Arguments

#**Camera** La caméra à utiliser comme caméra-miroir.

#**CameraPrincipale** La caméra à utiliser comme source d'image à refléter.

EntiteID L'EntityID() à utiliser comme source d'image à refléter.

Valeur de retour

Aucune.

Remarques

Ce sont à la fois #CameraPrincipale et EntiteID qui seront utilisés comme sources de l'image à refléter en image miroir.
Une texture RTT doit être créée à partir de #Camera à laide de CreateRenderTexture() .
La matière (matériau) qui utilisera cette texture RTT doit être définie avec SetMaterialAttribute() de cette manière : SetMaterialAttribute(Matiere, #PB_Material_ProjectiveTexturing, #Camera).
Cette commande doit être utilisée dans la boucle de rendu.

Voir aussi

SetMaterialAttribute()

OS Supportés

Tous

82.19 CameraRoll

Syntaxe

```
Resultat.f =
    CameraRoll(#Camera [,
    Mode])
```

Description

Renvoie le roulis d'une caméra.

Arguments

#Camera La caméra à utiliser.

Mode (optionnel) Peut être une des valeurs suivantes :

```
    #True : Valeur brute du
      roulis, mais elle ne
      peut pas être utilisée
      avec RotateCamera()
   pour récupérer la même
      orientation (par défaut).
    #False: Le roulis est
      ajusté, de sorte qu'il
      peut être réutilisé avec
      RotateCamera()
   pour récupérer la même
      orientation.
```

Valeur de retour

La valeur du roulis de la caméra.
Valeur entre -180.0 et 180.0 degrés.

Voir aussi

CameraYaw() , CameraPitch()

OS Supportés

Tous

82.20 CameraPitch

Syntaxe

```
Resultat.f =
    CameraPitch(#Camera [,
    Mode])
```

Description

Renvoie le tangage d'une caméra.

Arguments

#**Camera** La caméra à utiliser.

Mode (optionnel) Peut être une des valeurs suivantes :

```
    #True : Valeur brute du
      tangage, mais elle ne
      peut pas être utilisée
      avec RotateCamera()
  pour récupérer la même
      orientation (par défaut).
    #False: Le tangage est
      ajusté, de sorte qu'il
      peut être réutilisé avec
      RotateCamera()
  pour récupérer la même
      orientation.
```

Valeur de retour

La valeur du tangage de la caméra.
Valeur entre -180.0 et 180.0 degrés.

Voir aussi

CameraYaw() , CameraRoll()

OS Supportés

Tous

82.21 CameraYaw

Syntaxe

```
Resultat.f =
    CameraYaw(#Camera [, Mode])
```

Description

Renvoie le lacet d'une caméra.

Arguments

#Camera La caméra à utiliser.

Mode (optionnel) Peut être une des valeurs suivantes :

```
    #True : Valeur brute du
      lacet, mais elle ne peut
      pas être utilisée avec
      RotateCamera()
   pour récupérer la même
      orientation (par défaut).
    #False: Le lacet est
      ajusté, de sorte qu'il
      peut être réutilisé avec
      RotateCamera()
   pour récupérer la même
      orientation.
```

Valeur de retour

La valeur du lacet de la caméra.
Valeur entre -180.0 et 180.0 degrés.

Voir aussi

CameraPitch() , CameraRoll()

OS Supportés

Tous

82.22 CameraViewX

Syntaxe

```
Resultat =
    CameraViewX(#Camera)
```

Description

Renvoie la position en X de l'image affichée par une caméra.

Arguments

#Camera La caméra à utiliser.

Valeur de retour

Renvoie la position en X, en pixels de l'image affichée par la caméra.

Voir aussi

CameraViewY() , CameraViewHeight() , CameraViewWidth()

OS Supportés

Tous

82.23 CameraViewY

Syntaxe

```
Resultat =
    CameraViewY(#Camera)
```

Description

Renvoie la position en Y de l'image affichée par une caméra.

Arguments

#**Camera** La caméra à utiliser.

Valeur de retour

Renvoie la position en Y, en pixels de l'image affichée par la caméra.

Voir aussi

CameraViewX() , CameraViewHeight() , CameraViewWidth()

OS Supportés

Tous

82.24 CameraViewWidth

Syntaxe

```
Resultat =
    CameraViewWidth(#Camera)
```

Description

Renvoie la largeur de l'image affichée par une caméra.

Arguments

#**Camera** La caméra à utiliser.

Valeur de retour

Renvoie la largeur, en pixels de l'image affichée par la caméra.

Voir aussi

CameraViewHeight() , CameraViewX() , CameraViewY()

OS Supportés

Tous

82.25 CameraViewHeight

Syntaxe

```
Resultat =
    CameraViewHeight(#Camera)
```

Description

Renvoie la hauteur de l'image affichée par une caméra.

Arguments

#**Camera** La caméra à utiliser.

Valeur de retour

Renvoie la hauteur, en pixels de l'image affichée par la caméra.

Voir aussi

CameraViewWidth() , CameraViewX() , CameraViewY()

OS Supportés

Tous

82.26 CameraX

Syntaxe

```
Resultat = CameraX(#Camera [,
    Mode])
```

Description

Renvoie la position courante en X d'une caméra dans le monde 3D.

Arguments

#**Camera** La caméra à utiliser.

Mode (optionnel) Le mode d'obtention du vecteur de direction de la caméra. Peut être l'une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 direction de la caméra
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 direction de la caméra
 par rapport à son parent.
```

Valeur de retour

Renvoie la position en X de la caméra.

Voir aussi

CameraY() , CameraZ() , MoveCamera()

OS Supportés

Tous

82.27 CameraY

Syntaxe

```
Resultat = CameraY(#Camera [,
    Mode])
```

Description

Renvoie la position courante en Y d'une caméra dans le monde 3D.

Arguments

#**Camera** La caméra à utiliser.

Mode (optionnel) Le mode d'obtention du vecteur de direction de la caméra. Peut être l'une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 direction de la caméra
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 direction de la caméra
 par rapport à son parent.
```

Valeur de retour

Renvoie la position en Y de la caméra.

Voir aussi

CameraX() , CameraZ() , MoveCamera()

OS Supportés

Tous

82.28 CameraZ

Syntaxe

```
Resultat = CameraZ(#Camera [,
    Mode])
```

Description

Renvoie la position courante en Z d'une caméra dans le monde 3D.

Arguments

#**Camera** La caméra à utiliser.

Mode (optionnel) Le mode d'obtention du vecteur de direction de la caméra. Peut être l'une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 direction de la caméra
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 direction de la caméra
 par rapport à son parent.
```

Valeur de retour

Renvoie la position en Z de la caméra.

Voir aussi

CameraX() , CameraY() , MoveCamera()

OS Supportés

Tous

82.29 CreateCamera

Syntaxe

```
Resultat =
    CreateCamera(#Camera, X,
    Y, Largeur, Hauteur [,
    MasqueVisibilite] [,
    NiveauDetails]])
```

Description

Crée une nouvelle caméra dans le monde courant.

Arguments

#Camera Numéro de la nouvelle caméra. #PB_Any peut être utilisé pour autogénérer ce numéro.

X, Y La position (en pourcentage) du bord gauche de l'affichage de la camera. Voir l'image ci-dessous.

```
0%  : Affichage sur le
 bord gauche (haut) de
 l'écran
100%: Affichage sur le
 bord droit (bas) de
 l'écran
```

Largeur, Hauteur Les dimensions (en pourcentage) de l'affichage de la camera. Voir l'image ci-dessous.

MasqueVisibilite (optionnel) Un masque pour sélectionner les entités et les billboards à afficher par la caméra. La caméra définit son propre masque, et si l'entité ou le billboard masqué correspond, alors il sera affiché. Par défaut, les entités et les billboards n'ont pas de masque, ce qui signifie qu'ils seront toujours affichés par toutes les caméras.

NiveauDetails (optionnel) Le niveau de détails (Lod Bias = Level Of Details) à appliquer aux entités qui le supportent. S'il est inférieur à 1, un objet plus détaillé sera affiché, s'il est supérieur à 1, une entité moins détaillée sera affichée. La distance de l'entité est divisée par le niveau de détail, par exemple un niveau de détail de 0.5 double la distance.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro si la caméra n'a pas pu être crée.
Si #PB_Any est utilisé à la place du paramètre '#Camera' alors le nouveau numéro de la caméra sera renvoyé dans 'Resultat'.

Remarques

Le fait de poser la position et les dimensions d'une caméra (ce qu'elle affiche en fait) en pourcentage permet de

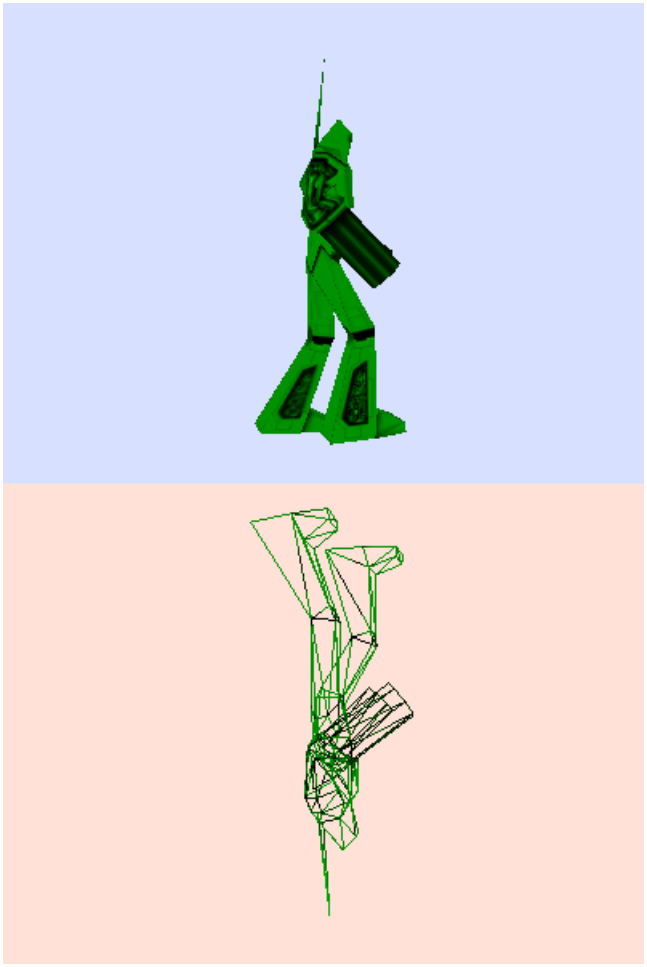
s'abstraire des dimensions de l'écran de l'utilisateur. Ainsi une caméra affichera la même chose sur un écran 800x600 que sur un écran 1024x768, etc.

Exemple

```
CreateCamera(0, 0, 0, 100, 100) ; Crée une caméra qui prend tout l'écran

CreateCamera(0, 0,  0, 100, 50) ; Crée une caméra qui ne prend que 50% de la hauteur (effet Split-Screen pour 2 joueurs)
CreateCamera(1, 0, 50, 100, 50) ; Crée la deuxieme caméra de 50% de hauteur mais dans la partie basse de l'écran

CreateCamera(0, 0, 0, 100, 100) ; Crée une caméra qui prend tout l'écran
CreateCamera(1, 25, 0, 50, 10)  ; Avec un effet rétroviseur (miroir).
                                                                    ;
                                Notez que le rétroviseur est le dernier, de sorte qu'il est affiché en haut de la caméra plein écran
```

Exemple

```
; 2 caméras qui séparent l'écran en deux parties égales :

; Caméra au dessus
CreateCamera(0, 0, 0, 100, 50)
MoveCamera(0, 0, 50, 150, #PB_Absolute)
CameraBackColor(0, RGB(215, 225, 255))

; Caméra en dessous
CreateCamera(1, 0, 50, 100, 50)
MoveCamera(1, 0, 50, -150, #PB_Absolute)
CameraBackColor(1, RGB(255, 225, 215))
RotateCamera(1, 180, 0, 0)
```



Voir aussi

FreeCamera() , ResizeCamera()

OS Supportés

Tous

82.30 FreeCamera

Syntaxe

```
FreeCamera(#Camera)
```

Description

Supprime une caméra et libère la mémoire associée.

Arguments

#Camera La caméra à libérer.
Si #PB_All est spécifié, toutes les caméras restantes sont libérées.

Valeur de retour

Aucune.

Remarques

Toutes les caméras restantes sont automatiquement libérées lorsque le programme se termine.

Voir aussi

CreateCamera()

OS Supportés

Tous

82.31 IsCamera

Syntaxe

```
Resultat = IsCamera(#Camera)
```

Description

Teste si une caméra est correctement initialisée.

Arguments

#**Camera** La caméra à tester.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'une caméra est correctement initialisée.

Voir aussi

CameraID() , CreateCamera()

OS Supportés

Tous

82.32 MoveCamera

Syntaxe

```
MoveCamera(#Camera, X, Y, Z
    [, Mode])
```

Description

Déplace une caméra dans le monde 3D.

Arguments

#Camera La caméra à déplacer.

X, Y, Z La nouvelle position de la caméra.

Mode (optionnel) Peut être une des valeurs suivantes :

```
#PB_Relative: Déplacement
 relatif, à partir de la
 position actuelle de la
 caméra (par défaut).
#PB_Absolute: Déplacement
 absolu à la position
 spécifiée.
```

combinée avec l'une des valeurs suivantes :

```
#PB_Local : Déplacement
 par rapport à sa propre
 orientation.
#PB_Parent: Déplacement
 par rapport à la
 position de son parent.
#PB_World : Déplacement
 par rapport au monde.
```

Valeur de retour

Aucune.

Voir aussi

CameraX() , CameraY() , CameraZ() , RotateCamera()

OS Supportés

Tous

82.33 ResizeCamera

Syntaxe

```
ResizeCamera(#Camera, X, Y,
    Largeur, Hauteur)
```

Description

Redimensionne l'affichage d'une caméra.

Arguments

#Camera La caméra à redimensionner.

X, Y La nouvelle position de l'affichage de la caméra.
Ces valeurs sont exprimées en pourcentage, soit de 0 à 100.

Largeur, Hauteur La nouvelle taille de l'affichage de la caméra.
Ces valeurs sont exprimées en pourcentage, soit de 0 à 100.

Valeur de retour

Aucune.

Voir aussi

CreateCamera()

OS Supportés

Tous

82.34 RotateCamera

Syntaxe

```
RotateCamera(#Camera, X, Y, Z
    [, Mode])
```

Description

Effectue une rotation d'une caméra.

Arguments

#Camera La caméra à tourner.

X, Y, Z Valeur des angles en x,y,z.
Tous les angles sont en degrés, de 0 à 360 degrés.

Mode (optionnel) Peut être une des valeurs suivantes :

```
PB_Absolute: Rotation
 absolue (par défaut).
PB_Relative: Rotation
 relative basée sur la
 rotation précédente.
```

Valeur de retour

Aucune.

Voir aussi

CameraPitch() , CameraRoll() , CameraYaw() , MoveCamera()

OS Supportés

Tous

82.35 SwitchCamera

Syntaxe

```
SwitchCamera(#Camera,
    #NouvelleCamera)
```

Description

Echange une caméra avec une autre.

Arguments

#**Camera** La caméra à échanger.

#**NouvelleCamera** La nouvelle caméra à afficher.

Valeur de retour

Aucune.

Voir aussi

CreateCamera()

OS Supportés

Tous

Chapitre 83

Cipher

Généralités

La bibliothèque Cipher propose un jeu de fonctions pour le chiffrement ou l'encodage des données. Par exemple la signature SHA-2 est utilisée dans de nombreux domaines en raison de sa résistance aux piratage.

OS Supportés

Tous

83.1 AddCipherBuffer

Syntaxe

```
AddCipherBuffer(#Cipher,
    *Entree, *Sortie, Taille)
```

Description

Ajoute de nouvelles données au chiffrement démarré avec StartAESCipher() et copie les données cryptées dans le tampon de sortie.

Arguments

#Cipher Le chiffrement auquel les données doivent être ajoutées.

***Entree** Le tampon d'entrée.

***Sortie** Le tampon de sortie.

Taille La taille des données à chiffrer. Il s'agit de la quantité d'octets qui sera lue à partir du tampon d'entrée et également écrite dans le tampon de sortie.

Valeur de retour

Aucune.

Voir aussi

StartAESCipher() , FinishCipher() , AESDecoder() , AESEncoder()

OS Supportés

Tous

83.2 AESEncoder

Syntaxe

```
Resultat =
    AESEncoder(*Entree,
    *Sortie, Taille, *Cle,
    Bits,
    *VecteurInitialisation [,
    Mode])
```

Description

Encode les données du tampon d'entrée en utilisant l'algorithme AES dans le tampon de sortie.

Arguments

***Entree** Le tampon d'entrée avec les données en clair.

***Sortie** Le tampon de sortie qui reçoit les données codées.
Il doit être différent du tampon d'entrée.

Taille Le nombre d'octets à encoder.
Au moins 16 octets sinon des données de rembourrage doivent être ajoutées dans le tampon d'entrée avant le codage, afin d'obtenir ces 16 octets minimum.

***Cle** Un tampon contenant la clé de codage.

```
Sa taille dépend du
 paramètre 'Bits':
  16 octets pour le
 cryptage 128 bits
  24 octets pour le
 cryptage 192 bits
  32 octets pour le
 cryptage 256 bits
```

Bits La taille de la clé utilisée par le chiffrement.
Les valeurs valides sont 128, 192 et 256.

***VecteurInitialisation** Est un bloc tampon de données aléatoires de 16 octets, utilisé pour initialiser le chiffrement afin d'empêcher toute violation de décodage (uniquement nécessaire si vous utilisez le mode #PB_Cipher_CBC).

Mode (optionnel)

#PB_Cipher_CBC: Mode par défaut (Cipher Block Chaining). Besoin de '*VecteurInitialisation'. Recommandé car plus sûr que le mode ECB.

#PB_Cipher_ECB: Mode Alternatif (Electronic CodeBook). Pas besoin de '*VecteurInitialisation' ni de chaînage(chaque bloc est codé indépendamment). Cryptage très faible comparé à CBC et ne devrait pas être utilisé pour un cryptage sérieux.

Valeur de retour

Renvoie une valeur non nulle si l'encodage a réussie, zéro sinon.

Remarques

AES est un algorithme de chiffrement de l'industrie qui a un bon équilibre entre la vitesse et la sécurité. Voici l'introduction wikipedia sur AES : "En cryptographie, le Advanced Encryption Standard (AES) est une norme de cryptage adopté par le gouvernement des États-Unis. La norme comprend trois chiffrements par bloc, AES-128, AES-192 et AES-256. Chaque chiffrement AES a une taille de bloc de 128 bits, avec des tailles de clés de 128, 192 et 256 bits, respectivement. Les chiffrements AES ont été analysés de façon approfondie et sont maintenant utilisés dans le monde entier.
PureBasic utilise une mise en uvre conforme à la RFC de l'AES. Plus d'informations peuvent être trouvées dans le RFC 3602 : http ://www.ietf.org/rfc/rfc3602.txt.

Exemple : CBC

```
1  ; Crypter une chaîne de
     caractères
2  ;
3  String$ = "Hello voici un
     test pour AES"
4
5  StringMemorySize =
     StringByteLength(String$)
```

```
    + SizeOf(Character) ;
    Espace pour la chaîne et
    son caractère 'nul' de
    terminaison de chaîne
6   *CipheredString =
    AllocateMemory(StringMemorySize)
7   *DecipheredString =
    AllocateMemory(StringMemorySize)
8
9   If AESEncoder(@String$,
    *CipheredString,
    StringByteLength(String$),
    ?Key, 128,
    ?InitializationVector)
10    Debug "Codé :
    "+PeekS(*CipheredString) ;
    Attention, ça s'arrêtera
    sur le premier octet nul,
    uniquement à des fins de
    démonstration
11
12    AESDecoder(*CipheredString,
    *DecipheredString,
    StringByteLength(String$),
    ?Key, 128,
    ?InitializationVector)
13    Debug "Décodé :
    "+PeekS(*DecipheredString)
14  EndIf
15
16  DataSection
17    Key:
18      Data.b $06, $a9, $21,
    $40, $36, $b8, $a1, $5b,
    $51, $2e, $03, $d5, $34,
    $12, $00, $06
19
20    InitializationVector:
21      Data.b $3d, $af, $ba,
    $42, $9d, $9e, $b4, $30,
    $b4, $22, $da, $80, $2c,
    $9f, $ac, $41
22  EndDataSection
```

Exemple : ECB

```
1   String$ = "Hello voici un
    test pour AES"
2   Caractere.c = '0'
3   *CipheredString   =
    AllocateMemory(StringByteLength(String$)
    + SizeOf(Caractere)) ;
    Espace pour la chaîne de
    caractères avec son
4   *DecipheredString =
    AllocateMemory(StringByteLength(String$)
    + SizeOf(Caractere)) ;
    caractère null de fin de
```

```
    chaîne
5
6   If AESEncoder(@String$,
     *CipheredString,
     StringByteLength(String$)
     + SizeOf(Caractere), ?Key,
     128, 0, #PB_Cipher_ECB)
7     Debug "Codé : " +
     PeekS(*CipheredString)
8
9     AESDecoder(*CipheredString,
     *DecipheredString,
     StringByteLength(String$)
     + SizeOf(Caractere), ?Key,
     128, 0, #PB_Cipher_ECB)
10    Debug "Décodé : " +
     PeekS(*DecipheredString)
11  EndIf
12
13  DataSection
14    Key:
15      Data.b $06, $a9, $21,
     $40, $36, $b8, $a1, $5b,
     $51, $2e, $03, $d5, $34,
     $12, $00, $06
16  EndDataSection
```

Voir aussi

AESDecoder() , StartAESCipher()

OS Supportés

Tous

83.3 AESDecoder

Syntaxe

```
Resultat =
    AESDecoder(*Entrée,
    *Sortie, Taille, *Cle,
    Bits,
    *VecteurInitialisation [,
    Mode])
```

Description

Décode les données du tampon d'entrée en utilisant l'algorithme AES dans le tampon de sortie.

Arguments

***Entree** Le tampon d'entrée avec les données en clair.

***Sortie** Le tampon de sortie qui reçoit les données codées.
Il doit être différent du tampon d'entrée.

Taille Le nombre d'octets à encoder.
Au moins 16 octets sinon des données de rembourrage doivent être ajoutées dans le tampon d'entrée avant le codage, afin d'obtenir ces 16 octets minimum.

***Cle** Un tampon contenant la clé de codage.

```
Sa taille dépend du
 paramètre 'Bits':
  16 octets pour le
cryptage 128 bits
  24 octets pour le
cryptage 192 bits
  32 octets pour le
cryptage 256 bits
```

Bits La taille de la clé utilisée par le chiffrement.
Les valeurs valides sont 128, 192 et 256.

***VecteurInitialisation** Est un bloc tampon de données aléatoires, utilisé pour initialiser le chiffrement afin d'empêcher toute violation de décodage (uniquement nécessaire si vous utilisez le mode #PB_Cipher_CBC). Sa taille vaut toujours 16 octets. Le contenu de ce bloc doit correspondre à celui qui a été utilisé pour encoder les données.

Mode (optionnel)

```
     #PB_Cipher_CBC: Mode
   par défaut (Cipher Block
   Chaining). Besoin de
   '*VecteurInitialisation'.
                 Recommandé
   car plus sûr que le mode
   ECB.
  #PB_Cipher_ECB: Mode
   Alternatif (Electronic
   CodeBook). Pas besoin de
   '*VecteurInitialisation'
                 ni de
   chaînage(chaque bloc est
   codé indépendamment).
   Cryptage très faible
   comparé à CBC
                 et ne
   devrait pas être utilisé
   pour un cryptage sérieux.
```

Valeur de retour

Renvoie une valeur non nulle si le décodage a réussi, zéro sinon.

Remarques

Pour plus d'informations à propos de AES et voir des exemples : AESEncoder() .

Voir aussi

AESEncoder() , StartAESCipher()

OS Supportés

Tous

83.4 DESFingerprint

Syntaxe

```
Resultat\$ =
    DESFingerprint(MotDePasse$,
    Cle$)
```

Description

Renvoie une version cryptée d'un mot de passe, en utilisant le cryptage DES.

Arguments

MotDePasse$ Le mot de passe à crypter. Il peut contenir jusqu'à 8 caractères au maximum, les autres caractéres sont tout simplement ignorés.
StringFingerprint() peut être utilisé pour traiter un tampon plus grand.

Cle$ Clé de cryptage.

Elle est est également appelée paramètre 'Salt' (grain de sel), bien connu des utilisateurs de Linux/Unix/BSD. Lorsque l'on utilise une clé de 2 caractères, cette fonction renvoie une chaîne 'Salt2', compatible avec tout mot de passe standard Linux (/etc/passwd). Cette commande est basée sur la fonction open source 'crypt()'.

Valeur de retour

Renvoie le mot de passe crypté.

Remarques

Cet algorithme est basé sur la méthode de chiffrement DES (Data Encryption Standard) pour générer une chaîne de 13 caractères. Cette chaîne est unique et non réversible entrainant une grande difficulté pour la 'cracker' lorsque le mot de passe est correctement choisi.

Exemple

```
Debug DESFingerprint("Mot de passe", "Key007")
Debug DESFingerprint("Nouveau mot de passe", "Key007")
```

Voir aussi

StringFingerprint() , Fingerprint()

OS Supportés

Tous

83.5 StartFingerprint

Syntaxe

```
Resultat = StartFingerprint(#Fingerprint, Plugin [, Bits])
```

Description

Initialise le calcul d'une empreinte en plusieurs étapes.
Contrairement à la commande Fingerprint() , elle permet de calculer l'empreinte de grandes quantités de données sans avoir à charger le tout dans un tampon de mémoire continue.

Arguments

#Fingerprint L'identifiant de ce calcul de type 'checksum'.
#PB_Any peut est utilisé pour générer automatiquement ce nombre.

Plugin Le plugin à utiliser. Peut être l'une des valeurs suivantes :

```
#PB_Cipher_CRC32: utilise l'algorithme CRC32. UseCRC32Fingerprint() doit être appelé avant pour utiliser le plugin.
#PB_Cipher_MD5  : utilise l'algorithme MD5. UseMD5Fingerprint() doit être appelé avant pour utiliser le plugin.
#PB_Cipher_SHA1 : utilise l'algorithme SHA1. UseSHA1Fingerprint() doit être appelé avant pour utiliser le plugin.
```

```
#PB_Cipher_SHA2 : utilise
  l'algorithme SHA2.
  UseSHA2Fingerprint()
doit être appelé avant pour
  utiliser le plugin.
#PB_Cipher_SHA3 : utilise
  l'algorithme SHA3.
  UseSHA3Fingerprint()
doit être appelé avant pour
  utiliser le plugin.
```

Bits (optionnel) Le nombre de bits à utiliser pour l'empreinte. Supporté seulement avec les plugins suivants :

```
#PB_Cipher_SHA2 : peut
  être 224, 256 (par
  defaut), 384 ou 512.
#PB_Cipher_SHA3 : peut
  être 224, 256 (par
  defaut), 384 or 512.
```

Valeur de retour

Renvoie la valeur #FingerPrint si #PB_Any a été utilisé pour ce paramètre.

Remarques

AddFingerprintBuffer() permet d'ajouter des blocs de données au calcul et FinishFingerprint() le terminera et renverra le résultat.

Exemple

```
1  UseMD5Fingerprint()
2
3  *Buffer =
     AllocateMemory(200) ;
     Prépare un tampon de
     données
4  If *Buffer
5    PokeS(*Buffer, "Le renard
     brun rapide saute sur le
     chien paresseux.", -1,
     #PB_Ascii)
6    Taille =
     MemoryStringLength(*Buffer,
     #PB_Ascii)
7
8    If StartFingerprint(0,
     #PB_Cipher_MD5)
                  ; démarre
     le calcul
9      AddFingerprintBuffer(0,
     *Buffer, Taille/2)
             ; calcule la
     partie 1
```

```
10      AddFingerprintBuffer(0,
    *Buffer+Taille/2,
    Taille/2) ; calcule la
    partie 2
11
12      MD5$ =
    FinishFingerprint(0)
                    ; termine
    le calcul
13      Debug "MD5 checksum = "
    + MD5$
14
15      MD5$ =
    Fingerprint(*Buffer,
    Length, #PB_Cipher_MD5)
          ; comparaison avec
    le calcul en une seule
    étape
16      Debug "MD5 checksum = "
    + MD5$
17    EndIf
18
19    FreeMemory(*Buffer)
20  EndIf
```

Voir aussi

Fingerprint() , FileFingerprint() , StringFingerprint()

OS Supportés

Tous

83.6 FinishCipher

Syntaxe

```
FinishCipher(#Cipher)
```

Description

Met fin au flux de cryptage préalablement ouvert avec StartAESCipher() .

Arguments

#**Cipher** Le cipher à fermer.

Valeur de retour

Aucune.

Remarques

Cette commande devrait être appelée pour clore un calcul de chiffrement même si le

chiffre n'est plus nécessaire car elle effacera toutes les données liées au calcul de chiffrement.

Voir aussi

StartAESCipher() , AddCipherBuffer()

OS Supportés

Tous

83.7 IsCipher

Syntaxe

```
Resultat = IsCipher(#Cipher)
```

Description

Teste si le numéro #Cipher donné est un chiffre valide et correctement initialisé.

Arguments

#Cipher Le cipher à utiliser.

Valeur de retour

Renvoie une valeur non nulle si #Cipher est un chiffre valide, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'un chiffre est correctement initialisé.

Voir aussi

StartAESCipher()

OS Supportés

Tous

83.8 AddFingerprintBuffer

Syntaxe

```
AddFingerprintBuffer(#Fingerprint,
   *Tampon, Taille)
```

Description

Ajoute un nouveau tampon mémoire dans le calcul d'une somme de contrôle commencé par StartFingerprint() .
La somme de contrôle renvoyée à la fin du calcul inclura tous les tampons ajoutés comme si la somme de contrôle avait été calculée avec chacun d'eux dans un seul tampon mémoire contigu.

Arguments

#FingerPrint L'identifiant du calcul en cours.

***Tampon** Le tampon qui contient les nouvelles données à ajouter à l'empreinte.

Taille Le nombre d'octets à ajouter à l'empreinte.

Valeur de retour

Aucune.

Remarques

Voir StartFingerprint() pour un exemple d'utilisation et plus d'informations.

Voir aussi

StartFingerprint() , FinishFingerprint()

OS Supportés

Tous

83.9 FinishFingerprint

Syntaxe

```
Resultat\$ =
    FinishFingerprint(#Fingerprint)
```

Description

Termine le calcul d'empreinte commencé avec StartFingerprint() et renvoie la signature générée dans une chaîne de caractères hexadécimale.

Arguments

#FingerPrint L'identifiant du calcul à terminer.

Valeur de retour

Renvoie l'empreinte sous forme d'une chaîne de caractères hexadécimale.

Remarques

Cette commande doit toujours être appelée pour terminer un calcul, même si la signature n'est plus nécessaire, car elle libère toutes les données associées au calcul. Voir StartFingerprint() pour un exemple d'utilisation et plus d'informations.

Voir aussi

StartFingerprint() , AddFingerprintBuffer()

OS Supportés

Tous

83.10 IsFingerprint

Syntaxe

```
Resultat =
    IsFingerprint(#Fingerprint)
```

Description

Teste si un '#Fingerprint' est bien un calcul d'empreinte valide commencé par StartFingerprint() .

Arguments

#FingerPrint L'empreinte à tester.

Valeur de retour

Renvoie une valeur non nulle si l'empreinte est valide, zéro sinon.

Remarques

Cette fonction a été conçue pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.

Voir aussi

StartFingerprint()

OS Supportés

Tous

83.11 FileFingerprint

Syntaxe

```
Resultat\$ =
    FileFingerprint(Fichier$,
    Plugin [, Bits  [,
    Decalage [, Longueur]]))
```

Description

Renvoie une empreinte d'un fichier.

Arguments

Fichier$ Le nom du fichier à utiliser.

Plugin Le plugin à utiliser. Peut être l'une des valeurs suivantes :

```
#PB_Cipher_CRC32: utilise
  l'algorithme CRC32.
  UseCRC32Fingerprint()
doit être appelé avant pour
  utiliser le plugin.
#PB_Cipher_MD5  : utilise
  l'algorithme MD5.
  UseMD5Fingerprint()
doit être appelé avant pour
  utiliser le plugin.
#PB_Cipher_SHA1 : utilise
  l'algorithme SHA1.
  UseSHA1Fingerprint()
doit être appelé avant pour
  utiliser le plugin.
#PB_Cipher_SHA2 : utilise
  l'algorithme SHA2.
  UseSHA2Fingerprint()
doit être appelé avant pour
  utiliser le plugin.
#PB_Cipher_SHA3 : utilise
  l'algorithme SHA3.
  UseSHA3Fingerprint()
doit être appelé avant pour
  utiliser le plugin.
```

Bits (optionnel) Le nombre de bits à utiliser pour l'empreinte. Supporté seulement avec les plugins suivants :

```
#PB_Cipher_SHA2 : peut
  être 224, 256 (par
  defaut), 384 ou 512.
#PB_Cipher_SHA3 : peut
  être 224, 256 (par
  defaut), 384 or 512.
```

Decalage (optionnel) Le décalage en octets (offset), à partir du début du fichier avant de lancer le calcul de la somme de contrôle.

Longueur (optionnel) La longueur (en octets) à utiliser pour le calcul de la somme de contrôle.

Valeur de retour

Renvoie l'empreinte si le calcul a été un succès.
Si le fichier est introuvable ou qu'une erreur s'est produite, le résultat sera une chaîne de caractères vide.

Voir aussi

Fingerprint() , StartFingerprint() , StringFingerprint()

OS Supportés

Tous

83.12 Fingerprint

Syntaxe

```
Resultat\$ =
    Fingerprint(*Tampon,
    Taille, Plugin [, Bits])
```

Description

Renvoie l'empreinte des données contenues dans un tampon.

Arguments

***Tampon** Le tampon contenant les données.

Taille La taille du tampon.

Plugin Le plugin à utiliser. Peut être l'une des valeurs suivantes :

```
    #PB_Cipher_CRC32: utilise
      l'algorithme CRC32.
      UseCRC32Fingerprint()
  doit être appelé avant pour
      utiliser le plugin.
    #PB_Cipher_MD5  : utilise
      l'algorithme MD5.
      UseMD5Fingerprint()
  doit être appelé avant pour
      utiliser le plugin.
    #PB_Cipher_SHA1 : utilise
      l'algorithme SHA1.
      UseSHA1Fingerprint()
  doit être appelé avant pour
      utiliser le plugin.
```

```
#PB_Cipher_SHA2 : utilise
  l'algorithme SHA2.
  UseSHA2Fingerprint()
doit être appelé avant pour
  utiliser le plugin.
#PB_Cipher_SHA3 : utilise
  l'algorithme SHA3.
  UseSHA3Fingerprint()
doit être appelé avant pour
  utiliser le plugin.
```

Bits (optionnel) Le nombre de bits à utiliser pour l'empreinte. Supporté seulement avec les plugins suivants :

```
#PB_Cipher_SHA2 : peut
  être 224, 256 (par
  defaut), 384 ou 512.
#PB_Cipher_SHA3 : peut
  être 224, 256 (par
  defaut), 384 or 512.
```

Valeur de retour

Renvoie l'empreinte sous forme d'une chaîne de caractères hexadécimale.

Exemple : (Une chaîne en tant que tampon mémoire)

```
1  UseMD5Fingerprint()
2  test.s = "Ceci est un test
     avec une chaîne!"
3  Debug Fingerprint(@test,
     Len(test), #PB_Cipher_MD5)
```

Exemple : (Avec un tampon mémoire)

```
1  UseMD5Fingerprint()
2
3  *Buffer =
     AllocateMemory(500)
4  If *Buffer
5    PokeS(*Buffer, "Le renard
     brun rapide saute sur le
     chien paresseux.", -1,
     #PB_Ascii)
6    MD5$ =
     Fingerprint(*Buffer,
     MemoryStringLength(*Buffer,
     #PB_Ascii), #PB_Cipher_MD5)
7    Debug "MD5 Fingerprint =
     " + MD5$
8    FreeMemory(*Buffer)  ;
     sera libéré aussi à la fin
     du programme
     automatiquement
```

```
9   EndIf
```

Voir aussi

FileFingerprint() , StartFingerprint() , StringFingerprint()

OS Supportés

Tous

83.13 StringFingerprint

Syntaxe

```
Resultat\$ =
    StringFingerprint(Texte$,
    Plugin [, Bits [, Format]])
```

Description

Renvoie l'empreinte d'une chaîne de caractères.

Arguments

Texte$ La chaîne de caractères.

Plugin Le plugin à utiliser. Peut être l'une des valeurs suivantes :

```
#PB_Cipher_CRC32: utilise
  l'algorithme CRC32.
  UseCRC32Fingerprint()
doit être appelé avant pour
  utiliser le plugin.
#PB_Cipher_MD5  : utilise
  l'algorithme MD5.
  UseMD5Fingerprint()
doit être appelé avant pour
  utiliser le plugin.
#PB_Cipher_SHA1 : utilise
  l'algorithme SHA1.
  UseSHA1Fingerprint()
doit être appelé avant pour
  utiliser le plugin.
#PB_Cipher_SHA2 : utilise
  l'algorithme SHA2.
  UseSHA2Fingerprint()
doit être appelé avant pour
  utiliser le plugin.
#PB_Cipher_SHA3 : utilise
  l'algorithme SHA3.
  UseSHA3Fingerprint()
doit être appelé avant pour
  utiliser le plugin.
```

Bits (optionnel) Le nombre de bits à utiliser pour l'empreinte. Supporté seulement avec les plugins suivants :

```
#PB_Cipher_SHA2 : peut
 être 224, 256 (par
 defaut), 384 ou 512.
#PB_Cipher_SHA3 : peut
 être 224, 256 (par
 defaut), 384 or 512.
```

Format (optionnel) Le format de la chaîne de caractères à utiliser avant le 'hashage'.
Peut être l'une des valeurs suivantes :

```
#PB_UTF8    : la chaîne
 sera 'hashée' au format
 UTF8 (par défaut).
#PB_Ascii   : la chaîne
 sera 'hashée' au format
 ASCII.
#PB_Unicode : la chaîne
 sera 'hashée' au format
 Unicode (UTF16).
```

Valeur de retour

Renvoie l'empreinte sous forme d'une chaîne de caractères hexadécimale.

Exemple

```
1  UseMD5Fingerprint()
2
3  Debug
    StringFingerprint("UnMotdePasse",
    #PB_Cipher_MD5)
```

Voir aussi

FileFingerprint() , StartFingerprint() , Fingerprint()

OS Supportés

Tous

83.14 UseMD5Fingerprint

Syntaxe

```
UseMD5Fingerprint()
```

Description

Déclare le plugin de prise d'empreinte MD5 pour une utilisation future.

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Voici une explication rapide tireé de la RFC 1321 sur MD5 :
L'algorithme prend en entrée un message de longueur arbitraire et produit en sortie une "empreinte digitale" 128 bits ou un "Message Digest" de l'entrée. On suppose qu'il est mathématiquement impossible de produire deux messages ayant la même empreinte, ou de produire un message ayant un message digest cible pré-spécifié. L'algorithme MD5 est destiné à la production de signatures numériques.'
MD5 est souvent utilisé pour le cryptage des mots de passe mais il doit être évité car il est a été jugé vulnérable à certaines attaques. Plus d'informations peuvent être trouvées dans la RFC 1321 :
http ://www.ietf.org/rfc/rfc1321.txt.

Exemple

```
1  UseMD5Fingerprint()
2
3  Debug
     StringFingerprint("UnMotdePasse",
     #PB_Cipher_MD5)
```

Voir aussi

UseSHA1Fingerprint() (),
UseSHA2Fingerprint() (),
UseSHA3Fingerprint() (),
UseCRC32Fingerprint() ()

OS Supportés

Tous

83.15 UseSHA1Fingerprint

Syntaxe

```
UseSHA1Fingerprint()
```

Description

Déclare le plugin de prise d'empreinte SHA1 pour une utilisation future.

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

SHA1 peut être utilisé pour calculer une somme de contrôle pour vérifier si un message n'a pas été modifié. contrairement à CRC32 , il est presque impossible de modifier le message d'origine et de produire la même empreinte SHA1.
Voici une explication rapide tiré de la RFC 3174 sur SHA1 :
'La SHA-1 est dite sécurisée car il est mathématiquement impossible de trouver un message digest qui correspond à un message donné, ou de trouver deux messages différents qui produisent le même message digest. Toute modification d'un message en transit donnera, avec une très forte probabilité, une autre empreinte, et la signature sera différente. "
Plus d'informations peuvent être trouvées dans la RFC 3174 :
http ://www.ietf.org/rfc/rfc3174.txt.

Exemple

```
1   UseSHA1Fingerprint()
2
3   Debug
      StringFingerprint("UnMotdePasse",
      #PB_Cipher_SHA1)
```

Voir aussi

UseMD5Fingerprint() (),
UseSHA2Fingerprint() (),
UseSHA3Fingerprint() (),
UseCRC32Fingerprint() ()

OS Supportés

Tous

83.16 UseSHA2Fingerprint

Syntaxe

```
UseSHA2Fingerprint()
```

Description

Déclare le plugin de prise d'empreinte SHA2 pour une utilisation future.

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Sur Wikipedia : SHA-2 comprend des changements importants par rapport à son prédécesseur, SHA-1. En 2005, un algorithme a émergé qui a trouvé des collisions SHA-1 avec environ 2000 fois moins d'étapes que prévu. Bien qu'(à partir de 2015) aucun exemple de collision SHA-1 n'aie encore été publié, la marge de sécurité laissée par SHA-1 est plus faible que prévu, et son utilisation n'est donc plus recommandée pour les applications qui réclament une forte résistance aux collisions, notamment concernant les signatures numériques. Bien que SHA-2 présente une certaine similitude avec l'algorithme SHA-1, ces attaques ont pas été étendues avec succès à SHA-2.

Exemple

```
1  UseSHA2Fingerprint()
2
3  Debug
     StringFingerprint("UnMotdePasse",
     #PB_Cipher_SHA2, 512) ;
     Use SHA2-512 variant
```

Voir aussi

UseMD5Fingerprint() (),
UseSHA1Fingerprint() (),
UseSHA3Fingerprint() (),
UseCRC32Fingerprint() ()

OS Supportés

Tous

83.17 UseSHA3Fingerprint

Syntaxe

```
UseSHA3Fingerprint()
```

Description

Déclare le plugin de prise d'empreinte SHA3 pour une utilisation future. Les

standards 224-bits, 256 bits, 384 bits et 512 bits sont pris en charge.

Arguments

Aucun.

Valeur de retour

Aucune.

Exemple

```
1  UseSHA3Fingerprint()
2
3  Debug
     StringFingerprint("UnMotdePasse",
     #PB_Cipher_SHA3, 512) ;
     Utilisation de SHA3-512
```

Voir aussi

UseMD5Fingerprint() (),
UseSHA1Fingerprint() (),
UseSHA2Fingerprint() (),
UseCRC32Fingerprint() ()

OS Supportés

Tous

83.18 UseCRC32Fingerprint

Syntaxe

```
UseCRC32Fingerprint()
```

Description

Déclare le plugin de prise d'empreinte CRC32 pour une utilisation future.

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

CRC32 est un codage 32 bits qui n'est pas conçu pour le stockage de mot de passe car il est facilement piratable, mais utilisable pour des contrôles rapides d'intégrité des données. Par exemple les fichiers ZIP ont une somme de contrôle à la fin du fichier pour vérifier qu'il n'est pas corrompu.
Le principal avantage de l'algorithme CRC32 sur le hachage MD5 ou autre est sa très grande vitesse.

Exemple

```
1  UseCRC32Fingerprint()
2
3  Debug StringFingerprint("du
     texte", #PB_Cipher_CRC32)
```

Voir aussi

UseMD5Fingerprint() (),
UseSHA1Fingerprint() (),
UseSHA2Fingerprint() (),
UseSHA3Fingerprint() ()

OS Supportés

Tous

83.19 Base64DecoderBuffer

Syntaxe

```
Resultat =
    Base64DecoderBuffer(*TamponSource,
    LongueurSource,
    *TamponDestination,
    LongueurDestination)
```

Description

Décode un tampon codé en Base64 .

Arguments

***TamponSource** Le tampon contenant les données codées.

LongueurSource La taille de la mémoire tampon d'entrée.

***TamponDestination** Le tampon de sortie où les données seront copiées.

LongueurDestination La taille de la mémoire tampon de sortie.

La mémoire tampon de sortie peut être jusqu'à 33% plus petite que le tampon d'entrée, avec une taille minimale de 64 octets.
Il est recommandé de faire un tampon légèrement plus grand, par exemple 30%, pour éviter les débordements.

Valeur de retour

Renvoie la longueur des données décodées en octets.

Exemple : Encodage & Décodage d'une variable texte

```
1  Exemple$ = "Voici une
    phrase !"
2  Decoded$ = Space(1024)
3  Encoded$ = Space(1024)
4
5  Debug
    Base64EncoderBuffer(@Exemple$,
    StringByteLength(Exemple$),
    @Encoded$,
    StringByteLength(Encoded$))
6  Debug Encoded$
7
8  Debug
    Base64DecoderBuffer(@Encoded$,
    StringByteLength(Encoded$),
    @Decoded$,
    StringByteLength(Decoded$))
9  Debug Decoded$
```

Exemple : Encodage & Décodage d'une DataSection

```
1  DataSection
2    TestDebut:
3    Data.a $00, $01, $02,
    $03, $04, $05, $06, $07
4    Data.a $08, $09, $0A,
    $0B, $0C, $0D, $0E, $0F
5    TestFin:
6  EndDataSection
7
8  Taille = (?TestFin -
    ?TestDebut) * 1.35
9  If Taille < 64
10 Taille = 64
11 EndIf
12
13 *EncodeBuffer =
    AllocateMemory(Taille)
```

```
14  Taille =
     Base64EncoderBuffer(?TestDebut,
     ?TestFin - ?TestDebut,
     *EncodeBuffer,
     MemorySize(*EncodeBuffer))
15  ChaineEncodee$ =
     PeekS(*EncodeBuffer,
     Taille, #PB_Ascii)
16  Debug ChaineEncodee$
17
18  *DecodeBuffer =
     AllocateMemory(Taille)
19  Taille =
     PokeS(*EncodeBuffer,
     ChaineEncodee$,
     StringByteLength(ChaineEncodee$,
     #PB_Ascii),
     #PB_Ascii|#PB_String_NoZero)
20  Taille =
     Base64DecoderBuffer(*EncodeBuffer,
     Taille, *DecodeBuffer,
     MemorySize(*DecodeBuffer))
21  ShowMemoryViewer(*DecodeBuffer,
     Taille)
```

Exemple : Problème de décodage d'une variable texte déjà codé en Base64 par un logiciel externe

```
1   ; Attention, il est délicat
     d'utiliser une variable
     pour décoder un texte
2   ; déjà codé en Base64 par
     un logiciel externe
     (messagerie par exemple)
3
4   Encoded$ =
     "Vm9pY2kgdW5lIHBocmFzZSAh"
     ;"Voici une phrase !",
     codée en Base64 utf8
5   Decoded$ = Space(1024)
6
7   Debug
     Base64DecoderBuffer(@Encoded$,
     StringByteLength(Encoded$),
     @Decoded$, 1024)
8   Debug Decoded$ ; ==>
     Affiche n'importe quoi
9
10  ; Pour résoudre ce
     problème, suivre la
     procédure ci-dessous
11
12  ; Différents exemples de
     textes déjà codés
13  Encode_utf8$      =
     "Vm9pY2kgdW5lIHBocmFzZSBow6kgIQ0K"
```

```
    ;Voici une phrase hé !,
    codée en Base64 utf8
14  Encode_iso8859_1$ =
    "Vm9pY2kgdW5lIHBocmFzZSBo6SAhDQo="
    ;Voici une phrase hé !,
    codée en Base64 iso8859_1$
15  Encode_ascii$     =
    "Vm9pY2kgdW5lIHBocmFzZSBoICENCg=="
    ;Voici une phrase hé !,
    codée en Base64 ascii
16
17  Procedure.s
    Decode64(Texte64.s,
    Option=#PB_UTF8)
18   Protected *in, *out
19   Protected Resultat.s
20
21   *in = Ascii(Texte64) ; Le
    texte codé en Base64 est
    supposé être en ascii pour
    PureBasic
22   If *in <> 0
23     ; La mémoire tampon de
    sortie peut être jusqu'à
    33% plus petite que le
    tampon d'entrée
24     *out =
    AllocateMemory(MemoryStringLength(*in,#PB_Ascii)
    * 0.8) ; Ici nous prenons
    une valeur de 20%
25     If *out <> 0
26       Base64DecoderBuffer(*in,
    MemorySize(*in), *out,
    MemorySize(*out))  ;
    Décodage...
27       Resultat =
    PeekS(*out, -1, Option) ;
    Le résultat est transféré
    dans une variable
28       FreeMemory(*out)
29     EndIf
30     FreeMemory(*in)
31   EndIf
32   ProcedureReturn Resultat
33  EndProcedure
34
35  Debug
    Decode64(Encode_utf8$)
                    ; Affiche:
    Voici une phrase hé !
36  Debug
    Decode64(Encode_iso8859_1$,
    #PB_Ascii) ; Affiche:
    Voici une phrase hé !
    (iso8859_1 est aussi
    appelée Latin-1 ou Europe
    occidentale et contient
    tous les caractères ascii)
```

```
37  Debug
      Decode64(Encode_ascii$,
      #PB_Ascii)      ; Affiche:
      Voici une phrase h !
      (problème avec lettres
      accentuées...)
38                                                          ;
      car le codage Base64 ascii
      s'arrête au caractère
      ascii n°127
```

Voir aussi

Base64EncoderBuffer()

OS Supportés

Tous

83.20 Base64EncoderBuffer

Syntaxe

```
Resultat =
    Base64EncoderBuffer(*TamponSource,
    LongueurSource,
    *TamponDestination,
    LongueurDestination [,
    Options])
```

Description

Encode le contenu d'un tampon avec l'algorithme Base64.

Arguments

***TamponSource** Le tampon contenant les données à coder.

LongueurSource La taille de la mémoire tampon d'entrée.

***TamponDestination** Le tampon de sortie où les données codées seront copiées.

LongueurDestination La taille de la mémoire tampon de sortie.

La mémoire tampon de sortie peut être jusqu'à 33% plus petite que le tampon d'entrée, avec une taille minimale de 64 octets.
Il est recommandé de faire un tampon légèrement plus grand, par exemple 30%, pour éviter les débordements.

Options (optionnel) Peut être une combinaison de :

```
#PB_Cipher_NoPadding: Il
 ne sera pas plus insérer
 de '=' à la fin de la
 mémoire tampon.
#PB_Cipher_URL       :
 Utilisation d'un codage
 légèrement différent,
 principalement utilisé
                        dans
 les URLs. Les caractères
 '+' et '/' seront
 respectivement codés
                        en
 '-' et '_'.
```

Valeur de retour

Renvoie la longueur des données codées en octets.

Remarques

Il est largement utilisé dans les programmes de messagerie mais peut également être utile dans toute application nécessitant un encodage basé sur le code ASCII seul (7 bits, caractères 32 à 127) pour les fichiers binaires.

Exemple

```
1  Exemple$ = "Ceci est une
     chaîne !"
2  Decoded$ = Space(1024)
3  Encoded$ = Space(1024)
4
5  Debug
     Base64EncoderBuffer(@Exemple$,
     StringByteLength(Exemple$),
     @Encoded$,
     StringByteLength(Encoded$))
6  Debug Encoded$
7
8  Debug
     Base64DecoderBuffer(@Encoded$,
     StringByteLength(Encoded$),
     @Decoded$,
     StringByteLength(Decoded$))
9  Debug Decoded$
```

Exemple : Encodage & Décodage d'une DataSection

```
1  DataSection
2    TestDebut:
3    Data.a $00, $01, $02,
   $03, $04, $05, $06, $07
```

```
4     Data.a $08, $09, $0A,
      $0B, $0C, $0D, $0E, $0F
5     TestFin:
6   EndDataSection
7
8   Taille = (?TestFin -
      ?TestDebut) * 1.35
9   If Taille < 64
10  Taille = 64
11  EndIf
12
13  *EncodeBuffer =
      AllocateMemory(Taille)
14  Taille =
      Base64EncoderBuffer(?TestDebut,
      ?TestFin - ?TestDebut,
      *EncodeBuffer,
      MemorySize(*EncodeBuffer))
15  ChaineEncodee$ =
      PeekS(*EncodeBuffer,
      Taille, #PB_Ascii)
16  Debug ChaineEncodee$
17
18  *DecodeBuffer =
      AllocateMemory(Taille)
19  Taille =
      PokeS(*EncodeBuffer,
      ChaineEncodee$,
      StringByteLength(ChaineEncodee$,
      #PB_Ascii),
      #PB_Ascii|#PB_String_NoZero)
20  Taille =
      Base64DecoderBuffer(*EncodeBuffer,
      Taille, *DecodeBuffer,
      MemorySize(*DecodeBuffer))
21  ShowMemoryViewer(*DecodeBuffer,
      Taille)
```

Voir aussi

Base64DecoderBuffer()

OS Supportés

Tous

83.21 Base64Decoder

Syntaxe

```
Resultat =
    Base64Decoder(Source$,
    *TamponDestination,
    LongueurDestination)
```

Description

Décode une chaîne de caractères encodée en Base64 .

Arguments

Source$ La chaîne de caractères à encoder.

***TamponDestination** Le tampon où les données brutes de sortie seront copiées.

LongueurDestination La taille du tampon de sortie.
Le tampon de sortie peut être jusqu'à 33% plus petit que le tampon d'entrée, avec une taille minimale de 64 octets.
Il est toutefois recommandé d'obtenir un tampon légèrement plus grand que 33%, comme 30% plus petit pour éviter les débordements.

Valeur de retour

Renvoie la longueur des données décodées en octets.

Exemple

```
1  *Texte = UTF8("Voici un
     texte !")
2
3  Encoder$ =
     Base64Encoder(*Texte,
     MemorySize(*Texte) - 1)
4  ;Encoder$ =
     Base64Encoder(*Texte,
     StringByteLength("Voici un
     texte !",#PB_UTF8))
5  Debug "Texte encodé: " +
     Encoder$
6
7  *TamponDecoder =
     AllocateMemory(1024)
8  Base64Decoder(Encoder$,
     *TamponDecoder, 1024)
9  Debug "Texte décodé: " +
     PeekS(*TamponDecoder, -1,
     #PB_UTF8)
```

Voir aussi

Base64Encoder()

OS Supportés

Tous

83.22 Base64Encoder

Syntaxe

```
Resultat\$ =
    Base64Encoder(*TamponSource,
    LongueurSource [, Options])
```

Description

Encode le tampon spécifié à l'aide de l'algorithme Base64. Ceci est largement utilisé dans les programmes de courrier électronique mais peut être utile pour tous les autres programmes qui ont besoin d'un codage ASCII uniquement (7 bits, seulement de 32 à 127 caractères) pour les fichiers binaires bruts.

Arguments

***TamponSource** Le tampon contenant les données brutes.

LongueurSource La taille du tampon d'entrée.

Options (optionnel) Peut être une combinaison des valeurs suivantes :

```
#PB_Cipher_NoPadding: Pas
 de '=' supplémentaire à
 la fin de la mémoire
 tampon pour la combler à
 3 octets.
#PB_Cipher_URL      :
 Codage légèrement
 différent,
 principalement utilisé
 dans les URLs. Les
 caractères '+' et '/'
 habituels
                      seront
 encodés respectivement
 en '-' et '_'
```

Valeur de retour

Renvoie les données codées sous forme de chaîne.

Exemple

```
1  *Texte = UTF8("Voici un
     texte !")
2
3  Encoder$ =
     Base64Encoder(*Texte,
     MemorySize(*Texte) - 1)
4  ;Encoder$ =
     Base64Encoder(*Texte,
     StringByteLength("Voici un
     texte !",#PB_UTF8))
5  Debug "Texte encodé: " +
     Encoder$
```

```
6
7   *TamponDecoder =
      AllocateMemory(1024)
8   Base64Decoder(Encoder$,
      *TamponDecoder, 1024)
9   Debug "Texte décodé: " +
      PeekS(*TamponDecoder, -1,
      #PB_UTF8)
```

Voir aussi

Base64Decoder() , Base64DecoderBuffer() , Base64EncoderBuffer()

OS Supportés

Tous

83.23 StartAESCipher

Syntaxe

```
Resultat =
    StartAESCipher(#Cipher,
    *Cle, Bits,
    *VecteurInitialisation,
    Mode)
```

Description

Initialise un flux de chiffrement AES.

Arguments

#Cipher L'identifiant du nouveau chiffrement.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

***Cle** Un tampon contenant la clé de décodage.

```
Sa taille dépend du
paramètre 'Bits':
  16 octets pour le
cryptage 128 bits.
  24 octets pour le
cryptage 196 bits.
  32 octets pour le
cryptage 256 bits.
```

Bits La taille de la clé utilisée par le chiffrement.
Les valeurs valides sont 128, 192 et 256.

***VecteurInitialisation** Est un bloc de données aléatoires, utilisé pour initialiser le chiffrement et pour empêcher toute violation de décodage (uniquement nécessaire si vous utilisez le mode #PB_Cipher_CBC).

```
    Sa taille dépend du
    paramètre 'Bits':
      16 octets pour le
    cryptage 128 bits.
      24 octets pour le
    cryptage 196 bits.
      32 octets pour le
    cryptage 256 bits.
```

Mode Peut être une combinaison de :

```
    #PB_Cipher_Decode: Le flux
     est utilisé pour décoder
     les données.
    #PB_Cipher_Encode: Le flux
     est utilisé pour encoder
     les données.
  avec
    #PB_Cipher_CBC: Mode par
     défaut (Cipher Block
     Chaining). Besoin de
     '*VecteurInitialisation'.
                    Recommandé
     car plus sûr que le mode
     ECB.
    #PB_Cipher_ECB: Mode
     alternatif(Electronic
     CodeBook). Pas besoin de
     '*VecteurInitialisation'
                     ni de
     chaînage(chaque bloc est
     codé indépendamment).
     Cryptage très faible
     comparé à CBC
                     et ne
     devrait pas être utilisé
     pour un cryptage sérieux.
```

Valeur de retour

Si #PB_Any a été utilisé pour le paramètre #Cipher alors un numéro généré automatiquement est retourné.

Remarques

Les nouveaux tampons à encoder ou à décoder peuvent être ajoutés avec AddCipherBuffer() .
Une fois le codage terminé, FinishCipher() doit être appelé.
Pour plus d'information à propos de AES, voir AESEncoder() .

Voir aussi

AddCipherBuffer() , FinishCipher() , AESEncoder() , AESDecoder()

OS Supportés

Tous

83.24 OpenCryptRandom

Syntaxe

```
Resultat = OpenCryptRandom()
```

Description

Initialise l'accès au générateur de nombres pseudo-aléatoires.

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle si le générateur est correctement initialisé, zéro s'il n'y a pas de générateur pseudo-aléatoire robuste disponible dans le système.

Remarques

Les commandes CryptRandom() et CryptRandomData() peuvent être utilisées pour lire les données disponibles.
Ce générateur est très robuste, assez pour être utilisé à des fins de cryptographie, comme la génération de clés pour la commande AESEncoder() . La source des données du générateur est "/dev/urandom" sous Linux et MacOS X, et la "Microsoft Cryptography API" sous Windows. Il est néanmoins bien plus lent que la fonction Random() classique.
Consulter CryptRandomData() pour un exemple d'utilisation.

Voir aussi

CryptRandom() , CryptRandomData() , CloseCryptRandom()

OS Supportés

Tous

83.25 CloseCryptRandom

Syntaxe

```
CloseCryptRandom()
```

Description

Libère les ressources utilisées par le générateur de nombres pseudo-aléatoires précédemment initialisé avec OpenCryptRandom() .

Arguments

Aucun.

Valeur de retour

Aucune.

Voir aussi

OpenCryptRandom() , CryptRandom() , CryptRandomData()

OS Supportés

Tous

83.26 CryptRandom

Syntaxe

```
Resultat =
    CryptRandom(Maximum)
```

Description

Renvoie un nombre aléatoire (entier) qui se trouve entre 0 (inclus) et la valeur maximale du générateur pseudo-aléatoire cryptographique.

Arguments

Maximum Valeur maximale renvoyée par la fonction.
Ne peut pas dépasser la valeur maximale d'un long : 2147483647.

Valeur de retour

Renvoie le nombre aléatoire généré.

Remarques

Le générateur doit d'abord être ouvert avec la commande OpenCryptRandom() .
Important : Utiliser une valeur 'Maximum' qui est égale à une puissance de deux peut causer des probabilités plus élevés pour certaines valeurs, favorisant les attaques statistiques. C'est la conséquence

de la division du nombre généré pour qu'il soit dans la plage spécifiée.
Pour générer des quantités plus importantes de nombres aléatoires, utilisez la commande CryptRandomData() . Pour une génération beaucoup plus rapide, mais moins robuste, utilisez la commande Random() .

Voir aussi

OpenCryptRandom() , CryptRandomData() , CloseCryptRandom() , Random()

OS Supportés

Tous

83.27 CryptRandomData

Syntaxe

```
Resultat =
    CryptRandomData(*Tampon,
    Longueur)
```

Description

Remplit une mémoire tampon avec des données aléatoires provenant du générateur pseudo-aléatoire cryptographique .

Arguments

***Tampon** Le tampon à remplir.

Longueur La taille de la mémoire tampon en octets.

Valeur de retour

Renvoie une valeur non nulle si les données aléatoires ont été générés avec succès, zéro sinon.

Remarques

Le générateur doit d'abord être ouvert avec la commande OpenCryptRandom() .
Pour une génération de données beaucoup plus rapide, mais moins robuste, utiliser la commande RandomData() .

Exemple

```
*Cle = AllocateMemory(16)

If OpenCryptRandom() And *Cle
  CryptRandomData(*Cle, 16)

  Texte$ = "Clé crée :"
  For i = 0 To 15
    Texte$ + " " + RSet(Hex(PeekB(*Cle+i), #PB_Byte), 2, "0")
  Next i

  CloseCryptRandom()
Else
  Texte$ = "La création de clé n'est pas disponible"
EndIf

MessageRequester("Exemple", Texte$)
```

Voir aussi

OpenCryptRandom() , CryptRandom() , CloseCryptRandom() RandomData()

OS Supportés

Tous

Chapitre 84

Clipboard

Généralités

Le presse-papier (clipboard) est le moyen standard pour partager des informations entre les applications lancées depuis le système d'exploitation. Par exemple lorsque l'on coupe un texte dans un éditeur, il est envoyé dans le presse-papier et peut ensuite être retrouvé et déposé dans un autre éditeur de texte. PureBasic permet de copier et récupérer facilement du texte et des images dans le presse-papier.

OS Supportés

Tous

84.1 ClearClipboard

Syntaxe

```
ClearClipboard()
```

Description

Efface le presse-papier.

Arguments

Aucun.

Valeur de retour

Aucune.

Exemple

```
1  ; On place un texte dans le
     presse-papier
2  SetClipboardText("Un
     exemple de texte")
```

```
3   MessageRequester("Presse-papier","Le
      presse papier contient le
      texte suivant : " +
      GetClipboardText())
4
5   ; Le presse-papier est
      effacé
6   ClearClipboard()
7   MessageRequester("Presse-papier","Le
      presse papier contient le
      texte suivant : " +
      GetClipboardText())
```

Voir aussi

SetClipboardText() , SetClipboardImage()

OS Supportés

Tous

84.2 GetClipboardImage

Syntaxe

```
Resultat =
    GetClipboardImage(#Image
    [, Profondeur])
```

Description

Crée une nouvelle image à partir d'une image contenue dans le presse-papier (si elle existe).

Arguments

#Image Numéro de la nouvelle image. PB_Any# peut être utilisé pour générer automatiquement ce numéro.

Profondeur (optionnel) La profondeur de couleur de l'image.
Les valeurs valides sont 24 bits (par défaut) ou 32 bits.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any est utilisé pour le paramètre '#Image', le numéro de la nouvelle image sera renvoyé dans 'Resultat'.

Remarques

L'image ainsi obtenue peut être détruite à l'aide de la commande FreeImage() .
Pour plus d'information sur les images, voir la bibliothèque Image ou le gadget ImageGadget() .

Exemple

```
If CreateImage(0,26,20)
  StartDrawing(ImageOutput(0))
    Box(0,0,26,20,RGB(255,255,255))
    Circle(13,10,5,RGB(255,0,0))
  StopDrawing()
  SetClipboardImage(0)
EndIf

If OpenWindow(0, 0, 0, 220, 100, "Presse-papier", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  If StartDrawing(WindowOutput(0))
    DrawImage(GetClipboardImage(0), 10,10)
    DrawImage(GetClipboardImage(0), 100, 30, 65, 50)
    StopDrawing()
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf

; Vous pouvez aussi ouvrir un logiciel de dessin (par exemple 'Paint')
; et collez le contenu du presse-papier (Edition puis Coller),
; et vous verrez le drapeau japonais ;-)
```

Voir aussi

SetClipboardImage() , GetClipboardText()

OS Supportés

Tous

84.3 GetClipboardText

Syntaxe

```
Resultat\$ =
    GetClipboardText()
```

Description

Renvoie le texte actuellement stocké dans le presse-papier.

Arguments

Aucun.

Valeur de retour

Renvoie le texte stocké dans le presse-papier ou une chaîne vide sinon.

Exemple

```
1  ; On place un texte dans le
     presse papier
2  SetClipboardText("Un
     exemple de texte")
3  MessageRequester("Presse-papier","Le
     presse papier contient le
     texte suivant : " +
     GetClipboardText())
```

Voir aussi

SetClipboardText() , GetClipboardImage()

OS Supportés

Tous

84.4 SetClipboardImage

Syntaxe

```
SetClipboardImage(#Image)
```

Description

Met une copie d'une image dans le presse-papier.

Arguments

#**Image** Numéro de l'image à mettre dans le presse-papier.

Valeur de retour

Aucune.

Remarques

Si une image est déjà présente dans le presse-papier, elle sera remplacée par la nouvelle.

Exemple

```
1   If CreateImage(0,26,20)
2     StartDrawing(ImageOutput(0))
3       Box(0,0,26,20,RGB(255,255,255))
4       Circle(13,10,5,RGB(255,0,0))
5     StopDrawing()
6     SetClipboardImage(0)
7   EndIf
8
9   If OpenWindow(0, 0, 0, 220,
      100, "Presse-papier",
      #PB_Window_SystemMenu |
      #PB_Window_ScreenCentered)
10    If
      StartDrawing(WindowOutput(0))
11      DrawImage(GetClipboardImage(0),
      10,10)
12      DrawImage(GetClipboardImage(0),
      100, 30, 65, 50)
13        StopDrawing()
14    EndIf
15
16    Repeat
17      Event = WaitWindowEvent()
18    Until Event =
      #PB_Event_CloseWindow
19  EndIf
20
21  ; Vous pouvez aussi ouvrir
      un logiciel de dessin (par
      exemple 'Paint')
22  ; et collez le contenu du
      presse-papier (Edition
      puis Coller),
23  ; et vous verrez le drapeau
      japonais ;-)
```

Voir aussi

GetClipboardImage() , SetClipboardText() , ClearClipboard()

OS Supportés

Tous

84.5 SetClipboardText

Syntaxe

```
SetClipboardText(Texte$)
```

Description

Place un texte dans le presse-papier.

Arguments

Texte$ Le texte à placer dans le presse-papier

Valeur de retour

Aucune.

Remarques

Si le presse-papier contient déjà un texte celui-ci est remplacé.

Exemple

```
; On place un texte dans le presse papier
SetClipboardText("Un exemple de texte")
MessageRequester("Presse-papier","Le presse papier contient le texte suivant : " + GetClipboardText())
```

Voir aussi

GetClipboardText() , SetClipboardImage() , ClearClipboard()

OS Supportés

Tous

Chapitre 85

Console

Généralités

La bibliothèque Console permet de créer une application en mode console. On utilise ce mode pour créer des programmes de petite taille ne nécessitant pas d'interface utilisateur avancé, ou pour une utilisation dans des scripts (ligne de commande). Ces instructions sont également utiles pour le programmeur lors du déboggage d'application en permettant d'afficher des informations sur la console sans interrompre le cours du programme.
Si votre programme est destiné à être une application console pure (pas une application fenêtrée qui ouvre une console), alors vous devez vous rappeler de régler le format de fichier exécutable sur "console" lorsque vous compilez votre programme. Voir le menu Compilateur \ Options du compilateur \ Options de compilation \ Format de l'exécutable.
Vous devriez commencer avec la fonction OpenConsole() , puisque vous devez utiliser cette fonction pour ouvrir une console.
Attention, ces programmes ne sont pas des programmes MS-DOS en mode réel ! Il faut donc les ouvrir à partir de Windows 95 ou plus.

OS Supportés

Tous

85.1 ClearConsole

Syntaxe

```
ClearConsole()
```

Description

Efface tout le contenu de la console.

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Le fond d'écran est déterminé par ConsoleColor() .
La console doit être en mode graphique .

Exemple

```
If OpenConsole()
  EnableGraphicalConsole(1)

  PrintN("Vous ne me verrez jamais")
  ClearConsole()

  PrintN("Appuyez sur [Entree] pour quitter")
  Input()
EndIf
```

Voir aussi

EnableGraphicalConsole() , ConsoleColor()

OS Supportés

Windows

85.2 CloseConsole

Syntaxe

```
CloseConsole()
```

Description

Ferme la console.

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Une fois que la console est fermée il n'est plus possible d'utiliser les fonctions de cette bibliothèque.
La console se ferme automatiquement à la fin de l'exécution du programme.
Cette fonction n'a pas d'effet sous Linux et MacOS.

Exemple

```
1  For i = 0 To 4
2    If OpenConsole()
3      PrintN("C'est la
     console #"+Str(i))
4      PrintN("Appuyez sur
     [Entree] pour quitter")
5      Input()
6      CloseConsole()
7    EndIf
8  Next
```

Voir aussi

OpenConsole()

OS Supportés

Tous

85.3 ConsoleError

Syntaxe

```
ConsoleError(Message$)
```

Description

Envoie une chaîne de caractères vers la sortie 'Erreur' (stderr) du programme.

Arguments

Message$ Chaîne de caractères avec un retour à la ligne.

Valeur de retour

Aucune.

Remarques

La chaîne de caractères contient un saut de ligne (LF) à la fin.
Cette sortie peut être lue par exemple avec la commande ReadProgramError() de la bibliothèque Process .

Voir aussi

Print() , PrintN()

OS Supportés

Tous

85.4 ConsoleTitle

Syntaxe

```
ConsoleTitle(Titre$)
```

Description

Change le titre de la console.
Sous Windows, le titre de la console est aussi le texte affiché dans la barre des tâches et dans le gestionnaire des tâches représentant votre application console.

Arguments

Titre$ Le nouveau titre de fenêtre.

Valeur de retour

Aucune.

Exemple

```
1  If OpenConsole()
2    ConsoleTitle("Ceci est le
     titre de la console")
3    PrintN("Appuyez sur
     [Entree] pour quitter")
4    Input()
5  EndIf
```

Voir aussi

EnableGraphicalConsole()

OS Supportés

Windows

85.5 ConsoleColor

Syntaxe

```
ConsoleColor(CouleurTexte,
    CouleurFond)
```

Description

Change les couleurs utilisées par l'écran texte.

Arguments

CouleurTexte Couleur du texte :

```
 0 - Noir (fond par défaut)
 1 - Bleu foncé
 2 - Vert foncé
 3 - Bleu-vert
 4 - Rouge foncé
 5 - Magenta foncé
 6 - Brun
 7 - Gris clair (couleur
texte par defaut)
 8 - Gris foncé
 9 - Bleu
10 - Vert
11 - Cyan
12 - Rouge
13 - Magenta
14 - Jaune
15 - Blanc
```

CouleurFond Couleur de fond :

```
 0 - Noir (fond par défaut)
 1 - Bleu foncé
 2 - Vert foncé
 3 - Bleu-vert
 4 - Rouge foncé
 5 - Magenta foncé
 6 - Brun
 7 - Gris clair (couleur
texte par defaut)
 8 - Gris foncé
 9 - Bleu
10 - Vert
11 - Cyan
12 - Rouge
13 - Magenta
14 - Jaune
15 - Blanc
```

Valeur de retour

Aucune.

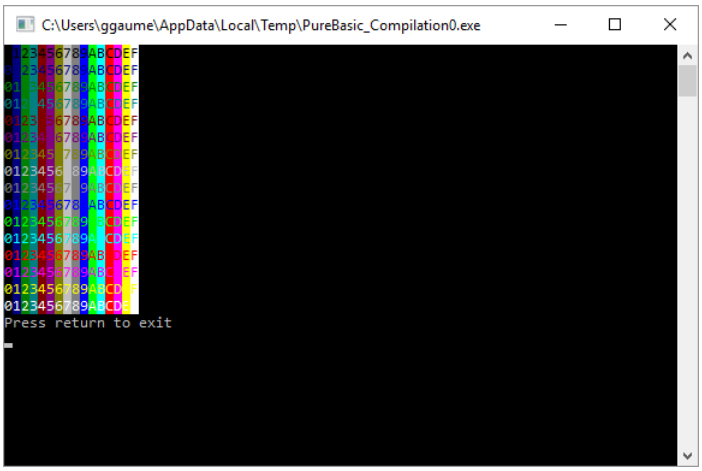
Remarques

Tous les caractères affichés après l'appel de cette fonction utiliseront ces nouvelles couleurs.

Exemple

```
1  If OpenConsole()
2    For CouleurTexte = 0 To 15
3      For CouleurFond = 0 To
   15
4        ConsoleColor(CouleurTexte,
   CouleurFond)
```

```
5           Print(Right(Hex(CouleurFond),
    1))
6       Next
7
8       PrintN("")
9     Next
10
11    ConsoleColor(7, 0)
12    PrintN("Appuyez sur
    [Entree] pour quitter")
13    Input()
14  EndIf
```



Voir aussi

EnableGraphicalConsole()

OS Supportés

Tous

85.6 EnableGraphicalConsole

Syntaxe

```
EnableGraphicalConsole(Mode)
```

Description

Change la manière d'afficher les caractères
dans la console : Texte/Graphique.

Arguments

```
Mode    0: Mode texte
        1: Mode graphique
```

Valeur de retour

Aucune.

Remarques

Le mode par défaut de la console est le mode texte, ce qui implique que le texte ne peut pas être positionné arbitrairement dans la console. Cependant, les redirections (pipes) fonctionnent correctement, ce qui peut être utile si le programme est destiné à être utilisé dans des scripts.
Quand le mode graphique est activé, les commandes comme ClearConsole() ou ConsoleLocate() sont disponibles, et le texte peut être positionné n'importe où dans la console, ce qui permet de faire des jeux ou des applications console (qui peuvent par exemple être accédées à distance via telnet ou ssh).
Attention : La redirection (pipes) ne fonctionne pas si la console est en mode graphique.

Exemple

```
1  If OpenConsole()
2    EnableGraphicalConsole(1)
3    ConsoleLocate(7, 8)
4    PrintN("Appuyez sur
       [Entree] pour quitter")
5    Input()
6  EndIf
```

Voir aussi

ConsoleLocate() , ConsoleColor() , ClearConsole()

OS Supportés

Windows

85.7 Inkey

Syntaxe

```
Resultat\$ = Inkey()
```

Description

Renvoie un caractère si une touche du clavier est appuyée.

Arguments

Aucun.

Valeur de retour

Renvoie une chaîne contenant le caractère correspondant à la touche enfoncée ou une chaîne vide si la touche ne correspond à aucun caractère (touche de fonction).

Remarques

- Cette commande ne bloque pas l'exécution du programme.
- Si des touches spéciales (non-ascii) doivent être gérées, RawKey() peut être appelé après Inkey().
- Les codes ASCII et les valeurs numériques présentées ici peuvent changer en fonction de la page de code clavier chargée pendant le boot.
- Un tableau contenant les valeurs ascii est disponible ici .

Exemple

```
1  If OpenConsole()
2    PrintN("Appuyez sur [Echap] pour quitter")
3
4    Repeat
5      KeyPressed$ = Inkey()
6
7      If KeyPressed$ <> ""
8        PrintN("Vous avez appuye sur : " + KeyPressed$)
9        PrintN("Son identifiant numerique est : "+Str(RawKey()))
10     ElseIf RawKey()
11       PrintN("Vous avez appuye sur une touche qui n'est pas alpha-numerique.")
12       PrintN("Son identifiant numerique est : "+Str(RawKey()))
13     Else
14       Delay(20) ; Evite de monopoliser tout le temps processeur. Utile pour un OS multi-tâches.
15     EndIf
16
17   Until KeyPressed$ = Chr(27) ; Attends jusqu'à ce que la touche [Echap] soit appuyée
18 EndIf
```

Voir aussi

RawKey() , Input()

OS Supportés

Tous

85.8 Input

Syntaxe

```
Resultat\$ = Input()
```

Description

Saisie une ligne complète de caractères.

Arguments

Aucun.

Valeur de retour

Renvoie la chaîne saisie par l'utilisateur avant d'appuyer sur la touche 'Entrée'.

Remarques

Cette commande bloque le programme et attend jusqu'à ce que l'utilisateur appuie sur la touche entrée. Si la console est en mode graphique la ligne saisie ne peut pas dépasser la largeur de la console.
Si la console n'est pas en mode graphique , une valeur de retour spéciale #PB_Input_Eof (égale à Chr(4)) sera reçue si l'utilisateur tape Ctrl+D dans la console, ou si la redirection d'un fichier vers la console est terminée.
Pour des raisons de compatibilité avec les autres applications console sous Windows, #PB_Input_Eof est aussi reçu quand l'utilisateur tape Ctrl+Z dans la console.
Si une entrée binaire est nécessaire, ReadConsoleData() peut être utilisé en mode non graphique.

Exemple

```
1 If OpenConsole()
2     Print("Entrez votre nom
      et appuyez sur [Entree]: ")
3     name$=Input()
4
5     ; L'appui sur la touche
      [Entree] n'est pas pris en
      compte par la console
```

```
    ; donc nous devons utiliser PrintN("") pour aller à la ligne
    PrintN("")

    PrintN("Bonjour, "+name$+", enchante.")
    PrintN("Appuyez sur [Entree] pour quitter")
    Input()
    CloseConsole()
EndIf
End
```

Voir aussi

Inkey() , RawKey()

OS Supportés

Tous

85.9 ConsoleLocate

Syntaxe

```
ConsoleLocate(X, Y)
```

Description

Déplace le curseur.

Arguments

X, Y Les nouvelles coordonnées du curseur (commence à 0).

Valeur de retour

Aucune.

Remarques

Les coordonnées sont en caractères et non en pixels.
La console doit être en mode graphique .

Exemple

```
If OpenConsole()
  EnableGraphicalConsole(1)

  For i = 0 To 200
    ConsoleLocate(Random(79), Random(24))
    Print("*")
  Next
```

```
8
9       ConsoleLocate(30, 10)
10      PrintN("Appuyez sur
        [Entree] pour quitter")
11      Input()
12    EndIf
```

Voir aussi

EnableGraphicalConsole()

OS Supportés

Windows

85.10 ConsoleCursor

Syntaxe

```
ConsoleCursor(Hauteur)
```

Description

Change le curseur.

Arguments

Hauteur La nouvelle hauteur du curseur.
Cette valeur peut aller de 0 à 10.

```
Par exemple:
0 : Curseur invisible
1 : Curseur souligné par
 défaut
5 : Curseur demi-pavé
10: Curseur pavé
```

Valeur de retour

Aucune.

Remarques

Par défaut, le curseur est représenté par un soulignement clignotant dans les consoles sous le système d'exploitation Windows.
Notez que vous devrez peut-être mettre la fenêtre de la console en mode plein écran pour voir l'effet de cette fonction.
La console doit être en mode graphique . Voir EnableGraphicalConsole() .

Exemple

```
If OpenConsole()
  EnableGraphicalConsole(1)

  For HauteurCurseur=0 To 10
    ConsoleCursor(HauteurCurseur)
    PrintN("Appuyez sur [Entree] pour augmenter la hauteur du curseur")
    Input()
  Next

  PrintN("Appuyez sur [Entree] pour quitter")
  Input()
EndIf
```

Voir aussi

EnableGraphicalConsole()

OS Supportés

Windows

85.11 Print

Syntaxe

```
Print(Texte$)
```

Description

Affiche un texte sur la console sans retour de ligne à la fin.

Arguments

Texte$ Le texte à afficher.

Valeur de retour

Aucune.

Remarques

Si la console est en mode graphique , la longueur de Texte$ ne peut pas dépasser la largeur de la console, sinon le texte sera tronqué.
Pour modifier la position du curseur, utilisez la commande ConsoleLocate() .
Pour changer la couleur du texte affiché, utilisez la commande ConsoleColor() .
Une fois le 'Texte$' affiché, le curseur est automatiquement déplacé après le dernier

caractère. Si le texte dépasse la largeur de la console, il est automatiquement coupé et affiché sur la ligne suivante. Si la console est remplie, le texte est automatiquement déplacé vers le haut.
Pour afficher des données binaires en mode non-graphique (par exemple lors d'une redirection), la commande WriteConsoleData() peut être utilisée.

Exemple

```
1  If OpenConsole()
2    Print("C'est une longue chaîne de caracteres. ")
3    Print("Celle-ci se place immediatement apres la première... ")
4    Print("C'est parce que le curseur se place en fin de chaîne, sans retour a la ligne. ")
5
6    PrintN("Appuyez sur [Entree] pour quitter")
7    Input()
8  EndIf
```

Voir aussi

PrintN() , Input()

OS Supportés

Tous

85.12 PrintN

Syntaxe

```
PrintN(Texte$)
```

Description

Affiche un texte sur la console avec un retour de ligne à la fin.

Arguments

Texte$ Le texte à afficher.

Valeur de retour

Aucune.

Remarques

Si la console est en mode graphique , la longueur de Texte$ ne peut pas dépasser la largeur de la console, sinon le texte sera tronqué.
Pour modifier la position du curseur, utilisez la commande ConsoleLocate() .
Pour changer la couleur du texte affiché, utilisez la commande ConsoleColor() .
Une fois le 'Texte$' affiché, le curseur est automatiquement déplacé après le dernier caractère. Si le texte dépasse la largeur de la console, il est automatiquement coupé et affiché sur la ligne suivante. Si la console est remplie, le texte est automatiquement déplacé vers le haut.
Pour afficher des données binaires en mode non-graphique (par exemple lors d'une redirection), la commande WriteConsoleData() peut être utilisée.

Exemple

```
1  If OpenConsole()
2    PrintN("C'est la premiere ligne..")
3    PrintN("..suivie de la deuxieme..")
4    PrintN(".. et de la troisieme !")
5
6    PrintN("Appuyez sur [Entree] pour quitter")
7    Input()
8    CloseConsole()
9  EndIf
```

Voir aussi

Print() , Input()

OS Supportés

Tous

85.13 OpenConsole

Syntaxe

```
Resultat =
    OpenConsole([Titre$] [,
    Mode]]))
```

Description

Ouvre une fenêtre console.
Cette fonction doit être appelée avant toute autre opération sur la console.

Arguments

Titre$ (optionnel) Le titre de la nouvelle fenêtre console.
Sous Windows, la présence d'un titre permet la sauvegarde des paramètres de la console, comme la police, la couleur, etc. Il n'a aucun effet sur les autres OS.

Mode (optionnel) Le mode à utiliser pour la sortie de la console. Il peut avoir l'une des valeurs suivantes :

```
- #PB_UTF8    : Le texte
  est au format UTF-8 (Par
  défaut).
- #PB_Ascii   : Le texte
  est au format ASCII.
- #PB_Unicode: Le texte
  est au format UTF-16.
                Peut être
  utile sous Windows
  lorsque vous utilisez la
  redirection de la sortie
                et que le
  programme cible attend
  une entrée UTF-16. N'a
  aucun effet sur Linux
                ou OS X
  (qui utiliseront UTF-8).
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon et la console ne peut pas être utilisée.

Remarques

Une seule console peut être ouverte au même moment sous PureBasic.
La console peut être fermée en utilisant la commande CloseConsole() .
La commande EnableGraphicalConsole() permet de permuter entre le mode texte et le mode graphique.
Sous Microsoft Windows, deux formats d'exécution sont disponibles : Win32 et Console. Si vous souhaitez créer une application standard sur console, telle que 'dir', 'del', etc vous devez compiler l'exécutable en utilisant le format Console du (menu Compilateur : Compilateur\Options du compilateur\ Options de compilation\Format de l'exécutable de l'éditeur de PureBasic).
Sous Linux ou OS X, cette commande n'a pas d'effet, car il n'y a pas de format spécial 'console' pour les exécutables. Cependant, l'option de compilation pour 'Console'

lancera une fenêtre de terminal automatiquement lorsque vous exécutez votre programme à partir de l'IDE.

Exemple

```
1  OpenConsole()
2  PrintN("Ce programme
     quittera dans 5
     secondes...")
3  Delay(5000)
```

Voir aussi

CloseConsole() , EnableGraphicalConsole()

OS Supportés

Tous

85.14 ReadConsoleData

Syntaxe

```
Resultat =
    ReadConsoleData(*Memoire,
    Taille)
```

Description

Lit l'entrée de la console de manière binaire.

Arguments

***Memoire** La mémoire tampon dans laquelle les données seront stockées. Mémoire auparavant allouée avec AllocateMemory .

Size La quantité maximale de données à lire (en octets).

Valeur de retour

Renvoie le nombre d'octets lu à partir de l'entrée.
Si la valeur zéro est renvoyée alors il n'y a plus de données à lire (un indicateur de fin de fichier a été reçu).

Remarques

Cette commande fonctionne uniquement si le console n'est pas en mode graphique . Elle est utile pour lire des données qui ne sont pas basées sur des lignes de texte (à l'inverse de la commande Input()) ou des

données qui on été redirigées vers le programme à travers un 'pipe'.
Cette commande est bloquante, ce qui signifie que si aucune donnée n'est disponible en entrée, alors le programme attendra l'arrivée d'une donnée indéfiniment sauf si une erreur survient ou un EOF (End Of File).

Exemple

```
1  ; Ce programme lit une image passée en paramètre depuis une console et l'affiche dans une fenêtre.
2  ; Il faut le compiler et créer un exécutable, ensuite le lancer de cette façon "MonExecutable < image.bmp"
3  ;
4  ; (Dans les options du compilateur sélectionnez "Console" pour le "format de l'exécutable" !)
5  ; (Fonctionne seulement avec des BMP ou des icônes à moins d'utiliser un décodeur d'images (voir ImagePlugin))
6  ;
7  OpenConsole()
8  TotalSize = 0
9  BufferFree = 10000
10 *Buffer = AllocateMemory(BufferFree)
11
12 Repeat
13   ReadSize = ReadConsoleData(*Buffer+TotalSize, BufferFree) ; Lit un bloc de données
14   TotalSize + ReadSize
15   BufferFree - ReadSize
16   If BufferFree < 100  ; Redimensionne le buffer s'il n'est pas assez grand
17     BufferFree = 10000
18     *Buffer = ReAllocateMemory(*Buffer, TotalSize+10000)
19   EndIf
20 Until ReadSize = 0 ; Une fois que 0 est retourné, il n'y a plus rien à lire
21
22 If TotalSize > 0 ; affiche l'image si tout se passe bien
```

```
23      If CatchImage(0, *Buffer,
        TotalSize)
24         If OpenWindow(0, 0, 0,
        ImageWidth(0),
        ImageHeight(0), "Image",
        #PB_Window_SystemMenu |
        #PB_Window_ScreenCentered)
25           ImageGadget(0, 0, 0,
        ImageWidth(0),
        ImageHeight(0), ImageID(0))
26           Repeat
27           Until
        WaitWindowEvent() =
        #PB_Event_CloseWindow
28           End
29         EndIf
30       EndIf
31    EndIf
32    MessageRequester("Erreur",
       "Ce n'est pas une image
       valide.")
```

Voir aussi

WriteConsoleData() , AllocateMemory()

OS Supportés

Tous

85.15 RawKey

Syntaxe

```
Resultat = RawKey()
```

Description

Renvoie l'identifiant numérique ('key code') de la touche capturée lors du dernier appel de la commande Inkey() . Cela permet de gérer les touches qui n'ont pas de caractères ASCII associés (par exemple F1, F2, les flèches etc.).

Arguments

Aucun.

Valeur de retour

Renvoie le code de la dernière touche pressée.

Remarques

Les touches alpha-numériques ne sont pas les seules à avoir une correspondance dans la table ASCII, par exemple la touche 'Echap' a la valeur ASCII 27, la touche 'Entrée' la valeur ASCII 13 , la touche 'Tab' a la valeur ASCII 9 et la touche 'Retour Arrière' a la valeur ASCII 8 , etc.
Voir les codes ASCII ici .

Exemple

```
1  If OpenConsole()
2    PrintN("Appuyez sur [Echap] pour quitter")
3
4    Repeat
5      KeyPressed$ = Inkey()
6
7      If KeyPressed$ <> ""
8
9        PrintN("Vous avez appuye sur : " + KeyPressed$)
10       PrintN("Son identifiant numerique est : "+Str(RawKey()))
11
12     ElseIf RawKey()
13
14       PrintN("Vous avez appuye sur une touche qui n'est pas alpha-numerique.")
15       PrintN("Son identifiant numerique est : "+Str(RawKey()))
16
17     Else
18       Delay(20) ; Evite de monopoliser tout le temps processeur. Utile pour un OS multi-tâches.
19     EndIf
20
21   Until KeyPressed$ = Chr(27) ; Attends jusqu'à ce que la touche [Echap] soit appuyée
22 EndIf
```

Voir aussi

Inkey() , Input()

OS Supportés

Windows

85.16 WriteConsoleData

Syntaxe

```
Resultat =
    WriteConsoleData(*Memoire,
    Taille)
```

Description

Ecrit le contenu binaire d'une zone en mémoire buffer dans la sortie standard de la console.

Arguments

***Memoire** La mémoire tampon dans laquelle les données doivent être lues.

Taille La quantité maximale de données (en octets) à écrire.

Valeur de retour

Renvoie le nombre d'octets réellement écrit dans la sortie standard de la console.

Remarques

Cette commande fonctionne uniquement si le console n'est pas en mode graphique . Elle est utile pour écrire autre chose que du texte dans la console, ce qui peut être intéressant lorsqu'elle est ensuite redirigée vers un fichier ou un autre programme.

Voir aussi

ReadConsoleData()

OS Supportés

Tous

Chapitre 86

Database

Généralités

La bibliothèque database est un jeu d'instructions simple pour l'accès à SQLite, PostgreSQL, MySQL, DBMaria ou à tous types de bases de données (Oracle, MySQL, Access, etc..) via l'API commune ODBC.
La bibliothèque est basée sur des requêtes SQL pour lire / écrire des données dans une base. Il est donc recommandé de disposer d'une documentation si nécessaire.
Voici quelques liens concernant la syntaxe SQL :
Tutorial SQL W3Schools
Commandes SQL de SQLite
Manuel de PostgreSQL
L'environnement de gestion des bases de données est initialisé à l'aide des commandes UseODBCDatabase() , UseSQLiteDatabase() , UseMySQLDatabase() et UsePostgreSQLDatabase() .
Note : Sous Windows, avant d'employer cette bibliothèque il est nécessaire d'établir une source de données utilisateur, ce qui rend votre base de données disponible par l'intermédiaire d'ODBC et utilisable avec cette bibliothèque. Pour plus d'informations, référez vous au document d'aide de l'ODBC de Windows.

OS Supportés

Tous

86.1 AffectedDatabaseRows

Syntaxe

```
Resultat =
    AffectedDatabaseRows(#BaseDeDonnees)
```

Description

Renvoie le nombre de lignes affectées par la dernière opération DatabaseUpdate() .

Arguments

#**BaseDeDonnees** La base de données à utiliser.

Valeur de retour

Renvoie le nombre de lignes affectées par la dernière opération DatabaseUpdate() .

Voir aussi

DatabaseUpdate()

OS Supportés

Tous

86.2 CloseDatabase

Syntaxe

```
CloseDatabase(#BaseDeDonnees)
```

Description

Ferme une base de données et les connexions / transactions si elles existent.

Arguments

#**BaseDeDonnees** La base de données à fermer.
Si #PB_All est spécifié, toutes les bases de données restantes sont fermées.

Valeur de retour

Aucune.

Remarques

Aucune opération ne peut plus être faite sur la base concernée.
Toutes les bases de données restant ouvertes sont automatiquement fermées quand le programme se termine.

Voir aussi

OpenDatabase() ,
OpenDatabaseRequester()

OS Supportés

Tous

86.3 DatabaseColumns

Syntaxe

```
Resultat =
    DatabaseColumns(#BaseDeDonnees)
```

Description

Renvoie le nombre de champs après la dernière requête exécutée avec DatabaseQuery() .

Arguments

#BaseDeDonnees La base de données à utiliser.

Valeur de retour

Renvoie le nombre de colonnes de la base de données depuis la dernière requête.

Voir aussi

DatabaseColumnName() , DatabaseColumnType() , DatabaseColumnSize()

OS Supportés

Tous

86.4 DatabaseColumnIndex

Syntaxe

```
Resultat =
    DatabaseColumnIndex(#BaseDeDonnees,
    NomColonne$)
```

Description

Renvoie l'index (le numéro) de la colonne après l'exécution d'une requête avec DatabaseQuery() dans la base de données.

Arguments

#BaseDeDonnees La base de données à utiliser.

NomColonne$ Le nom de la colonne.

Valeur de retour

Renvoie l'index de la colonne spécifiée ou -1 si le nom de la colonne est invalide. Ceci n'est valable qu'après avoir exécuté une requête avec DatabaseQuery() .

Remarques

Cela peut être utile pour utiliser des commandes telles que GetDatabaseLong() qui requiert un indice de colonne.

Voir aussi

DatabaseQuery() , GetDatabaseBlob() , GetDatabaseDouble() , GetDatabaseFloat() , GetDatabaseString() , GetDatabaseQuad()

OS Supportés

Tous

86.5 DatabaseColumnName

Syntaxe

```
Resultat\$ =
    DatabaseColumnName(#BaseDeDonnees,
    Colonne)
```

Description

Renvoie le nom d'un champ.

Arguments

#BaseDeDonnees La base de données à utiliser.

Colonne Le champ à utiliser. DatabaseColumnIndex() est disponible pour obtenir l'index d'une colonne.

Valeur de retour

Renvoie le nom de la colonne.

Voir aussi

DatabaseColumns() , DatabaseColumnType() , DatabaseColumnSize()

OS Supportés

Tous

86.6 DatabaseColumnSize

Syntaxe

```
Resultat =
    DatabaseColumnSize(#BaseDeDonnees,
    Colonne)
```

Description

Renvoie la taille d'un champ.

Arguments

#BaseDeDonnees La base de données à utiliser.

Colonne Le champ à utiliser.

Valeur de retour

Renvoie la taille de la colonne en octets.

Remarques

C'est particulièrement utile quand la taille de la colonne peut changer en fonction des enregistrements, comme une colonne de type blob ou string.

Voir aussi

DatabaseColumns() ,
DatabaseColumnType() ,
DatabaseColumnName()

OS Supportés

Tous

86.7 DatabaseColumnType

Syntaxe

```
Resultat =
    DatabaseColumnType(#BaseDeDonnees,
    Colonne)
```

Description

Renvoie le type d'un champ.

Arguments

#BaseDeDonnees La base de données à utiliser.

Colonne Le champ à utiliser.

Valeur de retour

Renvoie le type de donnée du champ.
Si le résultat est nul, le type est indéterminé ou la commande a échoué (impossibilité de retrouver le type).
Les valeur du type peuvent être :

```
#PB_Database_Long   : Format numérique          :  Long (.l) sous PureBasic
#PB_Database_String: Format chaîne             : String (.s) sous PureBasic
#PB_Database_Float  : Format numérique flottant:  Float (.f) sous PureBasic
#PB_Database_Double: Format numérique double   : Double (.d) sous PureBasic
#PB_Database_Quad   : Format numérique quad     :  Quad (.q) sous PureBasic
#PB_Database_Blob   : Format blob
```

Voir aussi

DatabaseColumns() ,
DatabaseColumnName() ,
DatabaseColumnSize()

OS Supportés

Tous

86.8 DatabaseDriverDescription

Syntaxe

```
Reultat\$ =
    DatabaseDriverDescription()
```

Description

Renvoie la description du pilote en cours d'examen avec ExamineDatabaseDrivers() .

Arguments

Aucun.

Valeur de retour

Renvoie la description du pilote.

Remarques

Les pilotes sont listés d'après les commandes ExamineDatabaseDrivers() et NextDatabaseDriver() .
C'est une commande spécifique aux bases de données ODBC .
Les bases de données SQLite ne supportent pas cette commande.

Voir aussi

ExamineDatabaseDrivers() , NextDatabaseDriver() , DatabaseDriverName()

OS Supportés

Tous

86.9 DatabaseDriverName

Syntaxe

```
Resultat\$ =
    DatabaseDriverName()
```

Description

Renvoie le nom du pilote en cours d'examen avec ExamineDatabaseDrivers() .

Arguments

Aucun.

Valeur de retour

Renvoie le nom du pilote.

Remarques

Les pilotes sont listés d'après les commandes ExamineDatabaseDrivers() et NextDatabaseDriver() .
C'est une commande spécifique aux bases de données ODBC .
Les bases de données SQLite ne supportent pas cette commande.

Voir aussi

ExamineDatabaseDrivers() , NextDatabaseDriver() , DatabaseDriverDescription()

OS Supportés

Tous

86.10 DatabaseError

Syntaxe

```
Resultat\$ = DatabaseError()
```

Description

Renvoie la dernière erreur survenue.

Arguments

Aucun.

Valeur de retour

Renvoie la dernière erreur survenue lors de l'exécution d'une commande sur la base de données.

Remarques

C'est particulièrement utile avec les commandes suivantes : OpenDatabase() , DatabaseQuery() et DatabaseUpdate() .

Exemple

```
1  ; D'abord, il faudra se
     connecter à une base de
     données qui contient une
     table 'employee'
2  ;
3  If
     DatabaseQuery(#BaseDeDonnees,
     "SELECT * FROM employee")
     ; Récupère tous les
     enregistrements de la
     table 'employee'
4      ; ...
5      FinishDatabaseQuery(#BaseDeDonnees)
6  Else
7      MessageRequester("Erreur",
     "Impossible d'exécuter la
     requête: "+DatabaseError())
8  EndIf
```

Voir aussi

DatabaseQuery() , DatabaseUpdate()

OS Supportés

Tous

86.11 DatabaseID

Syntaxe

```
Resultat =
    DatabaseID(#BaseDeDonnees)
```

Description

Renvoie l'identifiant unique d'une base de données dans le système d'exploitation.

Arguments

#BaseDeDonnees La base de données à utiliser.

Valeur de retour

Renvoie l'identifiant de la connexion à la base de données.

Remarques

Cette fonction est utile quand une autre bibliothèque nécessite la référence de la #BaseDeDonnees.

Voir aussi

OpenDatabase() ()

OS Supportés

Tous

86.12 DatabaseQuery

Syntaxe

```
Resultat =
    DatabaseQuery(#BaseDeDonnees,
    Requete$ [, Options])
```

Description

Exécute une requête SQL qui ne modifie pas la base de donnée (de type 'SELECT' par exemple).

Arguments

#BaseDeDonnees La base de données à utiliser.

Requete$ La requête SQL à exécuter.

Options (optionnel) Peut être une des valeurs suivantes :

```
  #PB_Database_StaticCursor
   : Exécute la requête
   pour accéder au résultat
   d'une manière
   séquentielle.
                              Il
   n'est pas possible de
   revenir en arrière avec
   PreviousDatabaseRow()
 ou FirstDatabaseRow()

                              sur
   certains pilotes, mais
   c'est le moyen le plus
   rapide pour obtenir les
   données (par défaut).
  #PB_Database_DynamicCursor:
   Exécute la requête pour
   accéder au résultat
   d'une manière aléatoire
                              en
   utilisant
   PreviousDatabaseRow()
 ou FirstDatabaseRow()
.
                              Cela
   peut être plus lent,
   voire non pris en charge
   sur certains pilotes.
```

Valeur de retour

Renvoie une valeur non nulle si la requête a réussi, zéro sinon (en raison d'une erreur SQL ou une requête mal formatée).

Remarques

Pour modifier une base de données, utiliser DatabaseUpdate() .
Si une requête n'a pas abouti, la description exacte de l'erreur peut être récupérée grâce à DatabaseError() .
Si une requête a abouti alors NextDatabaseRow() peut être utilisé pour lister les enregistrements renvoyés. Cette commande affecte toujours NextDatabaseRow() même si aucun enregistrement n'est renvoyé par la requête. Il est conseillé d'utiliser DatabaseUpdate() quand une requête ne renvoie jamais d'enregistrement.
Pour obtenir le nombre de colonnes (champs) renvoyé par la requête, utiliser DatabaseColumns() .
Pour exécuter une requête qui ne renvoie pas tous les enregistrements (données), utiliser DatabaseUpdate() au lieu de DatabaseQuery() ().

Une fois que les résultats de la requête ne sont plus nécessaires, FinishDatabaseQuery() doit être appelée pour libérer toutes les ressources de la requête.
La requête peut contenir des espaces réservés pour les variables de liaison. Ces variables doivent être définies avant d'appeler SetDatabaseString() , SetDatabaseLong() etc. Après l'exécution de la requête, les variables liées sont effacées et doivent être définies à nouveau pour de futurs appels. La syntaxe de spécification des variables liées à SQL dépend de la base de données. Voir le deuxième exemple ci-dessous.

Exemple

```
1  ; D'abord, il faudra se
     connecter à une base de
     données qui contient une
     table 'employee'
2  ;
3  If
     DatabaseQuery(#BaseDeDonnees,
     "SELECT * FROM employee")
     ; Recupère tous les
     enregistrements de la
     table 'employee'
4
5     While
     NextDatabaseRow(#BaseDeDonnees)
     ; Enumeration des
     enregistrements
6       Debug
     GetDatabaseString(#BaseDeDonnees,
     0) ; Affichage du contenu
     du premier champ
7     Wend
8
9     FinishDatabaseQuery(#BaseDeDonnees)
10 EndIf
```

Exemple : Variables liées avec SQLite, MySQL et ODBC

```
1  ; SQLite, MySQL et ODBC
     partagent la même syntaxe
     pour les variables de
     liaison, indiquées par le
     caractère "?"
2  ;
3  SetDatabaseString(#Database,
     0, "test")
4  If DatabaseQuery(#Database,
     "SELECT * FROM employee
     WHERE id=?")
```

```
5     ; ...
6   EndIf
```

Exemple : PostgreSQL

```
1   ; PostgreSQL utilise une
     autre syntaxe dans la
     déclaration: $1, $2 ..
     pour indiquer le paramètre
     indéfini
2   ;
3   SetDatabaseString(#Database,
     0, "test")
4   If DatabaseQuery(#Database,
     "SELECT * FROM employee
     WHERE id=$1")
5     ; ...
6   EndIf
```

Voir aussi

DatabaseUpdate() , NextDatabaseRow() SetDatabaseString() , SetDatabaseLong() , SetDatabaseQuad() , SetDatabaseFloat() , SetDatabaseDouble() , SetDatabaseBlob() , SetDatabaseNull()

OS Supportés

Tous

86.13 DatabaseUpdate

Syntaxe

```
Resultat =
    DatabaseUpdate(#BaseDeDonnees,
    Requete$)
```

Description

Exécute une requête SQL de modification de la base de données. Elle ne renvoie aucun enregistrement.

Arguments

#BaseDeDonnees La base de données à utiliser.

Requete$ La requête à exécuter.

Valeur de retour

Renvoie une valeur non nulle si la requête a réussi, zéro sinon (en raison d'une erreur SQL ou une requête mal formatée).

Remarques

Cette fonction est similaire à DatabaseQuery() mais sans affecter la fonction NextDatabaseRow() , il est donc possible d'utiliser cette commande tout en conservant les enregistrements obtenus par DatabaseQuery() . Par contre, il n'est pas possible d'effectuer des sélections ('SELECT' en SQL) ou d'autres types de requêtes qui renvoient des enregistrements, pour cela, utiliser DatabaseQuery() .
Cette fonction est utile pour mettre à jour des enregistrements dans la base de données.
En cas d'erreur, le texte d'erreur peut être récupéré par DatabaseError() .
La requête peut contenir des espaces réservés pour les variables de liaison. Ces variables doivent être définies avant d'appeler SetDatabaseString() , SetDatabaseLong() etc. Après l'exécution de la requête, les variables liées sont effacées et doivent être définies à nouveau pour de futurs appels. La syntaxe de spécification des variables liées à SQL dépend de la base de données. Voir le deuxième exemple ci-dessous.

Exemple

```
1  ; D'abord, il faudra se connecter à une base de données qui contient une table 'employee'
2  ;
3  If DatabaseQuery(#BaseDeDonnees, "SELECT * FROM employee") ; Recupère tous les enregistrements de la table 'employee'
4
5     While NextDatabaseRow(#BaseDeDonnees) ; Enumération des enregistrements
6
7        ; Mise à jour du champ 'checked' pour chaque enregistrement, en assumant que le champ 'id'
8        ; est le premier de la table 'employee'
9        ;
10       DatabaseUpdate("UPDATE employee SET checked=1 WHERE id="+GetDatabaseString(#BaseDeDonnees,
```

```
    0))
11    Wend
12
13    FinishDatabaseQuery(#BaseDeDonnees)
14  EndIf
```

Exemple : Variables liées avec SQLite, MySQL et ODBC

```
1   ; SQLite, MySQL et ODBC
     partagent la même syntaxe
     pour les variables de
     liaison, indiquées par le
     caractère "?"
2   ;
3   SetDatabaseString(#Database,
     0, "test")
4   If DatabaseQuery(#Database,
     "SELECT * FROM employee
     WHERE id=?")
5     ; ...
6   EndIf
```

Exemple : PostgreSQL

```
1   ; PostgreSQL utilise une
     autre syntaxe dans la
     déclaration: $1, $2 ..
     pour indiquer le paramètre
     indéfini
2   ;
3   SetDatabaseString(#Database,
     0, "test")
4   If DatabaseQuery(#Database,
     "SELECT * FROM employee
     WHERE id=$1")
5     ; ...
6   EndIf
```

Voir aussi

DatabaseQuery() SetDatabaseString() , SetDatabaseLong() , SetDatabaseQuad() , SetDatabaseFloat() , SetDatabaseDouble() , SetDatabaseBlob() , SetDatabaseNull()

OS Supportés

Tous

86.14 ExamineDatabaseDrivers

Syntaxe

```
Resultat =
    ExamineDatabaseDrivers()
```

Description

Examine les pilotes de bases de données disponibles dans le système.

Arguments

Aucun.

Valeur de retour

Si ODBC n'est pas installé ou qu'aucun pilote n'est disponible, la valeur retournée est 0, sinon NextDatabaseDriver() peut être utilisé pour lister tous les pilotes.

Remarques

C'est une commande spécifique aux bases de données ODBC .
Les bases de données SQLite ne supportent pas cette commande.

Voir aussi

NextDatabaseDriver() ,
DatabaseDriverName() ,
DatabaseDriverDescription()

OS Supportés

Tous

86.15 FinishDatabaseQuery

Syntaxe

```
FinishDatabaseQuery(#BaseDeDonnees)
```

Description

Termine la requête SQL en cours et libère les ressources associées.

Arguments

#BaseDeDonnees La base de données à utiliser.

Valeur de retour

Aucune.

Remarques

Les requêtes comme FirstDatabaseRow() ou NextDatabaseRow() ne peuvent plus être utilisées.

Exemple

```
; D'abord, il faudra se connecter à une base de données qui contient une table 'employee'
;
If DatabaseQuery(#BaseDeDonnees, "SELECT * FROM employee") ; Recupère tous les enregistrements de la table 'employee'

  While NextDatabaseRow(#BaseDeDonnees) ; Enumération des enregistrements
    Debug GetDatabaseString(#BaseDeDonnees, 0) ; Affiche le contenu du premier champ
  Wend

  FinishDatabaseQuery(#BaseDeDonnees)
EndIf
```

Voir aussi

DatabaseQuery()

OS Supportés

Tous

86.16 FirstDatabaseRow

Syntaxe

```
Resultat = FirstDatabaseRow(#BaseDeDonnees)
```

Description

Récupère les informations relatives au premier enregistrement d'une base de données.

Arguments

#BaseDeDonnees La base de données à utiliser.

Valeur de retour

Si le résultat est zéro alors aucun enregistrement n'est disponible.

Remarques

Attention, l'option #PB_Database_DynamicCursor doit être spécifiée pour que la fonction DatabaseQuery() puisse fonctionner. Pour accéder aux informations à l'intérieur de l'enregistrement on peut utiliser GetDatabaseLong() , GetDatabaseFloat() et GetDatabaseString() .

Voir aussi

NextDatabaseRow() , PreviousDatabaseRow() , GetDatabaseLong()

OS Supportés

Tous

86.17 GetDatabaseBlob

Syntaxe

```
Resultat =
    GetDatabaseBlob(#BaseDeDonnees,
    Colonne, *Memoire,
    TailleMemoire)
```

Description

Renvoie le contenu d'un champ sous forme de pointeur mémoire vers un blob.

Arguments

#BaseDeDonnees La base de données à utiliser.

Colonne La colonne à utiliser. L'index commence à 0. DatabaseColumnIndex() est disponible pour obtenir l'index d'une colonne.

***Memoire** L'adresse du blob.

TailleMemoire La taille du blob en octets.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon alors le blob ne peut pas être récupéré ou son contenu est vide.

Remarques

DatabaseColumnType() permet de déterminer le type d'une colonne et DatabaseColumnSize() permet de déterminer la taille du blob.
Note : Cette commande peut être appelée seulement une fois par colonne (il faut stocker le résultat dans une variable s'il doit être utilisé plusieurs fois, car un nouvel appel sur la même colonne renverrait une valeur erronée). C'est une limitation de la technologie ODBC.

Voir aussi

GetDatabaseDouble() , GetDatabaseFloat() , GetDatabaseLong() , GetDatabaseString() , GetDatabaseQuad()

OS Supportés

Tous

86.18 GetDatabaseDouble

Syntaxe

```
Resultat.d =
    GetDatabaseDouble(#BaseDeDonnees,
    Colonne)
```

Description

Renvoie le contenu d'un champ sous la forme d'un nombre à virgule en double précision (Double).

Arguments

#BaseDeDonnees La base de données à utiliser.

Colonne La colonne à utiliser. DatabaseColumnIndex() est disponible pour obtenir l'index d'une colonne.

Valeur de retour

Renvoie une valeur numérique à virgule flottante et en double précision.

Remarques

Cette commande n'est valide qu'après un succès de FirstDatabaseRow() , PreviousDatabaseRow() ou NextDatabaseRow() .

Pour connaître le type d'un champ il faut utiliser DatabaseColumnType() .
Note : Cette commande peut être appelée seulement une fois par colonne (il faut stocker le résultat dans une variable s'il doit être utilisé plusieurs fois, car un nouvel appel sur la même colonne renverrait une valeur erronée). C'est une limitation de la technologie ODBC.

Voir aussi

GetDatabaseBlob() , GetDatabaseFloat() , GetDatabaseLong() , GetDatabaseString() , GetDatabaseQuad()

OS Supportés

Tous

86.19 GetDatabaseFloat

Syntaxe

```
Resultat.f =
    GetDatabaseFloat(#BaseDeDonnees,
    Colonne)
```

Description

Renvoie le contenu d'un champ sous la forme d'un nombre à virgule en simple précision (Float).

Arguments

#BaseDeDonnees La base de données à utiliser.

Colonne La colonne à utiliser. DatabaseColumnIndex() est disponible pour obtenir l'index d'une colonne.

Valeur de retour

Renvoie une valeur numérique à virgule flottante et en simple précision.

Remarques

Cette commande n'est valide qu'après un succès de FirstDatabaseRow() , PreviousDatabaseRow() ou NextDatabaseRow() .
Pour connaître le type d'un champ il faut utiliser DatabaseColumnType() .
Note : Cette commande peut être appelée seulement une fois par colonne (il faut stocker le résultat dans une variable s'il doit

être utilisé plusieurs fois, car un nouvel appel sur la même colonne renverrait une valeur erronée). C'est une limitation de la technologie ODBC.

Voir aussi

GetDatabaseBlob() , GetDatabaseDouble() , GetDatabaseLong() , GetDatabaseString() , GetDatabaseQuad()

OS Supportés

Tous

86.20 GetDatabaseLong

Syntaxe

```
Resultat.l =
    GetDatabaseLong(#BaseDeDonnees,
    Colonne)
```

Description

Renvoie le contenu d'un champ au format numérique Long .

Arguments

#BaseDeDonnees La base de données à utiliser.

Colonne La colonne à utiliser. DatabaseColumnIndex() est disponible pour obtenir l'index d'une colonne.

Valeur de retour

Renvoie une valeur numérique entière (sur 4 octets).

Remarques

Cette commande n'est valide qu'après un succès de FirstDatabaseRow() , PreviousDatabaseRow() ou NextDatabaseRow() .
Pour connaître le type d'un champ il faut utiliser DatabaseColumnType() .
Note : Cette commande peut être appelée seulement une fois par colonne (il faut stocker le résultat dans une variable s'il doit être utilisé plusieurs fois, car un nouvel appel sur la même colonne renverrait une valeur erronée). C'est une limitation de la technologie ODBC.

Voir aussi

GetDatabaseBlob() , GetDatabaseDouble() , GetDatabaseFloat() , GetDatabaseString() , GetDatabaseQuad()

OS Supportés

Tous

86.21 GetDatabaseQuad

Syntaxe

```
Resultat.q =
    GetDatabaseQuad(#BaseDeDonnees,
    Colonne)
```

Description

Renvoie le contenu d'un champ au format numérique quad .

Arguments

#**BaseDeDonnees** La base de données à utiliser.

Colonne La colonne à utiliser. DatabaseColumnIndex() est disponible pour obtenir l'index d'une colonne.

Valeur de retour

Renvoie une valeur numérique entière (sur 8 octets).

Remarques

Cette commande n'est valide qu'après un succès de FirstDatabaseRow() , PreviousDatabaseRow() ou NextDatabaseRow() .
Pour connaître le type d'un champ il faut utiliser DatabaseColumnType() .
Note : Cette commande peut être appelée seulement une fois par colonne (il faut stocker le résultat dans une variable s'il doit être utilisé plusieurs fois, car un nouvel appel sur la même colonne renverrait une valeur erronée). C'est une limitation de la technologie ODBC.

Voir aussi

GetDatabaseBlob() , GetDatabaseDouble() , GetDatabaseFloat() , GetDatabaseString() , GetDatabaseLong()

OS Supportés

Tous

86.22 GetDatabaseString

Syntaxe

```
Resultat\$ =
    GetDatabaseString(#BaseDeDonnees,
    Colonne)
```

Description

Renvoie le contenu d'un champ au format chaîne de caractères.

Arguments

#BaseDeDonnees La base de données à utiliser.

Colonne La colonne à utiliser. DatabaseColumnIndex() est disponible pour obtenir l'index d'une colonne.

Valeur de retour

Renvoie une chaîne de caractères.

Remarques

Cette commande n'est valide qu'après un succès de FirstDatabaseRow() , PreviousDatabaseRow() ou NextDatabaseRow() .
Pour connaître le type d'un champ il faut utiliser DatabaseColumnType() .
Note : Cette commande peut être appelée seulement une fois par colonne (il faut stocker le résultat dans une variable s'il doit être utilisé plusieurs fois, car un nouvel appel sur la même colonne renverrait une valeur erronée). C'est une limitation de la technologie ODBC.

Voir aussi

GetDatabaseBlob() , GetDatabaseDouble() , GetDatabaseFloat() , GetDatabaseLong() , GetDatabaseQuad()

OS Supportés

Tous

86.23 CheckDatabaseNull

Syntaxe

```
Resultat =
    CheckDatabaseNull(#BaseDeDonnees,
    Colonne)
```

Description

Vérifie si le contenu d'un champ est vide.

Arguments

#BaseDeDonnees La base de données à utiliser.

Colonne La colonne à utiliser. DatabaseColumnIndex() est disponible pour obtenir l'index d'une colonne.

Valeur de retour

Renvoie #True si la colonne est vide, #False sinon.

Remarques

Cette commande n'est valide qu'après un succès de FirstDatabaseRow() , PreviousDatabaseRow() ou NextDatabaseRow() .
Pour connaître le type d'un champ il faut utiliser DatabaseColumnType() .
Note : Cette commande peut être appelée seulement une fois par colonne (il faut stocker le résultat dans une variable s'il doit être utilisé plusieurs fois, car un nouvel appel sur la même colonne renverrait une valeur erronée). C'est une limitation de la technologie ODBC.

Voir aussi

GetDatabaseBlob() , GetDatabaseDouble() , GetDatabaseFloat() , GetDatabaseLong() , GetDatabaseQuad()

OS Supportés

Tous

86.24 IsDatabase

Syntaxe

```
Resultat =
    IsDatabase(#BaseDeDonnees)
```

Description

Teste si une base de données est correctement initialisée.

Arguments

#**BaseDeDonnees** La base de données à utiliser.

Valeur de retour

Renvoie une valeur non nulle si l'objet est valide, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.

Voir aussi

OpenDatabase() , OpenDatabaseRequester()

OS Supportés

Tous

86.25 NextDatabaseDriver

Syntaxe

```
Resultat =
    NextDatabaseDriver()
```

Description

Renvoie les informations sur le prochain pilote de base de données.

Arguments

Aucun.

Valeur de retour

Si le résultat est nul, il n'y a plus aucun pilote suivant disponible.

Remarques

Cette commande doit être appelée après ExamineDatabaseDrivers() .
Pour obtenir les informations sur le pilote, utiliser DatabaseDriverName() et DatabaseDriverDescription() .
Les bases de données SQLite ne supportent pas cette commande.

Voir aussi

ExamineDatabaseDrivers() , DatabaseDriverName() , DatabaseDriverDescription()

OS Supportés

Tous

86.26 NextDatabaseRow

Syntaxe

```
Resultat =
    NextDatabaseRow(#BaseDeDonnees)
```

Description

Renvoie les informations relatives à l'enregistrement suivant.

Arguments

#**BaseDeDonnees** La base de données à utiliser.

Valeur de retour

Si le résultat est nul, il n'y a plus d'enregistrement suivant (fin de fichier).

Remarques

Pour accéder aux informations à l'intérieur de l'enregistrement, utiliser GetDatabaseLong() , GetDatabaseFloat() et GetDatabaseString() .

Voir aussi

GetDatabaseBlob() , GetDatabaseDouble() , GetDatabaseFloat() , GetDatabaseLong() , GetDatabaseQuad() , GetDatabaseString()

OS Supportés

Tous

86.27 OpenDatabase

Syntaxe

```
Resultat =
    OpenDatabase(#BaseDeDonnees,
    NomBaseDeDonnes$,
    Utilisateur$, MotdePasse$
    [, Plugin])
```

Description

Ouvrir une base de données.

Arguments

#BaseDeDonnees La base de données à utiliser.
#PB_Any peut être utiliser pour autogénérer ce numéro.

NomBaseDeDonnes$ Le nom de la base de données à ouvrir.

Utilisateur$ Le nom d'utilisateur de connexion.

MotdePasse$ Le mot de passe de connexion.
Le mot de passe peut être une chaîne vide si aucun mot de passe n'est requis.

Plugin (optionnel)

```
      #PB_Database_ODBC
    : La base de données
    utilisera ODBC
    (UseODBCDatabase()
  doit avoir été appelé).
   #PB_Database_SQLite     :
    La base de données
    utilisera SQLite
    (UseSQLiteDatabase()
  doit avoir été appelé).
   #PB_Database_PostgreSQL:
    La base de données
    utilisera PostgreSQL
    (UsePostgreSQLDatabase()
  doit avoir été appelé).
   #PB_Database_MySQL      :
    La base de données
    utilisera MySQL
    (UseMySQLDatabase()
  doit avoir été appelé).
```

Si 'Plugin' n'est pas spécifié, le premier plugin enregistré sera utilisé.

Valeur de retour

Renvoie une valeur non nulle en cas de connexion, zéro sinon. Dans ce cas, la base n'a pas pu être trouvée ou le compte

utilisateur n'est pas valable et la description exacte de l'erreur peut être récupérée grâce à DatabaseError() .
Si #PB_Any a été utilisé pour le paramètre #BaseDeDonnees,le nombre généré est renvoyé.

Remarques

Pour déclarer une base de données ODBC, lire le document d'aide de Windows.
Si une autre base de données a déjà été ouverte sous le même numéro, la base précédement gérée par ce numéro est automatiquement fermée.

Voir aussi

OpenDatabaseRequester() , CloseDatabase() , UseODBCDatabase() , UseSQLiteDatabase() , UsePostgreSQLDatabase() , UseMySQLDatabase()

OS Supportés

Tous

86.28 OpenDatabaseRequester

Syntaxe

```
Resultat =
    OpenDatabaseRequester(#BaseDeDonnees
    [, Plugin])
```

Description

Ouvre la boite de dialogue standard Windows ODBC pour sélectionner une base de données.

Arguments

#BaseDeDonnees La base de données à utiliser.

Plugin (optionnel)
```
      #PB_Database_ODBC
    : La base de données
    utilisera ODBC
    (UseODBCDatabase()
 doit avoir été appelé).
```

Si 'Plugin' n'est pas spécifié, le premier plugin (ODBC, SQLite ou PostgreSQL) enregistré sera utilisé.

Valeur de retour

Renvoie une valeur non nulle en cas de connexion, zéro sinon. Dans ce cas, la base n'a pas pu être trouvée ou le compte utilisateur n'est pas valable et la description exacte de l'erreur peut être récupérée grâce à DatabaseError()).
Si #PB_Any a été utilisé pour le paramètre #BaseDeDonnees, le nombre généré est renvoyé.

Remarques

C'est une commande spécifique aux bases de données ODBC . Les bases de données SQLite ne supportent pas cette commande. Note : cette commande n'est pas supportée sur Linux et MacOS X, elle renverra zéro.

Voir aussi

OpenDatabase() , CloseDatabase() ,

OS Supportés

Tous

86.29 PreviousDatabaseRow

Syntaxe

```
Resultat =
    PreviousDatabaseRow(#BaseDeDonnees)
```

Description

Renvoie les informations relatives à l'enregistrement précédent.

Arguments

#BaseDeDonnees La base de données à utiliser.

Valeur de retour

Si le résultat est zéro alors il n'y a plus d'enregistrement précédent disponible (début de fichier).

Remarques

Attention, l'option #PB_Database_DynamicCursor doit être spécifiée pour que la fonction DatabaseQuery() puisse fonctionner.

Si cette commande renvoie zéro même s'il y a des enregistrements avant l'enregistrement courant, alors cette commande n'est pas supportée par le driver ODBC. En effet, un driver ODBC n'est pas obligé d'implémenter cette commande (contrairement à NextDatabaseRow()) pour être conforme à la norme ODBC. Bien entendu, si cette commande fonctionne correctement avec un driver, elle fonctionnera correctement sur tous les ordinateurs utilisant ce driver. C'est une commande spécifique aux bases de données ODBC . Les bases de données SQLite ne supportent pas cette commande.

Voir aussi

GetDatabaseBlob() , GetDatabaseDouble() , GetDatabaseFloat() , GetDatabaseLong() , GetDatabaseQuad() , GetDatabaseString()

OS Supportés

Tous

86.30 SetDatabaseBlob

Syntaxe

```
SetDatabaseBlob(#BaseDeDonnees,
    Index, *Memoire,
    TailleMemoire)
```

Description

Indique le blob à insérer lors de la prochaine utilisation de DatabaseUpdate() .

Arguments

#BaseDeDonnees La base de données à utiliser.

Index Commence à 0 et indique sur lequel des paramètres non-définis, le blob doit être associé.
La syntaxe SQL pour spécifier un paramètre indéfini varie en fonction de la base de données.
Pour voir comment procéder, consulter les exemples ci-dessous.

***Memoire** L'adresse du blob.

TailleMemoire La taille du blob en octets.

Valeur de retour

Aucune.

Exemple : SQLite, MySQL et ODBC

```
1  ; SQLite, MySQL et ODBC
   partagent la même syntaxe
   pour insérer un blob.
   C'est indiqué par le
   caractère '?'
2  ;
3  ; La base de données doit
   être connectée et avoir
   une table PHOTOS avec 3
   colonnes (BLOB,
   VARCHAR(255), BLOB)
4  ;
5  SetDatabaseBlob(0, 0,
   ?Picture, PictureLength)
6  SetDatabaseBlob(0, 1,
   ?SmallPicture,
   SmallPictureLength)
7  DatabaseUpdate(0, "INSERT
   INTO PHOTOS (picture,
   name, small_picture)
   values (?, 'my
   description', ?);")
```

Exemple : PostgreSQL

```
1  ; PostgreSQL utilise une
   autre syntaxe: $1, $2..
   dans la requête pour
   indiquer les paramètres
   non-définis
2  ;
3  ; La base de données doit
   être connectée et avoir
   une table PHOTOS avec 3
   colonnes (BYTEA,
   VARCHAR(255), BYTEA)
4  ;
5  SetDatabaseBlob(0, 0,
   ?Picture, PictureLength)
6  SetDatabaseBlob(0, 1,
   ?SmallPicture,
   SmallPictureLength)
7  DatabaseUpdate(0, "INSERT
   INTO PHOTOS (picture,
   name, small_picture)
   values ($1, 'my
   description', $2);")
```

Note : PostgreSQL utilise le type 'BYTEA' pour stocker les blobs. La conversion nécessaire pour stocker du binaire dans ce type de colonne rend souvent le blob très volumineux. Un bon moyen d'éviter cela est de l'encoder

précédemment avec Base64Encoder() avant de l'insérer dans la base de données.

Voir aussi

DatabaseUpdate() , GetDatabaseBlob()

OS Supportés

Tous

86.31 UseMySQLDatabase

Syntaxe

```
UseMySQLDatabase([NomBibliotheque$])
```

Description

Initialise lenvironnement de base de données MySQL et MariaDB pour une utilisation ultérieure.

Arguments

NomBibliotheque$ (optionnel) Nom de fichier (et chemin si nécessaire) de la bibliothèque dynamique à utiliser. Comme la plupart des distributions Linux sont livrées avec le paquet libmysql.so, il peut être configuré avec un nom de fichier correct, de sorte que libmaria.so ne doit pas nécessairement être associé à l'exécutable. Si ce paramètre n'est pas spécifié, 'libmariadb.dll' (Windows), 'libmariadb.so' (Linux) ou 'libmariadb.dylib' (OSX) seront utilisés.

Valeur de retour

Aucune.

Remarques

MySQL et MariaDB (un fork open source de MySQL) (fork : Logiciel issu dune scission dun projet initial unique, et qui partage avec lui une part de son code source) sont de puissants gestionnaires de bases de données basés sur serveur qui prennent en charge des bases de données très volumineuses et des accès simultanés élevés. PureBasic utilise la bibliothèque opensource MariaDB pour connecter des bases de données MySQL et MariaDB sans aucun problème, ce qui peut être utilisé dans des applications commerciales sans licence supplémentaire. Lorsque vous

envoyez votre programme PureBasic, vous devrez ajouter 'libmariadb.dll' (Windows), 'libmariadb.so' (Linux) ou 'libmariadb.dylib' (OSX) trouvé dans le répertoire 'PureBasic/Compilers' de votre installation ou de votre paquet Purebasic. Il n'y a pas de pilote supplémentaire à installer, tout est prêt pour connecter un serveur MySQL ou MariaDB. Pour plus d'informations à propos de MariaDB : https ://mariadb.org/.
Une base de données MySQL ou MariaDB doit être connectée à l'aide de OpenDatabase() avant d'utiliser toutes fonctions de base de données.
Les paramètres spécifiques à MySQL doivent être passés dans le paramètre 'NomBaseDeDonnes$' de OpenDatabase() :

```
- host: Nom de l'hôte ou
  adresse IP à laquelle se
  connecter.
- port: Numéro de port
  auquel se connecter sur le
  serveur hôte.
- dbname: Nom de la base de
  données.
```

Exemple

```
1  UseMySQLDatabase()
2
3  ; Vous devez avoir un
     serveur fonctionnant sur
     localhost
4  ;
5  If OpenDatabase(0,
     "host=localhost port=3306
     dbname=test",
     "utilisateur",
     "motdepasse")
6    Debug "Connecté à MySQL"
7  Else
8    Debug "La connexion a
     échoué: "+DatabaseError()
9  EndIf
```

Voir aussi

OpenDatabase() , UseSQLiteDatabase() , UseODBCDatabase() , UsePostgreSQLDatabase()

OS Supportés

Tous

86.32 UsePostgreSQLDatabase

Syntaxe

```
UsePostgreSQLDatabase()
```

Description

Initialise l'environnement de base de données PostgreSQL pour un usage ultérieur.

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

PostgreSQL est un serveur de base de données très puissant qui supporte les bases de données très grandes, et une gestion des accès concurrents avancée. Il est réellement libre d'utilisation même dans un projet commercial, contrairement à MySQL qui nécessite une licence pour l'utiliser dans un programme non-GPL. Il n'y a pas de pilotes supplémentaires à installer, tout ce qui est nécessaire à la connexion au serveur PostgreSQL est présent. Pour plus d'informations à propos de PostgreSQL : http ://www.postgresql.org.
Une connexion à PostgreSQL doit être établie à l'aide de la fonction OpenDatabase() avant de pouvoir utiliser les autres fonctions relatives aux bases de données. Des commandes spécifiques à PostgresSQL peuvent être passées dans le paramètre 'NomBaseDeDonnes$' de OpenDatabase() :

```
- host: Nom d'ordinateur
  hôte pour se connecter.
- hostaddr: Adresse IP de
  l'ordinateur hôte pour se
  connecter.
- port: Numéro du port à
  utiliser pour la connexion.
- dbname: Le nom de la base
  de données. Par défaut, le
  même nom que l'utilisateur.
- connect_timeout: Attente
  maximum pour la connexion,
  en seconde (nombre décimal
  entier). Zéro ou non
  spécifié indique une
  attente infinie.
```

Il n'est pas recommandé d'utiliser une attente inférieure à 2 secondes.

Exemple

```
1  UsePostgreSQLDatabase()
2
3  ; Le serveur doit tourner sur la machine locale.
4  ;
5  If OpenDatabase(0, "host=localhost port=5432", "utilisateur", "motdepasse")
6    Debug "Connecté à PostgreSQL"
7  Else
8    Debug "La connexion a echoué: "+DatabaseError()
9  EndIf
```

Voir aussi

OpenDatabase() , UseSQLiteDatabase() , UseODBCDatabase() , UseMySQLDatabase()

OS Supportés

Tous

86.33 UseSQLiteDatabase

Syntaxe

```
UseSQLiteDatabase([NomBibliotheque$])
```

Description

Initialise l'environnement de base de données SQLite pour un usage ultérieur.

Arguments

NomBibliotheque$ (optionnel) Nom de fichier (et chemin si nécessaire) de la bibliothèque dynamique à utiliser.

Valeur de retour

Aucune.

Remarques

SQLite est un gestionnaire de base de données sans serveur, utilisant un fichier comme stockage. Il n'y a rien à déployer ou à configurer, il est tout de suite opérationnel. SQLite est largement utilisé dans l'industrie et est considéré comme un des meilleurs gestionnaire de base de données embarqué. Pour plus d'informations : http ://www.sqlite.org.
Pour créer une nouvelle base de données vide, il suffit de créer un fichier vide à l'aide de CreateFile() . Les commandes SQL sont maintenant disponibles pour créer les tables et insérer des enregistrements.
La base SQLite doit être ouverte avec OpenDatabase() avant de pouvoir utiliser les commandes de cette bibliothèque.
En utilisant "NomBibliotheque$", vous pourrez ainsi utiliser la dernière version du fichier 'dll' (so, dylib) sans attendre une mise à jour PB.
Sans ce fichier, c'est la bibliothèque statique qui sera utilisée comme d'habitude et l'exécutable sera plus gros.
Aucun message d'erreur n'est levé si le fichier est absent.

Exemple

```
1   UseSQLiteDatabase()
2
3   Filename$ =
     OpenFileRequester("Choisissez
     le nom d'un fichier",
     "PureBasic.sqlite",
     "*.sqlite|*.sqlite", 0)
4
5   If CreateFile(0, Filename$)
6     Debug "Fichier de base de
     données créé"
7     CloseFile(0)
8   EndIf
9
10  If OpenDatabase(0,
     Filename$, "", "")
11    Debug "Connecté à
     PureBasic.sqlite"
12    If DatabaseUpdate(0,
     "CREATE TABLE info (test
     VARCHAR(255));")
13      Debug "Table créée"
14    EndIf
15  EndIf
```

Voir aussi

OpenDatabase() ,
UsePostgreSQLDatabase() ,
UseODBCDatabase() ,
UseMySQLDatabase()

OS Supportés

Tous

86.34 UseODBCDatabase

Syntaxe

```
Resultat = UseODBCDatabase()
```

Description

Initialise l'environnement de base de données ODBC pour un usage ultérieur.

Arguments

Aucun.

Valeur de retour

Si le résultat est 0, ODBC n'est pas disponible ou dans une version non compatible (ODBC 3.0 ou supérieur est requis) et tous les appels de fonctions de base de données doivent être inhibés.

Remarques

Une fois l'environnement initialisé, une base de données doit être ouverte en utilisant OpenDatabase() avec un nom de base de données ODBC enregistré comme base de données ou OpenDatabaseRequester() avant d'utiliser toutes les autres fonctions des bases de données.
Il peut être utile de lister les pilotes (drivers) disponibles sur la machine à l'aide de la commande ExamineDatabaseDrivers() .

Exemple

```
1  UseODBCDatabase()
2
3  If OpenDatabase(0,
     "MySQL-ODBC",
     "utilisateur",
     "motdepasse")
4    Debug "Connecté à MySQL"
5  Else
```

```
6     Debug "Pas de Connexion : " + DatabaseError()
7   EndIf
```

Voir aussi

OpenDatabase() , UseSQLiteDatabase() , UsePostgreSQLDatabase() , UseMySQLDatabase()

OS Supportés

Tous

86.35 SetDatabaseString

Syntaxe

```
SetDatabaseString(#BaseDeDonnees,
    IndexDeDeclaration,
    Valeur$)
```

Description

Définit une variable de liaison de type chaîne de caractères pour le prochain appel à DatabaseQuery() ou à DatabaseUpdate() .

Arguments

#Database La base de données à utiliser.

IndexDeDeclaration L'indice de la variable de liaison dans la déclaration. La première variable a l'index 0.

Valeur$ La valeur à utiliser pour la variable de liaison.

Valeur de retour

Aucune.

Remarques

Lier des variables permet la création plus facile de déclarations avec des données, car il n'y a pas besoin d'ajouter les données dans la chaîne. La déclaration de type chaîne peut contenir les espaces réservés et les données sont liées avant l'exécution de la déclaration. Cette méthode permet également d'éviter les vulnérabilités dues à une éventuelle injection de code SQL qui peut être faite si les données (telles que des chaînes) sont directement insérées dans le texte de la déclaration. Tant que la

déclaration ne contient que l'espace réservé, il n'y a pas de danger.
Voir DatabaseQuery() et DatabaseUpdate() pour des exemples sur comment spécifier les variables de liaison dans une instruction SQL.

Voir aussi

SetDatabaseLong() , SetDatabaseQuad() , SetDatabaseFloat() , SetDatabaseDouble() SetDatabaseBlob() , SetDatabaseNull() , DatabaseQuery() , DatabaseUpdate()

OS Supportés

Tous

86.36 SetDatabaseLong

Syntaxe

```
SetDatabaseLong(#BaseDeDonnees,
    IndexDeDeclaration, Valeur)
```

Description

Définit une variable de liaison de type Long pour le prochain appel à DatabaseQuery() ou à DatabaseUpdate() .

Arguments

#Database La base de données à utiliser.

IndexDeDeclaration L'indice de la variable de liaison dans la déclaration. La première variable a l'index 0.

Valeur La valeur à utiliser pour la variable de liaison.

Valeur de retour

Aucune.

Remarques

Lier des variables permet la création plus facile de déclarations avec des données, car il n'y a pas besoin d'ajouter les données dans la chaîne. La déclaration de type chaîne peut contenir les espaces réservés et les données sont liées avant l'exécution de la déclaration. Cette méthode permet également d'éviter les vulnérabilités dues à une éventuelle injection de code SQL qui peut être faite si les données (telles que des chaînes) sont directement insérées dans le

texte de la déclaration. Tant que la déclaration ne contient que l'espace réservé, il n'y a pas de danger.
Voir DatabaseQuery() et DatabaseUpdate() pour des exemples sur comment spécifier les variables de liaison dans une instruction SQL.

Voir aussi

SetDatabaseString() , SetDatabaseQuad() , SetDatabaseFloat() , SetDatabaseDouble() SetDatabaseBlob() , SetDatabaseNull() , DatabaseQuery() , DatabaseUpdate()

OS Supportés

Tous

86.37 SetDatabaseQuad

Syntaxe

```
SetDatabaseQuad(#BaseDeDonnees,
    IndexDeDeclaration,
    Valeur.q)
```

Description

Définit une variable de liaison de type Quad pour le prochain appel à DatabaseQuery() ou à DatabaseUpdate() .

Arguments

#Database La base de données à utiliser.

IndexDeDeclaration L'indice de la variable de liaison dans la déclaration. La première variable a l'index 0.

Valeur.q La valeur à utiliser pour la variable de liaison.

Valeur de retour

Aucune.

Remarques

Lier des variables permet la création plus facile de déclarations avec des données, car il n'y a pas besoin d'ajouter les données dans la chaîne. La déclaration de type chaîne peut contenir les espaces réservés et les données sont liées avant l'exécution de la déclaration. Cette méthode permet également d'éviter les vulnérabilités dues à une éventuelle injection de code SQL qui

peut être faite si les données (telles que des chaînes) sont directement insérées dans le texte de la déclaration. Tant que la déclaration ne contient que l'espace réservé, il n'y a pas de danger.
Voir DatabaseQuery() et DatabaseUpdate() pour des exemples sur comment spécifier les variables de liaison dans une instruction SQL.

Voir aussi

SetDatabaseString() , SetDatabaseLong() , SetDatabaseFloat() , SetDatabaseDouble() SetDatabaseBlob() , SetDatabaseNull() , DatabaseQuery() , DatabaseUpdate()

OS Supportés

Tous

86.38 SetDatabaseFloat

Syntaxe

```
SetDatabaseFloat(#BaseDeDonnees,
    IndexDeDeclaration,
    Valeur.f)
```

Description

Définit une variable de liaison de type Float pour le prochain appel à DatabaseQuery() ou à DatabaseUpdate() .

Arguments

#Database La base de données à utiliser.

IndexDeDeclaration L'indice de la variable de liaison dans la déclaration. La première variable a l'index 0.

Valeur.f La valeur à utiliser pour la variable de liaison.

Valeur de retour

Aucune.

Remarques

Lier des variables permet la création plus facile de déclarations avec des données, car il n'y a pas besoin d'ajouter les données dans la chaîne. La déclaration de type chaîne peut contenir les espaces réservés et les données sont liées avant l'exécution de la déclaration. Cette méthode permet

également d'éviter les vulnérabilités dues à une éventuelle injection de code SQL qui peut être faite si les données (telles que des chaînes) sont directement insérées dans le texte de la déclaration. Tant que la déclaration ne contient que l'espace réservé, il n'y a pas de danger.
Voir DatabaseQuery() et DatabaseUpdate() pour des exemples sur comment spécifier les variables de liaison dans une instruction SQL.

Voir aussi

SetDatabaseString() , SetDatabaseLong() , SetDatabaseQuad() , SetDatabaseDouble() SetDatabaseBlob() , SetDatabaseNull() , DatabaseQuery() , DatabaseUpdate()

OS Supportés

Tous

86.39 SetDatabaseDouble

Syntaxe

```
SetDatabaseDouble(#BaseDeDonnees,
    IndexDeDeclaration,
    Valeur.d)
```

Description

Définit une variable de liaison de type Double pour le prochain appel à DatabaseQuery() ou à DatabaseUpdate() .

Arguments

#Database La base de données à utiliser.

IndexDeDeclaration L'indice de la variable de liaison dans la déclaration. La première variable a l'index 0.

Valeur.d La valeur à utiliser pour la variable de liaison.

Valeur de retour

Aucune.

Remarques

Lier des variables permet la création plus facile de déclarations avec des données, car il n'y a pas besoin d'ajouter les données dans la chaîne. La déclaration de type chaîne peut contenir les espaces réservés et

les données sont liées avant l'exécution de la déclaration. Cette méthode permet également d'éviter les vulnérabilités dues à une éventuelle injection de code SQL qui peut être faite si les données (telles que des chaînes) sont directement insérées dans le texte de la déclaration. Tant que la déclaration ne contient que l'espace réservé, il n'y a pas de danger.
Voir DatabaseQuery() et DatabaseUpdate() pour des exemples sur comment spécifier les variables de liaison dans une instruction SQL.

Voir aussi

SetDatabaseString() , SetDatabaseLong() , SetDatabaseQuad() , SetDatabaseFloat() SetDatabaseBlob() , SetDatabaseNull() , DatabaseQuery() , DatabaseUpdate()

OS Supportés

Tous

86.40 SetDatabaseNull

Syntaxe

```
SetDatabaseNull(#BaseDeDonnees,
    IndexDeDeclaration)
```

Description

Définit une variable de liaison à une valeur NULL pour le prochain appel à DatabaseQuery() ou à DatabaseUpdate() .

Arguments

#Database La base de données à utiliser.

IndexDeDeclaration L'indice de la variable de liaison dans la déclaration. La première variable a l'index 0.

Valeur de retour

Aucune.

Remarques

Voir DatabaseQuery() et DatabaseUpdate() pour des exemples sur comment spécifier les variables de liaison dans une instruction SQL.

Voir aussi

SetDatabaseString() , SetDatabaseLong() , SetDatabaseQuad() , SetDatabaseFloat() , SetDatabaseDouble() SetDatabaseBlob() , DatabaseQuery() , DatabaseUpdate()

OS Supportés

Tous

Chapitre 87

Date

Généralités

La bibliothèque Date permet de gérer le temps et les dates depuis l'année 1970 jusqu'à l'année 2038, en utilisant le modèle Unix 32 bits (nombre de secondes écoulées depuis le 1er janvier 1970).
Note : Seulement entre le 01 janvier 1970 à 00h 00m
00s et le 19 janvier 2038 à 03h 14m 07s". Voir https ://fr.wikipedia.org/wiki/Bug_de_l%27an_2038.

OS Supportés

Tous

87.1 AddDate

Syntaxe

```
Resultat = AddDate(Date,
    Plage, Valeur)
```

Description

Ajouter une quantité de temps à une date.

Arguments

Date La date à utiliser.

Plage #PB_Date_Year : Ajoute 'Valeur' années à la date
#PB_Date_Month : Ajoute 'Valeur' mois à la date
#PB_Date_Week : Ajoute 'Valeur' semaines à la date
#PB_Date_Day : Ajoute 'Valeur' jours à la date
#PB_Date_Hour : Ajoute 'Valeur' heures à la date

```
#PB_Date_Minute : Ajoute
  'valeur' minutes à la
  date
#PB_Date_Second : Ajoute
  'valeur' secondes à la
  date
```

Note : l'utilisation de #PB_Date_Month peut entrainer un arrondi automatique du jour si nécessaire : Par exemple, ajouter un mois au '31 mars 2008' renverra '30 avril 2008', car il n'y a pas 31 jours en avril.

Valeur La quantité à ajouter à la date. Une valeur négative est autorisée et permet de soustraire deux dates.

Valeur de retour

Renvoie la nouvelle date.
Si les paramètres donnés ne sont pas valides ou en dehors de la plage de dates supportée, -1 sera renvoyé.

Exemple

```
1
2   Debug
      FormatDate("%dd/%mm/%yyyy",
      AddDate(Date(),
      #PB_Date_Year, 2))   ;
      Renvoie la date courante
      augmentée de 2 ans
3   Debug
      FormatDate("%dd/%mm/%yyyy",
      AddDate(Date(),
      #PB_Date_Year, -2)) ;
      Renvoie la date courante
      diminuée de 2 ans
```

Voir aussi

Date() , FormatDate()

OS Supportés

Tous

87.2 Date

Syntaxe

```
Resultat = Date([Annee, Mois,
    Jour, Heure, Minute,
    Seconde])
```

Description

Renvoie la valeur du temps système (nombre de secondes écoulées depuis le '01/01/1970 0 :00 :00').

Arguments

Annee, Mois, Jour, Heure, Minute, Seconde (optionnel)
Si les variables Annee, Mois, Jour, Heure, Minute et Seconde sont spécifiées, la représentation en secondes de cette date est renvoyée.

Valeur de retour

Renvoie le nombre de secondes écoulée.
Si les paramètres donnés ne sont pas valides ou en dehors de la plage de dates supportée, -1 sera renvoyé.

Exemple

```
1  Debug Date() /
     (3600*24*365)  ; Affiche
     le nombre d'années depuis
     01/01/1970 et maintenant
2  Debug Date(1999, 12, 31,
     23, 59, 59)  ; Affiche
     '946684799' (nombre de
     secondes entre 01/01/1970
     0:00:00 et 12/31/1999
     23:59:59)
```

Note : Le temps et les dates supportées vont de '1970-01-01, 00 :00 :00' pour le minimum à '2038-01-19, 03 :14 :07' pour le maximum.

Voir aussi

FormatDate() , Year() , Month() , Day() , Hour() , Minute() , Second()

OS Supportés

Tous

87.3 Day

Syntaxe

```
Resultat = Day(Date)
```

Description

Renvoie le jour.

Arguments

Date La date à utiliser.

Valeur de retour

Renvoie le jour de la date spécifiée ou -1 en cas d'erreur.
Le résultat est compris entre 1 et 31.

Exemple

```
Debug Day(Date(2022, 10, 3, 0, 0, 0))  ; Le résultat est '3'.
```

Voir aussi

Date() , Year() , Month() , Hour() , Minute() , Second()

OS Supportés

Tous

87.4 DayOfWeek

Syntaxe

```
Resultat = DayOfWeek(Date)
```

Description

Renvoie le rang du jour de la semaine.

Arguments

Date La date à utiliser.

Valeur de retour

Renvoie le rang du jour de la semaine ou -1 en cas d'erreur.
Le résultat est compris entre 0 et 6.

```
0 : Dimanche
1 : Lundi
2 : Mardi
3 : Mercredi
4 : Jeudi
5 : Vendredi
6 : Samedi
```

Exemple

```
Debug DayOfWeek(Date(2026, 10, 30, 0, 0, 0))  ; Renvoie '1' pour Lundi.
```

Voir aussi

FormatDate() , Year() , Month() , Day() , Hour() , Minute() , Second()

OS Supportés

Tous

87.5 DayOfYear

Syntaxe

```
Resultat = DayOfYear(Date)
```

Description

Renvoie le nombre de jours écoulés depuis le début de l'année.

Arguments

Date La date à utiliser.

Valeur de retour

Renvoie le nombre de jours écoulés depuis le début de l'année d'une date spécifiée ou -1 en cas d'erreur.
Le résultat est compris entre 1 et 366.

Exemple

```
1  Debug DayOfYear(Date(2022, 2, 1, 0, 0, 0))  ; Le résultat est '32'. (31 jours pour janvier + 1)
```

Voir aussi

FormatDate() , Year() , Month() , Day() , Hour() , Minute() , Second()

OS Supportés

Tous

87.6 Month

Syntaxe

```
Resultat = Month(Date)
```

Description

Renvoie le mois.

Arguments

Date La date à utiliser.

Valeur de retour

Renvoie le mois de la date spécifiée ou -1 en cas d'erreur.
Le résultat est compris entre 1 et 12.

Exemple

```
1  Debug Month(Date(2022, 10, 3, 0, 0, 0))  ; Le résultat est '10'.
```

Voir aussi

FormatDate() , Year() , Day() , Hour() , Minute() , Second()

OS Supportés

Tous

87.7 Year

Syntaxe

```
Resultat = Year(Date)
```

Description

Renvoie l'année.

Arguments

Date La date à utiliser.

Valeur de retour

Renvoie l'année de la date spécifiée ou -1 en cas d'erreur.
Le résultat est compris entre 1970 et 2034.

Exemple

```
1  Debug Year(Date(2022, 10, 3, 0, 0, 0))  ; Le résultat est '2022'.
```

Voir aussi

FormatDate() , Month() , Day() , Hour() , Minute() , Second()

OS Supportés

Tous

87.8 Hour

Syntaxe

```
Resultat = Hour(Date)
```

Description

Renvoie l'heure.

Arguments

Date La date à utiliser.

Valeur de retour

Renvoie l'heure de la date spécifiée ou -1 en cas d'erreur.
Le résultat est compris entre 0 et 23.

Exemple

```
1  Debug Hour(Date(1970, 1, 1, 11, 3, 45))  ; Le résultat est '11'.
```

OS Supportés

Tous

87.9 Minute

Syntaxe

```
Resultat = Minute(Date)
```

Description

Renvoie les minutes.

Arguments

Date La date à utiliser.

Valeur de retour

Renvoie les minutes de la date spécifiée ou -1 en cas d'erreur.
Le résultat est compris entre 0 et 59.

Exemple

```
1   Debug Minute(Date(1970, 1,
      1, 11, 3, 45))   ; Le
      résultat est '3'.
```

Voir aussi

FormatDate() , Year() , Month() , Day() , Hour() , Second()

OS Supportés

Tous

87.10 Second

Syntaxe

```
Resultat = Second(Date)
```

Description

Renvoie les secondes.

Arguments

Date La date à utiliser.

Valeur de retour

Renvoie les secondes de la date spécifiée ou -1 en cas d'erreur.
Le résultat est compris entre 0 et 59.

Exemple

```
1   Debug Second(Date(1970, 1,
      1, 11, 3, 45))   ; Le
      résultat est '45'.
```

Voir aussi

FormatDate() , Year() , Month() , Day() , Hour() , Minute()

OS Supportés

Tous

87.11 FormatDate

Syntaxe

```
Resultat\$ =
    FormatDate(Masque$, Date)
```

Description

Renvoie une date formatée par l'utilisateur.

Arguments

Masque$ Une chaîne de caractères indiquant comment formater et afficher la date.

```
%yyyy: L'année sera
 affichée avec 4 chiffres.
 %yy: L'année sera
affichée avec 2 chiffres.
 %mm: Le mois sera
affiché avec 2 chiffres.
 %dd: Le jour sera
affiché avec 2 chiffres.
 %hh: L'heure sera
affichée avec 2 chiffres.
 %ii: Les minutes seront
affichées avec 2
chiffres.
 %ss: Les secondes seront
affichées avec 2
chiffres.
```

Date La date à utiliser.

Valeur de retour

Renvoie la date sous forme de chaîne de caractères correspondant au masque spécifié.
Renvoie "0" ou "-1" en cas d'erreur

Exemple

```
1  Debug FormatDate("A=%yyyy,
    M= %mm, J=%dd", Date()) ;
    Affiche la date sous la
    forme
2                                                              ;
    "A=2012, M=12, J=21"
3
4  Debug
    FormatDate("%dd/%mm/%yyyy",
    Date())          ; Affiche
    la date sous la forme
5                                                              ;
     "21/12/2012"
6
```

```
7  Debug
     FormatDate("%hh:%ii:%ss",
     Date())  ; affiche le
     temps selon un format
     00:00:00
```

Note : Le temps et les dates supportées vont de '1970-01-01, 00 :00 :00' pour le minimum à '2038-01-19, 03 :14 :07' pour le maximum.

Voir aussi

Date() , ParseDate()

OS Supportés

Tous

87.12 ParseDate

Syntaxe

```
Resultat = ParseDate(Masque$,
   Date$)
```

Description

Transforme une date donnée sous la forme d'une chaîne de caractères en valeur numérique.

Arguments

Masque$ Une chaîne de caractères indiquant comment formater la date.

```
%yyyy: L'année sera
 affichée avec 4 chiffres.
 %yy: L'année sera
affichée avec 2 chiffres.
 %mm: Le mois sera
affiché avec 2 chiffres.
 %dd: Le jour sera
affiché avec 2 chiffres.
 %hh: L'heure sera
affichée avec 2 chiffres.
 %ii: Les minutes seront
affichées avec 2
chiffres.
 %ss: Les secondes seront
affichées avec 2
chiffres.
```

Date$ La date à tester.

Valeur de retour

Renvoie la date représentant la chaîne à analyser.
Si la date est incorrecte, la valeur renvoyée est -1.
Cette fonction est particulièrement utile pour retrouver et traiter des dates stockées, dans un fichier de type journal d'évènements par exemple.

Exemple

```
1  Debug
     ParseDate("%yy/%mm/%dd",
     "20/12/01") ; Renvoie la
     date numérique
     correspondant à "01/12/20"
2  Debug
     ParseDate("%dd/%mm/%yyyy",
     "01/07/2020") ; Renvoie la
     date numérique
     correspondant à
     "01/07/2020"
```

Voir aussi

Date() , FormatDate()

OS Supportés

Tous

Chapitre 88

Debugger

Généralités

La bibliothèque 'Debugger' fournit les fonctions nécessaires au contrôle du débogueur , par exemple pour vider le contenu de la fenêtre de déboguage, ou pour ouvrir le visualisateur de mémoire à une adresse mémoire donnée. Tous les outils de déboguage mentionnés dans cette bibliothèque sont décrits dans le chapitre outils de déboguage .
Les fonctions de cette bibliothèque ne sont compilées dans l'exécutable que si le débogueur est activé. Si le débogueur est désactivé, les appels sont complètement ignorés et ne sont pas intégrés dans l'exécutable final.
Il y a aussi des mots clés spécifiques au contrôle du débogueur à partir du code.

OS Supportés

Tous

88.1 CopyDebugOutput

Syntaxe

```
CopyDebugOutput()
```

Description

Copie le contenu de la fenêtre de deboggage dans le presse-papier.

Arguments

Aucun.

Valeur de retour

Aucune.

Voir aussi

Included debugging tools , Debug , ShowDebugOutput() , ClearDebugOutput() , SaveDebugOutput()

OS Supportés

Tous

88.2 ShowDebugOutput

Syntaxe

```
ShowDebugOutput()
```

Description

Ouvre la fenêtre de déboguage ou la ramène au premier plan si elle était déjà ouverte.

Arguments

Aucun.

Valeur de retour

Aucune.

Voir aussi

Débogueur interne , Debug , ClearDebugOutput() , SaveDebugOutput() , CopyDebugOutput() , CloseDebugOutput()

OS Supportés

Tous

88.3 CloseDebugOutput

Syntaxe

```
CloseDebugOutput()
```

Description

Ferme la fenêtre de déboguage .

Arguments

Aucun.

Valeur de retour

Aucune.

Voir aussi

Débogueur interne , Debug , CopyDebugOutput() , ShowDebugOutput() , ClearDebugOutput() , SaveDebugOutput()

OS Supportés

Tous

88.4 ClearDebugOutput

Syntaxe

```
ClearDebugOutput()
```

Description

Efface le contenu de la fenêtre de déboguage.

Arguments

Aucun.

Valeur de retour

Aucune.

Exemple

```
1   ; Montre 10 valeurs dans le
      debugger tout les 500 ms
2   Repeat
3     ClearDebugOutput()
4     For i = 1 To 10
5       Debug x
6       x + 1
7     Next i
8
9     Delay(500)
10  ForEver
```

Voir aussi

Débogueur interne , Debug , ShowDebugOutput() , SaveDebugOutput() , CopyDebugOutput()

OS Supportés

Tous

88.5 DebuggerError

Syntaxe

```
DebuggerError(Message$)
```

Description

Génère une erreur du débogueur pendant l'exécution.
L'exécution du programme sera arrêté si le débogueur est activé.
Peut être utile lors de la création de modules réutilisables destinés à être partagés.

Arguments

Message$ Le message d'erreur à afficher.

Valeur de retour

Aucune.

Voir aussi

DebuggerWarning()

OS Supportés

Tous

88.6 DebuggerWarning

Syntaxe

```
DebuggerWarning(Message$)
```

Description

Génère un avertissement d'exécution du débogueur.
Peut être utile lors de la création de modules réutilisables destinés à être partagés.

Arguments

Message$ Le message d'avertissement à afficher.

Valeur de retour

Aucune.

Voir aussi

DebuggerError()

OS Supportés

Tous

88.7 SaveDebugOutput

Syntaxe

```
SaveDebugOutput(NomFichier$)
```

Description

Enregistre le contenu de la fenêtre de déboguage dans un fichier.

Arguments

NomFichier$ Le nom du fichier dans lequel sera enregistré le contenu de la fenêtre du débogueur.

Valeur de retour

Aucune.

Remarques

Une erreur est rapportée si le fichier ne peut pas être enregistré.

Exemple

```
1  For i = 1 To 100
2    Debug Random(i)
3  Next i
4  SaveDebugOutput("C:\log.txt")
```

Voir aussi

Débogueur interne , Debug , ShowDebugOutput() , ClearDebugOutput() , CopyDebugOutput()

OS Supportés

Tous

88.8 ShowProfiler

Syntaxe

```
ShowProfiler()
```

Description

Ouvre la fenêtre de profilage ou la ramène au premier plan si elle était déjà ouverte.

Arguments

Aucun.

Valeur de retour

Aucune.

Voir aussi

Débogueur interne , ResetProfiler() , StartProfiler() , StopProfiler()

OS Supportés

Tous

88.9 ResetProfiler

Syntaxe

```
ResetProfiler()
```

Description

Réinitialise le compteur pour le profileur.

Arguments

Aucun.

Valeur de retour

Aucune.

Voir aussi

Débogueur interne , ShowProfiler() , StartProfiler() , StopProfiler()

OS Supportés

Tous

88.10 StartProfiler

Syntaxe

```
StartProfiler()
```

Description

Démarre le décompte des lignes exécutées enregistré par le profileur.

Arguments

Aucun.

Valeur de retour

Aucune.

Voir aussi

Débogueur interne , ShowProfiler() , ResetProfiler() , StopProfiler()

OS Supportés

Tous

88.11 StopProfiler

Syntaxe

```
StopProfiler()
```

Description

Arrête le décompte des lignes exécutées enregistré par le profileur.

Arguments

Aucun.

Valeur de retour

Aucune.

Voir aussi

Débogueur interne , ShowProfiler() , ResetProfiler() , StartProfiler()

OS Supportés

Tous

88.12 ShowMemoryViewer

Syntaxe

```
ShowMemoryViewer([*AdresseMemoire,
    Longueur])
```

Description

Ouvre le visualisateur de mémoire ou le ramène au premier plan si il était déjà ouvert.

Arguments

***AdresseMemoire, Longueur (optionnel)** Si une zone mémoire (Buffer) représentée par *AdresseMemoire' de longueur 'Longueur' est spécifiée alors le contenu de cette zone mémoire sera affiché dans le visualisateur de mémoire.

Valeur de retour

Aucune.

Exemple

```
1  *Memoire =
     AllocateMemory(1000)
2  If *Memoire
3    RandomData(*Memoire,
     1000)       ; Remplir la
     mémoire avec des données
4
5    ShowMemoryViewer(*Memoire,
     1000) ; Ouvrir le
     visualisateur de mémoire
6    CallDebugger
                         ;
     Stoppe le programme qui ne
     se termine pas correctement
7  EndIf
```

Voir aussi

Débogueur interne

OS Supportés

Tous

88.13 ShowLibraryViewer

Syntaxe

```
ShowLibraryViewer([Bibliotheque$
    [, #Objet]])
```

Description

Ouvre le visualisateur de bibliothèques ou le ramène au premier plan si il était déjà ouvert.

Arguments

Bibliotheque$ (optionnel) Le visualisateur montre tous les objets de cette Bibliotheque$.

Sans cette option, le visualisateur n'affiche aucune bibliothèque en particulier.

#**Objet (optionnel)** Le visualisateur affichera seulement cette entrée de la 'Bibliotheque$'.
Sans cette option, le visualisateur n'affiche aucun objet en particulier.

Valeur de retour

Aucune.

Exemple

```
1   If CreateImage(0, 200, 200)
     And
    StartDrawing(ImageOutput(0))
2    DrawingMode(#PB_2DDrawing_Transparent)
3    Box(0, 0, 200, 200,
    RGB(255, 255, 255))
4    For i = 1 To 30
5      DrawText(Random(200),
    Random(200), "Hello le
    Monde !", RGB(Random(255),
    Random(255), Random(255)))
6    Next i
7    StopDrawing()
8
9    ShowLibraryViewer("Image",
    0)  ; Montre l'image
10   CallDebugger
                      ; Stoppe
    le programme qui ne se
    termine pas correctement
11  EndIf
```

Voir aussi

Débogueur interne

OS Supportés

Tous

88.14 ShowWatchlist

Syntaxe

```
ShowWatchlist()
```

Description

Ouvre la fenêtre de surveillance ou la ramène au premier plan si elle était déjà ouverte.

Arguments

Aucun.

Valeur de retour

Aucune.

Voir aussi

Débogueur interne

OS Supportés

Tous

88.15 ShowVariableViewer

Syntaxe

```
ShowVariableViewer()
```

Description

Ouvre la fenêtre des variables ou la ramène au premier plan si elle était déjà ouverte.

Arguments

Aucun.

Valeur de retour

Aucune.

Voir aussi

Débogueur interne

OS Supportés

Tous

88.16 ShowCallstack

Syntaxe

```
ShowCallstack()
```

Description

Ouvre la fenêtre d'historique des procédures ou la ramène au premier plan si elle était déjà ouverte.

Arguments

Aucun.

Valeur de retour

Aucune.

Voir aussi

Débogueur interne

OS Supportés

Tous

88.17 ShowAssemblyViewer

Syntaxe

```
ShowAssemblyViewer()
```

Description

Ouvre la fenêtre assembleur ou la ramène
au premier plan si elle était déjà ouverte.

Arguments

Aucun.

Valeur de retour

Aucune.

Voir aussi

Débogueur interne

OS Supportés

Tous

88.18 PurifierGranularity

Syntaxe

```
PurifierGranularity(GranularitéGlobale,
    GranularitéLocale,
    GranularitéChainesDeCaratere,
    GranularitéBlocsDynamique)
```

Description

Modifie l'intervalle de vérification
d'intégrité des données par le purificateur .

Arguments

GranularitéGlobale Le nombre de lignes de code source à exécuter entre les contrôles d'intégrité des données sur les variables globales.
#PB_Ignore gardera la valeur de l'intervalle existant.
Une valeur de 0 désactive la vérification.

GranularitéLocale Le nombre de lignes de code source à exécuter entre les contrôles d'intégrité des données sur les variables locales.
#PB_Ignore gardera la valeur de l'intervalle existant.
Une valeur de 0 désactive la vérification.

GranularitéChainesDeCaratere Le nombre de lignes de code source à exécuter entre les contrôles d'intégrité des données sur les chaînes de caractères en mémoire.
#PB_Ignore gardera la valeur de l'intervalle existant.
Une valeur de 0 désactive la vérification.

GranularitéBlocsDynamique Le nombre de lignes de code source à exécuter entre les contrôles d'intégrité des données sur les blocs de mémoire alloués dynamiquement.
#PB_Ignore gardera la valeur de l'intervalle existant.
Une valeur de 0 désactive la vérification.

Valeur de retour

Aucune.

Exemple

```
; Désactive la vérification des chaînes de caractères en mémoire ainsi que l'allocation mémoire toutes les 10 lignes
PurifierGranularity(#PB_Ignore, #PB_Ignore, 0, 10)
```

Voir aussi

Débogueur interne

OS Supportés

Tous

Chapitre 89

Desktop

Généralités

La bibliothèque desktop permet d'obtenir des informations sur l'environnement de travail de l'utilisateur : Le bureau.
Ces informations sont les dimensions du bureau, sa résolution, la position du curseur de la souris, etc.
Il est intéressant de signaler que vous avez la possibilité d'utiliser plusieurs écrans (moniteurs) simultanément.

OS Supportés

Tous

89.1 ExamineDesktops

Syntaxe

```
Resultat = ExamineDesktops()
```

Description

Examine les caractéristiques des bureaux que l'utilisateur a à sa disposition.

Arguments

Aucun.

Valeur de retour

Renvoie le nombre de bureaux détectés, zéro sinon.

Remarques

Cette commande doit être appelée avant d'utiliser les fonctions suivantes : DesktopDepth() , DesktopFrequency() , DesktopHeight() , DesktopName() et DesktopWidth() .

En règle générale, un utilisateur n'en utilise qu'un seul, mais il est possible d'en avoir plusieurs en cas de double-écrans par exemple.

Exemple

```
1  MessageRequester("Information
     bureau", "Vous avez
     "+Str(ExamineDesktops())+"
     bureau(x)")
```

Voir aussi

DesktopDepth() , DesktopFrequency() , DesktopHeight() , DesktopName() , DesktopWidth()

OS Supportés

Tous

89.2 DesktopDepth

Syntaxe

```
Resultat =
    DesktopDepth(#Bureau)
```

Description

Renvoie la profondeur de couleur d'un bureau.

Arguments

#**Bureau** Le numéro du bureau à tester. Le bureau principal a toujours la valeur 0.

Valeur de retour

Renvoie l'une des valeurs suivantes, en bits par pixels : 1, 2, 4, 8, 15, 16, 24 ou 32.

Remarques

ExamineDesktops() doit être appelée avant cette commande pour mettre à jour les informations concernant les bureaux.

Exemple

```
1  ExamineDesktops()
2  MessageRequester("Information
     d'affichage", "Résolution
     =
     "+Str(DesktopWidth(0))+"x"+Str(DesktopHeight(0))+"x"+Str(Des
```

Voir aussi

ExamineDesktops() , DesktopFrequency() , DesktopHeight() , DesktopName() , DesktopWidth()

OS Supportés

Tous

89.3 DesktopResolutionX

Syntaxe

```
Resultat.d =
    DesktopResolutionX()
```

Description

Renvoie le facteur de résolution DPI horizontal du bureau.

Arguments

Aucun.

Valeur de retour

Renvoie le facteur de résolution DPI du bureau sur l'axe 'x'.
Si la valeur est '1', aucun facteur DPI n'a été appliqué sur l'axe 'x'.
Si la valeur est '1.25', un facteur de 125% a été appliqué sur l'axe 'x'.

Remarques

L'application doit être compilée avec le commutateur 'DPI' pour que cette commande renvoie le facteur de résolution DPI réel. Sinon, le résultat sera toujours '1'.

Exemple

```
1   Debug "DPI du bureau:
      Facteur d'échelle
      d'affichage horizontal : "
      + DesktopResolutionX()
```

Voir aussi

DesktopResolutionY() , DesktopScaledX() , DesktopScaledY() , DesktopUnscaledX() , DesktopUnscaledY()

OS Supportés

Tous

89.4 DesktopResolutionY

Syntaxe

```
Resultat.d =
    DesktopResolutionY()
```

Description

Renvoie le facteur de résolution DPI vertical du bureau.

Arguments

Aucun.

Valeur de retour

Renvoie le facteur de résolution DPI du bureau sur l'axe 'y'.
Si la valeur est '1', aucun facteur DPI n'a été appliqué sur l'axe 'y'.
Si la valeur est '1.25', un facteur de 125% a été appliqué sur l'axe 'y'.

Remarques

L'application doit être compilée avec le commutateur 'DPI' pour que cette commande renvoie le facteur de résolution DPI réel. Sinon, le résultat sera toujours '1'.

Exemple

```
1  Debug "DPI du bureau:
     Facteur d'échelle
     d'affichage vertical : " +
     DesktopResolutionY()
```

Voir aussi

DesktopResolutionX() , DesktopScaledX() , DesktopScaledY() , DesktopUnscaledX() , DesktopUnscaledY()

OS Supportés

Tous

89.5 DesktopScaledX

Syntaxe

```
Resultat =
    DesktopScaledX(Valeur)
```

Description

Renvoie une valeur après mise à l'échelle, en fonction du DPI horizontal.

Arguments

Valeur La valeur avec un DPI de 1.

Valeur de retour

Renvoie la valeur mise à l'échelle en fonction de l'affichage actuel du DPI sur l'axe 'x'. Par exemple, sur un affichage avec une résolution de 125%, une valeur de 100 correspond à 125.

Remarques

Ceci est surtout utile pour calculer la position réelle du pixel indépendamment du DPI d'affichage.
L'application doit être compilée avec le commutateur 'DPI' pour que cette commande renvoie le facteur de résolution DPI réel. Sinon, le résultat sera toujours le même que celui du paramètre 'Valeur'.

Exemple

```
1   Debug "DPI horizontal du
      bureau, valeur mise à
      l'échelle de 100: " +
      DesktopScaledX(100)
```

Voir aussi

DesktopResolutionX() , DesktopResolutionY() , DesktopScaledY() , DesktopUnscaledX() , DesktopUnscaledY()

OS Supportés

Tous

89.6 DesktopScaledY

Syntaxe

```
Resultat =
    DesktopScaledY(Valeur)
```

Description

Renvoie une valeur après mise à l'échelle, en fonction du DPI vertical.

Arguments

Valeur La valeur avec un DPI de 1.

Valeur de retour

Renvoie la valeur mise à l'échelle en fonction de l'affichage actuel du DPI sur l'axe 'y'. Par exemple, sur un affichage avec une résolution de 125%, une valeur de 100 correspond à 125.

Remarques

Ceci est surtout utile pour calculer la position réelle du pixel indépendamment du DPI d'affichage.
L'application doit être compilée avec le commutateur 'DPI' pour que cette commande renvoie le facteur de résolution DPI réel. Sinon, le résultat sera toujours le même que celui du paramètre 'Valeur'.

Exemple

```
1   Debug "DPI vertical du
      bureau, valeur mise à
      l'échelle de 100: " +
      DesktopScaledY(100)
```

Voir aussi

DesktopResolutionX() , DesktopResolutionY() , DesktopScaledX() , DesktopUnscaledX() , DesktopUnscaledY()

OS Supportés

Tous

89.7 DesktopUnscaledX

Syntaxe

```
Resultat =
    DesktopUnscaledX(Valeur)
```

Description

Renvoie une valeur non mise à l'échelle en fonction du DPI horizontal en cours.

Arguments

Valeur La valeur avec le DPI en cours.

Valeur de retour

Renvoie la valeur non mise à l'échelle en fonction du DPI actuel sur l'axe 'x'.
Par exemple, sur un affichage avec une résolution de 125%, une valeur de 125 correspond à 100.

Remarques

Ceci est surtout utile pour calculer la position réelle du pixel indépendamment du DPI d'affichage.
L'application doit être compilée avec le commutateur 'DPI' pour que cette commande renvoie la valeur redimensionnée. Sinon, le résultat sera toujours le même que celui du paramètre "Valeur".

Exemple

```
1   Debug "DPI horizontal du
      bureau, valeur non mise à
      l'échelle de 125: " +
      DesktopUnscaledX(125)
```

Voir aussi

DesktopResolutionX() , DesktopResolutionY() , DesktopScaledX() , DesktopScaledY() , DesktopUnscaledY()

OS Supportés

Tous

89.8 DesktopUnscaledY

Syntaxe

```
Resultat =
    DesktopUnscaledY(Valeur)
```

Description

Renvoie une valeur non mise à l'échelle en fonction du DPI vertical en cours.

Arguments

Valeur La valeur avec le DPI en cours.

Valeur de retour

Renvoie la valeur non mise à l'échelle en fonction du DPI actuel sur l'axe 'y'.
Par exemple, sur un affichage avec une résolution de 125%, une valeur de 125 correspond à 100.

Remarques

Ceci est surtout utile pour calculer la position réelle du pixel indépendamment du DPI d'affichage.
L'application doit être compilée avec le commutateur 'DPI' pour que cette commande renvoie la valeur redimensionnée. Sinon, le résultat sera toujours le même que celui du paramètre "Valeur".

Exemple

```
Debug "DPI vertical du bureau, valeur non mise à l'échelle de 125: " + DesktopUnscaledY(125)
```

Voir aussi

DesktopResolutionX() , DesktopResolutionY() , DesktopScaledX() , DesktopScaledY() , DesktopUnscaledX()

OS Supportés

Tous

89.9 DesktopFrequency

Syntaxe

```
Resultat = DesktopFrequency(#Bureau)
```

Description

Renvoie le taux de rafraîchissement d'un bureau.

Arguments

#Bureau Le numéro du bureau à tester. Le bureau principal a toujours la valeur 0.

Valeur de retour

Renvoie le taux de rafraîchissement en Hertz du bureau.
Si Taux vaut 0 alors la fréquence matérielle par défaut est utilisée, ou la fréquence réelle n'a pu être déterminée.

Remarques

ExamineDesktops() doit être appelée avant cette commande pour mettre à jour les informations concernant les bureaux.
Note : sous Linux, cette fonction renvoie toujours 0.

Exemple

```
1  ExamineDesktops()
2  Frequency =
    DesktopFrequency(0)
3  If Frequency = 0
4    MessageRequester("Information
    d'affichage", "Il n'y a
    aucun réglage utilisateur,
    le taux de
    rafraîchissement par
    défaut du matériel est
    utilisé.")
5  Else
6    MessageRequester("Information
    d'affichage", "taux de
    rafraîchissement du
    bureau: "+Str(Frequency)+"
    Hz.")
7  EndIf
```

Voir aussi

ExamineDesktops() , DesktopDepth() , DesktopHeight() , DesktopName() , DesktopWidth()

OS Supportés

Windows, MacOS X

89.10 DesktopHeight

Syntaxe

```
Resultat =
    DesktopHeight(#Bureau)
```

Description

Renvoie la hauteur d'un bureau.

Arguments

#**Bureau** Le numéro du bureau à tester.
Le bureau principal a toujours la valeur 0.

Valeur de retour

Renvoie la hauteur en pixels du bureau.

Remarques

ExamineDesktops() doit être appelée avant cette commande pour mettre à jour les informations concernant les bureaux.

Exemple

```
1  ExamineDesktops()
2  MessageRequester("Information d'affichage", "Résolution =
    "+Str(DesktopWidth(0))+"x"+Str(DesktopHeight(0))+"x"+Str(Des
```

Voir aussi

ExamineDesktops() , DesktopDepth() , DesktopX() , DesktopY() , DesktopWidth()

OS Supportés

Tous

89.11 DesktopX

Syntaxe

```
Resultat = DesktopX(#Bureau)
```

Description

Renvoie la position 'X' d'un bureau.

Arguments

#Bureau Le numéro du bureau à tester.
Le bureau principal a toujours la valeur 0.

Valeur de retour

Renvoie la position 'X' en pixels, à partir du coin supérieur gauche du bureau.
La coordonnée est relative au coin supérieur gauche de l'écran principal.
Elle est négative si le bureau est indiqué à gauche de l'écran principal.

Remarques

ExamineDesktops() doit être appelée avant cette commande pour mettre à jour les informations concernant les bureaux.

Exemple

```
ExamineDesktops()
MessageRequester("Position
 du bureau principal",
 "Position en X =
 "+Str(DesktopX(0)) + #LF$
 + "Position en Y =
 "+Str(DesktopY(0)))
```

Voir aussi

ExamineDesktops() , DesktopDepth() , DesktopY() , DesktopHeight() , DesktopWidth()

OS Supportés

Tous

89.12 DesktopY

Syntaxe

```
Resultat = DesktopY(#Bureau)
```

Description

Renvoie la position 'Y' d'un bureau.

Arguments

#Bureau Le numéro du bureau à tester.
Le bureau principal a toujours la valeur 0.

Valeur de retour

Renvoie la position 'Y' en pixels, à partir du coin supérieur gauche du bureau.
La coordonnée est relative au coin supérieur gauche de l'écran principal.
Elle est négative si le bureau est indiqué au dessus de l'écran principal.

Remarques

ExamineDesktops() doit être appelée avant cette commande pour mettre à jour les informations concernant les bureaux.

Exemple

```
ExamineDesktops()
```

```
2   MessageRequester("Position
     du bureau principal",
    "Position en X =
    "+Str(DesktopX(0)) + #LF$
    + "Position en Y =
    "+Str(DesktopY(0)))
```

Voir aussi

ExamineDesktops() , DesktopDepth() , DesktopX() , DesktopHeight() , DesktopWidth()

OS Supportés

Tous

89.13 DesktopMouseX

Syntaxe

```
Resultat = DesktopMouseX()
```

Description

Renvoie la position absolue en X de la souris sur le bureau.

Arguments

Aucun.

Valeur de retour

Renvoie la position absolue en X en pixels de la souris à partir du coin supérieur gauche du moniteur principal.
La coordonnée est négative si la souris est sur un moniteur à gauche de l'écran principal.

Exemple

```
1   If OpenWindow(0, 0, 0, 300,
     30, "Position de la souris
     sur le bureau",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2     TextGadget(0, 10, 6, 200,
    20, "")
3
4     Repeat
5       Event = WindowEvent()
6
```

```
7       If Event = 0 ; Il n'y a plus d'événement dans la file d'attente, libère le processeur quelques millisecondes pour le multi-tâches
8         SetGadgetText(0, "Coordonnées : "+Str(DesktopMouseX())+","+Str(DesktopMouseY()))
9         Delay(20)
10      EndIf
11
12     Until Event = #PB_Event_CloseWindow
13   EndIf
```

Voir aussi

DesktopMouseY() , DesktopX() , DesktopWidth() , WindowMouseX()

OS Supportés

Tous

89.14 DesktopMouseY

Syntaxe

```
Resultat = DesktopMouseY()
```

Description

Renvoie la position absolue en Y de la souris sur le bureau.

Arguments

Aucun.

Valeur de retour

Renvoie la position absolue en Y en pixels de la souris à partir du coin supérieur gauche du moniteur principal.
La coordonnée est négative si la souris est sur un moniteur au dessus de l'écran principal.

Exemple

```
1   If OpenWindow(0, 0, 0, 300, 30, "Position de la souris sur le bureau", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
```

```
2       TextGadget(0, 10, 6, 200,
        20, "")
3
4       Repeat
5         Event = WindowEvent()
6
7         If Event = 0 ; Il n'y a
        plus d'événement dans la
        file d'attente, libère le
        processeur quelques
        millisecondes pour le
        multi-tâches
8           SetGadgetText(0,
        "Coordonnées :
        "+Str(DesktopMouseX())+","+Str(DesktopMouseY()))
9           Delay(20)
10        EndIf
11
12      Until Event =
        #PB_Event_CloseWindow
13    EndIf
```

Voir aussi

DesktopMouseX() , DesktopY() , DesktopHeight() , WindowMouseY()

OS Supportés

Tous

89.15 DesktopName

Syntaxe

```
Resultat\$ =
    DesktopName(#Bureau)
```

Description

Renvoie le nom d'un bureau.

Arguments

#**Bureau** Le numéro du bureau à tester.
Le bureau principal a toujours la valeur 0.

Valeur de retour

Renvoie le nom du bureau , s'il existe, une chaîne vide sinon.

Remarques

ExamineDesktops() doit être appelée avant cette commande pour mettre à jour les informations concernant les bureaux.

Exemple

```
1  ExamineDesktops()
2  MessageRequester("Information
    bureau", "Nom du bureau
    principal =
    "+DesktopName(0))
```

Voir aussi

ExamineDesktops() , DesktopDepth() , DesktopFrequency() , DesktopHeight() , DesktopWidth()

OS Supportés

Tous

89.16 DesktopWidth

Syntaxe

```
Resultat =
    DesktopWidth(#Bureau)
```

Description

Renvoie la largeur d'un bureau.

Arguments

#**Bureau** Le numéro du bureau à tester.
Le bureau principal a toujours la valeur 0.

Valeur de retour

Renvoie la largeur en pixel du bureau.

Remarques

ExamineDesktops() doit être appelée avant cette commande pour mettre à jour les informations concernant les bureaux.

Exemple

```
1  ExamineDesktops()
2  MessageRequester("Information
    d'affichage", "Résolution
    =
    "+Str(DesktopWidth(0))+"x"+Str(DesktopHeight(0))+"x"+Str(Des
```

Voir aussi

ExamineDesktops() , DesktopDepth() , DesktopX() , DesktopY() , DesktopHeight()

OS Supportés

Tous

Chapitre 90

Dialog

Généralités

La bibliothèque 'dialog' permet de créer facilement une interface utilisateur complexe (GUI) basée sur du code XML. C'est une autre façon de créer des fenêtres, des boîtes de dialogue, etc. Elle dispose de l'agencement et de la réorganisation automatique des gadgets (layout), ce qui est très utile lors de la création d'interface qui doit fonctionner sur différents systèmes d'exploitation ou qui utilise différentes tailles de police simultanément.
Le code XML peut provenir d'un fichier ou bien être créé à la volée, en mémoire, en utilisant la bibliothèque XML .

OS Supportés

Tous

90.1 CreateDialog

Syntaxe

```
Resultat =
    CreateDialog(#Dialog)
```

Description

Créer une nouvelle interface utilisateur non initialisée.
Pour l'afficher, utiliser OpenXMLDialog() .

Arguments

#Dialog Le numéro d'identification de la nouvelle interface utilisateur.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Valeur de retour

Renvoie une valeur non nulle si l'interface utilisateur a été créée avec succès ou zéro sinon.
Si #PB_Any a été utilisé comme paramètre '#Dialog' alors le nombre auto-généré est renvoyé en cas de succès.

Voir aussi

OpenXMLDialog() , FreeDialog()

OS Supportés

Tous

90.2 DialogError

Syntaxe

```
Resultat\$ =
    DialogError(#Dialog)
```

Description

Renvoie le dernier message d'erreur (en anglais) après l'échec de la fonction OpenXMLDialog() .

Arguments

#Dialog L'interface utilisateur à utiliser.

Valeur de retour

Renvoie le message d'erreur.
Si aucune information supplémentaire n'est disponible alors la fonction peut renvoyer une chaîne de caractères vide.

Voir aussi

CreateDialog() , OpenXMLDialog()

OS Supportés

Tous

90.3 DialogGadget

Syntaxe

```
Resultat =
    DialogGadget(#Dialog, Nom$)
```

Description

Renvoie le numéro du gadget associé à son nom.

Arguments

#**Dialog** L'interface utilisateur à utiliser.

Nom$ Le nom du gadget spécifié dans le fichier XML (après l'attribut 'name').

Valeur de retour

Renvoie le numéro du gadget à partir du nom de gadget spécifié, ou -1 si le gadget n'est pas trouvé.

Voir aussi

CreateDialog() , OpenXMLDialog()

OS Supportés

Tous

90.4 DialogWindow

Syntaxe

```
Resultat =
    DialogWindow(#Dialog)
```

Description

Renvoie le numéro de la fenêtre de l'interface utilisateur.

Arguments

#**Dialog** L'interface utilisateur à utiliser.

Valeur de retour

Renvoie le numéro de la fenêtre de l'interface utilisateur spécifiée.

Remarques

L'interface utilisateur doit être initialisée avec OpenXMLDialog() avant d'utiliser cette commande.
Ce numéro peut être utilisé avec n'importe quelle fonction de la bibliothèque Window .

Voir aussi

CreateDialog() , OpenXMLDialog()

OS Supportés

Tous

90.5 DialogID

Syntaxe

```
Resultat = DialogID(#Dialog)
```

Description

Renvoie l'identifiant unique (ID) de l'interface utilisateur dans le système d'exploitation.

Arguments

#**Dialog** L'interface utilisateur à utiliser.

Valeur de retour

Renvoie l'ID de l'interface utilisateur dans le système d'exploitation.

Voir aussi

CreateDialog() , OpenXMLDialog()

OS Supportés

Tous

90.6 FreeDialog

Syntaxe

```
FreeDialog(#Dialog)
```

Description

Libère l'interface utilisateur ainsi que sa mémoire associée.
Si l'interface utilisateur est toujours ouverte, elle sera automatiquement fermée.

Arguments

#**Dialog** L'interface utilisateur à libérer. Si #PB_All est spécifié, toutes les interfaces utilisateur restantes sont libérées.

Valeur de retour

Aucune.

Remarques

Toutes les interfaces utilisateur restantes sont automatiquement libérées lorsque le programme est fermé.

Voir aussi

CreateDialog()

OS Supportés

Tous

90.7 IsDialog

Syntaxe

```
Resultat = IsDialog(#Dialog)
```

Description

Teste si l'interface utilisateur donnée est une interface utilisateur valide.

Arguments

#Dialog L'interface utilisateur à tester.

Valeur de retour

Renvoie une valeur non nulle si #Dialog est une interface utilisateur valide ou zéro sinon.

Remarques

Cette fonction peut être utilisée avec n'importe quelle valeur sans crainte de crash.
C'est une bonne façon de s'assurer qu'une interface utilisateur est prête à être utilisée.

Voir aussi

CreateDialog()

OS Supportés

Tous

90.8 OpenXMLDialog

Syntaxe

```
Resultat =
    OpenXMLDialog(#Dialog,
    #XML, Nom$ [, X, Y [,
    Largeur, Hauteur [,
    ParentID]]]))
```

Description

Ouvre l'interface utilisateur spécifiée (fenêtre, boîte de dialogue, etc) et l'affiche à l'écran.
Pour accéder aux gadgets, voir DialogGadget() .
Pour obtenir le numéro de la fenêtre, voir DialogWindow() .

Arguments

#**Dialog** L'interface utilisateur à utiliser. Elle doit être créée au préalable avec CreateDialog() .

#**XML** Le code xml à utiliser.
Il doit être créé au préalable avec LoadXML() , CreateXML() , CatchXML() ou ParseXML() .
Cela signifie qu'il est possible de créer des interfaces utilisateur à la volée avec CreateXML() , CatchXML() ou ParseXML() .
Voir les remarques ci-dessous pour prendre connaissance des attributs XML supportés.
Lorsque le script XML est compris dans le code, il est peut être plus aisé d'utiliser la quote (guillemet simple) pour les attributs XML (c'est une syntaxe XML parfaitement admise).

Nom$ Le nom de l'interface utilisateur à ouvrir.
En effet, un fichier XML peut contenir le code de plusieurs interfaces utilisateur.

X, Y (optionnel) Les coordonnées x, y (en pixels) de l'interface utilisateur.

Largeur, Hauteur (optionnel) La taille (en pixels) de l'interface utilisateur.
Si la taille est plus petite que la taille calculée d'après le code du fichier XML et d'après l'agencement automatique des gadgets ou "layout", alors ces paramètres optionnels ne seront pas pris en compte et la taille calculée sera bien utilisée.
Sans ces paramètres optionnels, la taille de la boîte de dialogue aura la plus petite taille possible.

ParentID (optionnel) L'identifiant de la fenêtre parent.
Un identifiant de fenêtre valide peut être récupéré avec WindowID() .

Valeur de retour

Renvoie une valeur non nulle si l'interface utilisateur a été ouverte avec succès ou zéro sinon.

Pour obtenir plus d'informations sur l'erreur qui a eu lieu, voir DialogError() .

Remarques

```
------------------------------------------------------------------
   .: Format XML d'une
   interface utilisateur :.
------------------------------------------------------------------

  I. Attributs communs
  --------------------

  width      - Largeur
   minimale. Valeur positive
   ou nulle (par défaut="0")
  height     - Hauteur
   minimale. Valeur positive
   ou nulle (par défaut="0")

  id         - Numéro
   d'identification d'un
   gadget ou d'une fenêtre
   (par défaut c'est
   #PB_Any). Il peut être une
   constante runtime
  name       - Une chaîne de
   caractères qui identifie
   l'objet (principalement
   pour DialogGadget()
, insensible à la casse) (par
   défaut="")
  text       - Texte associé à
   l'objet (par défaut="")

  flags      - Options du
   gadget/fenêtre, comme
   "#PB_Window_Borderless |
   #PB_Window_ScreenCentered"
    (par défaut="")

  min         - Valeur minimum
  max         - Valeur maximum
  value       - Valeur courante

  invisible - Egal à "yes",
   l'objet créé est invisible
   (par défaut="no")
  disabled  - Egal à "yes",
   l'objet créé est désactivé
   (seulement pour les
   gadgets) (par défaut="no")

  colspan    - Seulement à
   l'intérieur de l'attribut
   <gridbox>, permet à un
   élément de couvrir
   plusieurs lignes / colonnes
  rowspan      (par défaut="1")
```

```
Remarque : Tous ces
 attributs sont facultatifs.


II. L'élément racine
--------------------

<window>  : Une fenêtre
 unique
</window>

ou

<dialogs> : Plusieurs
 fenêtres dans un même
 fichier XML
  <window
 name="PremiereFenetre">
  </window>
  <window
 name="SecondeFenetre">
  </window>
  ...
</dialogs>


III. L'élément fenêtre
----------------------

<window>
</window>

Clés reconnues dans le code
 XML:
..............................

Tous les attributs communs
 ainsi que les clés
 suivantes:

minwidth  = 'auto' ou une
 valeur numérique
maxwidth  = 'auto' ou une
 valeur numérique
minheight = 'auto' ou une
 valeur numérique
maxheight = 'auto' ou une
 valeur numérique

Permet de définir les
 tailles minimum et maximum
 d'une fenêtre.
Si la valeur est 'auto'
 alors la taille est
 calculée en fonction de la
 taille
(calculée ou prédéfinie)
 des éléments contenus dans
```

la fenêtre.

- Crée la fenêtre
- Peut avoir tous les attributs communs.
- Est un conteneur à un seul élément.
- Si plus d'un élément <window> est présent, l'attribut 'name' est utilisé pour les identifier.
- Tous les éléments d'interface graphique ne peuvent être placés qu'ici.

IV. Les élements Boîtes, agencement et réorganisation automatique des gadgets (Layout)

****
hbox et vbox
****

Boîtes qui disposent les éléments horizontalement ou verticalement.
Peut contenir n'importe quel nombre d'éléments.

Clés reconnues dans le code XML:
.....

Tous les attributs communs ainsi que les clés suivantes:

spacing = Espace à ajouter entre les éléments (par défaut=5)

expand = yes
- Les éléments deviennent plus grands pour remplir tout l'espace (par défaut)

no
- Les éléments ne s'étendent pas pour remplir tout l'espace

equal
- Force les éléments à avoir une taille égale

item:<numéro>
- Elargit un seul élément,

```
si l'espace est disponible

 align     =
top/center/bottom      -
Ne s'applique que lorsque
l'attribut expand="no"
pour les vbox et top est
l'attribut par défaut
          =
left/center/right      -
Ne s'applique que lorsque
l'attribut expand="no"
pour les hbox et left est
l'attribut par défaut
                                       -
Un assemblage de vbox et
de hbox permet un
alignement complexe
top/left/center/bottom/right

************************************************************
gridbox
************************************************************

 Aligne les éléments dans
un tableau.
 Peut contenir n'importe
quel nombre d'éléments.

 Clés reconnues dans le
code XML:
 ..............................

 Tous les attributs
communs ainsi que les clés
suivantes:

 columns    = Nombre de
colonnes (par défaut = 2)

 colspacing = Espace entre
les colonnes / lignes (par
défaut = 5)
 rowspacing

 colexpand = yes
- Les éléments deviennent
plus grands pour remplir
tout l'espace (par défaut)
 rowexpand   no
- Les éléments ne
s'étendent pas pour
remplir tout l'espace
             equal
- Force les éléments à
avoir une taille égale
             item:<numéro>
- Elargit un seul élément,
si l'espace est disponible
```

```
              Pour
 colexpand, par défaut=yes
              Pour
 rowexpand, par défaut=no

  Tous les éléments d'un
 gridbox peuvent utiliser
 les clés suivantes:
  colspan = nombre de
 colonnes à fusionner (par
 défaut = 1)
  rowspan = nombre de
 lignes à fusionner

**************************************************************
multibox
**************************************************************

  Une boîte avec de
 multiples éléments dans la
 même position.
  Permet d'utiliser
 plusieurs conteneurs et de
 n'en montrer qu'un seul à
 la fois.
  Peut contenir n'importe
 quel nombre d'éléments.

  Clés reconnues dans le
 code XML:
  ..............................

  Tous les attributs
 communs.

**************************************************************
singlebox
**************************************************************

  Une boîte avec un seul
 élément.
  Utilisée uniquement pour
 ajouter une marge
 supplémentaire ou
  des propriétés
 d'alignement
 supplémentaires à un
 élément.

   Clés reconnues dans le
 code XML:
  ..............................

   Tous les attributs
 communs ainsi que les clés
 suivantes:

  margin = Marge autour du
```

```
 contenu (par défaut = 10)
          Peut être un
 nombre unique (= pour
 toutes les marges), ou une
 combinaison de
          top:<num>,left:<num>,right:<num>,bottom:<num>,vert
          Exemple:
 "vertical:5,left:10,right:0"

  expand = yes        - Les
 éléments deviennent plus
 grands pour remplir tout
 l'espace (par défaut)
          no          - Les
 éléments ne s'étendent pas
 pour remplir tout l'espace
          vertical    -
 Elargissement
 verticalement seulement
          horizontal -
 Elargissement
 horizontalement seulement

  expandwidth  = Taille
 maximale d'élargissement
 des éléments. Si la taille
 demandée est supérieure à
  expandheight   ce
 paramètre alors la taille
 de la requête est utilisée
 (le contenu ne peut pas
 devenir plus petit)
                par
 défaut=0

  align  = C'est une
 combinaison de top, left,
 bottom, right et center.
 (n'est efficace que
 lorsque expand <> yes)
          Exemple:
 "top,center" ou "top,left"
 (par défaut)



V. Les éléments Gadget
----------------------

Clés reconnues dans le code
  XML:
..............................

Tous les attributs XML
  communs sont pris en
  charge.

De plus, pour lier une
  procédure d'événement
```

```
 directement dans le
 fichier XML,
les attributs suivants sont
 disponibles pour les
 gadgets:

  onevent             =
 EventProcedure() - Liaison
 d'événements génériques,
 pour tous les types
 d'événements.
  onchange            =
 EventProcedure() - Liaison
 de type
 #PB_EventType_Change
           (uniquement pour
 les gadgets qui supportent
 ce type d'événement).
  onfocus             =
 EventProcedure() - Liaison
 de type
 #PB_EventType_Focus
            (uniquement
 pour les gadgets qui
 supportent ce type
 d'événement).
  onlostfocus         =
 EventProcedure() - Liaison
 de type
 #PB_EventType_LostFocus
        (uniquement pour
 les gadgets qui supportent
 ce type d'événement).
  ondragstart         =
 EventProcedure() - Liaison
 de type
 #PB_EventType_DragStart
        (uniquement pour
 les gadgets qui supportent
 ce type d'événement).
  onrightclick        =
 EventProcedure() - Liaison
 de type
 #PB_EventType_RightClick
       (uniquement pour les
 gadgets qui supportent ce
 type d'événement).
  onleftclick         =
 EventProcedure() - Liaison
 de type
 #PB_EventType_LeftClick
        (uniquement pour
 les gadgets qui supportent
 ce type d'événement).
  onrightdoubleclick =
 EventProcedure() - Liaison
 de type
 #PB_EventType_RightDoubleClick
 (uniquement pour les
```

```
  gadgets qui supportent ce
  type d'événement).
   onleftdoubleclick  =
  EventProcedure() - Liaison
  de type
  #PB_EventType_LeftDoubleClick
   (uniquement pour les
  gadgets qui supportent ce
  type d'événement).

 'EventProcedure()' doit
  être déclarée comme
  'Runtime' dans le code
  principal, et doit
  respecter le format
 de la procedure BindEvent()
  de la bibliothèque
  PureBasic 'Window'.
 En fait c'est
  BindGadgetEvent() qui est
  appelée ;)

 Les gadgets supportés sont :
 <button>
 <buttonimage>
 <calendar>
 <canvas>
 <checkbox>
 <combobox>
 <container>    - Conteneur
  à un seul élément
 <date>
 <editor>
 <explorercombo>
 <explorerlist>
 <explorertree>
 <frame>        - Conteneur
  à un seul élément (Avec
  bordures standards
  seulement)
 <hyperlink>
 <ipaddress>
 <image>
 <listicon>
 <listview>
 <option group> - Utilise le
  même numéro de 'group'
  pour créer des
  OptionGadget()
liés entre eux.
 <panel>        - Peut
  contenir <tab> seulement
 <progressbar min max valeur>
 <scrollarea
  scrolling="="vertical,
  horizontal ou les deux
  (par défaut)"
  innerheight="valeur ou
  auto (par défaut)"
```

```
 innerwidth="valeur ou auto
 (par défaut)" pas>
                - Conteneur
 à un seul élément
<scrollbar min max page
 value>
                - page =
 Longueur de la page
<spin min max value>
<splitter firstmin="valeur
 ou auto" secondmin>
                - Doit
 contenir 2 éléments,
 (c'est un container a 2
 "éléments-enfants"),
                  la taille
 minimum est déterminée par
 les gadgets contenus par
 ces 2 éléments.
                  Si 'auto'
 est spécifié, la valeur
 min sera la taille
 minimale de l'enfant.
<string>
<text>
<trackbar min max valeur>
<tree>
<web>
<scintilla>    - Le
 callback reste vide

Éléments liés à Gadget:
<tab>          - Onglet,
 conteneur à un seul
 élément (l'attribut 'text'
 est supporté).

Element Spécial :
<empty>        - Un élément
 vide, utile quand il est
 nécessaire d'avoir un
 espace entre l'élément,
                  afin de
 les aligner aux bordures
 par exemple.
```

Exemple : Simple fenêtre redimensionnable

```
1  #Dialog = 0
2  #Xml = 0
3
4  XML$ = "<window
    id='#PB_Any' name='test'
    text='test'
    minwidth='auto'
    minheight='auto'
    flags='#PB_Window_ScreenCentered
    | #PB_Window_SystemMenu |
```

```
    #PB_Window_SizeGadget'>" +
5         "  <panel>" +
6         "    <tab
    text='Premier Onglet'>" +
7         "      <vbox
    expand='item:2'>" +
8         "        <hbox>" +
9         "          <button
    text='Bouton 1'/>" +
10        "          <checkbox
    text='Case à cocher 1'/>" +
11        "          <button
    text='Bouton 2'/>" +
12        "        </hbox>" +
13        "        <editor
    text='Contenu...'
    height='150'/>" +
14        "      </vbox>" +
15        "    </tab>" +
16        "    <tab
    text='Second Onglet'>" +
17        "    </tab>" +
18        "  </panel>" +
19        "</window>"
20
21    If ParseXML(#Xml, XML$)
    And XMLStatus(#Xml) =
    #PB_XML_Success
22
23    If CreateDialog(#Dialog)
    And OpenXMLDialog(#Dialog,
    #Xml, "test")
24
25      Repeat
26        Event =
    WaitWindowEvent()
27      Until Event =
    #PB_Event_CloseWindow
28
29    Else
30      Debug "Erreur de la
    bibliothèque -Dialog- : "
    + DialogError(#Dialog)
31    EndIf
32  Else
33    Debug "Erreur XML : " +
    XMLError(#Xml) + " (Ligne:
    " + XMLErrorLine(#Xml) +
    ")"
34  EndIf
```

Exemple : Boîte 'Multibox'

```
1   #Dialog = 0
2   #Xml = 0
3
4   Runtime Enumeration Gadget
5     #ListView
```

```
    #GeneralContainer
    #EditorContainer
    #BackupContainer
  EndEnumeration

  Procedure ShowPanels()

    HideGadget(#GeneralContainer, #True)
    HideGadget(#EditorContainer, #True)
    HideGadget(#BackupContainer, #True)

    Select GetGadgetState(#ListView)
      Case 0
        HideGadget(#GeneralContainer, #False)

      Case 1
        HideGadget(#EditorContainer, #False)

      Case 2
        HideGadget(#BackupContainer, #False)
    EndSelect
  EndProcedure

  Runtime Procedure OnListViewEvent()
    ShowPanels()
  EndProcedure

  XML$ = "<window
   id='#PB_Any' name='test'
   text='Préférences'
   minwidth='auto'
   minheight='auto'
   flags='#PB_Window_ScreenCentered
   | #PB_Window_SystemMenu |
   #PB_Window_SizeGadget'>" +
         "  <hbox
   expand='item:2'>" +
         "    <listview
   id='#ListView' width='100'
   onEvent='OnListViewEvent()'/>"
   +
         "    <multibox>" +
         "" +
         "      <container
   id='#GeneralContainer'
   invisible='yes'>" +
         "        <frame
   text='Général'>" +
         "          <vbox
   expand='no'>" +
```

```
        "
    <checkbox text='Activer la
    LED rouge'/>" +
        "
    <checkbox text='Activer la
    LED verte'/>" +
        "          </vbox>" +
        "        </frame>" +
        "      </container>"
    +
        "" +
        "      <container
    id='#EditorContainer'
    invisible='yes'>" +
        "        <frame
    text='Editeur'>" +
        "          <vbox
    expand='no'>" +
        "
    <checkbox text='Mode
    lecture seule'/>" +
        "
    <checkbox text='Dupliquer
    la ligne
    automatiquement'/>" +
        "
    <checkbox text='Activer
    les polices monospaces'/>"
    +
        "          </vbox>" +
        "        </frame>" +
        "      </container>"
    +
        "" +
        "      <container
    id='#BackupContainer'
    invisible='yes'>" +
        "      <frame
    text='Backup'>" +
        "        <vbox
    expand='no'>" +
        "          <checkbox
    text='Activer le
    backup'/>" +
        "        </vbox>" +
        "      </frame>" +
        "      </container>"
    +
        "" +
        "    </multibox>" +
        "  </hbox>" +
        "</window>"

If ParseXML(#Xml, XML$) And
 XMLStatus(#Xml) =
 #PB_XML_Success

  If CreateDialog(#Dialog)
 And OpenXMLDialog(#Dialog,
```

```
    #Xml, "test")
72
73      AddGadgetItem(#ListView,
    -1, "Général")
74      AddGadgetItem(#ListView,
    -1, "Editeur")
75      AddGadgetItem(#ListView,
    -1, "Backup")
76
77      SetGadgetState(#ListView,
    0)
78
79      ShowPanels()
80
81      Repeat
82        Event =
    WaitWindowEvent()
83      Until Event =
    #PB_Event_CloseWindow
84
85    Else
86      Debug "Erreur de la
    bibliothèque -Dialog- : "
    + DialogError(#Dialog)
87    EndIf
88  Else
89    Debug "Erreur XML : " +
    XMLError(#Xml) + " (Ligne:
    " + XMLErrorLine(#Xml) +
    ")"
90  EndIf
```

Exemple : Boîte Gridbox

```
1   #Dialog = 0
2   #Xml = 0
3
4   XML$ = "<window
     id='#PB_Any' name='test'
     text='Gridbox'
     minwidth='auto'
     minheight='auto'
     flags='#PB_Window_ScreenCentered
     | #PB_Window_SystemMenu |
     #PB_Window_SizeGadget'>" +
5           "    <gridbox
    columns='6'>" +
6           "          <button
    text='Bouton 1' />" +
7           "          <button
    text='Bouton 2' />" +
8           "          <button
    text='Bouton 3'
    colspan='3' />" +
9           "          <button
    text='Bouton 4' />" +
10          "          <button
    text='Bouton 5'
```

```
    rowspan='2' />" +
          "          <button
    text='Bouton 6' />" +
          "          <button
    text='Bouton 7' />" +
          "          <button
    text='Bouton 8' />" +
          "          <button
    text='Bouton 9' />" +
          "          <button
    text='Bouton 10' />" +
          "          <button
    text='Bouton 11' />" +
          "          <button
    text='Bouton 12' />" +
          "    </gridbox>" +
          "  </window>"

  If ParseXML(#Xml, XML$) And
   XMLStatus(#Xml) =
   #PB_XML_Success

    If CreateDialog(#Dialog)
   And OpenXMLDialog(#Dialog,
   #Xml, "test")

      Repeat
        Event =
   WaitWindowEvent()
      Until Event =
   #PB_Event_CloseWindow

    Else
      Debug "Erreur de la
   bibliothèque -Dialog- : "
   + DialogError(#Dialog)
    EndIf
  Else
    Debug "Erreur XML : " +
   XMLError(#Xml) + " (Ligne:
   " + XMLErrorLine(#Xml) +
   ")"
  EndIf
```

Voir aussi

CreateDialog()

OS Supportés

Tous

90.9 RefreshDialog

Syntaxe

```
RefreshDialog(#Dialog)
```

Description

Met à jour la taille d'un Dialog pour l'ajuster au mieux après un changement.

Arguments

#**Dialog** L'interface utilisateur à rafraîchir.

Valeur de retour

Aucune.

Remarques

Par exemple, lorsque vous modifiez le texte d'un gadget, la taille de la boîte de dialogue aura probablement besoin d'être réajustée.

Voir aussi

CreateDialog()

OS Supportés

Tous

Chapitre 91

DragDrop

Généralités

Le 'Drag & Drop' (glisser-déposer) est aujourd'hui une technologie très largement utilisée et permet d'échanger de manière intuitive des données entre applications. Cette bibliothèque ajoute ces fonctionnalités aux gadgets et aux fenêtres juste en rajoutant quelques lignes de code. De plus, pour les programmeurs expérimentés, elle donne accès aux fonctions bas-niveau de l'API pour exploiter au maximum les capacités d'un système d'exploitation particulier.
OSX est limité pour l'instant et seules les images peuvent être cliqué-déposées.

OS Supportés

Tous

91.1 DragText

Syntaxe

```
Resultat = DragText(Texte$ [,
    Actions])
```

Description

Lance une opération de 'Drag & Drop' en utilisant une donnée de type texte.

Arguments

Texte$ Le texte à transférer.

Actions (optionnel) Si ce paramètre est omis, #PB_Drag_Copy sera utilisé comme valeur par défaut.
Peut être une combinaison de :

```
#PB_Drag_Copy: Le texte
  peut être copié
```

```
#PB_Drag_Move: Le texte
 peut être déplacé
#PB_Drag_Link: Le texte
 peut être lié
```

L'utilisateur peut décider de l'action à prendre en utilisant les touches Ctrl ou Shift. Les actions disponibles dépendent aussi du composant sur lequel le texte va être déposé. (Sous MacOSX, les actions ne sont traitées que comme une suggestion. La cible de dépôt peut encore choisir une autre action.)

Valeur de retour

Renvoie une des valeurs 'Actions' pour indiquer laquelle a été choisie par l'utilisateur, ou #PB_Drag_None si le 'Drag & Drop' a été annulé.
Note : si #PB_Drag_Move est renvoyé, le texte déposé ne sera pas automatiquement effacé de l'application PureBasic. Il faudra le faire en réponse à ce message.

Remarques

Un 'Drag & Drop' peut être démarré n'importe quand, mais il faut que le bouton gauche de la souris soit appuyé sinon l'opération se terminera immédiatement. Le moment le plus approprié pour lancer un 'Drag & Drop' est en réponse à un évènement sur un Gadget avec un EventType() valant #PB_EventType_DragStart.

Voir aussi

DragFiles() , DragImage() , DragPrivate() , DragOSFormats() , SetDragCallback()

OS Supportés

Windows, Linux

91.2 DragImage

Syntaxe

```
Resultat = DragImage(ImageID
    [, Actions])
```

Description

Lance une opération de 'Drag & Drop' en utilisant une donnée de type image.

Arguments

ImageID L'image à transférer. ImageID() peut être utilisé pour obtenir cet identifiant.

Actions (optionnel) Si ce paramètre est omis, #PB_Drag_Copy sera utilisé comme valeur par défaut. Peut être une combinaison de :

```
#PB_Drag_Copy: L'image
 peut être copiée
#PB_Drag_Move: L'image
 peut être déplacée
#PB_Drag_Link: L'image
 peut être liée
```

L'utilisateur peut décider de l'action à prendre en utilisant les touches Ctrl ou Shift. Les actions disponibles dépendent aussi du composant sur lequel le texte va être déposé. (Sous MacOSX, les actions ne sont traitées que comme une suggestion. La cible de dépôt peut encore choisir une autre action.)

Valeur de retour

Renvoie une des valeurs 'Actions' pour indiquer laquelle a été choisie par l'utilisateur, ou #PB_Drag_None si le 'Drag & Drop' a été annulé.
Note : si #PB_Drag_Move est renvoyé, l'image déposée ne sera pas automatiquement effacée de l'application PureBasic. Il faudra le faire en réponse à ce message.

Remarques

Un 'Drag & Drop' peut être démarré n'importe quand, mais il faut que le bouton gauche de la souris soit appuyé sinon l'opération se terminera immédiatement. Le moment le plus approprié pour lancer un 'Drag & Drop' est en réponse à un évènement sur un Gadget avec un EventType() valant #PB_EventType_DragStart.

Exemple

```
1  ; Glissez une image vers
   une application qui les
   accepte comme une
   application de bureau ou
   un programme graphique.
2  ;
```

```
3   If LoadImage(1,
     #PB_Compiler_Home +
     "examples/sources/data/PureBasicLogo.bmp")
4    If OpenWindow(1, 200,
     200, 400, 90, "Drag &
     Drop",
     #PB_Window_SystemMenu)
5       ImageGadget(1, 10, 10,
     380, 70, ImageID(1))
6
7       Repeat
8         Event =
     WaitWindowEvent()
9         If Event =
     #PB_Event_Gadget And
     EventGadget() = 1 And
     EventType() =
     #PB_EventType_DragStart
10          DragImage(ImageID(1))
11        EndIf
12      Until Event =
     #PB_Event_CloseWindow
13   EndIf
14  EndIf
```

Voir aussi

DragFiles() , DragText() , DragPrivate() , DragOSFormats() , SetDragCallback()

OS Supportés

Tous

91.3 DragFiles

Syntaxe

```
Resultat =
    DragFiles(Fichiers$ [,
    Actions])
```

Description

Lance une opération de 'Drag & Drop' en utilisant une liste de fichiers.

Arguments

Fichiers$ La liste des noms de fichiers ou de répertoires à transférer.
Les noms doivent être séparés par le caractère Chr(10) (fin de ligne).
Chaque nom de fichier doit contenir son chemin absolu, car l'application qui recevra ces données ne sera pas capable de résoudre les chemins relatifs.

Ces noms de fichiers doivent se référer à des fichiers existants, accessibles par l'application cible.

Actions (optionnel) Si ce paramètre est omis, #PB_Drag_Copy sera utilisé comme valeur par défaut.
Peut être une combinaison de :

```
#PB_Drag_Copy: Les
 fichiers peuvent être
 copiés
#PB_Drag_Move: Les
 fichiers peuvent être
 déplacés
#PB_Drag_Link: Les
 fichiers peuvent être
 liés
```

L'utilisateur peut décider de l'action à prendre en utilisant les touches Ctrl ou Shift. Les actions disponibles dépendent aussi du composant sur lequel le texte va être déposé. (Sous MacOSX, les actions ne sont traitées que comme une suggestion. La cible de dépôt peut encore choisir une autre action.)

Valeur de retour

Renvoie une des valeurs 'Actions' pour indiquer laquelle a été choisie par l'utilisateur, ou #PB_Drag_None si le 'Drag & Drop' a été annulé.
Note : contrairement aux autres fonctions de 'Drag & Drop', aucune action n'est à prendre lorsque #PB_Drag_Move est renvoyé. Comme les données déposées ne sont que le nom du fichier et pas le fichier lui-même, toutes actions entreprises sur le fichier seront faites par l'application cible.

Remarques

Un 'Drag & Drop' peut être démarré n'importe quand, mais il faut que le bouton gauche de la souris soit appuyé sinon l'opération se terminera immédiatement. Le moment le plus approprié pour lancer un 'Drag & Drop' est en réponse à un évènement sur un Gadget avec un EventType() valant #PB_EventType_DragStart.

Exemple

```
1  ; Sélectionnez quelques
    fichiers ou dossiers et
    glissez-les dans une autre
    application
```

```
;
If OpenWindow(1, 200, 200, 400, 400, "Drag & Drop", #PB_Window_SystemMenu)
  ExplorerListGadget(1, 10, 10, 380, 380, "*", #PB_Explorer_MultiSelect)

  Repeat
    Event = WaitWindowEvent()

    If Event = #PB_Event_Gadget And EventGadget() = 1 And EventType() = #PB_EventType_DragStart
      Files$ = ""
      For i = 0 To CountGadgetItems(1)-1
        If GetGadgetItemState(1, i) & #PB_Explorer_Selected
          Files$ + GetGadgetText(1) + GetGadgetItemText(1, i) + Chr(10)
        EndIf
      Next i

      DragFiles(Files$)
    EndIf

  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

DragText() , DragImage() , DragPrivate() , DragOSFormats() , SetDragCallback()

OS Supportés

Windows, Linux

91.4 DragPrivate

Syntaxe

```
Resultat = DragPrivate(Type [, Actions])
```

Description

Lance une opération de 'Drag & Drop' limitée à l'appicaton elle-même.

Arguments

Type La valeur qui identifiera la donnée à déposer dans l'application.
N'importe quelle valeur de type Long (.l) est utilisable.
La même valeur devra être spécifiée à EnableGadgetDrop() ou EnableWindowDrop() pour les Gadgets / Fenêtres qui accepteront cette donnée.
Cela permet de définir avec exactitude quel Gadget/Fenêtre acceptera tel type de données, et ainsi de réaliser des schémas 'Drag & Drop' complexes.

Actions (optionnel) Si ce paramètre est omis, #PB_Drag_Copy sera utilisé comme valeur par défaut.
Peut être une combinaison de :

```
#PB_Drag_Copy: Le texte
 peut être copié
#PB_Drag_Move: Le texte
 peut être déplacé
#PB_Drag_Link: Le texte
 peut être lié
```

L'utilisateur peut décider de l'action à prendre en utilisant les touches Ctrl ou Shift. Les actions disponibles dépendent aussi du composant sur lequel le texte va être déposé.

Valeur de retour

Renvoie une des valeurs 'Actions' pour indiquer laquelle a été choisie par l'utilisateur, ou #PB_Drag_None si le 'Drag & Drop' a été annulé.
Si l'opération n'a pas été annulée, la boucle d'évènement recevra #PB_Event_WindowDrop ou #PB_Event_GadgetDrop de type #PB_Drop_Private.

Remarques

Contrairement aux autres commandes qui démarrent un 'Drag & Drop', la donnée ne pourra être déposée qu'au sein de l'application (les données transférées avec les commandes telles que DragText() ou DragImage() peuvent aussi être acceptés par les autres applications). Cette commande permet d'ajouter des fonctionnalités de 'Drag & Drop' entre les Gadgets ou les Fenêtres en utilisant des données qui ne seraient pas interprétables par d'autres applications.
Un 'Drag & Drop' peut être démarré n'importe quand, mais il faut que le bouton

gauche de la souris soit appuyé sinon l'opération se terminera immédiatement. Le moment le plus approprié pour lancer un 'Drag & Drop' est en réponse à un évènement sur un Gadget avec un EventType() valant #PB_EventType_DragStart.
Si l'opération n'a pas été interrompue, la boucle d'événements recevra un événement #PB_Event_WindowDrop ou #PB_Event_GadgetDrop de type #PB_Drop_Private.

Voir aussi

DragText() , DragImage() , DragFiles() , DragOSFormats() , SetDragCallback()

OS Supportés

Windows, Linux

91.5 DragOSFormats

Syntaxe

```
Resultat =
    DragOSFormats(Formats(),
    NbFormats [, Actions])
```

Description

Lance une opération de 'Drag & Drop' avec une liste de formats propre au système d'exploitation. Ceci permet de supporter des formats de données qui ne sont pas nativement gérés par PureBasic tout en gardant la simplicité d'utilisation fournie par cette bibliothèque.

Arguments

Formats() Est un tableau de structures 'DragDataFormat' contenant un ou plusieurs formats à gérer.

```
1  Structure DragDataFormat
2    Format.i     ; L'identifant de l'OS concernant le format (voir ci-dessous)
3    *Buffer      ; La zone mémoire contenant la donnée dans ce format
4    Size.i       ; La taille de la zone mémoire
5  EndStructure
```

Windows :

Sous Windows, le champ 'Format' est une valeur de type 'CLIPBOARDFORMAT'. Ce peut être n'importe quel format standard (voir le SDK Windows), ou un format enregistré avec l'API RegisterClipboardFormat_().

Linux :

Sous Linux, le champ 'Format' est une valeur de type 'GdkAtom'. Il peut être créé avec la fonction gdk_atom_intern_(). Les "atomes" sont généralement assimilés à des types mime usuels (i.e "text/html" pour des données HTML). L'atome peut aussi être créé avec n'importe quelle chaîne capable d'être correctement interprétée par l'application cible.

MacOSX :

Sur MacOSX, le champ 'Format' précise un type pour le contenu du presse-papier via le "scrap manager" (le gestionnaire du presse-papier). Il possède quatre caractères alphanumériques constants, 'TEXT' par exemple. Il y a un certain nombre de types prédéfinis, mais on peut également utiliser des valeurs personnalisées à condition que le programme cible puisse les interpréter correctement.

NbFormats Le nombre de formats dans le tableau.

Si plusieurs formats sont transférés, l'application cible acceptera le premier qu'elle reconnaîtra. Le format qui fournit le plus d'informations (donc qui représente le plus fidèlement la donnée) doit être le premier dans le tableau, suivi par les autres qui sont de moins en moins précis. De cette façon, chaque application recevra la meilleure réprésentation de la donnée, en fonction de ce qu'elle sait interpréter.

Actions (optionnel) Si ce paramètre est omis, #PB_Drag_Copy sera utilisé comme valeur par défaut. Peut être une combinaison de :

```
#PB_Drag_Copy: La donnée
 peut être copiée
#PB_Drag_Move: La donnée
 peut être déplacée
#PB_Drag_Link: La donnée
 peut être liée
```

L'utilisateur peut décider de l'action à prendre en utilisant les touches Ctrl ou Shift. Les actions disponibles dépendent aussi du composant sur lequel le texte va être déposé. (Sous MacOSX, les actions ne sont traitées que comme une suggestion. La cible de dépôt peut encore choisir une autre action.)

Valeur de retour

Renvoie une des valeurs 'Actions' pour indiquer laquelle a été choisie par l'utilisateur, ou #PB_Drag_None si le 'Drag & Drop' a été annulé.
Note : si #PB_Drag_Move est renvoyé, la donnée déposée ne sera pas automatiquement effacée de l'application PureBasic. Il faudra le faire en réponse à ce message.

Remarques

Un 'Drag & Drop' peut être démarré n'importe quand, mais il faut que le bouton gauche de la souris soit appuyé sinon l'opération se terminera immédiatement. Le moment le plus approprié pour lancer un 'Drag & Drop' est en réponse à un évènement sur un Gadget avec un EventType() valant #PB_EventType_DragStart.

Voir aussi

DragText() , DragImage() , DragFiles() , DragPrivate() , SetDragCallback()

OS Supportés

Tous

91.6 EnableGadgetDrop

Syntaxe

```
EnableGadgetDrop(#Gadget,
    Format, Actions [,
    TypeInterne])
```

Description

Permet à un gadget d'être la cible d'un 'Drag & Drop' d'un format spécifique. Quand l'utilisateur voudra déposer une donnée de ce format sur le gadget, le curseur sera modifié en conséquence, indiquant que l'action est possible.

Arguments

#Gadget Le numéro du gadget qui supportera le 'Drag & Drop'.

Format Indique quel type de donnée pourra être déposé sur le gadget.
Il peut prendre une des valeurs suivantes ou un identifiant de format système (voir DragOSFormats() pour plus d'informations).

```
#PB_Drop_Text    : Accepte
 du texte sur ce gadget
#PB_Drop_Image   : Accepte
 des images sur ce gadget
#PB_Drop_Files   : Accepte
 des noms de fichiers sur
 ce gadget
#PB_Drop_Private: Accepte
 un 'Drag & Drop' interne
```

Actions L'utilisateur peut décider de l'action à prendre en utilisant les touches Ctrl ou Shift. Les actions disponibles dépendent aussi du composant sur lequel le texte va être déposé.
Peut être une combinaison de :

```
#PB_Drag_None: Le format
 de données ne sera pas
 accepté par ce gadget
#PB_Drag_Copy: La donnée
 peut être copiée
#PB_Drag_Move: La donnée
 peut être déplacée
#PB_Drag_Link: La donnée
 peut être liée
```

TypeInterne (optionnel) Uniquement nécessaire quand #PB_Drop_Private est utilisé comme format.
Il indique le type du 'Drag & Drop' interne à accepter.
Voir DragPrivate() pour plus d'informations.
Ce paramètre est ignoré pour les autres formats.

Valeur de retour

Aucune.

Remarques

Plusieurs types de formats peuvent être acceptés par un même gadget. Si la source des données transférées a spécifié plusieurs formats qui sont pris en compte par le gadget, celui qui aura été activé en dernier sera accepté. Donc le format le plus approprié devra être activé en dernier.

Quand une donnée sera déposée sur le gadget, le programme recevra un évènement #PB_Event_GadgetDrop. EventGadget() indiquera le gadget concerné et les commandes 'Event' de cette bibliothèque permettront de récupérer la donnée déposée.

Voir aussi

EnableWindowDrop() , EventDropType() , EventDropAction() , SetDropCallback()

OS Supportés

Tous

91.7 EnableWindowDrop

Syntaxe

```
EnableWindowDrop(#Fenetre,
    Format, Actions [,
    TypeInterne])
```

Description

Permet à une fenêtre d'être la cible d'un 'Drag & Drop' d'un format spécifique. Quand l'utilisateur voudra déposer une donnée de ce format sur la fenêtre, le curseur sera modifié en conséquence, indiquant que l'action est possible.

Arguments

#**Fenetre** Le numéro de la fenêtre qui supportera le 'Drag & Drop'.

Format Indique quel type de donnée pourra être déposé sur la fenêtre.
Il peut prendre une des valeurs suivantes ou un identifiant de format système (voir DragOSFormats() pour plus d'informations).

```
#PB_Drop_Text    : Accepte
 du texte sur cette
 fenêtre
#PB_Drop_Image   : Accepte
 des images sur cette
 fenêtre
#PB_Drop_Files   : Accepte
 des noms de fichiers sur
 cette fenêtre
#PB_Drop_Private: Accepte
 un 'Drag & Drop' interne
```

Actions L'utilisateur peut décider de l'action à prendre en utilisant les touches Ctrl ou Shift. Les actions disponibles

dépendent aussi du composant sur lequel le texte va être déposé.
Peut être une combinaison de :

```
#PB_Drag_None: Le format
 de données ne sera pas
 accepté par cette fenêtre
#PB_Drag_Copy: La donnée
 peut être copiée
#PB_Drag_Move: La donnée
 peut être déplacée
#PB_Drag_Link: La donnée
 peut être liée
```

TypeInterne (optionnel) Uniquement nécessaire quand #PB_Drop_Private est utilisé comme format.
Il indique le type du 'Drag & Drop' interne à accepter.
Voir DragPrivate() pour plus d'informations.

Valeur de retour

Aucune.

Remarques

Plusieurs types de formats peuvent être acceptés par une même fenêtre. Si la source des données transférées a spécifié plusieurs formats qui sont pris en compte par la fenêtre, celui qui aura été activé en dernier sera accepté. Donc le format le plus approprié devra être activé en dernier.
Quand une donnée sera déposée sur la fenêtre, le programme recevra un évènement #PB_Event_WindowDrop.
EventWindow() indiquera la fenêtre concernée et les commandes 'Event' de cette bibliothèque permettront de récupérer la donnée déposée.

Voir aussi

EnableGadgetDrop() , EventDropType() , EventDropAction() , SetDropCallback()

OS Supportés

Tous

91.8 EventDropAction

Syntaxe

```
Resultat = EventDropAction()
```

Description

Après avoir reçu #PB_Event_GadgetDrop ou #PB_Event_WindowDrop avec WaitWindowEvent() ou WindowEvent() , cette fonction renvoie l'action qui devra être gérée pour la donnée transférée.

Arguments

Aucun.

Valeur de retour

```
#PB_Drag_Copy: La donnée
 devra être copiée
#PB_Drag_Move: La donnée
 devra être déplacée
                (La source
 du transfert est
 responsable
                d'effacer la
 donnée originale)
#PB_Drag_Link: La donnée
 devra être liée
```

Voir aussi

EnableGadgetDrop() , EnableWindowDrop() , EventDropType() , EventDropX() , EventDropY()

OS Supportés

Tous

91.9 EventDropType

Syntaxe

```
Resultat = EventDropType()
```

Description

Après avoir reçu #PB_Event_GadgetDrop ou #PB_Event_WindowDrop avec WaitWindowEvent() ou WindowEvent() , cette fonction renvoie le format de la donnée transférée.

Arguments

Aucun.

Valeur de retour

Renvoie une des valeurs suivantes, ou un identifiant de format système (voir DragOSFormats() pour plus d'informations).

```
  #PB_Drop_Text    : Un texte
    a été déposé.
    (EventDropText()
  pour le récupérer)
  #PB_Drop_Image   : Une image
    a été déposée.
    (EventDropImage()
 pour la récupérer)
  #PB_Drop_Files   : Des noms
    de fichiers ont été
    déposés.  (EventDropFiles()
 pour les récupérer)
  #PB_Drop_Private: Une
    opération interne a été
    effectuée.
    (EventDropPrivate()
 pour connaître le type)
```

Remarques

Pour gérer les formats systèmes, EventDropBuffer() et EventDropSize() peuvent être utilisés.

Voir aussi

EnableGadgetDrop() , EnableWindowDrop() , EventDropAction() , EventDropText() , EventDropImage() , EventDropFiles() , EventDropPrivate() , EventDropBuffer() , EventDropSize() , EventDropX() , EventDropY()

OS Supportés

Tous

91.10 EventDropText

Syntaxe

```
Resultat\$ = EventDropText()
```

Description

Renvoie le texte déposé.

Arguments

Aucun.

Valeur de retour

Renvoie le texte déposé après avoir reçu #PB_Event_GadgetDrop ou #PB_Event_WindowDrop avec WaitWindowEvent() ou WindowEvent() et

un format (récupérable avec EventDropType()) de type #PB_Drop_Text.

Voir aussi

EnableGadgetDrop() , EnableWindowDrop() , EventDropType() , EventDropAction() , EventDropX() , EventDropY()

OS Supportés

Tous

91.11 EventDropImage

Syntaxe

```
Resultat =
    EventDropImage(#Image [,
    Profondeur])
```

Description

Après avoir reçu #PB_Event_GadgetDrop ou #PB_Event_WindowDrop avec WaitWindowEvent() ou WindowEvent() et un format (récupérable avec EventDropType()) de type #PB_Drop_Image, cette fonction permet de récupérer l'image déposée.

Arguments

#Image Le numéro de la nouvelle image à créer.
Si #PB_Any est utilisé, le nouveau numéro de l'image sera renvoyé dans 'Resultat'.

Profondeur (optionnel) Profondeur de couleur de la nouvelle image.
Les valeurs possibles sont :

```
24 : 24 Bits (par défaut)
32 : 32 Bits
```

Valeur de retour

Renvoie une valeur non nulle si l'image a été créée avec succès et zéro sinon.
Si #PB_Any a été utilisée pour le paramètre #Image alors le nombre généré est renvoyé en cas de succès.

Voir aussi

EnableGadgetDrop() , EnableWindowDrop() , EventDropType() , EventDropAction() , EventDropX() , EventDropY()

OS Supportés

Tous

91.12 EventDropFiles

Syntaxe

```
Resultat\$ = EventDropFiles()
```

Description

Après avoir reçu #PB_Event_GadgetDrop ou #PB_Event_WindowDrop avec WaitWindowEvent() ou WindowEvent() et un format (récupérable avec EventDropType()) de type #PB_Drop_Files, cette fonction permet de récupérer les noms de fichiers déposés.

Arguments

Aucun.

Valeur de retour

Renvoie une liste de fichiers et de répertoires, avec leurs chemin absolu. Les noms de fichiers sont séparés par le caractère Chr(10) (fin de ligne).

Voir aussi

EnableGadgetDrop() , EnableWindowDrop() , EventDropType() , EventDropAction() , EventDropX() , EventDropY()

OS Supportés

Tous

91.13 EventDropPrivate

Syntaxe

```
Resultat = EventDropPrivate()
```

Description

Après avoir reçu #PB_Event_GadgetDrop ou #PB_Event_WindowDrop avec WaitWindowEvent() ou WindowEvent() et un format (récupérable avec EventDropType()) de type #PB_Drop_Private, cette fonction renvoie le 'TypeInterne' qui a été déposé (voir DragPrivate() pour plus d'informations).

Arguments

Aucun.

Valeur de retour

Renvoie la valeur de type privé utilisé lors du démarrage de l'opération de Drag & Drop. (Voir DragPrivate() pour plus d'informations.)

Voir aussi

EnableGadgetDrop() , EnableWindowDrop() , EventDropType() , EventDropAction() , EventDropX() , EventDropY()

OS Supportés

Tous

91.14 EventDropBuffer

Syntaxe

```
*Resultat = EventDropBuffer()
```

Description

Après avoir reçu #PB_Event_GadgetDrop ou #PB_Event_WindowDrop avec WaitWindowEvent() ou WindowEvent() et un format système, cette fonction permet d'accéder à la donnée déposée.

Arguments

Aucun.

Valeur de retour

Renvoie l'adresse mémoire de la donnée déposée.

Remarques

Cette zone de mémoire est gérée par la bibliothèque et ne doit en aucun cas être détruite. Elle reste valide jusqu'au prochain appel à la commande WaitWindowEvent() ou WindowEvent() , donc son contenu devra être copié s'il est nécessaire de le conserver plus longtemps.
EventDropSize() renvoie la taille de cette zone mémoire (buffer).

Voir aussi

EventDropSize() , EnableGadgetDrop() , EnableWindowDrop() , EventDropType() , EventDropAction() , EventDropX() , EventDropY()

OS Supportés

Tous

91.15 EventDropSize

Syntaxe

```
Resultat = EventDropSize()
```

Description

Après avoir reçu #PB_Event_GadgetDrop ou #PB_Event_WindowDrop avec WaitWindowEvent() ou WindowEvent() et un format système, cette fonction renvoie la taille de la donnée déposée.

Arguments

Aucun.

Valeur de retour

Renvoie la taille en octets de la zone de mémoire renvoyée par EventDropBuffer() .

Voir aussi

EventDropBuffer() , EnableGadgetDrop() , EnableWindowDrop() , EventDropType() , EventDropAction() , EventDropX() , EventDropY()

OS Supportés

Tous

91.16 EventDropX

Syntaxe

```
Resultat = EventDropX()
```

Description

Après avoir reçu #PB_Event_GadgetDrop ou #PB_Event_WindowDrop avec WaitWindowEvent() ou WindowEvent() , cette fonction renvoie la position en X dans la fenêtre ou le gadget sur lequel la donnée a été déposée.

Arguments

Aucun.

Valeur de retour

Renvoie la coordonnée X de la zone de dépôt par rapport à la fenêtre ou au gadget dans lequel les données ont été déposées.

Voir aussi

EnableGadgetDrop() , EnableWindowDrop() , EventDropType() , EventDropAction() , EventDropY()

OS Supportés

Tous

91.17 EventDropY

Syntaxe

```
Resultat = EventDropY()
```

Description

Après avoir reçu #PB_Event_GadgetDrop ou #PB_Event_WindowDrop avec WaitWindowEvent() ou WindowEvent() , cette fonction renvoie la position en Y dans la fenêtre ou le gadget sur lequel la donnée a été déposée.

Arguments

Aucun.

Valeur de retour

Renvoie la coordonnée Y de la zone de dépôt par rapport à la fenêtre ou au gadget dans lequel les données ont été déposées.

Voir aussi

EnableGadgetDrop() , EnableWindowDrop() , EventDropType() , EventDropAction() , EventDropX()

OS Supportés

Tous

91.18 SetDragCallback

Syntaxe

```
SetDragCallback(@DragCallback())
```

Description

Installe une fonction callback qui sera appelée automatiquement quand une opération de 'Drag & Drop' sera démarrée depuis cette application. La callback permet de modifier le comportement par défaut fourni par PureBasic, par exemple pour changer le curseur via l'API du système d'exploitation. La fonction callback est dépendante de l'OS.

Arguments

@DragCallback() L'adresse d'une fonction à appeler lors du début d'une opération de drag & drop.
La forme et l'objet du rappel dépend de l'OS.
Elle doit prendre la forme suivante :
Windows :

```
Procedure
 DragCallback(Action)

  ProcedureReturn #True
EndProcedure
```

La callback est appelée durant toute l'opération de 'Drag & Drop'. Le paramètre 'Action' indique quelle action serait prise en compte si l'utilisateur relachait le bouton de la souris à cet instant. Il peut prendre une des valeurs suivantes :

```
#PB_Drag_None: La
 donnée ne sera pas
 acceptée
#PB_Drag_Copy: La
 donnée sera copiée
#PB_Drag_Move: La
 donnée sera déplacée
#PB_Drag_Link: La
 donnée sera liée
```

La callback permet de fournir un curseur ou une image différente. Si c'est le cas, il faudra renvoyer #False, sinon le curseur par défaut sera utilisé.

Linux :

```
1  Procedure
    DragCallback(*Context.GdkDragContext,
    isStart)
2
3  EndProcedure
```

La callback est appelée seulement au début et à la fin d'une opération de 'Drag & Drop'. Le paramètre '*Context' indique le contexte gdk de cette opération, 'isStart' indique si c'est le début ou la fin de l'opération. La valeur renvoyée par la callback est ignorée.
Des fonctions Gtk comme gtk_drag_set_icon_pixbuf_() peuvent être utilisées dans la callback pour définir une image différente durant l'opération.

MacOSX :

```
1  Procedure
    DragCallback(DragReference,
    isStart)
2
3  EndProcedure
```

La callback n'est appelée seulement qu'au début ou à la fin de l'opération "Drag & Drop" (="glisser/déposer"). Le paramètre 'DragReference' précise le contexte 'Drag' de Carbon pour cette opération, contexte qui prend la forme d'une valeur 'DragRef'. 'isStart' précise s'il s'agit du début ou de la fin de l'opération 'Drag & Drop'. La valeur de retour de cette fonction callback est ignorée.
Toutes les fonctions du "Drag Manager" de Carbon peuvent être utilisées pour modifier l'opération 'Drag' avant qu'elle ne commence (ajouter plusieurs choix par exemple) et pour obtenir plus d'informations une fois qu'elle est terminée.

Valeur de retour

Aucune.

Voir aussi

SetDropCallback()

OS Supportés

Tous

91.19 SetDropCallback

Syntaxe

```
SetDropCallback(@DropCallback())
```

Description

Installe une fonction callback qui sera appelée automatiquement quand une donnée en cours de transfert se trouve au dessus d'une fenêtre ou d'un gadget pouvant l'accepter (voir EnableGadgetDrop() / EnableWindowDrop()). La callback permet de modifier le comportement par défaut de PureBasic, en fournissant par exemple des effets visuels supplémentaires en fonction de la position de la donnée.

Arguments

@DropCallback() L'adresse de la fonction callback.
La callback est appelée quand la souris entre, se déplace et quitte une zone propice au 'Drag & Drop', pour permettre de réagir contextuellement. De plus la callback peut interdire le transfert si nécessaire. Le curseur ne doit pas être modifié dans cette fonction, car seule la source du transfert en est responsable.
Les paramètres de la callback sont décrits ci-dessous :

```
1  Procedure
     DropCallback(Handle,
     Etat, Format, Action, x,
     y)
2
3    ProcedureReturn #True
4  EndProcedure
```

Le premier paramètre indique l' identifiant (Handle) spécifique de l'OS" pour le gadget ou la fenêtre cible. Sous Windows, il s'agit de la valeur HWND, sous Linux d'un pointeur GtkWidget et sous MacOSX de la valeur ControlRef ou WindowRef. Pour obtenir ces valeurs, vous pouvez utiliser GadgetID() ou WindowID() pour le gadget ou la fenêtre cible.
'Etat' représente l'état actuel du 'Drag & Drop' en cours. Peut prendre l'une des valeurs suivantes :

```
#PB_Drag_Enter : La souris
 est entrée dans la zone
 du gadget ou de la
 fenêtre
#PB_Drag_Update: La souris
 se déplace dans le
 gadget ou la fenêtre, ou
 l'action a changé
#PB_Drag_Leave : La souris
 est sortie de la zone du
 gadget ou de la fenêtre
 (Format, Action, x, y
 sont à 0)
#PB_Drag_Finish: Le 'Drag
 & Drop' est terminé
```

'Format' représente le format de la donnée. Peut prendre l'une des valeurs suivantes, ou un identifiant de format système (voir DragOSFormats() pour plus d'informations) :

```
#PB_Drop_Text    : Le texte
 est accepté sur ce
 gadget ou cette fenêtre
#PB_Drop_Image   : Les
 images sont acceptées
 sur ce gadget ou cette
 fenêtre
#PB_Drop_Files   : Les
 fichiers sont acceptés
 sur ce gadget ou cette
 fenêtre
#PB_Drop_Private: Le type
 interne est accepté sur
 ce gadget ou cette
 fenêtre
```

Le paramètre 'Action' indique quelle action serait prise en compte si l'utilisateur relachait le bouton de la souris à cet instant. Il peut prendre une des valeurs suivantes :

```
#PB_Drag_None: La donnée
 ne sera pas acceptée
#PB_Drag_Copy: La donnée
 sera copiée
#PB_Drag_Move: La donnée
 sera déplacée
#PB_Drag_Link: La donnée
 sera liée
```

En renvoyant #True, la callback permet que l'action se déroule normalement. En renvoyant #False, la callback interdit l'action (le curseur sera changé en "panneau d'interdiction" par la source du transfert). Si l'Etat' est #PB_Drag_Finish, renvoyer #False annulera le 'Drag & Drop'.

Valeur de retour

Aucune.

Voir aussi

SetDragCallback()

OS Supportés

Tous

Chapitre 92

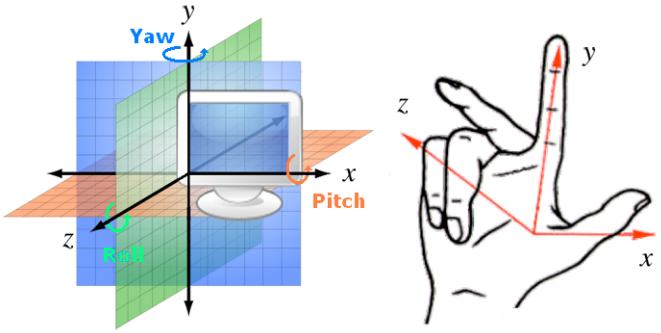
Engine3D

Généralités

PureBasic dispose d'un accès simplifié à OGRE, un moteur 3D OpenSource très performant.
Ce choix a été fait car il aurait été dommage de réinventer la roue plutôt que de supporter et de pousser le développement d'un excellent produit déjà disponible.
OGRE est toujours en développement et de nombreuses nouvelles fonctionnalités devraient lui être ajoutées au fur et à mesure.
Plus d'informations à propos de ce moteur 3D sont disponible sur :
http ://www.ogre3d.org. La licence relative à l'usage de OGRE est consultable ici (plus d'informations ici Licensing FAQ).
Note : Si vous utilisez le moteur 3D de PureBasic dans vos projets avec l'intention de distribuer votre exécutable, vous devrez copier la Engine3D.dll du dossier PureBasic/Compilers dans le répertoire de votre projet.

Repère dans l'espace

OGRE utilise les conventions spatiales suivantes :
Le repère spatial de Ogre est direct et la règle des 3 doigts de la main droite nous permet de retrouver facilement la position des axes X, Y et Z dans l'espace.



Attention, l'axe Z est face à vous.
Définition de : Roulis, Tangage et Lacet :
- Le tangage (pitch) est une rotation autour de l'axe X.
- Le lacet (yaw) est une rotation autour de l'axe Y.
- Le roulis (roll) est une rotation autour de l'axe Z.
Sens de rotation :
Le sens positif (direct) d'une rotation autour d'un axe peut être retrouvé grâce à ce shéma. Le pouce dans le sens de l'axe comme indiqué, la direction des doigts donne le sens positif de la rotation. (Utilisez votre main droite).

Importante note :

Dans toutes les bibliothèques 3D fournies avec PureBasic, toutes les variables et les valeurs renvoyées sont de type 'Float' exeptés les ID et les index, même si ce n'est pas indiqué par un '.f'.

OS Supportés

Tous

92.1 Add3DArchive

Syntaxe

```
Add3DArchive(Chemin$, Type)
```

Description

Ajoute un nouveau chemin absolu ou relatif dans la liste actuelle des répertoires disponibles pour les fonctions 3D telles que texture , mesh , sky , etc... world)

Arguments

Chemin$ Le chemin d'accès à ajouter à la liste des chemins vers les ressources 3D. Cela peut être un chemin sur le système réel, ou une archive ZIP.

Type

```
#PB_3DArchive_FileSystem
 : Répertoire classique,
situé sur une unité
disque
#PB_3DArchive_Zip
 : Fichier .zip (archive
compressée)
```

Valeur de retour

Aucune.

Remarques

Ce mode spécial de gestion des accès aux données est dû au fait que ces données peuvent se trouver indifféremment sur le disque (répertoire classique) ou dans une archive (fichier .zip). C'est un moyen très flexible d'abstraire cet accès aux données et permet une flexibilité importante. Par exemple il est possible de créer un .zip de tout le répertoire 'Data' où sont stockées les données relatives à l'application 3D et d'y accéder comme si c'était un répertoire classique.

Exemple

```
1  Add3DArchive("MyData.zip",
    #PB_3DArchive_Zip) ;
    Ajoute le .zip comme un
    répertoire possible
2  LoadTexture(0,
    "MyTexture.jpg")
                  ; Charge
    MyTexture.jpg à partir du
    zip
3  LoadTexture(1,
    "World/Grass.jpg")
                ; Charge
    Grass.jpg à partir du zip,
    situé dans le sous
    répertoire World/
```

Voir aussi

InitEngine3D()

OS Supportés

Tous

92.2 AmbientColor

Syntaxe

```
AmbientColor(Couleur)
```

Description

Change la couleur ambiante du monde 3D.

Arguments

Couleur Nouvelle couleur ambiante. RGB() peut être utilisée pour obtenir une couleur valide.

Valeur de retour

Aucune.

Voir aussi

Fog()

OS Supportés

Tous

92.3 AntialiasingMode

Syntaxe

```
AntialiasingMode(Mode)
```

Description

Change le mode de l'antialiasing (anti crénelage) en mode plein écran.

Arguments

Mode Peut être l'une des valeurs suivantes :

```
#PB_AntialiasingMode_None:
 Pas d'antialiasing
 (valeur par défaut).
#PB_AntialiasingMode_x2  :
 antialiasing  x2 (FSAA).
#PB_AntialiasingMode_x4  :
 antialiasing  x4 (FSAA).
#PB_AntialiasingMode_x6  :
 antialiasing  x6 (FSAA).
```

Valeur de retour

Aucune.

Remarques

Cette fonction doit être appelée avant OpenScreen() .
En fonction de la carte graphique, cette commande peut avoir un impact important sur les performances de rendu.

Voir aussi

InitEngine3D()

OS Supportés

Tous

92.4 ConvertLocalToWorldPosition

Syntaxe

```
ConvertLocalToWorldPosition(ObjetID,
   X, Y, Z)
```

Description

Convertit les coordonnées locales X, Y, Z en coordonnées dans le monde 3D.

Arguments

ObjetID L'ID (Identifiant) de l'objet concerné. Il peut être un des types suivants :

```
  - Camera          :
    Utiliser CameraID()
pour obtenir un ID valide.
  - Entité          :
    Utiliser EntityID()
pour obtenir un ID valide.
  - Lumière         :
    Utiliser LightID()
pour obtenir un ID valide.
  - Mesh            :
    Utiliser MeshID()
pour obtenir un ID valide.
  - Noeud           :
    Utiliser NodeID()
pour obtenir un ID valide.
  - Texte3D         :
    Utiliser Text3DID()
pour obtenir un ID valide.
  - Emetteur de Particle :
    Utiliser
    ParticleEmitterID()
pour obtenir un ID valide.
  - Groupe de billboards :
    Utiliser
    BillboardGroupID()
pour obtenir un ID valide.
```

X, Y, Z Les coordonnées locales à convertir.

Remarques

GetX() , GetY() et GetZ() seront utilisés pour obtenir les coordonnées converties.

Valeur de retour

Aucune.

Voir aussi

ConvertWorldToLocalPosition()

OS Supportés

Tous

92.5 ConvertWorldToLocalPosition

Syntaxe

```
ConvertWorldToLocalPosition(ObjetID,
   X, Y, Z)
```

Description

Convertit les coordonnées dans le monde 3D X, Y, Z en coordonnées locales.

Arguments

ObjetID L'ID (Identifiant) de l'objet concerné. Il peut être un des types suivants :

```
  - Camera          :
    Utiliser CameraID()
  pour obtenir un ID valide.
  - Entité          :
    Utiliser EntityID()
  pour obtenir un ID valide.
  - Lumière         :
    Utiliser LightID()
  pour obtenir un ID valide.
  - Mesh            :
    Utiliser MeshID()
  pour obtenir un ID valide.
  - Noeud           :
    Utiliser NodeID()
  pour obtenir un ID valide.
  - Texte3D         :
    Utiliser Text3DID()
  pour obtenir un ID valide.
  - Emetteur de Particle :
    Utiliser
    ParticleEmitterID()
```

```
pour obtenir un ID valide.
 - Groupe de billboards :
  Utiliser
  BillboardGroupID()
pour obtenir un ID valide.
```

X, Y, Z Les coordonnées locales à convertir.

Remarques

GetX() , GetY() et GetZ() seront utilisés pour obtenir les coordonnées converties.

Valeur de retour

Aucune.

Voir aussi

ConvertLocalToWorldPosition()

OS Supportés

Tous

92.6 Engine3DStatus

Syntaxe

```
Resultat =
    Engine3DStatus(Type)
```

Description

Renvoie le statut du moteur 3D.

Arguments

Type Peut être l'une des valeurs suivantes :

```
#PB_Engine3D_NbRenderedTriangles
  : Nombre de triangles
 affichés dans la
 dernière image.
#PB_Engine3D_NbRenderedBatches
  : Nombre de batches
 affichés dans la
 dernière image.
#PB_Engine3D_CurrentFPS :
 Nombre d'images par
 seconde actuel
#PB_Engine3D_AverageFPS :
 Nombre d'images par
 seconde moyen depuis que
 le moteur 3D est lancé
```

```
#PB_Engine3D_MaximumFPS :
 Nombre d'images par
 seconde maximum depuis
 que le moteur 3D est
 lancé
#PB_Engine3D_MinimumFPS :
 Nombre d'images par
 seconde minimum depuis
 que le moteur 3D est
 lancé
#PB_Engine3D_ResetFPS   :
 Réinitialise les
 statistiques (valeurs
 'Minimum', 'Maximum' et
 'Moyen')
```

Valeur de retour

La valeur renvoyée dépend du type demandé.

OS Supportés

Tous

92.7 EnableWorldCollisions

Syntaxe

```
EnableWorldCollisions(Etat)
```

Description

Active ou désactive les collisions dans l'environnement 3D.

Arguments

Etat Avec une valeur non nulle, la gestion des collisions est activée (mode par défaut).
Les collisions fonctionnent avec toutes les entités qui ont un corps (body) créé avec CreateEntityBody() .
Les collisions ne fonctionnent que si le moteur physique est activé avec EnableWorldPhysics() .

Valeur de retour

Aucune.

Voir aussi

EnableWorldPhysics()

OS Supportés

Tous

92.8 EnableWorldPhysics

Syntaxe

```
EnableWorldPhysics(Etat)
```

Description

Active ou désactive le moteur physique de l'environnement 3D.

Arguments

Etat Avec une valeur non nulle, le moteur physique est activé (mode par défaut). Toutes les entités qui ont un corps (body) attribué par la commande CreateEntityBody() seront affectées par le moteur physique.
Les collisions peuvent être activées avec EnableWorldCollisions() .
Les collisions ne fonctionnent que si le moteur physique est activé.

Valeur de retour

Aucune.

Voir aussi

EnableWorldCollisions()

OS Supportés

Tous

92.9 ExamineWorldCollisions

Syntaxe

```
Resultat =
    ExamineWorldCollisions(Contacts)
```

Description

Examine les collisions survenues dans le monde depuis le dernier appel.

Arguments

Contacts #True : Les informations de collision seront mémorisées et pourront être récupérées avec WorldCollisionContact() , WorldCollisionNormal() et WorldCollisionAppliedImpulse() .
#False : Aucune information de contact ne sera collectée (plus rapide).

Valeur de retour

Renvoie une valeur non nulle si les collisions peuvent être examinées, zéro sinon.

Remarques

Les collisions doivent être activées avec EnableWorldCollisions() avant d'utiliser cette commande.
Pour faire défiler les collisions, utilisez NextWorldCollision() .

Voir aussi

EnableWorldCollisions() , NextWorldCollision()

OS Supportés

Tous

92.10 NextWorldCollision

Syntaxe

```
Resultat =
    NextWorldCollision()
```

Description

Va à la collision suivante.

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle s'il y a une autre collision à examiner, ou zéro sinon.

Remarques

ExamineWorldCollisions() doit être appelé avec succès avant d'utiliser cette commande. Pour obtenir plus d'informations sur la collision actuelle, utilisez FirstWorldCollisionEntity() (), SecondWorldCollisionEntity() , WorldCollisionContact() , WorldCollisionNormal() et WorldCollisionAppliedImpulse() .

Voir aussi

ExamineWorldCollisions() , FirstWorldCollisionEntity() , SecondWorldCollisionEntity() , WorldCollisionContact() , WorldCollisionNormal() , WorldCollisionAppliedImpulse()

OS Supportés

Tous

92.11 FirstWorldCollisionEntity

Syntaxe

```
Resultat =
    FirstWorldCollisionEntity()
```

Description

Renvoie le numéro #Entity du premier objet dans la collision en cours d'examen avec ExamineWorldCollisions() .

Arguments

Aucun.

Valeur de retour

Renvoie le numéro #Entity du premier objet dans la collision en cours.

Voir aussi

ExamineWorldCollisions() , SecondWorldCollisionEntity()

OS Supportés

Tous

92.12 SecondWorldCollisionEntity

Syntaxe

```
Resultat =
    SecondWorldCollisionEntity()
```

Description

Renvoie le numéro #Entity du second objet dans la collision en cours d'examen avec ExamineWorldCollisions() ().

Arguments

Aucun.

Valeur de retour

Renvoie le numéro #Entity du second objet dans la collision en cours.

Voir aussi

ExamineWorldCollisions() , FirstWorldCollisionEntity()

OS Supportés

Tous

92.13 WorldCollisionContact

Syntaxe

```
WorldCollisionContact()
```

Description

Récupère les informations de contact lors de la collision en cours d'examen par ExamineWorldCollisions() ().

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Le paramètre 'Contacts' de la fonction ExamineWorldCollisions() () doit être mis à #True.
Les valeurs du vecteur de contact peuvent être récupérées avec GetX() , GetY() et GetZ() .

Voir aussi

ExamineWorldCollisions() , GetX() , GetY() , GetZ()

OS Supportés

Tous

92.14 WorldCollisionNormal

Syntaxe

```
WorldCollisionNormal()
```

Description

Récupère les informations de la 'normale' de contact lors de la collision en cours d'examen par ExamineWorldCollisions() ().

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Le paramètre 'Contacts' de la fonction ExamineWorldCollisions() () doit être mis à #True.
Les valeurs du vecteur de contact peuvent être récupérées avec GetX() , GetY() et GetZ() .

Voir aussi

ExamineWorldCollisions() , GetX() , GetY() , GetZ()

OS Supportés

Tous

92.15 WorldCollisionAppliedImpulse

Syntaxe

```
Resultat.f =
    WorldCollisionAppliedImpulse()
```

Description

Renvoie l'impulsion appliquée lors de la collision en cours d'examen par ExamineWorldCollisions() .

Arguments

Aucun.

Valeur de retour

L'impulsion appliquée lors de la collision en cours d'examen par ExamineWorldCollisions() .

Remarques

Le paramètre 'Contacts' de la fonction ExamineWorldCollisions() doit être mis à #True.

Voir aussi

ExamineWorldCollisions()

OS Supportés

Tous

92.16 FetchOrientation

Syntaxe

```
FetchOrientation(ObjetID [,
    Mode])
```

Description

Renvoie le quaternion représentant l'orientation d'un objet.

Arguments

ObjetID L'ID (Identifiant) de l'objet concerné. Il peut être un des types suivants :

```
  - Camera          :
    Utiliser CameraID()
pour obtenir un ID valide.
  - Entité          :
    Utiliser EntityID()
pour obtenir un ID valide.
  - Lumière         :
    Utiliser LightID()
pour obtenir un ID valide.
  - Mesh            :
    Utiliser MeshID()
pour obtenir un ID valide.
  - Noeud           :
    Utiliser NodeID()
pour obtenir un ID valide.
  - Texte3D         :
    Utiliser Text3DID()
```

```
pour obtenir un ID valide.
 - Emetteur de Particle :
  Utiliser
  ParticleEmitterID()
pour obtenir un ID valide.
 - Groupe de billboards :
  Utiliser
  BillboardGroupID()
pour obtenir un ID valide.
```

Mode (optionnel) Le mode d'obtention du vecteur de direction de la caméra. Peut être l'une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 direction de la caméra
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 direction de la caméra
 par rapport à son parent.
```

Valeur de retour

Aucune.

Remarques

Plutôt que de réaliser une série de rotation autour de l'axe X puis Y puis Z, on définit un axe de rotation unique et un angle de rotation unique lui aussi, qui est la résultante de ces trois rotations. Le tout en une seule opération. Ce qui évite le "blocage de cardan". Les coordonnées de cet axe unique est le triplet (X, Y, Z) et l'angle unique de rotation est 'W'.
Les valeurs du quaternion peuvent être obtenues avec GetX() , GetY() , GetZ() et GetW() .

Voir aussi

GetX() , GetY() , GetZ() , GetW() , SetOrientation()

OS Supportés

Tous

92.17 SetOrientation

Syntaxe

```
SetOrientation(ObjetID, X, Y,
   Z, W)
```

Description

Change le quaternion représentant l'orientation d'un objet.

Arguments

ObjetID L'ID (Identifiant) de l'objet concerné. Il peut être un des types suivants :

```
  - Camera          :
    Utiliser CameraID()
  pour obtenir un ID valide.
  - Entité          :
    Utiliser EntityID()
  pour obtenir un ID valide.
  - Lumière         :
    Utiliser LightID()
  pour obtenir un ID valide.
  - Mesh            :
    Utiliser MeshID()
  pour obtenir un ID valide.
  - Noeud           :
    Utiliser NodeID()
  pour obtenir un ID valide.
  - Texte3D         :
    Utiliser Text3DID()
  pour obtenir un ID valide.
  - Emetteur de Particle :
    Utiliser
    ParticleEmitterID()
  pour obtenir un ID valide.
  - Groupe de billboards :
    Utiliser
    BillboardGroupID()
  pour obtenir un ID valide.
```

X, Y, Z, W Les nouvelles valeurs du quaternion représentant l'orientation de l'objet.
'X, Y, Z' sont les coordonnées de l'axe de rotation et 'w' l'angle de rotation à appliquer.

Valeur de retour

Aucune.

Remarques

Plutôt que de réaliser une série de rotation autour de l'axe X puis Y puis Z, on définit un axe de rotation unique et un angle de rotation unique lui aussi, qui est la résultante de ces trois rotations. Le tout en une seule opération. Ce qui évite le "blocage de cardan". Les coordonnées de cet axe unique est le triplet (X, Y, Z) et l'angle unique de rotation est 'W'.

Les valeurs du quaternion peuvent être obtenues avec GetX() , GetY() , GetZ() et GetW() .

Voir aussi

GetX() , GetY() , GetZ() , GetW() , FetchOrientation()

OS Supportés

Tous

92.18 GetX

Syntaxe

```
Resultat = GetX()
```

Description

Renvoie la valeur X de la dernière commande appelée.

Arguments

Aucun.

Valeur de retour

Renvoie la valeur X de la dernière commande appelée.

Remarques

Les fonctions prises en charge sont FetchOrientation() , ConvertLocalToWorldPosition() et ConvertWorldToLocalPosition() .

Voir aussi

GetY() , GetZ() , GetW() , FetchOrientation() , ConvertLocalToWorldPosition() , ConvertWorldToLocalPosition()

OS Supportés

Tous

92.19 GetY

Syntaxe

```
Resultat = GetY()
```

Description

Renvoie la valeur Y de la dernière commande appelée.

Arguments

Aucun.

Valeur de retour

Renvoie la valeur Y de la dernière commande appelée.

Remarques

Les fonctions prises en charge sont FetchOrientation() , ConvertLocalToWorldPosition() et ConvertWorldToLocalPosition() .

Voir aussi

GetX() , GetZ() , GetW() , FetchOrientation() , ConvertLocalToWorldPosition() , ConvertWorldToLocalPosition()

OS Supportés

Tous

92.20 GetZ

Syntaxe

```
Resultat = GetZ()
```

Description

Renvoie la valeur Z de la dernière commande appelée.

Arguments

Aucun.

Valeur de retour

Renvoie la valeur Z de la dernière commande appelée.

Remarques

Les fonctions prises en charge sont FetchOrientation() , ConvertLocalToWorldPosition() et ConvertWorldToLocalPosition() .

Voir aussi

GetX() , GetY() , GetW() , FetchOrientation() , ConvertLocalToWorldPosition() , ConvertWorldToLocalPosition()

OS Supportés

Tous

92.21 GetW

Syntaxe

```
Resultat = GetW()
```

Description

Renvoie la valeur W de la dernière commande appelée.

Arguments

Aucun.

Valeur de retour

Renvoie la valeur W de la dernière commande appelée.

Remarques

La seule fonction prise en charge est FetchOrientation() .

Voir aussi

GetX() , GetY() , GetZ() , FetchOrientation()

OS Supportés

Tous

92.22 Fog

Syntaxe

```
Fog(Couleur, Intensite,
    DistanceDebut, DistanceFin)
```

Description

Affiche un effet de brouillard à la distance spécifiée et sur toutes les caméras.

Arguments

Couleur Couleur du brouillard.
RGB() peut être utilisée pour obtenir facilement la couleur désirée.

Intensite Epaisseur du brouillard.
Avec une valeur de 0, l'effet de brouillard est supprimé.

DistanceDebut Distance à laquelle le brouillard commence à apparaître.

DistanceFin Distance à laquelle le brouillard devient complètement opaque.

Valeur de retour

Aucune.

Remarques

L'effet de brouillard est aussi appliqué sur les SkyBox() et les SkyDome() si ces fonctions sont appellées avant la fonction Fog()

Voir aussi

AmbientColor() , SkyDome() , SkyBox()

OS Supportés

Tous

92.23 InitEngine3D

Syntaxe

```
Resultat =
    InitEngine3D([Options [,
    Bibliotheque$])
```

Description

Initialise l'environnement nécessaire au fonctionnement du moteur 3D.
Vous devez placer cette fonction au début de votre code source si vous souhaitez utiliser les fonctions 3D.

Arguments

Options (optionnel) Peut être une combinaison des constantes suivantes :

```
#PB_Engine3D_NoLog      :
 Pas de log sauvegardé
 sur disque ni affiché
 dans une console (par
 défaut).
```

```
#PB_Engine3D_DebugLog   :
 Un fichier journal de
 débogage nommé
 'Ogre.log' sera créé
 dans le répertoire
 courant.
                              pour
 aider au déboguage. Un
 grand nombre d'actions
 sont consignées dans ce
 fichier,
                              mais
 il ne devrait pas
 affecter les
 performances de sorte
 qu'il peut même être
 activé
                              dans
 un exécutable
 distribuable...
#PB_Engine3D_DebugOutput:
 Les actions sont
 affichées sur la
 console. Activez au
 préalable l'option
 Console du compilateur
                              dans
 menu Compilateur \
 Options du
 compilateur... \ Options
 de compilation \ Format
 de l'exécutable
```

Bibliotheque$ (optionnel) Le nom d'un fichier Engine3D à charger.
S'il se trouve dans autre endroit que le dossier courant, il pourra être spécifié ici.

Valeur de retour

Renvoie une valeur non nulle si la bibliothèque a été chargée avec succès, zéro sinon.
Si l'initialisation a échoué, le programme ne devrait pas continuer ou alors tous les appels aux fonctions 3D devraient être désactivés.

Remarques

Cette fonction tente de charger la bibliothèque de moteur 3d (nommé 'Engine3D.dll' sous Windows, 'engine3d.so' sur Linux et 'engine3d.dylib' sur Mac OS X, depuis le dossier PureBasic/compilateurs/). Si elle échoue, c'est probablement parce que la bibliothèque est introuvable. Sous Windows, une version récente du pilote graphique OpenGL doit être installée.

Voir aussi

OpenScreen() , OpenWindowedScreen() , Add3DArchive()

OS Supportés

Tous

92.24 LoadWorld

Syntaxe

```
Résultat =
    LoadWorld(NomFichier$)
```

Description

Charge un monde complet.

Arguments

NomFichier$ Le nom du fichier contenant le monde complet doit être accessible dans le chemin géré par Add3DArchive() .

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Actuellement, seulement le format BSP de iD Software (Quake3) est supporté mais d'autres vont suivre.
De tels mondes peuvent être facilement créés grâce à des outils comme 'Quark'. Un monde contient un ciel, des batiments, des lumières etc... Les mondes de Quake3 peuvent être chargés immediatement, sans aucune conversion.
Important : Le format BSP est la propriété intellectuelle d'iD Software et ne peut être utilisé librement que dans des logiciels gratuits. Les logiciels à but commerciaux doivent acquérir une licence d'exploitation auprès d'iD Software. Ce système de licence n'est pas lié à PureBasic et Fantaisie Software ne peut en aucun cas être tenu pour responsable d'un usage incorrect de cette commande.

Voir aussi

Add3DArchive()

OS Supportés

Tous

92.25 MousePick

Syntaxe

```
Resultat = MousePick(#Camera,
   X, Y [, MasqueSelection])
```

Description

Simule un clic de la souris et renvoie l'objet se trouvant au point 2D.

Arguments

#Camera La caméra à utiliser.

X, Y Les coordonnées du point, en pixels.

MasqueSelection (optionnel) Masque de sélection utilisé pour une reconnaissance d'entités (ray cast). Seules les entités avec un masque correspondant à la valeur 'MasqueSelection' seront reconnues mais si ce paramètre est omis alors toutes les entités seront validées. Il est possible de sélectionner plus d'un groupe d'entités en utilisant une combinaison de masques. Pour plus d'informations sur 'MasqueSelection', voir CreateEntity() .

Valeur de retour

La valeur renvoyée peut être :

```
-1        : Aucun objet
  détecté.
0 et plus: Le clic s'est
  fait sur une entité. La
  sélection de l'entité est
  basée sur sa boîte
  englobante et non sur son
  mesh.
#PB_World_WaterPick  : Le
  clic s'est fait sur l'eau.
#PB_World_TerrainPick: Le
  clic s'est fait sur un
  terrain.
```

Pour obtenir les coordonnées 3D de l'objet sélectionné, utiliser PickX() , PickY() et PickZ() .

Voir aussi

PickX() , PickY() , PickZ()

OS Supportés

Tous

92.26 PointPick

Syntaxe

```
Resultat = PointPick(#Camera,
    X, Y)
```

Description

Permet d'obtenir la direction entre un point 2D et une caméra.

Arguments

#Camera La caméra à utiliser.

X, Y Les coordonnées du point, en pixels.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Pour obtenir la direction du point, voir PickX() , PickY() et PickZ() .

Valeur de retour

Aucune.

Voir aussi

PickX() , PickY() , PickZ()

OS Supportés

Tous

92.27 BodyPick

Syntaxe

```
Resultat = BodyPick(#Camera,
    Corps, X, Y, Bloqué)
```

Description

Simule un clic de souris et commence la manipulation de l'entité à la coordonnée spécifiée.

Arguments

#**Camera** La caméra à utiliser.

Corps `#True : Le corps (Body) est sélectionné #False: Le corps est libéré.`

Il ne peut y avoir qu'un Body choisi.

X, Y Les coordonnées du point, en pixels.

Bloqué (optionnel) `#True : Le corps choisi ne sera pas en mesure de tourner librement lors de son déplacement. #False: Le corps va tourner librement en fonction du mouvement du corps.`

Valeur de retour

Le #Entity choisi, ou -1 si aucune entité se trouvent à ces coordonnées.

OS Supportés

Tous

92.28 PickX

Syntaxe

```
Resultat.f = PickX()
```

Description

Renvoie une position ou une direction, suivant l'axe X.

Arguments

Aucun.

Valeur de retour

Après MousePick() , cette commande renvoie la position en 'X' de l'objet sélectionné, dans le monde 3D.
Après PointPick() , cette commande renvoie la direction en 'X' du point sous le clic de la souris. Sa valeur se situe entre -1 et 1.

Voir aussi

PickY() , PickZ()

OS Supportés

Tous

92.29 PickY

Syntaxe

```
Resultat.f = PickY()
```

Description

Renvoie une position ou une direction, suivant l'axe Y.

Arguments

Aucun.

Valeur de retour

Après MousePick() , cette commande renvoie la position en 'Y' de l'objet sélectionné, dans le monde 3D.
Après PointPick() , cette commande renvoie la direction en 'Y' du point sous le clic de la souris. Sa valeur se situe entre -1 et 1.

Voir aussi

PickX() , PickZ()

OS Supportés

Tous

92.30 PickZ

Syntaxe

```
Resultat.f = PickZ()
```

Description

Renvoie une position ou une direction, suivant l'axe Z.

Arguments

Aucun.

Valeur de retour

Après MousePick() , cette commande renvoie la position en 'Z' de l'objet sélectionné, dans le monde 3D.
Après PointPick() , cette commande renvoie la direction en 'Z' du point sous le clic de la souris. Sa valeur se situe entre -1 et 1.

Voir aussi

PickX() , PickY()

OS Supportés

Tous

92.31 RayCollide

Syntaxe

```
Resultat = RayCollide(X, Y,
    Z, DestinationX,
    DestinationY,
    DestinationZ,
    [CollisionGroupe ,
    CollisionMasque]])
```

Description

Lance un rayon entre deux points et vérifie si le rayon rencontre une entité avec un corps (body).

Arguments

X, Y, Z Coordonnée du point de départ du rayon, dans l'unité de monde.

DestinationX, DestinationY, DestinationZ Coordonnée du point d'arrivée du rayon, dans l'unité de monde.

CollisionGroupe, CollisionMasque (optionnel) Le groupe de collision et le masque de collision à utiliser.
Peut être utile pour filtrer l'entité qui doit entrer en collision avec le rayon.
Le groupe et le masque de collision peuvent être modifiés avec SetEntityCollisionFilter() .

Valeur de retour

Renvoie une valeur le numéro de l'entité rencontrée par le rayon ou -1 sinon.

Remarques

Pour avoir un quelconque effet, le moteur physique doit être activé avec la fonction EnableWorldPhysics() .
Seules les entités avec un corps (body) réagissent au contact du rayon.
Pour obtenir la position du point de collision, utiliser PickX() , PickY() et PickZ() .
Les valeurs normales au point de collision sont disponibles avec NormalX() , NormalY() et NormalZ() .

Voir aussi

NormalX() , NormalY() , NormalZ() , SetEntityCollisionFilter()

OS Supportés

Tous

92.32 RayCast

Syntaxe

```
Resultat = RayCast(X, Y, Z,
    DestinationX,
    DestinationY,
    DestinationZ,
    MasqueSelection)
```

Description

Projette un rayon entre le premier point et le second point, et vérifie si un objet traverse le rayon.

Arguments

X, Y, Z Coordonnées, dans l'unité de monde, du premier point.

DestinationX, DestinationY, DestinationZ Coordonnées, dans l'unité de monde, du second point.

MasqueSelection Le masque entité à utiliser.
Seules les entités avec un masque correspondant à la valeur 'MasqueSelection' seront signalées. Si ce paramètre est omis, toutes les entités sont des valeurs admissibles pour la détection des rayons. Le masque peut être une combinaison, pour sélectionner plus d'un groupe d'entités.
Pour avoir plus d'informations sur le masque de sélection, voir CreateEntity() .

Valeur de retour

Renvoie -1 si le rayon est entré en collision avec un objet.

Remarques

Cette fonction ne repose pas sur le moteur physique.
Les valeur des normales au point d'impact sont disponibles avec NormalX() , NormalY() et NormalZ() .

Voir aussi

NormalX() , NormalY() , NormalZ()

OS Supportés

Tous

92.33 MouseRayCast

Syntaxe

```
Resultat =
    MouseRayCast(#Camera, X,
    Y, MasqueSelection)
```

Description

Jette un rayon depuis un point 2D à travers la scène, et vérifie si un objet traverse le rayon.

Arguments

#Camera La caméra à utiliser.

X, Y Coordonnées 2D, en pixels, du point de départ.

MasqueSelection Le masque entité à utiliser.
Seules les entités avec un masque correspondant à la valeur 'MasqueSelection' seront signalées. Si ce paramètre est omis, toutes les entités sont des valeurs admissibles pour la détection des rayons. Le masque peut être une combinaison, pour sélectionner plus d'un groupe d'entités.
Pour avoir plus d'informations sur le masque de sélection, voir CreateEntity() .

Valeur de retour

Renvoie une valeur non nulle si le rayon est entré en collision avec un objet.

Voir aussi

NormalX() , NormalY() , NormalZ()

OS Supportés

Tous

92.34 NormalX

Syntaxe

```
Resultat.f = NormalX()
```

Description

Renvoie la valeur 'X' de la normale au point de rencontre, après RayCast() , RayCollide() ou MouseRayCast() .

Arguments

Aucun.

Valeur de retour

Renvoie la valeur 'X' de la normale au point de rencontre.

Voir aussi

RayCast() , RayCollide() , MouseRayCast() , NormalY() , NormalZ()

OS Supportés

Tous

92.35 NormalY

Syntaxe

```
Resultat.f = NormalY()
```

Description

Renvoie la valeur 'Y' de la normale au point de rencontre, après RayCast() , RayCollide() ou MouseRayCast() .

Arguments

Aucun.

Valeur de retour

Renvoie la valeur 'Y' de la normale au point de rencontre.

Voir aussi

RayCast() , RayCollide() , MouseRayCast() , NormalX() , NormalZ()

OS Supportés

Tous

92.36 NormalZ

Syntaxe

```
Resultat.f = NormalZ()
```

Description

Renvoie la valeur 'Z' de la normale au point de rencontre, après RayCast() , RayCollide() ou MouseRayCast() .

Arguments

Aucun.

Valeur de retour

Renvoie la valeur 'Z' de la normale au point de rencontre.

Voir aussi

RayCast() , RayCollide() , MouseRayCast() , NormalX() , NormalY()

OS Supportés

Tous

92.37 RayPick

Syntaxe

```
Resultat = RayPick(X, Y, Z,
    DestinationX,
    DestinationY, DestinationZ
    [, MasqueSelection])
```

Description

Lance un rayon entre deux points et vérifie si un objet traverse la trajectoire du rayon.

Arguments

X, Y, Z Coordonnée du point de départ du rayon, dans l'unité de monde.

DestinationX, DestinationY, DestinationZ Coordonnée du point d'arrivée du rayon, dans l'unité de monde.

MasqueSelection (optionnel) Est un masque de sélection d'entités utilisé pendant une sélection d'objets. Seules les entités avec un masque correspondant à la valeur 'MasqueSelection' seront sélectionnées

mais si ce paramètre est omis alors toutes les entités seront sélectionnées.
Il est possible de sélectionner plus d'un groupe d'entités.
Pour plus d'informations sur 'MasqueSelection', voir CreateEntity() .

Valeur de retour

La valeur renvoyée peut être :

```
-1               : Rien n'a été franchi.
0 et au-dessus   : Le rayon traverse une entité.
                   La détection d'entité est basée sur sa boîte englobante et pas sur son mesh.
#PB_World_WaterPick  : Le rayon traverse l'eau.
#PB_World_TerrainPick: Le rayon traverse un terrain.
```

Pour obtenir plus d'informations sur la position de l'objet sélectionné, voir PickX() , PickY() and PickZ() .

Voir aussi

RayCollide()

OS Supportés

Tous

92.38 ShowGUI

Syntaxe

```
ShowGUI(Transparence,
    AfficherCurseur [,
    #Camera, Activer])
```

Description

Affiche ou cache l'ensemble des éléments de l'interface graphique (GUI), qui est composée de fenêtres 3d et de gadgets 3d .

Arguments

Transparence Transparence des éléments de l'interface graphique (GUI).
Peut prendre une valeur allant de 0 (invisible) à 255 (Opaque).
Les valeurs intermédiaires donneront un effet de fondu.

AfficherCurseur 0: Curseur de la souris masqué. 1: Curseur de la souris affiché.

#Camera (optionnel) Indique si la GUI doit être affichée ou non sur cette caméra. Fonctionne conjointement avec le paramètre 'Activer'.

Activer (optionnel) Active ou désactive l'affichage de la GUI sur la caméra indiquée.
Fonctionne conjointement avec le paramètre '#Camera'.

Valeur de retour

Aucune.

Voir aussi

SetGUITheme3D()

OS Supportés

Tous

92.39 SetGUITheme3D

Syntaxe

```
SetGUITheme3D(NomTheme$,
    NomPolice$)
```

Description

Cette commande permet de sélectionner le thème et la police utilisés pour personnaliser l'interface graphique 3D CEGUI.

Arguments

NomTheme$ Le nom du thème à appliquer, sans l'extension '.scheme'

NomPolice$ Le nom de la police à appliquer, sans l'extension '.font'.

Valeur de retour

Aucune.

Remarques

Pour plus d'informations sur les 'skins', visitez le site web CEGUI.

Voir aussi

ShowGUI()

OS Supportés

Tous

92.40 Parse3DScripts

Syntaxe

```
Parse3DScripts()
```

Description

Lit tous les fichiers scripts '.materials' d'OGRE trouvés dans les chemins définis avec Add3DArchive() .

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Cela permet d'utiliser les meshs qui ont des scripts complexes de matières directement en PureBasic.
Quand une entité utilisant ce type de mesh sera créée, la constante #PB_Material_None devra être spécifiée pour indiquer que les informations concernant les matières doivent être lues à partir du script (s'il a été correctement chargé).
Vous trouverez plus d'informations au sujet des scripts dans le manuel en ligne d'OGRE.

Voir aussi

SetGUITheme3D()

OS Supportés

Tous

92.41 RenderWorld

Syntaxe

```
Resultat =
    RenderWorld([TempsPhysiqueEcoulé])
```

Description

Affiche le rendu de la scène 3D sur l'écran courant.

Arguments

TempsPhysiqueEcoulé (optionnel)
Force le moteur physique à utiliser cette valeur, en millisecondes, qui est le temps écoulé depuis le dernier appel de RenderWorld.
Utile pour simuler un ralenti ou une accélération du temps réel.

Valeur de retour

Renvoie le temps écoulé depuis la dernière image en millisecondes.
Il peut être utile d'avoir une simulation basée sur un temps précis, si le temps de rendu d'image n'est pas stable.

Remarques

Cette commande doit être appelée quand toutes les opérations relatives à la 3D sont terminées et une seule fois par image (frame).
Il est parfaitement possible d'utiliser les fonctions 2D habituelles telles que DisplaySprite() après cette commande, pour afficher des informations en sur-impression.

Valeur de retour

Aucune.

Voir aussi

InitEngine3D() , InputEvent3D()

OS Supportés

Tous

92.42 SetRenderQueue

Syntaxe

```
SetRenderQueue(ObjetID, Queue
    [, Priorite])
```

Description

Modifie l'ordre de rendu de l'objet.

Arguments

ObjetID L'ID de l'objet. Il peut être l'un des types suivants :

```
  - Entité                  :
    Utiliser EntityID()
pour obtenir un ID valide.
  - Lumière                 :
    Utiliser LightID()
pour obtenir un ID valide.
  - Mesh                    :
    Utiliser MeshID()
pour obtenir un ID valide.
  - Emetteur de Particules  :
    Utiliser
    ParticleEmitterID()
pour obtenir un ID valide.
  - Groupe de Billboards    :
    Utiliser
    BillboardGroupID()
pour obtenir un ID valide.
  - Texte3D                 :
    Utiliser Text3DID()
pour obtenir un ID valide.
```

Queue Le nombre de files d'attente à utiliser pour rendre l'objet.
Le nombre de files d'attente peut aller de 0 (l'arrière plan) à 100 (le premier plan).
La file d'attente par défaut est 0, et toutes les files d'attente sont rendues.

Priorite (optionnel) La priorité à utiliser au sein de la file d'attente.
Les valeurs valides sont comprises entre 0 (arrière plan) et 10 000 (premier plan).

Valeur de retour

Aucune.

OS Supportés

Tous

92.43 SkyBox

Syntaxe

```
Resultat = SkyBox(NomTexture$
    [, CouleurBrouillard,
    EpaisseurBrouillard,
    DebutDistanceBrouillard,
    FinDistanceBrouillard])
```

Description

Crée une boîte artificielle (cube à 6 faces texturées) située très loin de la caméra pour fermer complètement le monde.

Arguments

NomTexture$ Le nom de chaque texture doit être nommé selon la règle suivante :

```
NomTexture_BK : Face
 arrière (BacK)
NomTexture_FR : Face avant
 (FRont)
NomTexture_DN : Face du
 bas (DowN)
NomTexture_UP : Face du
 haut (UP)
NomTexture_LF : Face de
 gauche (LeFt)
NomTexture_RT : Face de
 droite (RighT)
```

CouleurBrouillard (optionnel) La couleur du brouillard.
RGB() peut être utilisé pour obtenir une valeur valide.
Si non spécifié, les paramètres du brouillard seront ceux provenant de la fonction Fog() .

EpaisseurBrouillard (optionnel) L'épaisseur du brouillard.
Si ce paramètre est égal à zéro alors le brouillard est désactivé.

DebutDistanceBrouillard (optionnel) La distance en unité Monde à laquelle le brouillard commence (par rapport à la caméra).

FinDistanceBrouillard (optionnel) La distance en unité Monde à laquelle le brouillard est totalement opaque (par rapport à la caméra).

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Les textures doivent être dans un des chemins précédemment déclarés avec la fonction Add3DArchive() .
Si les textures ne peuvent être chargées, le skybox est créé avec des textures blanches.

Exemple

```
1 ...
2 ; Les 6 textures (images)
   sont réunies dans le
   fichier zip "skybox.zip"
```

```
3   Add3DArchive(#PB_Compiler_Home
     +
     "examples/3d/Data/Packs/skybox.zip",
     #PB_3DArchive_Zip)
4   Parse3DScripts()
5   ...
6   ; Création de la skybox
     avec les 6 images qui ont
     pour nom:
     stevecube_BK.jpg,
     stevecube_DN.jpg, etc.)
7   SkyBox("stevecube.jpg")
```

Voir aussi

SkyDome()

OS Supportés

Tous

92.44 SkyDome

Syntaxe

```
Resultat =
    SkyDome(NomTexture$,
    Courbure.f [,
    CouleurBrouillard,
    EpaisseurBrouillard,
    DebutDistanceBrouillard,
    FinDistanceBrouillard])
```

Description

Crée un ciel artificiel animé en forme de dôme, situé très loin de la caméra.

Arguments

NomTexture$ La texture à utiliser.
La texture doit être accessible dans l'un des chemins déclaré par Add3DArchive() .

Courbure.f Cette valeur indique de quelle manière le ciel doit être courbé.
(valeur positive ou négative).

CouleurBrouillard (optionnel) La couleur du brouillard.
RGB() peut être utilisé pour obtenir une valeur valide.
Si non spécifié, les paramètres du brouillard seront ceux provenant de la fonction Fog() .

EpaisseurBrouillard (optionnel)
L'épaisseur du brouillard.
Si ce paramètre est égal à zéro alors le brouillard est désactivé.

DebutDistanceBrouillard (optionnel) La distance en unité Monde à laquelle le brouillard commence (par rapport à la caméra).

FinDistanceBrouillard (optionnel) La distance en unité Monde à laquelle le brouillard est totalement opaque (par rapport à la caméra).

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Les textures doivent être dans un des chemins précédemment déclarés avec la fonction Add3DArchive() .
Si les textures ne peuvent être chargées, le skybox est créé avec des textures blanches.

Exemple

```
...
; La texture (image) doit être accessible
Add3DArchive(#PB_Compiler_Home + "examples/3d/Data/Textures", #PB_3DArchive_FileSystem)
...
; Création du skydome
SkyDome("clouds.jpg", 30)
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Voir aussi

SkyBox()

OS Supportés

Tous

92.45 CreateWater

Syntaxe

```
CreateWater(#Camera, X, Y, Z, Transparence, Options)
```

Description

Crée un plan d'eau.

Arguments

X, Y, Z Position du plan d'eau.

Transparence Transparence du plan d'eau qui peut varier de 0 (complètement opaque) à 255 (transparent).

Options Peut être une combinaison des constantes suivantes :

```
#PB_World_WaterMediumQuality:
 Qualité moyenne (valeur
 par défaut).
#PB_World_WaterLowQuality
   : Basse qualité, ce
 qui implique moins de
 polygones (et donc un
 rendu plus rapide).
#PB_World_WaterHighQuality
  : Haute qualité, ce qui
 implique plus de
 polygones (et donc un
 rendu plus lent).
#PB_World_WaterCaustics
     : Active les effets
 caustiques sur l'eau,
 qui sont des formes
 créées par la lumière à
 la surface de l'eau.
#PB_World_WaterSmooth
       : Active la
 transition douce entre
 les vagues.
#PB_World_WaterFoam
         : Active l'effet
 d'écume, qui affecte les
 objets sous-marins
 lorsqu'ils sont vus de
 l'extérieur du plan
 d'eau.
#PB_World_WaterSun
          : Active la
 réflexion du soleil sur
 l'eau. Utilisez Sun()
pour modifier la position
 et la couleur du soleil.
#PB_World_UnderWater
        : Active un effet
 sous-marin opaque.
#PB_World_WaterGodRays
      : Active un effet
 rayons de soleil divin
 lorsqu'on est sous
 l'eau.
 #PB_World_WaterSun doit
 être spécifié.
```

Valeur de retour

Aucune.

Voir aussi

WaterColor()

OS Supportés

Tous

92.46 FreeWater

Syntaxe

```
FreeWater(#Camera)
```

Description

Supprime le plan d'eau courant d'une caméra.

Arguments

#Camera La caméra à utiliser.

Valeur de retour

Aucune.

Voir aussi

CreateWater()

OS Supportés

Tous

92.47 WaterColor

Syntaxe

```
WaterColor(#Camera, Couleur)
```

Description

Change la couleur de l'eau.

Arguments

#Camera La caméra à utiliser.

Couleur La nouvelle couleur de l'eau. Une valeur de couleur valide peut être obtenue avec RGB() .

Valeur de retour

Aucune.

Remarques

CreateWater() doit être appelée avant d'appeler cette fonction.

Voir aussi

CreateWater()

OS Supportés

Tous

92.48 WaterHeight

Syntaxe

```
WaterHeight(#Camera, X, Y)
```

Description

Renvoie la hauteur d'eau actuelle à la position spécifiée.

Arguments

#Camera La caméra à utiliser.
X, Y Les coordonnées.

Valeur de retour

La hauteur d'eau absolue, dans les unités du monde.

Remarques

CreateWater() doit être appelée avant d'appeler cette fonction.
Comme l'eau peut avoir des vagues, la hauteur varie.
Cette fonction peut être utile pour permettre à un objet de flotter sur l'eau.

Voir aussi

CreateWater()

OS Supportés

Tous

92.49 Sun

Syntaxe

```
Sun(X, Y, Z, Couleur)
```

Description

Change la position et la couleur du soleil.

Arguments

X, Y, Z La nouvelle position du soleil.

Couleur La nouvelle couleur du soleil. RGB() peut être utilisé pour obtenir une couleur valide.

Valeur de retour

Aucune.

Remarques

Attention : Ceci affecte l' eau créée avec l'option #PB_World_WaterSun.

Voir aussi

SkyBox() , SkyDome() , CreateWater()

OS Supportés

Tous

92.50 WorldShadows

Syntaxe

```
WorldShadows(Type [,
    Distance.f [, Couleur [,
    TailleTexture]]])
```

Description

Change le type d'ombre qui sera appliqué dans le monde 3D.

Arguments

Type Peut prendre l'une des valeurs suivantes :

```
#PB_Shadow_None      :
  Aucune ombre ne sera
  affichée. Cela peut
  permettre d'économiser
  beaucoup
                       de
  temps processeur si les
  ombres ne sont pas
  nécessaires (mode par
  defaut).
#PB_Shadow_Modulative: Les
  ombres seront affichées
  pour les entity
qui ont le
                       mode
  'ombre' activé avec
  EntityRenderMode()
```

```
et #PB_Entity_CastShadow.
                          Ce
  mode est le plus rapide
  pour afficher des
  ombres, mais il n'est
  pas très
                          réaliste
  car les ombres ainsi
  projetées ne sont pas
  translucides.
 #PB_Shadow_Additive  : Les
  ombres seront affichées
  pour les entity
qui ont le
                          mode
  'ombre' activé avec
  EntityRenderMode()
et #PB_Entity_CastShadow.
                          Ce
  mode est le plus lent
  mais aussi le plus
  réaliste, étant donné
  que les ombres
                          sont
  transparentes et
  s'ajoutent si plusieurs
  ombres se chevauchent.
 #PB_Shadow_TextureAdditive
   : Les ombres seront
  affichées pour les entity
qui ont le
                          mode
  'ombre' activé avec
  EntityRenderMode()
et #PB_Entity_CastShadow.
  Ce mode
                          est
  moyen en terme de
  rapidité et de qualité,
  mais fonctionnera même
  avec un
                          plan
  d'eau
.
 #PB_Shadow_TextureModulative:
  Les ombres seront
  affichées pour les entity
qui ont le
                          mode
  'ombre' activé avec
  EntityRenderMode()
et #PB_Entity_CastShadow.
                          Ce
  mode d'ombre est plus
  rapide que
  #PB_Shadow_TextureAdditive
  mais pas très beau car
                          les
  ombres ne sont pas
```

```
translucides.
```

Distance.f (optionnel) Distance maximale de la caméra au delà de laquelle les ombres ne seront plus calculées ni affichées.

Couleur (optionnel) Couleur de l'ombre. RGB() peut être utilisé pour obtenir une valeur de couleur valide.

TailleTexture (optionnel) Taille en pixel de la texture utilisée pour rendre l'ombre. Plus elle est grande et plus l'ombre sera réaliste, mais plus ce sera lent.
La valeur par défaut est 512 et cette valeur ne doit pas être supérieure à 4096.

Valeur de retour

Aucune.

Voir aussi

EnableWorldPhysics()

OS Supportés

Tous

92.51 WorldGravity

Syntaxe

```
WorldGravity(Gravitation.f [,
   x, y, z])
```

Description

Change la gravité du monde 3D.

Arguments

Gravitation.f Valeur de la nouvelle pesanteur.
La valeur par défaut est -9.806, ce qui correspond à la gravité terrestre.

x, y, z (optionnel) Les coordonnées du vecteur gravitation à utiliser pour changer la direction de la gravité.

Valeur de retour

Aucune.

Remarques

Le moteur physique doit être activé avec EnableWorldPhysics() .
Quelques exemples de pesanteur :

```
Attraction de la pesanteur
 en % de celle de la terre
LUNE   : 16%
MERCURE: 38%
VENUS  : 90%
MARS   : 38%
JUPITER: 153%
SATURNE: 107%
URANUS : 92%
NEPTUNE: 112%
PLUTON : 10% (approximatif)
```

Par exemple, le champ de pesanteur de Mercure vaut 38% de 9.806 soit 3.72 en valeur absolue.

Voir aussi

AmbientColor()

OS Supportés

Tous

92.52 WorldDebug

Syntaxe

```
WorldDebug(Mode)
```

Description

Change le mode débogage du monde 3D.

Arguments

Mode Peut être une combinaison de :

```
#PB_World_DebugNone   : Pas
 d'information de
 débogage (Valeur par
 défaut).
#PB_World_DebugEntity:
 Montre les boîtes
 englobantes des entités.
#PB_World_DebugBody   :
 Montre les boîtes des
 corps physiques, à la
 fois statiques et
 dynamiques.
```

Valeur de retour

Aucune.

Remarques

Cette fonction est utile pour par exemple, aider à résoudre un problème de collision ou de picking lié à une mauvaise définition des boîtes.

OS Supportés

Tous

92.53 Pitch

Syntaxe

```
Pitch(ObjetID, Valeur.f, Mode)
```

Description

Applique un tangage sur l'objet spécifié.

Arguments

ObjetID L'entité spécifiée. Peut être l'un des types suivants :

```
  - Camera                 :
    Utiliser CameraID()
  pour obtenir un identifiant
    (ID) valide.
  - Entité                 :
    Utiliser EntityID()
  pour obtenir un identifiant
    (ID) valide.
  - Lumière                :
    Utiliser LightID()
  pour obtenir un identifiant
    (ID) valide.
  - Mesh                   :
    Utiliser MeshID()
  pour obtenir un identifiant
    (ID) valide.
  - Noeud                  :
    Utiliser NodeID()
  pour obtenir un identifiant
    (ID) valide.
  - Emetteur de Particules :
    Utiliser
    ParticleEmitterID()
  pour obtenir un identifiant
    (ID) valide.
  - Groupe de Billboards   :
    Utiliser
    BillboardGroupID()
  pour obtenir un identifiant
    (ID) valide.
  - Texte 3D               :
    Utiliser Text3DID()
  pour obtenir un identifiant
    (ID) valide.
```

Valeur.f La valeur du tangage en degré.

Mode Le mode de tangage. Peut être l'une des valeurs suivantes :

```
#PB_Local : Local.
#PB_Parent: Par rapport au
 parent.
#PB_World : Par rapport au
 monde 3D.
```

Valeur de retour

Aucune.

Voir aussi

Roll() , Yaw()

OS Supportés

Tous

92.54 Roll

Syntaxe

```
Roll(ObjetID, Valeur.f, Mode)
```

Description

Applique un roulis sur l'objet spécifié.

Arguments

ObjetID L'entité spécifiée. Peut être l'un des types suivants :

```
  - Camera                  :
    Utiliser CameraID()
  pour obtenir un identifiant
    (ID) valide.
  - Entité                  :
    Utiliser EntityID()
  pour obtenir un identifiant
    (ID) valide.
  - Lumière                 :
    Utiliser LightID()
  pour obtenir un identifiant
    (ID) valide.
  - Mesh                    :
    Utiliser MeshID()
  pour obtenir un identifiant
    (ID) valide.
  - Noeud                   :
    Utiliser NodeID()
  pour obtenir un identifiant
    (ID) valide.
```

```
  - Emetteur de Particules :
    Utiliser
    ParticleEmitterID()
  pour obtenir un identifiant
    (ID) valide.
  - Groupe de Billboards   :
    Utiliser
    BillboardGroupID()
  pour obtenir un identifiant
    (ID) valide.
  - Texte 3D               :
    Utiliser Text3DID()
  pour obtenir un identifiant
    (ID) valide.
```

Valeur.f La valeur du roulis en degré.

Mode Le mode de roulis. Peut être l'une des valeurs suivantes :

```
#PB_Local : Local.
#PB_Parent: Par rapport au
  parent.
#PB_World : Par rapport au
  monde 3D.
```

Valeur de retour

Aucune.

Voir aussi

Pitch() , Yaw()

OS Supportés

Tous

92.55 Yaw

Syntaxe

```
Yaw(ObjetID, Valeur.f, Mode)
```

Description

Applique un lacet sur l'objet spécifié.

Arguments

ObjetID L'entité spécifiée. Peut être l'un des types suivants :

```
  - Camera                 :
    Utiliser CameraID()
  pour obtenir un identifiant
    (ID) valide.
  - Entité                 :
    Utiliser EntityID()
```

```
pour obtenir un identifiant
  (ID) valide.
 - Lumière                :
  Utiliser LightID()
pour obtenir un identifiant
  (ID) valide.
 - Mesh                   :
  Utiliser MeshID()
pour obtenir un identifiant
  (ID) valide.
 - Noeud                  :
  Utiliser NodeID()
pour obtenir un identifiant
  (ID) valide.
 - Emetteur de Particules :
  Utiliser
  ParticleEmitterID()
pour obtenir un identifiant
  (ID) valide.
 - Groupe de Billboards   :
  Utiliser
  BillboardGroupID()
pour obtenir un identifiant
  (ID) valide.
 - Texte 3D               :
  Utiliser Text3DID()
pour obtenir un identifiant
  (ID) valide.
```

Valeur.f La valeur du lacet en degré.

Mode Le mode de lacet. Peut être l'une des valeurs suivantes :

```
#PB_Local : Local.
#PB_Parent: Par rapport au
  parent.
#PB_World : Par rapport au
  monde 3D.
```

Valeur de retour

Aucune.

Voir aussi

Pitch() , Roll()

OS Supportés

Tous

Chapitre 93

Entity

Généralités

Une entité, ou 'Entity' en anglais est un objet 3D composé d'un mesh (un maillage 3D) et d'une matière (ou matériau). Une entité peut être déplacée et transformée en temps réel. De plus, la bibliothèque EntityAnimation est disponible pour animer une partie d'une entité constituée d'un squelette, ce qui permet de faire marcher une entité, par exemple.
Il est possible de partager un mesh et/ou une matière entre plusieurs entités réduisant ainsi la consommation mémoire et l'utilisation du processeur.
InitEngine3D() doit être appelé avec succès avant de pouvoir utiliser les commandes relatives aux entités.

OS Supportés

Tous

93.1 ApplyEntityForce

Syntaxe

```
ApplyEntityForce(#Entity, X,
    Y, Z [, PositionX,
    PositionY, PositionZ [,
    Mode [, ModePosition]]])
```

Description

Applique une force sur une entité.

Arguments

#Entity L'entité à utiliser.

X, Y, Z Valeur de la force en x, y et z.

PositionX, PositionY, PositionZ (optionnel)
La position relative par rapport au centre de l'entité, où la force doit être appliquée.

Mode (optionnel) Le mode à appliquer. Il peut s'agir de l'une des valeurs suivantes :

```
#PB_Local : Force locale.
#PB_Parent: Force relative
 à la position du parent.
#PB_World : Force relative
 au monde.
```

ModePosition (optionnel) Le mode position à appliquer. Il peut s'agir de l'une des valeurs suivantes :

```
#PB_Local : Position
 locale.
#PB_Parent: Position
 relative à la position
 du parent.
#PB_World : Position
 relative au monde.
```

Valeur de retour

Aucune.

Voir aussi

ApplyEntityImpulse() ,
ApplyEntityTorque() ,
ApplyEntityTorqueImpulse()

OS Supportés

Tous

93.2 ApplyEntityImpulse

Syntaxe

```
ApplyEntityImpulse(#Entity,
   X, Y, Z [, PositionX,
   PositionY, PositionZ [,
   Mode]])
```

Description

Applique une impulsion à une entité.

Arguments

#Entity L'entité à utiliser.

X, Y, Z Valeur de la force d'impulsion en en x, y et z.

PositionX, PositionY, PositionZ (optionnel) La position relative par rapport au centre de l'entité, où l'impulsion doit être appliquée.

Mode (optionnel) Le mode d'impulsion appliqué. Il peut s'agir de l'une des valeurs suivantes :

```
#PB_Local : Impulsion
 locale.
#PB_Parent: Impulsion
 relative à la position
 du parent.
#PB_World : Impulsion
 relative au monde.
```

Valeur de retour

Aucune.

Voir aussi

ApplyEntityForce() , ApplyEntityTorque() , ApplyEntityTorqueImpulse()

OS Supportés

Tous

93.3 ApplyEntityTorque

Syntaxe

```
ApplyEntityTorque(#Entity, X,
    Y, Z [, Mode])
```

Description

Applique un couple de force à l'entité.

Arguments

#**Entity** L'entité à utiliser.

X, Y, Z Les valeurs du couple en x, y et z.

Mode (optionnel) Le mode de couple de force de rotation appliqué. Il peut s'agir de l'une des valeurs suivantes :

```
#PB_Local : Rotation
 locale.
#PB_Parent: Rotation
 relative à la position
 du parent.
#PB_World : Rotation
 relative au monde.
```

Valeur de retour

Aucune.

Voir aussi

ApplyEntityImpulse() , ApplyEntityForce() , ApplyEntityTorqueImpulse()

OS Supportés

Tous

93.4 ApplyEntityTorqueImpulse

Syntaxe

```
ApplyEntityTorqueImpulse(#Entity,
    X, Y, Z [, Mode])
```

Description

Applique une impulsion de couple à l'entité.

Arguments

#Entity L'entité à utiliser.

X, Y, Z Les valeurs de l'impulsion de couple en x, y et z.

Mode (optionnel) Le mode d'impulsion de couple de force de rotation appliqué. Il peut s'agir de l'une des valeurs suivantes :

```
#PB_Local : Impulsion de
 rotation locale.
#PB_Parent: Impulsion de
 rotation relative à la
 position du parent.
#PB_World : Impulsion de
 rotation relative au
 monde.
```

Valeur de retour

Aucune.

Voir aussi

ApplyEntityImpulse() , ApplyEntityForce() , ApplyEntityTorque()

OS Supportés

Tous

93.5 CopyEntity

Syntaxe

```
Resultat =
    CopyEntity(#Entity,
    #NouvelleEntity)
```

Description

Crée une copie d'une entité.

Arguments

#Entity L'entité à copier.

#NouvelleEntity La copie.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.
Tous ses attributs sont dupliqués.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si la '#NouvelleEntity' était déjà créée, elle est automatiquement supprimée et remplacée par la nouvelle.

Voir aussi

CreateEntity() , FreeEntity()

OS Supportés

Tous

93.6 CreateEntity

Syntaxe

```
Resultat =
    CreateEntity(#Entity,
    MeshID, MatiereID, [X, Y,
    Z [, MasqueSelection [,
    MasqueVisibilite]])
```

Description

Crée une nouvelle entité utilisant un mesh et une matière.

Arguments

#Entity Numéro de l'entité.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

MeshID Le mesh à utiliser.
Pour obtenir un identifiant valide, utiliser MeshID() .
Les meshes dynamiques (créés avec l'option #PB_Mesh_Dynamic) ne sont pas autorisés.

MatiereID La matière à utiliser.
Pour obtenir un identifiant valide, utilisez MaterialID() .

X, Y, Z (optionnel) La position de la nouvelle entité dans le monde.

MasqueSelection (optionnel) Utilisé par RayPick() et MousePick() pour sélectionner le groupe d'entités qui sera traité.
Comme il s'agit d'un masque, chaque valeur doit être une puissance de deux et 31 masques différents sont disponibles.
Pour créer une valeur de masque facilement, l'opérateur '«' peut être utilisé :

```
- 1 << 1  : Première
  valeur de masque valide
- 1 << 2  : Deuxième
  valeur de masque valide
- 1 << 3  : Troisième
  valeur de masque valide
- ...
- 1 << 31 : Dernières
  valeur de masque valide
```

Pour faciliter l'utilisation, les constantes doivent être utilisées pour stocker la valeur du masque.
Lorsque vous appelez les fonctions de sélection, les masques peuvent être combinés avec l'opérateur '||' pour sélectionner plus d'un type d'entité.

MasqueVisibilite (optionnel) Un masque à choisir sur la caméra qui affiche l'entité.
Si ce masque correspond au masque spécifié par CreateCamera() alors l'entité sera affichée par la caméra.
Voir l'option 'MasqueSelection' ci-dessus pour construire des masques appropriés.
Si ce paramètre est omis alors l'entité sera visible par toutes les caméras.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si l'entité était déjà créée, elle est automatiquement supprimée et remplacée par la nouvelle.

Voir aussi

FreeEntity() , HideEntity() , IsEntity()

OS Supportés

Tous

93.7 EntityFixedYawAxis

Syntaxe

```
EntityFixedYawAxis(#Entity,
    Actif [, VecteurX.f,
    VecteurY.f, VecteurZ.f])
```

Description

Change l'axe fixe de lacet d'une entité.

Arguments

#**Entity** L'entité à utiliser.

Actif Active ou désactive l'utilisation d'un axe de lacet personnalisé.

```
#True : Un nouvel axe
 vecteur doit être
 spécifié.
#False: L'entité fera un
 lacet autour de son axe
 propre Y.
```

VecteurX, VecteurY, VecteurZ (optionnel) Direction du vecteur du nouvel axe de lacet.
Valeur comprise entre -1.0 et 1.0.
Le paramètre "Actif" doit être posé à #True pour avoir un effet.

Valeur de retour

Aucune.

Remarques

Le comportement par défaut d'une entité est un lacet autour de son propre axe Y.

Voir aussi

EntityYaw()

OS Supportés

Tous

93.8 EntityID

Syntaxe

```
Resultat = EntityID(#Entity)
```

Description

Renvoie l'identifiant unique d'une entité.

Arguments

#**Entity** L'entité à utiliser.

Valeur de retour

Renvoie le numéro de l'entité.

Remarques

Cette fonction est très utile quand une fonction d'une autre bibliothèque nécessite l'identifiant d'une entité.

Voir aussi

CreateEntity() , FreeEntity()

OS Supportés

Tous

93.9 EntityLookAt

Syntaxe

```
EntityLookAt(#Entity, X, Y, Z
    [, DirectionX.f,
    DirectionY.f,
    DirectionZ.f])
```

Description

Spécifie le point (dans l'unité du monde) auquel une entité fait face.

Arguments

#**Entity** L'entité à utiliser.

X, Y, Z L'entité pointe vers la position 'X, Y, Z' (dans l'unité du monde).

DirectionX, DirectionY, DirectionZ (optionnel)
La direction de l'entité.
Valeur comprise entre -1.0 et 1.0.

Valeur de retour

Aucune.

Remarques

La position de l'entité n'est pas modifiée.

OS Supportés

Tous

93.10 EntityVelocity

Syntaxe

```
EntityVelocity(#Entity, X, Y,
    Z)
```

Description

Modifie la vitesse linéaire d'une entité.

Arguments

#Entity L'entité à utiliser.

X, Y, Z Le vecteur vitesse.

Valeur de retour

Aucune.

Remarques

Le facteur linéaire est appliqué à l'entité.
Consultez EntityLinearFactor() pour plus d'informations.
L'entité a besoin d'un corps physique pour avoir une vitesse linéaire.
Pour obtenir la vitesse courante de l'entité, utilisez GetEntityAttribute() .

Voir aussi

EntityAngularFactor() , EntityCustomParameter() , EntityLinearFactor() , MoveEntity()

OS Supportés

Tous

93.11 EntityAngularFactor

Syntaxe

```
EntityAngularFactor(#Entity,
    X, Y, Z)
```

Description

Modifie le facteur angulaire d'une entité.

Arguments

#Entity L'entité à utiliser.

X, Y, Z Les facteurs angulaires du vecteur.

Valeur de retour

Aucune.

Voir aussi

EntityVelocity() ,
EntityCustomParameter() ,
EntityLinearFactor() , MoveEntity()

OS Supportés

Tous

93.12 EntityLinearFactor

Syntaxe

```
EntityLinearFactor(#Entity,
    X, Y, Z)
```

Description

Modifie le facteur linéaire d'une entité.

Arguments

#**Entity** L'entité à utiliser.

X, Y, Z Les valeurs du facteur linéaire. 0 indique que l'entité ne sera plus en mesure de se déplacer sur l'axe spécifié 'x, y ou z'.

Valeur de retour

Aucune.

Remarques

En cas de déplacement, la vitesse linéaire est multipliée par le facteur linéaire pour obtenir la vitesse finale.
Ceci est très utile pour contraindre le mouvement d'une entité sur un ou plusieurs axes.
Par défaut, le facteur linéaire est de 1 pour tous les axes ce qui signifie aucun impact sur la vitesse.
L'entité a besoin d'un corps physique pour utiliser cette contrainte.

Voir aussi

EntityVelocity() ,
EntityCustomParameter() ,
EntityAngularFactor() , MoveEntity()

OS Supportés

Tous

93.13 EntityCustomParameter

Syntaxe

```
EntityCustomParameter(#Entity,
    SousEntity,
    IndexParametre, Valeur1,
    Valeur2, Valeur3, Valeur4)
```

Description

Définit des paramètres personnalisés pour le 'script shader' matériau d'une entité.
Pour avoir un effet, le matériau associé à l'entité doit avoir un 'script shader', soit GLSL soit HLSL.

Arguments

#**Entity** L'entité à utiliser.

SousEntity La sous-entité à utiliser.
Le premier indice commence à 0 (reprensentant l'entité principale).

IndexParametre L'indice de paramètre dans le shader script.

Valeur1, Valeur2, Valeur3, Valeur4
Les valeurs du paramètre.
Si le paramètre ne peut accepter une valeur, cette valeur sera ignorée.

Valeur de retour

Aucune.

Voir aussi

EntityVelocity() , EntityLinearFactor() , EntityAngularFactor() , MoveEntity()

OS Supportés

Tous

93.14 EntityBoundingBox

Syntaxe

```
Resultat =
    EntityBoundingBox(#Entity,
    Options)
```

Description

Renvoie la position de la boîte englobante d'une entité en coordonnées locales ou mondiales.

Arguments

#**Entity** L'entité à utiliser.

Options

```
      #PB_Entity_MinBoundingBoxX:
    Min 'x' de la boîte
    englobante
  #PB_Entity_MaxBoundingBoxX:
    Max 'x' de la boîte
    englobante
  #PB_Entity_MinBoundingBoxY:
    Min 'y' de la boîte
    englobante
  #PB_Entity_MaxBoundingBoxY:
    Max 'y' de la boîte
    englobante
  #PB_Entity_MinBoundingBoxZ:
    Min 'z' de la boîte
    englobante
  #PB_Entity_MaxBoundingBoxZ:
    Max 'z' de la boîte
    englobante
```

Il est possible de la combiner avec l'une des valeurs suivantes :

```
  #PB_Entity_WorldBoundingBox:
    Coordonnées mondiales
    (par défaut)
  #PB_Entity_LocalBoundingBox:
    Coordonnées locales
```

Valeur de retour

Renvoie la position de la boîte englobante en coordonnées locales ou mondiales.

OS Supportés

Tous

93.15 DisableEntityBody

Syntaxe

```
DisableEntityBody(#Entity,
    Desactiver)
```

Description

Désactive le corps d'une entité.

Arguments

#**Entity** L'entité à désactiver.

Desactiver
```
        #True : Le corps de
    l'entité est désactivé.
    #False: Le corps de
    l'entité est activé.
```

Valeur de retour

Aucune.

Remarques

Le moteur physique n'affecte plus l'entité quand son corps est désactivé.

Voir aussi

CreateEntityBody() , CreateEntity()

OS Supportés

Tous

93.16 EntityParentNode

Syntaxe

```
Resultat =
    EntityParentNode(#Entity)
```

Description

Renvoie le numéro du noeud parent d'une entité.

Arguments

#**Entity** L'entité à utiliser.

Valeur de retour

Renvoie le NodeID() parent, s'il existe ou zéro sinon.

Remarques

Il peut s'agir d'un noeud réel ou d'un os si l'entité est fixée à un os.

Voir aussi

FreeEntityJoints()

OS Supportés

Tous

93.17 FetchEntityMaterial

Syntaxe

```
Resultat =
    FetchEntityMaterial(#Entity,
    #Matiere [, SousEntity])
```

Description

Installe la matière associée à l'#Entity avec SetEntityMaterial() .

Arguments

#Entity L'entité à utiliser.

#Material Le numéro de la nouvelle matière .

SousEntity (optionnel) La sous-entité à utiliser.
Le premier indice commence à 0 (ce qui représente la principale entité).

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Voir aussi

SetEntityMaterial() , GetEntityAttribute() , SetEntityAttribute()

OS Supportés

Tous

93.18 SetEntityMaterial

Syntaxe

```
SetEntityMaterial(#Entity,
    MatiereID [, SousEntity])
```

Description

Assigne une matière à l'entité.

Arguments

#**Entity** L'entité à utiliser.

MatiereID La matière.
Si une matière était déjà assignée à l'entité alors elle est remplacée.
Peut être facilement obtenu avec la commande MaterialID() .

SousEntity (optionnel) La sous-entité à utiliser.
Le premier indice commence à 0 (ce qui représente la principale entité).

Valeur de retour

Aucune.

Voir aussi

FetchEntityMaterial() , GetEntityAttribute() , SetEntityAttribute()

OS Supportés

Tous

93.19 EntityCollide

Syntaxe

```
Resultat =
    EntityCollide(#Entity,
    #Entity2)
```

Description

Vérifie si deux entités entrent en collision.

Arguments

#**Entity** La première entité à tester.

#**Entity2** La seconde entité à tester.

Valeur de retour

Renvoie une valeur non nulle si les deux entités entrent en collision.

Remarques

Pour que les collisions soient gérées par le moteur physique, une entité a besoin d'un corps (body) créé avec CreateEntityBody() . De plus, le moteur physique doit être activé avec EnableWorldPhysics() .

Voir aussi

DetachEntityObject() , CreateEntity() , DisableEntityBody()

OS Supportés

Tous

93.20 CreateEntityBody

Syntaxe

```
CreateEntityBody(#Entity,
    Type [, Masse [,
    Restitution, Friction [,
    TailleX, TailleY, TailleZ
    [, AxeX, AxeY, AxeZ]])
```

Description

Crée ou change le type de corps (body) associé à une entité.

Arguments

#**Entity** L'entité à utiliser.

Type Définit comment le moteur physique gère l'entité :

```
#PB_Entity_None       :
 Aucun body associé à
 l'entity (mode par
 défaut).
#PB_Entity_StaticBody : Le
 body est statique, c'est
 à dire que le mesh ne
 peut pas être animé.
                       Ce
 mode permet des
 collisions très
 précises, car elles sont
 faites par rapport aux
 triangles
                       qui
 composent le mesh (connu
 comme la tri-mesh
 collision). Les
 collisions entre body
 static et les autres
 types de body
                       (sphere,
 box etc.) sont très
 rapides, mais entre deux
 body static, cela
 devient très lent.
                       C'est
 le type de body parfait
 pour utiliser une mesh
```

```
 comme un sol ou pour un
 monde statique.
#PB_Entity_PlaneBody  : Un
 plan 'virtuel' sur
 l'entity (avec les mêmes
 dimensions)
                        est
 utilisé pour gérer les
 collisions entre les
 autres entités.
#PB_Entity_ConeBody   : Un
 cône 'virtuel' englobant
 totalement l'entity
 (avec les mêmes
 dimensions)
                        est
 utilisée pour gérer les
 collisions entre les
 autres entités.
#PB_Entity_BoxBody    :
 Une boîte 'virtuelle'
 englobant totalement
 l'entity (avec les mêmes
 dimensions)
                        est
 utilisée pour gérer les
 collisions entre les
 autres entités.
#PB_Entity_SphereBody :
 Une sphère 'virtuelle'
 englobant totalement
 l'entity
                        est
 utilisée pour gérer les
 collisions entre les
 autres entités.
#PB_Entity_CylinderBody :
 Un cylindre 'virtuel'
 englobant totalement
 l'entity
                        est
 utilisée pour gérer les
 collisions entre les
 autres entités.
#PB_Entity_CapsuleBody :
 Une capsule 'virtuelle'
 englobant totalement
 l'entity
                        est
 utilisée pour gérer les
 collisions entre les
 autres entités.
#PB_Entity_ConvexHullBody
 : Une forme complexe
 'virtuelle' déduite du
 mesh réel gère les
 collisions contre les
 autres
                        entités.
```

```
    Ce mode est plus lent
    que le mode basic de
    collision.
  #PB_Entity_CompoundBody
    : Un corps composé
    'virtuelle' englobant
    totalement l'entity
                          est
    utilisée pour gérer les
    collisions entre les
    autres entités.
```

Masse (optionnel) Masse de l'objet.
Ne pas utiliser une valeur trop grande car il pourrait se produire des incohérences physiques (Préférer la valeur 1).

Restitution (optionnel) Coefficient de Restitution de la vitesse de l'objet suite à une collision (Rebond).
En théorie, ce coefficient est égal à la vitesse restituée, divisée par la vitesse initiale. Il est aussi égal à la racine carrée de la hauteur du rebond, divisée par la hauteur du lâché.
Il est généralement compris entre 0 et 1 mais s'il est supérieur à 1 ou inférieur à 0, la collision produit de l'énergie cinétique et donc de la vitesse (vitesse restituée > vitesse initiale).
Cette valeur peut également être obtenue ou définie via GetEntityAttribute() et SetEntityAttribute()

Friction (optionnel) Force de friction ou de frottement de l'objet.
Attention ce paramètre est proportionnel à la Masse.
Cette valeur peut également être obtenue ou définie via GetEntityAttribute() et SetEntityAttribute()

TailleX, TailleY, TailleZ (optionnel) Les dimensions de la boîte englobante autour du corps. Ne concerne que le type de corps suivant :

```
  #PB_Entity_BoxBody      :
    TailleX, TailleY,
    TailleZ sont disponibles.
  #PB_Entity_SphereBody   :
    TailleX est disponible.
  #PB_Entity_ConeBody     :
    TailleX, TailleY sont
    disponibles.
  #PB_Entity_CylinderBody:
    TailleX, TailleY sont
    disponibles.
  #PB_Entity_CapsuleBody  :
    TailleX, TailleY,
    TailleZ sont disponibles.
```

AxeX, AxeY, AxeZ (optionnel) Les axes du corps. Ne concerne que le type de corps suivant :

```
#PB_Entity_PlaneBody
#PB_Entity_CylinderBody
#PB_Entity_CapsuleBody
```

Valeur de retour

Aucune.

Remarques

Pour que les collisions soient gérées par le moteur physique, une entité doit avoir un corps.
De plus, le moteur physique doit être activé à l'aide de la commande EnableWorldPhysics() .
En fait, seul le corps est connu du moteur physique lequel fera tous les calculs concernant l'entité, vérifiera sa masse, les forces de frottement (friction) et s'il entre en collision, cela entraînera un mouvement de recul de l'entité (Restitution).

Voir aussi

EntityAngularFactor() , EntityCollide() , EntityCustomParameter() , EntityLinearFactor() , GetEntityAttribute() , FetchEntityMaterial() , SetEntityAttribute() , SetEntityMaterial()

OS Supportés

Tous

93.21 EntityRenderMode

Syntaxe

```
EntityRenderMode(#Entity,
    Mode)
```

Description

Change le mode de rendu d'une entité.

Arguments

#**Entity** L'entité à utiliser.

Mode Peut être une combinaison de :

```
#PB_Entity_CastShadow
 : L'entité projette des
 ombres si WorldShadows()
```

```
est activé (mode par
  défaut).
 #PB_Entity_DisplaySkeleton
  : Affiche le squelette
  de l'entité
 #PB_Shadow_None
  : Désactive la
  projection des ombres
  (utile pour l'affichage
  des "sols")
```

Valeur de retour

Aucune.

Voir aussi

CreateEntity()

OS Supportés

Tous

93.22 AttachEntityObject

Syntaxe

```
AttachEntityObject(#Entity,
   Os$, ObjetID [, X, Y, Z,
   Tangage, Roulis, Lacet])
```

Description

Attache un objet existant à un os d'une entité.

Arguments

#**Entity** L'entité à utiliser.

Os$ Le nom de l'os dans le mesh OGRE. Si le nom de l'os est vide, l'objet n'est pas attaché à un os, mais directement à l'entité.

ObjetID Peut être un des objets suivants :

```
  - Entity : Utilisation de
    EntityID()
en tant que 'ObjetID'.
  - Camera : Utilisation de
    CameraID()
en tant que 'ObjetID'.
  - Light  : Utilisation de
    LightID()
en tant que 'ObjetID'.
  - BillboardGroup :
    Utilisation de
    BillboardGroupID()
```

```
en tant que 'ObjetID'.
 - ParticleEmitter:
  Utilisation de
  ParticleEmitterID()
en tant que 'ObjetID'.
```

X, Y, Z (optionnel) Déplacement relatif par rapport à la position de l'objet.

Tangage, Roulis, Lacet (optionnel) Correspondent à la rotation de l'objet par rapport aux 3 axes.

Valeur de retour

Aucune.

Remarques

Un objet peut être détaché de l'os d'une entité avec DetachEntityObject() .

Voir aussi

DetachEntityObject()

OS Supportés

Tous

93.23 DetachEntityObject

Syntaxe

```
DetachEntityObject(#Noeud,
   ObjetID)
```

Description

Détache un objet précédemment attaché à un os d'une entité.

Arguments

#Noeud Le noeud à utiliser.

ObjetID Les objets pris en charge sont :

```
   - Entity : Utilisation de
    EntityID()
  en tant que 'ObjetID'.
   - Camera : Utilisation de
    CameraID()
  en tant que 'ObjetID'.
   - Light  : Utilisation de
    LightID()
  en tant que 'ObjetID'.
   - BillboardGroup :
    Utilisation de
    BillboardGroupID()
```

```
en tant que 'ObjetID'.
 - ParticleEmitter:
  Utilisation de
  ParticleEmitterID()
en tant que 'ObjetID'.
```

Valeur de retour

Aucune.

Remarques

Un objet peut être attaché à un os avec AttachEntityObject() .

Voir aussi

AttachEntityObject()

OS Supportés

Tous

93.24 EnableManualEntityBoneControl

Syntaxe

```
EnableManualEntityBoneControl(#Entity,
   Os$, Etat,
   HeritageOrientation)
```

Description

Activer la commande manuelle d'un os.

Arguments

#**Entity** L'entité à utiliser.

Os$ Le nom de l'os dans le mesh OGRE.

Etat Peut être une des valeurs suivantes :

```
#True : La commande
 manuelle de l'os est
 activée.
#False: La commande
 manuelle de l'os est
 désactivée.
```

HeritageOrientation (optionnel)
Indique si l'orientation de l'os doit hériter de l'orientation de l'entité. Peut être une des valeurs suivantes :

```
#True : L'orientation
 manuelle de l'os est
 héritée de l'entité.
```

```
#False: L'orientation
 manuelle de l'os ne
 prend pas en compte
 l'orientation de
 l'entité.
```

Valeur de retour

Aucune.

Remarques

Il peut être déplacé manuellement avec MoveEntityBone() et tourné avec RotateEntityBone() .

OS Supportés

Tous

93.25 MoveEntityBone

Syntaxe

```
MoveEntityBone(#Entity, Os$,
   X, Y, Z, Mode)
```

Description

Déplace l'os d'une entité.

Arguments

#Entity L'entité à utiliser.

Os$ Le nom de l'os dans le mesh OGRE.

X, Y, Z Nouvelle position de l'os.

Mode Peut être une des valeurs suivantes :

```
#PB_Relative: Déplacement
 relatif, à partir de la
 position actuelle de
 l'os (par défaut).
#PB_Absolute: Déplacement
 absolu à la position
 spécifiée.
```

combinée avec l'une des valeurs suivantes :

```
#PB_Local : Déplacement
 local.
#PB_Parent: Déplacement
 par rapport à la
 position du parent.
#PB_World : Déplacement
 par rapport au monde.
```

Valeur de retour

Aucune.

Remarques

L'os doit être en mode manuel en utilisant EnableManualEntityBoneControl() .

Voir aussi

RotateEntityBone()

OS Supportés

Tous

93.26 FreeEntityBody

Syntaxe

```
FreeEntityBody(#Entity)
```

Description

Libère le corps associé à l'entité.

Arguments

#**Entity** L'entité à utiliser.

Valeur de retour

Aucune.

Voir aussi

CreateEntityBody()

OS Supportés

Tous

93.27 FreeEntityJoints

Syntaxe

```
FreeEntityJoints(#Entity)
```

Description

Libère toutes les articulations associées à une entité.

Arguments

#**Entity** L'entité à utiliser.

Valeur de retour

Aucune.

Voir aussi

EntityParentNode()

OS Supportés

Tous

93.28 EntityBoneX

Syntaxe

```
Resultat =
    EntityBoneX(#Entity, Os$
    [, DeplacementX,
    DeplacementY,
    DeplacementZ])
```

Description

Renvoie la position en X d'un os d'une entité.

Arguments

#Entity L'entité à utiliser.

Os$ Le nom de l'os dans le mesh OGRE .

DeplacementX, DeplacementY, DeplacementZ (optionnel) Le décalage par rapport à l'os.

Valeur de retour

Renvoie la position en X de l'os dans le monde.

Voir aussi

EntityBoneY() , EntityBoneZ()

OS Supportés

Tous

93.29 EntityBoneY

Syntaxe

```
Resultat =
    EntityBoneY(#Entity, Os$
    [, DeplacementX,
    DeplacementY,
    DeplacementZ])
```

Description

Renvoie la position en Y d'un os d'une entité.

Arguments

#**Entity** L'entité à utiliser.

Os$ Le nom de l'os dans le mesh OGRE .

DeplacementX, DeplacementY, DeplacementZ (optionnel) Le décalage par rapport à l'os.

Valeur de retour

Renvoie la position en Y de l'os dans le monde.

Voir aussi

EntityBoneX() , EntityBoneZ()

OS Supportés

Tous

93.30 EntityBoneZ

Syntaxe

```
Resultat =
    EntityBoneZ(#Entity, Os$
    [, DeplacementX,
    DeplacementY,
    DeplacementZ])
```

Description

Renvoie la position en Z d'un os d'une entité.

Arguments

#**Entity** L'entité à utiliser.

Os$ Le nom de l'os dans le mesh OGRE .

DeplacementX, DeplacementY, DeplacementZ (optionnel) Le décalage par rapport à l'os.

Valeur de retour

Renvoie la position en Z de l'os dans le monde.

Voir aussi

EntityBoneX() , EntityBoneY()

OS Supportés

Tous

93.31 EntityBonePitch

Syntaxe

```
Resultat.f =
    EntityBonePitch(#Entity,
    Os$)
```

Description

Renvoie le tangage (torsion de haut en bas) d'un os d'une entité.

Arguments

#**Entity** L'entité à utiliser.

Os$ Le nom de l'os dans le mesh OGRE.

Valeur de retour

La valeur du tangage de l'os.
Valeur toujours comprise entre -180.0 et 180.0 degrés.

Voir aussi

EntityBoneYaw() , EntityBoneRoll()

OS Supportés

Tous

93.32 EntityBoneYaw

Syntaxe

```
Resultat.f =
    EntityBoneYaw(#Entity, Os$)
```

Description

Renvoie le lacet (torsion horizontale de gauche à droite) d'un os d'une entité.

Arguments

#**Entity** L'entité à utiliser.

Os$ Le nom de l'os dans le mesh OGRE.

Valeur de retour

La valeur du lacet de l'os.
Valeur toujours comprise entre -180.0 et 180.0 degrés.

Voir aussi

EntityBonePitch() , EntityBoneRoll()

OS Supportés

Tous

93.33 EntityBoneRoll

Syntaxe

```
Resultat.f =
    EntityBoneRoll(#Entity,
    Os$)
```

Description

Renvoie le roulis (torsion verticale de gauche à droite) d'un os d'une entité.

Arguments

#**Entity** L'entité à utiliser.

Os$ Le nom de l'os dans le mesh OGRE.

Valeur de retour

La valeur du roulis de l'os.
Valeur toujours comprise entre -180.0 et 180.0 degrés.

Voir aussi

EntityBoneRoll() , EntityBoneYaw()

OS Supportés

Tous

93.34 EntityX

Syntaxe

```
Resultat = EntityX(#Entity [,
    Mode])
```

Description

Renvoie la position en X d'une entité.

Arguments

#**Entity** L'entité à utiliser.

Mode (optionnel) Le mode d'obtention du vecteur de direction de la caméra. Peut être l'une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 direction de la caméra
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 direction de la caméra
 par rapport à son parent.
```

Valeur de retour

Renvoie la position en X de l'entité dans le monde 3D.

Voir aussi

EntityY() , EntityZ()

OS Supportés

Tous

93.35 EntityY

Syntaxe

```
Resultat = EntityY(#Entity [,
    Mode])
```

Description

Renvoie la position en Y d'une entité.

Arguments

#**Entity** L'entité à utiliser.

Mode (optionnel) Le mode d'obtention du vecteur de direction de la caméra. Peut être l'une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 direction de la caméra
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 direction de la caméra
 par rapport à son parent.
```

Valeur de retour

Renvoie la position en Y de l'entité dans le monde 3D.

Voir aussi

EntityX() , EntityZ()

OS Supportés

Tous

93.36 EntityZ

Syntaxe

```
Resultat = EntityZ(#Entity [,
    Mode])
```

Description

Renvoie la position en Z d'une entité.

Arguments

#**Entity** L'entité à utiliser.

Mode (optionnel) Le mode d'obtention du vecteur de direction de la caméra. Peut être l'une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 direction de la caméra
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 direction de la caméra
 par rapport à son parent.
```

Valeur de retour

Renvoie la position en Z de l'entité dans le monde 3D.

Voir aussi

EntityX() , EntityY()

OS Supportés

Tous

93.37 FreeEntity

Syntaxe

```
FreeEntity(#Entity)
```

Description

Supprime une entité.

Arguments

#**Entity** L'entité à supprimer. Si #PB_All est spécifié, toutes les entités restantes sont libérées.

Valeur de retour

Aucune.

Remarques

Toutes les entities restantes sont automatiquement supprimées quand le programme se termine.

Voir aussi

CreateEntity()

OS Supportés

Tous

93.38 HideEntity

Syntaxe

```
HideEntity(#Entity, Etat)
```

Description

Affiche ou cache une entité.

Arguments

#**Entity** L'entité à utiliser.

Etat `#False: L'entité est affichée`
`#True : L'entité est cachée`

Valeur de retour

Aucune.

Voir aussi

CreateEntity() , FreeEntity()

OS Supportés

Tous

93.39 IsEntity

Syntaxe

```
Resultat = IsEntity(#Entity)
```

Description

Teste si une entité est correctement initialisée.

Arguments

#**Entity** L'entité à utiliser.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.

Voir aussi

CreateEntity() , FreeEntity()

OS Supportés

Tous

93.40 MoveEntity

Syntaxe

```
MoveEntity(#Entity, X, Y, Z
    [, Mode])
```

Description

Déplace une entité.

Arguments

#**Entity** L'entité à utiliser.
X, Y, Z Nouvelle position de l'entité.
Mode (optionnel) Peut être une des valeurs suivantes :

```
#PB_Relative: Déplacement
 relatif, à partir de la
 position actuelle de
 l'entité (par défaut).
#PB_Absolute: Déplacement
 absolu à la position
 spécifiée.
```

combinée avec l'une des valeurs suivantes :

```
#PB_Local : Déplacement
 local.
#PB_Parent: Déplacement
 par rapport à la
 position du parent.
#PB_World : Déplacement
 par rapport au monde.
```

Valeur de retour

Aucune.

Remarques

Le déplacement dans le monde 3D est relatif à la position actuelle de l'entité.

Voir aussi

RotateEntity() , ScaleEntity()

OS Supportés

Tous

93.41 RotateEntity

Syntaxe

```
RotateEntity(#Entity, X, Y, Z
    [, Mode])
```

Description

Effectue une rotation d'une entité.

Arguments

#Entity L'entité à utiliser.

X, Y, Z Angles de rotation.
Tous les angles sont en degrés, de 0 à 359.

Mode (optionnel)

```
      #PB_Absolute: Rotation
    absolue (par défaut).
  #PB_Relative: Rotation
    relative basée sur la
    rotation précédente de
    l'entité.
```

Valeur de retour

Aucune.

Voir aussi

MoveEntity() , ScaleEntity()

OS Supportés

Tous

93.42 RotateEntityBone

Syntaxe

```
RotateEntityBone(#Entity,
    Os$, X, Y, Z, Mode)
```

Description

Fait pivoter un os d'une entité.

Arguments

#Entity L'entité à utiliser.

Os$ Le nom de l'os dans le mesh OGRE.

X, Y, Z La valeur de rotation en X, Y, Z, entre 0 à 359 degrés.

Mode (optionnel)
```
      #PB_Absolute: Rotation
    absolue (par défaut).
   #PB_Relative: Rotation
    relative basée sur la
    rotation précédente de
    l'os.
```

Valeur de retour

Aucune.

Voir aussi

MoveEntityBone()

OS Supportés

Tous

93.43 ScaleEntity

Syntaxe

```
ScaleEntity(#Entity, X, Y, Z
    [, Mode])
```

Description

Change les dimensions d'une entité.

Arguments

#Entity L'entité à utiliser.

X, Y, Z Les facteurs d'échelle sur les trois axes.
Les dimensions de l'entité vont être multipliées par les valeurs X, Y, Z.

Mode (optionnel) Peut être une des valeurs suivantes :

```
#PB_Relative: Facteur
 d'échelle relatif, sur
 la base de la taille
 initiale (par défaut).
              L'utilisation
 de la valeur 1.0
 permettra de garder la
 taille inchangée.
#PB_Absolute: Facteur
 d'échelle absolue, dans
 l'unité du monde.
```

Valeur de retour

Aucune.

Remarques

Lorsque vous utilisez le mode #PB_Relative, alors la taille de l'entité sera multipliée par la valeur donnée pour obtenir la nouvelle taille.

Exemple

```
1 ScaleEntity(0, 2, 2, 2) ;
   Double la taille courante
   de l'entité.
2 ScaleEntity(0, 1, 1, 1) ;
   Ne change pas la taille de
   l'entité (Multiplication
   par '1').
3 ScaleEntity(0, 3, 1, 1) ;
   Change la taille en 'x': 3
   fois plus grande.
4 ScaleEntity(0, 1, 1, 1,
   #PB_Absolute) ;
   Réinitialise la taille de
   l'entité à 1,1,1.
```

Voir aussi

MoveEntity() , RotateEntity()

OS Supportés

Tous

93.44 EntityRoll

Syntaxe

```
Resultat.f =
    EntityRoll(#Entity [,
    Mode])
```

Description

Renvoie le roulis d'une entité. (Mouvement vertical de gauche à droite)

Arguments

#**Entity** L'entité à utiliser.

Mode (optionnel) Peut être une des valeurs suivantes :

```
#PB_Absolute : Valeur
  absolue, et le roulis du
  parent est ignoré (par
  défaut).
#PB_Relative : Valeur
  relative par rapport au
  roulis courant du parent.
```

Combinée avec

```
#PB_Engine3D_Raw      :
  Valeur brute, mais elle
  ne peut pas être
  utilisée avec
  RotateEntity()
pour récupérer la même
  orientation (par défaut).
#PB_Engine3D_Adjusted :
  Valeur ajustée, de sorte
  qu'elle peut être
  réutilisée avec
  RotateEntity()
pour récupérer la même
  orientation.
```

Valeur de retour

La valeur du roulis actuel de l'entité. Valeur toujours comprise entre -180.0 et 180.0 degrés.

Voir aussi

EntityYaw() , EntityPitch()

OS Supportés

Tous

93.45 EntityPitch

Syntaxe

```
Resultat.f =
    EntityPitch(#Entity [,
    Mode])
```

Description

Renvoie le tangage d'une entité.
(Mouvement vertical de haut en bas)

Arguments

#**Entity** L'entité à utiliser.

Mode (optionnel) Peut être une des valeurs suivantes :

```
#PB_Absolute : Valeur
  absolue, et le tangage
  du parent est ignoré
  (par défaut).
#PB_Relative : Valeur
  relative par rapport au
  tangage courant du
  parent.
```

Combinée avec

```
#PB_Engine3D_Raw      :
  Valeur brute, mais elle
  ne peut pas être
  utilisée avec
  RotateEntity()
pour récupérer la même
  orientation (par défaut).
#PB_Engine3D_Adjusted :
  Valeur ajustée, de sorte
  qu'elle peut être
  réutilisée avec
  RotateEntity()
pour récupérer la même
  orientation.
```

Valeur de retour

La valeur du tangage actuel de l'entité.
Valeur toujours comprise entre -180.0 et 180.0 degrés.

Voir aussi

EntityYaw() , EntityRoll()

OS Supportés

Tous

93.46 EntityYaw

Syntaxe

```
Resultat.f =
    EntityYaw(#Entity [, Mode])
```

Description

Renvoie le lacet d'une entité. (Mouvement horizontal de gauche à droite)

Arguments

#**Entity** L'entité à utiliser.

Mode (optionnel) Peut être une des valeurs suivantes :

```
#PB_Absolute : Valeur
  absolue, et le roulis du
  parent est ignoré (par
  défaut).
#PB_Relative : Valeur
  relative par rapport au
  roulis courant du parent.
```

Combinée avec

```
#PB_Engine3D_Raw      :
  Valeur brute, mais elle
  ne peut pas être
  utilisée avec
  RotateEntity()
pour récupérer la même
  orientation (par défaut).
#PB_Engine3D_Adjusted :
  Valeur ajustée, de sorte
  qu'elle peut être
  réutilisée avec
  RotateEntity()
pour récupérer la même
  orientation.
```

Valeur de retour

La valeur du lacet actuel de l'entité. Valeur toujours comprise entre -180.0 et 180.0 degrés.

Voir aussi

EntityPitch() , EntityRoll()

OS Supportés

Tous

93.47 GetEntityAttribute

Syntaxe

```
Resultat =
    GetEntityAttribute(#Entity,
    Attribut)
```

Description

Renvoie la valeur d'un attribut d'une entité.

Arguments

#**Entity** L'entité à utiliser.

Attribut

```
    #PB_Entity_Friction
            : Renvoie la
  valeur de la force de
  friction ou de
  frottement.
#PB_Entity_Restitution
       : Renvoie la
  valeur du coefficient de
  restitution de la
  vitesse.
#PB_Entity_LinearVelocity
    : Renvoie la vitesse
  linéaire courante (tous
  les axes).
#PB_Entity_LinearVelocityX
   : Renvoie la vitesse
  linéaire courante sur
  l'axe des 'x'.
#PB_Entity_LinearVelocityY
   : Renvoie la vitesse
  linéaire courante sur
  l'axe des 'y'.
#PB_Entity_LinearVelocityZ
   : Renvoie la vitesse
  linéaire courante sur
  l'axe des 'z'.
#PB_Entity_assCenterX
        : Renvoie la
  position du centre de
  masse en 'x'.
#PB_Entity_MassCenterY
       : Renvoie la
  position du centre de
  masse en 'y'.
#PB_Entity_MassCenterZ
       : Renvoie la
  position du centre de
  masse en 'z'.
#PB_Entity_NbSubEntities
     : Renvoie le nombre
  de sous-entités.
#PB_Entity_LinearSleeping
    : Renvoie la valeur
  de vitesse linéaire
  minimum en dessous de
  laquelle l'entité sera
  endormie.
#PB_Entity_AngularSleeping
   : Renvoie la valeur de
  vitesse angulaire
  minimum en dessous de
```

```
 laquelle l'entité sera
 endormie.
#PB_Entity_DeactivationTime
  : Renvoie le temps
 d'attente (en
 millisecondes) avant de
 mettre l'entité en mode
 veille lorsque les
 conditions ci-dessus
 sont remplies.
#PB_Entity_IsActive
          : Renvoie si un
 corps d'entité est actif
 (pas de veille).
#PB_Entity_AngularVelocityX
  : Renvoie la vitesse
 angulaire courante sur
 l'axe 'x'.
#PB_Entity_AngularVelocityY
  : Renvoie la vitesse
 angulaire courante sur
 l'axe 'y'.
#PB_Entity_AngularVelocityZ
  : Renvoie la vitesse
 angulaire courante sur
 l'axe 'z'.
#PB_Entity_AngularVelocity
   : Renvoie la vitesse
 angulaire courante (tous
 les axes).
#PB_Entity_HasContactResponse:
 Teste si le corps de
 l'entité est en contact.
#PB_Entity_ScaleX
            : Renvoie la
 valeur courante de
 l'échelle sur l'axe 'x'.
#PB_Entity_ScaleY
            : Renvoie la
 valeur courante de
 l'échelle sur l'axe 'y'.
#PB_Entity_ScaleZ
            : Renvoie la
 valeur courante de
 l'échelle sur l'axe 'z'.
```

Valeur de retour

Renvoie la valeur de l'attribut spécifié ou zéro si l'entité ne supporte pas l'attribut.

Voir aussi

SetEntityAttribute() , SetEntityMaterial() , FetchEntityMaterial()

OS Supportés

Tous

93.48 SetEntityAttribute

Syntaxe

```
SetEntityAttribute(#Entity,
    Attribut, Valeur)
```

Description

Assigne la valeur d'un attribut de l'articulation associée à une entité.

Arguments

#**Entity** L'entité à utiliser.

Attribut L'attribut à définir parmi :

```
    #PB_Entity_Friction
     : Change la valeur de la
     force de friction ou de
     frottement.
    #PB_Entity_Restitution
     : Change la valeur du
     coefficient de
     restitution de la
     vitesse.
    #PB_Entity_MinVelocity
     : Change la vitesse
     minimum linéaire. Comme
     cette valeur n'est pas
     stockée,
                                il
     doit être appelé à
     chaque fois que l'entité
     est déplacée.
    #PB_Entity_MaxVelocity
     : Change la vitesse
     maximum linéaire. Comme
     cette valeur n'est pas
     stockée,
                                il
     doit être appelé à
     chaque fois que l'entité
     est déplacée.
    #PB_Entity_ForceVelocity
     : Définit la vitesse
     linéaire de l'entité.
     Comme cette valeur n'est
     pas stockée,
                                il
     doit être appelé à
     chaque fois que l'entité
     est déplacée.
    #PB_Entity_LinearSleeping
     : Change la valeur de
     vitesse linéaire minimum
     en dessous de laquelle
     l'entité sera endormie.
    #PB_Entity_AngularSleeping
     : Change la valeur de
```

```
    vitesse angulaire
    minimum en dessous de
    laquelle l'entité sera
    endormie.
  #PB_Entity_DeactivationTime:
    Temps d'attente (en
    millisecondes) avant de
    mettre l'entité en mode
    veille lorsque les
    conditions ci-dessus
    sont remplies.
  #PB_Entity_DisableContactResponse:
    Désactive ou active les
    contacts physiques pour
    cette entité. La valeur
    peut être #True ou
    #False.
```

Valeur Valeur de l'attribut.

Valeur de retour

Aucune.

Voir aussi

GetEntityAttribute() , SetEntityMaterial() , FetchEntityMaterial()

OS Supportés

Tous

93.49 GetEntityCollisionMask

Syntaxe

```
Resultat =
    GetEntityCollisionMask(#Entity)
```

Description

Renvoie le masque de collision de l'entité actuelle, telle que déterminé par SetEntityCollisionFilter() .

Arguments

#**Entity** L'entité à utiliser.

Valeur de retour

le masque de collision de l'entité actuelle.

Voir aussi

SetEntityCollisionFilter() , GetEntityCollisionGroup()

OS Supportés

Tous

93.50 GetEntityCollisionGroup

Syntaxe

```
Resultat =
    GetEntityCollisionGroup(#Entity)
```

Description

Renvoie le groupe de collision de l'entité actuelle, telle que déterminée par SetEntityCollisionFilter() .

Arguments

#**Entity** L'entité à utiliser.

Valeur de retour

Le groupe actuel de collision de l'entité.

Voir aussi

SetEntityCollisionFilter() , GetEntityCollisionMask()

OS Supportés

Tous

93.51 SetEntityCollisionFilter

Syntaxe

```
SetEntityCollisionFilter(#Entity,
    CollisionGroupe,
    CollisionMasque)
```

Description

Définit le groupe de collision de l'entité et le masque.

Arguments

#**Entity** L'entité à utiliser.

CollisionGroupe Le nouveau groupe de collision.

CollisionMasque Le nouveau masque de collision.

Valeur de retour

Aucune.

Voir aussi

GetEntityCollisionGroup() ,
GetEntityCollisionMask() , RayCollide()

OS Supportés

Tous

93.52 AddSubEntity

Syntaxe

```
Resultat =
    AddSubEntity(#Entity,
    #SubEntity, Type,
    [DecalageX, DecalageY,
    DecalageZ [, TailleX,
    TailleY, TailleZ [, AxeX,
    AxeY, AxeZ]]])
```

Description

Ajoute une entité en tant que sous-entité à une entité existante.

Arguments

#**Entity** L'entité à utiliser.

#**SubEntity** L'entité à ajouter.

Type Définit comment le moteur physique gère la sous-entité :

```
#PB_Entity_StaticBody : Le
 body est statique, c'est
 à dire que le mesh ne
 peut pas être animé.
                        Ce
 mode permet des
 collisions très
 précises, car elles sont
 faites par rapport aux
 triangles
                        qui
 composent le mesh (connu
 comme la tri-mesh
 collision). Les
 collisions entre body
 static et les autres
 types de body
                        (sphere,
 box etc.) sont très
 rapides, mais entre deux
 body static, cela
 devient très lent.
                        C'est
 le type de body parfait
 pour utiliser une mesh
 comme un sol ou pour un
 monde statique.
```

```
#PB_Entity_PlaneBody  : Un
 plan 'virtuel' sur
 l'entity (avec les mêmes
 dimensions)
                        est
 utilisé pour gérer les
 collisions entre les
 autres entités.
#PB_Entity_ConeBody   : Un
 cône 'virtuel' englobant
 totalement l'entity
 (avec les mêmes
 dimensions)
                        est
 utilisée pour gérer les
 collisions entre les
 autres entités.
#PB_Entity_BoxBody    :
 Une boîte 'virtuelle'
 englobant totalement
 l'entity (avec les mêmes
 dimensions)
                        est
 utilisée pour gérer les
 collisions entre les
 autres entités.
#PB_Entity_SphereBody :
 Une sphère 'virtuelle'
 englobant totalement
 l'entity
                        est
 utilisée pour gérer les
 collisions entre les
 autres entités.
#PB_Entity_CylinderBody :
 Un cylindre 'virtuel'
 englobant totalement
 l'entity
                        est
 utilisée pour gérer les
 collisions entre les
 autres entités.
#PB_Entity_CapsuleBody :
 Une capsule 'virtuelle'
 englobant totalement
 l'entity
                        est
 utilisée pour gérer les
 collisions entre les
 autres entités.
#PB_Entity_ConvexHullBody
 : Une forme arbitraire
 'virtuelle' englobant
 totalement l'entity
                        définie
 par un nuage de vertices
 et la forme sera la plus
 petite
                        forme
```

```
convexe qui entoure les
vertices.
```

DecalageX, DecalageY, DecalageZ (optionnel) Le décalage ou translation (Offset) du body en X, Y et Z.

TailleX, TailleY, TailleZ (optionnel) Les dimensions de la boîte englobante autour du corps. Ne concerne que le type de corps suivant :

```
#PB_Entity_BoxBody      :
 TailleX, TailleY,
 TailleZ sont disponibles.
#PB_Entity_SphereBody   :
 TailleX est disponible.
#PB_Entity_ConeBody     :
 TailleX, TailleY sont
 disponibles.
#PB_Entity_CylinderBody:
 TailleX, TailleY sont
 disponibles.
#PB_Entity_CapsuleBody  :
 TailleX, TailleY,
 TailleZ sont disponibles.
```

Si TailleX = -1 alors le système utilise la boîte englobante.

AxeX, AxeY, AxeZ (optionnel) Les axes du corps. Ne concerne que le type de corps suivant :

```
#PB_Entity_PlaneBody
#PB_Entity_CylinderBody
#PB_Entity_CapsuleBody
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Il est nécessaire d'utiliser la fonction CreateEntityBody() () avec l'option #PB_Entity_CompoundBody après l'ajout de toutes les sous-entités.

Voir aussi

CreateEntity()

OS Supportés

Tous

93.53 EntityDirection

Syntaxe

```
EntityDirection(#Entity, X,
    Y, Z [, Mode,
    VecteurDirectionLocal])
```

Description

Définit la direction de l'entité.

Arguments

#**Entity** L'entité à utiliser.

X, Y, Z Le vecteur direction (valeur entre -1.0 et 1.0).

Mode (optionnel) Le mode de direction. Il peut s'agir de l'une des valeurs suivantes :

```
#PB_Local : Mouvement
 local.
#PB_Parent: Mouvement
 relatif à la position
 des parents.
#PB_World : Mouvement
 relatif au monde.
```

VecteurDirectionLocal (optionnel) Le vecteur local de direction. Il peut s'agir de l'une des valeurs suivantes :

```
#PB_Vector_X
#PB_Vector_Y
#PB_Vector_Z
#PB_Vector_NegativeX
#PB_Vector_NegativeY
#PB_Vector_NegativeZ
```

Valeur de retour

Aucune.

Voir aussi

EntityDirectionX() , EntityDirectionY() , EntityDirectionZ()

OS Supportés

Tous

93.54 EntityDirectionX

Syntaxe

```
Resultat =
    EntityDirectionX(#Entity)
```

Description

Obtenir la direction 'x' de l'entité.

Arguments

#**Entity** L'entité à utiliser.

Valeur de retour

Renvoie la direction 'x' de l'entité.

Voir aussi

EntityDirection() , EntityDirectionY() , EntityDirectionZ()

OS Supportés

Tous

93.55 EntityDirectionY

Syntaxe

```
Resultat =
    EntityDirectionY(#Entity)
```

Description

Obtenir la direction 'y' de l'entité.

Arguments

#**Entity** L'entité à utiliser.

Valeur de retour

Renvoie la direction 'y' de l'entité.

Voir aussi

EntityDirection() , EntityDirectionX() , EntityDirectionZ()

OS Supportés

Tous

93.56 EntityDirectionZ

Syntaxe

```
Resultat =
    EntityDirectionZ(#Entity)
```

Description

Obtenir la direction 'z' de l'entité.

Arguments

#**Entity** L'entité à utiliser.

Valeur de retour

Renvoie la direction 'z' de l'entité.

Voir aussi

EntityDirection() , EntityDirectionX() , EntityDirectionY()

OS Supportés

Tous

93.57 GetEntityMesh

Syntaxe

```
Resultat =
    GetEntityMesh(#Entity)
```

Description

Obtenir le #Mesh utilisé par l'entité.

Arguments

#**Entity** L'entité à utiliser.

Valeur de retour

Renvoie le #Mesh utilisé par l'entité.

Voir aussi

CreateEntity()

OS Supportés

Tous

Chapitre 94

EntityAnimation

Généralités

Une entité, ou 'Entity' en anglais est un objet 3D composé d'un mesh (un maillage 3D) et d'une matière (ou matériau). Une entité peut être déplacée et transformées en temps réel.
Cette bibliothèque permet le contrôle des animations propres aux entités, comme faire marcher un personnage par exemple.
Attention, le mesh associé à l'entité doit avoir un squelette avec des animations prédéfinies.
InitEngine3D() doit être appelé avec succès avant d'utiliser les fonctions d'animations d'entités.

OS Supportés

Tous

94.1 AddEntityAnimationTime

Syntaxe

```
AddEntityAnimationTime(#Entité,
    Animation$, Temps)
```

Description

Ajoute du temps à l'animation d'une entité.

Arguments

#**Entité** L'entité à utiliser.

Animation$ Le nom de l'animation.
Les animations sont stockées dans l'objet mesh et de manière sensible à la casse (ex : "Marche" sera une animation différente de "marche").
Si l'animation n'est pas trouvée ou le mesh n'a pas de squelette alors cette fonction n'aura aucun effet.

Temps Le temps à ajouter (en millisecondes) à l'animation.

Valeur de retour

Aucune.

Voir aussi

StartEntityAnimation()

OS Supportés

Tous

94.2 StartEntityAnimation

Syntaxe

```
StartEntityAnimation(#Entité,
    Animation$ [, Options])
```

Description

Lance l'animation d'une entité.
L'animation est toujours recommencée depuis le début.

Arguments

#**Entité** L'entité à utiliser.

Animation$ Le nom de l'animation.
Les animations sont stockées dans l'objet mesh de manière sensible à la casse ex : "Marche" sera une animation différente de "marche").
Si l'animation n'est pas trouvée ou le mesh n'a pas de squelette alors cette fonction n'aura aucun effet.

Options Combinaison des valeurs suivantes :

```
    #PB_EntityAnimation_Once
      : Joue l'animation une
      seule fois.
                                  Par
      défaut, l'animation
      boucle automatiquement
                                  lorsque
      son extrémité est
      atteinte.
                                  EntityAnimationStatus()
   peut être utilisé pour
                                  détecter
      la fin de l'animation.
    #PB_EntityAnimation_Manual:
      Lance l'animation en
      mode manuel, le temps ne
```

sera pas automatiquement ajouté après chaque RenderWorld().

.

AddEntityAnimationTime() doit être appelé manuellement pour mettre à jour le temps de l'animation.

Valeur de retour

Aucune.

Voir aussi

StopEntityAnimation() ,
EntityAnimationStatus() ,
AddEntityAnimationTime()

OS Supportés

Tous

94.3 StopEntityAnimation

Syntaxe

```
StopEntityAnimation(#Entité,
    Animation$)
```

Description

Arrête l'animation d'une entité.

Arguments

#**Entité** L'entité à utiliser.

Animation$ Le nom de l'animation.
Les animations sont stockées dans l'objet mesh de manière sensible à la casse (ex : "Marche" sera une animation différente de "marche").
Si l'animation n'est pas trouvée ou le mesh n'a pas de squelette alors cette fonction n'aura aucun effet.

Valeur de retour

Aucune.

Voir aussi

StartEntityAnimation()

OS Supportés

Tous

94.4 EntityAnimationStatus

Syntaxe

```
Resultat =
    EntityAnimationStatus(#Entité,
    Animation$)
```

Description

Renvoie l'état de l'animation d'une entité.

Arguments

#**Entité** L'entité à utiliser.

Animation$ Le nom de l'animation.
Les animations sont stockées dans l'objet mesh de manière sensible à la casse (ex : "Marche" sera une animation différente de "marche").
Si l'animation n'est pas trouvée ou le mesh n'a pas de squelette alors cette fonction n'aura aucun effet.

Valeur de retour

```
#PB_EntityAnimation_Stopped:
 L'animation est à l'arrêt
 ou terminée.
#PB_EntityAnimation_Started:
 L'animation est en cours
 d'exécution.
#PB_EntityAnimation_Unknown:
 L'animation n'existe pas
 dans le mesh.
```

Voir aussi

StartEntityAnimation() ,
StopEntityAnimation()

OS Supportés

Tous

94.5 EntityAnimationBlendMode

Syntaxe

```
EntityAnimationBlendMode(#Entité,
    Mode)
```

Description

Change le blendmode (mélange des couleurs) de l'animation d'une entité.

Arguments

#**Entité** L'entité à utiliser.

Mode

```
      #PB_EntityAnimation_Average
      : Le mélange est le
    résultat de la moyenne
    des deux animations (par
    défaut).
                                    Par
    exemple, si la première
    animation tourne un bras
    à 40 degrés et la
                                    deuxième
    animation fait tourner
    le bras à 90 degrés, le
    bras va tourner
                                    de
    (40+90)/2 = 65 degrés
    (si les deux animations
    tournent en même temps
                                    en
    mode full weight).
   #PB_EntityAnimation_Cumulative:
    Le mélange est la somme
    des deux animations.
                                    Par
    exemple, si la première
    animation tourne un bras
    à 40 degrés
                                    et
    la deuxième animation
    fait tourner le bras à
    90 degrés, alors
                                    le
    bras tourne à 40+90 =
    130 degrés (si les deux
    animations tournent
                                    en
    même temps en mode full
    weight).
```

Valeur de retour

Aucune.

Remarques

Lors du passage d'une animation à l'autre avec SetEntityAnimationWeight() , un mélange est appliqué pour une transition en douceur entre les animations.

Voir aussi

StartEntityAnimation() , SetEntityAnimationWeight()

OS Supportés

Tous

94.6 GetEntityAnimationTime

Syntaxe

```
Resultat =
    GetEntityAnimationTime(#Entité,
    Animation$)
```

Description

Renvoie le temps de l'animation d'une entité.

Arguments

#**Entité** L'entité à utiliser.

Animation$ Le nom de l'animation.
Les animations sont stockées dans l'objet mesh de manière sensible à la casse (ex : "Marche" sera une animation différente de "marche").
Si l'animation n'est pas trouvée ou le mesh n'a pas de squelette alors cette fonction n'aura aucun effet.

Valeur de retour

Le temps courant de l'animation (en millisecondes) ou zéro si l'animation ne fonctionne pas.

Voir aussi

StartEntityAnimation() , AddEntityAnimationTime() , SetEntityAnimationTime()

OS Supportés

Tous

94.7 SetEntityAnimationTime

Syntaxe

```
SetEntityAnimationTime(#Entité,
    Animation$, Temps)
```

Description

Modifie le temps courant de l'animation d'une entité.

Arguments

#**Entité** L'entité à utiliser.

Animation$ Le nom de l'animation.
Les animations sont stockées dans l'objet mesh de manière sensible à la casse (ex : "Marche" sera une animation différente de "marche").
Si l'animation n'est pas trouvée ou le mesh n'a pas de squelette alors cette fonction n'aura aucun effet.

Temps Le temps absolu à définir (en millisecondes).

Valeur de retour

Aucune.

Remarques

Il s'agit d'un temps absolu.
Pour changer le temps par rapport au temps actuel, utiliser AddEntityAnimationTime() .

Voir aussi

StartEntityAnimation() , AddEntityAnimationTime() , GetEntityAnimationTime()

OS Supportés

Tous

94.8 GetEntityAnimationLength

Syntaxe

```
Resultat =
    GetEntityAnimationLength(#Entité,
    Animation$)
```

Description

Renvoie la longueur de l'animation d'une entité.

Arguments

#**Entité** L'entité à utiliser.

Animation$ Le nom de l'animation.
Les animations sont stockées dans l'objet mesh de manière sensible à la casse (ex : "Marche" sera une animation différente de "marche").
Si l'animation n'est pas trouvée ou le mesh n'a pas de squelette alors cette fonction n'aura aucun effet.

Valeur de retour

La durée de l'animation de l'entité (en millisecondes).

Voir aussi

StartEntityAnimation() , SetEntityAnimationLength()

OS Supportés

Tous

94.9 SetEntityAnimationLength

Syntaxe

```
SetEntityAnimationLength(#Entité,
    Animation$, Duree)
```

Description

Change la longueur de l'animation d'une entité.

Arguments

#Entité L'entité à utiliser.

Animation$ Le nom de l'animation.
Les animations sont stockées dans l'objet mesh de manière sensible à la casse (ex : "Marche" sera une animation différente de "marche").
Si l'animation n'est pas trouvée ou le mesh n'a pas de squelette alors cette fonction n'aura aucun effet.

Duree La nouvelle durée de l'animation de l'entité (en millisecondes).

Valeur de retour

Aucune.

Voir aussi

StartEntityAnimation() , GetEntityAnimationLength()

OS Supportés

Tous

94.10 GetEntityAnimationWeight

Syntaxe

```
Resultat.f =
    GetEntityAnimationWeight(#Entité,
    Animation$)
```

Description

Renvoie le poids de l'animation d'une entité.

Arguments

#**Entité** L'entité à utiliser.

Animation$ Le nom de l'animation.
Les animations sont stockées dans l'objet mesh de manière sensible à la casse (ex : "Marche" sera une animation différente de "marche").
Si l'animation n'est pas trouvée ou le mesh n'a pas de squelette alors cette fonction n'aura aucun effet.

Valeur de retour

Le poids actuel de l'animation (valeur comprise entre 0.0 et 1.0). Si le poids est égal à 0, alors l'animation est sans effet. Si le poids est égal à 1 alors l'animation joue pleinement.

Remarques

Le poids est utile lorsque vous jouez plusieurs animations à la fois.
Par exemple, pour faire une transition en douceur d'une animation à une autre, il est possible de réduire progressivement le poids de la première animation et d'augmenter le poids de la deuxième animation.
L'EntityAnimationBlendMode() affecte également la façon dont les animations sont mélangées.

Voir aussi

StartEntityAnimation() ,
EntityAnimationBlendMode()

OS Supportés

Tous

94.11 SetEntityAnimationWeight

Syntaxe

```
SetEntityAnimationWeight(#Entité,
    Animation$, Poids.f)
```

Description

Modifie le poids de l'animation d'une entité.

Arguments

#Entité L'entité à utiliser.

Animation$ Le nom de l'animation.
Les animations sont stockées dans l'objet mesh de manière sensible à la casse (ex : "Marche" sera une animation différente de "marche").
Si l'animation n'est pas trouvée ou le mesh n'a pas de squelette alors cette fonction n'aura aucun effet.

Poids Le nouveau poids de l'animation (valeur comprise entre 0.0 et 1.0). Si le poids est égal à 0 alors l'animation est sans effet. Si le poids est égal à 1 alors l'animation joue pleinement.

Valeur de retour

Aucune.

Remarques

Le poids est utile lorsque vous jouez plusieurs animations à la fois.
Par exemple, pour faire une transition en douceur d'une animation à une autre, il est possible de réduire progressivement le poids de la première et d'augmenter le poids de la deuxième animation.
L'EntityAnimationBlendMode() affecte également la façon dont les animations sont mélangées.

Voir aussi

StartEntityAnimation() ,
EntityAnimationBlendMode()

OS Supportés

Tous

94.12 UpdateEntityAnimation

Syntaxe

```
UpdateEntityAnimation(#Entité,
    Animation$)
```

Description

Mise à jour de l'animation d'une entité.

Arguments

#**Entité** L'entité à utiliser.

Animation$ Le nom de l'animation.
Les animations sont stockées dans l'objet mesh de manière sensible à la casse (ex : "Marche" sera une animation différente de "marche").
Si l'animation n'est pas trouvée ou le mesh n'a pas de squelette alors cette fonction n'aura aucun effet.

Valeur de retour

Aucune.

Remarques

Par exemple, si les sommets du mesh ont été modifiés alors le cache d'animation doit être recalculé.

Voir aussi

StartEntityAnimation()

OS Supportés

Tous

Chapitre 95

File

Généralités

Les fichiers sont les principaux modes de stockage d'informations des ordinateurs actuels. PureBasic vous permet de les gérer de manière simple et efficace. Un nombre quelconque de fichiers peuvent être ouverts simultanément. Cette bibliothèque utilise des fonctions optimisées pour accélérer la vitesse de lecture/écriture des informations. Toutes les fonctions permettent la gestion des très gros fichiers, jusqu'à 2 ^64 octets si le système de fichiers les supporte.
Pour de grandes quantités de données, il peut être utile de charger les données dans un tableau , une liste ou une map (carte) et l'utilisation de blocs de mémoire peut aussi être une bonne idée.
Pour obtenir les chemins de fichiers valides pour la lecture et l'enregistrement des données, jetez un oeil à la bibliothèque FileSystem et à la bibliothèque Requester .

OS Supportés

Tous

95.1 CloseFile

Syntaxe

```
CloseFile(#Fichier)
```

Description

Ferme un fichier.

Arguments

#Fichier Le fichier à fermer.
Si #PB_All est spécifié, tous les fichiers restants sont fermés.

Valeur de retour

Aucune.

Remarques

Une fois que le fichier est fermé, il ne peut plus être utilisé.
Fermer un fichier permet de s'assurer que toutes ses informations sont effectivement écrites sur le disque.
Tous les fichiers restant ouverts sont automatiquement fermés quand le programme se termine.
Voir un exemple dans ReadFile() ou CreateFile() .

Exemple

```
1  If CreateFile(0,
    "Text.txt")          ;
    création d'un nouveau
    fichier texte...
2    For a=1 To 10
3      WriteStringN(0, "Ligne
    "+Str(a)) ; écriture de 10
    lignes (suivies du code
    'Fin de Ligne')
4    Next
5    For a=1 To 10
6      WriteString(0,
    "Chaîne"+Str(a))  ; ajoute
    10 chaînes sur la même
    ligne (le code 'Fin de
    Ligne' n'est pas ajouté)
7    Next
8    CloseFile(0)
                          ;
    ferme le fichier
    précédemment ouvert et
    enregistre les données
9  Else
10   MessageRequester("Information","Impossible
    de créer le fichier!")
11 EndIf
```

Voir aussi

CreateFile() , OpenFile() , ReadFile()

OS Supportés

Tous

95.2 CreateFile

Syntaxe

```
Resultat =
    CreateFile(#Fichier,
    NomFichier$ [, Options])
```

Description

Crée un fichier vide ou recrée une fichier vide s'il existe déjà.

Arguments

#Fichier Le numéro du nouveau fichier. #PB_Any peut être utilisé pour générer automatiquement ce numéro.

NomFichier$ Le nom et le chemin vers le nouveau fichier.
Si le nom de fichier ne contient pas de chemin complet alors le chemin courant sera utilisé.

Options (optionnel) Peut être une combinaison de :

```
    #PB_File_SharedRead : Le
     fichier ouvert par un
     processus peut être lu
     par un autre processus
     (Windows uniquement).
    #PB_File_SharedWrite: le
     fichier ouvert par un
     processus peut être
     écrit par un autre
     processus (Windows
     uniquement).
    #PB_File_NoBuffering: Le
     système interne de mise
     en mémoire tampon de
     PureBasic sera désactivé
     pour ce fichier.
                          FileBuffersSize()
  ne peut pas être utilisé
     sur ce fichier.
```

combiné avec l'une des valeurs suivantes (les options suivantes affectent le comportement de WriteString() , WriteStringN() , ReadString() , ReadCharacter() et WriteCharacter()) :

```
    #PB_Ascii   : Toute
     opération de
     lecture/écriture des
     chaînes de caractères
     utilisera le mode ASCII
                 (Par défaut
     pour les exécutables
     compilés en mode ASCII).
    #PB_UTF8    : Toute
     opération de
     lecture/écriture des
     chaînes de caractères
     utilisera le mode UTF-8
```

```
                (Par défaut
  pour les exécutables
  compilés en mode
  Unicode).
 #PB_Unicode: Toute
  opération de
  lecture/écriture des
  chaînes de caractères
  utilisera le mode
  Unicode.
```

Valeur de retour

Renvoie une valeur non nulle si le fichier a été créé avec succès, zéro sinon.
Si #PB_Any a été utilisé comme paramètre #Fichier alors le nouveau numéro généré est renvoyé en cas de succès.

Remarques

Attention, si le fichier existe déjà, il sera remplacé par un fichier vide !
La fonction FileSize() peut être utilisée pour déterminer si un fichier existe avant de l'écraser.
Pour ouvrir un fichier existant en lecture et en écriture, utilisez la fonction OpenFile() .
Pour ouvrir un fichier en lecture seule, utilisez ReadFile() .

Exemple

```
1  If CreateFile(0,
    "Text.txt")          ; crée
    un nouveau fichier texte
    ou recrée une fichier
    texte vide s'il existe
    déjà ...
2   For a=1 To 10
3     WriteStringN(0, "Ligne
    "+Str(a)) ; écriture de 10
    lignes (suivies du code
    'Fin de Ligne')
4   Next
5   For a=1 To 10
6     WriteString(0,
    "Chaîne"+Str(a))  ; ajoute
    10 chaînes sur la même
    ligne (le code 'Fin de
    Ligne' n'est pas ajouté)
7   Next
8   CloseFile(0)
                          ;
    ferme le fichier
    précédemment ouvert et
    enregistre les données
9  Else
```

```
10    MessageRequester("Information","Impossible
      de créer le fichier!")
11  EndIf
```

Voir aussi

OpenFile() , ReadFile() , CloseFile()

OS Supportés

Tous

95.3 Eof

Syntaxe

```
Resultat = Eof(#Fichier)
```

Description

Teste si la fin d'un fichier (Eof : End Of File) a été atteinte.

Arguments

#**Fichier** Le fichier à utiliser.

Valeur de retour

Renvoie une valeur non nulle si la fin du fichier a été atteinte, zéro sinon.

Exemple

```
1   If ReadFile(0, "Texte.txt")
      ; Si le fichier peut être
      lu , on continue...
2     While Eof(0) = 0
                ; Boucle tant
      que la fin du fichier
      n'est pas atteinte. (Eof =
      'End Of File')
3       Debug ReadString(0)
          ; Affiche ligne par
      ligne le contenu du fichier
4     Wend
5     CloseFile(0)
                    ; Ferme le
      fichier précédemment ouvert
6   Else
7     MessageRequester("Information","Impossible
      d'ouvrir le fichier!")
8   EndIf
```

Voir aussi

Lof() , Loc() , CreateFile() , OpenFile() , ReadFile()

OS Supportés

Tous

95.4 FileBuffersSize

Syntaxe

```
FileBuffersSize(#Fichier,
    Taille)
```

Description

Change la taille des caches de lecture/écriture.

Arguments

#Fichier Le fichier à utiliser.
Si #PB_Default est utilisé alors la valeur par défaut de la taille des caches sera changée, et tous les futurs appels à OpenFile() , CreateFile() et ReadFile() utiliseront cette nouvelle valeur.

Taille La nouvelle taille du cache (en octet).
La taille par défaut du cache est de 4096 octets.
Une taille de zéro désactive complètement le cache et les informations sont immédiatement écrites dans le fichier.

Valeur de retour

Aucune.

Remarques

Pour des raisons de performances, le cache devrait être d'au moins 1024 octets. Quand le cache est actif, les informations ne sont réellement écrites dans le fichier que lorsque le cache est plein, ou que le fichier est fermé. La commande FlushFileBuffers() permet de forcer cette écriture à tout moment.

Exemple

```
1  Fichier$=OpenFileRequester("Ouvrir
     un fichier","","",0)
2  texte$="FileBuffersSize()
     resultats:"+#CRLF$+"================"+#CRLF$
3
```

```
4   TailleTampon=4096
5   While TailleTampon<1048577
6   texte$=texte$+"Buffer:
     "+Str(TailleTampon)+#CRLF$
7
8    If ReadFile(0,Fichier$)
9    FileBuffersSize(0,TailleTampon)
    ; Changement de la taille
    du tampon
10   start=ElapsedMilliseconds()
        ; Début du
    chronométrage
11
12   While Eof(0)=#False
           ; Lecture du
    fichier octet par octet
13       ReadByte(0)
14   Wend
15   stop=ElapsedMilliseconds()
     ; Fin du chronométrage
16
17   CloseFile(0)
18   EndIf
19   texte$+"ReadByte:
    "+Str(stop-start)+#CRLF$+#CRLF$
20
21  TailleTampon =
     TailleTampon*2 ;
     Augmentation du tampon
22  Wend
23
24  MessageRequester("FileBuffersSize",texte$)
     ; Affichage
```

Voir aussi

FlushFileBuffers()

OS Supportés

Tous

95.5 FileID

Syntaxe

```
Resultat = FileID(#Fichier)
```

Description

Renvoie l'identifiant unique d'un fichier dans le système d'exploitation.

Arguments

#**Fichier** Le fichier à utiliser.

Valeur de retour

Renvoie le handle du fichier.

Exemple

```
1   hWnd=CreateFile(0,
     "Text.txt")
2       Debug FileID(0)
3   CloseFile(0)
4       Debug hWnd
```

Voir aussi

CreateFile() , OpenFile() , ReadFile()

OS Supportés

Tous

95.6 FileSeek

Syntaxe

```
FileSeek(#Fichier, Position.q
    [, Mode])
```

Description

Change la position du pointeur de lecture/écriture.

Arguments

#**Fichier** Le fichier à utiliser.

Position.q La nouvelle position en octet par rapport au début du fichier.

Mode (optionnel) Le mode de recherche. Peut être l'une des valeurs suivantes :

```
#PB_Absolute: Position
 absolue (par défaut).
#PB_Relative: Position
 relative, un décalage
 (offset) positif ou
 négatif
              par rapport
 à la position courante
 du pointeur dans le
 fichier.
```

Valeur de retour

Aucune.

Exemple

```
Fichier$ =
  OpenFileRequester("Sélectionner
  un fichier","","Texte
  (.txt)|*.txt|Tous
  (*.*)|*.*",0)
If Fichier$
  If ReadFile(0, Fichier$)
    length = Lof(0)
                          ;
  Lit la taille en octets du
  fichier
    FileSeek(0, length -
  10)              ; place
  le pointeur 10 caractères
  avant la fin du fichier
    Debug "Position: " +
  Str(Loc(0))      ; Affiche
  la position du pointeur
    *MemoryID =
  AllocateMemory(10)
  ; alloue un bloc mémoire
  pour 10 octets
    If *MemoryID
      bytes = ReadData(0,
  *MemoryID, 10)  ; Lit les
  10 derniers caractères du
  fichier
      Debug PeekS(*MemoryID)
    EndIf
    CloseFile(0)
  EndIf
EndIf
```

Voir aussi

Loc() , Lof()

OS Supportés

Tous

95.7 FlushFileBuffers

Syntaxe

```
Resultat =
    FlushFileBuffers(#Fichier)
```

Description

Ecrit immédiatement le contenu du cache dans un fichier.

Arguments

#**Fichier** Le fichier à utiliser.

Valeur de retour

Renvoie une valeur non nulle si le tampon a été écrit correctement sur le disque. Si une erreur survient (Ex : Erreur de disque, disque plein...), il renverra zéro.

Remarques

Pour plus d'informations sur la gestion du cache des fichiers, consulter FileBuffersSize() .

Voir aussi

FileBuffersSize()

OS Supportés

Tous

95.8 IsFile

Syntaxe

```
Resultat = IsFile(#Fichier)
```

Description

Teste si le numéro de fichier #Fichier est correctement initialisé.

Arguments

#Fichier Le fichier à tester.

Valeur de retour

Renvoie une valeur non nulle en cas de succès zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.

Voir aussi

CreateFile() , OpenFile() , ReadFile()

OS Supportés

Tous

95.9 Loc

Syntaxe

```
Resultat.q = Loc(#Fichier)
```

Description

Renvoie la position du pointeur de lecture/écriture dans un fichier.

Arguments

#**Fichier** Le fichier à utiliser.

Valeur de retour

Renvoie la position du pointeur en octet par rapport au début du fichier.

Exemple

```
Fichier$ = OpenFileRequester("Sélectionner un fichier","","Texte (.txt)|*.txt|Tous (*.*)|*.*",0)
If Fichier$
  If ReadFile(0, Fichier$)
    length = Lof(0)                    ; Lit la taille en octets du fichier
    FileSeek(0, length - 10)           ; place le pointeur 10 caractères avant la fin du fichier
    Debug "Position: " + Str(Loc(0))   ; Affiche la position du pointeur
    *MemoryID = AllocateMemory(10)     ; alloue un bloc mémoire pour 10 octets
    If *MemoryID
      bytes = ReadData(0, *MemoryID, 10)  ; Lit les 10 derniers caractères du fichier
      Debug PeekS(*MemoryID)
    EndIf
    CloseFile(0)
  EndIf
EndIf
```

Voir aussi

FileSeek() , Lof()

OS Supportés

Tous

95.10 Lof

Syntaxe

```
Resultat.q = Lof(#Fichier)
```

Description

Lof (Length of File) renvoie la taille d'un fichier.

Arguments

#**Fichier** Le fichier à utiliser.

Valeur de retour

Renvoie la longueur d'un fichier en octets.

Exemple

```
1   Fichier$ =
     OpenFileRequester("Sélectionnez
     un fichier","","Texte
     (.txt)|*.txt|Tous
     (*.*)|*.*",0)
2   If Fichier$
3     If ReadFile(0, Fichier$)
4       length = Lof(0)

      ; Lit la taille en octets
      du fichier
5       *MemoryID =
      AllocateMemory(length)
              ; alloue un bloc
      mémoire de la taille du
      fichier
6       If *MemoryID
7         bytes = ReadData(0,
      *MemoryID, length)    ; Lit
      les données du fichier et
      les place dans le bloc
      mémoire
8         Debug "Nombre
      d'octets lus: " +
      Str(bytes)
9       EndIf
10      CloseFile(0)
11    EndIf
12  EndIf
```

Voir aussi

Loc() , FileSeek() , FileSize()

OS Supportés

Tous

95.11 OpenFile

Syntaxe

```
Resultat = OpenFile(#Fichier,
    NomFichier$ [, Options])
```

Description

Ouvre un fichier en lecture et en écriture ou le crée s'il n'existe pas.

Arguments

#Fichier Le fichier à utiliser. #PB_Any peut être utilisé pour générer automatiquement ce numéro.

NomFichier$ Le nom du fichier à ouvrir.

Options (optionnel) Peut être une combinaison de :

```
#PB_File_SharedRead : Le
 fichier ouvert peut être
 lu par un autre
 processus (Windows
 uniquement).
#PB_File_SharedWrite: Le
 fichier ouvert peut être
 écrit par un autre
 processus (Windows
 uniquement).
#PB_File_Append     : La
 position du pointeur de
 fichier sera fixée à la
 fin du fichier.
#PB_File_NoBuffering: Le
 système interne de mise
 en mémoire tampon de
 PureBasic sera désactivé
 pour ce fichier.
                        FileBuffersSize()
 ne peut pas être utilisé
 sur ce fichier.
```

combiné avec l'une des valeurs suivantes (les options suivantes affectent le comportement de WriteString() , WriteStringN() , ReadString() , ReadCharacter() et WriteCharacter()) :

```
#PB_Ascii  : Toute
 opération de
 lecture/écriture des
 chaînes de caractères
 utilisera le mode ASCII
```

```
#PB_UTF8    : Toute
 opération de
 lecture/écriture des
 chaînes de caractères
 utilisera le mode UTF-8
 (Par défaut).
#PB_Unicode: Toute
 opération de
 lecture/écriture des
 chaînes de caractères
 utilisera le mode
 Unicode.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès zéro sinon.
Si #PB_Any est utilisé pour le paramètre '#Fichier', le numéro du nouveau fichier sera renvoyé dans 'Resultat'.

Remarques

Pour que l'ouverture du fichier avec OpenFile() soit un succès, le fichier doit avoir les droits en lecture et écriture. Ceci peut devenir un problème s'il faut manipuler des fichiers sur un CD-Rom par exemple. Dans ce cas, il convient d'utiliser ReadFile() quand seule la lecture du fichier est nécessaire.
Pour écraser un fichier existant avec un nouveau fichier vide, utilisez la fonction CreateFile() .
Pour ajouter des données à la fin d'un fichier existant il faut au préalable positionner le pointeur de lecture/écriture en utilisant l'option #PB_File_Append.

Exemple

```
1  If OpenFile(0, "Test.txt")
       ; Ouvre un fichier
   existant ou en crée un
   nouveau s'il n'existait pas
2   FileSeek(0, Lof(0))
           ; Place le
   pointeur à la fin du
   fichier en utilisant le
   résultat de Lof()
3   WriteStringN(0, "... une
   autre ligne à la fin du
   fichier.")
4   CloseFile(0)
5  EndIf
```

Voir aussi

CreateFile() , ReadFile() , CloseFile()

OS Supportés

Tous

95.12 TruncateFile

Syntaxe

```
TruncateFile(#Fichier)
```

Description

Coupe le fichier à la position courante et supprime toutes les données qui suivent.

Arguments

#**Fichier** Le fichier à utiliser.

Valeur de retour

Aucune.

Remarques

Cette fonction peut être employée pour rendre un fichier plus court sans le recréer entièrement. Pour faire un fichier plus long, ajoutez simplement plus de données avec les commandes d'écriture de cette bibliothèque.

Exemple

```
1   If OpenFile(0, "Test.txt")
        ; Ouvre un fichier
     existant ou en crée un
     nouveau s'il n'existait pas
2     FileSeek(0, Lof(0))
            ; Place le
     pointeur à la fin du
     fichier en utilisant le
     résultat de Lof()
3     For i = 1 To 10
4       WriteStringN(0, "...
     une autre ligne à la fin
     du fichier.")
5     Next i
6     CloseFile(0)
7   EndIf
8
9   Taille.q=FileSize("Test.txt")
     ; Taille
10  Debug Taille
11  If OpenFile(0, "Test.txt")
```

```
12      FileSeek(0,Taille/2) ; Le pointeur se place à la moitié du fichier
13      TruncateFile(0)      ; On coupe le fichier au niveau du pointeur
14      CloseFile(0)
15    EndIf
16    Debug FileSize("Test.txt")  ; Nouvelle Taille
```

Voir aussi

FileSeek() , Loc()

OS Supportés

Tous

95.13 ReadAsciiCharacter

Syntaxe

```
Resultat.a =
    ReadAsciiCharacter(#Fichier)
```

Description

Lit un caractère ascii (1 octet non signé).

Arguments

#**Fichier** Le fichier à utiliser.

Valeur de retour

Renvoie le caractère ascii ou zéro si une erreur s'est produite.

Remarques

Exemple

```
1   If CreateFile(0, "Test.txt")  ; Ouvre un fichier existant ou en crée un nouveau s'il n'existait pas
2     FileSeek(0, Lof(0))         ; Place le pointeur à la fin du fichier en utilisant le résultat de Lof()
3     WriteAsciiCharacter(0,65)
4     WriteStringN(0, "... une autre ligne à la fin du fichier.")
```

```
5     CloseFile(0)
6   EndIf
7   If ReadFile(0, "Test.txt")
      ; Si le fichier peut être
      lu , on continue...
8     While Eof(0) = 0
      ; Boucle tant que la fin
      du fichier n'est pas
      atteinte. (Eof = 'End Of
      File')
9       Debug
      ReadAsciiCharacter(0)
      ; Affiche ligne par ligne
      le contenu du fichier
10    Wend
11    CloseFile(0)
      ; Ferme le fichier
      précédemment ouvert
12  Else
13    MessageRequester("Information","Impossible
      d'ouvrir le fichier!")
14  EndIf
```

Voir aussi

WriteAsciiCharacter() , ReadUnicodeCharacter() , ReadCharacter() , OpenFile() , ReadFile()

OS Supportés

Tous

95.14 ReadByte

Syntaxe

```
Resultat.b =
    ReadByte(#Fichier)
```

Description

Lit un 'byte' (1 octet signé).

Arguments

#Fichier Le fichier à utiliser.

Valeur de retour

Renvoie l'octet lu ou zéro si une erreur s'est produite.

Exemple

```
1  If CreateFile(0,
    "Test.txt")  ; Ouvre un
    fichier existant ou en
    crée un nouveau s'il
    n'existait pas
2    FileSeek(0, Lof(0))
            ; Place le
    pointeur à la fin du
    fichier en utilisant le
    résultat de Lof()
3    WriteByte(0,65)
4    WriteStringN(0, "... une
    autre ligne à la fin du
    fichier.")
5    CloseFile(0)
6  EndIf
7  If ReadFile(0, "Test.txt")
    ; Si le fichier peut être
    lu , on continue...
8    While Eof(0) = 0
    ; Boucle tant que la fin
    du fichier n'est pas
    atteinte. (Eof = 'End Of
    File')
9       Debug ReadByte(0)
    ; Affiche ligne par ligne
    le contenu du fichier
10   Wend
11   CloseFile(0)
    ; Ferme le fichier
    précédemment ouvert
12 Else
13   MessageRequester("Information","Impossible
    d'ouvrir le fichier!")
14 EndIf
```

Voir aussi

WriteByte() , OpenFile() , ReadFile()

OS Supportés

Tous

95.15 ReadCharacter

Syntaxe

```
Resultat.c =
    ReadCharacter(#Fichier [,
    Format]))
```

Description

Lit un caractère.

Arguments

#**Fichier** Le fichier à utiliser.

Format (optionnel) Le format du caractère à lire. Peut être l'une des valeurs suivantes :

```
#PB_Ascii   : 1 octet non
  signé.
#PB_Unicode: 2 octets non
  signés (par défaut en
  mode unicode
).
#PB_UTF8    : entre 1 et 4
  octets non signés.
```

Valeur de retour

Renvoie le caractère lu ou zéro si une erreur s'est produite.

Exemple

```
If CreateFile(0,
 "Test.txt")  ; Ouvre un
 fichier existant ou en
 crée un nouveau s'il
 n'existait pas
 FileSeek(0, Lof(0))
        ; Place le
 pointeur à la fin du
 fichier en utilisant le
 résultat de Lof()
 WriteCharacter(0,65)
 WriteStringN(0, "... une
 autre ligne à la fin du
 fichier.")
 CloseFile(0)
EndIf
If ReadFile(0, "Test.txt")
 ; Si le fichier peut être
 lu , on continue...
 While Eof(0) = 0
 ; Boucle tant que la fin
 du fichier n'est pas
 atteinte. (Eof = 'End Of
 File')
   Debug ReadCharacter(0)
 ; Affiche ligne par ligne
 le contenu du fichier
 Wend
 CloseFile(0)
 ; Ferme le fichier
 précédemment ouvert
Else
 MessageRequester("Information","Impossible
 d'ouvrir le fichier!")
EndIf
```

Voir aussi

WriteCharacter() , ReadAsciiCharacter() , ReadUnicodeCharacter() , OpenFile() , ReadFile()

OS Supportés

Tous

95.16 ReadDouble

Syntaxe

```
Resultat.d =
    ReadDouble(#Fichier)
```

Description

Lit un nombre à virgule 'double' (8 octets).

Arguments

#**Fichier** Le fichier à utiliser.

Valeur de retour

Renvoie le double lu ou zéro si une erreur s'est produite.

Exemple

```
1  If CreateFile(0,
    "Test.txt")  ; Ouvre un
    fichier existant ou en
    crée un nouveau s'il
    n'existait pas
2    FileSeek(0, Lof(0))
    ; Place le pointeur à la
    fin du fichier en
    utilisant le résultat de
    Lof()
3    WriteDouble(0,123456789.123456789)
4    WriteStringN(0, "... une
    autre ligne à la fin du
    fichier.")
5    CloseFile(0)
6  EndIf
7  If ReadFile(0, "Test.txt")
    ; Si le fichier peut être
    lu , on continue...
8    While Eof(0) = 0
    ; Boucle tant que la fin
    du fichier n'est pas
    atteinte. (Eof = 'End Of
    File')
9       Debug ReadDouble(0)
    ; Affiche ligne par ligne
    le contenu du fichier
```

```
10      Wend
11      CloseFile(0)
        ; Ferme le fichier
        précédemment ouvert
12    Else
13      MessageRequester("Information","Impossible
        d'ouvrir le fichier!")
14    EndIf
```

Voir aussi

WriteDouble() , OpenFile() , ReadFile()

OS Supportés

Tous

95.17 ReadFile

Syntaxe

```
Resultat = ReadFile(#Fichier,
    NomFichier$ [, Options])
```

Description

Ouvre un fichier pour des opérations de lecture seule.

Arguments

#Fichier Le fichier à utiliser.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

NomFichier$ Le nom et le chemin d'accès du fichier.
Si le nom de fichier ne contient pas de chemin complet, il est interprété par rapport au répertoire courant .

Options (optionnel) Peut être une combinaison de :

```
#PB_File_SharedRead : Le
 fichier ouvert par un
 processus peut être lu
 par un autre processus
 (Windows uniquement).
#PB_File_SharedWrite: Le
 fichier ouvert par un
 processus peut être
 écrit par un autre
 processus (Windows
 uniquement).
#PB_File_NoBuffering: Le
 système interne de mise
 en mémoire tampon de
 PureBasic sera désactivé
 pour ce fichier.
```

```
                                    FileBuffersSize()
 ne peut pas être utilisé
   sur ce fichier.
```

combiné avec l'une des valeurs suivantes (les options suivantes affectent le comportement de WriteString() , WriteStringN() , ReadString() , ReadCharacter() et WriteCharacter()) :

```
#PB_Ascii   : Toute
 opération de
 lecture/écriture des
 chaînes de caractères
 utilisera le mode ASCII
#PB_UTF8    : Toute
 opération de
 lecture/écriture des
 chaînes de caractères
 utilisera le mode UTF-8
 (Par défaut).
#PB_Unicode: Toute
 opération de
 lecture/écriture des
 chaînes de caractères
 utilisera le mode
 Unicode.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé comme paramètre #Fichier alors le nouveau numéro généré est renvoyé en cas de succès.

Remarques

Pour ouvrir un fichier en lecture et en écriture, utilisez la fonction OpenFile() .
Pour créer un nouveau fichier vide, utilisez la fonction CreateFile() .

Exemple

```
1  If CreateFile(0,
    "Test.txt")  ; Crée un
    nouveau fichier vide ou
    recrée une fichier vide
    s'il existe déjà
2   FileSeek(0, Lof(0))
           ; Place le
    pointeur à la fin du
    fichier en utilisant le
    résultat de Lof()
3     WriteStringN(0, "...
    une autre ligne à la fin
    du fichier.")
4   CloseFile(0)
```

```
5   EndIf
6   If ReadFile(0, "Test.txt")
        ; Si le fichier peut
    être lu , on continue...
7     While Eof(0) = 0
                ; Boucle tant
    que la fin du fichier
    n'est pas atteinte. (Eof =
    'End Of File')
8       Debug ReadString(0)
          ; Affiche du fichier
9     Wend
10    CloseFile(0)
                  ; Ferme le
    fichier précédemment créé
    ou ouvert
11  Else
12    MessageRequester("Information","Impossible
    d'ouvrir le fichier!")
13  EndIf
```

Voir aussi

OpenFile() , CreateFile() , CloseFile()

OS Supportés

Tous

95.18 ReadFloat

Syntaxe

```
Resultat.f =
    ReadFloat(#Fichier)
```

Description

Lit un nombre à virgule 'float' (4 octets).

Arguments

#**Fichier** Le fichier à utiliser.

Valeur de retour

Renvoie le nombre à virgule flottante en simple précision ou zéro si une erreur s'est produite.

Exemple

```
1   If CreateFile(0,
    "Test.txt")  ; Ouvre un
    fichier existant ou en
    crée un nouveau s'il
    n'existait pas
```

```
2     FileSeek(0, Lof(0))
              ; Place le
    pointeur à la fin du
    fichier en utilisant le
    résultat de Lof()
3     WriteFloat(0,1234.1234)
4     WriteStringN(0, "... une
    autre ligne à la fin du
    fichier.")
5     CloseFile(0)
6   EndIf
7   If ReadFile(0, "Test.txt")
    ; Si le fichier peut être
    lu , on continue...
8     While Eof(0) = 0
    ; Boucle tant que la fin
    du fichier n'est pas
    atteinte. (Eof = 'End Of
    File')
9        Debug ReadFloat(0)
    ; Affiche ligne par ligne
    le contenu du fichier
10    Wend
11    CloseFile(0)
    ; Ferme le fichier
    précédemment ouvert
12  Else
13    MessageRequester("Information","Impossible
    d'ouvrir le fichier!")
14  EndIf
```

Voir aussi

WriteFloat() , ReadDouble() , OpenFile() , ReadFile()

OS Supportés

Tous

95.19 ReadInteger

Syntaxe

```
Resultat.i =
    ReadInteger(#Fichier)
```

Description

Lit un nombre entier 'integer' (4 octets sur les exécutables 32 bits, ou 8 octets sur les exécutables 64 bits).

Arguments

#**Fichier** Le fichier à utiliser.

Valeur de retour

Renvoie le nombre entier ou zéro si une erreur s'est produite.

Exemple

```
1   If CreateFile(0,
     "Test.txt")  ; Ouvre un
     fichier existant ou en
     crée un nouveau s'il
     n'existait pas
2    FileSeek(0, Lof(0))
            ; Place le
     pointeur à la fin du
     fichier en utilisant le
     résultat de Lof()
3    WriteInteger(0,123456789)
4    WriteStringN(0, "... une
     autre ligne à la fin du
     fichier.")
5    CloseFile(0)
6   EndIf
7   If ReadFile(0, "Test.txt")
     ; Si le fichier peut être
     lu , on continue...
8    While Eof(0) = 0
     ; Boucle tant que la fin
     du fichier n'est pas
     atteinte. (Eof = 'End Of
     File')
9       Debug ReadInteger(0)
     ; Affiche ligne par ligne
     le contenu du fichier
10   Wend
11   CloseFile(0)
     ; Ferme le fichier
     précédemment ouvert
12  Else
13   MessageRequester("Information","Impossible
     d'ouvrir le fichier!")
14  EndIf
```

Voir aussi

WriteInteger() , OpenFile() , ReadFile()

OS Supportés

Tous

95.20 ReadLong

Syntaxe

```
Resultat.l =
   ReadLong(#Fichier)
```

Description

Lit un nombre entier 'long' (4 octets).

Arguments

#**Fichier** Le fichier à utiliser.

Valeur de retour

Renvoie le nombre entier ou zéro si une erreur s'est produite.

Exemple

```
If CreateFile(0, "Test.txt") ; Ouvre un fichier existant ou en crée un nouveau s'il n'existait pas
  FileSeek(0, Lof(0))       ; Place le pointeur à la fin du fichier en utilisant le résultat de Lof()
  WriteLong(0,123456789)
  WriteStringN(0, "... une autre ligne à la fin du fichier.")
  CloseFile(0)
EndIf
If ReadFile(0, "Test.txt")
  ; Si le fichier peut être lu , on continue...
  While Eof(0) = 0
  ; Boucle tant que la fin du fichier n'est pas atteinte. (Eof = 'End Of File')
    Debug ReadLong(0)
  ; Affiche ligne par ligne le contenu du fichier
  Wend
  CloseFile(0)
  ; Ferme le fichier précédemment ouvert
Else
  MessageRequester("Information","Impossible d'ouvrir le fichier!")
EndIf
```

Voir aussi

WriteLong() , OpenFile() , ReadFile()

OS Supportés

Tous

95.21 ReadQuad

Syntaxe

```
Resultat.q =
    ReadQuad(#Fichier)
```

Description

Lit un nombre entier 'quad' (8 octets).

Arguments

#**Fichier** Le fichier à utiliser.

Valeur de retour

Renvoie le nombre entier ou zéro si une erreur s'est produite.

Exemple

```
1   If CreateFile(0,
     "Test.txt")    ; Ouvre un
     fichier existant ou en
     crée un nouveau s'il
     n'existait pas
2    FileSeek(0, Lof(0))
            ; Place le
     pointeur à la fin du
     fichier en utilisant le
     résultat de Lof()
3    WriteQuad(0,123456789)
4    WriteStringN(0, "... une
     autre ligne à la fin du
     fichier.")
5    CloseFile(0)
6   EndIf
7   If ReadFile(0, "Test.txt")
     ; Si le fichier peut être
     lu , on continue...
8    While Eof(0) = 0
     ; Boucle tant que la fin
     du fichier n'est pas
     atteinte. (Eof = 'End Of
     File')
9       Debug ReadQuad(0)
     ; Affiche ligne par ligne
     le contenu du fichier
10   Wend
11   CloseFile(0)
     ; Ferme le fichier
     précédemment ouvert
12  Else
13   MessageRequester("Information","Impossible
     d'ouvrir le fichier!")
14  EndIf
```

Voir aussi

WriteQuad() , OpenFile() , ReadFile()

OS Supportés

Tous

95.22 ReadData

Syntaxe

```
Resultat = ReadData(#Fichier,
    *Memoire, Longueur)
```

Description

Lit le contenu d'un fichier et place les données en mémoire.

Arguments

#Fichier Le fichier à utiliser.

***Memoire** L'adresse mémoire où seront stockées les données du fichier.

Longueur Le nombre d'octets à lire.

Valeur de retour

Renvoie le nombre d'octets effectivement lus dans le fichier.
S'il y a une erreur, le valeur renvoyée est zéro.

Exemple

```
1   Fichier$ =
      OpenFileRequester("Sélectionnez
      un fichier","","Texte
      (.txt)|*.txt|Tous
      (*.*)|*.*",0)
2   If Fichier$
3     If ReadFile(0, Fichier$)
4       length = Lof(0)

      ; Lit la taille en octets
      du fichier
5       *MemoryID =
      AllocateMemory(length)
              ; alloue un bloc
      mémoire de la taille du
      fichier
6       If *MemoryID
7         bytes = ReadData(0,
      *MemoryID, length)    ; Lit
      les données du fichier et
      les place dans le bloc
      mémoire
```

```
8           Debug "Nombre
    d'octets lus: " +
    Str(bytes)
9       EndIf
10      CloseFile(0)
11    EndIf
12  EndIf
```

Voir aussi

WriteData() , OpenFile() , ReadFile()

OS Supportés

Tous

95.23 ReadString

Syntaxe

```
Resultat\$ =
    ReadString(#Fichier [,
    Options [, Longueur]])
```

Description

Lit une chaîne de caractères jusqu'au prochain caractère 'Fin de Ligne' (EOL : End Of Line) ou 'Null' (Les formats des fichiers texte DOS, Unix et Macintosh sont supportés).

Arguments

#**Fichier** Le fichier à utiliser.

Options (optionnel) `#PB_Ascii : Lit le texte en ASCII.`
`#PB_UTF8 : Lit le texte en UTF-8.`
`#PB_Unicode: Lit le texte en UTF-16.`

A combiner avec :

`#PB_File_IgnoreEOL: Ignore le caractère de fin de ligne et la lecture se poursuit jusqu'à ce que la longueur spécifiée soit atteinte ou jusqu'à la fin du fichier.`

Longueur (optionnel) Le nombre de caractères à lire.

Si un caractère de fin de ligne est rencontré avant que la longueur ne soit atteinte, la lecture s'arrête (à moins que l'option #PB_File_IgnoreEOL ait été définie).

Valeur de retour

Renvoie la chaîne lue ou une chaîne vide en cas d'erreur.

Remarques

Si le fichier contient un en-tête BOM (byte order mark), utilisez la fonction ReadStringFormat() .

Exemple

```
If CreateFile(0, "Test.txt")  ; Ouvre un fichier existant ou en crée un nouveau s'il n'existait pas
  FileSeek(0, Lof(0))         ; Place le pointeur à la fin du fichier en utilisant le résultat de Lof()
    WriteStringN(0, "... une autre ligne à la fin du fichier.")
  CloseFile(0)
EndIf
If ReadFile(0, "Test.txt")    ; Si le fichier peut être lu , on continue...
  While Eof(0) = 0            ; Boucle tant que la fin du fichier n'est pas atteinte. (Eof = 'End Of File')
    Debug ReadString(0)       ; Affiche du fichier
  Wend
  CloseFile(0)                ; Ferme le fichier précédemment ouvert
Else
  MessageRequester("Information","Impossible d'ouvrir le fichier!")
EndIf
```

Voir aussi

WriteString() , ReadStringFormat() , OpenFile() , ReadFile()

OS Supportés

Tous

95.24 ReadStringFormat

Syntaxe

```
Resultat =
    ReadStringFormat(#Fichier)
```

Description

Vérifie la présence d'un en-tête BOM (Byte Order Mark) et identifie le format d'encodage des chaînes de caractères. Si un BOM est détecté, le curseur du fichier sera placé juste après le BOM.

Arguments

#**Fichier** Le fichier à utiliser.

Valeur de retour

```
#PB_Ascii   : Aucun BOM
 détecté. Généralement un
 fichier texte standard
#PB_UTF8    : UTF-8 et BOM
 détecté.
#PB_Unicode: UTF-16 (little
 endian) et BOM détecté.
#PB_UTF16BE: UTF-16 (big
 endian) et BOM détecté.
#PB_UTF32   : UTF-32 (little
 endian) et BOM détecté.
#PB_UTF32BE: UTF-32 (big
 endian) et BOM détecté.
```

Les resultats #PB_Ascii, #PB_UTF8 et #PB_Unicode peuvent être utilisés directement avec ReadString() pour lire le texte à partir du fichier. Les autres constantes sont incluses pour faciliter la détection du type de fichier, mais ne sont pas prises en compte nativement par PureBasic.

Remarques

Si le BOM est détecté, le pointeur de fichier sera placé à la fin du BOM.
Si aucun BOM n'est détecté, la position du pointeur de fichier reste inchangé.
Le BOM (Byte Order Mark) est un moyen couramment utilisé pour indiquer le type d'encodage d'un fichier texte. Il est habituellement placé au début du fichier. Néanmoins, ce n'est pas un standard ni une norme, mais une pratique courante. Ainsi, si

aucun BOM n'est détecté au début du fichier, cela ne veut pas forcément dire qu'il s'agit d'un fichier ASCII.
WriteStringFormat() peut être utilisé pour écrire un BOM dans un fichier.
Pour plus d'informations, consulter cet article Wikipedia.
Vous trouverez plus d'informations sur le mode unicode avec PureBasic ici .

Exemple

```
Fichier$ = OpenFileRequester("Sélectionnez un fichier","","Texte (.txt)|*.txt|Tous (*.*)|*.*",0)
If Fichier$
  If ReadFile(0, Fichier$)
    Format=ReadStringFormat(0)
    CloseFile(0)
    Select Format
      Case #PB_Ascii
        Debug "Aucun BOM détecté. Généralement un fichier texte standard"
      Case #PB_UTF8
        Debug "UTF-8 et BOM détecté."
      Case #PB_Unicode
        Debug "UTF-16 (little endian) et BOM détecté."
      Case #PB_UTF16BE
        Debug "UTF-16 (big endian) et BOM détecté."
      Case #PB_UTF32
        Debug "UTF-32 (little endian) et BOM détecté."
      Case #PB_UTF32BE
        Debug "UTF-32 (big endian) et BOM détecté."
      Default
        Debug "Format inconnu"
    EndSelect
  EndIf
EndIf
```

Voir aussi

WriteStringFormat() , ReadString() , OpenFile() , ReadFile()

OS Supportés

Tous

95.25 ReadUnicodeCharacter

Syntaxe

```
Resultat.u =
    ReadUnicodeCharacter(#Fichier)
```

Description

Lit un caractère unicode (2 octets non signés).

Arguments

#**Fichier** Le fichier à utiliser.

Valeur de retour

Renvoie le caractère lu ou zéro en cas d'erreur.

Exemple

```
1   If CreateFile(0,
     "Test.txt")  ; Ouvre un
     fichier existant ou en
     crée un nouveau s'il
     n'existait pas
2    FileSeek(0, Lof(0))
            ; Place le
     pointeur à la fin du
     fichier en utilisant le
     résultat de Lof()
3    WriteUnicodeCharacter(0,142)
4    WriteStringN(0, "... une
     autre ligne à la fin du
     fichier.")
5    CloseFile(0)
6   EndIf
7   If ReadFile(0, "Test.txt")
     ; Si le fichier peut être
     lu , on continue...
8    While Eof(0) = 0
     ; Boucle tant que la fin
     du fichier n'est pas
     atteinte. (Eof = 'End Of
     File')
9       Debug
     ReadUnicodeCharacter(0)
          ; Affiche ligne par
     ligne le contenu du fichier
10   Wend
11   CloseFile(0)
     ; Ferme le fichier
     précédemment ouvert
12  Else
13   MessageRequester("Information","Impossible
     d'ouvrir le fichier!")
14  EndIf
```

Voir aussi

WriteUnicodeCharacter() , ReadAsciiCharacter() , ReadCharacter() , OpenFile() , ReadFile()

OS Supportés

Tous

95.26 ReadWord

Syntaxe

```
Resultat.w =
    ReadWord(#Fichier)
```

Description

Lit une donnée de type 'word' (2 octets signés).

Arguments

#**Fichier** Le fichier à utiliser.

Valeur de retour

Renvoie le nombre lu ou zéro en cas d'erreur.

Exemple

```
1  If CreateFile(0,
     "Test.txt")   ; Ouvre un
     fichier existant ou en
     crée un nouveau s'il
     n'existait pas
2     FileSeek(0, Lof(0))
              ; Place le
     pointeur à la fin du
     fichier en utilisant le
     résultat de Lof()
3     WriteWord(0,142)
4     WriteStringN(0, "... une
     autre ligne à la fin du
     fichier.")
5     CloseFile(0)
6  EndIf
7  If ReadFile(0, "Test.txt")
     ; Si le fichier peut être
     lu , on continue...
8     While Eof(0) = 0
     ; Boucle tant que la fin
     du fichier n'est pas
     atteinte. (Eof = 'End Of
     File')
```

```
9         Debug ReadWord(0)
      ; Affiche ligne par ligne
      le contenu du fichier
10      Wend
11      CloseFile(0)
      ; Ferme le fichier
      précédemment ouvert
12    Else
13      MessageRequester("Information","Impossible
      d'ouvrir le fichier!")
14    EndIf
```

Voir aussi

WriteWord() , OpenFile() , ReadFile()

OS Supportés

Tous

95.27 WriteAsciiCharacter

Syntaxe

```
Resultat =
    WriteAsciiCharacter(#Fichier,
    Valeur.a)
```

Description

Ecrit une valeur de type caractère ascii (1 octet non signé).

Arguments

#Fichier Le fichier à utiliser.

Valeur La valeur du caractère ASCII à écrire.

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi et zéro si elle a échoué.

Remarques

En raison de la mise en mémoire tampon , cette fonction peut renvoyer une valeur avec succès même s'il n'y a pas assez d'espace disponible sur le périphérique de sortie pour l'opération d'écriture. Le fichier doit être ouvert en utilisant une fonction d'écriture compatible (c'est à dire pas avec ReadFile()).

Exemple

```
If CreateFile(0, "Test.txt")  ; Ouvre un fichier existant ou en crée un nouveau s'il n'existait pas
 FileSeek(0, Lof(0))          ; Place le pointeur à la fin du fichier en utilisant le résultat de Lof()
 WriteAsciiCharacter(0,65)
 WriteStringN(0, "... une autre ligne à la fin du fichier.")
 CloseFile(0)
EndIf
If ReadFile(0, "Test.txt")  ; Si le fichier peut être lu , on continue...
 While Eof(0) = 0            ; Boucle tant que la fin du fichier n'est pas atteinte. (Eof = 'End Of File')
    Debug ReadAsciiCharacter(0)  ; Affiche ligne par ligne le contenu du fichier
 Wend
 CloseFile(0)                ; Ferme le fichier précédemment ouvert
Else
 MessageRequester("Information","Impossible d'ouvrir le fichier!")
EndIf
```

Voir aussi

ReadAsciiCharacter() , WriteUnicodeCharacter() , WriteCharacter() , CreateFile() , OpenFile()

OS Supportés

Tous

95.28 WriteByte

Syntaxe

```
Resultat = WriteByte(#Fichier, Valeur.b)
```

Description

Ecrit une valeur de type 'byte' (1 octet signé).

Arguments

#**Fichier** Le fichier à utiliser.
Valeur La valeur de l'octet à écrire.

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi et zéro si elle a échoué.

Remarques

En raison de la mise en mémoire tampon , cette fonction peut retourner une valeur avec succès même s'il n'y a pas assez d'espace disponible sur le périphérique de sortie pour l'opération d'écriture.
Le fichier doit être ouvert en utilisant une fonction d'écriture compatible (c'est à dire pas avec ReadFile()).

Exemple

```
1  If CreateFile(0, "Test.txt")  ; Ouvre un fichier existant ou en crée un nouveau s'il n'existait pas
2   FileSeek(0, Lof(0))          ; Place le pointeur à la fin du fichier en utilisant le résultat de Lof()
3   WriteByte(0,65)
4   WriteStringN(0, "... une autre ligne à la fin du fichier.")
5   CloseFile(0)
6  EndIf
7  If ReadFile(0, "Test.txt")    ; Si le fichier peut être lu , on continue...
8   While Eof(0) = 0             ; Boucle tant que la fin du fichier n'est pas atteinte. (Eof = 'End Of File')
9      Debug ReadByte(0)         ; Affiche ligne par ligne le contenu du fichier
10  Wend
11  CloseFile(0)                 ; Ferme le fichier précédemment ouvert
```

```
12  Else
13    MessageRequester("Information","Impossible
      d'ouvrir le fichier!")
14  EndIf
```

Voir aussi

ReadByte() , CreateFile() , OpenFile()

OS Supportés

Tous

95.29 WriteCharacter

Syntaxe

```
Resultat =
    WriteCharacter(#Fichier,
    Caractere.c [, Format])
```

Description

Ecrit un caractère (1 octet non signé en mode ascii ou 2 octets non signés en mode unicode).

Arguments

#Fichier Le fichier à utiliser.

Caractere.c La valeur du caractère à écrire.

Format (optionnel) Le format du caractère à écrire. Peut être l'une des valeurs suivantes :

```
#PB_Ascii   : 1 octet non
  signé.
#PB_Unicode: 2 octets non
  signés (par défaut, voir
  unicode
).
#PB_UTF8    : entre 1 et 4
  octets non signés.
```

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi ou zéro si elle a échoué.

Remarques

En raison de la mise en mémoire tampon , cette fonction peut retourner une valeur avec succès même s'il n'y a pas assez d'espace disponible sur le périphérique de sortie pour l'opération d'écriture.

Le fichier doit être ouvert en utilisant une fonction d'écriture compatible (c'est à dire pas avec ReadFile()).

Exemple

```
If CreateFile(0, "Test.txt")  ; Ouvre un fichier existant ou en crée un nouveau s'il n'existait pas
 FileSeek(0, Lof(0))          ; Place le pointeur à la fin du fichier en utilisant le résultat de Lof()
 WriteCharacter(0,65)
 WriteStringN(0, "... une autre ligne à la fin du fichier.")
 CloseFile(0)
EndIf
If ReadFile(0, "Test.txt")
 ; Si le fichier peut être lu , on continue...
 While Eof(0) = 0
 ; Boucle tant que la fin du fichier n'est pas atteinte. (Eof = 'End Of File')
    Debug ReadCharacter(0)
      ; Affiche ligne par ligne le contenu du fichier
 Wend
 CloseFile(0)
 ; Ferme le fichier précédemment ouvert
Else
  MessageRequester("Information","Impossible d'ouvrir le fichier!")
EndIf
```

Voir aussi

ReadCharacter() , WriteAsciiCharacter() , WriteUnicodeCharacter() , CreateFile() , OpenFile()

OS Supportés

Tous

95.30 WriteDouble

Syntaxe

```
Resultat =
    WriteDouble(#Fichier,
    Valeur.d)
```

Description

Ecrit une valeur de type 'double' (8 octets).

Arguments

#**Fichier** Le fichier à utiliser.
Valeur La valeur à écrire.

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi et zéro si elle a échoué.

Remarques

En raison de la mise en mémoire tampon , cette fonction peut retourner une valeur avec succès même s'il n'y a pas assez d'espace disponible sur le périphérique de sortie pour l'opération d'écriture.
Le fichier doit être ouvert en utilisant une fonction d'écriture compatible (c'est à dire pas avec ReadFile()).

Exemple

```
1   If CreateFile(0,
     "Test.txt")    ; Ouvre un
     fichier existant ou en
     crée un nouveau s'il
     n'existait pas
2    FileSeek(0, Lof(0))
            ; Place le
     pointeur à la fin du
     fichier en utilisant le
     résultat de Lof()
3    WriteDouble(0,123456789.123456789)
4    WriteStringN(0, "... une
     autre ligne à la fin du
     fichier.")
5    CloseFile(0)
6   EndIf
7   If ReadFile(0, "Test.txt")
     ; Si le fichier peut être
     lu , on continue...
8    While Eof(0) = 0
     ; Boucle tant que la fin
     du fichier n'est pas
     atteinte. (Eof = 'End Of
     File')
9       Debug ReadDouble(0)
     ; Affiche ligne par ligne
     le contenu du fichier
10   Wend
```

```
11      CloseFile(0)
                    ; Ferme le
    fichier précédemment ouvert
12    Else
13      MessageRequester("Information","Impossible
    d'ouvrir le fichier!")
14    EndIf
```

Voir aussi

ReadDouble() , WriteFloat() , CreateFile() , OpenFile()

OS Supportés

Tous

95.31 WriteFloat

Syntaxe

```
Resultat =
    WriteFloat(#Fichier,
    Valeur.f)
```

Description

Ecrit une valeur de type 'float' (4 octets).

Arguments

#Fichier Le fichier à utiliser.

Valeur La valeur à écrire.

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi et zéro si elle a échoué.

Remarques

En raison de la mise en mémoire tampon , cette fonction peut retourner une valeur avec succès même s'il n'y a pas assez d'espace disponible sur le périphérique de sortie pour l'opération d'écriture.
Le fichier doit être ouvert en utilisant une fonction d'écriture compatible (c'est à dire pas avec ReadFile()).

Exemple

```
1   If CreateFile(0,
      "Test.txt")  ; Ouvre un
      fichier existant ou en
      crée un nouveau s'il
      n'existait pas
```

```
2     FileSeek(0, Lof(0))
              ; Place le
    pointeur à la fin du
    fichier en utilisant le
    résultat de Lof()
3     WriteFloat(0,1234.1234)
4     WriteStringN(0, "... une
    autre ligne à la fin du
    fichier.")
5     CloseFile(0)
6   EndIf
7   If ReadFile(0, "Test.txt")
    ; Si le fichier peut être
    lu , on continue...
8     While Eof(0) = 0
    ; Boucle tant que la fin
    du fichier n'est pas
    atteinte. (Eof = 'End Of
    File')
9        Debug ReadFloat(0)
    ; Affiche ligne par ligne
    le contenu du fichier
10    Wend
11    CloseFile(0)
    ; Ferme le fichier
    précédemment ouvert
12  Else
13    MessageRequester("Information","Impossible
    d'ouvrir le fichier!")
14  EndIf
```

Voir aussi

ReadFloat() , WriteDouble() , CreateFile() , OpenFile()

OS Supportés

Tous

95.32 WriteInteger

Syntaxe

```
Resultat =
    WriteInteger(#Fichier,
    Valeur)
```

Description

Ecrit une valeur de type 'integer' (4 octets sur un exécutable 32 bits ou 8 octets sur un exécutable 64 bits).

Arguments

#**Fichier** Le fichier à utiliser.

Valeur La valeur à écrire.

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi et zéro si elle a échoué.

Remarques

En raison de la mise en mémoire tampon , cette fonction peut retourner une valeur avec succès même s'il n'y a pas assez d'espace disponible sur le périphérique de sortie pour l'opération d'écriture.
Le fichier doit être ouvert en utilisant une fonction d'écriture compatible (c'est à dire pas avec ReadFile()).

Exemple

```
1   If CreateFile(0, "Test.txt")    ; Ouvre un fichier existant ou en crée un nouveau s'il n'existait pas
2     FileSeek(0, Lof(0))           ; Place le pointeur à la fin du fichier en utilisant le résultat de Lof()
3     WriteInteger(0,1234)
4     WriteStringN(0, "... une autre ligne à la fin du fichier.")
5     CloseFile(0)
6   EndIf
7   If ReadFile(0, "Test.txt")      ; Si le fichier peut être lu , on continue...
8     While Eof(0) = 0              ; Boucle tant que la fin du fichier n'est pas atteinte. (Eof = 'End Of File')
9       Debug ReadInteger(0)        ; Affiche ligne par ligne le contenu du fichier
10    Wend
11    CloseFile(0)                  ; Ferme le fichier précédemment ouvert
12  Else
13    MessageRequester("Information","Impossible d'ouvrir le fichier!")
14  EndIf
```

Voir aussi

ReadInteger() , CreateFile() , OpenFile()

OS Supportés

Tous

95.33 WriteLong

Syntaxe

```
Resultat =
    WriteLong(#Fichier,
    Valeur.l)
```

Description

Ecrit une valeur de type 'long' (4 octets).

Arguments

#**Fichier** Le fichier à utiliser.

Valeur La valeur à écrire.

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi et zéro si elle a échoué.

Remarques

En raison de la mise en mémoire tampon , cette fonction peut retourner une valeur avec succès même s'il n'y a pas assez d'espace disponible sur le périphérique de sortie pour l'opération d'écriture.
Le fichier doit être ouvert en utilisant une fonction d'écriture compatible (c'est à dire pas avec ReadFile()).

Exemple

```
1  If CreateFile(0,
     "Test.txt")    ; Ouvre un
     fichier existant ou en
     crée un nouveau s'il
     n'existait pas
2    FileSeek(0, Lof(0))
                ; Place le
     pointeur à la fin du
     fichier en utilisant le
     résultat de Lof()
3    WriteLong(0,123456789)
4    WriteStringN(0, "... une
     autre ligne à la fin du
     fichier.")
5    CloseFile(0)
6  EndIf
7  If ReadFile(0, "Test.txt")
       ; Si le fichier peut
     être lu , on continue...
```

```
8     While Eof(0) = 0
                ; Boucle tant
    que la fin du fichier
    n'est pas atteinte. (Eof =
    'End Of File')
9       Debug ReadLong(0)
              ; Affiche ligne par
    ligne le contenu du fichier
10    Wend
11    CloseFile(0)
                  ; Ferme le
    fichier précédemment ouvert
12  Else
13    MessageRequester("Information","Impossible
    d'ouvrir le fichier!")
14  EndIf
```

Voir aussi

ReadLong() , CreateFile() , OpenFile()

OS Supportés

Tous

95.34 WriteData

Syntaxe

```
Resultat =
    WriteData(#Fichier,
    *Memoire, Longueur)
```

Description

Ecrit le contenu d'une zone mémoire dans un fichier.

Arguments

#Fichier Le fichier à utiliser.

***Memoire** L'adresse mémoire des données à écrire dans le fichier.

Longueur Le nombre d'octets à écrire dans le fichier.

Valeur de retour

Renvoie le nombre d'octets qui ont été réellement écrits dans le fichier, ou zéro en cas d'erreur.

Remarques

En raison de la mise en mémoire tampon , cette fonction peut retourner une valeur avec succès même s'il n'y a pas assez

d'espace disponible sur le périphérique de sortie pour l'opération d'écriture.

Exemple

```
*MemoryID = AllocateMemory(1000) ; Alloue un bloc mémoire de 1000 octets
If *MemoryID
  PokeS(*MemoryID, "Enregister ce texte dans le bloc mémoire", -1, #PB_Unicode) ; Ecriture d'une chaîne de caractères dans le bloc mémoire
EndIf
If CreateFile(0, GetHomeDirectory()+"Test.txt")
  ; Création d'un nouveau fichier...
  WriteData(0, *MemoryID, SizeOf(Character)*Len("Enregister ce texte dans le bloc mémoire"))   ; On écrit le texte du bloc mémoire dans le fichier
  CloseFile(0)                  ; Ferme le fichier précédemment ouvert et enregistre les données
Else
  Debug "Impossible de créer le fichier!"
EndIf

If ReadFile(0, GetHomeDirectory()+"Test.txt")
  ; Si le fichier peut être lu , on continue...
  While Eof(0) = 0              ; Boucle tant que la fin du fichier n'est pas atteinte. (Eof = 'End Of File')
    Debug ReadString(0, #PB_Unicode)      ; Affiche du fichier
  Wend
  CloseFile(0)                  ; Ferme le fichier précédemment ouvert
Else
  MessageRequester("Information","Impossible d'ouvrir le fichier!")
EndIf
```

Voir aussi

ReadData() , CreateFile() , OpenFile()

OS Supportés

Tous

95.35 WriteQuad

Syntaxe

```
Resultat =
    WriteQuad(#Fichier,
    Valeur.q)
```

Description

Ecrit une valeur de type 'quad' (8 octets).

Arguments

#**Fichier** Le fichier à utiliser.

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi et zéro si elle a échoué.

Remarques

En raison de la mise en mémoire tampon , cette fonction peut retourner une valeur avec succès même s'il n'y a pas assez d'espace disponible sur le périphérique de sortie pour l'opération d'écriture.
Le fichier doit être ouvert en utilisant une fonction d'écriture compatible (c'est à dire pas avec ReadFile()).

Exemple

```
1  If CreateFile(0,
    "Test.txt")    ; Ouvre un
    fichier existant ou en
    crée un nouveau s'il
    n'existait pas
2    FileSeek(0, Lof(0))
              ; Place le
    pointeur à la fin du
    fichier en utilisant le
    résultat de Lof()
3    WriteQuad(0,123456789999)
4    WriteStringN(0, "... une
    autre ligne à la fin du
    fichier.")
5    CloseFile(0)
6  EndIf
```

```
7   If ReadFile(0, "Test.txt")
        ; Si le fichier peut
    être lu , on continue...
8    While Eof(0) = 0
                ; Boucle tant
    que la fin du fichier
    n'est pas atteinte. (Eof =
    'End Of File')
9       Debug ReadQuad(0)
              ; Affiche ligne par
    ligne le contenu du fichier
10   Wend
11   CloseFile(0)
                    ; Ferme le
    fichier précédemment ouvert
12  Else
13   MessageRequester("Information","Impossible
    d'ouvrir le fichier!")
14  EndIf
```

Voir aussi

ReadQuad() , CreateFile() , OpenFile()

OS Supportés

Tous

95.36 WriteString

Syntaxe

```
Resultat =
    WriteString(#Fichier,
    Texte$ [, Format])
```

Description

Ecrit une chaîne de caractères dans un fichier.

Arguments

#**Fichier** Le fichier à utiliser.

Texte$ Le texte à écrire dans le fichier.

Format (optionnel)

```
      #PB_Ascii   : Ecrit le
    texte en ASCII.
   #PB_UTF8    : Ecrit le
    texte en UTF-8.
   #PB_Unicode: Ecrit le
    texte en UTF-16.
```

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi et zéro si elle a échoué.

Remarques

En raison de la mise en mémoire tampon , cette fonction peut retourner une valeur avec succès même s'il n'y a pas assez d'espace disponible sur le périphérique de sortie pour l'opération d'écriture.
Le fichier doit être ouvert en utilisant une fonction d'écriture compatible (c'est à dire pas avec ReadFile()).
Le caractère 'NUL' de fin de chaîne n'est pas écrit dans le fichier.
Afin d'identifier plus facilement le type d'encodage d'un fichier, utilisez la commande WriteStringFormat() pour écrire un BOM (Byte Order Mark).

Exemple

```
1  If CreateFile(0, "Test.txt")    ; Ouvre un fichier existant ou en crée un nouveau s'il n'existait pas
2    FileSeek(0, Lof(0))           ; Place le pointeur à la fin du fichier en utilisant le résultat de Lof()
3    WriteString(0, "Une ligne.")
4    WriteStringN(0, "... une autre ligne à la fin du fichier.")
5    CloseFile(0)
6  EndIf
7  If ReadFile(0, "Test.txt")      ; Si le fichier peut être lu , on continue...
8    While Eof(0) = 0              ; Boucle tant que la fin du fichier n'est pas atteinte. (Eof = 'End Of File')
9      Debug ReadString(0)         ; Affiche ligne par ligne le contenu du fichier
10   Wend
11   CloseFile(0)                  ; Ferme le fichier précédemment ouvert
12 Else
13   MessageRequester("Information","Impossible d'ouvrir le fichier!")
14 EndIf
```

Voir aussi

ReadString() , WriteStringN() , WriteStringFormat() , CreateFile() , OpenFile()

OS Supportés

Tous

95.37 WriteStringFormat

Syntaxe

```
Resultat =
    WriteStringFormat(#Fichier,
    Format)
```

Description

Ecrit un BOM (Byte Order Mark) à la position courante.

Arguments

#**Fichier** Le fichier à utiliser.

Format #PB_Ascii : N'écrit aucun BOM (fichier texte standard). #PB_UTF8 : UTF-8 avec un BOM. #PB_Unicode: UTF-16 (little endian) avec un BOM. #PB_UTF16BE: UTF-16 (big endian) avec un BOM. #PB_UTF32 : UTF-32 (little endian) avec un BOM. #PB_UTF32BE: UTF-32 (big endian) avec un BOM.

Les constantes #PB_Ascii, #PB_UTF8 et #PB_Unicode correspondent aux options supportées par WriteString() et WriteStringN() . Après avoir placé un BOM, toutes les opérations d'écriture devraient utiliser ce format.

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi et zéro si elle a échoué.

Remarques

En raison de la mise en mémoire tampon , cette fonction peut retourner une valeur

avec succès même s'il n'y a pas assez d'espace disponible sur le périphérique de sortie pour l'opération d'écriture.
Le fichier doit être ouvert en utilisant une fonction d'écriture compatible (c'est à dire pas avec ReadFile()).
Le BOM (Byte Order Mark) est un moyen couramment utilisé pour indiquer le type d'encodage d'un fichier texte. Il est habituellement placé au début du fichier. Néanmoins, ce n'est pas un standard ni une norme, mais une pratique courante. Ainsi, si aucun BOM n'est détecté au début du fichier, cela ne veut pas forcément dire qu'il s'agit d'un fichier ASCII.
ReadStringFormat() peut être utilisé pour detecter le BOM d'un fichier.
Pour plus d'informations, consulter cet article Wikipedia.
Vous trouverez plus d'informations sur le mode unicode avec PureBasic ici .

Exemple

```
1  If CreateFile(0, "Test.txt")          ; Création d'un nouveau fichier texte...
2    WriteStringFormat(0, #PB_Unicode)  ; Fichier au format unicode
3      WriteStringN(0, "Ligne hé hé à ù Ê Ï")  ; Ecriture de quelques signes (suivis du code 'Fin de Ligne')
4    CloseFile(0)                       ; Ferme le fichier précédemment ouvert et enregistre les données
5  Else
6    MessageRequester("Information","Impossible de créer le fichier!")
7  EndIf
8    If ReadFile(0, "Test.txt")
9      Format=ReadStringFormat(0)
10   While Eof(0) = 0                   ; Boucle tant que la fin du fichier n'est pas atteinte. (Eof = 'End Of File')
11     Debug ReadString(0)              ; Affiche ligne par ligne le contenu du fichier
12   Wend
13     CloseFile(0)
14     Select Format
15       Case #PB_Ascii
```

```
16          Debug "Aucun BOM
    détecté. Généralement un
    fichier texte standard"
17        Case #PB_UTF8
18          Debug "UTF-8 et
    BOM détecté."
19        Case #PB_Unicode
20          Debug "UTF-16
    (little endian) et BOM
    détecté."
21        Case #PB_UTF16BE
22          Debug "UTF-16 (big
    endian) et BOM détecté."
23        Case #PB_UTF32
24          Debug "UTF-32
    (little endian) et BOM
    détecté."
25        Case #PB_UTF32BE
26          Debug "UTF-32 (big
    endian) et BOM détecté."
27        Default
28          Debug "Format
    inconnu"
29      EndSelect
30    EndIf
```

Voir aussi

ReadStringFormat() , WriteString() , WriteStringN() , CreateFile() , OpenFile()

OS Supportés

Tous

95.38 WriteStringN

Syntaxe

```
Resultat =
    WriteStringN(#Fichier,
    Texte$ [, Format])
```

Description

Ecrit une chaîne de caractères dans un fichier suivie du code 'Fin de Ligne'.

Arguments

#Fichier Le fichier à utiliser.

Texte$ Le texte à écrire dans le fichier.

Format (optionnel)

```
      #PB_Ascii   : Ecrit le
    texte en ASCII.
   #PB_UTF8    : Ecrit le
    texte en UTF-8.
```

```
#PB_Unicode: Ecrit le
 texte en UTF-16.
```

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi et zéro si elle a échoué.

Remarques

En raison de la mise en mémoire tampon , cette fonction peut retourner une valeur avec succès même s'il n'y a pas assez d'espace disponible sur le périphérique de sortie pour l'opération d'écriture.
Le fichier doit être ouvert en utilisant une fonction d'écriture compatible (c'est à dire pas avec ReadFile()).
Vous trouverez un exemple d'utilisation avec la commande CreateFile() .
Afin d'identifier plus facilement le type d'encodage d'un fichier, utilisez la commande WriteStringFormat() pour écrire un BOM (Byte Order Mark).

Exemple

```
1  If CreateFile(0,
    "Test.txt")    ; Ouvre un
    fichier existant ou en
    crée un nouveau s'il
    n'existait pas
2    FileSeek(0, Lof(0))
              ; Place le
    pointeur à la fin du
    fichier en utilisant le
    résultat de Lof()
3    WriteString(0, "Une
    ligne.")
4    WriteStringN(0, "... une
    autre ligne à la fin du
    fichier.")
5    CloseFile(0)
6  EndIf
7  If ReadFile(0, "Test.txt")
     ; Si le fichier peut
    être lu , on continue...
8    While Eof(0) = 0
              ; Boucle tant
    que la fin du fichier
    n'est pas atteinte. (Eof =
    'End Of File')
9      Debug ReadString(0)
         ; Affiche ligne par
    ligne le contenu du fichier
10   Wend
```

```
11      CloseFile(0)
                    ; Ferme le
    fichier précédemment ouvert
12    Else
13      MessageRequester("Information","Impossible
    d'ouvrir le fichier!")
14    EndIf
```

Voir aussi

ReadString() , WriteString() , WriteStringFormat() , CreateFile() , OpenFile()

OS Supportés

Tous

95.39 WriteUnicodeCharacter

Syntaxe

```
Resultat =
    WriteUnicodeCharacter(#Fichier,
    Valeur.c)
```

Description

Ecrit une valeur de type caractère unicode (2 octets non signés).

Arguments

#Fichier Le fichier à utiliser.

Valeur La valeur du caractère unicode à écrire.

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi et zéro si elle a échoué.

Remarques

En raison de la mise en mémoire tampon , cette fonction peut retourner une valeur avec succès même s'il n'y a pas assez d'espace disponible sur le périphérique de sortie pour l'opération d'écriture.
Le fichier doit être ouvert en utilisant une fonction d'écriture compatible (c'est à dire pas avec ReadFile()).

Exemple

```
If CreateFile(0, "Test.txt")    ; Ouvre un fichier existant ou en crée un nouveau s'il n'existait pas
  FileSeek(0, Lof(0))           ; Place le pointeur à la fin du fichier en utilisant le résultat de Lof()
  WriteString(0, "Une ligne hé hé à.")
  WriteStringN(0, "... une autre ligne à la fin du fichier.")
  CloseFile(0)
EndIf
If ReadFile(0, "Test.txt")      ; Si le fichier peut être lu , on continue...
  While Eof(0) = 0              ; Boucle tant que la fin du fichier n'est pas atteinte. (Eof = 'End Of File')
    Debug ReadString(0)         ; Affiche ligne par ligne le contenu du fichier
  Wend
  CloseFile(0)                  ; Ferme le fichier précédemment ouvert
Else
  MessageRequester("Information","Impossible d'ouvrir le fichier!")
EndIf
```

Voir aussi

ReadUnicodeCharacter() , WriteAsciiCharacter() , WriteCharacter() , CreateFile() , OpenFile()

OS Supportés

Tous

95.40 WriteWord

Syntaxe

```
Resultat = WriteWord(#Fichier, Valeur)
```

Description

Ecrit une valeur de type 'word' (2 octets signés).

Arguments

#**Fichier** Le fichier à utiliser.
Valeur La valeur à écrire.

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi et zéro si elle a échoué.

Remarques

En raison de la mise en mémoire tampon , cette fonction peut retourner une valeur avec succès même s'il n'y a pas assez d'espace disponible sur le périphérique de sortie pour l'opération d'écriture.
Le fichier doit être ouvert en utilisant une fonction d'écriture compatible (c'est à dire pas avec ReadFile()).

Exemple

```
1  If CreateFile(0, "Test.txt")    ; Ouvre un fichier existant ou en crée un nouveau s'il n'existait pas
2   FileSeek(0, Lof(0))           ; Place le pointeur à la fin du fichier en utilisant le résultat de Lof()
3   WriteWord(0, 142)
4   WriteStringN(0, "... une autre ligne à la fin du fichier.")
5   CloseFile(0)
6  EndIf
7  If ReadFile(0, "Test.txt")    ; Si le fichier peut être lu , on continue...
8   While Eof(0) = 0              ; Boucle tant que la fin du fichier n'est pas atteinte. (Eof = 'End Of File')
9     Debug ReadWord(0)           ; Affiche ligne par ligne le contenu du fichier
10  Wend
11  CloseFile(0)                  ; Ferme le fichier précédemment ouvert
```

```
12    Else
13      MessageRequester("Information","Impossible
        d'ouvrir le fichier!")
14    EndIf
```

Voir aussi

ReadWord() , CreateFile() , OpenFile()

OS Supportés

Tous

Chapitre 96

FileSystem

Généralités

La bibliothèque FileSystem vous permet de manipuler les fichiers et répertoires de vos périphériques de sauvegarde (Clé USB, disques durs, etc...). Vous pourrez par exemple examiner le contenu d'un répertoire, en créer un nouveau, etc...
Si vous voulez examiner le contenu d'un répertoire commencez par la commande ExamineDirectory() .
Les chemins des fichiers et des répertoires peuvent être complets ou relatifs. Les chemins relatifs sont interprétés par rapport au répertoire courant du programme. Les fonctions GetCurrentDirectory() et SetCurrentDirectory() peuvent être utilisés pour modifier le répertoire courant.
Des caractères de séparation de chemin d'accès aux fichiers spécifiques aux systèmes d'exploitation sont disponibles : #PS, #NPS, #PS$ ('\') et #NPS$ ('/').
En complément de cette bibliothèque, vous pourrez manipuler des fichiers avec les commandes de la bibliothèque File .
Pour obtenir le nom du programme en cours, utilisez la commande ProgramFilename() de la bibliothèque Process .

OS Supportés

Tous

96.1 CopyDirectory

Syntaxe

```
Resultat =
    CopyDirectory(RepertoireSource$,
    RepertoireDestination$,
    Masque$ [, Mode])
```

Description

Copie un répertoire.

Arguments

RepertoireSource$ Le dossier à copier.

RepertoireDestination$ Le dossier de destination de la copie.

Masque$ Le masque de copie.

Par exemple : "*.*" copie tous les fichiers situés dans le répertoire.
"*.exe" copie seulement les fichiers .exe.
Par défaut, une chaîne Masque$ vide permet la copie de tous les fichiers.

Mode (optionnel) Peut être une combinaison des valeurs suivantes :

```
#PB_FileSystem_Recursive:
 Copie le répertoire et
 tous ses
 sous-répertoires.
#PB_FileSystem_Force     :
 Remplace également les
 fichiers protégés
 (Lecture seule).
```

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi, zéro sinon.

Remarques

Si le répertoire existe déjà, son contenu sera automatiquement écrasé.
La fonction FileSize() peut être utilisée pour déterminer si la cible existe ou non.

Exemple

```
1  Debug
    CopyDirectory("D:\Jeux\MonJeu\",
    "D:\Jeux\Recup\", "",
    #PB_FileSystem_Recursive)
```

Voir aussi

CreateDirectory() , ExamineDirectory() , DeleteDirectory()

OS Supportés

Tous

96.2 CopyFile

Syntaxe

```
Resultat =
    CopyFile(NomFichierSource$,
    NomFichierDestination$)
```

Description

Copie un fichier.

Arguments

NomFichierSource$ Le fichier à copier.

NomFichierDestination$ Le fichier copié.

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi, zéro sinon.

Remarques

Si le répertoire existe déjà son contenu sera automatiquement écrasé.
La fonction FileSize() peut être utilisée pour déterminer si la cible existe ou non.
Si le fichier source et le fichier de destination sont les mêmes alors la copie ne se produit pas et zéro sera renvoyé.

Exemple

```
1   Debug
      CopyFile("D:\PB\Nouveau.pb",
      "E:\Projet.pb)
```

Voir aussi

RenameFile() , DeleteFile() , FileSize() , CreateFile() , OpenFile()

OS Supportés

Tous

96.3 CreateDirectory

Syntaxe

```
Resultat =
    CreateDirectory(NomRepertoire$)
```

Description

Crée un nouveau répertoire.

Arguments

NomRepertoire$ Le nom du nouveau répertoire.

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi, zéro sinon.

Remarques

Cette fonction échoue si le répertoire parent du nouveau dossier n'existe pas.
Pour créer plusieurs niveaux de répertoire, il suffit d'appeler cette fonction pour chacun d'entre-eux.

Exemple

```
1  If CreateDirectory("c:\PB")
2    RunProgram("c:\PB")
3  EndIf
```

Voir aussi

CopyDirectory() , ExamineDirectory() , DeleteDirectory()

OS Supportés

Tous

96.4 DeleteDirectory

Syntaxe

```
Resultat =
    DeleteDirectory(Repertoire$,
    Masque$ [, Mode])
```

Description

Efface un répertoire.

Arguments

Repertoire$ Le dossier à détruire.

Masque$ Le masque de suppression des fichiers contenu dans le répertoire.

Par exemple :
"*.*" détruit tous les fichiers situés dans le répertoire.
"*.exe" détruit seulement les fichiers .exe.
Par défaut, une chaîne Masque$ vide ("") permet la supression de tous les fichiers.

Mode (optionnel) Peut être une combinaison des valeurs suivantes :

```
#PB_FileSystem_Recursive:
 Efface le répertoire et
 tous les
 sous-répertoires.
#PB_FileSystem_Force    :
 Efface également les
 fichiers protégés
 (Lecture seule).
```

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi, zéro sinon.

Exemple

```
1  If FileSize("c:\PB") = -2
2    DeleteDirectory("c:\PB","*.*",
     #PB_FileSystem_Recursive|#PB_FileSystem_Force)
3    Debug "Dossier c:\PB
     supprimé"
4  EndIf
```

Voir aussi

CreateDirectory() , ExamineDirectory() , CopyDirectory()

OS Supportés

Tous

96.5 DeleteFile

Syntaxe

```
Resultat =
   DeleteFile(NomFichier$ [,
   Mode]))
```

Description

Supprime un fichier.

Arguments

NomFichier$ Le fichier à supprimer.

Mode (optionnel) Peut être une des valeurs suivantes :

```
#PB_FileSystem_Force:
 Supprime aussi les
 fichiers qui sont
 protégés (en lecture
 seule).
```

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi, zéro sinon.

Exemple

```
1  If FileSize("c:\PB") = -2
2    DeleteFile("c:\PB\ok.txt")
3    Debug "Fichier supprimé"
4  EndIf
```

Voir aussi

CopyFile() , RenameFile() , FileSize() , CreateFile() , OpenFile()

OS Supportés

Tous

96.6 DirectoryEntryAttributes

Syntaxe

```
Resultat =
    DirectoryEntryAttributes(#Repertoire)
```

Description

Renvoie les attributs du fichier courant dans le #Repertoire examiné par les fonctions ExamineDirectory() et NextDirectoryEntry() .

Arguments

#Repertoire Le répertoire examiné avec ExamineDirectory() .

Valeur de retour

Avec Windows, les attributs sont une combinaison des valeurs suivantes :

```
#PB_FileSystem_Hidden     :
 Fichier caché
#PB_FileSystem_Archive    :
 Fichier inchangé et
 archivé depuis le dernier
 examen
#PB_FileSystem_Compressed:
 Fichier compressé
#PB_FileSystem_Normal     :
 Fichier normal
#PB_FileSystem_ReadOnly   :
 Fichier en lecture seule
#PB_FileSystem_System     :
 Fichier système.
```

Avec Linux ou MacOSX, les attributs sont une combinaison des valeurs suivantes :

```
#PB_FileSystem_Link      :
 Le fichier est un lien
 symbolique
#PB_FileSystem_ReadUser  :
 Permission d'accès en
 lecture pour le
 propriétaire
#PB_FileSystem_WriteUser :
 Permission d'accès en
 écriture pour le
 propriétaire
#PB_FileSystem_ExecUser  :
 Permission d'accès en
 exécution pour le
 propriétaire
#PB_FileSystem_ReadGroup :
 Permission d'accès en
 lecture pour les membres
 du groupe propriétaire
#PB_FileSystem_WriteGroup:
 Permission d'accès en
 écriture pour les membres
 du groupe propriétaire
#PB_FileSystem_ExecGroup :
 Permission d'accès en
 exécution pour les membres
 du groupe propriétaire
#PB_FileSystem_ReadAll   :
 Permission d'accès en
 lecture pour les autres
 utilisateurs
#PB_FileSystem_WriteAll  :
 Permission d'accès en
 écriture pour les autres
 utilisateurs
#PB_FileSystem_ExecAll   :
 Permission d'accès en
 exécution pour les autres
 utilisateurs
```

Remarques

Pour vérifier si un attribut est présent, il suffit d'utiliser l'operateur '&' (And) :

Exemple

```
1  Repertoire$ =
    GetHomeDirectory()  ;
    Liste tous les fichiers et
    les dossiers du répertoire
    racine de l'utilisateur
    qui est actuellement logué
    (Home)
2  If ExamineDirectory(0,
    Repertoire$, "*.*")
```

```
3       While NextDirectoryEntry(0)
4          If DirectoryEntryType(0) = #PB_DirectoryEntry_File
5             Type$ = " [Fichier] "
6             Taille$ = " (Taille : " + DirectoryEntrySize(0) + ")"
7             Attributs = GetFileAttributes(DirectoryEntryName(0))
8             If Attributs & #PB_FileSystem_System
9                Debug "Attribut : Système"
10            EndIf
11            Else
12            Type$ = " [Dossier] "
13            Taille$ = "" ; Un Dossier n'a pas de taille
14            Attributs = DirectoryEntryAttributes(0)
15            If Attributs & #PB_FileSystem_System
16               Debug "Attribut : Système"
17            EndIf
18         EndIf
19         Debug Type$ + DirectoryEntryName(0) + Taille$
20      Wend
21      FinishDirectory(0)
22   EndIf
```

Voir aussi

ExamineDirectory() , NextDirectoryEntry() , DirectoryEntryType() , DirectoryEntryName() , DirectoryEntrySize() , DirectoryEntryDate()

OS Supportés

Tous

96.7 DirectoryEntryDate

Syntaxe

```
Resultat = DirectoryEntryDate(#Repertoire, TypeDate)
```

Description

Renvoie la date du fichier courant dans le #Repertoire examiné par les fonctions

ExamineDirectory() et
NextDirectoryEntry() .

Arguments

#**Repertoire** Le répertoire examiné avec ExamineDirectory() .

TypeDate Peut être une des valeurs suivantes :

```
#PB_Date_Created : Renvoie
 la date de création du
 fichier.
#PB_Date_Accessed: Renvoie
 la date du dernier accès
 au fichier.
#PB_Date_Modified: Renvoie
 la date de la dernière
 modification du fichier.
```

Valeur de retour

La date renvoyée est dans le même format que celui utilisé dans la bibliothèque Date , donc toutes les commandes telles que FormatDate() peuvent être utilisées.

Remarques

Sous Linux et Mac OSX, la date de retour pour #PB_Date_Created est la même que la date de #PB_Date_Modified, parce que la plupart des systèmes de fichiers ne stockent pas une date de création du fichier.

Exemple

```
1  Repertoire$ =
    GetHomeDirectory()  ;
    Liste tous les fichiers et
    les dossiers du répertoire
    racine de l'utilisateur
    qui est actuellement logué
    (Home)
2  If ExamineDirectory(0,
    Repertoire$, "*.*")
3     While
    NextDirectoryEntry(0)
4        If
    DirectoryEntryType(0) =
    #PB_DirectoryEntry_File
5           Type$ = " [Fichier] "
6           Taille$ = " (Taille :
    " + DirectoryEntrySize(0)
    + ")"
7           DateAcces =
    DirectoryEntryDate(0,
    #PB_Date_Accessed)
```

```
8         Else
9           Type$ = " [Dossier] "
10          Taille$ = "" ; Un
    Dossier n'a pas de taille
11          DateAcces =
    DirectoryEntryDate(0,
    #PB_Date_Accessed)
12        EndIf
13
14        Debug Type$ +
    DirectoryEntryName(0) +
    Taille$
15        Debug "Dernier accès le
    : " +
    FormatDate("%dd/%mm/%yyyy",
    DateAcces)
16        Debug ""
17      Wend
18      FinishDirectory(0)
19    EndIf
```

Voir aussi

ExamineDirectory() , NextDirectoryEntry() , DirectoryEntryType() , DirectoryEntryName() , DirectoryEntrySize() , DirectoryEntryAttributes()

OS Supportés

Tous

96.8 DirectoryEntryName

Syntaxe

```
Resultat\$ =
    DirectoryEntryName(#Repertoire)
```

Description

Renvoie le nom du fichier courant dans le #Repertoire examiné par les fonctions ExamineDirectory() et NextDirectoryEntry() .

Arguments

#Repertoire Le répertoire examiné avec ExamineDirectory() .

Valeur de retour

Renvoie le nom du répertoire courant.

Remarques

Les pseudo-répertoires "." et ".." peuvent être renvoyés dans une énumération de répertoire, de sorte qu'ils doivent être filtrés s'ils ne doivent pas être inclus.

Exemple

```
Repertoire$ = GetHomeDirectory()  ; Liste tous les fichiers et les dossiers du répertoire racine de l'utilisateur qui est actuellement logué (Home)
If ExamineDirectory(0, Repertoire$, "*.*")
 While NextDirectoryEntry(0)
    If DirectoryEntryType(0) = #PB_DirectoryEntry_File
       Type$ = " [Fichier] "
       Taille$ = " (Taille : " + DirectoryEntrySize(0) + ")"
    Else
       Type$ = " [Dossier] "
       Taille$ = "" ; Un Dossier n'a pas de taille
    EndIf

    Debug Type$ + DirectoryEntryName(0) + Taille$
 Wend
 FinishDirectory(0)
EndIf
```

Voir aussi

ExamineDirectory() , NextDirectoryEntry() , DirectoryEntryType() , DirectoryEntrySize() , DirectoryEntryAttributes() , DirectoryEntryDate()

OS Supportés

Tous

96.9 DirectoryEntryType

Syntaxe

```
Resultat =
    DirectoryEntryType(#Repertoire)
```

Description

Renvoie le type du fichier courant dans le #Repertoire examiné par les fonctions ExamineDirectory() et NextDirectoryEntry() .

Arguments

#Repertoire Le répertoire examiné avec ExamineDirectory() .

Valeur de retour

```
#PB_DirectoryEntry_File
 : C'est un fichier.
#PB_DirectoryEntry_Directory:
 C'est un répertoire.
```

Exemple

```
Repertoire$ =
  GetHomeDirectory()  ;
  Liste tous les fichiers et
  les dossiers du répertoire
  racine de l'utilisateur
  qui est actuellement logué
  (Home)
If ExamineDirectory(0,
 Repertoire$, "*.*")
  While
 NextDirectoryEntry(0)
    If
 DirectoryEntryType(0) =
 #PB_DirectoryEntry_File
      Type$ = " [Fichier] "
      Taille$ = " (Taille :
 " + DirectoryEntrySize(0)
 + ")"
    Else
      Type$ = " [Dossier] "
      Taille$ = "" ; Un
 Dossier n'a pas de taille
    EndIf

    Debug Type$ +
 DirectoryEntryName(0) +
 Taille$
  Wend
  FinishDirectory(0)
EndIf
```

Voir aussi

ExamineDirectory() , NextDirectoryEntry() , DirectoryEntryName() ,

DirectoryEntrySize() ,
DirectoryEntryAttributes() ,
DirectoryEntryDate()

OS Supportés

Tous

96.10 DirectoryEntrySize

Syntaxe

```
Resultat.q =
    DirectoryEntrySize(#Repertoire)
```

Description

Renvoie la taille en octets du fichier courant dans le #Repertoire examiné par les fonctions ExamineDirectory() et NextDirectoryEntry() .

Arguments

#Repertoire Le répertoire examiné avec ExamineDirectory() .

Valeur de retour

Renvoie la taille de l'entrée du répertoire courant en octets.

Remarques

DirectoryEntrySize() renvoie un quad (8 octets) supportant les tailles de fichier supérieures à 4Go.

Exemple

```
1  Repertoire$ =
     GetHomeDirectory()  ;
     Liste tous les fichiers et
     les dossiers du répertoire
     racine de l'utilisateur
     qui est actuellement logué
     (Home)
2    If ExamineDirectory(0,
     Repertoire$, "*.*")
3      While
     NextDirectoryEntry(0)
4        If
     DirectoryEntryType(0) =
     #PB_DirectoryEntry_File
5          Type$ = " [Fichier] "
6          Taille$ = " (Taille :
     " + DirectoryEntrySize(0)
     + ")"
```

```
7         Else
8           Type$ = " [Dossier] "
9           Taille$ = "" ; Un
      Dossier n'a pas de taille
10        EndIf
11
12        Debug Type$ +
      DirectoryEntryName(0) +
      Taille$
13       Wend
14       FinishDirectory(0)
15    EndIf
```

Voir aussi

ExamineDirectory() , NextDirectoryEntry() , DirectoryEntryType() , DirectoryEntryName() , DirectoryEntryAttributes() , DirectoryEntryDate()

OS Supportés

Tous

96.11 ExamineDirectory

Syntaxe

```
Resultat =
    ExamineDirectory(#Repertoire,
    NomRepertoire$, Filtre$)
```

Description

Examine un répertoire et crée une liste qui peut être ensuite parcourue avec les fonctions NextDirectoryEntry() et DirectoryEntryName() .

Arguments

#Repertoire Le numéro qui identifie le listing de fichier.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

NomRepertoire$ Le dossier à examiner.

Filtre$ Permet de sélectionner quels types de fichiers doivent être retenus.

Par exemple, un 'Filtre$' "*.*" ou "" retiendra tous les fichiers (et sous-dossiers) du répertoire, "*.exe" ne retient que les fichiers dont l'extension est .exe (ou les sous-dossiers se terminant par .exe). Veuillez noter le comportement spécifique de MS Windows quand

vous utilisez une extension de nom de fichier à 3 caractères car les extensions plus longues seront prises en compte aussi. Par exemple, le filtre "*.log" trouvera aussi les fichiers "*.log*" comme le fichier "test.log_1".

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi, zéro sinon.
Si #PB_Any est utilisé pour le paramètre '#Repertoire', le numéro du nouveau répertoire sera renvoyé dans 'Resultat'.

Remarques

Une fois que l'examen du répertoire est terminé, la commande FinishDirectory() doit être appelée pour libérer toutes les ressources associées à ce listing.
Des caractères de séparation de chemin d'accès aux fichiers spécifiques aux systèmes d'exploitation sont disponibles : #PS, #NPS, #PS$ ('\') et #NPS$ ('/').

Exemple

```
1   Repertoire$ =
     GetHomeDirectory()  ;
     Liste tous les fichiers et
     les dossiers du répertoire
     racine de l'utilisateur
     qui est actuellement logué
     (Home)
2   If ExamineDirectory(0,
     Repertoire$, "*.*")
3    While
     NextDirectoryEntry(0)
4       If
     DirectoryEntryType(0) =
     #PB_DirectoryEntry_File
5          Type$ = " [Fichier] "
6          Taille$ = " (Taille :
    " + DirectoryEntrySize(0)
    + ")"
7       Else
8          Type$ = " [Dossier] "
9          Taille$ = "" ; Un
    Dossier n'a pas de taille
10      EndIf
11
12      Debug Type$ +
     DirectoryEntryName(0) +
     Taille$
13   Wend
14   FinishDirectory(0)
15  EndIf
```

Voir aussi

FinishDirectory() , NextDirectoryEntry() , DirectoryEntryType() , DirectoryEntryName() , DirectoryEntrySize() , DirectoryEntryAttributes() , DirectoryEntryDate()

OS Supportés

Tous

96.12 FinishDirectory

Syntaxe

```
FinishDirectory(#Repertoire)
```

Description

Termine l'examen du répertoire commencé par ExamineDirectory() . Ceci permet de libérer les ressources associées au listing du #Répertoire.

Arguments

#Repertoire Le répertoire examiné avec ExamineDirectory() .

Valeur de retour

Aucune.

Exemple

```
1  Repertoire$ =
     GetHomeDirectory()  ;
     Liste tous les fichiers et
     les dossiers du répertoire
     racine de l'utilisateur
     qui est actuellement logué
     (Home)
2    If ExamineDirectory(0,
     Repertoire$, "*.*")
3      While
     NextDirectoryEntry(0)
4        If
     DirectoryEntryType(0) =
     #PB_DirectoryEntry_File
5          Type$ = " [Fichier] "
6          Taille$ = " (Taille :
     " + DirectoryEntrySize(0)
     + ")"
7        Else
8          Type$ = " [Dossier] "
```

```
9           Taille$ = "" ; Un
    Dossier n'a pas de taille
10      EndIf
11
12      Debug Type$ +
    DirectoryEntryName(0) +
    Taille$
13     Wend
14     FinishDirectory(0)
15  EndIf
```

Voir aussi

ExamineDirectory()

OS Supportés

Tous

96.13 GetExtensionPart

Syntaxe

```
Resultat\$ =
    GetExtensionPart(CheminComplet$)
```

Description

Extrait l'extension d'un fichier d'après son chemin complet.

Arguments

CheminComplet$ Le chemin d'accès complet au fichier.

Valeur de retour

Renvoie l'extension du fichier.
Par exemple, si le chemin d'accès complet est "C :\PureBasic\PB.exe", le résultat sera "exe".

Remarques

Pour récupérer le nom du fichier ou son chemin, voir les fonctions GetFilePart() et GetPathPart() .

Exemple

```
1  Repertoire$ =
    GetHomeDirectory()   ;
    Liste tous les fichiers et
    les dossiers du répertoire
    racine de l'utilisateur
    qui est actuellement logué
    (Home)
```

```
2   If ExamineDirectory(0,
    Repertoire$, "*.*")
3     While
    NextDirectoryEntry(0)
4       If
    DirectoryEntryType(0) =
    #PB_DirectoryEntry_File
5         Type$ = " [Fichier] "
6         Taille$ = " (Taille :
    " + DirectoryEntrySize(0)
    + ")"
7       Else
8         Type$ = " [Dossier] "
9         Taille$ = "" ; Un
    Dossier n'a pas de taille
10      EndIf
11
12      Debug Type$ +
    DirectoryEntryName(0) +
    Taille$
13      Debug "Extension : " +
    GetExtensionPart(DirectoryEntryName(0))
14      Debug ""
15    Wend
16    FinishDirectory(0)
17  EndIf
```

Voir aussi

GetFilePart() , GetPathPart()

OS Supportés

Tous

96.14 GetFilePart

Syntaxe

```
Resultat\$ =
    GetFilePart(CheminComplet$
    [, Mode]))
```

Description

Extrait le nom d'un fichier d'un chemin complet.

Arguments

CheminComplet$ Le chemin d'accès complet au fichier.

Mode (optionnel) Il peut s'agir d'une des valeurs suivantes :

```
#PB_FileSystem_NoExtension:
 Obtenir le nom du
 fichier sans son
 extension (le cas
 échéant).
```

Valeur de retour

Renvoie le nom du fichier.
Si, par exemple, la chaîne CheminComplet$ est ”C :\PureBasic\PB.exe”, la chaîne retournée dans Fichier$ sera ”PB.exe”.

Remarques

Pour récupérer l'extension ou le chemin, voir les fonctions GetExtensionPart() et GetPathPart() .

Exemple

```
1  Repertoire$ =
    GetHomeDirectory()  ;
    Liste tous les fichiers et
    les dossiers du répertoire
    racine de l'utilisateur
    qui est actuellement logué
    (Home)
2  If ExamineDirectory(0,
    Repertoire$, "*.*")
3     While
    NextDirectoryEntry(0)
4        If
    DirectoryEntryType(0) =
    #PB_DirectoryEntry_File
5           Type$ = " [Fichier] "
6           Taille$ = " (Taille :
    " + DirectoryEntrySize(0)
    + ")"
7        Else
8           Type$ = " [Dossier] "
9           Taille$ = "" ; Un
    Dossier n'a pas de taille
10       EndIf
11
12       Debug Type$ +
    DirectoryEntryName(0) +
    Taille$
13       Debug "Nom : " +
    GetFilePart(DirectoryEntryName(0))
14       Debug ""
15    Wend
16    FinishDirectory(0)
17 EndIf
```

Voir aussi

GetExtensionPart() , GetPathPart()

OS Supportés

Tous

96.15 GetPathPart

Syntaxe

```
Resultat\$ =
    GetPathPart(CheminComplet$)
```

Description

Extrait le chemin d'un fichier d'un chemin complet.

Arguments

CheminComplet$ Le chemin d'accès complet au fichier.

Valeur de retour

Renvoie le chemin.
Si, par exemple, la chaîne CheminComplet$ est "C :\PureBasic\PB.exe", la chaîne renvoyée dans Chemin$ sera "C :\PureBasic\".

Remarques

Pour récupérer l'extension ou le nom du fichier, voir les fonctions GetExtensionPart() et GetFilePart() .

Exemple

```
1  Repertoire$ =
     GetHomeDirectory()  ;
     Liste tous les fichiers et
     les dossiers du répertoire
     racine de l'utilisateur
     qui est actuellement logué
     (Home)
2  If ExamineDirectory(0,
     Repertoire$, "*.*")
3     While
     NextDirectoryEntry(0)
4        If
     DirectoryEntryType(0) =
     #PB_DirectoryEntry_File
5           Type$ = " [Fichier] "
6           Taille$ = " (Taille :
     " + DirectoryEntrySize(0)
     + ")"
7        Else
8           Type$ = " [Dossier] "
```

```
9           Taille$ = "" ; Un Dossier n'a pas de taille
10          EndIf
11
12          Debug Type$ + DirectoryEntryName(0) + Taille$
13          Debug "Chemin : " + GetPathPart(DirectoryEntryName(0))
14          Debug ""
15       Wend
16       FinishDirectory(0)
17    EndIf
```

Voir aussi

GetExtensionPart() , GetFilePart()

OS Supportés

Tous

96.16 IsDirectory

Syntaxe

```
Resultat =
    IsDirectory(#Repertoire)
```

Description

Teste si un listing est correctement initialisé.

Arguments

#**Repertoire** Le dossier à tester.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.

Voir aussi

ExamineDirectory()

OS Supportés

Tous

96.17 CheckFilename

Syntaxe

```
Resultat =
    CheckFilename(NomFichier$)
```

Description

Vérifie si un nom de fichier ne contient aucun caractère interdit par le système de fichiers.
Par exemple, les caractères '/' et '\' sont interdits sous Windows.

Arguments

NomFichier$ Le nom du fichier à tester sans son chemin.

Valeur de retour

Renvoie une valeur non nulle si le fichier ne contient pas de caractères non valides et zéro si c'est le cas.

Remarques

En fonction des OS, certains nom de fichiers sont interdits. Par exemple, Windows n'autorise pas de créer un fichier portant le nom COM1 jusqu'à COM9, LPT1 jusqu'à LPT9, CON, PRN, AUX, CLOCK$, NUL, KEYBD$, CONFIG$, $Mft, $MftMirr, $LogFile, $Volume, $AttrDef, $Bitmap, $Boot, $BadClus, $Secure, $Upcase, $Extend, $Quota, $ObjId, $Reparse. Et à éviter les noms LST, SCREEN$ et $IDLE$ sur les systèmes anciens.
Pour plus d'informations voir ici, en anglais.

Exemple

```
1  Debug
     CheckFilename("PureBasic.exe")
       ; Affiche 1
2  Debug
     CheckFilename("PureBasic")
           ; Affiche 1
3  Debug
     CheckFilename("\PureBasic.exe")
     ; Affiche 0
```

OS Supportés

Tous

96.18 FileSize

Syntaxe

```
Resultat.q =
    FileSize(NomFichier$)
```

Description

Renvoie la taille d'un fichier.

Arguments

NomFichier$ Le nom du fichier.

Valeur de retour

Renvoie la taille du fichier en octets ou l'une des valeurs suivantes :

```
-1: Fichier inexistant.
-2: Le fichier est en fait
 un répertoire.
```

Exemple

```
1   Procedure.s
     TailleFichier(NomFichier$)
2   Taille =
     FileSize(NomFichier$)
3   Select Taille
4     Case -2
5       Reponse$  = "C'est un
     répertoire"
6     Case -1
7       Reponse$ = "Le fichier
     n'existe pas"
8     Default
9       Reponse$  = "La taille
     du fichier est de " +
     Str(Taille) + " Octets."
10  EndSelect
11  ProcedureReturn Reponse$
12  EndProcedure
13
14  Debug
     TailleFichier(#PB_Compiler_Home)
15  Debug
     TailleFichier(#PB_Compiler_Home
     +
     "Compilers\FichierInconnu.exe")
16  Debug
     TailleFichier(#PB_Compiler_Home
     +
     "Compilers\PureBasic.exe")
```

Voir aussi

DirectoryEntrySize()

OS Supportés

Tous

96.19 GetCurrentDirectory

Syntaxe

```
Resultat\$ =
    GetCurrentDirectory()
```

Description

Renvoie le chemin du répertoire courant de l'application.

Arguments

Aucun.

Valeur de retour

Renvoie le chemin complet du répertoire courant.
Il se termine par un séparateur de répertoire ('\', #PS, #PS$ pour Windows ou '/', #NPS, #NPS$ pour les autres OS).
Cette fonction renvoie une chaîne vide dans le cas très improbable d'un échec.

Remarques

Tous les accès aux fichiers sont relatifs à ce répertoire quand un chemin absolu n'est pas spécifié. SetCurrentDirectory() peut être utilisé pour changer le répertoire courant.

Exemple

```
1  Debug GetCurrentDirectory()
2  Debug GetHomeDirectory()
3  Debug
     GetUserDirectory(#PB_Directory_Documents)
4  Debug
     GetTemporaryDirectory()
5  Debug #PB_Compiler_Home
6  Debug #PB_Compiler_File
7  Debug #PB_Compiler_Filename
8  Debug #PB_Compiler_FilePath
```

Voir aussi

SetCurrentDirectory() ,
GetHomeDirectory() , GetUserDirectory()
GetTemporaryDirectory()

OS Supportés

Tous

96.20 GetHomeDirectory

Syntaxe

```
Resultat\$ =
    GetHomeDirectory()
```

Description

Renvoie le chemin du répertoire racine de l'utilisateur qui est actuellement logué.

Arguments

Aucun.

Valeur de retour

Renvoie le chemin complet du répertoire courant.
Il se termine par un séparateur de répertoire ('\', #PS, #PS$ pour Windows ou '/', #NPS, #NPS$ pour les autres OS).
Cette fonction renvoie une chaîne vide dans le cas très improbable d'un échec.

Remarques

Ce répertoire a les droits de l'utilisateur, donc il est possible d'y lire et ecrire des fichiers. Il est spécifique à chaque utilisateur et peut donc être utile pour stocker des informations concernant uniquement cet utilisateur (fichiers de préférences, plugins etc.).

Exemple

```
1  Debug GetCurrentDirectory()
2  Debug GetHomeDirectory()
3  Debug
     GetUserDirectory(#PB_Directory_Documents)
4  Debug
     GetTemporaryDirectory()
5  Debug #PB_Compiler_Home
6  Debug #PB_Compiler_File
7  Debug #PB_Compiler_Filename
8  Debug #PB_Compiler_FilePath
```

Voir aussi

GetCurrentDirectory() ,
GetTemporaryDirectory() ,
GetUserDirectory()

OS Supportés

Tous

96.21 GetUserDirectory

Syntaxe

```
Resultat\$ =
    GetUserDirectory(Type)
```

Description

Renvoie le chemin du répertoire du type d'annuaire spécifié.

Arguments

Type Le type de répertoire. Peut s'agir de l'une des valeurs suivantes :

```
#PB_Directory_Desktop    :
 Répertoire 'Bureau' de
 l'utilisateur connecté
#PB_Directory_Downloads  :
 Répertoire
 'Téléchargements' de
 l'utilisateur connecté
#PB_Directory_Documents  :
 Répertoire 'Documents'
 de l'utilisateur connecté
#PB_Directory_Pictures   :
 Répertoire 'Images' de
 l'utilisateur connecté
#PB_Directory_Musics     :
 Répertoire 'Musiques' de
 l'utilisateur connecté
#PB_Directory_Videos     :
 Répertoire 'Videos' de
 l'utilisateur connecté
#PB_Directory_Public     :
 Répertoire 'Documents
 Publics' de
 l'utilisateur connecté
#PB_Directory_ProgramData:
 Répertoire des données
 stokées par les
 programmes de
 l'utilisateur connecté
                          Sous
 Linux et OSX, c'est le
 dossier 'home' suivi de
 '/.' pour pouvoir créer
                          un
 répertoire de
 configuration caché par
 l'utilisateur connecté
#PB_Directory_AllUserData:
 Répertoire de données
```

```
    des programmes communs
    (accessible à tous les
    utilisateurs)
  #PB_Directory_Programs    :
    Répertoire  des fichiers
    de programme globaux
    (lecture seule)
```

Valeur de retour

Renvoie le chemin complet du répertoire demandé et terminé par un séparateur d'annuaire ('\', #PS, #PS$ pour Windows ou '/', #NPS, #NPS$ pour les autres OS). Renvoie une chaîne vide si le type n'est pas trouvé.

Exemple

```
1   Debug
      GetUserDirectory(#PB_Directory_Desktop)
2   Debug
      GetUserDirectory(#PB_Directory_Downloads)
3   Debug
      GetUserDirectory(#PB_Directory_Documents)
4   Debug
      GetUserDirectory(#PB_Directory_Pictures)
5   Debug
      GetUserDirectory(#PB_Directory_Videos)
6   Debug
      GetUserDirectory(#PB_Directory_Musics)
7   Debug
      GetUserDirectory(#PB_Directory_Public)
8   Debug
      GetUserDirectory(#PB_Directory_ProgramData)
9   Debug
      GetUserDirectory(#PB_Directory_AllUserData)
10  Debug
      GetUserDirectory(#PB_Directory_Programs)
```

Voir aussi

GetCurrentDirectory() ,
GetHomeDirectory() ,
GetTemporaryDirectory()

OS Supportés

Tous

96.22 GetTemporaryDirectory

Syntaxe

```
Resultat\$ =
    GetTemporaryDirectory()
```

Description

Renvoie le chemin du répertoire temporaire.

Arguments

Aucun.

Valeur de retour

Renvoie le chemin complet et le nom du répertoire temporaire.
Il se termine par un séparateur de répertoire ('\', #PS, #PS$ pour Windows ou '/', #NPS, #NPS$ pour les autres OS).
Cette fonction renvoie une chaîne vide dans le cas très improbable d'un échec.

Remarques

Ce répertoire a les droits de l'utilisateur, donc il est possible d'y lire et ecrire des fichiers. Il est spécifique à chaque utilisateur et peut donc être utile pour stocker des informations concernant uniquement cet utilisateur (fichiers de préférences, plugins etc.).

Exemple

```
1  Debug GetCurrentDirectory()
2  Debug GetHomeDirectory()
3  Debug
     GetUserDirectory(#PB_Directory_Documents)
4  Debug
     GetTemporaryDirectory()
5  Debug #PB_Compiler_Home
6  Debug #PB_Compiler_File
7  Debug #PB_Compiler_Filename
8  Debug #PB_Compiler_FilePath
```

Voir aussi

GetCurrentDirectory() , GetUserDirectory() , GetHomeDirectory()

OS Supportés

Tous

96.23 GetFileDate

Syntaxe

```
Resultat =
    GetFileDate(NomFichier$,
    TypeDate)
```

Description

Renvoie la date d'un fichier.

Arguments

NomFichier$ Le fichier à tester.

TypeDate Le type de date :

```
#PB_Date_Created : Renvoie
 la date de création du
 fichier.
#PB_Date_Accessed: Renvoie
 la date du dernier accès
 au fichier.
#PB_Date_Modified: Renvoie
 la date de la dernière
 modification du fichier.
```

Valeur de retour

La date renvoyée est dans le même format que celui utilisé dans la bibliothèque Date , donc toutes les commandes telles que FormatDate() peuvent être utilisées.

Remarques

Sous Linux et Mac OSX, la date de retour pour #PB_Date_Created est la même que la date de #PB_Date_Modified, parce que la plupart des systèmes de fichiers ne stockent pas une date de création de fichier.

Exemple

```
1  Repertoire$ =
    GetHomeDirectory()  ;
    Liste tous les fichiers et
    les dossiers du répertoire
    racine de l'utilisateur
    qui est actuellement logué
    (Home)
2  If ExamineDirectory(0,
    Repertoire$, "*.*")
3     While
    NextDirectoryEntry(0)
4        If
    DirectoryEntryType(0) =
    #PB_DirectoryEntry_File
5           Type$ = " [Fichier] "
6           Taille$ = " (Taille :
    " + DirectoryEntrySize(0)
    + ")"
7           DateAcces =
    GetFileDate(DirectoryEntryName(0),
    #PB_Date_Accessed)
8        Else
```

```
9           Type$ = " [Dossier] "
10          Taille$ = "" ; Un
    Dossier n'a pas de taille
11          DateAcces =
    DirectoryEntryDate(0,
    #PB_Date_Accessed)
12       EndIf
13
14       Debug Type$ +
    DirectoryEntryName(0) +
    Taille$
15       Debug "Dernier accès le
    : " +
    FormatDate("%dd/%mm/%yyyy",
    DateAcces)
16       Debug ""
17     Wend
18     FinishDirectory(0)
19   EndIf
```

Voir aussi

SetFileDate() , DirectoryEntryDate()

OS Supportés

Tous

96.24 GetFileAttributes

Syntaxe

```
Resultat =
    GetFileAttributes(NomFichier$)
```

Description

Renvoie les attributs d'un fichier.

Arguments

NomFichier$ Le fichier à tester.
Cela peut également être le nom d'un répertoire.

Valeur de retour

Renvoie les attributs du fichier.
Si le fichier n'existe pas ou si les attributs du fichier ne peuvent être lus, la commande renvoie -1.
Sur Windows, les attributs sont une combinaison des valeurs suivantes :

```
#PB_FileSystem_Hidden     :
 Fichier caché.
```

```
#PB_FileSystem_Archive   :
 Fichier inchangé et
 archivé depuis le dernier
 test.
#PB_FileSystem_Compressed:
 Fichier compressé.
#PB_FileSystem_Normal    :
 Fichier normal.
#PB_FileSystem_ReadOnly  :
 Fichier en lecture seule.
#PB_FileSystem_System    :
 Fichier système.
```

Sur Linux ou MacOSX, les attributs sont une combinaison des valeurs suivantes :

```
#PB_FileSystem_Link      :
 Le fichier est un lien
 symbolique
#PB_FileSystem_ReadUser  :
 Permission d'accès en
 lecture pour le
 propriétaire
#PB_FileSystem_WriteUser :
 Permission d'accès en
 écriture pour le
 propriétaire
#PB_FileSystem_ExecUser  :
 Permission d'accès en
 exécution pour le
 propriétaire
#PB_FileSystem_ReadGroup :
 Permission d'accès en
 lecture pour les membres
 du groupe propriétaire
#PB_FileSystem_WriteGroup:
 Permission d'accès en
 écriture pour les membres
 du groupe propriétaire
#PB_FileSystem_ExecGroup :
 Permission d'accès en
 exécution pour les membres
 du groupe propriétaire
#PB_FileSystem_ReadAll   :
 Permission d'accès en
 lecture pour les autres
 utilisateurs
#PB_FileSystem_WriteAll  :
 Permission d'accès en
 écriture pour les autres
 utilisateurs
#PB_FileSystem_ExecAll   :
 Permission d'accès en
 exécution pour les autres
 utilisateurs
```

Remarques

Pour vérifier si un attribut est présent, il suffit d'utiliser l'operateur '&' (And) :

```
1  Attributs =
    GetFileAttributes("C:\Text.txt")
2  If Attributs &
    #PB_FileSystem_Hidden
3    Debug "Fichier caché !"
4  EndIf
```

Exemple

```
1  Attributs =
    GetFileAttributes("c:\autoexec.bat")
2
3  If Attributs = -1
4    Debug "Erreur à la
    lecture des attributs du
    fichier!"
5  Else
6    If Attributs &
    #PB_FileSystem_Hidden
    : texte$ + "H" : Else :
    texte$+"-" : EndIf
7    If Attributs &
    #PB_FileSystem_Archive
    : texte$ + "A" : Else :
    texte$+"-" : EndIf
8    If Attributs &
    #PB_FileSystem_Compressed
    : texte$ + "C" : Else :
    texte$+"-" : EndIf
9    If Attributs &
    #PB_FileSystem_Normal
    : texte$ + "N" : Else :
    texte$+"-" : EndIf
10   If Attributs &
    #PB_FileSystem_ReadOnly
    : texte$ + "R" : Else :
    texte$+"-" : EndIf
11   If Attributs &
    #PB_FileSystem_System
    : texte$ + "S" : Else :
    texte$+"-" : EndIf
12   Debug texte$
13 EndIf
```

Exemple

```
1  Repertoire$ =
    GetHomeDirectory()  ;
    Liste tous les fichiers et
    les dossiers du répertoire
    racine de l'utilisateur
    qui est actuellement logué
    (Home)
2  If ExamineDirectory(0,
    Repertoire$, "*.*")
```

```
3       While NextDirectoryEntry(0)
4         If DirectoryEntryType(0) = #PB_DirectoryEntry_File
5           Type$ = " [Fichier] "
6           Taille$ = " (Taille : " + DirectoryEntrySize(0) + ")"
7           Attributs = GetFileAttributes(DirectoryEntryName(0))
8           If Attributs & #PB_FileSystem_System
9             Debug "Attribut : Système"
10          EndIf
11          Else
12          Type$ = " [Dossier] "
13          Taille$ = "" ; Un Dossier n'a pas de taille
14          Attributs = DirectoryEntryAttributes(0)
15          If Attributs & #PB_FileSystem_System
16            Debug "Attribut : Système"
17          EndIf
18        EndIf
19        Debug Type$ + DirectoryEntryName(0) + Taille$
20      Wend
21      FinishDirectory(0)
22    EndIf
```

Voir aussi

SetFileAttributes() ,
DirectoryEntryAttributes()

OS Supportés

Tous

96.25 NextDirectoryEntry

Syntaxe

```
Resultat = NextDirectoryEntry(#Repertoire)
```

Description

Cette fonction doit être appelée à la suite de ExamineDirectory() . Elle examine à la suite chaque entrée de la liste du répertoire.

Arguments

#**Repertoire** Le dossier à examiner avec ExamineDirectory()

Valeur de retour

Renvoie une valeur non nulle si une nouvelle entrée a été lue ou zéro s'il n'y a pas d'entrée supplémentaire.

Remarques

Le type, le nom, la taille, la date et les attributs du fichier ou sous-répertoire correspondant à l'entrée peuvent être obtenus grâce aux fonctions DirectoryEntryType() , DirectoryEntryName() , DirectoryEntrySize() , DirectoryEntryDate() et DirectoryEntryAttributes() .

Exemple

```
1   Repertoire$ =
     GetHomeDirectory()  ;
     Liste tous les fichiers et
     les dossiers du répertoire
     racine de l'utilisateur
     qui est actuellement logué
     (Home)
2   If ExamineDirectory(0,
     Repertoire$, "*.*")
3     While
     NextDirectoryEntry(0)
4       If
     DirectoryEntryType(0) =
     #PB_DirectoryEntry_File
5         Type$ = " [Fichier] "
6         Taille$ = " (Taille :
     " + DirectoryEntrySize(0)
     + ")"
7         Else
8         Type$ = " [Dossier] "
9         Taille$ = "" ; Un
     Dossier n'a pas de taille
10        Attributs =
     DirectoryEntryAttributes(0)
11        Debug Type$ +
     DirectoryEntryName(0) +
     Taille$
12      EndIf
13    Wend
14    FinishDirectory(0)
15  EndIf
```

Voir aussi

ExamineDirectory() , DirectoryEntryType() , DirectoryEntryName() , DirectoryEntrySize() , DirectoryEntryAttributes() , DirectoryEntryDate()

OS Supportés

Tous

96.26 RenameFile

Syntaxe

```
Resultat =
    RenameFile(AncienNom$,
    NouveauNom$)
```

Description

Renomme ou déplace un fichier.
Renomme un dossier.

Arguments

AncienNom$ L'ancien nom du fichier.

NouveauNom$ Le nouveau nom du fichier.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Comme il n'est pas nécessaire que l'ancien et le nouveau nom de fichier soient dans le même répertoire, cette fonction peut donc être utilisée pour déplacer un fichier d'un répertoire vers un autre ou même d'un lecteur vers un autre.

Exemple

```
1  RenameFile("C:\Temp\Test.txt",
     "C:\Temp\TestNew.txt")
     ; Renomme le fichier
     Test.txt du répertoire Temp
2  RenameFile("C:\Temp\",
     "C:\TempNew\")
                        ;
     Renomme le répertoire Temp
     en TempNew
3
```

```
4  If RenameFile("C:\test.txt",
   "D:\temp\test_backup.txt")
5    Debug "Fichier déplacé et
   renommé avec succès."
6  Else
7    Debug "Echec, le fichier
   ne peut pas être renommé
   et déplacé." ; Echec si
   par exemple le fichier (ou
   le disque) n'existe pas.
8  EndIf
```

Voir aussi

CopyFile() , DeleteFile() , CreateFile() , CopyDirectory()

OS Supportés

Tous

96.27 SetFileDate

Syntaxe

```
Resultat =
    SetFileDate(NomFichier$,
    TypeDate, Date)
```

Description

Change la date d'un fichier.

Arguments

NomFichier$ Le fichier à utiliser.

TypeDate Le type de date à modifier :

```
#PB_Date_Created : Change
  la date de création du
  fichier.
#PB_Date_Accessed: Change
  la date du dernier accès
  au fichier.
#PB_Date_Modified: Change
  la date de la dernière
  modification du fichier.
```

Date La nouvelle date. Ce doit être une valeur de la bibliothèque Date .

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi, zéro sinon.

Remarques

La date est dans le même format que celui utilisé dans la bibliothèque Date , donc toutes les commandes telles que FormatDate() peuvent être utilisées.
Sous Linux et Mac OSX, la date utilisée pour #PB_Date_Created est la même que pour #PB_Date_Modified, parce que la plupart des systèmes de fichiers ne stockent pas une date de création de fichier.

```
SetFileDate("F:\Test.txt",
  #PB_Date_Accessed, Date())
```

Voir aussi

GetFileDate()

OS Supportés

Tous

96.28 SetFileAttributes

Syntaxe

```
Resultat =
    SetFileAttributes(NomFichier$,
    Attributs)
```

Description

Change les attributs d'un fichier.

Arguments

NomFichier$ Le nom du fichier à utiliser. Cela peut-être un répertoire.

Attributs Les nouveaux attributs.
Sur Windows, les attributs sont une combinaison de :

```
#PB_FileSystem_Hidden     :
  Fichier caché.
#PB_FileSystem_Archive    :
  Fichier inchangé et
  archivé depuis le
  dernier test.
#PB_FileSystem_Normal     :
  Fichier normal.
#PB_FileSystem_ReadOnly   :
  Fichier en lecture seule.
#PB_FileSystem_System     :
  Fichier système.
```

Sur Linux ou MacOSX, les attributs sont une combinaison de :

```
#PB_FileSystem_ReadUser  :
 Permission d'accès en
 lecture pour le
 propriétaire
#PB_FileSystem_WriteUser :
 Permission d'accès en
 écriture pour le
 propriétaire
#PB_FileSystem_ExecUser  :
 Permission d'accès en
 exécution pour le
 propriétaire
#PB_FileSystem_ReadGroup :
 Permission d'accès en
 lecture pour les membres
 du groupe propriétaire
#PB_FileSystem_WriteGroup:
 Permission d'accès en
 écriture pour les
 membres du groupe
 propriétaire
#PB_FileSystem_ExecGroup :
 Permission d'accès en
 exécution pour les
 membres du groupe
 propriétaire
#PB_FileSystem_ReadAll   :
 Permission d'accès en
 lecture pour les autres
 utilisateurs
#PB_FileSystem_WriteAll  :
 Permission d'accès en
 écriture pour les autres
 utilisateurs
#PB_FileSystem_ExecAll   :
 Permission d'accès en
 exécution pour les
 autres utilisateurs
```

Remarques

Pour combiner plusieurs attributs, il suffit d'utiliser l'operateur '||' (Ou).

Exemple

```
1  SetFileAttributes("C:\Test.txt",
    #PB_FileSystem_Hidden |
    #PB_FileSystem_ReadOnly)
```

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi, zéro sinon.

Voir aussi

GetFileAttributes()

OS Supportés

Tous

96.29 SetCurrentDirectory

Syntaxe

```
Resultat =
    SetCurrentDirectory(Repertoire$)
```

Description

Change le répertoire courant de l'application.

Arguments

Repertoire$ Le chemin complet vers le nouveau répertoire courant, ou un chemin relatif au répertoire courant.

Valeur de retour

Renvoie une valeur non nulle si le répertoire courant a été changé avec succès, zéro sinon.

Remarques

Tous les accès aux fichiers sont relatifs à ce répertoire quand un chemin absolu n'est pas spécifié. GetCurrentDirectory() peut être utilisé pour récupérer le répertoire courant. Des caractères de séparation de chemin d'accès aux fichiers spécifiques aux systèmes d'exploitation sont disponibles : #PS, #NPS, #PS$ ('\') et #NPS$ ('/').

Exemple

```
1  Debug GetCurrentDirectory()
2  Debug
     SetCurrentDirectory("c:\")
3  Debug GetCurrentDirectory()
```

Voir aussi

GetCurrentDirectory()

OS Supportés

Tous

Chapitre 97

Font

Généralités

Les polices de caractères sont largement utilisées sur les ordinateurs. Elles permettent d'agrémenter les textes avec différentes tailles et formes de polices. Note : En PureBasic, les polices de caractères colorées ne sont pas encore supportées. Il est néanmoins possible de contourner cette limitation en utilisant la commande StartDrawing() et dessiner directement sur la surface souhaitée.

OS Supportés

Tous

97.1 FreeFont

Syntaxe

```
FreeFont(#Police)
```

Description

Libère une police.

Arguments

#**Police** Le numéro de la police à libérer. Si #PB_All est spécifié, toutes les polices restantes seront libérées.

Valeur de retour

Aucune.

Remarques

Toutes les polices restantes sont automatiquement libérées quand le programme se termine.

Voir aussi

LoadFont()

OS Supportés

Tous

97.2 FontID

Syntaxe

```
Resultat = FontID(#Police)
```

Description

Renvoie l'identifiant d'une police dans le système d'exploitation.

Arguments

#**Police** Le numéro d'identification de la police à tester.

Valeur de retour

Renvoie l'identifiant système de la police. Ce résultat est aussi parfois appelé 'Handle'. Jetez un oeil au chapitre Numéros et Identifiants pour plus d'informations.

Exemple

```
1   If OpenWindow(0, 0, 0, 222, 130, "FontID()", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2     ButtonGadget(0, 10, 10, 200, 30, "Cliquez pour changer la police...")
3     Font1 = LoadFont(#PB_Any, "Arial"  ,  8, #PB_Font_Bold)
4     Font2 = LoadFont(#PB_Any, "Verdana", 12, #PB_Font_StrikeOut)
5     UsedFont = 1
6   EndIf
7
8   Repeat
9     Event = WaitWindowEvent()
10
11    If Event = #PB_Event_Gadget
12      If EventGadget() = 0
13        If UsedFont = 1
14          SetGadgetFont(0, FontID(Font2))
```

```
15            UsedFont = 2
16          Else
17            SetGadgetFont(0,
    FontID(Font1))
18            UsedFont = 1
19          EndIf
20        EndIf
21      EndIf
22    Until Event =
      #PB_Event_CloseWindow
```

Voir aussi

IsFont() , LoadFont() , SetGadgetFont() , DrawingFont()

OS Supportés

Tous

97.3 IsFont

Syntaxe

```
Resultat = IsFont(#Police)
```

Description

Teste si une police est correctement initialisée.

Arguments

#**Police** Le numéro de la police à tester.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.

Voir aussi

FontID() , LoadFont()

OS Supportés

Tous

97.4 LoadFont

Syntaxe

```
Resultat = LoadFont(#Police,
    NomPolice$, Taille [,
    Options])
```

Description

Charge une police de caractères.

Arguments

#Police Le numéro de la police. #PB_Any peut être utilisés pour générer automatiquement ce numéro.

NomPolice$ Le nom de la police.

Taille Taille de la police.

Options (optionnel) Peut être une combinaison de :

```
#PB_Font_Bold        : La
 police de caractères
 sera en gras
#PB_Font_Italic      : La
 police de caractères
 sera en italique
#PB_Font_Underline   : La
 police de caractères
 sera soulignée (Windows
 seulement)
#PB_Font_StrikeOut   : La
 police de caractères
 sera barrée (Windows
 seulement)
#PB_Font_HighQuality: La
 police de caractères
 sera en qualité
 supérieure (plus
 lent)(Windows seulement)
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Si une autre police de caractères était déjà chargé avec le même numéro #Police, elle sera automatiquement fermée lors du chargement de la nouvelle.
Sous Windows, le gestionnaire de police de caractères essaiera toujours de trouver une police équivalente à celle demandée, si cette dernière n'est pas disponible. Par exemple, si vous essayez de charger la police "Tim

Now Ronin" et qu'elle n'existe pas, une autre police sera chargée automatiquement, en fonction de sa taille, du style etc. Il est donc peu probable que cette commande échoue.

Exemple

```
1   If OpenWindow(0, 0, 0,
    270, 160, "Chargement
    police...",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2    If LoadFont(1, "Arial",
    24)
3       SetGadgetFont(#PB_Default,
    FontID(1))
4       TextGadget(0, 10, 10,
    250, 40, "Arial 24")
5    EndIf
6    Repeat : Until
    WaitWindowEvent() =
    #PB_Event_CloseWindow
7  EndIf
```

Voir aussi

FreeFont() , FontID()

OS Supportés

Tous

97.5 RegisterFontFile

Syntaxe

```
Resultat =
    RegisterFontFile(Fichier$)
```

Description

Enregistre un fichier de police en vue d'être utilisé avec la commande LoadFont() . Toutes les polices contenues dans le fichier seront alors disponibles.

Arguments

Fichier$ Le fichier contenant les polices. Le fichier doit être au format TrueType.

Valeur de retour

Renvoie une valeur non nulle si le fichier a été enregistré correctement.

Remarques

Le fichier de police est inscrit pour le programme en cours. Cela signifie que la ou les polices ne sont pas accessibles par les autres programmes.
Le fichier est désinscrit lorsque le programme se termine.
Aucun changement dans le système d'exploitation n'est fait par cette commande.
Vous ne pouvez pas utiliser de police enregistrée avec la bibliothèque VectorDrawing.

Exemple

```
1  ; Utilisons une nouvelle police qui a été téléchargée à partir d'Internet dans le répertoire temporaire ...
2  ; Le nom de la police est "ascii" et le fichier est "ascii.ttf"
3  If RegisterFontFile(GetTemporaryDirectory() + "ascii.ttf") ; Nous devons l'enregistrer avant de l'utiliser
4    LoadFont(0, "ascii", 12) ; Maintenant, nous pouvons charger la police, le système d'exploitation la connait
5    SetGadgetFont(0, FontID(0))
6  ...
```

Voir aussi

LoadFont()

Chapitre 98

Ftp

Généralités

Le FTP (File Transfer Protocol) est un moyen de partage de fichiers basé sur le modèle client-serveur en réseau. Cette bibliothèque implémente la partie client du FTP, qui permet de se connecter à un serveur distant et de manipuler les fichiers sur ce serveur comme le téléchargement, l'envoi de fichiers, renommer, naviguer dans l'arborescence des dossiers, etc.).

OS Supportés

Tous

98.1 AbortFTPFile

Syntaxe

```
AbortFTPFile(#FTP)
```

Description

Annule le transfert en cours.

Arguments

#**FTP** La connexion FTP à utiliser.

Valeur de retour

Aucune.

Remarques

Si aucun transfert initié avec SendFTPFile() ou ReceiveFTPFile() n'est en cours, cette fonction n'a aucun effet.

Voir aussi

SendFTPFile() , ReceiveFTPFile() , FTPProgress()

OS Supportés

Tous

98.2 CheckFTPConnection

Syntaxe

```
Resultat =
    CheckFTPConnection(#FTP)
```

Description

Vérifie qu'une connexion FTP est toujours connectée au serveur distant.

Arguments

#**FTP** La connexion FTP à utiliser.

Valeur de retour

Renvoie une valeur non nulle si la connexion est toujours ouverte, zéro si le serveur a fermé la connexion.

Voir aussi

OpenFTP()

OS Supportés

Tous

98.3 CloseFTP

Syntaxe

```
CloseFTP(#FTP)
```

Description

Ferme une connexion FTP.

Arguments

#**FTP** La connexion FTP à utiliser préalablement ouverte avec OpenFTP() . Si #PB_All est spécifié, toutes les connexions FTP restantes seront fermées.

Valeur de retour

Aucune.

Remarques

Libère les ressources associées.
Toutes les connexions FTP restantes sont automatiquement fermées quand le programme se termine.

Voir aussi

OpenFTP() , IsFTP()

OS Supportés

Tous

98.4 CreateFTPDirectory

Syntaxe

```
Resultat =
    CreateFTPDirectory(#FTP,
    Repertoire$)
```

Description

Crée un nouveau répertoire sur le serveur FTP.

Arguments

#FTP La connexion FTP à utiliser.

Repertoire$ Le nom du dossier à créer.
Le nouveau répertoire sera créé dans le répertoire courant (voir GetFTPDirectory() et SetFTPDirectory()).
Il n'est pas possible de spécifier un chemin dans le nom du 'Repertoire$'.

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi, zéro sinon.

Voir aussi

DeleteFTPDirectory() , SetFTPDirectory() , GetFTPDirectory()

OS Supportés

Tous

98.5 DeleteFTPDirectory

Syntaxe

```
Resultat =
    DeleteFTPDirectory(#FTP,
    Repertoire$)
```

Description

Efface un répertoire d'un serveur FTP.

Arguments

#**FTP** La connexion FTP à utiliser.

Repertoire$ Le nom du dossier à supprimer.
Le répertoire doit se trouver dans le répertoire courant (voir GetFTPDirectory() et SetFTPDirectory()). Il n'est pas possible de spécifier un chemin dans le nom du 'Repertoire$'.
Le répertoire doit être vide ou la suppression échouera.

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi, zéro sinon.

Voir aussi

CreateFTPDirectory() , SetFTPDirectory() , GetFTPDirectory()

OS Supportés

Tous

98.6 DeleteFTPFile

Syntaxe

```
Resultat =
    DeleteFTPFile(#FTP,
    Fichier$)
```

Description

Efface un fichier du serveur FTP.

Arguments

#**FTP** La connexion FTP à utiliser.

Fichier$ Le fichier à détruire.
Le fichier doit se trouver dans le répertoire courant (voir GetFTPDirectory() et SetFTPDirectory()).
Il n'est pas possible de spécifier un chemin dans le nom du 'Fichier$'.

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi, zéro sinon.

Voir aussi

SendFTPFile() , SetFTPDirectory() , GetFTPDirectory()

OS Supportés

Tous

98.7 ExamineFTPDirectory

Syntaxe

```
Resultat =
    ExamineFTPDirectory(#FTP)
```

Description

Examine le contenu du répertoire FTP courant.

Arguments

#**FTP** La connexion FTP à utiliser.

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi, zéro sinon.

Remarques

Pour le moment, seuls les serveurs de type unix sont supportés. Le protocole FTP ne spécifie pas formellement comment la liste des fichiers doit être retournée, donc chaque serveur peut potentiellement la renvoyer sous une forme spécifique. Heureusement, la plupart des serveurs FTP dans le monde fonctionnent sous unix/linux, et utilisent la même convention pour lister un répertoire. Ceci dit, il n'est pas garanti que cette commande fonctionne avec tous les serveurs. Elle sera mise à jour au fur et à mesure des besoins pour supporter un plus

grand nombre de serveurs. Si le serveur n'est pas correctement supporté, il est possible d'utiliser FTPDirectoryEntryRaw() pour récupérer l'information brute de chaque élément.
La liste créée peut être ensuite parcourue avec les fonctions NextFTPDirectoryEntry() , FTPDirectoryEntryName() , FTPDirectoryEntryType() , FTPDirectoryEntryAttributes() , FTPDirectoryEntryDate() et FTPDirectoryEntrySize() .
Pour changer le répertoire courant, utilisez SetFTPDirectory() .
Une fois que l'examen du répertoire est terminé, il faut appeler FinishFTPDirectory() pour libérer les ressources associées.

Exemple

```
1   If OpenFTP(0, "ftp.free.fr", "anonymous", "")
2     If ExamineFTPDirectory(0)
3       While NextFTPDirectoryEntry(0)
4         Debug FTPDirectoryEntryName(0)
5       Wend
6       FinishFTPDirectory(0)
7     EndIf
8   Else
9     Debug "Connexion avec ftp.free.fr impossible"
10  EndIf
```

Voir aussi

NextFTPDirectoryEntry() , FinishFTPDirectory()

OS Supportés

Tous

98.8 GetFTPDirectory

Syntaxe

```
Resultat\$ =
    GetFTPDirectory(#FTP)
```

Description

Renvoie le répertoire FTP courant.

Arguments

#**FTP** La connexion FTP à utiliser.

Valeur de retour

Renvoie le répertoire FTP courant relatif à la racine du compte FTP.

Remarques

Pour modifier le répertoire courant, voir SetFTPDirectory() .

Voir aussi

SetFTPDirectory() , ExamineFTPDirectory() , SendFTPFile() , ReceiveFTPFile()

OS Supportés

Tous

98.9 FinishFTPDirectory

Syntaxe

```
FinishFTPDirectory(#FTP)
```

Description

Termine l'énumération préalablement commencée avec ExamineFTPDirectory() .

Arguments

#**FTP** La connexion FTP à utiliser.

Valeur de retour

Aucune.

Remarques

Permet de libérer les ressources allouées lors de l'examen du répertoire.

Voir aussi

ExamineFTPDirectory()

OS Supportés

Tous

98.10 FTPDirectoryEntryAttributes

Syntaxe

```
Resultat =
    FTPDirectoryEntryAttributes(#FTP)
```

Description

Renvoie les attributs d'un fichier.

Arguments

#**FTP** La connexion FTP à utiliser.

Valeur de retour

Renvoie les attributs du fichier pendant un examen avec ExamineFTPDirectory() et NextFTPDirectoryEntry() .
C'est une combinaison de :

```
#PB_FTP_ReadUser   : Droits
 d'accès pour l'utilisateur
#PB_FTP_WriteUser
#PB_FTP_ExecuteUser
#PB_FTP_ReadGroup : Droits
 d'accès pour le groupe
#PB_FTP_WriteGroup
#PB_FTP_ExecuteGroup
#PB_FTP_ReadAll    : Droits
 d'accès pour le reste des
 utilisateurs
#PB_FTP_WriteAll
#PB_FTP_ExecuteAll
```

Remarques

Pour tester si un attribut est présent, il convient d'utiliser '&' (ET binaire) et la constante de l'attribut.

Exemple

```
1  [...]
2
3  FileAttributes =
    FTPDirectoryEntryAttributes(#FTP)
4  If FileAttributes &
    #PB_FTP_ReadUser
5    Debug "Ce fichier à le
    droit de lecture pour
    l'utilisateur"
6  EndIf
```

Voir aussi

ExamineFTPDirectory() ,
NextFTPDirectoryEntry() ,
FTPDirectoryEntryType() ,
FTPDirectoryEntryName() ,
FTPDirectoryEntryDate() ,
FTPDirectoryEntrySize() ,
FTPDirectoryEntryRaw()

OS Supportés

Tous

98.11 FTPDirectoryEntryDate

Syntaxe

```
Resultat =
    FTPDirectoryEntryDate(#FTP)
```

Description

Renvoie la date d'un fichier

Arguments

#**FTP** La connexion FTP à utiliser.

Valeur de retour

Renvoie la date du fichier courant sur le
FTP en cours d'examen avec
ExamineFTPDirectory() et
NextFTPDirectoryEntry() .
La date est dans le même format que celui
utilisé dans la bibliothèque Date , donc
toutes les commandes s'y rapportant
comme FormatDate() sont utilisables.

Voir aussi

ExamineFTPDirectory() ,
NextFTPDirectoryEntry() ,
FTPDirectoryEntryType() ,
FTPDirectoryEntryName() ,
FTPDirectoryEntrySize() ,
FTPDirectoryEntryRaw() ,
FTPDirectoryEntryAttributes()

OS Supportés

Tous

98.12 FTPDirectoryEntryName

Syntaxe

```
Resultat\$ =
    FTPDirectoryEntryName(#FTP)
```

Description

Renvoie le nom d'un fichier.

Arguments

#**FTP** La connexion FTP à utiliser.

Valeur de retour

Renvoie le nom du fichier courant sur le
FTP en cours d'examen avec
ExamineFTPDirectory() et
NextFTPDirectoryEntry() .

Voir aussi

ExamineFTPDirectory() ,
NextFTPDirectoryEntry() ,
FTPDirectoryEntryType() ,
FTPDirectoryEntryDate() ,
FTPDirectoryEntrySize() ,
FTPDirectoryEntryRaw() ,
FTPDirectoryEntryAttributes()

OS Supportés

Tous

98.13 FTPDirectoryEntryType

Syntaxe

```
Resultat =
    FTPDirectoryEntryType(#FTP)
```

Description

Renvoie le type de l'élément courant.

Arguments

#**FTP** La connexion FTP à utiliser.

Valeur de retour

Renvoie le type de l'élément courant sur le
#FTP en cours d'examen avec
ExamineFTPDirectory() et
NextFTPDirectoryEntry() .
Peut être une des valeurs suivantes :

```
#PB_FTP_File     :
 L'élément est un fichier.
#PB_FTP_Directory:
 L'élément est un
 répertoire.
```

Voir aussi

ExamineFTPDirectory() ,
NextFTPDirectoryEntry() ,
FTPDirectoryEntryName() ,
FTPDirectoryEntryDate() ,
FTPDirectoryEntrySize() ,
FTPDirectoryEntryRaw() ,
FTPDirectoryEntryAttributes()

OS Supportés

Tous

98.14 FTPDirectoryEntryRaw

Syntaxe

```
Resultat\$ =
    FTPDirectoryEntryRaw(#FTP)
```

Description

Renvoie la ligne brute de l'élément courant.

Arguments

#**FTP** La connexion FTP à utiliser.

Valeur de retour

Renvoie la ligne d'entrée comme envoyée par le serveur FTP en cours d'examen avec ExamineFTPDirectory() et NextFTPDirectoryEntry() , comme il a été envoyé par le serveur FTP.

Remarques

Cela peut être utile quand le serveur n'est pas correctement supporté par ExamineFTPDirectory() . Les informations concernant le contenu du répertoire pourront quand même être reçues et analysées manuellement.

Exemple

```
1  If OpenFTP(0,
     "ftp.free.fr",
     "anonymous", "")
2    If ExamineFTPDirectory(0)
3      While
     NextFTPDirectoryEntry(0)
4        Debug
     FTPDirectoryEntryRaw(0)
5      Wend
6      ExamineFTPDirectory(0)
7    EndIf
8  Else
9    Debug "Connexion avec
     ftp.free.fr impossible"
10 EndIf
```

Voir aussi

ExamineFTPDirectory() ,
NextFTPDirectoryEntry() ,
FTPDirectoryEntryType() ,
FTPDirectoryEntryName() ,
FTPDirectoryEntryDate() ,
FTPDirectoryEntrySize() ,
FTPDirectoryEntryAttributes()

OS Supportés

Tous

98.15 FTPDirectoryEntrySize

Syntaxe

```
Resultat =
   FTPDirectoryEntrySize(#FTP)
```

Description

Renvoie la taille du fichier courant.

Arguments

#**FTP** La connexion FTP à utiliser.

Valeur de retour

Renvoie la taille en octets du fichier courant
sur le FTP en cours d'examen avec
ExamineFTPDirectory() et
NextFTPDirectoryEntry() ..

Voir aussi

ExamineFTPDirectory() ,
NextFTPDirectoryEntry() ,

FTPDirectoryEntryType() ,
FTPDirectoryEntryName() ,
FTPDirectoryEntryDate() ,
FTPDirectoryEntryRaw() ,
FTPDirectoryEntryAttributes()

OS Supportés

Tous

98.16 FTPProgress

Syntaxe

```
Resultat.q = FTPProgress(#FTP)
```

Description

Renvoie la progression du fichier en cours de transfert FTP.

Arguments

#**FTP** La connexion FTP à utiliser.

Valeur de retour

Renvoie le nombre d'octets qui ont été reçus ou émis, ou l'une des valeurs suivantes :

```
#PB_FTP_Started : Le
 transfert de fichier est
 dans la phase
 d'initialisation.
#PB_FTP_Finished: Le
 transfert de fichier s'est
 terminé correctement.
#PB_FTP_Error   : Le
 transfert de fichier a été
 interrompu car une erreur
 est survenue.
```

Voir aussi

SendFTPFile() , ReceiveFTPFile() , AbortFTPFile()

OS Supportés

Tous

98.17 IsFTP

Syntaxe

```
Resultat = IsFTP(#FTP)
```

Description

Teste si une connexion FTP est correctement initialisée.

Arguments

#**FTP** La connexion FTP à utiliser.

Valeur de retour

Renvoie une valeur non nulle si le nombre donné est une connexion client valide et correctement initialisée.

Remarques

Cette fonction a été conçue pour accepter n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage.
C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.

Voir aussi

OpenFTP() , CloseFTP() , CheckFTPConnection()

OS Supportés

Tous

98.18 NextFTPDirectoryEntry

Syntaxe

```
Resultat =
    NextFTPDirectoryEntry(#FTP)
```

Description

Permet de passer au fichier suivant lors de l'examen d'un répertoire FTP avec ExamineFTPDirectory() .

Arguments

#**FTP** La connexion FTP à utiliser.

Valeur de retour

Renvoie une valeur non nulle si l'entrée suivante est disponible, zéro sinon.

Remarques

Le nom du fichier est disponible avec la commande FTPDirectoryEntryName() . Pour savoir si l'élément est un fichier ou un répertoire, utiliser FTPDirectoryEntryType() .

Voir aussi

ExamineFTPDirectory() , FTPDirectoryEntryType() , FTPDirectoryEntryName() , FTPDirectoryEntryDate() , FTPDirectoryEntrySize() , FTPDirectoryEntryRaw() , FTPDirectoryEntryAttributes()

OS Supportés

Tous

98.19 OpenFTP

Syntaxe

```
Resultat = OpenFTP(#FTP,
    Serveur$, Utilisateur$,
    MotDePasse$ [, Passif [,
    Port]])
```

Description

Ouvre une connexion sur un serveur FTP.

Arguments

#FTP La connexion FTP à utiliser. #PB_Any peut être utilisé pour générer automatiquement ce numéro.

Serveur$ L'URL ou l'adresse du server FTP.

Utilisateur$ Le nom d'utilisateur pour l'authentification sur le serveur.

MotDePasse$ Le mot de passe pour l'authentification sur le serveur.

Passif (optionnel) Active ou désactive le mode passif pour la connexion.

```
#True : Mode passif activé
  (par défaut)
#False: Mode passif
  désactivé
```

Port (optionnel) Le port à utiliser pour la connexion.
La valeur par défaut est le port 21.

Valeur de retour

Renvoie une valeur non nulle si la connexion a été établie correctement, zéro sinon.
Si #PB_Any est utilisé pour le paramètre '#FTP', le numéro de la nouvelle connexion ftp sera renvoyé dans 'Resultat'.

Remarques

Pour les serveurs ftp publics, 'Utilisateur$' sera "anonymous" et le 'MotDePasse$' sera vide.

Exemple

```
1  If OpenFTP(0,
     "ftp.free.fr",
     "anonymous", "")
2    Debug "Connecté avec
     succès"
3  Else
4    Debug "Impossible de se
     connecter à ftp.free.fr"
5  EndIf
```

Voir aussi

CloseFTP() , SetFTPDirectory() , ReceiveFTPFile() , SendFTPFile() , ExamineFTPDirectory()

OS Supportés

Tous

98.20 ReceiveFTPFile

Syntaxe

```
Resultat =
    ReceiveFTPFile(#FTP,
    FichierDistant$, Fichier$
    [, Asynchrone])
```

Description

Télécharge un fichier depuis un serveur FTP.

Arguments

#FTP La connexion FTP à utiliser.

FichierDistant$ Le nom du fichier à télécharger.
Le 'FichierDistant$' doit être dans le répertoire courant (voir

GetFTPDirectory() et SetFTPDirectory()).

Fichier$ Le nom que prend le fichier une fois enregistré en local.
Si le nom de fichier ne comporte pas de chemin d'accès complet, il est interprété par rapport au répertoire courant .
Si le fichier existe, il sera écrasé.

Asynchrone (optionnel) `#True : Le téléchargement sera effectué en arrière-plan. #False: Le programme est bloqué et attend la fin du téléchargement (Par défaut).`

Valeur de retour

Renvoie une valeur non nulle si le fichier a été téléchargé correctement ou le transfert asynchrone a été initialisé correctement, zéro sinon.

Remarques

Le mode 'Asynchrone' permet de lancer le téléchargement en tâche de fond, et FTPProgress() permet de suivre sa progression. Il peut être arrêté avec AbortFTPFile() . Un seul fichier peut être reçu ou envoyé (voir SendFTPFile()) à la fois.

Voir aussi

SendFTPFile() , SetFTPDirectory() , GetFTPDirectory() , FTPProgress() , AbortFTPFile()

OS Supportés

Tous

98.21 RenameFTPFile

Syntaxe

```
Resultat =
    RenameFTPFile(#FTP,
    Fichier$,
    NouveauNomFichier$)
```

Description

Renomme un fichier sur le serveur FTP.

Arguments

#**FTP** La connexion FTP à utiliser.

Fichier$ Le fichier à renommer.
Il doit être dans le répertoire courant (voir GetFTPDirectory() et SetFTPDirectory()).
Il n'est pas possible de spécifier un chemin dans le nom du 'Fichier$'

NouveauNomFichier$ Le fichier renommé.
Il doit être dans le répertoire courant (voir GetFTPDirectory() et SetFTPDirectory()).
Il n'est pas possible de spécifier un chemin dans le nom du 'NouveauNomFichier$'

Valeur de retour

Renvoie une valeur non nulle si le fichier a été renommé avec succès, zéro sinon.

Voir aussi

SendFTPFile() , SetFTPDirectory() , GetFTPDirectory() ,

OS Supportés

Tous

98.22 SendFTPFile

Syntaxe

```
Resultat = SendFTPFile(#FTP,
    Fichier$, FichierDistant$
    [, Asynchrone])
```

Description

Envoie un fichier sur un serveur FTP.

Arguments

#**FTP** La connexion FTP à utiliser.

Fichier$ Le nom du fichier à envoyer.
Si le nom de fichier ne comporte pas de chemin d'accès complet, il est interprété par rapport au répertoire courant .

FichierDistant$ Le nom de fichier distant.
Il doit être dans le répertoire courant (voir GetFTPDirectory() et SetFTPDirectory()).

Asynchrone (optionnel) `#True` : Le téléchargement sera effectué en arrière-plan. `#False`: Le programme est bloqué et attend la fin du téléchargement (Par défaut).

Valeur de retour

Renvoie une valeur non nulle si le fichier a été téléchargé correctement ou le transfert asynchrone a été initialisé correctement, zéro sinon.

Remarques

Le mode 'Asynchrone' permet de lancer le téléchargement en tâche de fond, et FTPProgress() permet de suivre sa progression. Il peut être arrêté avec AbortFTPFile() . Un seul fichier peut être reçu ou envoyé (voir SendFTPFile()) à la fois.

Voir aussi

ReceiveFTPFile() , SetFTPDirectory() , GetFTPDirectory() , FTPProgress() , AbortFTPFile()

OS Supportés

Tous

98.23 SetFTPDirectory

Syntaxe

```
Resultat =
    SetFTPDirectory(#FTP,
    Repertoire$)
```

Description

Change le répertoire courant du FTP, relativement au répertoire courant.

Arguments

#FTP La connexion FTP à utiliser.

Repertoire$ Le nouveau répertoire.

Ce paramètre doit correspondre à un répertoire dans le répertoire courant du FTP.
Le chemins imbriqués ne sont pas autorisés donc pour changer de

plusieurs niveaux de répertoire, il faut appeler cette commande plusieurs fois à la suite.
Utilisez la valeur ".." pour revenir vers le répertoire parent du répertoire courant.

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi, zéro sinon.

Remarques

Pour obtenir le répertoire courant FTP, voir GetFTPDirectory() .

Exemple

```
1   If OpenFTP(0,
     "ftp.free.fr",
     "anonymous", "")
2
3     Debug "Connecté au
    serveur"
4
5     If SetFTPDirectory(0,
    "pub")
6       If SetFTPDirectory(0,
    "linux")
7         Debug "Changement de
    repertoire: '/pub/linux'"
8         Debug
    GetFTPDirectory(0)
9       Else
10        Debug "Changement de
    repertoire impossible:
    '/pub/linux'"
11      EndIf
12    Else
13      Debug "Changement de
    repertoire impossible:
    'pub'"
14    EndIf
15
16  Else
17    Debug "Impossible de se
     connecter à ftp.free.fr"
18  EndIf
```

Voir aussi

GetFTPDirectory() , ExamineFTPDirectory() , SendFTPFile() , ReceiveFTPFile()

OS Supportés

Tous

Chapitre 99

Gadget

Généralités

En PureBasic, un Gadget est un nom générique donné pour tous les composants de l'interface graphique d'un programme : boutons, listes d'éléments, cases à cocher, boîtes à onglets, etc... (Dans d'autres langages, ils sont appelés "controls" ou "widgets")
Cette bibliothèque est indépendante du système d'exploitation utilisé (OS). Elle utilise les composants de l'Interface Graphique Utilisateur (GUI) de chaque système.
Avant d'utiliser les gadgets, normalement vous aurez au préalable ouvert une fenêtre , d'autre part pour la création d'une interface utilisateurs vous pouvez aussi utiliser un menu , une barre d'outils et une barre d'états .
Les fonctions qui créent les nouveaux gadgets renvoient le numéro du nouveau gadget (appelé #Gadget dans cette bibliothèque) dans le cas où la constante #PB_Any a été utilisée pour le créer. Par contre, si c'est une constante choisie par le programmeur pour identifier le gadget (au lieu de #PB_Any) alors les fonctions renvoient l'identifiant du système d'exploitation.
Les identifiants (aussi appelés handles) peuvent être utilisés pour les autres commandes de PureBasic, aussi bien que pour les commandes de l'API Windows comme SendMessage_() etc.
Pour plus d'informations référez vous au chapitre Numéros et identifiants .
Si vous voulez créer des boîtes de dialogue ou même des fenêtres 'GUI' avec le support de mise en page automatique (layout en anglais), jeter un oeil à la bibliothèque Dialog .

OS Supportés

Tous

99.1 AddGadgetColumn

Syntaxe

```
AddGadgetColumn(#Gadget,
    Position, Titre$, Largeur)
```

Description

Ajoute une colonne à un gadget.

Arguments

#Gadget Le gadget à utiliser.

Position Le numéro de colonne où sera insérée la nouvelle colonne.
La première colonne commence à la position 0 (la colonne la plus à gauche), puis on augmente de 1 pour chaque colonne à droite.
Lorsque vous insérez une colonne, toutes les anciennes colonnes qui se trouvent à droite de la nouvelle auront une position augmentée de 1 (mise à jour automatique de leur position).

Titre$ Texte de l'en-tête de la colonne.

Largeur La largeur initiale de la colonne.

Valeur de retour

Aucune.

Remarques

Les gadgets supportant cette commande sont :
- ListIconGadget()
- ExplorerListGadget()

Cette commande permet de paramétrer complètement l'affichage d'un ExplorerListGadget() , soit en retirant les colonnes standards à l'aide de RemoveGadgetColumn() et en ajoutant celles de votre choix.
A noter que la colonne 'Nom des fichiers' (#PB_Explorer_Name) n'a pas à être obligatoirement en première position.
Note 1
Pour rafraîchir l'affichage du gadget après avoir ajouté une colonne, il convient d'utiliser SetGadgetText() .
Note 2
Pour remplir une colonne personnalisée (qui n'est pas parmi les colonnes supportées

nativement par le gadget, voir ci-dessous), utiliser SetGadgetItemText() pour chaque élément, dès qu'un évènement du type #PB_EventType_Change est détecté par le gadget.

Note 3

Les constantes suivantes sont disponibles à la place du Titre$ pour créer une colonne prédéfinie (automatiquement rafraîchie par le gadget) :

```
#PB_Explorer_Name      :
 Affiche le nom du fichier
 (ou répertoire)
#PB_Explorer_Size      :
 Affiche la taille du
 fichier (en Ko)
#PB_Explorer_Type      :
 Affiche le type du fichier
#PB_Explorer_Attributes:
 Affiche les attributs du
 fichier (ou répertoire)
#PB_Explorer_Created   :
 Affiche la date de
 création du fichier (ou
 répertoire)
#PB_Explorer_Modified  :
 Affiche la date de
 dernière modification du
 fichier (ou répertoire)
#PB_Explorer_Accessed  :
 Affiche la date du dernier
 accès au fichier (ou
 répertoire)
```

Exemple

```
1  ;Liste avec icônes - Ajoute
    des colonnes
2  If OpenWindow(0, 0, 0, 400,
    150, "AddGadgetColumn",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
3   ListIconGadget(0, 10, 10,
    380, 100, "Colonne
    standard", 150,
    #PB_ListIcon_GridLines)
4   ButtonGadget(1, 10, 120,
    150, 20, "Ajouter une
    nouvelle colonne")
5   index = 1     ; La
    "colonne standard" a déjà
    l'index 0
6   Repeat
7     Evenement =
    WaitWindowEvent()
8     If Evenement =
    #PB_Event_Gadget
9       If EventGadget() = 1
```

```
10          AddGadgetColumn(0,
    index, "Colonne
    "+Str(index), 80)
11          index + 1
12        EndIf
13      EndIf
14    Until Evenement =
    #PB_Event_CloseWindow
15  EndIf
```

OS Supportés

Tous

99.2 AddGadgetItem

Syntaxe

```
Resultat =
    AddGadgetItem(#Gadget,
    Position, Texte$ [,
    ImageID [, Options]])
```

Description

Ajoute un élément à un gadget.

Arguments

#Gadget Le gadget à utiliser.

Position La position ou l'index du nouvel élément.
Pour ajouter un élément au début, utilisez la valeur 0.
Pour ajouter un élément à la fin de la liste des éléments en cours, utilisez la valeur -1.

N'oubliez pas que lorsque vous insérez un nouvel élément entre deux éléments existants, la position de chaque élément de droite est augmentée de 1.
Pour le MDIGadget() ce paramètre spécifie le numéro de la nouvelle fenêtre fille. PB_Any # peut être utilisé, dans ce cas, la valeur de retour sera le nouveau numéro attribué par PB.

Texte$ Le texte du nouvel élément.

Lorsque vous ajoutez un élément à un ListIconGadget() , ce paramètre peut contenir le texte de plusieurs colonnes séparées par un caractère Chr (10) .

ImageID (optionnel) Le numéro d'une image (icône) à afficher à côté de l'élément inséré.
La taille de l'image doit être standard, c'est à dire de 16x16 pixels.
'ImageID' peut être récupéré facilement grâce à la commande ImageID() .

Options (optionnel) Ce paramètre n'est utilisable qu'avec les gadgets suivants :
TreeGadget()

Ce paramètre spécifie le sous-niveau du nouvel item.
Si le sous-niveau est supérieur à celui de l'item précédent, le nouvel item devient l'enfant de celui-ci.
S'il est inférieur, il sera ajouté après le parent de l'élément précédent.

MDIGadget()

Ce paramètre peut être utilisé pour spécifier les options de la nouvelle fenêtre (voir OpenWindow()).
Les options #PB_Window_Borderless, #PB_Window_Screencentered et #PB_Window_WindowCentered ne sont pas prises en charge pour les fenêtres MDI.

Valeur de retour

La valeur de retour est uniquement valable qu'avec le MDIGadget() . Si #PB_Any a été utilisé à la place de 'Position' lors de l'ajout d'un élément à la MDIGadget() , la valeur de retour est le numéro d'identification de la nouvelle fenêtre MDI.

Remarques

Les gadgets supportant cette commande sont :

```
  - ComboBoxGadget()
: Support de l'ImageID si
   l'option
   #PB_ComboBox_Image est
   spécifiée.
  - EditorGadget()

  - ListViewGadget()

  - ListIconGadget()
: Support de l'ImageID.
  - MDIGadget()
     : ImageID ajoute une
   icône dans la barre de
   titre de la fenêtre fille
   et 'Options', les
   paramètres de la fenêtre.
```

```
- PanelGadget()
  : Support de l'ImageID.
- TreeGadget()
  : Support de l'ImageID.
  Le paramètre 'Options'
  spécifie le nouveau
  sous-niveau.
```

Voir aussi

RemoveGadgetItem() , ClearGadgetItems() , CountGadgetItems() , ComboBoxGadget() , ListIconGadget() , ListViewGadget() , MDIGadget() , PanelGadget() , TreeGadget()

OS Supportés

Tous

99.3 ButtonImageGadget

Syntaxe

```
Resultat =
    ButtonImageGadget(#Gadget,
    X, Y, Largeur, Hauteur,
    ImageID [, Options])
```

Description

Crée un bouton avec image dans la GadgetList en cours.

Arguments

#Gadget Le numéro du nouveau gadget. PB_Any # peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

ImageID Le numéro d'identification de l'image du gadget.
Utiliser la fonction ImageID() pour obtenir ce numéro.
Ce paramètre peut être égal à zéro pour créer un bouton sans image.
La fonction SetGadgetAttribute() peut être utilisée pour modifier l'image plus tard.

Options (optionnel)
#PB_Button_Toggle crée un bouton à bascule (garde son état ON/OFF) qui alterne l'état 'appuyé' et 'normal'.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé à la place de #Gadget alors la valeur de retour est le numéro du gadget, généré automatiquement.

Remarques

```
  - GadgetToolTip()
permet d'ajouter une 'mini
  aide' à ce gadget.

 Les fonctions suivantes
  peuvent être utilisées
  pour contrôler le gadget:
 - GetGadgetAttribute()
avec une des valeurs
  suivantes:
   #PB_Button_Image       :
  Renvoie l'identifiant de
  l'image du bouton, ex.
  ImageID(#ImageRelachee).
   #PB_Button_PressedImage:
  Renvoie l'identifiant de
  l'image du bouton
  lorsqu'il est enfoncé, ex.
  ImageID(#ImageEnfoncee).

 - SetGadgetAttribute()
avec une des valeurs
  suivantes:
   #PB_Button_Image       :
  Change l'image du bouton.
   #PB_Button_PressedImage:
  Change l'image du bouton
  lorsqu'il est enfoncé.

Bouton à bascule: Option
   #PB_Button_Toggle
  - SetGadgetState()
:  Change d'état (1 = pressé,
   0 = normal).
  - GetGadgetState()
:  Renvoie l'état (1 =
   pressé, 0 = normal).
```

Exemple

```
1   If OpenWindow(0, 0, 0, 200,
    60, "ButtonImageGadget",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2    If LoadImage(0,
    #PB_Compiler_Home +
    "/Examples/Sources/Data/PureBasic.bmp")
       ; changez le 2ème
```

```
    paramètre en indiquant le
    chemin/fichier contenant
    votre image
3       ButtonImageGadget(0,
    10, 10, 180, 40,
    ImageID(0))
4    EndIf
5    Repeat : Until
    WaitWindowEvent() =
    #PB_Event_CloseWindow
6  EndIf
```

Voir aussi

GetGadgetState() , SetGadgetState() , GetGadgetAttribute() , SetGadgetAttribute() , ButtonGadget() , ImageID() , EventGadget()

OS Supportés

Tous

99.4 ButtonGadget

Syntaxe

```
Resultat =
    ButtonGadget(#Gadget, X,
    Y, Largeur, Hauteur,
    Texte$ [, Options])
```

Description

Crée un bouton dans la GadgetList en cours.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur Les coordonnées et la position du bouton dans la fenêtre.

Note : Sur OS X, utiliser une hauteur de 25 activera un bouton avec une hauteur fixe, ce qui est couramment utilisé dans les applications OS X. Il activera également l'option #PB_Button_Default.

Texte$ Le texte qui sera affiché dans le bouton.

Options (optionnel) Peut être une combinaison des constantes suivantes :

```
#PB_Button_Right     :
 Aligne le texte du
 bouton à droite (non
 supporté sous OS X).
#PB_Button_Left      :
 Aligne le texte du
 bouton à gauche (non
 supporté sous OS X).
#PB_Button_Default   :
 Définit le bouton comme
 bouton par défaut (sur
 OS X, la hauteur du
 bouton doit être 25).
#PB_Button_MultiLine:
 Affiche le texte sur
 plusieurs lignes s'il
 est trop long (non
 supporté sous OS X).
#PB_Button_Toggle    : Crée
 un bouton de type
 'Toggle' (bascule). Ce
 bouton alterne l'état
 'appuyé' et 'normal'.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

```
  - GadgetToolTip()
 permet d'ajouter une 'mini
   aide' à ce gadget.

  Les commandes suivantes
   peuvent être utilisées
   pour agir sur le gadget:
  - SetGadgetText()
: Change le texte affiché par
   le bouton.
  - GetGadgetText()
: Renvoie le texte affiché
   par le bouton.

  Utile avec les boutons de
   type bascule
   (#PB_Button_Toggle)
  - SetGadgetState()
```

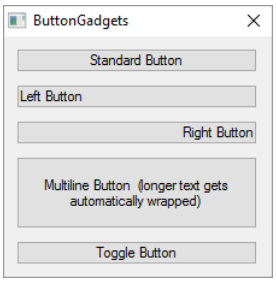
```
:  Changer d'état (1 =
   pressé, 0 = normal).
  - GetGadgetState()
:  Renvoie l'état (1 =
   pressé, 0 = normal).
```

Pour Windows uniquement :

Vous pouvez utiliser le caractère '&' pour souligner une lettre particulière dans le 'Texte$' du bouton : "&Bouton" affichera : Bouton
(Note : Sous Windows XP et suivant, le support des thèmes d'affichage peut empêcher le caractère souligné de s'afficher.)
Vous pouvez associer un raccourci clavier au bouton avec la commande AddKeyboardShortcut() .

Exemple

```
1   ; Démonstration des options possibles pour le gadget bouton
2   ;
3   If OpenWindow(0, 0, 0, 222, 200, "ButtonGadget", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
4    ButtonGadget(0, 10,  10, 200, 20, "Bouton standard")
5    ButtonGadget(1, 10,  40, 200, 20, "Texte aligné à gauche", #PB_Button_Left)
6    ButtonGadget(2, 10,  70, 200, 20, "Texte aligné à droite", #PB_Button_Right)
7    ButtonGadget(3, 10, 100, 200, 60, "Texte sur plusieurs lignes (les textes longs retournent automatiquement à la ligne)", #PB_Button_MultiLine)
8    ButtonGadget(4, 10, 170, 200, 20, "Bouton à bascule", #PB_Button_Toggle)
9    Repeat : Until WaitWindowEvent() = #PB_Event_CloseWindow
10  EndIf
```



Voir aussi

SetGadgetText() , GetGadgetText() , SetGadgetState() , GetGadgetState() , ButtonImageGadget()

OS Supportés

Tous

99.5 CalendarGadget

Syntaxe

```
Resultat =
    CalendarGadget(#Gadget, X,
    Y, Largeur, Hauteur [,
    Date [, Option]])
```

Description

Crée un gadget de type calendrier dans la GadgetList en cours.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Date (optionnel) La date initiale à définir.
La valeur par défaut est la date du jour.

Option (optionnel)
#PB_Calendar_Borderless : Créer un gadget sans bordure (pas supporté par Linux).

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

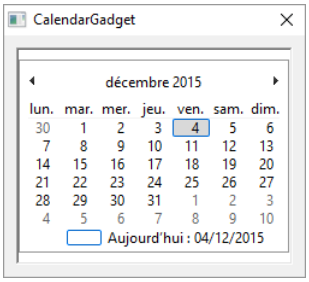
La date utilisée par ce gadget est identique à celle utilisée dans la bibliothèque Date.

```
  - GadgetToolTip()
 permet d'ajouter une 'mini
   aide' à ce gadget.

  Les commandes suivantes
   peuvent être utilisées
   pour agir sur le gadget :
  - SetGadgetState()
: Change la date actuellement
   affichée.
  - GetGadgetState()
: Renvoie la date
   actuellement affichée.

  - SetGadgetItemState()
: Fait apparaître une date
   donnée en gras (Windows
   seulement).
  - GetGadgetItemState()
: Détermine si une date
   donnée est en gras
   (Windows seulement).

  - SetGadgetAttribute()
 avec les attributs suivants:
    #PB_Calendar_Minimum:
   Change la date minimale
   que l'utilisateur peut
   choisir
    #PB_Calendar_Maximum:
   Change la date maximale
   que l'utilisateur peut
   choisir avec ce gadget
                          (Remarque:
   la date sélectionnable
   n'est pas prise en charge
   sous Linux.)

  - GetGadgetAttribute()
 avec les attributs suivants:
    #PB_Calendar_Minimum:
   Renvoie la date minimale
   que l'utilisateur peut
   choisir
    #PB_Calendar_Maximum:
   Renvoie la date maximale
   que l'utilisateur peut
   choisir avec ce gadget
                          (Remarque:
   la date sélectionnable
   n'est pas prise en charge
   sous Linux.)

  - SetGadgetColor()
 et GetGadgetColor()
```

```
avec les constantes
  suivantes comme
  'TypeCouleur':
   #PB_Gadget_BackColor
  : Couleur de fond
   #PB_Gadget_FrontColor
  : Couleur du texte pour
  les jours affichés (non
  pris en charge sur Windows
  Vista +)
   #PB_Gadget_TitleBackColor
  : Couleur du fond du titre
  du mois (non pris en
  charge sur Windows Vista +)
   #PB_Gadget_TitleFrontColor:
  Couleur du texte du titre
  du mois (non pris en
  charge sur Windows Vista +)
   #PB_Gadget_GrayTextColor
  : Couleur du texte pour
  les jours n'appartenant
  pas au mois courant (non
  pris en charge sur Windows
  Vista +)
```

Exemple

```
1  If OpenWindow(0, 0, 0, 250,
    200, "CalendarGadget",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2    CalendarGadget(0, 10, 10,
    230, 180)
3    Repeat
4    Until WaitWindowEvent() =
    #PB_Event_CloseWindow
5  EndIf
```



Voir aussi

SetGadgetState() , GetGadgetState() , SetGadgetItemState() , GetGadgetItemState() , SetGadgetAttribute() , GetGadgetAttribute() , SetGadgetColor() , GetGadgetColor() , DateGadget() , Date() , FormatDate()

OS Supportés

Tous

99.6 CanvasGadget

Syntaxe

```
Resultat =
    CanvasGadget(#Gadget, X,
    Y, Largeur, Hauteur [,
    Options])
```

Description

Crée un canvas dans la GadgetList en cours. Ce gadget offre une surface de dessin sans canal alpha et d'évènements pour la souris et le clavier dans le but de créer facilement des affichages personnalisées.

Arguments

#Gadget Numéro d'identification du canvas.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur Position et dimensions du canvas en pixels.
Les valeurs maximales pour la largeur et la hauteur sont de 16 000 pixels.

Options (optionnel) Combinaison des constantes suivantes :

```
#PB_Canvas_Border    :
 Trace une bordure autour
 du gadget.

#PB_Canvas_ClipMouse:
 Capture automatiquement
 la souris dans la zone
                      du
 gadget tant qu'un bouton
 de la souris est appuyé.
                      (Non
 supporté par OS X et
 Linux gtk3).

#PB_Canvas_Keyboard :
 Reçoit le focus du
 clavier ainsi que les
 évènements clavier.
                      Avec
 cette option, utilisez
 les évènements
 #PB_EventType_Focus
                      et
 #PB_EventType_LostFocus
 pour identifier le
 gadget qui a le focus.
```

```
                        Une
  autre façon de savoir si
  le canvas a le focus,
  est l'utilisation de
  #PB_Canvas_DrawFocus
                        qui
  dessine un rectangle de
  focus chaque fois qu'il
  a le focus.

#PB_Canvas_DrawFocus:
  Dessine un rectangle de
  focus sur le gadget s'il
  a le focus clavier.

#PB_Canvas_Container:
  Permet le support du
  mode conteneur afin d'y
  ajouter des gadgets.
  CloseGadgetList()

                        doit
  être appelé pour revenir
  à la liste de gadgets
  précédente comme tout
  autre conteneur.
                        Sous
  Windows, la transparence
  des gadgets n'est pas
  prise en compte ainsi le
  texte des gadgets
                        suivants
  sera affiché sur un fond
  opaque: CheckBoxGadget,
  FrameGadget,
  HyperlinkGadget,
                        OptionGadget,
  TextGadget et
  TrackBarGadget.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès ou une valeur nulle en cas d'échec. Si #PB_Any a été utilisé alors la valeur retournée est son numéro d'dentification en cas de succès.

Remarques

- Le CanvasGadget() n'ayant pas de canal alpha (transparence), les modes #PB_2DDrawing_AlphaChannel de la fonction DrawingMode() n'aura aucun effet et le mode #PB_2DDrawing_AllChannels sera équivalent à #PB_2DDrawing_Default
- Le gadget apparaît comme un simple fond blanc.

- Utiliser la commande CanvasOutput() ou la commande CanvasVectorOutput() pour dessiner sur le canvas.
- Le contenu reste persistant jusqu'à ce qu'il soit effacé par une opération de dessin.
- Il n'est pas nécessaire de redessiner le contenu à chaque fois qu'un évènement #PB_Event_Repaint est reçu.

. :Evènements :.

```
Le canvas renvoie les types
   d'évènements EventType()
 suivant:
Connaitre l'état du canvas:
   GetGadgetAttribute()
 avec les options suivantes:
\begin{quote}

  #PB_EventType_MouseEnter
        : Le curseur de la
   souris est entré dans le
   gadget
  #PB_EventType_MouseLeave
        : Le curseur de la
   souris est sorti du gadget
  #PB_EventType_MouseMove
         : Le curseur de la
   souris a bougé
  #PB_EventType_MouseWheel(*)
     : La molette de la
   souris a bougé

  #PB_EventType_LeftButtonDown
    : Le bouton gauche de la
   souris a été pressé
  #PB_EventType_LeftButtonUp
      : Le bouton gauche de
   la souris a été relâché
  #PB_EventType_LeftClick
         : Un clic avec le
   bouton gauche de la souris
  #PB_EventType_LeftDoubleClick
   : Un double-clic avec le
   bouton gauche de la souris
  #PB_EventType_RightButtonDown
   : Le bouton droit de la
   souris a été pressé
  #PB_EventType_RightButtonUp
     : Le bouton droit de la
   souris a été relâché
  #PB_EventType_RightClick
        : Un clic avec le
   bouton droit de la souris
  #PB_EventType_RightDoubleClick:
   Un double-clic avec le
   bouton droit de la souris
  #PB_EventType_MiddleButtonDown:
   Le bouton central de la
```

```
  souris a été pressé
  #PB_EventType_MiddleButtonUp
    : Le bouton central de la
  souris a été relâché

  #PB_EventType_Focus
             : Le gadget a
  obtenu le focus clavier
  #PB_EventType_LostFocus
          : Le gadget a perdu
  le focus clavier
  #PB_EventType_KeyDown(*)
        : Une touche clavier
  a été pressée
  #PB_EventType_KeyUp(*)
          : Une touche
  clavier a été relâchée
  #PB_EventType_Input(*)
          : Du texte a été
  entré dans le gadget

  #PB_EventType_Resize
           : Le gadget a été
  redimensionné

  (*)    Les évènements
  #PB_EventType_KeyDown,
  #PB_EventType_KeyUp
         et
  #PB_EventType_Input sont
  renvoyés seulement si le
  gadget
         a le focus clavier
  (Voir l'option
  #PB_Canvas_Keyboard). \\

         Avec Windows,
  l'évènement
  #PB_EventType_MouseWheel
  est renvoyé seulement
         si le gadget a le
  focus clavier. Avec les
  autres OS, cet évènement
  est renvoyé au gadget
         sous le curseur,
  qu'il ait le focus clavier
  ou non.
\end{quote}
```

. :Attributs :. Les connaître.

```
  - GetGadgetAttribute()
 permet de connaître les
   attributs du canvas avec
   les options suivantes:
```

Souris :

```
  - #PB_Canvas_MouseX    :
   Renvoie les coordonnées de
```

```
  la souris dans la zone de
  dessin du canvas
 - #PB_Canvas_MouseY     : au
  moment où l'évènement a
  été généré.
                          Note:
  Le résultat peut être
  différent des coordonnées
  envoyées par WindowMouseX()
et par
                                WindowMouseY()
qui elles, renvoient
  l'emplacement actuel de la
  souris indépendamment des
                                évènements.

 - #PB_Canvas_Buttons    :
  Renvoie l'état des boutons
  de la souris.
                          Peut
  être une combinaison de:
                            #PB_Canvas_LeftButton
   : Le bouton de gauche est
  actuellement enfoncé.
                            #PB_Canvas_RightButton
  : Le bouton de droite est
  actuellement enfoncé.
                            #PB_Canvas_MiddleButton:
  Le bouton du milieu est
  actuellement enfoncé.

 - #PB_Canvas_WheelDelta:
  Renvoie le mouvement de la
  molette de la souris par
  multiple de 1 ou -1.
                          Une
  valeur positive indique
  que la roue a été tournée
  vers le haut (en
  l'éloignant de
                          l'utilisateur)
  et une valeur négative
  indique que la roue a été
  déplacée vers le bas
                          (vers
  l'utilisateur).
                          Cet
  attribut est à 0 si
  l'évènement en cours n'est
  pas un évènement
  #PB_EventType_MouseWheel.

 - #PB_Canvas_Clip       :
  Renvoie une valeur non
  nulle si la souris est
  confinée dans le canvas,
  zéro sinon.

 - #PB_Canvas_Cursor     :
```

Renvoie le curseur qui est actuellement utilisé dans le gadget. Voir ci-dessous pour obtenir une liste des valeurs possibles. Si le gadget utilise un handle de curseur personnalisé, la valeur retournée est -1.

- #PB_Canvas_CustomCursor: Renvoie le handle personnalisé du curseur qui a été défini à l'aide de SetGadgetAttribute() . Si le gadget utilise un curseur standard, la valeur retournée est zéro.

Clavier :

- #PB_Canvas_Modifiers: Renvoie l'état des touches de contrôle du clavier. Peut être une combinaison de: #PB_Canvas_Shift : La touche 'Shift' est actuellement pressée. #PB_Canvas_Alt : La touche 'Alt' est actuellement pressée. #PB_Canvas_Control: La touche 'Control' est actuellement pressée. #PB_Canvas_Command: La touche 'Command' (ou 'Pomme') est actuellement pressée. (OS X seulement)

- #PB_Canvas_Key : Renvoie la touche du clavier qui a été enfoncée ou relâchée après un évènement #PB_EventType_KeyDown ou un évènement #PB_EventType_KeyUp et la valeur retournée est l'une des constantes #PB_Shortcut_ ... qui sont utilisées par la fonction AddKeyboardShortcut() . Note:

Cet attribut renvoie la
première touche pressée.
Note:
Pour obtenir un texte
saisi au clavier dans le
Canvasgadget, il est
préférable d'utiliser
l'évènement
#PB_EventType_Input
avec
l'attribut
#PB_Canvas_Input, car il
permet la saisie de texte
à partir d'une combinaison
de plusieurs touches telles
que
la touche 'Shift' et
autres touches muettes.

- #PB_Canvas_Input: Renvoie
le caractère qui a été
généré par une ou
plusieurs touches. Cet
attribut n'est présent
qu'après un évènement
#PB_EventType_Input.
La
valeur du caractère
retourné peut être
convertie en une chaîne de
caractères en utilisant la
fonction Chr()

.

Image :

- #PB_Canvas_Image: Renvoie
un ImageID, valeur qui
représente le numéro d'une
image avec le contenu
actuel du CanvasGadget.
Cette
valeur peut être utilisée
pour dessiner le contenu
du gadget vers une autre
surface de dessin en
utilisant la fonction
DrawImage()

.

Note:
La valeur retournée est
valable uniquement jusqu'à
l'apparition d'une
modification apportée au
gadget comme le
redimensionnement
ou un changement dans le
dessin, donc elle ne
devrait être utilisée
directement que dans une

```
commande de
                              type
DrawImage()
et ne pas être stockée pour
une utilisation future.
```

Attention : Ces informations ne sont disponibles que si l'évènement courant reçu par WaitWindowEvent() ou WindowEvent() provient d'un canvas.

. :Attributs :. Les modifier.

```
- SetGadgetAttribute()
permet de changer les
attributs du canvas avec
les options suivantes:
```

Souris :

```
- #PB_Canvas_Clip: Une
valeur non nulle et le
mouvement du curseur de la
souris sera confiné à la
zone du canvas gadget.
                  Avec une
valeur nulle, cela
supprime le confinement.
                  Note: Le
confinement de la souris
doit être uniquement le
résultat direct de
l'action de l'utilisateur
sur le canvas, comme un
clic de souris par exemple,
                        et
il faut prendre soin de
bien enlever le
confinement à nouveau,
sinon l'utilisateur sera
emprisonné à l'intérieur
du gadget.
                  L'option
#PB_Canvas_ClipMouse peut
être utilisée pour
automatiquement
capturer/relâcher la
souris lorsque
l'utilisateur appuie ou
relâche le bouton
                  de la
souris dans le canvas.

- #PB_Canvas_Cursor: Change
le curseur lorsque la
souris survole le gadget.
                    Les
valeurs suivantes sont
possibles:
                      #PB_Cursor_Default
  : Flèche du curseur par
défaut
```

#PB_Cursor_Cross
: Curseur en forme de croix
#PB_Cursor_IBeam
: Barre d'insertion 'I' utilisée pour la sélection de texte
#PB_Cursor_Hand
: Curseur main
#PB_Cursor_Busy
: Curseur sablier ou une montre
#PB_Cursor_Denied
: Curseur cercle barré ou curseur X
#PB_Cursor_Arrows
: Flèches dans toutes les directions (non disponible sur OS X)
#PB_Cursor_LeftRight
: Flèches gauche et droite
#PB_Cursor_UpDown
: Flèches haut et bas
#PB_Cursor_LeftUpRightDown: Flèches diagonales (Windows uniquement)
#PB_Cursor_LeftDownRightUp: Flèches diagonales (Windows uniquement)
#PB_Cursor_Invisible
: Cache le curseur

- #PB_Canvas_CustomCursor: Change le curseur lorsque la souris survole le gadget en un curseur personnalisé créé en utilisant l'API de l'OS correspondant.

Le type attendu est:

Avec Windows: Un handle HCURSOR

Avec Linux : Un pointeur GtkCursor

Avec OS X: Un pointeur vers une structure Curseur

Image :

- #PB_Canvas_Image: Applique l'ImageID au CanvasGadget.

Le gadget fait une copie de l'image d'entrée afin qu'elle puisse être

```
    libérée ou réutilisée
    après cet appel.
                        Utiliser
    cet attribut revient à
    utiliser StartDrawing()
, CanvasOutput()
 et DrawImage()
 pour dessiner l'image sur le
    CanvasGadget.

  - GadgetToolTip()
 permet d'ajouter une 'mini
    aide' à ce gadget.
```

Exemple

```
1   If OpenWindow(0, 0, 0, 220,
    220, "CanvasGadget",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2    CanvasGadget(0, 10, 10,
    200, 200)
3
4    Repeat
5      Event =
    WaitWindowEvent()
6
7      If Event =
    #PB_Event_Gadget And
    EventGadget() = 0
8        If EventType() =
    #PB_EventType_LeftButtonDown
    Or (EventType() =
    #PB_EventType_MouseMove
    And GetGadgetAttribute(0,
    #PB_Canvas_Buttons) &
    #PB_Canvas_LeftButton)
9          If
    StartDrawing(CanvasOutput(0))
10           x =
    GetGadgetAttribute(0,
    #PB_Canvas_MouseX)
11           y =
    GetGadgetAttribute(0,
    #PB_Canvas_MouseY)
12           Circle(X, Y, 10,
    RGB(Random(255),
    Random(255), Random(255)))
13           StopDrawing()
14         EndIf
15       EndIf
16     EndIf
17
18    Until Event =
    #PB_Event_CloseWindow
19  EndIf
```

Exemple : Canvas container

```
If OpenWindow(0, 0, 0, 220, 220, "Canvas container", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)

  CanvasGadget(0, 10, 10, 200, 200, #PB_Canvas_Container)
  ButtonGadget(1, 10, 10, 80, 30, "Effacer")
  CloseGadgetList()

  Repeat
    Event = WaitWindowEvent()

    If Event = #PB_Event_Gadget
      Select EventGadget()
        Case 0
          If EventType() = #PB_EventType_LeftButtonDown Or (EventType() = #PB_EventType_MouseMove And GetGadgetAttribute(0, #PB_Canvas_Buttons) & #PB_Canvas_LeftButton)
            If StartDrawing(CanvasOutput(0))
              x = GetGadgetAttribute(0, #PB_Canvas_MouseX)
              y = GetGadgetAttribute(0, #PB_Canvas_MouseY)
              Circle(x, y, 10, RGB(Random(255), Random(255), Random(255)))
              StopDrawing()
            EndIf
          EndIf

        Case 1
          If StartDrawing(CanvasOutput(0))
            Box(0, 0, 200, 200, #White)
            StopDrawing()
          EndIf
      EndSelect
    EndIf

  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

CanvasOutput() , GetGadgetAttribute() , SetGadgetAttribute() , EventType() , StartDrawing()

OS Supportés

Tous

99.7 CanvasOutput

Syntaxe

```
Resultat =
    CanvasOutput(#Gadget)
```

Description

Renvoie le numéro d'identification OutputID d'un CanvasGadget pour effectuer l'opération de rendu 2D.

Arguments

#Gadget Le numéro du CanvasGadget() .

Valeur de retour

Renvoie l'identifiant ouputID ou zéro si le dessin n'est pas possible.
Cette valeur doit être transmise directement à la fonction StartDrawing() pour lancer l'opération de dessin.
La valeur de retour n'est valable que pour une seule opération de dessin et ne peut pas être réutilisée.

Exemple

```
1  ...
2  StartDrawing(CanvasOutput(#Gadget))
3    ; code de dessin ici ...
4  StopDrawing()
```

Remarques

Le dessin sur un CanvasGadget() utilise un double tampon. Cela signifie que les opérations de dessin ne deviennent visibles qu'après la commande StopDrawing() pour éviter un effet de scintillement pendant le rendu.
Le CanvasGadget() n'ayant pas de canal alpha (transparence), les mode #PB_2DDrawing_AlphaChannel de la fonction DrawingMode() n'aura aucun effet et le mode #PB_2DDrawing_AllChannels sera équivalent à #PB_2DDrawing_Default

Voir aussi

StartDrawing() , CanvasGadget() , CanvasVectorOutput()

OS Supportés

Tous

99.8 CanvasVectorOutput

Syntaxe

```
Resultat =
    CanvasVectorOutput(#Gadget
    [, UniteDeMesure])
```

Description

Renvoie le numéro d'identification OutputID d'un CanvasGadget afin d'effectuer des opérations de dessin vectoriel.

Arguments

#Gadget Le numéro du CanvasGadget() .

UniteDeMesure (optionnel) Spécifie l'unité utilisée pour mesurer les distances sur le dessin.

```
#PB_Unit_Pixel      : Les
 valeurs sont mesurées en
 pixels (Par défaut)(ou
 point (dots) pour les
 imprimantes)
#PB_Unit_Point      : Les
 valeurs sont mesurées en
 points (1/72 pouce =
 25.4/72 mm = 0,352 778
 mm)
#PB_Unit_Inch       : Les
 valeurs sont mesurées en
 pouces (25,4 millimètres)
```

```
#PB_Unit_Millimeter: Les
 valeurs sont mesurées en
 millimètres (0,039 370
 pouce)
```

Valeur de retour

Renvoie l'identifiant ouputID ou zéro si le dessin n'est pas possible.
Cette valeur doit être transmise directement à la fonction StartVectorDrawing() pour lancer l'opération de dessin.
La valeur de retour n'est valable que pour une seule opération de dessin et ne peut pas être réutilisée.

Exemple

```
1  ...
2  StartVectorDrawing(CanvasVectorOutput(#Gadget))
3    ; code de dessin ici ...
4  StopVectorDrawing()
```

Remarques

Le dessin sur un CanvasGadget() utilise un double tampon. Cela signifie que les opérations de dessin ne deviennent visibles qu'après la commande StopVectorDrawing() pour éviter un effet de scintillement pendant le rendu.

Voir aussi

StartVectorDrawing() , CanvasGadget() , CanvasOutput()

OS Supportés

Tous

99.9 OpenGLGadget

Syntaxe

```
Resultat =
    OpenGLGadget(#Gadget, X,
    Y, Longueur, Hauteur [,
    Options])
```

Description

Crée une surface d'affichage 3D OpenGL dans la GadgetListe.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.
La largeur et la hauteur maximale est de 16 000 pixels.

Options (optionnel) Peut être une combinaison des constantes suivantes :

```
#PB_OpenGL_Keyboard
                : permet
 de recevoir le focus et
 les évènements du
 clavier.
#PB_OpenGL_NoFlipSynchronization
  : désactive la
 synchronisation
 verticale vsync.
#PB_OpenGL_FlipSynchronization
    : permet la
 synchronisation
 verticale vsync (par
 défaut).
#PB_OpenGL_NoDepthBuffer
          : désactive le
 tampon de profondeur.
#PB_OpenGL_16BitDepthBuffer
       : crée un tampon
 de profondeur 16 bits
 (par défaut).
#PB_OpenGL_24BitDepthBuffer
       : crée un tampon
 de profondeur 24 bits.
#PB_OpenGL_NoStencilBuffer
        : désactive le
 stencil buffer (par
 défaut).
#PB_OpenGL_8BitStencilBuffer
      : crée un stencil
 buffer 8 bits.
#PB_OpenGL_NoAccumulationBuffer
   : désactive le tampon
 d'accumulation (par
 défaut).
#PB_OpenGL_32BitAccumulationBuffer:
 crée un tampon
 d'accumulation 32 bits.
#PB_OpenGL_64BitAccumulationBuffer:
 crée un tampon
 d'accumulation 64 bits.
```

L'option #PB_OpenGL_Keyboard est nécessaire pour recevoir des évènements du clavier dans le gadget. Si vous utilisez cette option, vous devriez utiliser les évènements #PB_EventType_Focus et

#PB_EventType_LostFocus afin de créer une indication visuelle autour ou sur le gadget quand il reçoit le focus. Ainsi il sera clair pour l'utilisateur que le gadget a bien reçu le focus.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

Ce gadget offre une surface d'affichage qui gère les évènements de la souris et du clavier afin de créer facilement des dessins 3D avec OpenGL. La plupart des commandes OpenGL sont directement disponibles dans PureBasic avec la notation de l'API qui utilise le caractère "soulignement" (la commande se termine par un trait de soulignement, par exemple : 'glBegin_()').
Le gadget créé affiche un fond noir et le contexte courant est réglé sur ce nouveau gadget. Pour modifier le contexte courant, utiliser l'attribut #PB_OpenGL_SetContext. Une fois établi, un cadre peut être affiché à l'aide de l'attribut #PB_OpenGL_FlipBuffers.
. : Evènements :
La fonction EventType() indique le type de l'évènement :

```
#PB_EventType_MouseEnter
      : Le curseur de la
 souris est entré dans le
 gadget
#PB_EventType_MouseLeave
      : Le curseur de la
 souris a quitté le gadget
#PB_EventType_MouseMove
       : Le curseur de la
 souris a bougé
#PB_EventType_MouseWheel
      : La molette de la
 souris a été utilisée
#PB_EventType_LeftButtonDown
  : Le bouton gauche de la
 souris a été pressé
#PB_EventType_LeftButtonUp
    : Le bouton gauche de
 la souris a été relâché
#PB_EventType_LeftClick
       : Un clic avec le
 bouton gauche de la souris
```

```
#PB_EventType_LeftDoubleClick
 : Un double-clic avec le
 bouton gauche de la souris
#PB_EventType_RightButtonDown
 : Le bouton droit de la
 souris a été pressé
#PB_EventType_RightButtonUp
   : Le bouton droit de la
 souris a été relâché
#PB_EventType_RightClick
      : Un clic avec le
 bouton droit de la souris
#PB_EventType_RightDoubleClick:
 Un double-clic avec le
 bouton droit de la souris
#PB_EventType_MiddleButtonDown:
 Le bouton du milieu de la
 souris a été pressé
#PB_EventType_MiddleButtonUp
  : Le bouton du milieu de
 la souris a été relâché
#PB_EventType_Focus
           : Le gadget a
 reçu le focus clavier
#PB_EventType_LostFocus
       : Le gadget a perdu
 le focus clavier
#PB_EventType_KeyDown
         : Une touche a été
 pressée
#PB_EventType_KeyUp
           : Une touche a
 été relâchée
#PB_EventType_Input
           : La saisie de
 texte a été générée
```

Notez que les évènements #PB_EventType_KeyDown, #PB_EventType_KeyUp et #PB_EventType_Input ne sont signalés que lorsque le gadget a le focus clavier. Cela signifie que l'option #PB_OpenGL_Keyboard doit être mise lors de la création du gadget pour permettre de recevoir les évènements du clavier. Sous Windows, l'évènement #PB_EventType_MouseWheel n'est également signalé que si le gadget a le focus clavier. Sur les autres systèmes d'exploitation, cet évènement est signalé au gadget sous le curseur, indépendamment de l'état du focus clavier.

. : GetGadgetAttribute()

Les attributs suivants peuvent être utilisés : #PB_OpenGL_MouseX, #PB_OpenGL_MouseY

Renvoie les coordonnées de la souris par rapport à la zone de dessin du gadget. Cela renvoie la position de la

souris au moment où l'évènement a été généré, de sorte que le résultat peut différer des coordonnées envoyées par WindowMouseX() et WindowMouseY() qui renvoient l'emplacement actuel de la souris indépendamment des évènements traités. Les coordonnées retournées à l'aide de ces attributs doivent être utilisés pour ce gadget pour s'assurer que les coordonnées de la souris sont en phase avec l'évènement en cours.

#PB_OpenGL_Buttons

Renvoie l'état des boutons de la souris. Le résultat est une combinaison (en utilisant au niveau du bit ou) des valeurs suivantes :

```
#PB_OpenGL_LeftButton  :
 Le bouton gauche est
 actuellement enfoncé.
#PB_OpenGL_RightButton :
 Le bouton droit est
 actuellement enfoncé.
#PB_OpenGL_MiddleButton:
 Le bouton du milieu est
 actuellement enfoncé.
```

#PB_OpenGL_Modifiers

Renvoie l'état des modificateurs de clavier pour l'évènement.
Le résultat est une combinaison (en utilisant au niveau du bit ou) des valeurs suivantes :

```
#PB_OpenGL_Shift   : La
 touche 'majuscule
 ('shift') est
 actuellement pressée.
#PB_OpenGL_Alt     : La
 touche 'alt' est
 actuellement pressée.
#PB_OpenGL_Control: La
 touche 'control' est
 actuellement pressée.
#PB_OpenGL_Command: La
 touche 'command' (ou
 'pomme') est
 actuellement pressée.
 (OS X seulement)
```

#PB_OpenGL_WheelDelta

Renvoie le mouvement de la molette de la souris par multiple de 1 ou -1.
Une valeur positive indique que la roue a été déplacée vers le haut (loin de l'utilisateur) et une valeur négative indique que la roue a été déplacée vers le bas (vers l'utilisateur). Cet attribut est égal à zéro si l'évènement en cours

n'est pas un évènement #PB_EventType_MouseWheel.

#PB_OpenGL_Key

Renvoie la touche qui a été enfoncée ou relâchée dans un évènement #PB_EventType_KeyDown ou un évènement #PB_EventType_KeyUp. La valeur retournée est l'une des valeurs #PB_Shortcut_ ... utilisées par la fonction AddKeyboardShortcut() . Cet attribut renvoie les touches, une à une. Pour la saisie de texte, il est préférable de voir l'évènement #PB_EventType_Input et d'utiliser l'attribut #PB_OpenGL_Input car il permet la saisie de texte même avec des combinaisons de touches (majuscule, accent circonflexe, ...).

#PB_OpenGL_Input

Renvoie la valeur du caractère généré par l'appui d'une ou de plusieurs touches.
Cet attribut n'est possible qu'après un évènement #PB_EventType_Input.
La valeur du caractère retourné peut être convertie en une chaîne de caractère en utilisant la fonction Chr() .

#PB_OpenGL_Cursor

Renvoie le curseur qui est actuellement utilisé dans le gadget.
Voir ci-dessous la liste des valeurs possibles.
Si le gadget utilise un handle de curseur personnalisé, le valeur de retour est -1.

#PB_OpenGL_CustomCursor

Renvoie le handle du curseur personnalisé qui a été défini avec SetGadgetAttribute() .
Si le gadget utilise un curseur standard, la valeur de retour est 0.

. : SetGadgetAttribute() modifie les attributs suivants :

#PB_OpenGL_SetContext

Modifie le contexte courant du gadget.

```
#True  : Utilise le
 contexte OpenGL du
 gadget.
#False : Retire le
 contexte courant. Plus
 de contexte disponible.
```

#PB_OpenGL_FlipBuffers

Renvoie les tampons arrière-plan et avant plan. Tous les dessins sont effectués dans la mémoire tampon d'arrière plan. Pour être visible, les

tampons doivent être échangés, de sorte que le tampon d'arrière plan devienne le tampon d'avant plan (celui qui est affiché).

```
#True  : Echange les
 tampons
#False : Ne pas échanger
 les tampons (pour
 l'instant).
```

#PB_OpenGL_Cursor

Change le curseur qui est affiché lorsque la souris passe au dessus du gadget.
Les valeurs suivantes sont possibles :

```
#PB_Cursor_Default
         : Par défaut le
 curseur flèche
#PB_Cursor_Cross
          : Croix
#PB_Cursor_IBeam
          : Point
 d'insertion "I"
#PB_Cursor_Hand
           : Main
#PB_Cursor_Busy
           : Sablier ou
 montre
#PB_Cursor_Denied
         : Cercle barré
 ou grand "X"
#PB_Cursor_Arrows
         : Flèches dans
 toutes les directions
 (non disponible sur OS
 X)
#PB_Cursor_LeftRight
       : Flèches gauche
 et droite
#PB_Cursor_UpDown
         : Flèches haut
 et bas
#PB_Cursor_LeftUpRightDown:
 Flèches diagonales
 (Windows uniquement)
#PB_Cursor_LeftDownRightUp:
 Flèches diagonales
 (Windows uniquement)
#PB_Cursor_Invisible
       : Curseur invisible
```

#PB_OpenGL_CustomCursor

Change le curseur qui est affiché lorsque la souris survole le gadget en un handle de curseur personnalisé créé à l'aide de l'API du système d'exploitation correspondant.
Cet attribut prévoit le type d'entrée suivants :

Windows : Un handle HCURSOR
Linux : Un pointeur GtkCursor
OS X : Un pointeur vers une structure de curseur

. : Une 'mini aide' peut être ajoutée à ce gadget à l'aide de GadgetToolTip() .

Voir aussi

GetGadgetAttribute() , SetGadgetAttribute() , EventType()

OS Supportés

Tous

99.10 CheckBoxGadget

Syntaxe

```
Resultat =
    CheckBoxGadget(#Gadget, X,
    Y, Largeur, Hauteur,
    Texte$ [, Options])
```

Description

Crée un gadget case à cocher dans la GadgetList en cours.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Texte$ Le texte à afficher à côté de la case, il est placé à sa droite.

Options (optionnel) Peut être une combinaison de :

```
#PB_CheckBox_Right      :
 Aligne le texte de la
 case à cocher à droite.
#PB_CheckBox_Center     :
 Centre le texte de la
 case à cocher.
#PB_CheckBox_ThreeState:
 Crée une case à cocher
 qui peut avoir trois
 états (dont un état
 intermédiaire).
```

L'option #PB_CheckBox_ThreeState permet à une case à cocher de représenter l'état de plusieurs éléments. L'état indéterminé (ni coché, ni décoché) indique alors que certains éléments sont dans un état différent des autres. En cliquant sur la case à cocher, l'utilisateur peut alors les remettre tous dans le même état, soit coché, soit décoché. L'état indéterminé peut donc uniquement être activé à l'aide de SetGadgetState() .

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

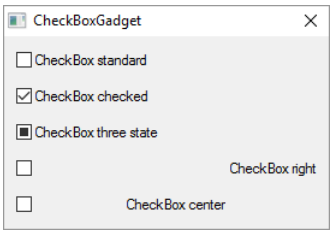
Remarques

```
 - GadgetToolTip()
permet d'ajouter une 'mini
  aide' à ce gadget.

 - GetGadgetState()
est utilisé pour récupérer
  l'état du gadget.
 - SetGadgetState()
est utilisé pour changer
  l'état du gadget.
```

Exemple

```
1  If OpenWindow(0, 0, 0, 270,
     160, "CheckBoxGadget",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2     CheckBoxGadget(0, 10,
     10, 250, 20, "Case à
     cocher standard")
3     CheckBoxGadget(1, 10,
     40, 250, 20, "Case à
     cocher (cochée)") :
     SetGadgetState(1,
     #PB_Checkbox_Checked)
4     CheckBoxGadget(2, 10,
     70, 250, 20, "Case à
     cocher à trois états",
     #PB_CheckBox_ThreeState) :
     SetGadgetState(2,
     #PB_Checkbox_Inbetween)
5     CheckBoxGadget(3, 10,
     100, 250, 20, "Case à
```

```
    cocher (texte aligné à
    droite)",
    #PB_CheckBox_Right)
6    CheckBoxGadget(4, 10,
    130, 250, 20, "Case à
    cocher (texte centré)",
    #PB_CheckBox_Center)
7    Repeat : Until
    WaitWindowEvent() =
    #PB_Event_CloseWindow
8  EndIf
```



Voir aussi

GetGadgetState() , SetGadgetState() , OptionGadget()

OS Supportés

Tous

99.11 ClearGadgetItems

Syntaxe

```
ClearGadgetItems(#Gadget)
```

Description

Supprime tous les éléments d'un gadget.

Arguments

#**Gadget** Le gadget à nettoyer.

Valeur de retour

Aucune.

Remarques

Le gadget doit être de l'un des types suivants :

- `ComboBoxGadget()`
- `EditorGadget()`
- `ListViewGadget()`

- ListIconGadget()

- MDIGadget()

- PanelGadget()

- TreeGadget()

Exemple

```
If OpenWindow(0, 0, 0, 260, 150, "ClearGadgetItems", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
 ListViewGadget(0,10,10,240,100)
 For a = 1 To 10 : AddGadgetItem (0, -1, "Elément "+Str(a)) : Next ; ajoute 10 éléments
 ButtonGadget(1, 10, 120, 200, 20, "Supprimer le contenu de la boîte de liste")
 Repeat
   Evenement = WaitWindowEvent()
   If Evenement = #PB_Event_Gadget
     If EventGadget() = 1
       ClearGadgetItems(0)
     EndIf
   EndIf
 Until Evenement = #PB_Event_CloseWindow
EndIf
```

Voir aussi

AddGadgetItem() , RemoveGadgetItem() , CountGadgetItems()

OS Supportés

Tous

99.12 CloseGadgetList

Syntaxe

```
CloseGadgetList()
```

Description

Ferme la liste de gadgets courante et revient à la précédente.

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

```
C'est particulièrement
 utile pour les gadgets
 suivants:
  - CanvasGadget()

  - ContainerGadget()

  - PanelGadget()

  - ScrollAreaGadget()
```

Voir aussi

OpenGadgetList() , ContainerGadget() , PanelGadget() , ScrollAreaGadget()

OS Supportés

Tous

99.13 ComboBoxGadget

Syntaxe

```
Resultat =
    ComboBoxGadget(#Gadget, X,
    Y, Largeur, Hauteur [,
    Options])
```

Description

Crée un gadget Liste déroulante dans la GadgetList en cours.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Options (optionnel) Peut être une combinaison de :

```
#PB_ComboBox_Editable :
 Rend la liste déroulante
 éditable
```

```
#PB_ComboBox_LowerCase:
 Tous les textes entrés
 dans la combobox sont
 convertis en minuscules
#PB_ComboBox_UpperCase:
 Tous les textes entrés
 dans la combobox sont
 convertis en majuscules
#PB_ComboBox_Image    :
 Active le support des
 images dans les éléments
 (sauf ComboBox éditables
 sous MacOS X)
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

```
 - GadgetToolTip()
permet d'ajouter une 'mini
  aide' à ce gadget.

 - AddGadgetItem()
    : Ajoute un élément
 - CountGadgetItems()
 : Renvoie le nombre
  d'éléments que contient le
  ComboBox.
 - ClearGadgetItems()
 : Supprime tous les éléments
 - RemoveGadgetItem()
 : Supprime un élément

 - SetGadgetItemText()
: Change le texte de
  l'élément spécifié
 - GetGadgetItemText()
: Renvoie le texte de
  l'élément spécifié

 - SetGadgetItemImage()
: Change l'image associée au
  gadget (doit être créé
  avec l'option
  #PB_ComboBox_Image).

 - SetGadgetState()
   : Change l'élément
  sélectionné.
 - GetGadgetState()
```

: Renvoie l'index de l'élément sélectionné ou -1 si aucun élément n'a été ajouté ou sélectionné.

- SetGadgetText() : Change le texte affiché. Si le ComboBoxGadget n'est pas éditable, le texte doit être dans la liste déroulante.
- GetGadgetText() : Renvoie le contenu texte de la zone visible de la ComboBox.

- SetGadgetItemData() : Associe une valeur personnalisée à cet élément.
- GetGadgetItemData() : Renvoie la valeur personnalisée associée à cet élément.

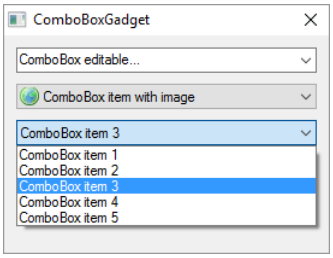
ComboBoxGadget prend en charge les évènements suivants, voir EventType() :

#PB_EventType_Change : La sélection du texte dans le champ d'édition a changé.
#PB_EventType_Focus : Le champ d'édition a reçu le focus clavier (ComboBox modifiable uniquement).
#PB_EventType_LostFocus: Le champ d'édition a perdu le focus du clavier (ComboBox modifiable uniquement).

Exemple

```
1  UsePNGImageDecoder()
2  LoadImage(0, #PB_Compiler_Home + "examples/sources/Data/world.png")
3
4  If OpenWindow(0, 0, 0, 270, 180, "ComboBoxGadget", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
5     ComboBoxGadget(0, 10, 10, 250, 21, #PB_ComboBox_Editable)
6        AddGadgetItem(0, -1, "Liste déroulante éditable...")
7
8     ComboBoxGadget(1, 10, 40, 250, 21,
```

```
    #PB_ComboBox_Image)
9      AddGadgetItem(1, -1,
    "Liste déroulante avec une
    image", ImageID(0))
10
11     ComboBoxGadget(2, 10, 70,
    250, 21)
12     For a = 1 To 5
13       AddGadgetItem(2,
    -1,"Elément de liste
    déroulante " + Str(a))
14     Next
15
16     SetGadgetState(0, 0)
17     SetGadgetState(1, 0)
18     SetGadgetState(2, 2) ;
    Sélectionne le troisième
    élément (la numérotation
    commence à 0)
19
20     Repeat : Until
    WaitWindowEvent() =
    #PB_Event_CloseWindow
21   EndIf
```



Voir aussi

AddGadgetItem() , RemoveGadgetItem() , CountGadgetItems() , ClearGadgetItems() , GetGadgetState() , SetGadgetState() , GetGadgetText() , SetGadgetText() , SetGadgetItemImage() , GetGadgetItemText() , SetGadgetItemText() , GetGadgetItemData() , SetGadgetItemData() , ExplorerComboGadget()

OS Supportés

Tous

99.14 ContainerGadget

Syntaxe

```
Resultat =
    ContainerGadget(#Gadget,
```

```
X, Y, Largeur, Hauteur [,
Options])
```

Description

Crée un nouveau gadget container qui est un simple panneau destiné à contenir d'autres gadgets.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Options (optionnel) Peut être une combinaison de :

```
#PB_Container_BorderLess:
 Sans bordure (Defaut)
#PB_Container_Flat      :
 Cadre simple
#PB_Container_Raised    :
 Cadre élevé
#PB_Container_Single    :
 Cadre enfoncé
#PB_Container_Double    :
 Cadre doublement enfoncé
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

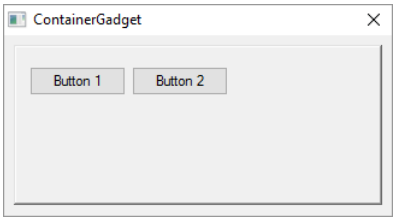
Une fois le gadget créé, les gadgets créés utérieurement feront partie du container. Pour ne plus ajouter de gadgets au container, utilisez la fonction CloseGadgetList() pour fermer la 'GadgetList' du container et retourner à la 'GadgetList' précédente.
OpenGadgetList() pourra être utilisé pour ajouter des gadgets dynamiquement dans ce #Gadget.

```
 - GadgetToolTip()
permet d'ajouter une 'mini
  aide' à ce gadget.

 Les fonctions suivantes
  peuvent être appelées pour
  agir sur le container
```

```
 - SetGadgetColor()
et GetGadgetColor()
avec la constante
    #PB_Gadget_BackColor
   comme 'TypeCouleur' pour
   changer la couleur de fond
   du gadget.

L'évènement suivant est pris
   en charge par EventType()
:
  #PB_EventType_Resize: Le
   gadget a été redimensionné.
```

Exemple

```
1  If OpenWindow(0, 0, 0, 322,
    150, "ContainerGadget",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2    ContainerGadget(0, 8, 8,
    306, 133,
    #PB_Container_Raised)
3       ButtonGadget(1, 10, 15,
    80, 24, "Bouton 1")
4       ButtonGadget(2, 95, 15,
    80, 24, "Bouton 2")
5    CloseGadgetList()
6    Repeat : Until
    WaitWindowEvent() =
    #PB_Event_CloseWindow
7  EndIf
```



Voir aussi

OpenGadgetList() , CloseGadgetList() , SetGadgetColor() , GetGadgetColor()

OS Supportés

Tous

99.15 CountGadgetItems

Syntaxe

```
Resultat =
    CountGadgetItems(#Gadget)
```

Description

Renvoie le nombre d'éléments que contient un gadget.

Arguments

#**Gadget** Le gadget à utiliser.

Valeur de retour

Renvoie le nombre d'éléments du gadget.

Remarques

C'est une fonction universelle qui fonctionne avec tous les gadgets qui gèrent plusieurs éléments :

- `ComboBoxGadget()`
- `EditorGadget()`
- `ExplorerListGadget()`
- `ListIconGadget()`
- `ListViewGadget()`
- `MDIGadget()`
- `PanelGadget()`
- `TreeGadget()`

Voir aussi

AddGadgetItem() , RemoveGadgetItem() , ClearGadgetItems()

OS Supportés

Tous

99.16 DateGadget

Syntaxe

```
Resultat =
    DateGadget(#Gadget, X, Y,
    Largeur, Hauteur [,
    Masque$ [, Date [,
    Options]]])
```

Description

Crée un champ de saisie (identique à un StringGadget()) dans la GadgetList en cours, dans lequel il sera possible d'entrer une date et/ou une heure.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Masque$ (optionnel) Le format de la date. Voir FormatDate() pour le format de ce masque.

Note importante : Le gadget ne supporte pas l'affichage des secondes, donc si vous spécifiez "%ss" dans le paramètre 'Masque$, il sera tout simplement ignoré ! Si vous ne spécifiez pas de masque ou si vous spécifiez une chaîne vide, un masque par défaut sera utilisé. Le masque peut être modifié avec la fonction SetGadgetText() .

Date (optionnel) La date initialement prévue pour le gadget.
Ne pas spécifier ce paramètre ou utiliser une valeur 0 permet d'afficher la date et l'heure du jour.

Options (optionnel) Peut être une combinaison de :
#PB_Date_UpDown
Par défaut le gadget a un bouton pour afficher un calendrier à partir duquel l'utilisateur peut choisir une date facilement (voir l'image ci-dessous). Avec cette option, deux boutons (un pour monter, un pour descendre) seront présents pour permettre de modifier la date directement, sans passer par un calendrier.
Cette option est seulement disponible sous Windows.
#PB_Date_CheckBox, le gadget aura une case à cocher à partir de laquelle l'utilisateur pourra mettre le gadget sur 'Aucune date' (si la case à cocher est décochée).
Tant que la case à cocher est décochée, GetGadgetState() renverra 0.
Pour changer l'état de la case à cocher, utiliser SetGadgetState() avec soit 0 (aucune date), soit avec une date valide.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

Ce gadget utilise le même format de date que celui utilisé par les fonctions de la bibliothèque Date . Ainsi, vous pouvez, par exemple, utiliser FormatDate() pour afficher le résultat renvoyé par GetGadgetState() , avec un format personnalisé.

```
  - GadgetToolTip()
 permet d'ajouter une 'mini
   aide' à ce gadget.

  Les commandes suivantes
   peuvent être utilisées
   pour agir sur ce gadget :
  - SetGadgetState()
: Change la date du gadget.
  - GetGadgetState()
: Renvoie la valeur numérique
   de la date actuellement
   affichée.

  - SetGadgetText()
 : Change le masque du gadget.
  - GetGadgetText()
 : Renvoie la date
   actuellement affichée sous
   forme de texte (exactement
   comme elle est affichée).

  - GetGadgetAttribute()
 avec les attributs suivants:
    #PB_Date_Minimum: Renvoie
   la date minimale que
   l'utilisateur peut choisir
    #PB_Date_Maximum: Renvoie
   la date maximale que
   l'utilisateur peut choisir
   avec ce gadget
                       (Remarque:
   la date sélectionnable
   n'est pas prise en charge
   sous Linux.)

  - SetGadgetAttribute()
: avec les attributs suivants:
    #PB_Date_Minimum: Change
   la date minimale que
   l'utilisateur peut choisir
    #PB_Date_Maximum: Change
   la date maximale que
   l'utilisateur peut choisir
   avec ce gadget
                       (Remarque:
   la date sélectionnable
   n'est pas prise en charge
   sous Linux.)

  - SetGadgetColor()
```

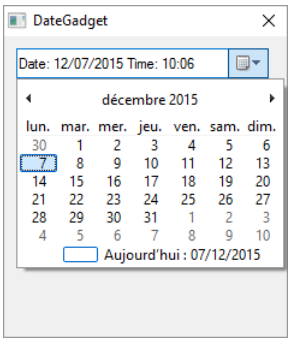
```
et GetGadgetColor()
avec les constantes
  suivantes comme
  'TypeCouleur':
   #PB_Gadget_BackColor
  : Couleur de fond
   #PB_Gadget_FrontColor
  : Couleur du texte pour
  les jours affichés
   #PB_Gadget_TitleBackColor
  : Couleur du fond du titre
  du mois
   #PB_Gadget_TitleFrontColor:
  Couleur du texte du titre
  du mois
   #PB_Gadget_GrayTextColor
  : Couleur du texte pour
  les jours n'appartenant
  pas au mois courant.
```

L'évènement suivant est pris en charge par EventType() :

```
#PB_EventType_Change: La
 date a été modifiée par
 l'utilisateur.
```

Exemple

```
1  If OpenWindow(0, 0, 0, 250,
    250,"DateGadget",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2    DateGadget(0, 10, 10,
    190, 25, "Date :
    %dd/%mm/%yyyy Heure :
    %hh:%ii")
3    Repeat
4    Until WaitWindowEvent() =
    #PB_Event_CloseWindow
5  EndIf
```



Voir aussi

GetGadgetState() , SetGadgetState() , GetGadgetText() , SetGadgetText() ,

GetGadgetAttribute() , SetGadgetAttribute() , GetGadgetColor() , SetGadgetColor() , CalendarGadget() , Date() , FormatDate()

OS Supportés

Tous

99.17 DisableGadget

Syntaxe

```
DisableGadget(#Gadget, Etat)
```

Description

Active ou désactive un gadget.

Arguments

#Gadget Le gadget à activer ou à désactiver.

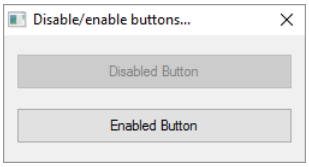
Etat `#False: Le gadget est activé.`
`#True : Le gadget est désactivé.`

Valeur de retour

Aucune.

Exemple

```
1  If OpenWindow(0, 0, 0, 250, 105, "Activer/Désactiver un Gadget", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2    ButtonGadget(0, 10, 15, 230, 30, "Bouton désactivé") : DisableGadget(0, #True)
3    ButtonGadget(1, 10, 60, 230, 30, "Bouton activé") : DisableGadget(1, #False)
4    Repeat : Until WaitWindowEvent() = #PB_Event_CloseWindow
5  EndIf
```



Voir aussi

HideGadget()

OS Supportés

Tous

99.18 EditorGadget

Syntaxe

```
Resultat =
    EditorGadget(#Gadget, X,
    Y, Largeur, Hauteur [,
    Options])
```

Description

Crée un gadget de type 'éditeur' permettant de saisir une grande quantité de texte dans la GadgetList courante.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Options (optionnel) Peut être une combinaison de :

```
#PB_Editor_ReadOnly:
 Gadget en lecture seule.
#PB_Editor_WordWrap:
 Retour à la ligne
 automatique.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

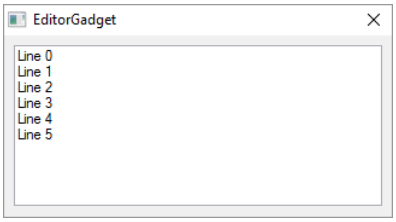
```
 - GadgetToolTip()
permet d'ajouter une 'mini
  aide' à ce gadget.

 - Les évènements suivants
  sont pris en charge par
  EventType()
```

```
:
    #PB_EventType_Change   :
   Le texte a été modifié par
   l'utilisateur.
    #PB_EventType_Focus    :
   L'éditeur a obtenu le
   focus.
    #PB_EventType_LostFocus:
   l'éditeur a perdu le focus.


  Les commandes suivantes
   peuvent être utilisées
   pour agir sur le contenu
   du gadget:
  - AddGadgetItem()
    : Ajoute une ligne de
   texte
  - CountGadgetItems()
 : Renvoie le nombre de
   lignes contenues dans le
   gadget.
  - GetGadgetItemText()
: Renvoie la ligne de texte
   spécifié
  - GetGadgetText()
    : Renvoie tout le texte
   contenu dans le gadget.
                          Notez
   que le retour à la ligne
   se fait avec "Chr (13) +
   Chr (10)"
                          sous
   Windows et avec "Chr (10)"
   sous Linux et OS X.
  - RemoveGadgetItem()
 : Efface la ligne spécifiée
  - ClearGadgetItems()
 : Efface tout le texte
  - SetGadgetItemText()
: Change le contenu de la
   ligne spécifiée
  - SetGadgetText()
    : Remplace tout le texte
   contenu par le gadget par
   un autre

  - SetGadgetAttribute()
 avec l'attribut suivant:
    #PB_Editor_ReadOnly:
   Change l'état 'lecture
   seule' du gadget (0 =
   éditable, 1 = non
   éditable).
    #PB_Editor_WordWrap:
   Retour à la ligne
   automatique.

  - GetGadgetAttribute()
```

```
avec l'attribut suivant:
   #PB_Editor_ReadOnly:
  Renvoie l'état 'lecture
  seule' du gadget (0 =
  éditable, 1 = non
  éditable).
   #PB_Editor_WordWrap:
  Renvoie l'état du retour à
  la ligne automatique.

 - SetGadgetColor()
et GetGadgetColor()
avec les valeurs
  'TypeCouleur' suivantes:
   #PB_Gadget_BackColor :
  Couleur de fond
   #PB_Gadget_FrontColor:
  Couleur du texte
```

Exemple

```
1  If OpenWindow(0, 0, 0, 322,
    150, "EditorGadget",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2    EditorGadget(0, 8, 8,
    306, 133)
3    For a = 0 To 5
4      AddGadgetItem(0, a,
    "Ligne "+Str(a))
5    Next
6    Repeat : Until
    WaitWindowEvent() =
    #PB_Event_CloseWindow
7  EndIf
```



Voir aussi

AddGadgetItem() , RemoveGadgetItem() , CountGadgetItems() , ClearGadgetItems() , GetGadgetText() , SetGadgetText() , GetGadgetItemText() , SetGadgetItemText() , GetGadgetAttribute() , SetGadgetAttribute() , GetGadgetColor() , SetGadgetColor() , StringGadget()

OS Supportés

Tous

99.19 ExplorerComboGadget

Syntaxe

```
Resultat =
    ExplorerComboGadget(#Gadget,
    X, Y, Largeur, Hauteur,
    Repertoire$, [, Options])
```

Description

Crée une combobox qui affiche un répertoire ainsi que tous ses répertoires parents permettant à l'utilisateur d'en choisir un. Utilisée dans OpenFileRequester() par exemple.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Repertoire$ Le répertoire initial affiché (doit être défini comme un chemin d'accès complet), une chaîne vide spécifie le dossier racine.
Si l'option #PB_Explorer_DrivesOnly est posée alors Repertoire$ ne peut être qu'une lettre de lecteur (”A :”, ”C :” etc.). Tout caractère qui suit la lettre du lecteur sera ignoré.

Options (optionnel) Peut être une combinaison de :

```
#PB_Explorer_DrivesOnly
  : Le Gadget n'affichera
  que les disques pour en
  sélectionner un.
#PB_Explorer_Editable
  : Le Gadget sera
  éditable, avec un mode
  d'auto-complétion
  automatique.
#PB_Explorer_NoMyDocuments:
  Le répertoire
  'MesDocuments' ne sera
  pas affiché séparément.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon. Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

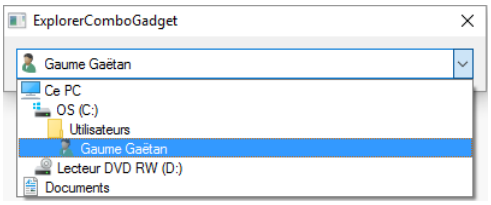
Remarques

Les commandes suivantes peuvent être utilisées pour agir sur le gadget :

```
  - GetGadgetText()
: Renvoie le répertoire
   actuellement affiché.
  - SetGadgetText()
: Change le répertoire
   actuellement affiché.
```

Exemple

```
1  If OpenWindow(0, 0, 0, 400,
    45, "ExplorerComboGadget",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2    ExplorerComboGadget(0,
    10, 10, 380, 25,
    GetHomeDirectory(),
    #PB_Explorer_Editable)
3    Repeat
4      Event =
    WaitWindowEvent()
5    Until Event =
    #PB_Event_CloseWindow
6  EndIf
```



Voir aussi

GetGadgetText() , SetGadgetText() , ExplorerListGadget() , ExplorerTreeGadget() , ComboBoxGadget()

OS Supportés

Tous

99.20 ExplorerListGadget

Syntaxe

```
Resultat =
    ExplorerListGadget(#Gadget,
    X, Y, Largeur, Hauteur,
    Repertoire$, [, Options])
```

Description

Créé un gadget qui affiche une liste des dossiers.
Il permet à l'utilisateur de choisir un fichier ou un dossier situé à n'importe quel endroit des disques.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Repertoire$ Le répertoire initial affiché.

'Repertoire$' est l'emplacement de départ lorsque le gadget est créé. Il peut être composé de motifs ('patterns'), tels que :
"C :*.pb;*.pbi". Si aucun 'pattern' n'est spécifié, le répertoire doit se terminer par un '\'. Si 'Repertoire$' est nul (chaîne vide) alors la liste des disques disponibles sera affichée.

Options (optionnel) Peut être une combinaison de :

```
#PB_Explorer_BorderLess
          : Crée un
 #Gadget sans bordure.
#PB_Explorer_AlwaysShowSelection:
 Affiche la sélection
 même quand le gadget
 n'est plus actif.
#PB_Explorer_MultiSelect
        : Active le mode
 'sélection multiple'.
#PB_Explorer_GridLines
          : Affiche des
 lignes de séparation.
#PB_Explorer_HeaderDragDrop
      : L'ordre des
 colonnes peut être
 changé avec un
 glisser/déposer
 (drag'n'drop).
#PB_Explorer_FullRowSelect
       : La sélection
 s'étend à toute la ligne
 au lieu de la première
 colonne.

#PB_Explorer_NoFiles
             : Aucun
 fichier ne sera affiché.
#PB_Explorer_NoFolders
           : Aucun
```

```
  répertoire ne sera
  affiché.
#PB_Explorer_NoParentFolder
      : Le 'pseudo'
  répertoire père [..] ne
  sera pas affiché.
#PB_Explorer_NoDirectoryChange
   : Le répertoire ne
  pourra pas être changé
  par l'utilisateur.
#PB_Explorer_NoDriveRequester
    : Aucun message du
  type 'Insérer un disque
  dans le lecteur A:' ne
  sera affiché.
#PB_Explorer_NoSort
              :
  L'utilisateur ne pourra
  pas trier les colonnes
  en cliquant sur les
  titres des colonnes.
#PB_Explorer_NoMyDocuments
       : Le répertoire
  spécial 'Mes documents'
  ne sera pas affiché
  séparément.
#PB_Explorer_AutoSort
            : Le contenu
  sera automatiquement
  trié par nom.
#PB_Explorer_HiddenFiles
         : Affiche les
  fichiers cachés
  (seulement avec Linux et
  OS X).
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

```
  - GadgetToolTip()
permet d'ajouter une 'mini
  aide' à ce gadget.

 Les commandes suivantes
  sont disponibles pour
  contrôler le gadget:
 - AddGadgetColumn()
  : Ajoute une colonne
  (prédéfinie ou
  additionnelle). Voir
  AddGadgetColumn()
```

```
 pour plus d'informations.
  - RemoveGadgetColumn()
: Supprime une colonne (ainsi
   que ses données).
  - GetGadgetText()
     : Renvoie le répertoire
   affiché.
  - SetGadgetText()
     : Change le répertoire
   affiché, ou les 'patterns'
   actuellement utilisés.
  - GetGadgetState()
    : Renvoie l'index du
   premier élément
   selectionné (-1: aucune
   selection).
  - GetGadgetItemText()
 : Renvoie le texte de
   l'élément, ou du titre de
   la colonne si 'Position' =
   -1.
  - SetGadgetItemText()
 : Modifie le texte de
   l'élément, ou du titre de
   la colonne si 'Position' =
   -1.
  - GetGadgetItemState()
: Vérifie si un élément est
   un répertoire ou un
   fichier et si il est
   sélectionné.
  - SetGadgetItemState()
: Modifier l'état sélectionné
   de l'élément spécifié.
  - CountGadgetItems()
  : Compte le nombre
   d'éléments dans le
   répertoire.

  - GetGadgetAttribute()
 / SetGadgetAttribute()
: avec l'attribut
   suivant(seulement sous
   Windows):
    #PB_Explorer_DisplayMode
   : Change le mode
   d'affichage du gadget. Le
   mode peut être l'une des
   constantes suivantes:
    #PB_Explorer_LargeIcon
   : Mode grandes icônes
    #PB_Explorer_SmallIcon
   : Mode petites icônes
    #PB_Explorer_List
   : Mode Liste
    #PB_Explorer_Report
   : Mode Détails (colonnes,
   mode par défaut)
```

```
  - GetGadgetItemAttribute()
 / SetGadgetItemAttribute()
: Avec l'attribut suivant:

    #PB_Explorer_ColumnWidth
    : Renvoie / Change la
    largeur de la 'Colonne'
    spécifiée. Le paramètre
    'Element' est ignoré.

  - SetGadgetColor()
 et GetGadgetColor()
 avec les valeurs suivantes
    comme 'TypeCouleur':
    #PB_Gadget_FrontColor:
    Couleur du texte
    #PB_Gadget_BackColor :
    Couleur du fond
    #PB_Gadget_LineColor :
    Couleur de la grille si
    l'option
    #PB_Explorer_GridLines est
    utilisée.

  Note: SetGadgetColor()
 n'est pas pris en charge sur
    la plate-forme MacOS X.
```

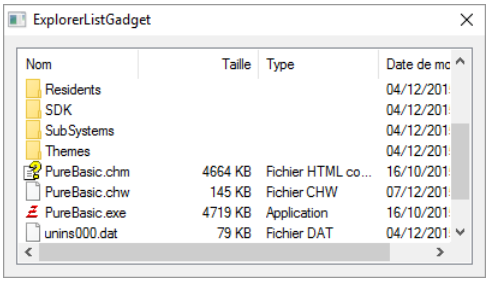
Les évènements suivants sont disponibles par l'intermédiaire d' EventType() :

```
  #PB_EventType_Change
            : La sélection de
    l'élément courant vient de
    changer.
  #PB_EventType_LeftClick
          : L'utilisateur a
    cliqué sur un élément avec
    le bouton gauche de la
    souris.
  #PB_EventType_RightClick
        : L'utilisateur a
    cliqué sur un élément avec
    le bouton droit de la
    souris.
  #PB_EventType_LeftDoubleClick
    : L'utilisateur a
    double-cliqué sur un
    élément avec le bouton
    gauche de la souris.
  #PB_EventType_RightDoubleClick:
    L'utilisateur a
    double-cliqué sur un
    élément avec le bouton
    droit de la souris.
  #PB_EventType_DragStart
          : L'utilisateur a
    essayé de lancer une
    opération 'Glisser &
    Déposer'.
```

Après un évènement #PB_EventType_DragStart, la bibliothèque Drag & Drop peut être utilisée pour commencer une opération de 'Glisser & Déposer'.

Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "ExplorerListGadget", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  ExplorerListGadget(0, 10, 10, 380, 180, "*.*", #PB_Explorer_MultiSelect)
  Repeat : Until WaitWindowEvent() = #PB_Event_CloseWindow
EndIf
```



Voir aussi

AddGadgetColumn() , RemoveGadgetColumn() , GetGadgetText() , SetGadgetText() , GetGadgetState() , GetGadgetItemState() , SetGadgetItemState() , GetGadgetItemText() , SetGadgetItemText() , CountGadgetItems() , GetGadgetAttribute() , SetGadgetAttribute() , GetGadgetItemAttribute() , SetGadgetItemAttribute() ,SetGadgetColor() , GetGadgetColor() , ExplorerComboGadget() , ExplorerTreeGadget() , ListIconGadget()

OS Supportés

Tous

99.21 ExplorerTreeGadget

Syntaxe

```
Resultat =
    ExplorerTreeGadget(#Gadget,
    X, Y, Largeur, Hauteur,
    Repertoire$, [, Options])
```

Description

Créé un gadget qui affiche une liste des dossiers.
Il permet à l'utilisateur de choisir un fichier ou un dossier situé à n'importe quel endroit des disques.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Repertoire$ Le répertoire initial affiché.

'Repertoire$' est l'emplacement de départ lorsque le gadget sera créé. Il peut être composé de motifs ('patterns'), tels que :
"C :*.pb;*.pbi". Si aucun 'pattern' n'est spécifié, le répertoire doit se terminer par un '\'. Si 'Repertoire$' est nul (chaîne vide), alors la liste des disques disponibles sera affichée.

Options (optionnel) Peut être une combinaison de :

```
#PB_Explorer_BorderLess
          : Crée un
 #Gadget sans bordure.
#PB_Explorer_AlwaysShowSelection:
 Affiche la sélection
 même quand le gadget
 n'est plus actif.
#PB_Explorer_NoLines
            : Cache les
 lignes de liaisons entre
 les noeuds '+'.
#PB_Explorer_NoButtons
           : Cache les
 noeuds '+'.

#PB_Explorer_NoFiles
            : Aucun
 fichier ne sera affiché.
#PB_Explorer_NoDriveRequester
   : Aucun message du
 type 'Insérer un disque
 dans le lecteur A:' ne
 sera affiché.
#PB_Explorer_NoMyDocuments
      : Le répertoire
 spécial 'Mes documents'
 ne sera pas affiché
 séparément.
#PB_Explorer_AutoSort
            : Le contenu
```

```
    sera automatiquement
    trié par nom.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

```
  - GadgetToolTip()
 permet d'ajouter une 'mini
   aide' à ce gadget.

 Les commandes suivantes sont
   disponibles pour contrôler
   le gadget:
  - GetGadgetText()
 : Renvoie le chemin complet
   du répertoire ou fichier
   sélectionné.
  - SetGadgetText()
 : Change l'élément
   actuellement sélectionné.
  - GetGadgetState()
: Permet de savoir si
   l'élément sélectionné est
   un répertoire ou un
   fichier.

  - SetGadgetColor()
 et GetGadgetColor()
 avec les valeurs suivantes
   comme 'TypeCouleur':
    #PB_Gadget_FrontColor:
   Couleur du texte
    #PB_Gadget_BackColor :
   Couleur du fond
    #PB_Gadget_LineColor :
   Couleur de la grille et
   des marqueurs si l'option
   #PB_Explorer_GridLines est
   utilisée.

  Note: SetGadgetColor()
 n'est pas pris en charge sur
   la plate-forme MacOS X.
```

ExplorerTreeGadget() génère les évènements suivants, renvoyés par EventType() :

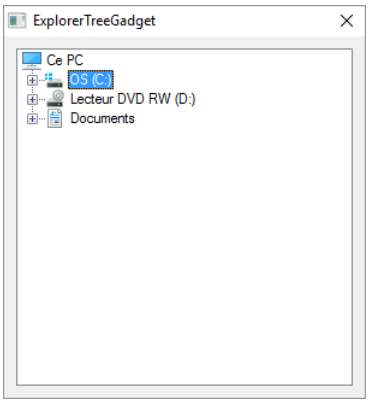
```
  #PB_EventType_Change
            : La sélection de
   l'élément courant vient de
   changer.
```

```
#PB_EventType_LeftClick
        : L'utilisateur a
  cliqué sur un élément avec
  le bouton gauche de la
  souris.
#PB_EventType_RightClick
       : L'utilisateur a
  cliqué sur un élément avec
  le bouton droit de la
  souris.
#PB_EventType_LeftDoubleClick
  : L'utilisateur a
  double-cliqué sur un
  élément avec le bouton
  gauche de la souris.
#PB_EventType_RightDoubleClick:
  L'utilisateur a
  double-cliqué sur un
  élément avec le bouton
  droit de la souris.
#PB_EventType_DragStart
        : L'utilisateur a
  essayé de lancer une
  opération 'Glisser &
  Déposer'.
```

Après un évènement #PB_EventType_DragStart , la bibliothèque Drag & Drop peut être utilisée pour commencer une opération de 'Glisser & Déposer'.

Exemple

```
1  If OpenWindow(0, 0, 0, 300,
    300, "ExplorerTreeGadget",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2    ExplorerTreeGadget(0, 10,
    10, 280, 280, "*.pb;*.pbi")
3    Repeat : Until
    WaitWindowEvent() =
    #PB_Event_CloseWindow
4  EndIf
```



Voir aussi

GetGadgetText() , SetGadgetText() , GetGadgetState() , GetGadgetColor() , SetGadgetColor() , ExplorerComboGadget() , ExplorerListGadget() , TreeGadget()

OS Supportés

Tous

99.22 FrameGadget

Syntaxe

```
Resultat =
    FrameGadget(#Gadget, X, Y,
    Largeur, Hauteur, Texte$
    [, Options])
```

Description

Crée dans la GadgetList un cadre en relief qui permet de regrouper un ensemble de gadgets.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Texte$ Titre qui s'affiche près du bord supérieur gauche du cadre (ce texte est ignoré si le gadget a l'option #PB_Frame_Single, #PB_Frame_Double ou #PB_Frame_Flat).

Options (optionnel) Peut être une combinaison de :

```
#PB_Frame_Single: Cadre
 enfoncé (Windows
 seulement)
#PB_Frame_Double: Cadre
 doublement enfoncé
 (Windows seulement)
#PB_Frame_Flat  : Cadre
 simple (Windows
 seulement)
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

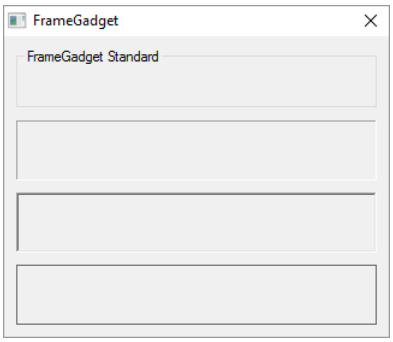
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

Comme ce gadget est seulement décoratif, GadgetToolTip() ne peut pas être utilisé (ce gadget ne reçoit aucun évènement).

Exemple

```
1  If OpenWindow(0, 0, 0, 320, 250, "FrameGadget", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2    FrameGadget(0, 10, 10, 300, 50, "Cadre en relief standard")
3    FrameGadget(1, 10, 70, 300, 50, "", #PB_Frame_Single)
4    FrameGadget(2, 10, 130, 300, 50, "", #PB_Frame_Double)
5    FrameGadget(3, 10, 190, 300, 50, "", #PB_Frame_Flat)
6
7    Repeat
8    Until WaitWindowEvent() = #PB_Event_CloseWindow
9  EndIf
```



Voir aussi

GetGadgetText() , SetGadgetText() , ContainerGadget()

OS Supportés

Tous

99.23 FreeGadget

Syntaxe

```
FreeGadget(#Gadget)
```

Description

Supprime un gadget de la fenêtre à laquelle il appartient et le détruit.
Pour un gadget container, sa 'gadgetlist' est détruite aussi.

Arguments

#Gadget Le gadget à détruire.
Si #PB_All est spécifié, tous les gadgets restants sont libérés.

Valeur de retour

Aucune.

Remarques

Un gadget est libéré automatiquement si l'un des évènements suivants se produit :
- La fenêtre qui contient le gadget est fermée .
- Le gadget parent (ContainerGadget() , PanelGadget() , etc) est libéré.
- Le programme se termine.

Voir aussi

IsGadget() , CloseWindow()

OS Supportés

Tous

99.24 GadgetID

Syntaxe

```
Resultat = GadgetID(#Gadget)
```

Description

Renvoie l'identifiant propre à l'OS d'un gadget.

Arguments

#Gadget Le gadget à utiliser.

Valeur de retour

Renvoie le numéro d'identification du gadget.
Ce résultat est parfois aussi appelé 'Handle'.
Voir le chapitre Numéros et Identifiants (Handles) pour plus d'informations.

Remarques

Utile pour utiliser les commandes de l'OS (API) directement avec le gadget.

OS Supportés

Tous

99.25 GadgetItemID

Syntaxe

```
Resultat =
    GadgetItemID(#Gadget,
    Element)
```

Description

Renvoie l'identifiant propre à l'OS de l'élément d'un gadget.

Arguments

#Gadget Le gadget à utiliser.

Element L'élément dont le handle doit être renvoyé.
Le premier élément dans le gadget commence à 0.

Valeur de retour

Renvoie le Handle de l'élément ou 0 en cas d'échec.

Remarques

Cette fonction est pour l'instant seulement supportée par le TreeGadget() sous Windows.
Elle renvoie 0 dans les autres cas.
C'est utile pour pouvoir utiliser l'API avec les éléments des gadgets PureBasic.

Voir aussi

TreeGadget()

OS Supportés

Windows

99.26 GadgetToolTip

Syntaxe

```
GadgetToolTip(#Gadget, Texte$)
```

Description

Associe ou remplace une bulle d'aide à un gadget.
Un 'Tooltip' est un texte flottant qui apparaît au bout d'un certain temps lorsque le curseur de la souris est immobile au dessus d'un gadget.

Arguments

#Gadget Le gadget à utiliser

Texte$ Le texte flottant ou bulle d'aide.

Valeur de retour

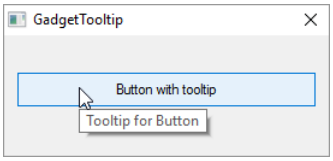
Aucune.

Remarques

Les gadgets suivants sont supportés :
- ButtonGadget()
- ButtonImageGadget()
- CalendarGadget()
- CanvasGadget()
- CheckBoxGadget()
- ComboBoxGadget()
- ContainerGadget()
- DateGadget()
- EditorGadget()
- ExplorerListGadget()
- ExplorerTreeGadget()
- HyperLinkGadget()
- ImageGadget()
- IPAddressGadget()
- ListIconGadget()
- ListViewGadget()
- MDIGadget()
- OpenGLGadget()
- OptionGadget()
- PanelGadget()
- ProgressBarGadget()
- ScrollBarGadget()
- ShortcutGadget()
- SpinGadget()
- SplitterGadget()
- StringGadget()
- TrackBarGadget()
- TreeGadget()

Exemple

```
1  If OpenWindow(0, 0, 0, 270,
    100, "GadgetToolTip",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2    ButtonGadget(0, 10, 30,
    250, 30, "Bouton avec
    texte flottant")
3    GadgetToolTip(0, "Texte
    flottant pour bouton")
4    Repeat : Until
    WaitWindowEvent() =
    #PB_Event_CloseWindow
5  EndIf
```



OS Supportés

Tous

99.27 GadgetX

Syntaxe

```
Resultat = GadgetX(#Gadget [,
    Mode])
```

Description

Renvoie la position en X d'un gadget.

Arguments

#**Gadget** Le gadget à utiliser

Mode (optionnel)
#PB_Gadget_ContainerCoordinate: La position en X du gadget (en pixels) dans son conteneur, ou dans une fenêtre (par défaut). #PB_Gadget_WindowCoordinate : La position absolue en X du gadget (en pixels) dans la fenêtre. #PB_Gadget_ScreenCoordinate : La position absolue en X du gadget (en pixels) dans l'écran. Il peut être utile d'afficher quelque chose sur le gadget comme une

fenêtre flottante ou un menu contextuel.

Valeur de retour

Renvoie la position en X par rapport au gadget parent ou à la fenêtre, en pixels.

Voir aussi

GadgetY() , GadgetWidth() , GadgetHeight() , ResizeGadget()

OS Supportés

Tous

99.28 GadgetY

Syntaxe

```
Resultat = GadgetY(#Gadget [,
    Mode])
```

Description

Renvoie la position en Y d'un gadget.

Arguments

#Gadget Le gadget à utiliser

Mode (optionnel)

#PB_Gadget_ContainerCoordinate : La position en Y du gadget (en pixels) dans son conteneur, ou dans une fenêtre (par défaut).

#PB_Gadget_WindowCoordinate : La position absolue en Y du gadget (en pixels) dans la fenêtre.

#PB_Gadget_ScreenCoordinate : La position absolue en Y du gadget (en pixels) dans l'écran. Il peut être utile d'afficher quelque chose sur le gadget comme une fenêtre flottante ou un menu contextuel.

Valeur de retour

Renvoie la position en Y par rapport au gadget parent ou à la fenêtre, en pixels.

Voir aussi

GadgetX() , GadgetWidth() , GadgetHeight() , ResizeGadget()

OS Supportés

Tous

99.29 GadgetHeight

Syntaxe

```
Resultat =
    GadgetHeight(#Gadget [,
    Mode])
```

Description

Renvoie la hauteur d'un gadget.

Arguments

#**Gadget** Le gadget à utiliser

Mode (optionnel) Peut-être l'une des valeurs suivantes :

```
#PB_Gadget_ActualSize  :
 Renvoie la hauteur
 actuelle du gadget, en
 pixels (par défaut).
#PB_Gadget_RequiredSize:
 Renvoie la hauteur
 nécessaire pour bien
 afficher le gadget, en
 pixels.
                          Seulement
 pour les gadgets qui ont
 l'obligation d'afficher
 du texte:
                          ButtonGadget,
 ButtonImageGadget,
 CheckBoxGadget,
 EditorGadget,
 HyperLinkGadget,
                          ImageGadget,
 OptionGadget, TextGadget
```

Valeur de retour

Renvoie la hauteur du gadget en pixels.

Voir aussi

GadgetWidth() , GadgetX() , GadgetY() , ResizeGadget()

OS Supportés

Tous

99.30 GadgetType

Syntaxe

```
Resultat = GadgetType(#Gadget)
```

Description

Renvoie le type d'un gadget.

Arguments

#**Gadget** Le gadget à utiliser

Valeur de retour

Renvoie une des valeurs suivantes :

```
#PB_GadgetType_Button
         : ButtonGadget()

#PB_GadgetType_ButtonImage
    : ButtonImageGadget()

#PB_GadgetType_Calendar
       : CalendarGadget()

#PB_GadgetType_Canvas
         : CanvasGadget()

#PB_GadgetType_CheckBox
       : CheckBoxGadget()

#PB_GadgetType_ComboBox
       : ComboBoxGadget()

#PB_GadgetType_Container
      : ContainerGadget()

#PB_GadgetType_Date
           : DateGadget()

#PB_GadgetType_Editor
         : EditorGadget()

#PB_GadgetType_ExplorerCombo
  : ExplorerComboGadget()

#PB_GadgetType_ExplorerList
   : ExplorerListGadget()

#PB_GadgetType_ExplorerTree
   : ExplorerTreeGadget()

#PB_GadgetType_Frame
          : FrameGadget()
```

```
#PB_GadgetType_HyperLink
      : HyperLinkGadget()

#PB_GadgetType_Image
          : ImageGadget()

#PB_GadgetType_IPAddress
      : IPAddressGadget()

#PB_GadgetType_ListIcon
       : ListIconGadget()

#PB_GadgetType_ListView
       : ListViewGadget()

#PB_GadgetType_MDI
            : MDIGadget()

#PB_GadgetType_OpenGL
         : OpenGLGadget()

#PB_GadgetType_Option
         : OptionGadget()

#PB_GadgetType_Panel
          : PanelGadget()

#PB_GadgetType_ProgressBar
    : ProgressBarGadget()

#PB_GadgetType_Scintilla
      : ScintillaGadget()

#PB_GadgetType_ScrollArea
     : ScrollAreaGadget()

#PB_GadgetType_ScrollBar
      : ScrollBarGadget()

#PB_GadgetType_Shortcut
       : ShortcutGadget()

#PB_GadgetType_Spin
           : SpinGadget()

#PB_GadgetType_Splitter
       : SplitterGadget()

#PB_GadgetType_String
         : StringGadget()

#PB_GadgetType_Text
           : TextGadget()

#PB_GadgetType_TrackBar
       : TrackBarGadget()
```

```
#PB_GadgetType_Tree
          : TreeGadget()

#PB_GadgetType_Web
           : WebGadget()

#PB_GadgetType_Unknown
        : Type inconnu,
 probablement pas un gadget
 PureBasic.
```

Remarques

Cette commande peut être utile pour écrire des fonctions génériques qui fonctionnent avec plusieurs types de gadgets.

OS Supportés

Tous

99.31 GadgetWidth

Syntaxe

```
Resultat =
    GadgetWidth(#Gadget [,
    Mode])
```

Description

Renvoie la largeur d'un gadget.

Arguments

#Gadget Le gadget à utiliser

Mode (optionnel) Peut-être l'une des valeurs suivantes :

- `#PB_Gadget_ActualSize` : Renvoie la largeur actuelle du gadget, en pixels (par défaut).
- `#PB_Gadget_RequiredSize`: Renvoie la largeur nécessaire pour bien afficher le gadget, en pixels. Seulement pour les gadgets qui ont l'obligation d'afficher du texte: `ButtonGadget`, `ButtonImageGadget`, `CheckBoxGadget`, `EditorGadget`, `HyperLinkGadget`, `ImageGadget`, `OptionGadget`, `TextGadget`

Valeur de retour

Renvoie la largeur du gadget en pixels.

Voir aussi

GadgetHeight() , GadgetX() , GadgetY() , ResizeGadget()

OS Supportés

Tous

99.32 GetActiveGadget

Syntaxe

```
Resultat = GetActiveGadget()
```

Description

Renvoie le numéro d'un gadget qui a actuellement le 'focus' clavier.

Arguments

Aucun.

Valeur de retour

Renvoie le numéro #Gadget du gadget qui a le focus.
Si aucun gadget n'a le focus, -1 est renvoyé.

Exemple

```
If OpenWindow(0, 0, 0, 270, 70, "GetActiveGadget", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  StringGadget  (0, 10, 10, 250, 20, "Appuyez sur la touche [Echap]...")
  StringGadget  (1, 10, 40, 250, 20, "Appuyez sur la touche [Echap]...")
  AddKeyboardShortcut(0, #PB_Shortcut_Escape, 1)
  SetActiveGadget(0)
  Repeat
    Evenement = WaitWindowEvent()
    If Evenement = #PB_Event_Menu And EventMenu() = 1
      MessageRequester("Test", "La touche [Echap] a été appuyée dans le gadget " + Str(GetActiveGadget()))
```

```
10      EndIf
11    Until Evenement =
      #PB_Event_CloseWindow
12  EndIf
```

OS Supportés

Tous

99.33 GetGadgetAttribute

Syntaxe

```
Resultat =
    GetGadgetAttribute(#Gadget,
    Attribut)
```

Description

Renvoie la valeur de l'attribut d'un gadget.

Arguments

#Gadget Le gadget à utiliser
Attribut L'attribut à tester.

Valeur de retour

Renvoie la valeur de l'attribut spécifié ou 0 si le gadget ne supporte pas l'attribut.

Remarques

Cette fonction est disponible avec les gadgets suivants :
- ButtonImageGadget()
- CalendarGadget()
- CanvasGadget()
- DateGadget()
- EditorGadget()
- ExplorerListGadget()
- ListIconGadget()
- OpenGLGadget()
- PanelGadget()
- ProgressBarGadget()
- ScrollAreaGadget()
- ScrollBarGadget()
- SpinGadget()
- SplitterGadget()
- StringGadget()
- TrackBarGadget()
- WebGadget()

Voir aussi

SetGadgetAttribute() , GetGadgetItemAttribute() , SetGadgetItemAttribute()

OS Supportés

Tous

99.34 GetGadgetColor

Syntaxe

```
Resultat =
    GetGadgetColor(#Gadget,
    TypeCouleur)
```

Description

Renvoie la couleur du 'TypeCouleur' d'un gadget.

Arguments

#Gadget Le gadget à utiliser

TypeCouleur Peut prendre l'une des valeurs suivantes (tous les gadgets ne supportent pas toutes ces valeurs) :

```
    #PB_Gadget_FrontColor
      : Texte du gadget
    #PB_Gadget_BackColor
      : Fond du gadget
    #PB_Gadget_LineColor
      : Couleur de la grille
    #PB_Gadget_TitleFrontColor:
      Couleur du texte dans le
      titre (pour
      CalendarGadget()
  )
    #PB_Gadget_TitleBackColor
      : Couleur du fond dans
      le titre  (pour
      CalendarGadget()
  )
    #PB_Gadget_GrayTextColor
      : Couleur du texte
      inactif       (pour
      CalendarGadget()
  )
```

Valeur de retour

Renvoie la couleur courante.
Cette fonction renvoie la couleur qui a été précédemment définie par SetGadgetColor() .
Si aucune couleur personnalisée n'est définie pour le #Gadget et TypeCouleur alors la fonction renvoie #PB_Default.

Remarques

Cette commande est supportée par les gadgets suivants :
- CalendarGadget()
- ContainerGadget()
- DateGadget()
- EditorGadget()
- ExplorerListGadget()
- ExplorerTreeGadget()
- HyperLinkGadget()
- ListViewGadget()
- ListIconGadget()
- MDIGadget()
- ProgressBarGadget()
- ScrollAreaGadget()
- SpinGadget()
- StringGadget()
- TextGadget()
- TreeGadget() **Note :** Avec le support des skins activé sur Windows XP, les couleurs personnalisées ne seront probablement pas prises en compte sur certains gadgets.

Voir aussi

SetGadgetColor() , GetGadgetItemColor() , SetGadgetItemColor()

OS Supportés

Tous

99.35 GetGadgetData

Syntaxe

```
Resultat =
    GetGadgetData(#Gadget)
```

Description

Renvoie la valeur qui a été stockée dans un gadget.

Arguments

#**Gadget** Le gadget à utiliser

Valeur de retour

Renvoie la valeur courante de la donnée stockée avec la commande SetGadgetData() .
Sans donnée définie pour ce gadget, la valeur de retour est 0.

Remarques

Cela permet d'associer une valeur personnalisée pour chaque gadget.
Cette commande fonctionne avec tous les types de gadget du PureBasic.
Pour un exemple d'utilisation, consulter la définition de la commande SetGadgetData() .

Voir aussi

SetGadgetData() , GetGadgetItemData() , SetGadgetItemData()

OS Supportés

Tous

99.36 GetGadgetFont

Syntaxe

```
Resultat =
    GetGadgetFont(#Gadget)
```

Description

Renvoie le numéro d'identification (FontID) de la police associée au #Gadget.

Arguments

#Gadget Le gadget à utiliser.
Si la constante #PB_Default est utilisée à la place du numéro de #Gadget, l'identifiant par défaut de la police est renvoyé lors de la création de nouveaux gadgets.

Valeur de retour

Renvoie le numéro d'identification pour le gadget spécifié ou celui utilisé pour les gadgets nouvellement créés.
Note : Sur OS X, cette fonction renvoie 0 si le gadget ne dispose pas d'une police spécifique qui lui est associée.

OS Supportés

Tous

99.37 GetGadgetItemAttribute

Syntaxe

```
Resultat =
    GetGadgetItemAttribute(#Gadget,
    Element, Attribut [,
    Colonne])
```

Description

Renvoie la valeur de l'attribut d'un élément d'un gadget.

Arguments

#Gadget Le gadget à utiliser.

Element L'élément spécifié.
Le premier élément dans le gadget commence à 0.

Attribut L'attribut à tester.

Colonne (optionnel) La colonne spécifiée.
La première colonne commence à 0.
La valeur par défaut est 0 colonne.

Valeur de retour

Renvoie la valeur de l'attribut, zéro si l'élément ou l'attribut n'existe pas.

Remarques

Cette fonction est disponible pour tous les gadgets et éléments suivants :

```
  - ExplorerListGadget()

    #PB_Explorer_ColumnWidth:
    Renvoie la largeur de la
    'Colonne' spécifiée. Le
    paramètre 'Element' est
    ignoré.

  - ListIconGadget()

    #PB_ListIcon_ColumnWidth:
    Renvoie la largeur de la
    'Colonne' spécifiée. Le
    paramètre 'Element' est
    ignoré.

  - TreeGadget()

    #PB_Tree_SubLevel:
    Renvoie le niveau de
    l'élément indiqué
     Le niveau des éléments du
    TreeGadget()
 peut être utilisé pour
    déterminer la relation
```

```
 entre deux éléments de
l'arbre. Par exemple, le
premier élément avec un
niveau inférieur à
l'élément courant
 (si la liste est examinée
en remontant) est
l'élément 'parent'.
```

Voir aussi

SetGadgetItemAttribute() ,
GetGadgetAttribute() ,
SetGadgetAttribute()

OS Supportés

Tous

99.38 GetGadgetItemColor

Syntaxe

```
Resultat =
    GetGadgetItemColor(#Gadget,
    Element, TypeCouleur [,
    Colonne])
```

Description

Renvoie la couleur d'un élément d'un gadget.

Arguments

#Gadget Le gadget à utiliser.

Element L'élément à tester.
Le premier élément dans le gadget commence à 0.

TypeCouleur Le paramètre 'TypeCouleur' peut prendre l'une des valeurs suivantes :

```
#PB_Gadget_FrontColor:
 Texte de l'élément.
#PB_Gadget_BackColor :
 Fond de l'élément.
```

Colonne (optionnel) La colonne spécifiée.
La première colonne commence à 0.
La valeur par défaut est 0 colonne.

Valeur de retour

Renvoie la couleur courante.
Cette fonction renvoie la couleur qui a été précédemment définie par

SetGadgetItemColor() . Si aucune couleur personnalisée n'est définie pour le #Gadget et TypeCouleur alors la fonction renvoie #PB_Default.

Remarques

Cette commande est supportée par les gadgets suivants :
- ListIconGadget()
- TreeGadget() **Note :** Avec le support des skins activé sur Windows XP, les couleurs personnalisées ne seront probablement pas prises en compte sur certains gadgets.

Voir aussi

SetGadgetItemColor() , GetGadgetColor() , SetGadgetColor()

OS Supportés

Tous

99.39 GetGadgetItemData

Syntaxe

```
Resultat =
    GetGadgetItemData(#Gadget,
    Element)
```

Description

Renvoie la valeur qui a été stockée dans un élément d'un gadget .

Arguments

#Gadget Le gadget à utiliser

Element L'élément à partir duquel les données doivent être lues.
Le premier élément dans le gadget commence à 0.

Valeur de retour

Renvoie la donnée stockée avec la commande SetGadgetItemData() .
Si aucune valeur n'a été stockée avec l'élément alors la valeur de retour est 0.

Remarques

Cela permet d'associer une valeur personnalisée pour chaque élément d'un gadget.

Cette commande est supportée par les gadgets suivants :
- ComboBoxGadget()
- ListIconGadget()
- ListViewGadget()
- PanelGadget()
- TreeGadget()
Pour un exemple d'utilisation, consulter la définition de la commande SetGadgetItemData() .

Voir aussi

SetGadgetItemData() , GetGadgetData() , SetGadgetData()

OS Supportés

Tous

99.40 GetGadgetState

Syntaxe

```
Resultat =
    GetGadgetState(#Gadget)
```

Description

Renvoie l'état actuel d'un gadget.

Arguments

#**Gadget** Le gadget à utiliser.

Valeur de retour

Renvoie l'état du gadget.
Voir ci-dessous en fonction du type gadget.

Remarques

Il s'agit d'une fonction universelle qui fonctionne avec presque tous les gadgets :

```
  - ButtonGadget()
     : renvoie l'état d'un
   bouton à bascule
   (#PB_Button_Toggle): 1 =
   pressé, 0 = normal.
  - ButtonImageGadget()
: renvoie l'état d'un bouton
   à bascule
   (#PB_Button_Toggle): 1 =
   pressé, 0 = normal.
  - CalendarGadget()
   : renvoie la date
   actuellement sélectionnée.
```

```
- CheckBoxGadget()
 : renvoie une des valeurs
 suivantes:
  #PB_CheckBox_Checked  :
 La case est cochée.
  #PB_CheckBox_Unchecked:
 La case est décochée.
  #PB_CheckBox_Inbetween:
 L'état de la case à cocher
 est indéterminé (Seulement
 pour les cases à cocher de
 type
 #PB_CheckBox_ThreeState).

- ComboBoxGadget()
  : Renvoie le numéro de
 l'élément sélectionné ou
 -1 si pas de sélection.
- DateGadget()
      : Renvoie la date ou
 l'heure actuellement
 sélectionnée. Si
 #PB_Date_CheckBox
                        est
 utilisé et que la case à
 cocher est décochée, 0
 sera renvoyé.
- ExplorerListGadget()
: Renvoie le numéro de
 l'élément sélectionné ou
 -1 si pas de sélection.
- ExplorerTreeGadget()
: Renvoie le type de
 l'élément sélectionné
 (#PB_Explorer_File ou
 #PB_Explorer_Directory).
- ImageGadget()
     : Renvoie l'ImageID de
 l'image actuellement
 affichée.
- IPAddressGadget()
 : Renvoie l'adresse IP
 courante.
- ListIconGadget()
  : Renvoie le numéro du
 premier élément
 sélectionné ou -1 s'il n'y
 a pas d'élément
 sélectionné.
- ListViewGadget()
  : Renvoie le numéro de
 l'élément sélectionné ou
 -1 s'il n'y a pas
 d'élément sélectionné.
- MDIGadget()
       : Renvoie le numéro
 de la fenêtre fille
 active, ou -1 si aucune
 n'est active.
```

```
  - OptionGadget()
      : Renvoie 1 s'il est
    activé, 0 sinon.
  - PanelGadget()
       : Renvoie le numéro
    d'onglet sélectionné ou -1
    si pas de sélection.
  - ProgressBarGadget()
: Renvoie la valeur actuelle
    de la barre de progression.
  - ScrollBarGadget()
    : Renvoie la position
    actuelle de l'ascenseur.
  - ShortcutGadget()
     : Renvoie le raccourci
    clavier sélectionné.
  - SpinGadget()
        : Renvoie la valeur
    actuelle du SpinGadget.
  - SplitterGadget()
     : Renvoie la position
    actuelle de la barre de
    séparation, en pixels.
  - TrackBarGadget()
     : Renvoie la position
    actuelle de la TrackBar
    (valeur comprise entre la
    valeur minimale et
    maximale).
  - TreeGadget()
        : Renvoie le numéro
    de l'élément sélectionné
    ou  -1 si pas de sélection.
```

Voir aussi

SetGadgetState() , GetGadgetItemState() , SetGadgetItemState()

OS Supportés

Tous

99.41 GetGadgetItemText

Syntaxe

```
Resultat\$ =
    GetGadgetItemText(#Gadget,
    Element [, Colonne])
```

Description

Renvoie le texte d'un élément d'un gadget.

Arguments

#Gadget Le gadget à utiliser

Element L'élément à tester.
Le premier élément dans le gadget commence à 0.

Colonne (optionnel) La colonne à utiliser pour les gadgets qui prennent en charge plusieurs colonnes.
La première colonne commence à 0.
La valeur par défaut est 0 colonne.

Valeur de retour

Renvoie le texte de l'élément du gadget ou une chaîne vide en cas d'erreur.

Remarques

Il s'agit d'une fonction universelle qui fonctionne avec presque tous les gadgets comportant des éléments :

```
  - ComboBoxGadget()
    : Renvoie le texte de
   l'élément de la liste
   déroulante ('Colonne' est
   ignorée).
  - EditorGadget()
      : Renvoie le texte de
   l'élément (la ligne) de
   l'editor ('Colonne' est
   ignorée).
  - ExplorerListGadget()
: Renvoie le nom du
   l'élément, sans son chemin
   complet.
                          Si
   'Element' = -1, alors le
   titre de la colonne est
   renvoyé.
  - ListIconGadget()
    : Renvoie le texte de
   l'élément de la liste.
                          'Colonne'
   est l'index de la colonne
   (commence à partir de 0).
                          Si
   'Element' = -1, alors le
   titre de la colonne est
   renvoyé.
  - ListViewGadget()
    : Renvoie le texte de
   l'élément de la liste
   ('Colonne' est ignorée).
  - MDIGadget()
        : Renvoie le titre
   de la fenêtre fille
   (élément) spécifiée
   ('Colonne' est ignorée).
  - PanelGadget()
```

```
      : Renvoie le texte de
   l'élément spécifié
   ('Colonne' est ignorée).
 - TreeGadget()
       : Renvoie le texte de
   l'élément d'arbre
   ('Colonne' est ignorée).
 - WebGadget()
        : Renvoie le code
   html, le titre de la page,
   message statut ou la
   sélection actuelle.
```

Note : La numérotation des éléments commence à partir de 0.

Voir aussi

SetGadgetItemText() , GetGadgetText() , SetGadgetText()

OS Supportés

Tous

99.42 GetGadgetItemState

Syntaxe

```
Resultat =
    GetGadgetItemState(#Gadget,
    Element)
```

Description

Renvoie l'état de l'élément d'un gadget.

Arguments

#Gadget Le gadget à utiliser

Element L'élément à tester.
Le premier élément dans le gadget commence à 0.

Valeur de retour

Renvoie l'état de l'élément du gadget ou 0 en cas d'erreur.
Voir ci-dessous pour connaître la signification de cette valeur en fonction du type du gadget.

Remarques

Il s'agit d'une fonction universelle qui fonctionne avec presque tous les gadgets comportant des éléments :

- CalendarGadget()
  : Renvoie
  #PB_Calendar_Bold quand la
  date spécifiée est
  affichée en gras, sinon
  #PB_Calendar_Normal
                          est
  renvoyé. 'Element' doit
  être une date
au format PureBasic.

- ExplorerListGadget()
: Renvoie une combinaison des
  valeurs suivantes:
   #PB_Explorer_File   :
  L'élément est un fichier.
   #PB_Explorer_Directory:
  L'élément est un
  répertoire (ou un disque).
   #PB_Explorer_Selected :
  L'élément est sélectionné.

- ListViewGadget()
  : Renvoie 1 si l'élément
  est sélectionné, 0 sinon.

- ListIconGadget()
  : Renvoie une combinaison
  des valeurs suivantes:
   #PB_ListIcon_Selected :
  L'élément est sélectionné
   #PB_ListIcon_Checked :
  L'élément a sa case à
  cocher associée cochée
  (créé avec l'option
  #PB_ListIcon_CheckBoxes)
   #PB_ListIcon_Inbetween:
  L'élément a sa case à
  cocher associée
  "indéterminée" (Option
  #PB_ListIcon_ThreeState).

- TreeGadget()
      : Renvoie une
  combinaison des valeurs
  suivantes:
   #PB_Tree_Selected :
  L'élément est sélectionné.
   #PB_Tree_Expanded :
  L'élément est déployé (la
  branche de l'arbre est
  ouverte).
   #PB_Tree_Collapsed:
  L'élément est fermé.
   #PB_Tree_Checked : La
  case à cocher de l'élément
  est cochée. (Si l'option
  #PB_Tree_CheckBoxes est
  utilisée)

#PB_Tree_Inbetween : L'élément a sa case à cocher associée "indéterminée" (Option #PB_ListIcon_ThreeState)

Pour vérifier si un état est actif, utiliser l'opérateur '&' :

```
If Resultat & #PB_Tree_Checked
  ; La 'case à cocher' de l'élément est cochée
EndIf
```

Exemple

L'exemple ci-dessous montre comment tester différentes combinaisons pour le gadget ListIconGadget() :

```
; ... Ce code est à placer dans une boucle avec WaitWindowEvent() :
If GetGadgetItemState(#Listicon, n) & #PB_ListIcon_Checked
  ; L'élément 'n' est coché (indépendamment qu'il soit sélectionné ou non)
EndIf

If GetGadgetItemState(#Listicon, n) & #PB_ListIcon_Selected
  ; L'élément 'n' est sélectionné (indépendamment qu'il soit coché ou non)
EndIf

If GetGadgetItemState(#Listicon, n) = #PB_ListIcon_Checked | #PB_ListIcon_Selected
  ; L'élément 'n' est coché et sélectionné
EndIf

If GetGadgetItemState(#Listicon, n) & (#PB_ListIcon_Checked | #PB_ListIcon_Selected)
  ; L'élément 'n' est coché ou sélectionné ou les deux
EndIf
```

Voir aussi

SetGadgetItemState() , GetGadgetState() , SetGadgetState()

OS Supportés

Tous

99.43 GetGadgetText

Syntaxe

```
Resultat\$ =
    GetGadgetText(#Gadget)
```

Description

Renvoie le texte contenu dans un gadget.

Arguments

#**Gadget** Le gadget à utiliser

Valeur de retour

Renvoie le texte du gadget, ou une chaîne vide si le gadget ne supporte la présence de texte.

Remarques

Cette commande est particulièrement utile pour :

```
  - ButtonGadget()
       : Renvoie le texte
    affiché par le
    BoutonGadget.
  - ComboBoxGadget()
     : Renvoie le texte de
    l'élément sélectionné.
  - EditorGadget()
       : Renvoie le texte
    contenu dans l'éditeur.
                             Notez
    que le retour à la ligne
    se fait avec "Chr (13) +
    Chr (10)"
                             sous
    Windows et avec "Chr (10)"
    sous Linux et OS X.
  - ExplorerComboGadget()
: Renvoie le répertoire
    affiché et sélectionné.
  - ExplorerListGadget()
 : Renvoie le répertoire
    actuellement affiché.
  - ExplorerTreeGadget()
```

```
: Renvoie le chemin complet
  du fichier ou répertoire
  sélectionné.
 - FrameGadget()
       : Renvoie le titre du
  FrameGadget.
 - HyperLinkGadget()
    : Renvoie le contenu de
  l'HyperLinkGadget.
 - ListIconGadget()
     : Renvoie le texte de la
  première colonne de
  l'élément sélectionné.
 - ListViewGadget()
     : Renvoie le texte de
  l'élément sélectionné.
 - StringGadget()
      : Renvoie le contenu
  du StringGadget.
 - TextGadget()
        : Renvoie le contenu
  du TextGadget.
 - TreeGadget()
        : Renvoie le texte
  de l'élément sélectionné.
 - WebGadget()
         : Renvoie l'URL du
  website affiché.
```

Voir aussi

SetGadgetText() , GetGadgetItemText() , SetGadgetItemText()

OS Supportés

Tous

99.44 HideGadget

Syntaxe

```
HideGadget(#Gadget, Etat)
```

Description

Cache ou affiche un gadget.

Arguments

#Gadget Le gadget à utiliser

Etat
```
#True : Le gadget
   sera caché.
 #False: Le gadget sera
   affiché.
```

Valeur de retour

Aucune.

Exemple

```
1  If OpenWindow(0,0,0,180,120,"Cacher un Gadget",
   #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2   ButtonGadget(0,10,10,160,50,"Bouton 1") : bouton = #True   ; le bouton est affiché
3   ButtonGadget(1,10,80,160,30,"Cacher le bouton 1")
4   Repeat
5     Evenement = WaitWindowEvent()
6     If Evenement = #PB_Event_Gadget
7       If EventGadget()=1
8         If bouton = #True      ; si le bouton est affiché
9           HideGadget(0,1)      ; cache le bouton
10          bouton = #False
11          SetGadgetText(1,"Afficher le bouton 1")
12        Else                   ; si le bouton est caché
13          HideGadget(0,0)      ; affiche le bouton
14          bouton = #True
15          SetGadgetText(1,"Cacher le bouton 1")
16        EndIf
17      EndIf
18    EndIf
19  Until Evenement = #PB_Event_CloseWindow
20 EndIf
```

Voir aussi

DisableGadget()

OS Supportés

Tous

99.45 HyperLinkGadget

Syntaxe

```
Resultat =
    HyperLinkGadget(#Gadget,
    X, Y, Largeur, Hauteur,
    Texte$, Couleur [,
    Options])
```

Description

Crée un lien cliquable, similaire à ceux que l'on trouve sur les pages web, dans la GadgetList courante.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Texte$ Le texte à afficher.

Couleur La couleur du texte lorsque la souris survole le gadget.
La couleur du texte pour l'état non survolé peut être modifié avec SetGadgetColor() .

Options (optionnel)

```
      #PB_Hyperlink_Underline:
    Affiche une ligne en
    dessous du texte sans
    avoir à utiliser une
    police soulignée.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

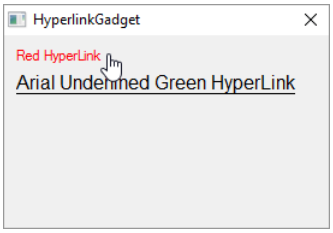
Remarques

```
  - GadgetToolTip()
 permet d'ajouter une 'mini
   aide' à ce gadget.

  - Le texte peut être
   manipulé à l'aide des
   commandes SetGadgetText()
 et GetGadgetText()
.
  - SetGadgetColor()
 et GetGadgetColor()
 avec la constante
   #PB_Gadget_FrontColor
```

```
    et #PB_Gadget_BackColor
   pour changer la couleur du
   texte et du fond (si la
   couleur de fond
    de la fenêtre a changé).

  Note: SetGadgetColor()
 n'est pas pris en charge sur
   la plate-forme MacOS X.
```

Exemple

```
1  If
    OpenWindow(0,0,0,270,160,"HyperLinkGadget",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2    HyperLinkGadget(0, 10,
    10,250,20,"Lien rouge",
    RGB(255,0,0))
3    HyperLinkGadget(1, 10,
    30,250,40,"Lien vert
    souligné (Police Arial)",
    RGB(0,255,0),
    #PB_Font_Underline)
4    SetGadgetFont(1,
    LoadFont(0, "Arial", 12))
5    Repeat : Until
    WaitWindowEvent()=#PB_Event_CloseWindow
6  EndIf
```



Voir aussi

GetGadgetText() , SetGadgetText() , GetGadgetColor() , SetGadgetColor()

OS Supportés

Tous

99.46 ImageGadget

Syntaxe

```
Resultat =
    ImageGadget(#Gadget, X, Y,
    Largeur, Hauteur, ImageID
    [, Options])
```

Description

Crée un gadget Image dans la GadgetList en cours.
Un gadget Image permet de positionner une image dans une fenêtre.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.
Le gadget ajuste sa largeur et sa hauteur pour s'adapter à l'image affichée.
La largeur et la hauteur spécifiées ne sont utilisés que si aucune image ne s'affiche.

ImageID L'image à afficher. Utiliser la fonction ImageID() pour obtenir l'ID d'une image.
Si ce paramètre est 0, aucune image ne sera affichée.

Options (optionnel)

```
      #PB_Image_Border :
    Affiche un cadre autour
    de l'image.
  #PB_Image_Raised : Affiche
    une bordure surélevée
    autour de l'image.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

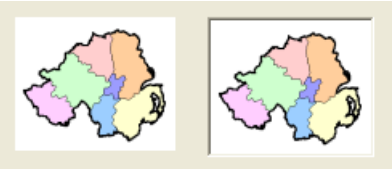
```
  - GadgetToolTip()
 permet d'ajouter une 'mini
   aide' à ce gadget.

  - SetGadgetState()
: Permet de changer
   dynamiquement l'image
   contenue dans le gadget.
    Un ImageID valide peut
   être obtenu via ImageID()
. Si 'ImageID' est 0, alors
   l'image est retirée du
   gadget.
```

```
  Les évènements suivants
   sont supportés par
   EventType()
:
    #PB_EventType_Change
            : L'élément
   courant a changé
    #PB_EventType_LeftClick
          : L'utilisateur a
   cliqué sur un élément avec
   le bouton gauche de la
   souris.
    #PB_EventType_RightClick
         : L'utilisateur a
   cliqué sur un élément avec
   le bouton droit de la
   souris.
    #PB_EventType_LeftDoubleClick
   : L'utilisateur a
   double-cliqué sur un
   élément avec le bouton
   gauche de la souris.
    #PB_EventType_RightDoubleClick:
   L'utilisateur a
   double-cliqué sur un
   élément avec le bouton
   droit de la souris.
    #PB_EventType_DragStart
          : L'utilisateur a
   essayé de lancer une
   opération 'Glisser &
   Déposer'.

  Après un évènement
   #PB_EventType_DragStart ,
   la bibliothèque Drag & Drop
 peut être
  utilisée pour commencer une
   opération 'Glisser &
   Déposer'.

  Si vous avez besoin de plus
   de types d'évènements ou
   de l'affichage avec
   double-tampon pour une
   mise à jour
  régulière de l'image alors
   jetez un oeil au canvas
   gadget CanvasGadget()
.
```

Exemple

```
1  If OpenWindow(0, 0, 0, 245,
     105, "ImageGadget",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2    If LoadImage(0,
     "map.bmp")    ; changez le
```

```
    2ème paramètre en
    indiquant le
    chemin/fichier contenant
    votre image
3       ImageGadget(0,  10, 10,
    100, 83, ImageID(0))
                          ;
    imagegadget standard
4       ImageGadget(1, 130, 10,
    100, 83, ImageID(0),
    #PB_Image_Border)     ;
    imagegadget avec cadre
5    EndIf
6    Repeat : Until
    WaitWindowEvent() =
    #PB_Event_CloseWindow
7  EndIf
```



Voir aussi

GetGadgetState() , SetGadgetState() , ButtonImageGadget() , ImageID() , CanvasGadget()

OS Supportés

Tous

99.47 IPAddressGadget

Syntaxe

```
Resultat =
    IPAddressGadget(#Gadget,
    X, Y, Largeur, Hauteur)
```

Description

Crée un gadget de saisie d'adresse IP dans la GadgetList courante.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Valeur de retour

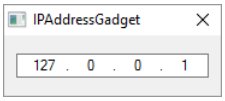
Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

```
  - GadgetToolTip()
 permet d'ajouter une 'mini
   aide' à ce gadget.

  Les commandes suivantes
   peuvent être utilisées
   pour contrôler le gadget:
  - GetGadgetState()
: Renvoie l'adresse IP
   courante (IPAddressField()
 est utile pour récupérer la
   valeur d'un champ).
  - SetGadgetState()
: Change l'adresse IP
   courante (MakeIPAddress()
 pour construire une IP
   valide).
  - GetGadgetText()
 : Renvoie l'adresse IP en
   format texte: par exemple
   "127.0.0.1".
  - SetGadgetText()
 : Efface complètement le
   gadget quand une chaîne
   vide est spécifiée. C'est
   la seule action valide
   pour l'instant.
```

Exemple

```
1  If OpenWindow(0, 0, 0, 210,
    50, "IPAddressGadget",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2    IPAddressGadget(0, 10,
    15, 160, 20)
3    SetGadgetState(0,
    MakeIPAddress(127, 0, 0,
    1))    ; affiche une
    adresse IP valide
4    Repeat : Until
    WaitWindowEvent() =
    #PB_Event_CloseWindow
5  EndIf
```



Voir aussi

GetGadgetState() , SetGadgetState() , GetGadgetText() , SetGadgetText() , IPAddressField() , IPString() , MakeIPAddress()

OS Supportés

Tous

99.48 IsGadget

Syntaxe

```
Resultat = IsGadget(#Gadget)
```

Description

Teste si un gadget est correctement initialisé.

Arguments

#Gadget Le gadget à utiliser

Valeur de retour

Renvoie une valeur non nulle si l'entrée est un gadget valide, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est une bonne façon de s'assurer qu'un gadget est prêt à l'emploi.

Voir aussi

FreeGadget()

OS Supportés

Tous

99.49 ListIconGadget

Syntaxe

```
Resultat =
    ListIconGadget(#Gadget, X,
    Y, Largeur, Hauteur,
    TitrePremiereColonne$,
    LargeurTitrePremiereColonne
    [, Options])
```

Description

Crée un gadget de listes avec icônes dans la GadgetList en cours.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

TitrePremiereColonne$ Le titre de la première colonne dans le gadget.
Le gadget est créé avec une colonne initiale.

LargeurTitrePremiereColonne La largeur de la première colonne dans le gadget.

Options (optionnel) Peut être une combinaison de :

```
#PB_ListIcon_CheckBoxes
          : Affiche une
  case à cocher dans la
  première colonne.
#PB_ListIcon_ThreeState
          : La case à
  cocher peut avoir un
  état 'indéterminé'.
#PB_ListIcon_MultiSelect
         : Active le mode
  'sélection multiple'.
#PB_ListIcon_GridLines
           : Affiche des
  lignes de séparation
  (non supporté sous Mac
  OS X).
#PB_ListIcon_FullRowSelect
       : La sélection
  s'étend à toute la ligne
  au lieu de la première
  colonne (Windows
  seulement).
#PB_ListIcon_HeaderDragDrop
      : L'ordre des
  colonnes peut être
  changé avec un
  glisser/déposer
  (drag'n'drop).
#PB_ListIcon_AlwaysShowSelection:
  Affiche la selection
  même quand le gadget
  n'est plus actif
  (Windows seulement).
```

L'option #PB_ListIcon_ThreeState peut être utilisée en combinaison

avec l'option #PB_ListIcon_CheckBoxes pour obtenir une case à cocher qui peut avoir un état "on", "off" et "indéterminé". L'utilisateur ne peut sélectionner que l'état "on" ou "off". L'état "indéterminé" peut être défini en utilisant la fonction SetGadgetItemState() .

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

```
  - GadgetToolTip()
 permet d'ajouter une 'mini
   aide' à ce gadget.

  Les fonctions suivantes
   peuvent être utilisées
   pour agir sur le contenu
   de cette liste :
  - AddGadgetColumn()
   : Ajoute une colonne au
   gadget.
  - RemoveGadgetColumn()
: Supprime une colonne (ainsi
   que ses données).
  - AddGadgetItem()
     : Ajoute un élément et
   éventuellement une image.
  - RemoveGadgetItem()
  : Supprime un élément.
  - ClearGadgetItems()
  : Supprime tous les
   éléments.
  - CountGadgetItems()
  : Renvoie le nombre
   d'éléments actuellement
   contenus dans le gadget.
  - GetGadgetItemColor()
: Renvoie la couleur du texte
   ou du fond de l'élément.
  - SetGadgetItemColor()
: Change la couleur du texte
   ou du fond de l'élément.
                          (Couleur
   de fond pas pris en charge
   sur MacOS X)
  - GetGadgetItemData()
 : Renvoie la valeur
   personnalisée associée à
```

```
    cet élément.
  - SetGadgetItemData()
 : Associe une valeur
    personnalisée à cet
    élément.
  - GetGadgetItemState()
: Renvoie l'état de l'élément
    spécifié.
                          Ce
    peut être une combinaison
    de #PB_ListIcon_Selected
                          et
    #PB_ListIcon_Checked si
    les 'cases à cocher' sont
    affichées
  - SetGadgetItemState()
: Change l'état de l'élément
    spécifié.
                          Ce
    peut être une combinaison
    de #PB_ListIcon_Selected
                          et
    #PB_ListIcon_Checked si
    les 'cases à cocher' sont
    affichées
  - GetGadgetItemText()
 : Renvoie le texte de
    l'élément spécifié.
                          Si
    'Element' = -1, alors le
    titre de la colonne est
    renvoyé.
  - SetGadgetItemText()
 : Change le texte de
    l'élément spécifié.
                          Si
    'Element' = -1, alors le
    titre de la colonne est
    modifié.
                          Comme
    avec AddGadgetItem()
, il est possible de définir
    le texte de
                          plusieurs
    colonnes à la fois, avec
    le séparateur Chr(10)
.
  - SetGadgetItemImage()
: Modifie l'image actuelle de
    l'élément spécifié.
  - GetGadgetState()
     : Renvoie le premier
    élément selectionné ou -1
    s'il n'y a pas d'élément
    sélectionné.
  - SetGadgetState()
     : Change l'élément
    sélectionné (tous les
    autres éléments sont alors
```

```
  désélectionnés).
                          Si
  -1 est utilisé, tous les
  éléments seront
  désélectionnés.
 - GetGadgetAttribute()
avec l'attribut suivant :
     #PB_ListIcon_ColumnCount:
  Renvoie le nombre de
  colonnes
     #PB_ListIcon_DisplayMode:
  Renvoie le mode
  d'affichage du gadget
  (Windows uniquement)

 - SetGadgetAttribute()
avec l'attribut suivant
  (seulement sous Windows):
     #PB_ListIcon_DisplayMode
  : Change le mode
  d'affichage du gadget.
          Le mode peut être
  l'une des constantes
  suivantes:
          #PB_ListIcon_LargeIcon
    : Mode grandes icônes
          #PB_ListIcon_SmallIcon
    : Mode petites icônes
          #PB_ListIcon_List
         : Mode Liste
          #PB_ListIcon_Report
       : Mode Détails
  (colonnes, mode par défaut)

 - GetGadgetItemAttribute()
/ SetGadgetItemAttribute()
: Avec l'attribut suivant:
     #PB_ListIcon_ColumnWidth
  : Renvoie / Change la
  largeur de la 'Colonne'
  spécifiée.
     Le paramètre 'Element'
  est ignoré.


 Si vous voulez ajouter du
  contenu à un gadget
  ListIcon contenant
  plusieurs colonnes en
  utilisant
 AddGadgetItem()
, utilisez le format
  "première
  colonne"+chr(10)+"deuxième
  colonne" comme paramètre
  Texte$.

 - SetGadgetColor()
et GetGadgetColor()
```

```
avec les valeurs
  'TypeCouleur' suivantes:
   #PB_Gadget_FrontColor:
  Couleur du texte
   #PB_Gadget_BackColor :
  Couleur du fond
   #PB_Gadget_LineColor :
  Couleur de la grille si
  l'option
  #PB_ListIcon_GridLines est
  utilisée.

 -  SetGadgetItemColor()
et GetGadgetItemColor()
avec les  valeurs
  'TypeCouleur' suivantes:
   #PB_Gadget_FrontColor:
  Texte de l'élément.
   #PB_Gadget_BackColor :
  Fond de l'élément.

 Note: SetGadgetColor()
n'est pas pris en charge sur
  la plate-forme MacOS X.
```

Les évènements suivants sont supportés par EventType() :

```
#PB_EventType_LeftClick
        : Clic avec le
  bouton gauche de la
  souris, ou une case à
  cocher a été utilisée.
#PB_EventType_LeftDoubleClick
   : Double-clic avec le
  bouton gauche de la souris
#PB_EventType_RightClick
       : Clic avec le
  bouton droit de la souris
#PB_EventType_RightDoubleClick
  : Double-clic avec le
  bouton droit de la souris
#PB_EventType_Change
           : L'élément
  sélectionné a été changé
#PB_EventType_DragStart
        : L'utilisateur a
  essayé de lancer une
  opération de 'Glisser &
  Déposer'.
```

Après un évènement #PB_EventType_DragStart, la bibliothèque Drag & Drop peut être utilisée pour commencer une opération 'Glisser & Déposer'.

Exemple

```
1  If OpenWindow(0, 100, 100,
     300, 100,
```

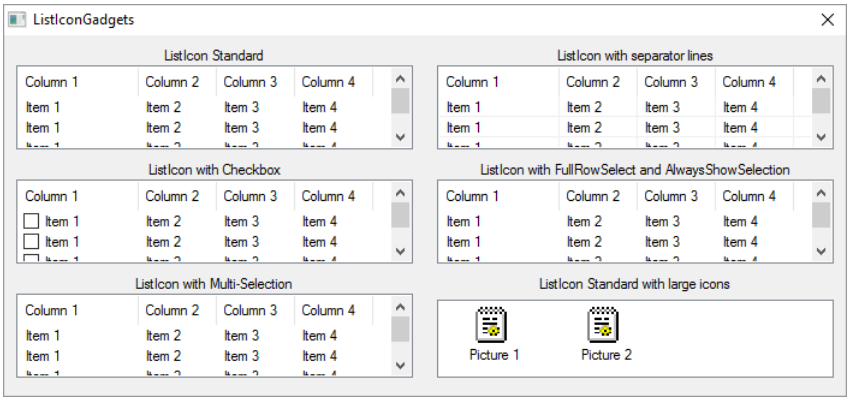
```
    "ListIconGadget",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2   ListIconGadget(0, 5, 5,
    290, 90, "Nom", 100,
    #PB_ListIcon_FullRowSelect
    |
    #PB_ListIcon_AlwaysShowSelection)
3   AddGadgetColumn(0, 1,
    "Adresse", 250)
4   AddGadgetItem(0, -1,
    "Harry Rannit"+Chr(10)+"12
    Parliament Way, Battle
    Street, By the Bay")
5   AddGadgetItem(0, -1,
    "Ginger
    Brokeit"+Chr(10)+"130
    PureBasic Road, BigTown,
    CodeCity")
6
7   Repeat
8     Evenement =
    WaitWindowEvent()
9   Until Evenement =
    #PB_Event_CloseWindow
10 EndIf
```

Exemple

```
1   ; Démonstration des options
    possibles pour le gadget
    liste avec icônes...
2  If OpenWindow(0, 0, 0, 700,
    300, "ListIconGadget",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
3    ; colonne de gauche
4    TextGadget    (6,  10,
    10, 330, 20, "Liste avec
    icônes standard",
    #PB_Text_Center)
5    ListIconGadget(0,  10,
    25, 330, 70, "Colonne 1",
    100)
6    TextGadget    (7,  10,
    105, 330, 20, "Liste avec
    icônes + cases à cocher",
    #PB_Text_Center)
7    ListIconGadget(1,  10,
    120, 330, 75, "Colonne 1",
    100,
    #PB_ListIcon_CheckBoxes)
    ; Liste avec icônes et
    cases à cocher
8    TextGadget    (8,  10,
    205, 330, 20, "Liste avec
    icônes + sélection
    multiple", #PB_Text_Center)
```

```
 ListIconGadget(2,  10,
220, 330, 70, "Colonne 1",
100,
#PB_ListIcon_MultiSelect)
; Liste avec icônes et
sélection multiple
 ; colonne de droite
 TextGadget    (9, 360,
10, 330, 20, "Liste avec
icônes + lignes de
séparation",#PB_Text_Center)
 ListIconGadget(3, 360,
25, 330, 70, "Colonne 1",
100,
#PB_ListIcon_GridLines)
 TextGadget   (10, 360,
105, 330, 20, "Liste avec
icônes + sélection de
ligne entière + sélection
toujours
affichée",#PB_Text_Center)
 ListIconGadget(4, 360,
120, 330, 75, "Colonne 1",
100,
#PB_ListIcon_FullRowSelect
|
#PB_ListIcon_AlwaysShowSelection)
 TextGadget   (11, 360,
205, 330, 20, "Liste avec
icônes standard + mode
grandes
icônes",#PB_Text_Center)
 ListIconGadget(5, 360,
225, 330, 65, "",
200,#PB_ListIcon_GridLines)
 For a = 0 To 4
; pour chacune des 5
premières listes avec
icônes
   For b = 2 To 4
; ajouter 3 colonnes
supplémentaires
     AddGadgetColumn(a, b,
"Colonne " + Str(b), 65)
   Next
   For b = 0 To 2
; ajouter 4 éléments à
chaque ligne des listes
avec icônes
     AddGadgetItem(a, b,
"Elément
1"+Chr(10)+"Elément
2"+Chr(10)+"Elément
3"+Chr(10)+"Elément 4")
   Next
 Next
 ; Ici nous changeons
l'affichage de la liste
avec icônes en mode
```

```
    grandes icônes et nous
    affichons une image
26   If LoadImage(0,
    #PB_Compiler_Home+"Examples\Sources\Data\File.bmp")
        ; changez le
    chemin/fichier contenant
    votre image 32x32 pixel
27     SetGadgetAttribute(5,
    #PB_ListIcon_DisplayMode,
    #PB_ListIcon_LargeIcon)
28     AddGadgetItem(5, 1,
    "Image 1", ImageID(0))
29     AddGadgetItem(5, 2,
    "Image 2", ImageID(0))
30   EndIf
31   Repeat : Until
    WaitWindowEvent() =
    #PB_Event_CloseWindow
32 EndIf
```



Voir aussi

AddGadgetColumn() ,
RemoveGadgetColumn() ,
AddGadgetItem() , RemoveGadgetItem() ,
ClearGadgetItems() , CountGadgetItems() ,
GetGadgetState() , SetGadgetState() ,
GetGadgetAttribute() ,
SetGadgetAttribute() ,
GetGadgetItemText() ,
SetGadgetItemText() ,
GetGadgetItemState() ,
SetGadgetItemState() ,
SetGadgetItemImage() ,
GetGadgetItemData() ,
SetGadgetItemData() ,
GetGadgetItemAttribute() ,
SetGadgetItemAttribute() ,
GetGadgetColor() , SetGadgetColor() ,
GetGadgetItemColor() ,
SetGadgetItemColor() ,
ExplorerListGadget() , ListViewGadget()

OS Supportés

Tous

99.50 ListViewGadget

Syntaxe

```
Resultat =
    ListViewGadget(#Gadget, X,
    Y, Largeur, Hauteur [,
    Options])
```

Description

Crée une boîte à listes (ListView) dans la GadgetList en cours.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Options (optionnel) Peut être une combinaison de :

```
#PB_ListView_Multiselect:
 Permet la sélection de
 plusieurs éléments
 consécutifs.
#PB_ListView_ClickSelect:
 Permet la sélection de
 plusieurs éléments.
                          Cliquez
 sur un élément pour le
 sélectionner ou le
 désélectionner
                          (sous
 OS X, même comportement
 que
 #PB_ListView_Multiselect).
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

```
  - GadgetToolTip()
 permet d'ajouter une 'mini
   aide' à ce gadget.

  Les fonctions suivantes
   peuvent être utilisées
   pour agir sur le contenu
   de cette liste:
  - AddGadgetItem()
     : Ajoute un élément.
   Limité à 65 536 éléments.
  - CountGadgetItems()
  : Renvoie le nombre de
   lignes contenues dans le
   gadget.
  - RemoveGadgetItem()
  : Supprime un élément
  - ClearGadgetItems()
  : Supprime tous les éléments
  - GetGadgetItemData()
 : Renvoie la valeur
   personnalisée associée à
   cet élément.
  - SetGadgetItemData()
 : Associe une valeur
   personnalisée à cet
   élément.
  - GetGadgetItemText()
 : Renvoie le texte de
   l'élément spécifié.
  - SetGadgetItemText()
 : Change le texte de
   l'élément spécifié.
  - GetGadgetItemState()
: Renvoie 0 si l'élément
   n'est pas séléctionné,
   sinon une valeur non-nulle.
  - SetGadgetItemState()
: Sélectionne ou
   désélectionne l'élément
   spécifié.
  - GetGadgetState()
     : Renvoie le numéro de
   l'élément qui est
   sélectionné, -1 si il n'y
   a pas de sélection.
  - SetGadgetState()
     : Change l'état
   sélectionné/désélectionné
   de l'élément spécifié et
   avec -1 tous les éléments
   sont désélectionnés.
  - GetGadgetText()
     : Renvoie le texte de
   l'élément sélectionné.
  - SetGadgetText()
     : Sélectionne l'élément
   correspondant au texte
   indiqué. Le texte doit
```

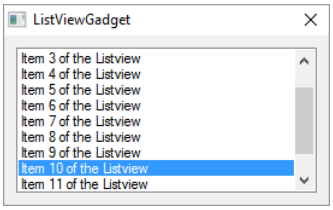
```
  exactement correspondre.

 - SetGadgetColor()
et GetGadgetColor()
avec les valeurs
  'TypeCouleur' suivantes:
   #PB_Gadget_BackColor :
  Couleur de fond
   #PB_Gadget_FrontColor:
  Couleur du texte

 Note: SetGadgetColor()
n'est pas pris en charge sur
  la plate-forme MacOS X.
```

Les évènements suivants sont supportés par EventType() :

```
#PB_EventType_LeftClick
  (Également déclenché lors
  d'un changement de
  sélection)
#PB_EventType_LeftDoubleClick
#PB_EventType_RightClick
```

Exemple

```
1  If
     OpenWindow(0,0,0,270,140,"ListViewGadget",#PB_Window_SystemM
     |
     #PB_Window_ScreenCentered)
2     ListViewGadget(0,10,10,250,120)
3     For a=1 To 12
4       AddGadgetItem
     (0,-1,"Elément "+Str(a)+"
     de la boîte à liste")   ;
     défini le contenu de la
     boîte de liste
5     Next
6     SetGadgetState(0,9)     ;
     sélectionne le dixième
     élément (la numérotation
     commmence à 0)
7     Repeat : Until
     WaitWindowEvent()=#PB_Event_CloseWindow
8  EndIf
```



Voir aussi

AddGadgetItem() , RemoveGadgetItem() , ClearGadgetItems() , CountGadgetItems() , GetGadgetState() , SetGadgetState() ,

GetGadgetText() , SetGadgetText() , GetGadgetItemState() , SetGadgetItemState() , GetGadgetItemText() , SetGadgetItemText() , GetGadgetItemData() , SetGadgetItemData() , GetGadgetColor() , SetGadgetColor() , ListIconGadget()

OS Supportés

Tous

99.51 MDIGadget

Syntaxe

```
Resultat = MDIGadget(#Gadget,
    X, Y, Largeur, Hauteur,
    SousMenu, ElementMenu [,
    Options])
```

Description

Crée un espace dans lequel peut s'afficher une ou plusieurs fenêtres filles. Ces fenêtres peuvent être librement déplacées et redimensionnées à l'intérieur de cet espace. Ce type de gestion de fenêtres s'appelle MDI (Multiple Document Interface).

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

SousMenu L'indice du menu pour lequel l'élément de fenêtres MDI doit être ajouté.

Un MDIGadget() a toujours besoin d'une fenêtre avec menu (voir CreateMenu()). Il n'est pas possible de créer plusieurs contrôles MDI dans une seule fenêtre, car il n'y a qu'un seul menu disponible par fenêtre. Une fois créé, le gadget mettra automatiquement à jour le menu auquel il est rattaché, avec la liste des fenêtres filles ouvertes. Dans le paramètre 'SousMenu', il faut préciser l'index du sous-menu (créé avec MenuTitle()) qui sera le point d'attache aux éléments gérés par le gadget (l'index du premier sous-menu commence à 0). Le gadget

ajoutera alors une barre de séparation à la fin de ce sous-menu et ajoutera la liste des fenêtres filles disponibles.

ElementMenu Le premier indice de menu à utiliser pour les fenêtres MDI.

Le gadget a besoin d'un certain nombre d'identifiants pour ses propres éléments du menu (voir le paramètre 'MenuID' de la commande MenuItem()). Dans le paramètre 'ElementMenu', il faudra indiquer le chiffre à partir duquel le gadget pourra identifier ses propres éléments. Il utilisera autant d'identifiants qu'il y aura de fenêtres filles ouvertes. Il est donc recommandé d'utiliser un nombre plus grand que le plus grand des nombres utilisé par les menus normaux du programme, pour éviter tout risque de collisions.

Options (optionnel) Peut être une combinaison de :

```
#PB_MDI_AutoSize     : Le gadget sera automatiquement redimensionné en fonction de la taille de la fenêtre mère.
                                                    S'il n'y a pas d'autres gadgets sur la fenêtre mère, cela peut être une option très pratique.
#PB_MDI_BorderLess   : Il n'y aura pas de bords autour du gadget.
#PB_MDI_NoScrollBars: Quand une fenêtre est déplacée en dehors de l'espace du gadget, il n'y aura pas de barres de défilement.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

A cause de la connexion avec les menus, il ne peut y avoir qu'un seul gadget MDI par

fenêtre, cependant vous pouvez utiliser plusieurs fenêtres contenant un gadget MDI chacune. Ce gadget ne peut être créé que dans une fenêtre principale donc pas dans un ContainerGadget() , SplitterGadget() ou PanelGadget()).
Comme l'intérêt premier de ce gadget est de gérer de manière dynamique des fenêtres, il est recommandé d'utiliser #PB_Any (création dynamique) pour les gadgets qui seront créés dans les fenêtres filles.
Quand la commande AddGadgetItem() est utilisée avec ce gadget, l'élément créé est en fait une nouvelle fenêtre .
Ainsi, toutes les commandes de la bibliothèque Window sont disponibles pour gérer cette nouvelle fenêtre (sauf StickyWindow()). Le numéro choisi pour cet élément ne doit pas entrer en conflit avec un numéro de fenêtre déjà existant sinon l'autre fenêtre sera fermée. Le gadget MDI ne génère pas d'évènements. Les évènements concernant les fenêtres MDI filles seront reçus comme des évènements fenêtre normaux (#PB_Event_SizeWindow, #PB_Event_CloseWindow, ...).

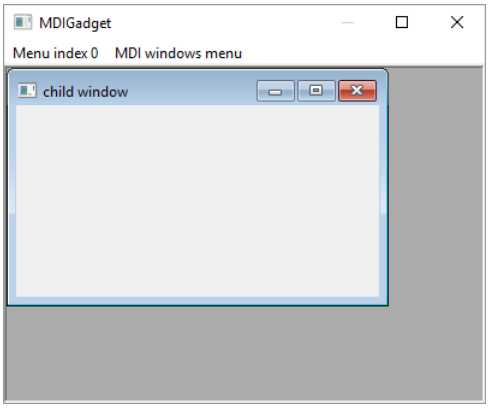
```
  - GadgetToolTip()
 permet d'ajouter une 'mini
   aide' à ce gadget.

  En complément des commandes
   de la bibliothèque Window
 (sauf StickyWindow()
),
  les commandes suivantes
   sont disponibles pour agir
   sur le gadget:
  - CountGadgetItems()
  : Renvoie le nombre de
   fenêtres fille.
  - AddGadgetItem()
     : Crée une nouvelle
   fenêtre fille.
  - ClearGadgetItems()
  : Ferme toutes les fenêtres
   filles.
  - GetGadgetState()
    : Renvoie l'identifiant
   de la fenêtre fille active.
  - SetGadgetState()
    : Change la fenêtre fille
   active ou ré-arrange la
   disposition des fenêtres
   filles (voir
   GetGadgetState()
).
  - SetGadgetAttribute()
: Avec une des constantes
   suivantes:
```

```
    #PB_MDI_Image     :
    Applique une image de fond
    à la fenêtre MDI.
    #PB_MDI_TileImage:
    Applique une image de fond
    à la fenêtre MDI.
                          L'image
    est répétée le nombre de
    fois nécessaire
                          pour
    remplir complètement la
    surface de la fenêtre.

  - SetGadgetAttribute()
: Avec une des constantes
    suivantes:
    #PB_ScrollArea_InnerWidth
    : Modifie la largeur (en
    pixels) de la zone interne
    du gadget.
    #PB_ScrollArea_InnerHeight:
    Modifie la hauteur (en
    pixels) de la zone interne
    du gadget.

  - SetGadgetColor()
 et GetGadgetColor()
 avec la constante
    #PB_Gadget_BackColor comme
    'TypeCouleur'
    pour changer la couleur
    de fond de la fenêtre MDI.
```

Exemple

```
1  #FenetrePrincipale = 0
2  #FenetreFille = 1
3  If
    OpenWindow(#FenetrePrincipale,
    0, 0, 400, 300,
    "MDIGadget",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered
    | #PB_Window_SizeGadget |
    #PB_Window_MaximizeGadget)
4    If
    CreateMenu(#FenetrePrincipale,
    WindowID(#FenetrePrincipale))
5       MenuTitle("Index de
    menu 0")
6       MenuTitle("Menu des
    fenêtres filles")
7         MenuItem(0, "Elément
    auto-créé")
8         MenuItem(1, "Elément
    auto-créé")
9
```

```
10      MDIGadget(0, 0, 0, 0,
    0, 1, 2, #PB_MDI_AutoSize)
11        AddGadgetItem(0,
    #FenetreFille, "Fenêtre
    fille")
12          ; ajouter des
    gadgets ici...
13      UseGadgetList(WindowID(#FenetrePrincipale))
    ; renvoie à la liste de
    gadgets de la fenêtre
    principale
14    EndIf
15    Repeat : Until
    WaitWindowEvent()=#PB_Event_CloseWindow
16  EndIf
```



OS Supportés

Windows

99.52 OpenGadgetList

Syntaxe

```
OpenGadgetList(#Gadget [,
    Element])
```

Description

Utilise un gadget comme GadgetList permettant d'ajouter des nouveaux gadgets à la volée.

Arguments

#Gadget Le gadget dans lequel de nouveaux gadgets seront créés.

Element (optionnel) Pour le PanelGadget() : Indique l'onglet sur lequel les gadgets doivent être ajoutés. Pour ajouter un nouvel onglet dynamiquement, le paramètre 'Element' doit être omis.

Valeur de retour

Aucune.

Remarques

Les gadgets suivants sont pris en charge par OpenGadgetList() :

- `ContainerGadget()`
- `PanelGadget()`
- `ScrollAreaGadget()`

Une fois tous les changements effectués, CloseGadgetList() doit être appelée pour fermer la GadgetList.

Voir aussi

CloseGadgetList() , ContainerGadget() , PanelGadget() , ScrollAreaGadget()

OS Supportés

Tous

99.53 OptionGadget

Syntaxe

```
Resultat =
    OptionGadget(#Gadget, X,
    Y, Largeur, Hauteur,
    Texte$)
```

Description

Crée un gadget case à options (connu aussi sous le nom de 'boutons radio') dans la GadgetList en cours.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Texte$ (optionnel) Le texte placé à droite.

Valeur de retour

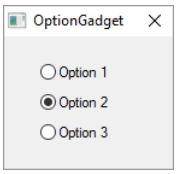
Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

Ce gadget permet de regrouper plusieurs cases à options, sachant qu'une seule d'entre elles peut être sélectionnée à la fois. Au premier appel de cette fonction, un groupe de cases à options est créé, et tous les appels suivants ajouteront une nouvelle case à options au groupe. Pour terminer le groupe, il suffit d'appeler un autre type de gadget.
- GadgetToolTip() permet d'ajouter une 'mini aide' à ce gadget.

Exemple

```
1  If OpenWindow(0, 0, 0, 170,
     110, "OptionGadget",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2    OptionGadget(0, 30, 20,
     60, 20, "Option 1")
3    OptionGadget(1, 30, 45,
     60, 20, "Option 2")
4    OptionGadget(2, 30, 70,
     60, 20, "Option 3")
5    SetGadgetState(1, 1)    ;
     sélectionne la deuxième
     option
6    Repeat : Until
     WaitWindowEvent() =
     #PB_Event_CloseWindow
7  EndIf
```



Voir aussi

GetGadgetText() , SetGadgetText() , GetGadgetState() , SetGadgetState() , CheckBoxGadget()

OS Supportés

Tous

99.54 PanelGadget

Syntaxe

```
Resultat =
    PanelGadget(#Gadget, X, Y,
    Largeur, Hauteur)
```

Description

Crée un gadget boîte à onglets (Panel) dans la GadgetList.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

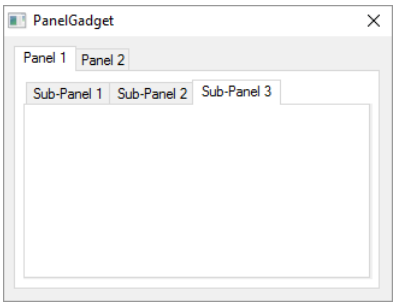
```
  - GadgetToolTip()
 permet d'ajouter une 'mini
   aide' à ce gadget.

  Les fonctions suivantes
   peuvent être appelées pour
   agir sur la liste :
  - AddGadgetItem()
     : Ajoute un élément.
  - RemoveGadgetItem()
  : Supprime un élément.
  - CountGadgetItems()
  : Renvoie le nombre
   d'éléments.
  - ClearGadgetItems()
  : Supprime tous les
   éléments.
  - GetGadgetItemText()
 : Renvoie le texte de
   l'élément spécifié.
  - SetGadgetItemText()
 : Change le texte de
   l'élément spécifié.
  - SetGadgetItemImage()
```

```
: Change l'image de l'élément
    spécifié (non pris en
    charge sur OS X).
  - GetGadgetItemData()
 : Renvoie la valeur associée
    à l'élément spécifié.
  - SetGadgetItemData()
 : Associe une valeur à
    l'élément spécifié.
  - SetGadgetState()
     : Change l'onglet affiché.
  - GetGadgetState()
     : Renvoie le numéro de
    l'onglet actuellement
    affiché.

  - GetGadgetAttribute()
 avec un des attributs
    suivants (il doit y avoir
    au moins un élément dans
    le PanelGadget()):
     #PB_Panel_ItemWidth :
    Renvoie la largeur de la
    zone utilisable d'un
    onglet.
     #PB_Panel_ItemHeight:
    Renvoie la hauteur de la
    zone utilisable d'un
    onglet.
     #PB_Panel_TabHeight :
    Renvoie la hauteur d'un
    bouton de changement
    d'onglet.

  Les évènements suivants
    sont pris en charge par
    EventType()
:
  #PB_EventType_Change:
    L'onglet courant a changé.
  #PB_EventType_Resize: Le
    gadget a été redimensionné.
```

Lorsqu'une boîte à onglets est créée, sa liste d'éléments est vide. Avant de pouvoir ajouter des gadgets ou des onglets, il faut qu'il existe au moins un onglet au préalable, il est nécessaire d'appeler la commande AddGadgetItem() pour cela. Les prochains gadgets créés le seront automatiquement sur le dernier onglet. Lorsque tous les gadgets de la boîte à onglets ont été placés, CloseGadgetList() doit être appelé pour revenir à la GadgetList précédente. Il est ainsi parfaitement possible de créer une boîte à onglets dans une autre boîte à onglets...

Exemple

```
; Exemple d'utilisation de plusieurs onglets...
If OpenWindow(0, 0, 0, 322, 220, "PanelGadget", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  PanelGadget      (0, 8, 8, 306, 203)
    AddGadgetItem (0, -1, "Onglet 1")
      PanelGadget (1, 5, 5, 290, 166)
        AddGadgetItem(1, -1, "Sous-onglet 1")
        AddGadgetItem(1, -1, "Sous-onglet 2")
        AddGadgetItem(1, -1, "Sous-onglet 3")
      CloseGadgetList()
    AddGadgetItem (0, -1,"Onglet 2")
      ButtonGadget(2, 10, 15, 80, 24,"Bouton 1")
      ButtonGadget(3, 95, 15, 80, 24,"Bouton 2")
  CloseGadgetList()
  Repeat : Until WaitWindowEvent() = #PB_Event_CloseWindow
EndIf
```



Voir aussi

AddGadgetItem() , RemoveGadgetItem() , CountGadgetItems() , ClearGadgetItems() , GetGadgetItemText() , SetGadgetItemText() , GetGadgetState() , SetGadgetState() , GetGadgetAttribute() , CloseGadgetList() , OpenGadgetList() , SetGadgetItemImage()

OS Supportés

Tous

99.55 ProgressBarGadget

Syntaxe

```
Resultat =
    ProgressBarGadget(#Gadget,
    X, Y, Largeur, Hauteur,
    Minimum, Maximum [,
    Options])
```

Description

Crée un gadget Barre de progression dans la GadgetList.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Minimum, Maximum L'intervalle qui sera utilisé par la barre de progression. La valeur doit être comprise entre 0 et 65536 pour être compatible avec tous les systèmes d'exploitation.

Options (optionnel) Peut être une combinaison de :

```
#PB_ProgressBar_Smooth  :
 La progression est
 précise, au lieu
 d'utiliser des blocs
 (Note: sous Windows XP
 avec
                          le
 support des skins activé
 et sous OS X, cette
 option n'a aucun effet).
#PB_ProgressBar_Vertical:
 La barre de progression
 sera verticale.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

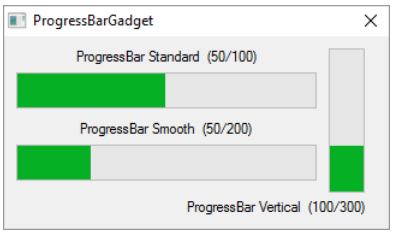
Remarques

```
  - GadgetToolTip()
 permet d'ajouter une 'mini
   aide' à ce gadget.

  Les fonctions suivantes
   peuvent être appelées pour
   agir sur la barre de
   progression :
  - SetGadgetState()
: Change la valeur de la
   barre de progression.
    #PB_ProgressBar_Unknown
   peut être utilisée pour
   indiquer que la
   progression est inconnue.

  - GetGadgetState()
: Renvoie la valeur courante
   de la barre de progression.

  - SetGadgetAttribute()
 avec les attributs suivants:
    #PB_ProgressBar_Minimum:
   Change la valeur minimale.
    #PB_ProgressBar_Maximum:
   Change la valeur maximale.

  - GetGadgetAttribute()
 avec les attributs suivants:
    #PB_ProgressBar_Minimum:
   Renvoie la valeur minimale.
    #PB_ProgressBar_Maximum:
   Renvoie la valeur maximale.

  -SetGadgetColor()
 et GetGadgetColor()
 avec les valeurs suivantes
   comme 'TypeCouleur'
  (Note: Sous Windows XP avec
   le support des skins
   activé, la couleur n'a
   aucun effet):
    #PB_Gadget_FrontColor:
   Couleur de la barre de
   progression
    #PB_Gadget_BackColor :
   Couleur du fond

  Note: SetGadgetColor()
 n'est pas pris en charge sur
   la plate-forme MacOS X.
```

Exemple

```
1  If OpenWindow(0, 0, 0, 320,
     160, "ProgressBarGadget",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
```

```
    TextGadget        (3,  10, 10, 250,  20, "Barre de progression standard (50/100)", #PB_Text_Center)
    ProgressBarGadget(0,  10, 30, 250,  30, 0, 100)
    SetGadgetState    (0, 50)  ; change la valeur de la 1ère barre de progression (ID = 0) à 50 sur 100
    TextGadget        (4,  10, 70, 250,  20, "Barre de progression précise (50/200)", #PB_Text_Center)
    ProgressBarGadget(1,  10, 90, 250,  30, 0, 200, #PB_ProgressBar_Smooth)
    SetGadgetState    (1, 50)  ; change la valeur de la 2ème barre de progression (ID = 1) à 50 sur 200
    TextGadget        (5, 100,135, 200,  20, "Barre de progression verticale (100/300)", #PB_Text_Right)
    ProgressBarGadget(2, 270, 10,  30, 120, 0, 300, #PB_ProgressBar_Vertical)
    SetGadgetState    (2, 100)  ; change la valeur de la 3ème barre de progression (ID = 2) à 100 sur 300
    Repeat : Until WaitWindowEvent()=#PB_Event_CloseWindow
  EndIf
```



Voir aussi

GetGadgetState() , SetGadgetState() , GetGadgetAttribute() , SetGadgetAttribute() , GetGadgetColor() , SetGadgetColor() , SetGadgetItemImage()

OS Supportés

Tous

99.56 RemoveGadgetColumn

Syntaxe

```
RemoveGadgetColumn(#Gadget,
    Colonne)
```

Description

Supprime une colonne d'un gadget.

Arguments

#Gadget Le gadget à utiliser

Colonne La colonne à supprimer.
#PB_All peut être utilisé pour supprimer toutes les colonnes.
La première colonne commence à 0

Valeur de retour

Aucune.

Remarques

Peut être l'un des gadgets suivants :
- ListIconGadget()
- ExplorerListGadget()

Exemple

```
1   If OpenWindow(0, 0, 0, 320,
     160, "RemoveGadgetColumn",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2
3     ListIconGadget(0, 10, 10,
    300, 140, "Salut", 100)
4       AddGadgetColumn(0, 1,
    "Colonne 2", 70)
5       AddGadgetColumn(0, 2,
    "Colonne 3", 70)
6
7     RemoveGadgetColumn(0, 1)
    ; Supprime la 'colonne 2'
8
9     Repeat
10    Until WaitWindowEvent() =
    #PB_Event_CloseWindow
11  EndIf
```

Voir aussi

AddGadgetColumn() , ListIconGadget() , ExplorerListGadget()

OS Supportés

Tous

99.57 RemoveGadgetItem

Syntaxe

```
RemoveGadgetItem(#Gadget,
    Position)
```

Description

Supprime un élément d'un gadget.

Arguments

#Gadget Le gadget à utiliser

Position La position de l'élément dans la liste.
L'index commence à 0.

Valeur de retour

Aucune.

Remarques

Cette fonction s'applique aux gadgets suivants :
- ComboBoxGadget()
- EditorGadget()
- PanelGadget()
- ListViewGadget()
- ListIconGadget()
- MDIGadget()
- TreeGadget() - La suppression d'un noeud entraîne la suppression de tous les éléments enfants du noeud.

Voir aussi

AddGadgetItem() , ClearGadgetItems() , CountGadgetItems()

OS Supportés

Tous

99.58 ResizeGadget

Syntaxe

```
ResizeGadget(#Gadget, X, Y,
    Largeur, Hauteur)
```

Description

Change la taille et la position d'un gadget.

Arguments

#Gadget Le gadget à utiliser

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Pour faciliter le redimensionnement de l'interface en temps réel, la constante #PB_Ignore peut être spécifiée à chacun des paramètres (X, Y, Largeur ou Hauteur) conservant la valeur précédente de ce paramètre.

Valeur de retour

Aucune.

Exemple

```
[...]

ResizeGadget(0, #PB_Ignore, #PB_Ignore, 300, #PB_Ignore) ; Change seulement la largeur du gadget.

ResizeGadget(0, 150, 100, #PB_Ignore, #PB_Ignore) ; Déplacement du gadget sans redimensionnement.
```

Voir aussi

GadgetX() , GadgetY() , GadgetWidth() , GadgetHeight()

OS Supportés

Tous

99.59 ScrollBarGadget

Syntaxe

```
Resultat =
    ScrollBarGadget(#Gadget,
    X, Y, Largeur, Hauteur,
    Minimum, Maximum,
    LongueurPage [, Options])
```

Description

Crée une nouvelle barre de défilement (ascenseur vertical ou horizontal) dans la GadgetList courante.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Minimum, Maximum L'intervalle de défilement (valeurs comprises entre 0 et 10 000).

LongueurPage Valeur de déplacement de la page.
Un ascenseur permet de se déplacer par page lorque l'on clique dans la barre de défilement mais en dehors du curseur de déplacement

Exemple : La longueur totale du composant (image, document, container etc...) fait 500 pixels. On ne peut en afficher que 100 pixels. Donc on aura une valeur minimale de 0, une valeur Maximale de 500 et une longueur de page de 100. Le curseur sera 5 fois plus petit que la barre de déplacement, car sa dimension est proportionnelle (dimension = Maximum/LongueurPage).

```
#PB_ScrollBar_Vertical: La
  barre de déplacement
  verticale.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

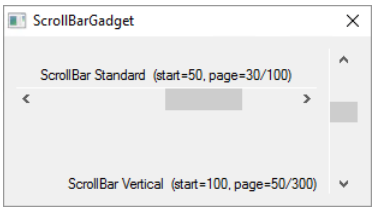
```
  - GadgetToolTip()
 permet d'ajouter une 'mini
   aide' à ce gadget.

  Les fonctions suivantes
   peuvent être appelées pour
   agir sur la barre de
   déplacement :
  - GetGadgetState()
: Renvoie la position
   actuelle du curseur
   (valeur comprise dans
   l'intervalle Minimum -
   longueur de la page + 1)
```

```
  - SetGadgetState()
: Change la position actuelle
   du curseur.
  - GetGadgetAttribute()
 avec un des attributs
   suivants:
    #PB_ScrollBar_Minimum   :
   Renvoie la position
   minimale de l'ascenseur.
    #PB_ScrollBar_Maximum   :
   Renvoie la position
   maximale de l'ascenseur.
    #PB_ScrollBar_PageLength:
   Renvoie la longueur de la
   page.

  - SetGadgetAttribute()
: avec un des attributs
   suivants:
    #PB_ScrollBar_Minimum   :
   Change la position
   minimale de l'ascenseur.
    #PB_ScrollBar_Maximum   :
   Change la position
   maximale de l'ascenseur.
    #PB_ScrollBar_PageLength:
   Change la longueur de la
   page.
```

Exemple

```
1  If OpenWindow(0, 0, 0, 305,
     140, "ScrollBarGadget",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2    TextGadget       (2,  10,
     25, 250,  20, "Ascenseur
     standard  (début = 50,
     page =
     30/100)",#PB_Text_Center)
3    ScrollBarGadget  (0,  10,
     42, 250,  20, 0, 100, 30)
4    SetGadgetState   (0,  50)
       ; change la valeur de la
     1ère barre de défilement
     (ID = 0) à 50 sur 100
5    TextGadget       (3,
     10,115, 250,  20,
     "Ascenseur vertical
     (début = 100, page =
     50/300)",#PB_Text_Right)
6    ScrollBarGadget  (1, 270,
     10,  25, 120 ,0, 300, 50,
     #PB_ScrollBar_Vertical)
7    SetGadgetState   (1, 100)
       ; change la valeur de la
     2ème barre de défilement
     (ID = 1) à 100 sur 300
8
```

```
    ; Enlevez les commentaires si dessous si vous avez un problème d'affichage
    ;     ResizeGadget(2, #PB_Ignore, #PB_Ignore, GadgetWidth(2, #PB_Gadget_RequiredSize), #PB_Ignore)
    ;     ResizeGadget(0, #PB_Ignore, GadgetY(2) + GadgetHeight(2), #PB_Ignore, #PB_Ignore)
    ;     ResizeGadget(1, GadgetX(2) + GadgetWidth(2), #PB_Ignore, #PB_Ignore, #PB_Ignore)
    ;     ResizeGadget(3, #PB_Ignore, #PB_Ignore, GadgetWidth(2, #PB_Gadget_RequiredSize), #PB_Ignore)
    ;     ResizeWindow(0, #PB_Ignore, #PB_Ignore, GadgetX(2) + GadgetWidth(2)+GadgetWidth(1) + 13, #PB_Ignore)

    Repeat : Until WaitWindowEvent() = #PB_Event_CloseWindow
  EndIf
```



Exemple : Evènement scrollBar

```
Procedure BindHScrollDatas()
  SetWindowTitle(0, "ScrollBarGadget (" + GetGadgetState(0) + ")" )
EndProcedure

Procedure BindVScrollDatas()
  SetWindowTitle(0, "ScrollBarGadget (" + GetGadgetState(1) + ")" )
EndProcedure
```

```
10  If OpenWindow(0, 0, 0, 400, 400, "ScrollBarGadget", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
11    TextGadget        (2,  10, 25, 350,  30, "Ascenseur standard  (début = 50, page = 30/100)")
12    ScrollBarGadget  (0,  10, 50, 350,  20, 0, 100, 30)
13    SetGadgetState    (0,  50)  ; change la valeur de la 1ère barre de défilement (ID = 0) à 50 sur 100
14    TextGadget        (3,  10, 120, 350,  30, "Ascenseur vertical  (début = 100, page = 50/300)")
15    ScrollBarGadget  (1, 175, 160,  25, 120 ,0, 300, 50, #PB_ScrollBar_Vertical)
16    SetGadgetState    (1, 100)  ; change la valeur de la 2ème barre de défilement (ID = 1) à 100 sur 300
17
18    BindGadgetEvent(0, @BindHScrollDatas())
19    BindGadgetEvent(1, @BindVScrollDatas())
20
21    Repeat
22      Select WaitWindowEvent()
23        Case #PB_Event_CloseWindow
24          End
25        Case  #PB_Event_Gadget
26          Select EventGadget()
27            Case 0
28              MessageRequester("Info","L'ascenseur 0 a été utilisé ! (" + GetGadgetState(0) +
29                               ")",#PB_MessageRequester_Ok)
30            Case 1
31              MessageRequester("Info","L'ascenseur 1 a été utilisé ! (" + GetGadgetState(1) +
32                               ")",#PB_MessageRequester_Ok)
33
34          EndSelect
35      EndSelect
36    ForEver
37  EndIf
```

Voir aussi

GetGadgetState() , SetGadgetState() , GetGadgetAttribute() , SetGadgetAttribute() , ScrollAreaGadget()

OS Supportés

Tous

99.60 ScrollAreaGadget

Syntaxe

```
Resultat =
    ScrollAreaGadget(#Gadget,
    X, Y, Largeur, Hauteur,
    LargeurZoneInterne,
    HauteurZoneInterne [,
    ValeurDeplacement [,
    Options]])
```

Description

Crée un gadget zone de défilement dans la GadgetList en cours. C'est un container muni de barres de défilement.

Arguments

Options (optionnel)¥#Gadget Le numéro d'identification du nouveau gadget. #PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

LargeurZoneInterne, HauteurZoneInterne Les dimensions de la zone interne. Elles peuvent également être plus petits que les dimensions extérieures et dans ce cas, les barres de défilements seront masquées. (MS Windows limite ces paramètres à 32 000 pixels)

ValeurDeplacement (optionnel) Déplacement de la zone interne quand on presse sur l'une des flèches de défilement.

Options (optionnel) Peut être une combinaison de :

```
#PB_ScrollArea_Flat      :
 Cadre simple
#PB_ScrollArea_Raised    :
 Cadre élevé
#PB_ScrollArea_Single    :
 Cadre enfoncé
#PB_ScrollArea_BorderLess:
 Sans bordure
```

```
#PB_ScrollArea_Center    :
 La zone interne est
 automatiquement centrée
 si sa taille est
                              plus
 petite que les
 dimensions du gadget.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

Une fois créé, tous les gadgets suivants seront placés dans ce gadget.
CloseGadgetList() permet de revenir à la GadgetList précédente.
OpenGadgetList() pourra être utilisé pour ajouter des gadgets dynamiquement.
Un évènement est généré lorsque l'utilisateur déplace les ascenseurs du gadget.
Les commandes suivantes peuvent être utilisées pour agir sur un ScrollAreaGadget :

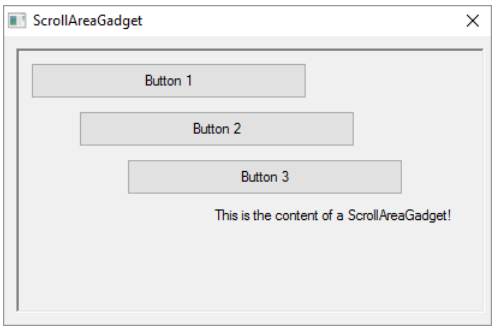
```
  - GetGadgetAttribute()
: Avec une des constantes
   suivantes:
    #PB_ScrollArea_InnerWidth
    : Renvoie la largeur (en
   pixels) de la zone interne
   du gadget.
    #PB_ScrollArea_InnerHeight
   : Renvoie la hauteur (en
   pixels) de la zone interne
   du gadget.
    #PB_ScrollArea_X
              : Renvoie la
   position horizontale
   actuelle de l'ascenseur
   (en pixels).
    #PB_ScrollArea_Y
              : Renvoie la
   position verticale
   actuelle de l'ascenseur
   (en pixels).
    #PB_ScrollArea_ScrollStep
    : Renvoie la valeur du
   pas de défilement (en
   pixels).

  - SetGadgetAttribute()
```

```
: Avec une des constantes
   suivantes:
    #PB_ScrollArea_InnerWidth
    : Modifie la largeur (en
   pixels) de la zone interne
   du gadget.
    #PB_ScrollArea_InnerHeight
   : Modifie la hauteur (en
   pixels) de la zone interne
   du gadget.
    #PB_ScrollArea_X
             : Modifie la
   position horizontale
   actuelle de l'ascenseur
   (en pixels).
    #PB_ScrollArea_Y
             : Modifie la
   position verticale
   actuelle de l'ascenseur
   (en pixels).
    #PB_ScrollArea_ScrollStep
    : Modifie la valeur du
   pas de défilement (en
   pixels).

  - SetGadgetColor()
 et GetGadgetColor()
 avec la constante
   #PB_Gadget_BackColor comme
   'TypeCouleur'
    pour changer la couleur
   de fond du gadget.

 L'évènement suivant est pris
   en charge par EventType()
:
  #PB_EventType_Resize: Le
   gadget a été redimensionné.
```

Exemple

```
1
2   Procedure BindScrollDatas()
3     SetWindowTitle(0,
      "ScrollAreaGadget " +
4                    "(" +
5                    GetGadgetAttribute(0,
      #PB_ScrollArea_X) +
6                    "," +
7                    GetGadgetAttribute(0,
      #PB_ScrollArea_Y) +
8                    ")" )
9   EndProcedure
10
11  If OpenWindow(0, 0, 0, 405,
     240, "ScrollAreaGadget",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
```

```
12      ScrollAreaGadget(0, 10,
        10, 390,220, 575, 575, 30)
13        ButtonGadget  (1, 10,
        10, 230, 30,"Bouton 1")
14        ButtonGadget  (2, 50,
        50, 230, 30,"Bouton 2")
15        ButtonGadget  (3, 90,
        90, 230, 30,"Bouton 3")
16        TextGadget
        (4,130,130, 230, 60,"Ceci
        est le contenu d'une zone
        de défilement
        !",#PB_Text_Right)
17        CloseGadgetList()
18
19        BindGadgetEvent(0, @
        BindScrollDatas())
20
21      Repeat
22        Select WaitWindowEvent()
23          Case
        #PB_Event_CloseWindow
24            End
25          Case  #PB_Event_Gadget
26            Select EventGadget()
27              Case 0
28                MessageRequester("Info","Un
        ascenseur a été utilisé !
        (" +
29                                      GetGadgetAttribute(0,
        #PB_ScrollArea_X) +
30                                      ","
        +
31                                      GetGadgetAttribute(0,
        #PB_ScrollArea_Y) +
32                                      ")"
        ,#PB_MessageRequester_Ok)
33              Case 1
34                MessageRequester("Info","Le
        bouton 1 a été appuyé
        !",#PB_MessageRequester_Ok)
35              Case 2
36                MessageRequester("Info","Le
        bouton 2 a été appuyé
        !",#PB_MessageRequester_Ok)
37              Case 3
38                MessageRequester("Info","Le
        bouton 3 a été appuyé
        !",#PB_MessageRequester_Ok)
39            EndSelect
40        EndSelect
41      ForEver
42    EndIf
```



Voir aussi

GetGadgetAttribute() , SetGadgetAttribute() , ScrollBarGadget()

OS Supportés

Tous

99.61 SetActiveGadget

Syntaxe

```
SetActiveGadget(#Gadget)
```

Description

Active un gadget et lui donne le focus clavier.

Arguments

#Gadget Le gadget à activer.
La valeur '-1' supprime le focus clavier de la fenêtre active.

Valeur de retour

Aucune.

Remarques

L'activation d'un gadget lui permet de devenir l'objet courant et de recevoir les messages et la gestion des touches.

Exemple

```
If OpenWindow(0, 0, 0, 270, 140, "SetActiveGadget", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  StringGadget   (0, 10, 10, 250, 25, "bla bla...")
  ComboBoxGadget(1, 10, 40, 250, 25)
```

```
4     For a = 1 To 5 :
      AddGadgetItem(1, -1,
      "Elément de liste
      déroulante " + Str(a)) :
      Next
5     SetGadgetState(1, 2)
                    ;
      sélectionne le troisième
      élément (la numérotation
      commence à 0)
6     ButtonGadget  (2, 10,
      90, 250, 20, "Activer le
      gadget de saisie de texte")
7     ButtonGadget  (3, 10,
      115, 250, 20, "Activer la
      liste déroulante")
8     Repeat
9       Evenement =
      WaitWindowEvent()
10      If Evenement =
      #PB_Event_Gadget
11        Select EventGadget()
12          Case 2 :
      SetActiveGadget(0)   ;
      Activer le gadget 'saisie
      de texte'
13          Case 3 :
      SetActiveGadget(1)   ;
      Activer la liste déroulante
14        EndSelect
15      EndIf
16    Until Evenement =
      #PB_Event_CloseWindow
17  EndIf
```

Voir aussi

GetActiveGadget() , SetActiveWindow()

OS Supportés

Tous

99.62 SetGadgetAttribute

Syntaxe

```
SetGadgetAttribute(#Gadget,
    Attribut, Valeur)
```

Description

Change la valeur d'un attribut d'un gadget.

Arguments

#Gadget Le gadget à utiliser

Attribut L'attribut à définir.
Consultez la documentation de chaque gadget pour les attributs pris en charge et leur signification.

Valeur La valeur à donner à l'attribut.

Valeur de retour

Aucune.

Remarques

Cette fonction est disponible avec les gadgets suivants :
- ButtonImageGadget()
- CalendarGadget()
- CanvasGadget()
- DateGadget()
- EditorGadget()
- ExplorerListGadget()
- ListIconGadget()
- MDIGadget()
- OpenGLGadget()
- ProgressBarGadget()
- ScrollAreaGadget()
- ScrollBarGadget()
- SpinGadget()
- SplitterGadget()
- StringGadget()
- TrackBarGadget()
- WebGadget()

Voir aussi

GetGadgetAttribute() ,
GetGadgetItemAttribute() ,
SetGadgetItemAttribute()

OS Supportés

Tous

99.63 SetGadgetColor

Syntaxe

```
SetGadgetColor(#Gadget,
    TypeCouleur, Couleur)
```

Description

Change la couleur de l'attribut 'TypeCouleur' d'un gadget.

Arguments

#**Gadget** Le gadget à utiliser

TypeCouleur L'attribut de couleur à changer.
Le paramètre 'TypeCouleur' peut prendre l'une des valeurs suivantes (tous les gadgets ne supportent pas toutes ces valeurs) :

```
#PB_Gadget_FrontColor
 : Texte du gadget
#PB_Gadget_BackColor
 : Fond du gadget
#PB_Gadget_LineColor
 : Couleur de la grille
#PB_Gadget_TitleFrontColor:
 Couleur du texte dans le
 titre (pour
 CalendarGadget()
)
#PB_Gadget_TitleBackColor
 : Couleur du fond dans
 le titre (pour
 CalendarGadget()
)
#PB_Gadget_GrayTextColor
 : Couleur du texte
 inactif (pour
 CalendarGadget()
)
```

Couleur La nouvelle couleur de l'attribut.
RGB() peut être utilisé pour obtenir une couleur valide.
Pour supprimer la couleur personnalisée et revenir à la couleur par défaut, utiliser la valeur #PB_Default.

Valeur de retour

Aucune.

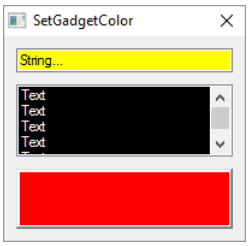
Remarques

Cette commande supporte les gadgets suivants :
- CalendarGadget()
- ContainerGadget()
- DateGadget()
- EditorGadget()
- ExplorerListGadget()
- ExplorerTreeGadget()
- HyperLinkGadget()
- ListViewGadget()
- ListIconGadget()
- MDIGadget()
- ProgressBarGadget() (Windows seulement)
- ScrollAreaGadget()
- SpinGadget()
- StringGadget()
- TextGadget()

- TreeGadget()
Note : Avec le support des thèmes activés sur Windows XP et suivants, les couleurs personnalisées ne seront probablement pas prises en compte sur certains gadgets.

Exemple

```
1   If OpenWindow(0, 0, 0, 210,
     170, "SetGadgetColor",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2    StringGadget(0, 10, 10,
    180, 20, "Chaîne...")
3    ListViewGadget(1, 10, 40,
    180, 60)
4    For i = 0 To 4
5      AddGadgetItem(1, -1,
    "Texte")
6    Next i
7    ContainerGadget(2, 10,
    110, 180, 50,
    #PB_Container_Raised)
8
9    SetGadgetColor(0,
    #PB_Gadget_BackColor,
    $00FFFF)
10   SetGadgetColor(1,
    #PB_Gadget_FrontColor,
    $FFFFFF)
11   SetGadgetColor(1,
    #PB_Gadget_BackColor,
    $000000)
12   SetGadgetColor(2,
    #PB_Gadget_BackColor,
    $0000FF)
13
14   Repeat
15   Until WaitWindowEvent() =
     #PB_Event_CloseWindow
16  EndIf
```



Voir aussi

GetGadgetColor() , GetGadgetItemColor() , SetGadgetItemColor()

OS Supportés

Tous

99.64 SetGadgetData

Syntaxe

```
SetGadgetData(#Gadget, Valeur)
```

Description

Associe une valeur à un gadget.

Arguments

#Gadget Le gadget à utiliser
Value La valeur à donner.

Valeur de retour

Aucune.

Remarques

Cette valeur peut être récupérée avec GetGadgetData() .
Tous les gadgets sont supportés par cette commande.

Exemple

```
; Ce programme utilise SetGadgetData pour associer un indice du tableau Messages()
; à chaque bouton. Cela rend la boucle d'évènements plus simple car les gadgets
; n'ont plus besoin d'être traités séparément un par un
;
Dim Messages.s(2)
Messages(0) = "Bonjour"
Messages(1) = "Salut le Monde"
Messages(2) = "Rien à dire"
If OpenWindow(0, 0, 0, 230, 100, "SetGadgetData", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  ButtonGadget(0,  10, 10, 80, 20, "Bouton 0"):
  SetGadgetData(0, 1)
```

```
11      ButtonGadget(1,  10, 40,
        80, 20, "Bouton 1"):
        SetGadgetData(1, 2)
12      ButtonGadget(2,  10, 70,
        80, 20, "Bouton 2"):
        SetGadgetData(2, 1)
13      ButtonGadget(3, 100, 10,
        80, 20, "Bouton 3"):
        SetGadgetData(3, 2)
14      ButtonGadget(4, 100, 40,
        80, 20, "Bouton 4") ; aura
        la valeur 0 car aucune
        valeur n'a été attribuée
15      ButtonGadget(5, 100, 70,
        80, 20, "Bouton 5")
16      Repeat
17        Evenement =
        WaitWindowEvent()
18        If Evenement =
        #PB_Event_Gadget
19          Valeur =
        GetGadgetData(EventGadget())
20          MessageRequester("Message",
        Messages(Valeur))
21        EndIf
22      Until Evenement =
        #PB_Event_CloseWindow
23    EndIf
```

Voir aussi

GetGadgetData() , GetGadgetItemData() , SetGadgetItemData()

OS Supportés

Tous

99.65 SetGadgetFont

Syntaxe

```
SetGadgetFont(#Gadget,
    PoliceID)
```

Description

Change la police d'un gadget.

Arguments

#Gadget Le gadget à utiliser.
Si la valeur de #Gadget est #PB_Default, alors la police par défaut utilisée par les nouveaux gadgets est changée par la nouvelle valeur.

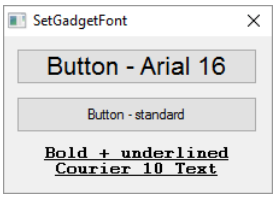
PoliceID Le numéro d'identification de la police à utiliser.
Si la valeur de 'PoliceID' est #PB_Default, alors la police par défaut du système d'exploitation sera utilisée.
'PoliceID' peut être facilement obtenu par la fonction FontID() de la bibliothèque Font.

Valeur de retour

Aucune.

Exemple

```
If OpenWindow(0, 0, 0, 222, 130, "SetGadgetFont", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
 If LoadFont(0, "Arial", 16)
    SetGadgetFont(#PB_Default, FontID(0))   ; la police par défaut est remplacée par celle chargée (Arial 16)
 EndIf
 ButtonGadget(0, 10, 10, 200, 30, "Bouton - Arial 16")
 SetGadgetFont(#PB_Default, #PB_Default)  ; remet la police par défaut originale (standard)
 ButtonGadget(1, 10, 50, 200, 30, "Bouton - standard")
 If LoadFont(1,"Courier",10, #PB_Font_Bold | #PB_Font_Underline)
    SetGadgetFont(#PB_Default, FontID(1))   ; la police par défaut est remplacée par celle chargée (Courier 10)
 EndIf
 TextGadget(2, 10, 90, 200, 40, "Texte en Courier 10 Gras + souligné", #PB_Text_Center)
 Repeat : Until WaitWindowEvent() = #PB_Event_CloseWindow
EndIf
```



Voir aussi

GetGadgetFont() , FontID() , LoadFont()

OS Supportés

Tous

99.66 SetGadgetItemAttribute

Syntaxe

```
SetGadgetItemAttribute(#Gadget,
    Element, Attribut, Valeur
    [, Colonne])
```

Description

Change la valeur d'un attribut d'un élément d'un gadget.

Arguments

#Gadget Le gadget à utiliser.

Element L'élément à utiliser.
Le premier élément commence à 0.

Attribut L'attribut à définir.
Voir ci-dessous pour les valeurs prises en charge.

Valeur La valeur à définir pour l'attribut.

Colonne (optionnel) La colonne à utiliser pour les gadgets qui prennent en charge plusieurs colonnes.
La première colonne a indice 0.
La valeur par défaut est 0 colonne.

Valeur de retour

Aucune.

Remarques

Cette fonction est disponible avec les gadgets suivants :

- `ExplorerListGadget()`

:

```
    #PB_Explorer_ColumnWidth:
    Change la largeur de la
    'Colonne' spécifiée. Le
    paramètre 'Element' est
    ignoré.

  - ListIconGadget()
:
    #PB_ListIcon_ColumnWidth:
    Change la largeur de la
    'Colonne' spécifiée. Le
    paramètre 'Element' est
    ignoré.
```

Voir aussi

GetGadgetItemAttribute() ,
GetGadgetAttribute() ,
SetGadgetAttribute()

OS Supportés

Tous

99.67 SetGadgetItemColor

Syntaxe

```
SetGadgetItemColor(#Gadget,
    Element, TypeCouleur,
    Couleur [, Colonne])
```

Description

Change la couleur d'un élément d'un gadget.

Arguments

#Gadget Le gadget à utiliser.

Element L'élément à utiliser.
Le premier élément commence à 0.
S'il est égal à #PB_All, la couleur sera appliquée à tous les éléments de la colonne spécifiée.

TypeCouleur Peut prendre l'une des valeurs suivantes :

```
#PB_Gadget_FrontColor:
 Texte de l'élément.
#PB_Gadget_BackColor :
 Fond de l'élément.
```

Couleur La nouvelle couleur de l'attribut.
RGB() peut être utilisé pour obtenir une couleur valide.
Pour supprimer la couleur personnalisée et revenir à la couleur par défaut, utiliser la valeur #PB_Default.

Colonne (optionnel) La colonne à utiliser pour les gadgets qui prennent en charge plusieurs colonnes.
La première colonne a indice 0.
La valeur par défaut est 0 colonne.
Si le paramètre 'Colonne' est égal à #PB_All, la couleur sera appliquée à toutes les colonnes de l'élément spécifié.

Valeur de retour

Aucune.

Remarques

Cette commande est supportée par les gadgets suivants :

- `ListIconGadget()`

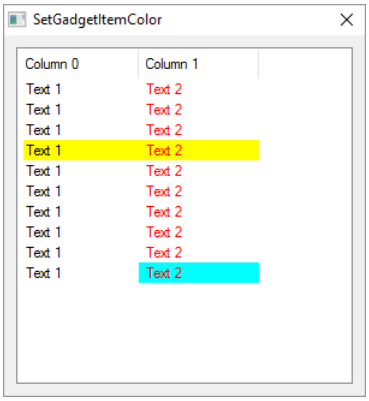
- `TreeGadget()`

Note : Avec le support des skins activé sur Windows XP, les couleurs personnalisées ne seront probablement pas prises en compte sur certains gadgets.

Exemple

```
If OpenWindow(0, 0, 0, 300, 300, "SetGadgetItemColor", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  ListIconGadget(0, 10, 10, 280, 280, "Colonne 0", 100)
  AddGadgetColumn(0, 1, "Colonne 1", 100)
  For i = 1 To 10
    AddGadgetItem(0, -1, "Texte "+Str(i)+Chr(10)+"Texte "+Str(i))
  Next

  la numérotation commence à 0 pour les éléments/colonnes
  SetGadgetItemColor(0, #PB_All, #PB_Gadget_FrontColor, $0000FF,  1) ; tous les éléments de la 2ème colonne en rouge
  SetGadgetItemColor(0,  3, #PB_Gadget_BackColor, $00FFFF, -1) ; le 4ème élément de toutes les colonnes sur fond jaune
```

```
10      SetGadgetItemColor(0,  9,
        #PB_Gadget_BackColor,
        $FFFF00,  1) ; le 10ème
        élément de la 2ème colonne
        sur fond bleu
11      Repeat
12      Until WaitWindowEvent() =
        #PB_Event_CloseWindow
13    EndIf
```



Voir aussi

GetGadgetItemColor() , GetGadgetColor() , SetGadgetColor()

OS Supportés

Tous

99.68 SetGadgetItemData

Syntaxe

```
SetGadgetItemData(#Gadget,
    Element, Valeur)
```

Description

Associe une valeur à un élément d'un gadget.

Arguments

#Gadget Le gadget à utiliser.

Element L'élément à utiliser.
Le premier élément commence à 0.

Valeur La valeur à définir.

Valeur de retour

Aucune.

Remarques

Cette valeur peut être récupérée avec la commande GetGadgetItemData() .
Si l'index de l'élément change (par exemple d'autres éléments sont effacés), la valeur restera toujours associée à son élément.
Cette commande est supportée par les gadgets suivants :
- ComboBoxGadget()
- ListIconGadget()
- ListViewGadget()
- PanelGadget()
- TreeGadget()

Exemple

```
1  ; Ce programme utilise SetGadgetItemData pour stocker la position d'origine
2  ; de chaque élément pour la connaitre plus tard, même si l'index des éléments a changé.
3  ;
4  If OpenWindow(0, 0, 0, 290, 250, "SetGadgetItemData", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
5     ButtonGadget(0,  10, 10, 80, 20, "Ajouter")
6     ButtonGadget(1, 100, 10, 80, 20, "Supprimer")
7     ButtonGadget(2, 190, 10, 80, 20, "Test")
8     ListViewGadget(3, 10, 40, 260, 200)
9     For i = 0 To 10
10       AddGadgetItem(3, i, "Ancien élément "+Str(i))
11       SetGadgetItemData(3, i, i)
12    Next i
13
14    Repeat
15       Evenement = WaitWindowEvent()
16       If Evenement = #PB_Event_Gadget
17          element = GetGadgetState(3)
18
19          Select EventGadget()
20            Case 0 ; Ajouter
21              AddGadgetItem(3, element, "Nouvel élément")
22              If element <> -1
```

```
23                SetGadgetItemData(3,
    element, -1)
24              Else
25                SetGadgetItemData(3,
    CountGadgetItems(3)-1, -1)
26              EndIf
27
28            Case 1 ; Supprimer
29              If element <> -1
30                RemoveGadgetItem(3,
    element)
31              EndIf
32
33            Case 2 ; Test
34              If element <> -1
35                valeur =
    GetGadgetItemData(3,
    element)
36                If valeur = -1
37                  MessageRequester("",
    "C'est un nouvel élément")
38                Else
39                  MessageRequester("",
    "C'était l'élément numéro
    "+Str(valeur))
40                EndIf
41              EndIf
42
43          EndSelect
44        EndIf
45      Until Evenement =
    #PB_Event_CloseWindow
46    EndIf
```

Voir aussi

GetGadgetItemData() , GetGadgetData() , SetGadgetData()

OS Supportés

Tous

99.69 SetGadgetItemImage

Syntaxe

```
SetGadgetItemImage(#Gadget,
    Element, ImageID)
```

Description

Change l'image d'un élément.

Arguments

#**Gadget** Le gadget à utiliser.

Element L'élément concerné.
Le premier élément dans le gadget commence à zéro.

ImageID La nouvelle image à utiliser pour l'élément.
L'image utilisée doit être dans le format standard 16x16.
Utilisez la commande ImageID() pour obtenir l'ID de l'image.

Valeur de retour

Aucune.

Remarques

Il s'agit d'une fonction universelle qui fonctionne avec les gadgets suivants :

- ComboBoxGadget()
- ListIconGadget()
- PanelGadget()
(non pris en charge sur OS X)
- TreeGadget()

Voir aussi

AddGadgetItem()

OS Supportés

Tous

99.70 SetGadgetItemState

Syntaxe

```
SetGadgetItemState(#Gadget,
    Element, Etat)
```

Description

Change l'état d'un élément d'un gadget.

Arguments

#Gadget Le gadget à utiliser.

Element L'élément à utiliser.
Le premier élément commence à 0.

Etat Le nouvel état de l'élément.

Valeur de retour

Aucune.

Remarques

Cette commande est disponible pour les gadgets suivants :

```
  - CalendarGadget()
: 'Etat'
    #PB_Calendar_Bold :
   Afficher la date en gras.
    #PB_Calendar_Normal
   Afficher la date sans
   graisse (normal).
                       'Element'
   est une date au format
   PureBasic.

  - ExplorerListGadget()
: 'Etat'
    #PB_Explorer_Selected :
   L'élément sera selectionné.

  - ListViewGadget()
: 'Etat' = 1 si l'élément
   doit être sélectionné, 0
   sinon.

  - ListIconGadget()
: 'Etat' est une combinaison
   des valeurs suivantes:
    #PB_ListIcon_Selected :
   L'élément sera selectionné.
    #PB_ListIcon_Checked  :
   L'élément aura sa case à
   cocher associée cochée
   (créé avec l'option
   #PB_ListIcon_CheckBoxes).
    #PB_ListIcon_Inbetween:
   L'élément doit avoir sa
   case à cocher associée
   dans l'état indéterminé
   (#PB_ListIcon_ThreeState).

  - TreeGadget()
    : 'Etat' est une
   combinaison des valeurs
   suivantes:
    #PB_Tree_Selected :
   L'élément sera sélectionné.
    #PB_Tree_Expanded :
   L'élément sera déployé
   (noeud ouvert).
    #PB_Tree_Collapsed:
   L'élément sera fermé.
    #PB_Tree_Checked  : La
   case à cocher associée à
   l'élément sera cochée
   (créé avec l'option
   #PB_Tree_CheckBoxes).
    #PB_Tree_Inbetween:
   L'élément doit avoir sa
```

```
    case à cocher associée
    dans l'état indéterminé.
```

Exemple

```
1   SetGadgetItemState(0, 1,
     #PB_Tree_Expanded |
     #PB_Tree_Selected) ;
     l'élément 2 sera déployé
     et sélectionné.
```

Voir aussi

GetGadgetItemState() , GetGadgetState() , SetGadgetState()

OS Supportés

Tous

99.71 SetGadgetItemText

Syntaxe

```
SetGadgetItemText(#Gadget,
    Element, Texte$ [,
    Colonne])
```

Description

Change le texte d'un élément d'un gadget.

Arguments

#Gadget Le gadget à utiliser.

Element L'élément à utiliser.
Le premier élément commence à 0.

Texte$ Le nouveau texte de l'élément.

Colonne (optionnel) La colonne à utiliser pour les gadgets qui prennent en charge plusieurs colonnes.
La première colonne a indice 0.
La valeur par défaut est 0 colonne.

Valeur de retour

Aucune.

Remarques

Cette commande est disponible pour les gadgets suivants :

```
  - ComboBoxGadget()

  - EditorGadget()

  - ExplorerListGadget()
: Si Element = -1 alors
   l'entête de la colonne
   indiquée est changé.
  - ListIconGadget()
    : Si Element = -1 alors
   l'entête de la colonne
   indiquée est changé.
  - ListViewGadget()

  - MDIGadget()

  - PanelGadget()

  - TreeGadget()

  - WebGadget()
: Change le code html dans le
   gadget avec
   #PB_Web_HtmlCode comme
   'Element'.
```

Voir aussi

GetGadgetItemText() , GetGadgetText() , SetGadgetText()

OS Supportés

Tous

99.72 SetGadgetState

Syntaxe

```
SetGadgetState(#Gadget, Etat)
```

Description

Change l'état d'un gadget.

Arguments

#Gadget Le gadget à utiliser.
Etat Le nouvel état de l'élément.

Valeur de retour

Aucune.

Remarques

Cette fonction est universelle et s'applique à pratiquement tous les gadgets :

```
  - ButtonGadget()
     : change l'état d'un
    bouton poussoir
    (#PB_Button_Toggle): 1 =
    pressé, 0 = normal.
  - ButtonImageGadget()
: change l'état d'un bouton
    poussoir
    (#PB_Button_Toggle): 1 =
    pressé, 0 = normal.
  - CalendarGadget()
    : Change la date affichée.
  - CheckBoxGadget()
    : Change l'etat de la case
    à cocher. Valeurs
    disponibles:
     #PB_CheckBox_Checked  :
    Coche la case.
     #PB_CheckBox_Unchecked:
    Décoche la case.
     #PB_CheckBox_Inbetween:
    Active l'état intermédiaire
                              (seulement
    pour les case à cocher de
    type
    #PB_CheckBox_ThreeState).
  - ComboBoxGadget()
    : Change l'élément
    sélectionné en cours.
  - DateGadget()
       : Change la date ou
    l'heure affichée. Si
    #PB_Date_CheckBox est
    utilisé,
                            mettre
    'Etat' à 0 pour décocher
    la case à cocher.
  - ImageGadget()
      : Change l'image
    affichée par le gadget
    ('Etat' sera un ImageID()
).
                            Si
    Etat = 0, l'image est
    retirée du gadget.
  - IPAddressGadget()
  : change l'adresse IP
    courante.
  - ListIconGadget()
    : Change l'élément
    sélectionné (tous les
    autres éléments sont alors
    désélectionnés).
                            Si
    'Etat' = -1, alors aucun
```

```
    élément n'est sélectionné.
  - ListViewGadget()
    : Change l'élément
    sélectionné. Si 'Etat' =
    -1, alors aucun élément
    n'est sélectionné.
  - MDIGadget()
        : Change la fenêtre
    fille active (En indiquant
    son numéro de #fenetre),
    ou exécute une des actions
    suivantes:
     #PB_MDI_Cascade          :
    Ré-organise les fenêtres
    filles sous forme de
    cascade.
     #PB_MDI_TileVertically   :
    Ré-organise les fenêtres
    filles sous forme de
    mosaïque verticale.
     #PB_MDI_TileHorizontally:
    Ré-organise les fenêtres
    filles sous forme de
    mosaïque horizontale.
     #PB_MDI_Next             :
    Active la fenêtre fille
    suivante.
     #PB_MDI_Previous         :
    Active la fenêtre fille
    précédente.
     #PB_MDI_Arrange          :
    Ré-organise les fenêtres
    filles qui sont réduites.

  - OptionGadget()
      : 'Etat' = 1 pour
    l'activer, sinon 0.
  - PanelGadget()
      : Change l'onglet en
    cours.
  - ProgressBarGadget()
: Change la valeur de la
    barre de progression.
  - ScrollBarGadget()
  : Change la position du
    curseur.
  - ShortcutGadget()
    : Change le raccourci
    clavier.
  - SpinGadget()
       : Change la valeur
    actuelle.
  - SplitterGadget()
    : Change la position de la
    barre de séparation (en
    pixels).
  - TrackBarGadget()
    : Change la valeur du
    curseur. La valeur
```

```
  #PB_ProgressBar_Unknown
                        peut
  être utilisée pour
  indiquer que la
  progression est inconnue.
- TreeGadget()
      : Change l'élément
  actuellement sélectionné,
  -1 pour tout
  déselectionner.
- WebGadget()
       : Effectue des
  actions sur le navigateur.
```

Voir aussi

GetGadgetState() , GetGadgetItemState() , SetGadgetItemState()

OS Supportés

Tous

99.73 SetGadgetText

Syntaxe

```
SetGadgetText(#Gadget, Texte$)
```

Description

Modifie le texte contenu dans un gadget.

Arguments

#Gadget Le gadget à utiliser.

Texte$ Le nouveau texte de l'élément.

Valeur de retour

Aucune.

Remarques

Cette fonction est particulièrement utile pour :
- ButtonGadget()
- ComboBoxGadget()
- DateGadget()
- EditorGadget()
- ExplorerComboGadget()
- ExplorerListGadget()
- ExplorerTreeGadget()
- FrameGadget()
- HyperLinkGadget()
- ListViewGadget()
- StringGadget()

- TextGadget()
- TreeGadget()
- WebGadget()

Voir aussi

GetGadgetText() , GetGadgetItemText() , SetGadgetItemText()

OS Supportés

Tous

99.74 ShortcutGadget

Syntaxe

```
Resultat =
    ShortcutGadget(#Gadget, X,
    Y, Largeur, Hauteur,
    Raccourci)
```

Description

Crée un gadget raccourci clavier dans la GadgetList.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Raccourci La combinaison de touches.
Les valeurs possibles sont les mêmes que celles citées dans AddKeyboardShortcut() .
La valeur 0 indique qu'aucun raccourci n'est défini.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

- GadgetToolTip() permet d'ajouter une 'mini aide' à ce gadget.
Les fonctions suivantes peuvent être appelées pour contrôler le gadget :

- GetGadgetState() : Renvoie le raccourci clavier actuel.
- SetGadgetState() : Change le raccourci clavier actuel.

Exemple

```
1  If OpenWindow(0, 0, 0, 240,
     70, "ShortcutGadget",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2     ShortcutGadget(0, 20, 20,
     200, 25,
     #PB_Shortcut_Control |
     #PB_Shortcut_A)
3     Repeat
4       Event =
     WaitWindowEvent()
5     Until Event =
     #PB_Event_CloseWindow
6  EndIf
```

Voir aussi

GetGadgetState() , SetGadgetState() , AddKeyboardShortcut()

OS Supportés

Tous

99.75 SpinGadget

Syntaxe

```
Resultat =
    SpinGadget(#Gadget, X, Y,
    Largeur, Hauteur, Minimum,
    Maximum [, Options])
```

Description

Crée un gadget incrémentiel dans la GadgetList.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Minimum, Maximum Les valeurs minimales et maximales.

Options (optionnel) Peut être une combinaison de :

```
    #PB_Spin_ReadOnly: Le
     gadget n'est pas
     éditable, sa valeur ne
     peut être changée que
     par les flèches (pas
     supporté par Linux).
    #PB_Spin_Numeric : Le
     gadget va mettre à jour
     automatiquement le texte
     affiché, donc
     SetGadgetText()
  n'est pas nécessaire.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

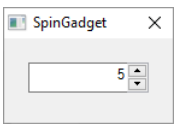
Remarques

```
  - GadgetToolTip()
 permet d'ajouter une 'mini
   aide' à ce gadget.

  Les fonctions suivantes
   peuvent être appelées pour
   contrôler le gadget:
  GetGadgetState()
    : Renvoie la valeur du
   curseur interne.
  SetGadgetState()
    : Change la valeur du
   curseur interne.
  GetGadgetText()
     : Renvoie le texte du
   gadget.
  SetGadgetText()
     : Change le texte du
   gadget.
  GetGadgetAttribute()
 avec un des attributs
   suivants:
    #PB_Spin_Minimum: Renvoie
   la valeur minimale
   possible.
    #PB_Spin_Maximum: Renvoie
   la valeur maximale
   possible.

  SetGadgetAttribute()
: avec un des attributs
   suivants:
```

```
    #PB_Spin_Minimum: Change
    la valeur minimale
    possible.
    #PB_Spin_Maximum: Change
    la valeur maximale
    possible.

  Les évènements suivants
    sont supportés par
    EventType()
:
    #PB_EventType_Change:
    L'utilisateur a changé le
    texte dans la zone
    d'édition.
    #PB_EventType_Up    : Le
    bouton 'Haut' a été pressé.
    #PB_EventType_Down  : Le
    bouton 'Bas' a été pressé.

  - SetGadgetColor()
 et GetGadgetColor()
 avec les valeurs
    'TypeCouleur' suivantes:
    #PB_Gadget_FrontColor:
    couleur du texte
    #PB_Gadget_BackColor :
    couleur du fond
```

Exemple

```
1   If OpenWindow(0, 0, 0, 160,
    70, "SpinGadget",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2    SpinGadget     (0, 20,
    20, 100, 25, 0, 1000)
3    SetGadgetState (0, 5) :
    SetGadgetText(0, "5")   ;
    définit la valeur initiale
4    Repeat
5      Evenement =
    WaitWindowEvent()
6      If Evenement =
    #PB_Event_Gadget
7        If EventGadget() = 0
8          SetGadgetText(0,Str(GetGadgetState(0)))
9        EndIf
10     EndIf
11   Until Evenement =
    #PB_Event_CloseWindow
12  EndIf
```



Voir aussi

GetGadgetState() , SetGadgetState() , GetGadgetText() , SetGadgetText() , GetGadgetAttribute() , SetGadgetAttribute() , GetGadgetColor() , SetGadgetColor()

OS Supportés

Tous

99.76 SplitterGadget

Syntaxe

```
Resultat =
    SplitterGadget(#Gadget, X,
    Y, Largeur, Hauteur,
    #Gadget1, #Gadget2 [,
    Options])
```

Description

Crée une zone séparée en deux parties (horizontalement ou verticalement), chacune contenant un gadget fils.
La barre de séparation peut être déplacée librement par l'utilisateur et les deux gadgets associés seront redimensionnés automatiquement.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

#Gadget1, #Gadget2 Les gadgets fils.

Options (optionnel) Peut être une combinaison de :

```
#PB_Splitter_Vertical   :
 La zone sera séparée
 verticalement.
#PB_Splitter_Separator  :
 Un motif 3D est affiché
 dans la barre de
 séparation.
#PB_Splitter_FirstFixed :
 Quand le SplitterGadget
 sera redimensionné, le
 premier gadget associé
 gardera sa taille.
```

```
    #PB_Splitter_SecondFixed:
     Quand le SplitterGadget
     sera redimensionné, le
     deuxième gadget associé
     gardera sa taille.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

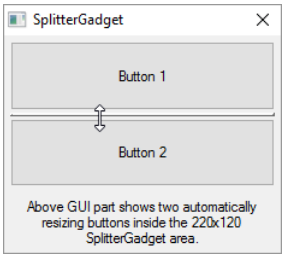
```
  - GadgetToolTip()
 permet d'ajouter une 'mini
   aide' à ce gadget.

  Les fonctions suivantes
   peuvent être utilisées
   pour agir sur le gadget:
  GetGadgetState()
: Renvoie la position de la
   barre de séparation, en
   pixels.
  SetGadgetState()
: Change la position de la
   barre de séparation, en
   pixels.
  GetGadgetAttribute()
 avec un des attributs
   suivants:
    #PB_Splitter_FirstMinimumSize
   : Renvoie la taille
   minimale (en pixels) que
   le premier gadget associé
   peut avoir.
    #PB_Splitter_SecondMinimumSize:
   Renvoie la taille minimale
   (en pixels) que le
   deuxième gadget associé
   peut avoir.
    #PB_Splitter_FirstGadget
        : Renvoie le numéro
   du premier gadget.
    #PB_Splitter_SecondGadget
       : Renvoie le numéro du
   deuxième gadget.

  SetGadgetAttribute()
 avec un des attributs
   suivants:
    #PB_Splitter_FirstMinimumSize
   : Fixe la taille minimale
   (en pixels) que le premier
   gadget associé peut avoir.
```

```
    #PB_Splitter_SecondMinimumSize:
    Fixe la taille minimale
    (en pixels) que le
    deuxième gadget associé
    peut avoir.
    #PB_Splitter_FirstGadget
        : Remplace le premier
    gadget avec un nouveau.
    #PB_Splitter_SecondGadget
       : Remplace le deuxième
    gadget avec un nouveau.

  Note: Quand un gadget est
    remplacé à l'aide de
    SetGadgetAttribute()
, l'ancien gadget n'est pas
    automatiquement supprimé.
  Il sera remis sur la
    fenêtre qui contient le
    splitter, pour permettre
    de permuter les gadgets
    entre différents splitters
  sans avoir à les recréer.
    Si l'ancien gadget doit
    être supprimé, son numéro
    peut être recupéré avec
    GetGadgetAttribute()

  puis supprimé par
    FreeGadget()
 après son remplacement. Un
    gadget ne peut pas être
    dans deux splitters à la
    fois.
```

Exemple

```
1  If OpenWindow(0, 0, 0, 240,
     230, "SplitterGadget",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2    #Bouton1    = 0
3    #Bouton2    = 1
4    #Separateur = 2
5
6    ButtonGadget(#Bouton1, 0,
     0, 0, 0, "Bouton 1") ;
     Inutile de préciser la
     taille ou les coordonnées
7    ButtonGadget(#Bouton2, 0,
     0, 0, 0, "Bouton 2") ; car
     elles seront déterminées
     automatiquement
8    SplitterGadget(#Separateur,
     5, 5, 230, 130, #Bouton1,
     #Bouton2,
     #PB_Splitter_Separator)
9
```

```
10      TextGadget(3, 10, 135,
        210, 90, "Le composant
        d'interface graphique
        ci-dessus affiche deux
        boutons automatiquement
        redimensionnés à
        l'intérieur de la zone
        séparée de taille
        230x130.",#PB_Text_Center)
11
12      Repeat
13      Until WaitWindowEvent() =
        #PB_Event_CloseWindow
14    EndIf
```



Voir aussi

GetGadgetState() , SetGadgetState() , GetGadgetAttribute() , SetGadgetAttribute()

OS Supportés

Tous

99.77 StringGadget

Syntaxe

```
Resultat =
    StringGadget(#Gadget, X,
    Y, Largeur, Hauteur,
    Texte$ [, Options])
```

Description

Crée un gadget de saisie de texte (une seule ligne) dans la GadgetList en cours.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Texte$ Le texte initial.
Ce gadget accepte une seule ligne de texte. Pour obtenir plusieurs lignes d'entrée, utiliser le gadget EditorGadget() .

Options (optionnel) Peut être une combinaison de :

```
#PB_String_Numeric   :
 Seuls des nombres
 entiers positifs peuvent
 être saisis.
#PB_String_Password  :
 Mode 'mot de passe',
 n'affiche que des '*'.
#PB_String_ReadOnly  :
 Mode 'lecture seulement'.
#PB_String_LowerCase :
 Tous les caractères
 saisis sont transformés
 en minuscules.
#PB_String_UpperCase :
 Tous les caractères
 saisis sont transformés
 en majuscules.
#PB_String_BorderLess:
 Aucune bordure n'est
 affichée autour du
 gadget.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

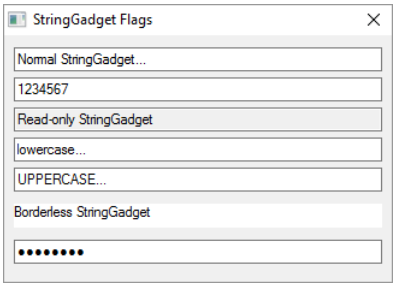
```
  - GadgetToolTip()
 permet d'ajouter une 'mini
   aide' à ce gadget.

  - Le texte pourra être
   ensuite modifié par les
   fonctions SetGadgetText()
 et GetGadgetText()
.

  - Les évènements suivants
   sont disponibles par
   l'intermédiaire
   d'EventType()
:
    #PB_EventType_Change    :
   Le texte a été modifié par
   l'utilisateur.
```

```
   #PB_EventType_Focus     :
  Le StringGadget obtient le
  focus.
   #PB_EventType_LostFocus :
  Le StringGadget a perdu le
  focus.

 - Les fonctions suivantes
  peuvent agir sur le gadget:
   SetGadgetColor()
et GetGadgetColor()
avec les valeurs
  'TypeCouleur' suivantes:
     #PB_Gadget_BackColor
  : Couleur de fond
     #PB_Gadget_FrontColor
  : Couleur du texte

 - GetGadgetAttribute()
avec les attributs suivants:
     #PB_String_MaximumLength:
  Renvoie le nombre maximal
  de caractères qui peuvent
  être entrés.

 - SetGadgetAttribute()
avec les attributs suivants:
     #PB_String_MaximumLength:
  Limite le nombre maximum
  de caractères qui peuvent
  être entrés.
```

Exemple

```
1  ; Démonstration des options
     possibles pour le gadget
     de saisie de texte...
2  If OpenWindow(0, 0, 0, 322,
     205, "Les options de
     StringGadget",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
3    StringGadget(0, 8,  10,
     306, 20, "Gadget saisie de
     texte normal...")
4    StringGadget(1, 8,  35,
     306, 20, "1234567",
     #PB_String_Numeric)
5    StringGadget(2, 8,  60,
     306, 20, "Gadget saisie de
     texte en lecture seule",
     #PB_String_ReadOnly)
6    StringGadget(3, 8,  85,
     306, 20, "minuscules...",
     #PB_String_LowerCase)
7    StringGadget(4, 8, 110,
     306, 20, "majuscules...",
     #PB_String_UpperCase)
```

```
8     StringGadget(5, 8, 140,
      306, 20, "Gadget saisie de
      texte sans bordure",
      #PB_String_BorderLess)
9     StringGadget(6, 8, 170,
      306, 20, "Mot de passe",
      #PB_String_Password)
10    Repeat : Until
      WaitWindowEvent() =
      #PB_Event_CloseWindow
11  EndIf
```



Voir aussi

GetGadgetText() , SetGadgetText() , GetGadgetColor() , SetGadgetColor() , EditorGadget()

OS Supportés

Tous

99.78 TextGadget

Syntaxe

```
Resultat =
    TextGadget(#Gadget, X, Y,
    Largeur, Hauteur, Texte$
    [, Options])
```

Description

Crée un gadget étiquette dans la GadgetList en cours.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Texte$ Le texte à afficher.

Options (optionnel) Peut être une combinaison de :

```
#PB_Text_Center: Le texte
 est centré.
#PB_Text_Right : Le texte
 est aligné à droite.
#PB_Text_Border: Un bord
 est dessiné autour du
 gadget.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

Son contenu peut être modifié à l'aide des fonctions SetGadgetText() et GetGadgetText() . La police d'un TextGadget() peut facilement être changée avec SetGadgetFont() .
Ce gadget supporte les commandes SetGadgetColor() et GetGadgetColor() avec les valeurs 'TypeCouleur' suivantes :

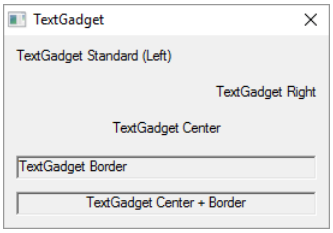
```
#PB_Gadget_BackColor :
 Couleur de fond
#PB_Gadget_FrontColor:
 Couleur du texte
```

Note : Ce gadget ne reçoit pas les évènements utilisateur.
Note : GadgetToolTip() ne fonctionne qu'avec Linux.

Exemple

```
1  If
    OpenWindow(0,0,0,270,160,"TextGadget",#PB_Window_SystemMenu
    |
    #PB_Window_ScreenCentered)
2    TextGadget(0, 10,
    10,250,20,"Gadget texte
    standard (texte aligné à
    gauche)")
3    TextGadget(1, 10,
    40,250,20,"Gadget texte
    (texte aligné à droite)",
    #PB_Text_Right)
4    TextGadget(2, 10,
    70,250,20,"Gadget texte
    (texte
    centré)",#PB_Text_Center)
5    TextGadget(3,
    10,100,250,20,"Gadget
    texte avec
    bordure",#PB_Text_Border)
```

```
6     TextGadget(4,
      10,130,250,20,"Gadget
      texte (texte centré) +
      bordure", #PB_Text_Center
      | #PB_Text_Border)
7     Repeat : Until
      WaitWindowEvent()=#PB_Event_CloseWindow
8   EndIf
```



Voir aussi

GetGadgetText() , SetGadgetText() , GetGadgetColor() , SetGadgetColor()

OS Supportés

Tous

99.79 TrackBarGadget

Syntaxe

```
Resultat =
    TrackBarGadget(#Gadget, X,
    Y, Largeur, Hauteur,
    Minimum, Maximum [,
    Options])
```

Description

Crée un gadget Curseur dans la GadgetList en cours.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Minimum, Maximum La plage 'Minimum-Maximum' doit être comprise entre 0 et 10 000.

Options (optionnel) Peut être une combinaison de :

```
#PB_TrackBar_Ticks   :
 Affiche un trait de
 marquage à chaque valeur.
#PB_TrackBar_Vertical: Le
 gadget est vertical.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

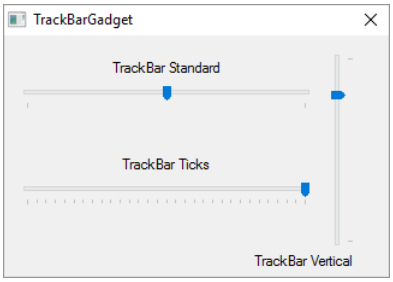
Remarques

```
  - GadgetToolTip()
 permet d'ajouter une 'mini
   aide' à ce gadget.

  Les commandes suivantes
   peuvent être utilisées
   pour contrôler le gadget:
  - GetGadgetState()
: Renvoie la position
   actuelle du curseur
   (Valeur comprise entre
   'Minimum' et 'Maximum')
  - SetGadgetState()
: Change la position actuelle
   du curseur.
  - GetGadgetAttribute()
 avec l'un des attributs
   suivants:
    #PB_TrackBar_Minimum:
   Renvoie la valeur minimale.
    #PB_TrackBar_Maximum:
   Renvoie la valeur maximale.

  - SetGadgetAttribute()
 avec un des attributs
   suivants:
    #PB_TrackBar_Minimum:
   Change la valeur minimale.
    #PB_TrackBar_Maximum:
   Change la valeur maximale.
```

Exemple

```
1 If OpenWindow(0, 0, 0, 320,
    200, "TrackBarGadget",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2   TextGadget    (3, 10,
    20, 250, 20,"Barre à
    curseur standard",
    #PB_Text_Center)
```

```
3     TrackBarGadget(0, 10,
      40, 250, 20, 0, 10000)
4     SetGadgetState(0, 5000)
5     TextGadget    (4, 10,
      100, 250, 20, "Barre à
      curseur avec traits de
      marquage", #PB_Text_Center)
6     TrackBarGadget(1, 10,
      120, 250, 20, 0, 30,
      #PB_TrackBar_Ticks)
7     SetGadgetState(1, 3000)
8     TextGadget    (5,  90,
      180, 200, 20, "Barre à
      curseur verticale",
      #PB_Text_Right)
9     TrackBarGadget(2, 270,
      10, 20, 170, 0, 10000,
      #PB_TrackBar_Vertical)
10    SetGadgetState(2, 8000)
11    Repeat : Until
      WaitWindowEvent() =
      #PB_Event_CloseWindow
12  EndIf
```



Voir aussi

GetGadgetState() , SetGadgetState() , GetGadgetAttribute() , SetGadgetAttribute()

OS Supportés

Tous

99.80 TreeGadget

Syntaxe

```
Resultat =
    TreeGadget(#Gadget, X, Y,
    Largeur, Hauteur [,
    Options])
```

Description

Crée un gadget liste arborescente dans la GadgetList en cours.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Options (optionnel) Peut être une combinaison de :

```
#PB_Tree_AlwaysShowSelection:
 Montre toujours la
 sélection, même si le
 gadget n'est plus actif.
#PB_Tree_NoLines
            : Cache les
 lignes reliant les
 noeuds.
#PB_Tree_NoButtons
          : Cache les
 boutons des noeuds.
#PB_Tree_CheckBoxes
         : Ajoute une
 case à cocher devant
 chaque élément.
#PB_Tree_ThreeState
         : Les cases à
 cocher peuvent avoir un
 état "indéterminé".
```

L'option #PB_Tree_ThreeState peut être utilisée en combinaison avec l'option #PB_Tree_CheckBoxes pour obtenir des cases à cocher qui peuvent avoir un état "on", "off" et "indéterminé". L'utilisateur ne peut sélectionner que les états "on" ou "off". L'état "indéterminé" peut être défini en utilisant la fonction SetGadgetItemState() .

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.

Remarques

Chaque branche de l'arbre a un niveau assigné qui détermine la relation qu'il a avec l'élément du dessus et du dessous. Les éléments avec le même niveau appartiennent à la même branche, les

éléments avec un niveau supérieur sont les fils etc. Le paramètre 'Options' de la commande AddGadgetItem() est toujours requis pour les éléments du TreeGadget() et sert à spécifier le niveau de l'élément ajouté. A noter que si un élément est ajouté avec un niveau invalide alors il sera quand même ajouté, au niveau le plus proche.

```
  - GadgetToolTip()
 permet d'ajouter une 'mini
   aide' à ce gadget.

  Les fonctions suivantes
   peuvent alors être
   utilisées pour agir sur le
   contenu de cette liste :
  - AddGadgetItem()
     : Ajoute un élément, et
   éventuellement une icône
  - RemoveGadgetItem()
  : Efface un élément (et ses
   sous-éléments)
  - ClearGadgetItems()
  : Supprime tous les éléments
  - CountGadgetItems()
  : Renvoie le nombre
   d'éléments actuellement
   contenus dans le gadget.
  - GetGadgetItemState()
: Renvoie l'état de l'élément
   spécifié.
  - SetGadgetItemState()
: Change l'état de l'élément
   spécifié.
  - GetGadgetItemText()
 : Renvoie le texte de
   l'élément spécifié.
  - SetGadgetItemText()
 : Change le texte de
   l'élément spécifié.
  - SetGadgetItemImage()
: Change l'image actuelle de
   l'élément spécifié.
  - GetGadgetItemData()
 : Renvoie la valeur
   personnalisée associée à
   cet élément.
  - SetGadgetItemData()
 : Associe une valeur
   personnalisée à cet
   élément.
  - GetGadgetState()
     : Renvoie l'élément
   selectionné.
  - SetGadgetState()
     : Change l'élément
   selectionné.
  - GetGadgetText()
```

```
    : Renvoie le texte de
   l'élément sélectionné.
  - SetGadgetText()
    : Change le texte de
   l'élément sélectionné.
  - GetGadgetItemAttribute()
 avec l'attribut suivant:
    #PB_Tree_SubLevel:
   Renvoie le niveau actuel
   de l'élément dans l'arbre.

  - GadgetItemID()
: Renvoie le 'handle' système
   de l'élément spécifié
   (utile pour les fonctions
   API)

  - SetGadgetColor()
 et GetGadgetColor()
 avec les  valeurs
   'TypeCouleur' suivantes:
    #PB_Gadget_FrontColor:
   Couleur du texte
    #PB_Gadget_BackColor :
   Couleur du fond
    #PB_Gadget_LineColor :
   Couleur des lignes reliant
   les noeuds

  - SetGadgetItemColor()
 et GetGadgetItemColor()
 avec les valeurs
   'TypeCouleur' suivantes:
    #PB_Gadget_FrontColor:
   Texte de l'élément.
    #PB_Gadget_BackColor :
   Fond de l'élément.

  Les évènements suivants
   sont supportés par
   EventType()
:
    #PB_EventType_LeftClick
          : Clic gauche sur
   un élément, ou une case à
   cocher a été
   cochée/décochée.
    #PB_EventType_LeftDoubleClick
    : Double-clic gauche sur
   un élément.
    #PB_EventType_RightClick
         : Clic droit sur un
   élément.
    #PB_EventType_RightDoubleClick
   : Double-Clic droit sur un
   élément.
    #PB_EventType_Change
             : L'élément
   courant a changé
```

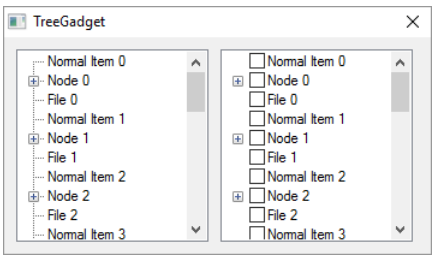
#PB_EventType_DragStart
: L'utilisateur a essayé de lancer une opération 'Glisser & Déposer'.

Après un évènement #PB_EventType_DragStart, la bibliothèque Drag & Drop peut être utilisée pour commencer une opération de 'Glisser & Déposer'.

Exemple

```
If OpenWindow(0, 0, 0, 355, 180, "TreeGadget", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  TreeGadget(0, 10, 10, 160, 160)

  ; liste arborescente standard
  TreeGadget(1, 180, 10, 160, 160, #PB_Tree_CheckBoxes | #PB_Tree_NoLines)  ; liste arborescente avec cases à cocher + sans lignes reliant les noeuds
  For ID = 0 To 1
    For a = 0 To 10
      AddGadgetItem(ID, -1, "Elément normal "+Str(a), 0, 0) ; si vous souhaitez ajouter une image, utilisez
      AddGadgetItem(ID, -1, "Noeud "+Str(a), 0, 0)
          ; ImageID(x) comme 4ème paramètre
      AddGadgetItem(ID, -1, "Sous-élément 1", 0, 1)
          ; Ceux-là sont au premier sous-niveau
      AddGadgetItem(ID, -1, "Sous-élément 2", 0, 1)
      AddGadgetItem(ID, -1, "Sous-élément 3", 0, 1)
      AddGadgetItem(ID, -1, "Sous-élément 4", 0, 1)
      AddGadgetItem(ID, -1, "Fichier "+Str(a), 0, 0) ; sous-niveau 0 à nouveau
    Next
  Next
  Repeat : Until WaitWindowEvent() =
```

```
    #PB_Event_CloseWindow
16  EndIf
```



Voir aussi

AddGadgetItem() , RemoveGadgetItem() , ClearGadgetItems() , CountGadgetItems() , GetGadgetItemState() , SetGadgetItemState() , GetGadgetItemText() , SetGadgetItemText() , SetGadgetItemImage() , GetGadgetItemData() , SetGadgetItemData() , GetGadgetState() , SetGadgetState() , GetGadgetText() , SetGadgetText() , GetGadgetItemAttribute() , SetGadgetItemImage() , GadgetItemID() , GetGadgetColor() , SetGadgetColor() , ExplorerTreeGadget()

OS Supportés

Tous

99.81 UseGadgetList

Syntaxe

```
Resultat =
    UseGadgetList(FenetreID)
```

Description

Sélectionne la GadgetList dans laquelle les gadgets seront ajoutés.
Si la GadgetList n'existe pas pour cette fenêtre alors elle sera créée.
(Par exemple dans le cas d'une fenêtre créée avec l'option #PB_Window_NoGadgets, voir OpenWindow() ou parce que ce n'est pas une fenêtre PB)

Arguments

FenetreID L'identifiant de la nouvelle fenêtre dans laquelle les gadgets seront ajoutés.

'FenetreID' peut être obtenu facilement avec la commande WindowID() .
Si égal à 0, c'est la GadgetList courante qui sera renvoyée par la commande et il n'y aura aucun changement.

Valeur de retour

Renvoie l'identifiant 'FenetreID' de la GadgetList précédente, ou zéro s'il n'y en a pas.
Cette valeur peut être utilisée pour revenir à la GadgetList précédente.

Exemple

Cet exemple montre comment utiliser cette commande pour créer une nouvelle fenêtre avec des gadgets sans interrompre la création de gadgets sur la première fenêtre :

```
1   If OpenWindow(0, 0, 0, 500, 500, "Fenêtre principale", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2    ButtonGadget(0, 10, 10, 150, 25, "Bouton 1")
3
4    ; Creation d'une seconde fenêtre avec #PB_Window_NoGadgets pour empêcher la création automatique d'une GadgetList
5    If OpenWindow(1, 0, 0, 300, 200, "Fenêtre secondaire", #PB_Window_TitleBar | #PB_Window_WindowCentered | #PB_Window_NoGadgets, WindowID(0))
6       OldGadgetList = UseGadgetList(WindowID(1)) ; Creation d'une GadgetList et sauvegarde l'ancienne GadgetList
7
8       ButtonGadget(10, 10, 10, 150, 25, "Bouton Fenêtre secondaire") ; Ajoute ce bouton dans la nouvelle GadgetList(Fenêtre secondaire)
9
10      UseGadgetList(OldGadgetList)
                    ; Retour à la GadgetList précédente
```

```
11      EndIf
12
13      ButtonGadget(1, 10, 45,
        150, 25, "Bouton 2") ; Ce
        bouton sera sur la fenêtre
        principale
14
15      Repeat
16      Until WaitWindowEvent() =
        #PB_Event_CloseWindow
17    EndIf
```

Voir aussi

OpenWindow() , WindowID()

OS Supportés

Tous

99.82 WebGadget

Syntaxe

```
Resultat = WebGadget(#Gadget,
    X, Y, Largeur, Hauteur,
    URL$)
```

Description

Crée un Navigateur Internet (WebGadget) dans la GadgetList en cours.

Arguments

#Gadget Le numéro d'identification du nouveau gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

URL$ L'URL à charger.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Gadget, alors la valeur de retour est le numéro d'identification du gadget, généré automatiquement.
Cette fonction échoue si les composants requis pour le WebGadget ne peuvent pas être chargés. Voir ci-dessous les exigences pour le WebGadget sur chaque OS.

Remarques

Les composants suivants sont requis pour utiliser le WebGadget sur chaque OS. Ces composants sont nécessaires pour utiliser le WebGadget, pas seulement pour la compilation du programme.

Windows

Sur Microsoft Windows, ce gadget utilise l'object ActiveX Internet Explorer 4.0+.
Pensez à remplacer les "\" par des "/" dans vos codes.

Linux

Le WebGadget utilise la bibliothèque WebKitGtk sur Linux. Le paquet est nommé "libwebkit". Certaines distributions peuvent comprendre une ancienne version de ce paquet qui est nommé «WebKitGtk». Si votre distribution ne contient pas ce paquet alors les sources peuvent être téléchargées sur Internet WebKitGtk home page.

OS X

Le WebGadget utilise le composant WebKit de OS X. Ce composant est fourni avec le système d'exploitation. Il n'y a pas d'autres exigences.

Les fonctions suivantes peuvent être utilisées avec WebGadget :

```
  - SetGadgetText()
: Change l'URL.
  - GetGadgetText()
: Renvoie l'URL actuelle.
  - SetGadgetState()
: Effectue une action sur le
   gadget. 'Etat' peut
   prendre une des valeurs
   suivantes:
    #PB_Web_Back    : Retour
   en arrière dans la
   navigation.
    #PB_Web_Forward: Avance
   d'un pas dans la
   navigation.
    #PB_Web_Stop    : Arrête
   le chargement de la page.
    #PB_Web_Refresh: Recharge
   la page actuelle.


  - SetGadgetItemText()
: avec #PB_Web_HtmlCode comme
   'Element' le code html
   peut être inséré dans le
   Gadget. (Windows seulement)
```

```
  - GetGadgetItemText()
: Les constantes suivantes
   peuvent être utilisées
   pour obtenir une
   information: (Windows
   seulement)
    #PB_Web_HtmlCode      :
   Renvoie le code html du
   gadget.
    #PB_Web_PageTitle     :
   Renvoie le titre de la
   page affichée.
    #PB_Web_StatusMessage :
   Renvoie le message actuel
   de la barre d'état.
    #PB_Web_SelectedText  :
   Renvoie le texte
   sélectionné à l'intérieur
   du gadget.

  - SetGadgetAttribute()
: Définit les attributs
   suivants: (Windows
   seulement)
    #PB_Web_ScrollX
              : Fixe la
   position de la barre de
   défilement horizontale .
    #PB_Web_ScrollY
              : Fixe la
   position de la barre de
   défilement verticale.
    #PB_Web_BlockPopups
          : Bloque les
   fenêtres popup (Valeur=1
   pour bloquer et 0 pour
   débloquer).
   #PB_EventType_PopupWindow
   est renvoyé si cet
   attribut est validé.
    #PB_Web_BlockPopupMenu
       : Bloque le menu popup
   standard.
   #PB_EventType_PopupMenu
   est renvoyé si cet
   attribut est validé.
    #PB_Web_NavigationCallback
   : Définit une callback
   pour contrôler (et
   désactiver) la navigation.
```

La callback de la navigation doit avoir le format suivant : (Windows seulement)

```
1  Procedure
     NavigationCallback(Gadget,
     Url$)
2     ;
3     ; Renvoie #True pour
     autoriser cette
```

```
    ;navigation ou #False
    pour la refuser.
    ;
    ProcedureReturn #True
EndProcedure
```

- GetGadgetAttribute()
: Pour obtenir les attributs suivants :
  #PB_Web_ScrollX
  : Renvoie la position de la barre de défilement horizontale.
  #PB_Web_ScrollY
  : Renvoie la position de la barre de défilement verticale.
  #PB_Web_Busy
  : Renvoie une valeur différente de zéro si le gadget est occupé à charger une page.
  #PB_Web_Progress
  : Renvoie la progression actuelle (parfois estimée) après un évènement #PB_EventType_DownloadProgress.
  #PB_Web_ProgressMax
  : Renvoie la progression actuelle maximum (parfois estimée) après un évènement #PB_EventType_DownloadProgress.
  #PB_Web_BlockPopups
  : Renvoie la valeur actuelle de l'attribut de blocage des fenêtres popup.
  #PB_Web_BlockPopupMenu
  : Renvoie la valeur actuelle de l'attribut de blocage du menu popup.
  #PB_Web_NavigationCallback: Renvoie la valeur actuelle de la callback (si elle existe).

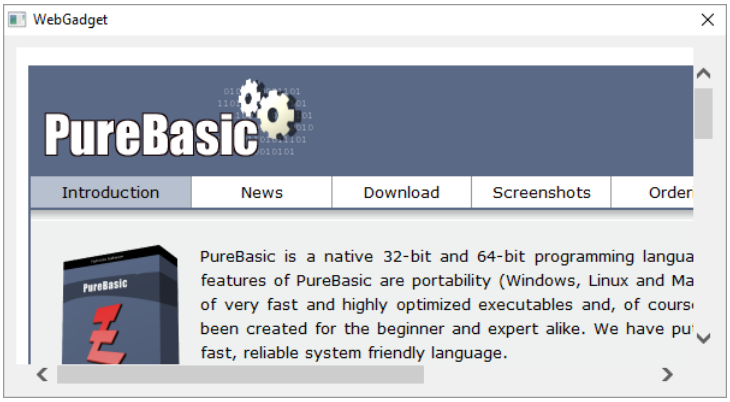
Les évènements suivants sont supportés par EventType() pour ce gadget :

#PB_EventType_TitleChange
: Le titre de la page a changé (Windows seulement).
#PB_EventType_StatusChange
: Le message de la barre d'état a changé (Windows seulement).
#PB_EventType_DownloadStart
: Le téléchargement

```
 d'une page a commencé
 (Windows, OS X).
#PB_EventType_DownloadProgress
 : L'information de
 progression du
 téléchargement est
 disponible avec
 GetGadgetAttribute()
(Windows seulement).
#PB_EventType_DownloadEnd
      : Le téléchargement
 d'une page s'est terminé
 (fini ou annulé) (Windows,
 OS X).
#PB_EventType_PopupWindow
      : Une fenêtre popup
 vient d'être bloquée
 (Windows seulement).
#PB_EventType_PopupMenu
        : Le menu popup
 vient d'être bloqué (utile
 pour afficher un menu
 personnalisé) (Windows
 seulement).
```

Exemple

```
1  If
    OpenWindow(0,0,0,600,300,"WebGadget",#PB_Window_SystemMenu
    |
    #PB_Window_ScreenCentered)
2    WebGadget(0,10,10,580,280,"https://www.purebasic.com")
3    ; Note: si vous désirez
    utiliser un fichier local,
    changez le dernier
    paramètre en "file://" +
    chemin + nomdufichier
4    Repeat
5    Until WaitWindowEvent() =
    #PB_Event_CloseWindow
6  EndIf
```



Example 2 : (Windows seulement)
Utilisation de la procedure
NavigationCallback()

```
; Cet exemple affiche le site Web PureBasic.com. La procédure callback interdit la
; navigation vers les 'News' du site (#False retourné), mais l'autorise
; pour tous les autres sites (#True retourné).

Procedure NavigationCallback(Gadget, Url$)
  If Url$="https://www.purebasic.com/news.php"
    MessageRequester("", "Pas de News aujourd'hui !")
    ProcedureReturn #False
  Else
    ProcedureReturn #True
  EndIf
EndProcedure

If OpenWindow(0, 0, 0, 600, 300, "WebGadget", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  WebGadget(0, 10, 10, 580, 280, "https://www.purebasic.com")
  SetGadgetAttribute(0, #PB_Web_NavigationCallback, @NavigationCallback())
  Repeat
  Until WaitWindowEvent() = #PB_Event_CloseWindow
EndIf
```

Voir aussi

GetGadgetText() , SetGadgetText() , GetGadgetItemText() , SetGadgetItemText() , SetGadgetState() , GetGadgetAttribute() , SetGadgetAttribute()

OS Supportés

Tous

99.83 WebGadgetPath

Syntaxe

```
Resultat =
    WebGadgetPath(FichierBibliotheque$
    [, MozillaPath$])
```

Description

Attention
Cette fonction est dépréciée, elle sera peut-être supprimée dans une future version de PureBasic. Elle ne doit pas être utilisée dans du nouveau code.

Cette fonction n'est plus nécessaire. Elle ne fait rien et renvoie une valeur différente de zéro.

OS Supportés

Tous

99.84 BindGadgetEvent

Syntaxe

```
BindGadgetEvent(#Gadget,
    @Callback() [,
    TypeEvenement])
```

Description

Ajoute un évènement d'un gadget à la liste des évènements de la fenêtre.
Cet évènement est lié à la callback de la fenêtre.

Arguments

#Gadget Le gadget à utiliser.

@Callback() La procédure à appeler lorsque l'évènement se produit.
Elle doit être déclarée comme ceci :

```
1  Procedure EventHandler()
2    ; Code ici...
3  EndProcedure
```

EventGadget() , EventWindow() , EventMenu() , EventType() et EventData() sont disponibles pour obtenir plus d'informations sur l'évènement.
Note : WindowEvent() et WaitWindowEvent() ne devraient jamais être appelées à l'intérieur du Callback(), car ça peut verrouiller un programme ou occasionner un comportement erroné.

TypeEvenement (optionnel) Le type d'évènement à ajouter.
- #PB_All peut être utilisé pour lier tous les types d'évènements.

- Pour une liste complète des types pris en charge, voir EventType() .

Valeur de retour

Aucune.

Remarques

D'ordinaire, les évènements sont gérés avec les fonctions WindowEvent() et WaitWindowEvent() . Cette fonction ajoute un moyen supplémentaire de gestion des évènements avec PureBasic. Avec cependant un petit plus qu'est la possibilité de gérer les évènements en temps réel car la callback peut être invoquée dès que l'évènement se produit (utile pour ScrollBarGadget() , ScrollAreaGadget() etc.).
L'évènement ajouté peut être retiré avec UnbindGadgetEvent() .

Exemple

```
1   Procedure ButtonHandler()
2     Debug "Clic sur le
      ButtonGadget #" +
      EventGadget()
3   EndProcedure
4
5   OpenWindow(0, 100, 100,
      200, 90, "Test Clic",
      #PB_Window_SystemMenu)
6
7   ButtonGadget(0, 10, 10,
      180, 30, "Cliquez moi !")
8   BindGadgetEvent(0,
      @ButtonHandler())
9
10  ButtonGadget(1, 10, 50,
      180, 30, "Cliquez moi !")
11  BindGadgetEvent(1,
      @ButtonHandler())
12
13  Repeat
14    Event = WaitWindowEvent()
15  Until Event =
      #PB_Event_CloseWindow
```

Voir aussi

BindEvent() , BindMenuEvent() , UnbindGadgetEvent() , WindowEvent() , WaitWindowEvent()

OS Supportés

Tous

99.85 UnbindGadgetEvent

Syntaxe

```
UnbindGadgetEvent(#Gadget,
    @Callback() [,
    TypeEvenement])
```

Description

Retire un évènement d'un gadget de la liste des évènements de la fenêtre initialement ajouté avec la fonction BindGadgetEvent() .

Arguments

#Gadget Le gadget à utiliser.

@Callback() La procédure à retirer.

TypeEvenement (optionnel) Le type d'évènement à retirer.
Pour une liste complète des types pris en charge, voir EventType() .

Valeur de retour

Aucune.

Remarques

Si l'évènement correspondant n'est pas trouvé, cette commande n'a aucun effet.

Exemple

```
1   Procedure ButtonHandler()
2     Debug "Clic sur le
      ButtonGadget #" +
      EventGadget()
3   EndProcedure
4
5   OpenWindow(0, 100, 100,
      200, 50, "Test clic",
      #PB_Window_SystemMenu)
6
7   ButtonGadget(0, 10, 10,
      180, 30, "Cliquez moi !")
8
9   BindGadgetEvent(0,
      @ButtonHandler())
10  UnbindGadgetEvent(0,
      @ButtonHandler()) ;
      Suppression immédiate
11
12  Repeat
13    Event = WaitWindowEvent()
14  Until Event =
      #PB_Event_CloseWindow
```

Voir aussi

BindEvent() , BindGadgetEvent() ,
BindMenuEvent() , WindowEvent() ,
WaitWindowEvent()

OS Supportés

Tous

Chapitre 100

Gadget3D

Généralités

La bibliothèque Gadget3D permet de créer des interfaces graphiques complexes (GUI) directement sur la zone d'écran utilisant le moteur 3D. Elle est principalement destinée à la création de jeux ou d'applications qui doivent se lancer en mode plein écran et qui nécessitent une interface utilisateur.
La syntaxe de cette bibliothèque est similaire à la bibliothèque Gadget .
Le moteur utilisé pour l'interface graphique se nomme CEGUI, il permet l'utilisation de skins, il est rapide et contient de nombreux gadgets prédéfinis. Vous trouverez plus d'informations au sujet de CEGUI ici : http ://www.cegui.org.uk. ou là :.
InitEngine3D() doit être appelé avec succès avant de pouvoir utiliser les commandes relatives aux gadgets 3D.
Pour utiliser les gadgets, une fenêtre 3D doit être ouverte au préalable.

OS Supportés

Tous

100.1 AddGadgetItem3D

Syntaxe

```
AddGadgetItem3D(#Gadget3D,
    Position, Texte$)
```

Description

Ajoute un élément à un gadget 3D

Arguments

#Gadget3D Le gadget 3D à utiliser.

Position Le numéro d'emplacement où sera inséré le nouvel élément.

0 pour ajouter l'élément au début.
-1 pour ajouter l'élément à la fin.
Souvenez vous que lors de l'ajout d'un élément, tous les éléments qui suivent verront leur position s'incrémenter de 1.

Texte$ Le texte du nouvel élément.

Valeur de retour

Aucune.

Remarques

Les gadgets supportant cette commande sont :
- ComboBoxGadget3D()
- ListViewGadget3D()
- PanelGadget3D()

Voir aussi

ComboBoxGadget3D() , ListViewGadget3D() , PanelGadget3D()

OS Supportés

Tous

100.2 ButtonGadget3D

Syntaxe

```
Resultat =
    ButtonGadget3D(#Gadget3D,
    X, Y, Largeur, Hauteur,
    Texte$)
```

Description

Crée un bouton 3D.

Arguments

#Gadget3D Le numéro d'identification du gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Texte$ Le texte du gadget.

Valeur de retour

Renvoie une valeur non nulle si le gadget a été créé avec succès, zéro sinon. Si #PB_Any a été spécifié comme paramètre #Gadget3D alors le numéro généré automatiquement est renvoyé dans 'Resultat' en cas de succès.

Remarques

Vous pouvez ajouter une 'mini aide' à ce gadget en utilisant GadgetToolTip3D() . Les commandes suivantes peuvent être utilisées pour agir sur le gadget :
- SetGadgetText3D() : Change le texte affiché par le gadget.
- GetGadgetText3D() : Renvoie le texte affiché par le gadget.

Voir aussi

GadgetToolTip3D() , SetGadgetText3D() , GetGadgetText3D()

OS Supportés

Tous

100.3 CheckBoxGadget3D

Syntaxe

```
Resultat =
    CheckBoxGadget3D(#Gadget3D,
    X, Y, Largeur, Hauteur,
    Texte$)
```

Description

Crée un gadget case à cocher 3D.

Arguments

#Gadget3D Le numéro d'identification du gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Texte$ Le texte du gadget.

Valeur de retour

Renvoie une valeur non nulle si le gadget a été créé avec succès, zéro sinon. Si #PB_Any a été spécifié comme paramètre #Gadget3D alors le numéro généré automatiquement est renvoyé dans 'Resultat' en cas de succès.

Remarques

Vous pouvez ajouter une 'mini aide' à ce gadget en utilisant GadgetToolTip3D() . Les commandes suivantes peuvent être utilisées pour agir sur le gadget :

- SetGadgetText3D() : Change le texte affiché par le gadget.
- GetGadgetText3D() : Renvoie le texte affiché par le gadget.

Voir aussi

GadgetToolTip3D() , GetGadgetState3D() , SetGadgetState3D()

OS Supportés

Tous

100.4 ClearGadgetItems3D

Syntaxe

```
ClearGadgetItems3D(#Gadget3D)
```

Description

Supprime les éléments d'un gadget

Arguments

#**Gadget3D** Le gadget 3D à utiliser.

Valeur de retour

Aucune.

Remarques

Le gadget doit être de l'un des types suivants :
- ComboBoxGadget3D()
- ListViewGadget3D()
- PanelGadget3D()

Voir aussi

AddGadgetItem3D()

OS Supportés

Tous

100.5 CloseGadgetList3D

Syntaxe

```
CloseGadgetList3D()
```

Description

Ferme la liste courante des gadgets et revient à la précédente.

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

C'est particulièrement utile pour les gadgets suivants :
- ContainerGadget3D()
- PanelGadget3D()
- ScrollAreaGadget3D()

Voir aussi

OpenGadgetList3D()

OS Supportés

Tous

100.6 ComboBoxGadget3D

Syntaxe

```
Resultat =
    ComboBoxGadget3D(#Gadget3D,
    X, Y, Largeur, Hauteur [,
    Options])
```

Description

Crée un gadget Liste déroulante 3D.

Arguments

#Gadget3D Le numéro d'identification du gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Options (optionnel) Peut être une combinaison des constantes suivantes :

```
#PB_ComboBox3D_Editable :
Rend la liste déroulante
éditable
```

Valeur de retour

Renvoie une valeur non nulle si le gadget a été créé avec succès, zéro sinon. Si #PB_Any a été spécifié comme paramètre #Gadget3D alors le numéro généré automatiquement est renvoyé dans 'Resultat' en cas de succès.

Remarques

Vous pouvez ajouter une 'mini aide' à ce gadget en utilisant GadgetToolTip3D() . Les commandes suivantes peuvent être utilisées pour agir sur le gadget :
- AddGadgetItem3D() : Ajoute un élément.
- GetGadgetItemText3D() : Renvoie le texte de l'élément spécifié.
- CountGadgetItems3D() : Renvoie le nombre d'éléments que contient le ComboBox.
- ClearGadgetItems3D() : Supprime tous les éléments.
- RemoveGadgetItem3D() : Supprime un élément.
- SetGadgetItemText3D() : Change le texte de l'élément spécifié.
- SetGadgetState3D() : Change l'élément sélectionné.
- GetGadgetState3D() : Récupère l'index de l'élément sélectionné.
- SetGadgetText3D() : Change le texte affiché. Si le ComboBoxGadget n'est pas éditable, le texte doit être dans la liste déroulante.
- GetGadgetText3D() : Renvoie le contenu texte de la zone visible de la ComboBox.

OS Supportés

Tous

100.7 ContainerGadget3D

Syntaxe

```
Resultat =
    ContainerGadget3D(#Gadget3D,
    X, Y, Largeur, Hauteur)
```

Description

Crée un gadget container 3D.

Arguments

#Gadget3D Le numéro d'identification du gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Valeur de retour

Renvoie une valeur non nulle si le gadget a été créé avec succès, zéro sinon. Si

#PB_Any a été spécifié comme paramètre #Gadget3D alors le numéro généré automatiquement est renvoyé dans 'Resultat' en cas de succès.

Remarques

Vous pouvez ajouter une 'mini aide' à ce gadget en utilisant GadgetToolTip3D() . Il s'agit simplement d'un gadget qui peut contenir d'autres gadgets. La GadgetList courante est automatiquement changée et les nouveaux gadgets créés après cette commande seront placés dans ce gadget 3D. CloseGadgetList3D() permet de revenir à la GadgetList précédente. OpenGadgetList3D() pourra être utilisé pour ajouter des gadgets dynamiquement.

OS Supportés

Tous

100.8 CountGadgetItems3D

Syntaxe

```
Resultat =
    CountGadgetItems3D(#Gadget3D)
```

Description

Renvoie le nombre d'éléments que contient un gadget 3D.

Arguments

#Gadget3D Le gadget 3D à utiliser.

Valeur de retour

le nombre d'éléments que contient le gadget 3D spécifié.

Remarques

Cette fonction est disponible avec les gadgets suivants :
- ComboBoxGadget3D()
- ListViewGadget3D()
- PanelGadget3D()

OS Supportés

Tous

100.9 DisableGadget3D

Syntaxe

```
DisableGadget3D(#Gadget3D,
    Etat)
```

Description

Active ou désactive un gadget 3D.

Arguments

#Gadget3D Le gadget 3D à utiliser.

Etat 0 : Activé
1 : Désactivé

Valeur de retour

Aucune.

OS Supportés

Tous

100.10 EditorGadget3D

Syntaxe

```
Resultat =
    EditorGadget3D(#Gadget3D,
    X, Y, Largeur, Hauteur [,
    Options])
```

Description

Crée un gadget 3D de type editeur.

Arguments

#Gadget3D Le numéro d'identification du gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Options (optionnel) Peut prendre la valeur suivante :

```
#PB_Editor3D_ReadOnly:
 L'utilisateur ne peut
 pas éditer le texte.
```

Valeur de retour

Renvoie une valeur non nulle si le gadget a été créé avec succès, zéro sinon. Si #PB_Any a été spécifié comme paramètre #Gadget3D alors le numéro généré automatiquement est renvoyé dans 'Resultat' en cas de succès.

Remarques

Vous pouvez ajouter une 'mini aide' à ce gadget en utilisant GadgetToolTip3D() . Les commandes suivantes peuvent être utilisées pour agir sur le contenu du gadget :
- GetGadgetText3D() : Renvoie tout le texte contenu dans le gadget.
- SetGadgetText3D() : Remplace tout le texte contenu par le gadget par un autre.
- SetGadgetAttribute3D() avec l'attribut suivant :

```
#PB_Editor3D_ReadOnly: 0 =
 éditable, 1 = non éditable.
```

- GetGadgetAttribute() avec l'attribut suivant :

```
#PB_Editor3D_ReadOnly: 0 =
 éditable, 1 = non éditable.
```

OS Supportés

Tous

100.11 FrameGadget3D

Syntaxe

```
Resultat =
   FrameGadget3D(#Gadget3D,
   X, Y, Largeur, Hauteur,
   Texte$)
```

Description

Crée un gadget de type cadre 3D.

Arguments

#Gadget3D Le numéro d'identification du gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Texte$ Le texte du gadget.

Valeur de retour

Renvoie une valeur non nulle si le gadget a été créé avec succès, zéro sinon. Si #PB_Any a été spécifié comme paramètre #Gadget3D alors le numéro généré automatiquement est renvoyé dans 'Resultat' en cas de succès.

Remarques

Comme ce gadget est seulement décoratif, GadgetToolTip3D() ne peut pas être utilisé. Ce gadget ne reçoit aucun évènement.

OS Supportés

Tous

100.12 FreeGadget3D

Syntaxe

```
FreeGadget3D(#Gadget3D)
```

Description

Supprime un gadget 3D.

Arguments

#Gadget3D Le gadget 3D à utiliser.

Valeur de retour

Aucune.

Remarques

Tous les gadgets restants sont automatiquement supprimés quand le programme se termine.

OS Supportés

Tous

100.13 GadgetID3D

Syntaxe

```
Resultat =
    GadgetID3D(#Gadget3D)
```

Description

Renvoie l'identifiant d'un gadget 3D.

Arguments

#**Gadget3D** Le gadget 3D à utiliser.

Valeur de retour

Renvoie l'identifiant système unique du gadget 3D.

OS Supportés

Tous

100.14 GadgetToolTip3D

Syntaxe

```
GadgetToolTip3D(#Gadget3D,
    Texte$)
```

Description

Associe un texte flottant au gadget 3D.

Arguments

#**Gadget3D** Le gadget 3D à utiliser.
Texte$ Le texte flottant à afficher.

Valeur de retour

Aucune.

Remarques

Un 'Tooltip' est un texte flottant qui apparaît au bout d'un certain temps lorsque le curseur de la souris reste immobile au dessus d'un gadget.
Les gadgets suivants sont supportés :
- ButtonGadget3D()
- CheckBoxGadget3D()
- ComboBoxGadget3D()
- ContainerGadget3D()
- EditorGadget3D()
- ImageGadget3D()
- ListViewGadget3D()
- OptionGadget3D()
- PanelGadget3D()
- ProgressBarGadget3D()
- ScrollBarGadget3D()
- SpinGadget3D()
- StringGadget3D()

OS Supportés

Tous

100.15 GadgetX3D

Syntaxe

```
Resultat =
    GadgetX3D(#Gadget3D)
```

Description

Renvoie la position en X d'un gadget 3D.

Arguments

#**Gadget3D** Le gadget 3D à utiliser.

Valeur de retour

La position en X, en pixels, du gadget 3D. Par rapport au bord gauche de l'écran ou du container.

Voir aussi

GadgetY3D()

OS Supportés

Tous

100.16 GadgetY3D

Syntaxe

```
Resultat =
    GadgetY3D(#Gadget3D)
```

Description

Renvoie la position en Y d'un gadget 3D.

Arguments

#**Gadget3D** Le gadget 3D à utiliser.

Valeur de retour

La position en Y, en pixels, du gadget 3D. Par rapport au bord haut de l'écran ou du container.

Voir aussi

GadgetX3D()

OS Supportés

Tous

100.17 GadgetHeight3D

Syntaxe

```
Resultat =
    GadgetHeight3D(#Gadget3D)
```

Description

Renvoie la hauteur d'un gadget 3D.

Arguments

#Gadget3D Le gadget 3D à utiliser.

Valeur de retour

La hauteur, en pixels, du gadget 3D.

Voir aussi

GadgetWidth3D()

OS Supportés

Tous

100.18 GadgetType3D

Syntaxe

```
Resultat =
    GadgetType3D(#Gadget3D)
```

Description

Renvoie le type d'un gadget 3D.

Arguments

#Gadget3D Le gadget 3D à utiliser.

Valeur de retour

Peut prendre l'une des valeurs suivantes :

```
#PB_GadgetType3D_Button
          : ButtonGadget3D()

#PB_GadgetType3D_CheckBox
        : CheckBoxGadget3D()

#PB_GadgetType3D_ComboBox
        : ComboBoxGadget3D()

#PB_GadgetType3D_Container
       : ContainerGadget3D()
```

```
#PB_GadgetType3D_Editor
        : EditorGadget3D()

#PB_GadgetType3D_Frame
         : FrameGadget3D()

#PB_GadgetType3D_Image
         : ImageGadget3D()

#PB_GadgetType3D_ListView
       : ListViewGadget3D()

#PB_GadgetType3D_Option
        : OptionGadget3D()

#PB_GadgetType3D_Panel
         : PanelGadget3D()

#PB_GadgetType3D_ProgressBar
    : ProgressBarGadget3D()

#PB_GadgetType3D_ScrollArea
     : ScrollAreaGadget3D()

#PB_GadgetType3D_ScrollBar
      : ScrollBarGadget3D()

#PB_GadgetType3D_Spin
          : SpinGadget3D()

#PB_GadgetType3D_String
         : StringGadget3D()

#PB_GadgetType3D_Text
          : TextGadget3D()

#PB_GadgetType3D_Unknown
        : Type inconnu,
  probablement pas un gadget
  PureBasic.
```

OS Supportés

Tous

100.19 GadgetWidth3D

Syntaxe

```
Resultat =
   GadgetWidth3D(#Gadget3D)
```

Description

Renvoie la largeur d'un gadget 3D.

Arguments

#**Gadget3D** Le gadget 3D à utiliser.

Valeur de retour

La largeur, en pixels, du gadget 3D.

Voir aussi

GadgetHeight3D()

OS Supportés

Tous

100.20 GetActiveGadget3D

Syntaxe

```
Resultat = GetActiveGadget3D()
```

Description

Renvoie le numéro du gadget 3D qui a le focus clavier.

Arguments

Aucun.

Valeur de retour

Le numéro du gadget 3D qui a actuellement le 'focus' clavier. Si aucun gadget a le focus, -1 est renvoyé.

Voir aussi

SetActiveGadget3D()

OS Supportés

Tous

100.21 GetGadgetAttribute3D

Syntaxe

```
Resultat =
    GetGadgetAttribute3D(#Gadget3D,
    Attribut)
```

Description

Renvoie la valeur de l'attribut d'un gadget 3D.

Arguments

#Gadget3D Le gadget 3D à utiliser.

Attribut L'attribut à tester.

Valeur de retour

Cette fonction est disponible avec les gadgets suivants :
- EditorGadget3D()
- PanelGadget3D()
- ProgressBarGadget3D()
- ScrollAreaGadget3D()
- ScrollBarGadget3D()
- SpinGadget3D()

OS Supportés

Tous

100.22 GetGadgetData3D

Syntaxe

```
Resultat =
    GetGadgetData3D(#Gadget3D)
```

Description

Renvoie la valeur personnalisée et stockée dans un gadget 3D.

Arguments

#**Gadget3D** Le gadget 3D à utiliser.

Valeur de retour

La valeur précédemment associée au gadget 3D avec la commande SetGadgetData3D() , zéro sinon.

Remarques

Cette commande fonctionne avec tous les types de gadget 3D de PureBasic.
Pour un exemple d'utilisation, consulter la définition de la commande SetGadgetData3D() .

Voir aussi

SetGadgetData3D()

OS Supportés

Tous

100.23 GetGadgetItemData3D

Syntaxe

```
Resultat =
    GetGadgetItemData3D(#Gadget3D,
    Element)
```

Description

Renvoie la valeur qui a été stockée dans un élément d'un gadget 3D.

Arguments

#Gadget3D Le gadget 3D à utiliser.

Element L'élément à tester.

Valeur de retour

La valeur précédemment associée au gadget 3D avec la commande SetGadgetItemData3D() , zéro sinon.

Remarques

Cette commande est supportée par les gadgets suivants :
- ComboBoxGadget3D()
- ListViewGadget3D()

Pour un exemple d'utilisation, consulter la définition de la commande SetGadgetItemData3D() .

Voir aussi

SetGadgetItemData3D()

OS Supportés

Tous

100.24 GetGadgetState3D

Syntaxe

```
Resultat =
    GetGadgetState3D(#Gadget3D)
```

Description

Renvoie l'état d'un gadget 3D.

Arguments

#Gadget3D Le gadget 3D à utiliser.

Valeur de retour

L'état courant du gadget 3D.

Remarques

Fonctionne avec les gadgets 3D suivants :
- CheckBoxGadget3D() : renvoie 1 s'il est activé, 0 sinon.
- ComboBoxGadget3D() : renvoie le numéro de l'élément sélectionné ou -1 si pas de sélection.
- ImageGadget3D() : renvoie le TextureID de l'image actuellement affichée.
- ListViewGadget3D() : renvoie le numéro de l'élément sélectionné ou -1 s'il n'y a pas d'élément sélectionné.
- OptionGadget3D() : renvoie 1 s'il est activé, 0 sinon.
- PanelGadget3D() : renvoie le numéro de l'onglet sélectionné ou -1 si pas de sélection.
- ProgressBarGadget3D() : renvoie la valeur actuelle de la barre de progression.
- ScrollBarGadget3D() : renvoie la position actuelle de l'ascenseur.
- SpinGadget3D() : renvoie la valeur actuelle du SpinGadget.

Voir aussi

SetGadgetState3D()

OS Supportés

Tous

100.25 GetGadgetItemText3D

Syntaxe

```
Resultat\$ =
    GetGadgetItemText3D(#Gadget3D,
    Element [, Colonne])
```

Description

Renvoie le texte d'un élément d'un gadget 3D.

Arguments

#Gadget3D Le gadget 3D à utiliser.

Element L'élément à tester.
La numérotation des éléments commence à partir de zéro.

Colonne (optionnel) Pas utilisé pour l'instant.

Valeur de retour

Le texte de l'élément du gadget 3D.

Remarques

Cette fonction est disponible avec les gadgets suivants :
- ComboBoxGadget3D() - renvoie le texte de l'élément de la liste déroulante ('Colonne' est ignorée).
- ListViewGadget3D() - renvoie le texte de l'élément de la liste ('Colonne' est ignorée).
- PanelGadget3D() - renvoie le texte de l'élément spécifié ('Colonne' est ignorée).

Voir aussi

SetGadgetItemText3D()

OS Supportés

Tous

100.26 GetGadgetItemState3D

Syntaxe

```
Resultat =
    GetGadgetItemState3D(#Gadget3D,
    Element)
```

Description

Renvoie l'état d'un élément d'un gadget 3D.

Arguments

#Gadget3D Le gadget 3D à utiliser.

Element L'élément à tester.
La numérotation des éléments commence à partir de zéro.

Valeur de retour

L'état de l'élément du gadget 3D.

Remarques

Cette fonction est disponible avec les gadgets suivants :
- ListViewGadget3D() : Renvoie 1 si l'élément est sélectionné, 0 sinon.

Voir aussi

SetGadgetItemState3D()

OS Supportés

Tous

100.27 GetGadgetText3D

Syntaxe

```
Resultat\$ =
    GetGadgetText3D(#Gadget3D)
```

Description

Renvoie le texte contenu dans un gadget 3D.

Arguments

#**Gadget3D** Le gadget 3D à utiliser.

Valeur de retour

Le texte du gadget 3D.

Remarques

Cette fonction est disponible avec les gadgets suivants :
- ButtonGadget3D() - renvoie le texte affiché par le BoutonGadget3D.
- ComboBoxGadget3D() - renvoie le texte de l'élément sélectionné.
- EditorGadget3D() - Renvoie le texte contenu dans l'éditeur. Les différentes lignes seront séparées par la combinaison "Chr(13)+Chr(10)".
- ListViewGadget3D() - renvoie le texte de l'élément sélectionné.
- StringGadget3D() - renvoie le contenu du StringGadget3D.
- TextGadget3D() - renvoie le contenu du TextGadget3D.

Voir aussi

SetGadgetText3D()

OS Supportés

Tous

100.28 HideGadget3D

Syntaxe

```
HideGadget3D(#Gadget3D, Etat)
```

Description

Cache ou affiche un gadget 3D.

Arguments

#**Gadget3D** Le gadget 3D à utiliser.

Etat 0 : Le gadget est affiché
1 : Le gadget est caché

Valeur de retour

Aucune.

Voir aussi

DisableGadget3D()

OS Supportés

Tous

100.29 ImageGadget3D

Syntaxe

```
Resultat =
    ImageGadget3D(#Gadget3D,
    X, Y, Largeur, Hauteur,
    TextureID [, Options])
```

Description

Crée un gadget 3D Image.

Arguments

#**Gadget3D** Le numéro d'identification du gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

TextureID La texture à aficher.
TextureID() permet d'obtenir une valeur valide.
Zéro permet de supprimer la texture.

Options (optionnel) Peut prendre la valeur suivante :

```
#PB_Image3D_Border :
 Affiche un cadre autour
 de l'image.
```

Valeur de retour

Renvoie une valeur non nulle si le gadget a été créé avec succès, zéro sinon. Si #PB_Any a été spécifié comme paramètre #Gadget3D alors le numéro généré automatiquement est renvoyé dans 'Resultat' en cas de succès.

Remarques

Vous pouvez ajouter une 'mini aide' à ce gadget en utilisant GadgetToolTip3D() . Les commandes suivantes peuvent être utilisées pour agir sur le contenu du gadget : - SetGadgetState3D() : Permet de changer dynamiquement l'image contenue dans le gadget.

OS Supportés

Tous

100.30 IsGadget3D

Syntaxe

```
Resultat =
    IsGadget3D(#Gadget3D)
```

Description

Teste si un gadget 3D est correctement initialisé.

Arguments

#Gadget3D Le gadget 3D à utiliser.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage.

OS Supportés

Tous

100.31 ListViewGadget3D

Syntaxe

```
Resultat =
    ListViewGadget3D(#Gadget3D,
    X, Y, Largeur, Hauteur)
```

Description

Crée une boîte à liste 3D.

Arguments

#**Gadget3D** Le numéro d'identification du gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Valeur de retour

Renvoie une valeur non nulle si le gadget a été créé avec succès, zéro sinon. Si #PB_Any a été spécifié comme paramètre #Gadget3D alors le numéro généré automatiquement est renvoyé dans 'Resultat' en cas de succès.

Remarques

Vous pouvez ajouter une 'mini aide' à ce gadget en utilisant GadgetToolTip3D() .
Les fonctions suivantes peuvent être utilisées pour agir sur le contenu de la liste :
- AddGadgetItem3D() : Ajoute un élément
- RemoveGadgetItem3D() : Supprime un élément
- ClearGadgetItems3D() : Supprime tous les éléments
- CountGadgetItems3D() : Renvoie le nombre d'éléments.
- GetGadgetItemData3D() : Renvoie la valeur personnalisée associée à cet élément.
- GetGadgetItemState3D() : Renvoie 0 si l'élément n'est pas séléctionné, sinon une valeur non-nulle.
- GetGadgetItemText3D() : Renvoie le texte de l'élément spécifié.
- GetGadgetState3D() : Renvoie le numéro de l'élément qui est sélectionné, -1 s'il n'y a pas de sélection.
- GetGadgetText3D() : Renvoie le texte de l'élément sélectionné.
- SetGadgetItemData3D() : Associe une valeur personnalisée à cet élément.
- SetGadgetItemState3D() : Sélectionne ou désélectionne l'élément spécifié.
- SetGadgetItemText3D() : Change le texte de l'élément spécifié.
- SetGadgetState3D() : Change l'état de l'élément spécifié.
- SetGadgetText3D() : Sélectionne l'élément correspondant au texte indiqué. Le texte doit exactement correspondre.

OS Supportés

Tous

100.32 OpenGadgetList3D

Syntaxe

```
OpenGadgetList3D(#Gadget3D [,
    Element])
```

Description

Change la GadgetList courante.

Arguments

#Gadget3D Le gadget 3D à utiliser.
Element (optionnel) L'élément à utiliser.

Valeur de retour

Aucune.

Remarques

En principe, c'est un gadget qui fait office de conteneur. Cela permet de rajouter des nouveaux gadgets à la volée sans avoir à recréer le gadget.
- ContainerGadget3D()
- PanelGadget3D() : Le paramètre 'Element' doit être spécifié (correspond à l'onglet)
- ScrollAreaGadget3D()
Une fois que les nouveaux gadgets ont été ajoutés, il convient d'appeler CloseGadgetList3D() pour revenir à l'état précédent.

OS Supportés

Tous

100.33 OptionGadget3D

Syntaxe

```
Resultat =
    OptionGadget3D(#Gadget3D,
    X, Y, Largeur, Hauteur,
    Texte$)
```

Description

Crée un gadget 3D option (boutons radio).

Arguments

#Gadget3D Le numéro d'identification du gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Texte$ Le texte du gadget.
Décrit l'utilité de la case à option, il est placé à sa droite.

Valeur de retour

Renvoie une valeur non nulle si le gadget a été créé avec succès, zéro sinon. Si #PB_Any a été spécifié comme paramètre #Gadget3D alors le numéro généré automatiquement est renvoyé dans 'Resultat' en cas de succès.

Remarques

Ce gadget permet de regrouper plusieurs cases à options, sachant qu'une seule d'entre elles ne peut être sélectionnée à la fois. Au premier appel de cette fonction, un groupe de cases à options est créé, et tous les appels suivants à la fonction OptionGadget3D ajouteront une nouvelle case à options au groupe. Pour terminer le groupe, il suffit d'appeler un autre type de gadget.
Vous pouvez ajouter une 'mini aide' à ce gadget en utilisant GadgetToolTip3D() .

OS Supportés

Tous

100.34 PanelGadget3D

Syntaxe

```
Resultat =
    PanelGadget3D(#Gadget3D,
    X, Y, Largeur, Hauteur)
```

Description

Crée un gadget boîte à onglets 3D.

Arguments

#Gadget3D Le numéro d'identification du gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Valeur de retour

Renvoie une valeur non nulle si le gadget a été créé avec succès, zéro sinon. Si #PB_Any a été spécifié comme paramètre #Gadget3D alors le numéro généré automatiquement est renvoyé dans 'Resultat' en cas de succès.

Remarques

Lorsqu'une boîte à onglets est créée, sa liste d'éléments est vide.
Vous pouvez ajouter une 'mini aide' à ce gadget en utilisant GadgetToolTip3D() .
Les fonctions suivantes peuvent être appelées pour agir sur la liste :
- AddGadgetItem3D() : Ajoute un élément
- RemoveGadgetItem3D() : Supprime un élément
- CountGadgetItems3D() : Renvoie le nombre d'éléments.
- ClearGadgetItems3D() : Supprime tous les éléments
- GetGadgetItemText3D() : Renvoie le texte de l'élément spécifié.
- SetGadgetItemText3D() : Change le texte de l'élément spécifié.
- SetGadgetState3D() : Change l'onglet affiché.
- GetGadgetState3D() : Renvoie le numéro de l'onglet actuellement affiché.
- GetGadgetAttribute3D() avec un des attributs suivants (il doit y avoir au moins un élément dans le PanelGadget3D()) :

```
#PB_Panel3D_ItemWidth :
 Renvoie la largeur de la
 zone utilisable d'un
 onglet.
#PB_Panel3D_ItemHeight:
 Renvoie la hauteur de la
 zone utilisable d'un
 onglet.
#PB_Panel3D_TabHeight :
 Renvoie la hauteur d'un
 bouton de changement
 d'onglet.
```

Avant de pouvoir ajouter des gadgets, il est nécessaire d'appeler la commande AddGadgetItem3D() pour ajouter au moins un onglet. Les prochains gadgets créés le seront automatiquement sur le dernier onglet. Lorsque tous les gadgets de la boîte à onglets ont été placés, CloseGadgetList3D() doit être appelé pour revenir à la GadgetList précédente. Il est ainsi parfaitement possible de créer une boîte à onglets dans une autre boîte à onglets...

OS Supportés

Tous

100.35 ProgressBarGadget3D

Syntaxe

```
Resultat =
    ProgressBarGadget3D(#Gadget3D,
    X, Y, Largeur, Hauteur,
    Minimum, Maximum)
```

Description

Crée un gadget Barre de progression 3D.

Arguments

#Gadget3D Le numéro d'identification du gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Minimum, Maximum Les valeurs minimales et maximales possibles.

Valeur de retour

Renvoie une valeur non nulle si le gadget a été créé avec succès, zéro sinon. Si #PB_Any a été spécifié comme paramètre #Gadget3D alors le numéro généré automatiquement est renvoyé dans 'Resultat' en cas de succès.

Remarques

Vous pouvez ajouter une 'mini aide' à ce gadget en utilisant GadgetToolTip3D() .
Les fonctions suivantes peuvent être appelées pour agir sur la barre de progression :
- SetGadgetState3D() : Change la valeur de la barre de progression.
- GetGadgetState3D() : Renvoie la valeur courante de la barre de progression.
- SetGadgetAttribute3D() avec les attributs suivants :

```
#PB_ProgressBar3D_Minimum:
  Change la valeur minimale.
#PB_ProgressBar3D_Maximum:
  Change la valeur maximale.
```

- GetGadgetAttribute3D() avec les attributs suivants :

```
#PB_ProgressBar3D_Minimum:
 Renvoie la valeur minimale.
#PB_ProgressBar3D_Maximum:
 Renvoie la valeur maximale.
```

OS Supportés

Tous

100.36 RemoveGadgetItem3D

Syntaxe

```
RemoveGadgetItem3D(#Gadget3D,
    Position)
```

Description

Supprime un élément d'un gadget.

Arguments

#Gadget3D Le gadget 3D à utiliser.

Position La position de l'élément dans la liste.
L'index commence à zéro.

Valeur de retour

Aucune.

Remarques

Cette fonction s'applique aux gadgets suivants :
- ComboBoxGadget3D()
- PanelGadget3D()
- ListViewGadget3D()

OS Supportés

Tous

100.37 ResizeGadget3D

Syntaxe

```
ResizeGadget3D(#Gadget3D, X,
    Y, Largeur, Hauteur)
```

Description

Change la taille et la position d'un gadget 3D.

Arguments

#Gadget3D Le gadget 3D à utiliser.

X, Y, Largeur, Hauteur Les nouvelles a position et dimensions du gadget.
Pour faciliter le redimensionnement de l'interface en temps réel, la constante #PB_Ignore peut être spécifiée à chacun des paramètres (X, Y, Largeur ou Hauteur) et la valeur précédente de ce paramètre sera conservée.

Valeur de retour

Aucune.

Exemple

```
[...]

ResizeGadget3D(0, #PB_Ignore, #PB_Ignore, 300, #PB_Ignore) ; Change seulement la largeur du gadget.
```

OS Supportés

Tous

100.38 ScrollBarGadget3D

Syntaxe

```
Resultat = ScrollBarGadget3D(#Gadget3D, X, Y, Largeur, Hauteur, Minimum, Maximum, LongueurPage [, Options])
```

Description

Crée une barre de défilement (ascenseur vertical ou horizontal).

Arguments

#Gadget3D Le numéro d'identification du gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Minimum, Maximum Les valeurs minimales et maximales possibles.
Il est préférable de se limiter à un minimum de 0 et un maximum de 10 000

pour garder une compatibilité entre les systèmes d'exploitation.

LongueurPage La proportion de la page affichée.
Par exemple, avec un ascenseur horizontal et une image d'une largeur de 100 pixels et un paramètre "LongueurPage" de 25, seulement 25 pixels de large seront affichés. Le minimum affiché sera de 0 et le maximum sera de 100.

Options (optionnel) Peut être une combination de :

```
#PB_ScrollBar3D_Vertical:
 La barre de défilement
 est verticale
 (horizontale par défaut).
```

Valeur de retour

Renvoie une valeur non nulle si le gadget a été créé avec succès, zéro sinon. Si #PB_Any a été spécifié comme paramètre #Gadget3D alors le numéro généré automatiquement est renvoyé dans 'Resultat' en cas de succès.

Remarques

Vous pouvez ajouter une 'mini aide' à ce gadget en utilisant GadgetToolTip3D() .
Les fonctions suivantes peuvent être appelées pour agir sur la barre de déplacement :
- GetGadgetState3D() : Renvoie la position actuelle du curseur (valeur comprise dans l'intervalle Minimum-Maximum)
- SetGadgetState3D() : Change la position actuelle du curseur.
- GetGadgetAttribute3D() avec un des attributs suivants :

```
#PB_ScrollBar3D_Minimum   :
 Renvoie la position
 minimale de l'ascenseur.
#PB_ScrollBar3D_Maximum   :
 Renvoie la position
 maximale de l'ascenseur.
#PB_ScrollBar3D_PageLength:
 Renvoie la longueur de la
 page.
```

- SetGadgetAttribute3D() : avec un des attributs suivants :

```
#PB_ScrollBar3D_Minimum   :
 Change la position
 minimale de l'ascenseur.
#PB_ScrollBar3D_Maximum   :
 Change la position
 maximale de l'ascenseur.
```

```
#PB_ScrollBar3D_PageLength:
 Change la longueur de la
 page.
```

OS Supportés

Tous

100.39 ScrollAreaGadget3D

Syntaxe

```
Resultat =
   ScrollAreaGadget3D(#Gadget3D,
   X, Y, Largeur, Hauteur,
   LargeurZoneInterne,
   HauteurZoneInterne,
   ValeurDeplacement)
```

Description

Crée un gadget zone d'affichage 3D.

Arguments

#Gadget3D Le numéro d'identification du gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Minimum, Maximum Les valeurs minimales et maximales possibles.
Il est préférable de se limiter à un minimum de 0 et un maximum de 10 000 pour garder une compatibilité entre les systèmes d'exploitation.

LargeurZoneInterne, HauteurZoneInterne Dimensions de la zone interne, affichable.
Si elle est inférieure aux dimensions du gadget, les ascenseurs ne s'afficheront pas.

ValeurDeplacement Le déplacement par défaut de l'ascenseur, en pixels.

Valeur de retour

Renvoie une valeur non nulle si le gadget a été créé avec succès, zéro sinon. Si #PB_Any a été spécifié comme paramètre #Gadget3D alors le numéro généré automatiquement est renvoyé dans 'Resultat' en cas de succès.

Remarques

Il est très utile pour afficher un gadget trop grand pour être afficher en entier, comme une image par exemple. Cette zone peut contenir n'importe quel type de gadgets. Quand ce gadget est créé la 'GadgetList' courante est automatiquement changée et les nouveaux gadgets qui sont créés après cette commande seront placés dans ce gadget. CloseGadgetList3D() permet de revenir à la 'GadgetList' précédente. OpenGadgetList3D() pourra être utiliser pour ajouter des gadgets dynamiquement dans ce gadget.
Les commandes suivantes peuvent être utilisées :
GetGadgetAttribute3D() : Avec une des constantes suivantes :

```
#PB_ScrollArea3D_InnerWidth
 : Renvoie la largeur (en
 pixels) de la zone interne
 du gadget.
#PB_ScrollArea3D_InnerHeight:
 Renvoie la hauteur (en
 pixels) de la zone interne
 du gadget.
#PB_ScrollArea3D_X
 : Renvoie la position
 horizontale actuelle de
 l'ascenseur (en pixels).
#PB_ScrollArea3D_Y
 : Renvoie la position
 verticale actuelle de
 l'ascenseur (en pixels).
```

SetGadgetAttribute3D() : Avec une des constantes suivantes :

```
#PB_ScrollArea3D_InnerWidth
 : Modifie la largeur (en
 pixels) de la zone interne
 du gadget.
#PB_ScrollArea3D_InnerHeight:
 Modifie la hauteur (en
 pixels) de la zone interne
 du gadget.
#PB_ScrollArea3D_X
 : Modifie la position
 horizontale actuelle de
 l'ascenseur (en pixels).
#PB_ScrollArea3D_Y
 : Modifie la position
 verticale actuelle de
 l'ascenseur (en pixels).
```

OS Supportés

Tous

100.40 SetActiveGadget3D

Syntaxe

```
SetActiveGadget3D(#Gadget3D)
```

Description

Active un gadget 3D (donne le focus).

Arguments

#Gadget3D Le gadget 3D à utiliser.

Valeur de retour

Aucune.

Remarques

Cette fonction est utilisée principalement avec ComboBoxGadget3D() and StringGadget3D() . Cela permet a un gadget d'obtenir le focus du clavier. Si #Gadget3D est -1 alors le focus (s'il y'en avait un) est enlevé.

Voir aussi

GetActiveGadget3D()

OS Supportés

Tous

100.41 SetGadgetAttribute3D

Syntaxe

```
SetGadgetAttribute3D(#Gadget3D,
    Attribut, Valeur)
```

Description

Change la valeur d'un attribut.

Arguments

#Gadget3D Le gadget 3D à utiliser.

Attribut L'attribut à utiliser.

Valeur La valeur à donner à l'attribut.

Valeur de retour

Aucune.

Remarques

Cette fonction est disponible avec les gadgets suivants :
- EditorGadget3D()
- ProgressBarGadget3D()
- ScrollAreaGadget3D()
- ScrollBarGadget3D()
- SpinGadget3D()

OS Supportés

Tous

100.42 SetGadgetData3D

Syntaxe

```
SetGadgetData3D(#Gadget3D,
    Valeur)
```

Description

Associe une donnée à un gadget 3D.

Arguments

#Gadget3D Le gadget 3D à utiliser.
Valeur La valeur à donner.

Valeur de retour

Aucune.

Remarques

Cette valeur peut être récupérée avec GetGadgetData3D() .
Tous les gadgets sont supportés par cette commande.

Voir aussi

GetGadgetData3D()

OS Supportés

Tous

100.43 SetGadgetItemData3D

Syntaxe

```
SetGadgetItemData3D(#Gadget3D,
    Element, Valeur)
```

Description

Associe une valeur à un élément.

Arguments

#**Gadget3D** Le gadget 3D à utiliser.

Element L'élément à utiliser.

Valeur La valeur à donner à l'élément.

Valeur de retour

Aucune.

Remarques

Cette valeur peut être récupérée avec la commande GetGadgetItemData3D() . Si l'index de l'élément change (par exemple d'autres éléments sont effacés), la valeur restera toujours associée à son élément. Cette commande est supportée par les gadgets suivants :
- ComboBoxGadget3D()
- ListViewGadget3D()

Voir aussi

GetGadgetItemData3D()

OS Supportés

Tous

100.44 SetGadgetItemState3D

Syntaxe

```
SetGadgetItemState3D(#Gadget3D,
    Element, Etat)
```

Description

Change l'état d'un élément.

Arguments

#**Gadget3D** Le gadget 3D à utiliser.

Element L'élément à utiliser.

Etat L'état à donner à l'élément.

Valeur de retour

Aucune.

Remarques

Cette commande est disponible pour les gadgets suivants :
- ListViewGadget3D() : 1 si l'élément doit être sélectionné, 0 sinon.

OS Supportés

Tous

100.45 SetGadgetItemText3D

Syntaxe

```
SetGadgetItemText3D(#Gadget3D,
    Element, Texte$ [,
    Colonne])
```

Description

Change le texte d'un élément.

Arguments

#Gadget3D Le gadget 3D à utiliser.

Element L'élément à utiliser.

Texte$ Le texte à donner à l'élément.

Colonne (optionnel) Pas utilisé pour l'instant.

Valeur de retour

Aucune.

Remarques

Cette commande est disponible pour les gadgets suivants :
- ComboBoxGadget3D()
- ListViewGadget3D()
- PanelGadget3D()

OS Supportés

Tous

100.46 SetGadgetState3D

Syntaxe

```
SetGadgetState3D(#Gadget3D,
    Etat)
```

Description

Change l'état d'un gadget 3D.

Arguments

#Gadget3D Le gadget 3D à utiliser.

Etat Le nouvel état.

Valeur de retour

Aucune.

Remarques

Cette commande est disponible pour les gadgets suivants :
- CheckBoxGadget3D() : 1 pour cocher la case, sinon 0.
- ComboBoxGadget3D() : Change l'élément sélectionné en cours.
- ImageGadget3D() : Change l'image affichée par le gadget.
- ListViewGadget3D() : Change l'élément sélectionné. Si -1, alors aucun élément n'est sélectionné.
- OptionGadget3D() : 1 pour l'activer, sinon 0.
- PanelGadget3D() : Change l'onglet en cours.
- ProgressBarGadget3D() : Change la valeur de la barre de progression.
- ScrollBarGadget3D() : Change la position du curseur.
- SpinGadget3D() : Change la valeur actuelle.

OS Supportés

Tous

100.47 SetGadgetText3D

Syntaxe

```
SetGadgetText3D(#Gadget3D,
    Texte$)
```

Description

Change le texte d'un gadget 3D.

Arguments

#Gadget3D Le gadget 3D à utiliser.

Texte$ Le nouveau texte.

Valeur de retour

Aucune.

Remarques

Cette commande est disponible pour les gadgets suivants :
- ButtonGadget3D() : change le texte du ButtonGadget3D.

- ComboBoxGadget3D()
- EditorGadget3D() : change le texte contenu dans l'EditorGadget3D.
- FrameGadget3D() : change le titre du FrameGadget3D.
- ListViewGadget3D() : sélectionne l'élément qui correspond exactement au texte.
- StringGadget3D() : change le contenu du StringGadget3D.
- TextGadget3D() : change le contenu du TextGadget3D.

OS Supportés

Tous

100.48 SpinGadget3D

Syntaxe

```
Resultat =
    SpinGadget3D(#Gadget3D, X,
    Y, Largeur, Hauteur,
    Minimum, Maximum)
```

Description

Crée un gadget incrémentiel 3D.

Arguments

#Gadget3D Le numéro d'identification du gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Minimum, Maximum Les valeurs minimales et maximales possibles.

Valeur de retour

Renvoie une valeur non nulle si le gadget a été créé avec succès, zéro sinon. Si #PB_Any a été spécifié comme paramètre #Gadget3D alors le numéro généré automatiquement est renvoyé dans 'Resultat' en cas de succès.

Remarques

Vous pouvez ajouter une 'mini aide' à ce gadget en utilisant GadgetToolTip3D() . Les fonctions suivantes peuvent être appelées pour contrôler le gadget : GetGadgetState3D() : Renvoie la valeur du curseur interne.

SetGadgetState3D() : Change la valeur du curseur interne.
GetGadgetText3D() : Renvoie le texte du gadget.
SetGadgetText3D() : Change le texte du gadget.
GetGadgetAttribute3D() avec un des attributs suivants :

```
#PB_Spin3D_Minimum: Renvoie
 la valeur minimale
 possible.
#PB_Spin3D_Maximum: Renvoie
 la valeur maximale
 possible.
```

SetGadgetAttribute3D() : avec un des attributs suivants :

```
#PB_Spin3D_Minimum: Change
 la valeur minimale
 possible.
#PB_Spin3D_Maximum: Change
 la valeur maximale
 possible.
```

OS Supportés

Tous

100.49 StringGadget3D

Syntaxe

```
Resultat =
    StringGadget3D(#Gadget3D,
    X, Y, Largeur, Hauteur,
    Texte$ [, Options])
```

Description

Crée un gadget de saisie de texte 3D.

Arguments

#**Gadget3D** Le numéro d'identification du gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Texte$ Le texte affiché.

Options (optionnel) Peut être une combinaison des constantes suivantes :

```
#PB_String3D_Numeric    :
 Seuls des chiffres
 peuvent être saisis.
```

```
#PB_String3D_Password   :
 Mode mot de passe,
 n'affiche que des '*'.
#PB_String3D_ReadOnly   :
 Mode lecture seule.
```

Valeur de retour

Renvoie une valeur non nulle si le gadget a été créé avec succès, zéro sinon. Si #PB_Any a été spécifié comme paramètre #Gadget3D alors le numéro généré automatiquement est renvoyé dans 'Resultat' en cas de succès.

Remarques

Vous pouvez ajouter une 'mini aide' à ce gadget en utilisant GadgetToolTip3D() . Ce texte pourra être ensuite modifié par les fonctions SetGadgetText3D() et GetGadgetText3D()
Les évènements suivants sont disponibles par l'intermédiaire d'EventType3D() :

```
#PB_EventType3D_Change     :
 Le texte a été modifié par
 l'utilisateur.
#PB_EventType3D_Focus      :
 Le StringGadget obtient le
 focus.
#PB_EventType3D_LostFocus  :
 Le StringGadget a perdu le
 focus.
```

OS Supportés

Tous

100.50 TextGadget3D

Syntaxe

```
Resultat =
    TextGadget3D(#Gadget3D, X,
    Y, Largeur, Hauteur,
    Texte$)
```

Description

Crée un gadget étiquette 3D.

Arguments

#Gadget3D Le numéro d'identification du gadget.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

Texte$ Le texte affiché.

Valeur de retour

Renvoie une valeur non nulle si le gadget a été créé avec succès, zéro sinon. Si #PB_Any a été spécifié comme paramètre #Gadget3D alors le numéro généré automatiquement est renvoyé dans 'Resultat' en cas de succès.

Remarques

Son contenu peut être modifié à l'aide des fonctions SetGadgetText3D() et GetGadgetText3D() .
Ce gadget ne reçoit aucun évènement et GadgetToolTip3D() ne peut pas être utilisé.

OS Supportés

Tous

Chapitre 101

Help

Généralités

L'aide (help) est un composant clé d'un logiciel. Cela permet à l'utilisateur d'en découvrir facilement et efficacement les fonctions. PureBasic permet d'afficher des fichiers d'aide normalisés traitant un niveau global et un niveau contextuel.
Sous Microsoft Windows, deux types de formats sont supportés : .hlp (ancien format) et .chm (nouveau format compatible HTML).

OS Supportés

Tous

101.1 CloseHelp

Syntaxe

```
CloseHelp()
```

Description

Ferme la fenêtre d'aide préalablement ouverte avec OpenHelp() .

Arguments

Aucun.

Valeur de retour

Aucune.

Exemple

```
1  If OpenWindow(0, 0, 0, 220,
     150, "Exemple...",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2    StickyWindow(0,#True)
```

```
3
4       ButtonGadget  (0, 10, 10,
        200, 30, "Aide")
5       ButtonGadget  (1, 10, 60,
        200, 30, "Aide
        DrawingMode()")
6       ButtonGadget  (2, 10,
        110, 200, 30, "Fermer
        l'aide")
7
8     Repeat
9       Event = WaitWindowEvent()
10
11      Select Event
12
13        Case #PB_Event_Gadget
14          Select EventGadget()
15            Case 0
16              OpenHelp(#PB_Compiler_Home
      + "PureBasic.chm","")
17
18            Case 1
19              OpenHelp(#PB_Compiler_Home
      +
        "PureBasic.chm","2ddrawing/drawingmode.html")
20
21            Case 2
22              CloseHelp()
23
24          EndSelect
25
26      EndSelect
27    Until Event =
        #PB_Event_CloseWindow
28    EndIf
```

Voir aussi

OpenHelp()

OS Supportés

Tous

101.2 OpenHelp

Syntaxe

```
OpenHelp(NomFichier$, Theme$)
```

Description

Ouvre et affiche une fenêtre d'aide.

Arguments

NomFichier$ Le nom du fichier d'aide .chm ou .hlp à utiliser.

Theme$ Le nom de la page à afficher. Très utile pour l'aide en ligne / contextuelle).

Valeur de retour

Aucune.

Exemple

```
If OpenWindow(0, 0, 0, 220, 100, "Exemple...", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  ButtonGadget  (0, 10, 10, 200, 30, "Aide")
  ButtonGadget  (1, 10, 60, 200, 30, "Aide DrawingMode()")

Repeat
  Event = WaitWindowEvent()

  Select Event

    Case #PB_Event_Gadget
      Select EventGadget()
        Case 0
          OpenHelp(#PB_Compiler_Home + "PureBasic.chm","")

        Case 1
          OpenHelp(#PB_Compiler_Home + "PureBasic.chm","2ddrawing/drawingmode.html")

      EndSelect

  EndSelect
Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

CloseHelp()

OS Supportés

Tous

Chapitre 102

Http

Généralités

Http est le nom du protocole utilisé par les navigateurs Web pour accéder aux sites distants, comme par exemple une page Web. Chaque donnée distante a son adresse unique : l'URL (Uniform Resource Locator). Cette bibliothèque permet de manipuler facilement les URLs et de télécharger des fichiers distants.
Sous Linux, 'libcurl' doit être installé pour que certaines commandes HTTP fonctionnent (déjà installé dans la plupart des distributions Linux).

OS Supportés

Tous

102.1 AbortHTTP

Syntaxe

```
AbortHTTP(ConnexionHttp)
```

Description

Annule la progression du téléchargement asynchrone spécifié, commencé soit avec ReceiveHTTPFile() soit avec ReceiveHTTPMemory() .

Arguments

ConnexionHttp La connexion Http à interrompre.

Valeur de retour

Aucune.

Remarques

La valeur #PB_HTTP_Aborted sera envoyée par la fonction HTTPProgress() . FinishHTTP() doit être appelée une fois le téléchargement interrompu.
Cette fonction peut être utilisée aussi avec HTTPRequest() ou HTTPRequestMemory() (l'option #PB_HTTP_Asynchronous doit être appliquée).

Voir aussi

HTTPProgress() , FinishHTTP()

OS Supportés

Tous

102.2 FinishHTTP

Syntaxe

```
Resultat =
    FinishHTTP(ConnexionHttp)
```

Description

Libère les ressources associées au téléchargement asynchrone spécifié, commencé soit avec ReceiveHTTPFile() soit avec ReceiveHTTPMemory() .

Arguments

ConnexionHttp La connexion Http à libérer.

Valeur de retour

Selon la commande de réception utilisée pour démarrer le téléchargement, FinishHTTP() renvoie l'un des résultats suivants :
Depuis ReceiveHTTPFile() , la quantité d'octets reçus (taille du fichier stocké) est renvoyée.
Depuis ReceiveHTTPMemory() avec l'option #PB_HTTP_Asynchronous, l'adresse de la zone de mémoire où les données reçues ont été stockées, est renvoyée. MemorySize() permet de connaitre la quantité de données reçues peut alors être déterminée.

Remarques

La valeur #PB_HTTP_Aborted sera envoyée par la fonction HTTPProgress() .

Voir aussi

HTTPProgress() , AbortHTTP()

OS Supportés

Tous

102.3 GetHTTPHeader

Syntaxe

```
Resultat\$ =
    GetHTTPHeader(URL$ [,
    Options [,
    AgentUtilisateur$]])
```

Description

Récupère l'en-tête HTTP d'une URL.

> **Attention**
> Cette fonction est dépréciée, elle sera peut-être supprimée dans une future version de PureBasic. Elle ne doit pas être utilisée dans du nouveau code.

Note : Veuillez utiliser HTTPRequest() à la place.

Arguments

URL$ L'URL à utiliser.
L'URL doit être complète, en incluant le préfixe "http ://" ou "https ://".

Options (optionnel) Peut avoir l'une des valeurs suivantes :

```
#PB_HTTP_NoRedirect: Pas
 de redirections
 automatiques.
```

AgentUtilisateur$ (optionnel) Change l'agent utilisateur (UserAgent) pour la requête HTTP.
L'agent utilisateur par défaut est défini sur "Mozilla / 5.0 Gecko / 41.0 Firefox / 41.0" pour une compatibilité maximale.

Valeur de retour

Renvoie une chaîne de caractères contenant l'en-tête.
Chaque ligne est terminée par #LF$ (voir aussi Chr(10)).
StringField() peut servir à séparer facilement chaque ligne de l'en-tête.
Le contenu des en-têtes est dépendant du type de serveur Web, il est donc possible

d'avoir des informations différentes en fonction du serveur.
Les serveurs produisent des informations utiles comme la date, le type de serveur, sa version, et plus encore.
Exemple d'un en-tête :

```
HTTP/1.1 200 OK
Date: Sat, 02 Aug 2014
 09:15:32 GMT
Server: Apache/2.2.16
 (Debian)
X-Powered-By:
 PHP/5.3.3-7+squeeze19
Vary: Accept-Encoding
Content-Type: text/html
```

Remarques

Certains antivirus sont tellement stricts qu'ils empêchent l'envoi de telles requêtes à moins d'activer leur mode 'Jeu'.
Sous Linux, 'libcurl' doit être installé pour que cette commande fonctionne (déjà installé dans la plupart des distributions Linux).

Exemple

```
1  EnTete$ =
    GetHTTPHeader("http://www.purebasic.com/index.php")
2
3  Repeat
4    Index+1
5    Ligne$ =
    StringField(EnTete$,
    Index, #LF$)
6    Debug Ligne$
7  Until Ligne$ = ""
```

Voir aussi

HTTPRequest() , ReceiveHTTPFile() , URLEncoder()

OS Supportés

Tous

102.4 GetURLPart

Syntaxe

```
Resultat\$ = GetURLPart(URL$,
   Parametre$)
```

Description

Renvoie une partie d'une URL.

Arguments

URL$ L'URL à utiliser.
Une URL peut contenir des paramètres.
C'est utile quand un langage de script est utilisé sur le serveur Web (comme PHP).
La syntaxe est la suivante :
http ://www.purebasic.com/index.php ?test=1.
Ici le paramètre se nomme "test" et sa valeur associée est "1".

Parametre$ La valeur à envoyer.
Les paramètres ne sont pas sensibles à la casse.
De plus, il peut prendre l'une des valeurs prédéfinies suivantes pour accéder facilement à une partie standard de l'URL :

```
#PB_URL_Protocol   : Pour
 renvoyer le protocole
#PB_URL_Site       : Pour
 renvoyer le site
#PB_URL_Port       : Pour
 renvoyer le port (s'il
 existe)
#PB_URL_Parameters: Pour
 renvoyer tous les
 paramètres
#PB_URL_Path       : Pour
 renvoyer le chemin
#PB_URL_User       : Pour
 renvoyer le nom
 d'utilisateur (s'il
 existe)
#PB_URL_Password   : Pour
 renvoyer le mot de passe
 (s'il existe)
```

Valeur de retour

Renvoie la valeur du paramètre ou une autre partie de l'URL.

Exemple

```
1  URL$ =
    "http://user:pass@www.purebasic.com:80/index.php?test=1&ok=2
2
3  Debug GetURLPart(URL$,
    #PB_URL_Protocol)   ;
    Affiche "http"
4  Debug GetURLPart(URL$,
    #PB_URL_Site)       ;
    Affiche "www.purebasic.com"
5  Debug GetURLPart(URL$,
    #PB_URL_Port)       ;
    Affiche "80"
```

```
6   Debug GetURLPart(URL$,
      #PB_URL_Parameters) ;
      Affiche "test=1&ok=2"
7   Debug GetURLPart(URL$,
      #PB_URL_Path)       ;
      Affiche "index.php"
8   Debug GetURLPart(URL$,
      #PB_URL_User)       ;
      Affiche "user"
9   Debug GetURLPart(URL$,
      #PB_URL_Password)   ;
      Affiche "pass"
10  Debug GetURLPart(URL$,
      "test")             ;
      Affiche "1"
11  Debug GetURLPart(URL$,
      "ok")               ;
      Affiche "2"
```

Voir aussi

SetURLPart() , URLDecoder()

OS Supportés

Tous

102.5 HTTPProgress

Syntaxe

```
Resultat =
    HTTPProgress(ConnexionHttp)
```

Description

Renvoie la progression du téléchargement asynchrone spécifié, commencé soit avec ReceiveHTTPFile() soit avec ReceiveHTTPMemory() .

Arguments

ConnexionHttp La connexion HTTP à utiliser.

Valeur de retour

Le nombre d'octets reçu ou une des valeurs suivantes :

```
#PB_Http_Success    : Le
  téléchargement s'est
  terminé avec succès.
#PB_Http_Failed     : Le
  téléchargement a échoué.
```

```
#PB_Http_Aborted    : Le
 téléchargement a été
 interrompu avec AbortHTTP()
```

.

Exemple

```
Telechargement =
 ReceiveHTTPMemory("http://www.purebasic.com/download/OgreAss
 #PB_HTTP_Asynchronous)
If Telechargement
  Repeat
    Progression =
 HTTPProgress(Telechargement)
    Select Progression
      Case #PB_Http_Success
        *Memoire =
 FinishHTTP(Telechargement)
        Debug
 "Téléchargement terminé
 (Taille: " +
 MemorySize(*Memoire) + ")"
        FreeMemory(*Memoire)
        End

      Case #PB_Http_Failed
        Debug "Le
 téléchargement a échoué"
        FinishHTTP(Telechargement)
        End

      Case #PB_Http_Aborted
        Debug "Le
 téléchargement a été
 interrompu"
        FinishHTTP(Telechargement)
        End

      Default
        Debug
 "Téléchargement en cours:
 " + Progression + "octets
 reçus"

    EndSelect

    Delay(500) ; Libérer le
 CPU
  ForEver
Else
  Debug "Erreur de
 téléchargement"
EndIf
```

Voir aussi

ReceiveHTTPFile() ,
ReceiveHTTPMemory()

OS Supportés

Tous

102.6 HTTPInfo

Syntaxe

```
Resultat =
    HTTPInfo(RequeteHttp, Type
    [, Options])
```

Description

Renvoie des informations sur une requête HTTP créée avec HTTPRequest() ou HTTPRequestMemory() .

Arguments

RequeteHttp La requête HTTP.

Type L'information spécifique à obtenir. Cela peut être l'une des valeurs suivantes :

```
#PB_Http_StatusCode  : Le
 code de l'état du
 serveur (200: OK, 404:
 Page non trouvée, etc).
#PB_Http_Response    : La
 réponse du serveur, sous
 forme de texte. Pour
 obtenir la réponse brute
 (raw) , utilisez
 HTTPMemory()
```

.

```
#PB_Http_Headers     : Les
 en-têtes de requête.
#PB_Http_ErrorMessage: Le
 message d'erreur
 (principalement à des
 fins de débogage).
```

Options (optionnel) Format de la réponse HTTP ou de l'encodage d'en-tête. Il peut s'agir de l'une des valeurs suivantes :

```
#PB_UTF8  (par défaut)
#PB_Ascii
```

Valeur de retour

Une chaîne de caractères dépendant du paramètre 'Type'.

Exemple

```
1  HttpRequest =
    HTTPRequest(#PB_HTTP_Get,
    "https://www.google.com")
2  If HttpRequest
3    Debug "Réponse: " +
    HTTPInfo(HTTPRequest,
    #PB_HTTP_Response)
4    Debug "Status : " +
    HTTPInfo(HTTPRequest,
    #PB_HTTP_StatusCode)
5
6    FinishHTTP(HTTPRequest)
7  Else
8    Debug "La requête a
    échoué"
9  EndIf
```

Voir aussi

HTTPRequest() , HTTPRequestMemory()

OS Supportés

Tous

102.7 HTTPMemory

Syntaxe

```
*Resultat =
    HTTPMemory(RequeteHttp)
```

Description

Renvoie l'adresse d'une zone mémoire (tampon) contenant la réponse complète d'une requête HTTP créée avec HTTPRequest() ou HTTPRequestMemory() .

Arguments

RequeteHttp La requête HTTP.

Valeur de retour

Le tampon contenant la réponse complète (raw) d'une requête HTTP.

Remarques

Le tampon doit être libéré avecFreeMemory() .

Exemple

```
RequeteHTTP = HTTPRequest(#PB_HTTP_Get, "https://www.google.com")
If RequeteHTTP
  Debug "Réponse: " + HTTPInfo(RequeteHTTP, #PB_HTTP_Response)

  *Reponse = HTTPMemory(RequeteHTTP)

  FinishHTTP(RequeteHTTP)

  Debug "Taille de la réponse: " + MemorySize(*Reponse)
  FreeMemory(*Reponse)

Else
  Debug "La requête a échoué"
EndIf
```

Voir aussi

HTTPRequest() , HTTPRequestMemory()

OS Supportés

Tous

102.8 HTTPProxy

Syntaxe

```
HTTPProxy(URL$ [, Utilisateur$, MotdePasse$])
```

Description

Spécifie un proxy à utiliser pour les commandes HTTP suivantes : ReceiveHTTPFile() ReceiveHTTPMemory() , HTTPRequest() et HTTPRequestMemory() .

Arguments

URL$ L'URL à utiliser pour le proxy.
Par défaut c'est un proxy HTTP si aucun préfixe n'est spécifié.
Pour les autres types de proxy, voici les préfixes disponibles :

```
http://    - Proxy
 HTTP(par défaut)
socks4://  - Proxy SOCKS4
socks4a:// - Proxy SOCKS4
 avec le support de nom
 de domaine plutôt que
 l'adresse IP
socks5://  - Proxy SOCKS5
socks5h:// - Proxy SOCKS5
 et demande au proxy de
 faire la résolution du
 nom d'hôte
```

Utilisateur$, MotdePasse$ (optionnel)
L'utilisateur et le mot de passe à utiliser pour se connecter au proxy (le cas échéant).

Valeur de retour

Aucune.

Remarques

Sur Linux, 'libcurl' doit être installé pour que cette commande fonctionne (déjà installées sur la plupart des distributions Linux).

Exemple

```
1  HTTPProxy("socks4://127.0.0.1")
2
3  Fichier$ =
    SaveFileRequester("Enregistrer
    le fichier index.php ?",
    "", "", 0)
4
5  If
    ReceiveHTTPFile("http://www.purebasic.com/index.php",
    Fichier$)
6    Debug "Succès"
7  Else
8    Debug "Echèc"
9  EndIf
```

Voir aussi

ReceiveHTTPFile() , ReceiveHTTPMemory() , HTTPRequest() , HTTPRequestMemory()

OS Supportés

Tous

102.9 ReceiveHTTPFile

Syntaxe

```
Resultat =
    ReceiveHTTPFile(URL$,
    NomFichier$ [, Options [,
    AgentUtilisateur$]])
```

Description

Télécharge un fichier sur le disque à partir d'une URL.

Arguments

URL$ L'URL de téléchargement.

NomFichier$ Le nom du fichier qui sera enregistré sur le disque.
Si le nom de fichier ne contient pas de chemin complet, il sera enregistré dans le répertoire courant .
Si le fichier existe, il sera écrasé.

Options (optionnel) Peut être une combinaison des valeurs suivantes :

```
#PB_HTTP_Asynchronous:
 Démarre le
 téléchargement
 asynchrone.
#PB_HTTP_NoRedirect   : Ne
 pas suivre les
 redirections
 automatiques.
```

AgentUtilisateur$ (optionnel) Change l'agent utilisateur (UserAgent) pour la requête HTTP.
L'agent utilisateur par défaut est défini sur "Mozilla / 5.0 Gecko / 41.0 Firefox / 41.0" pour une compatibilité maximale.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_HTTP_Asynchronous a été spécifié, la fonction renvoie la valeur 'ConnexionHttp' nécessaire pour HTTPProgress() , AbortHTTP() et FinishHTTP() .
Si #PB_HTTP_Asynchronous est utilisé alors FinishHTTP() doit être appelé, que le téléchargement ait été effectué avec succès ou non.

Remarques

Sous Linux, 'libcurl' doit être installé pour que cette commande fonctionne (déjà

installé dans la plupart des distributions Linux).

Exemple

```
1  NomDeFichier$ =
     SaveFileRequester("Enregistrer
     index.php ?", "", "", 0)
2
3  If
     ReceiveHTTPFile("http://www.purebasic.com/index.php",
     NomDeFichier$)
4    Debug "Succès"
5  Else
6    Debug "Echec"
7  EndIf
```

Voir aussi

URLEncoder()

OS Supportés

Tous

102.10 ReceiveHTTPMemory

Syntaxe

```
*Resultat =
    ReceiveHTTPMemory(URL$ [,
    Options [,
    AgentUtilisateur$]])
```

Description

Télécharge (Download) un fichier dans un tampon mémoire.

Arguments

URL$ L'URL (l'adresse) du fichier à utiliser.

Options (optionnel) Peut être une combinaison des valeurs suivantes :

```
#PB_HTTP_Asynchronous:
 Démarre le
 téléchargement
 asynchrone.
#PB_HTTP_NoRedirect   : Ne
 pas suivre les
 redirections
 automatiques.
```

AgentUtilisateur$ (optionnel) Change l'agent utilisateur (UserAgent) pour la requête HTTP.
L'agent utilisateur par défaut est défini sur "Mozilla / 5.0 Gecko / 41.0 Firefox / 41.0" pour une compatibilité maximale.

Valeur de retour

Renvoie l'adresse du tampon mémoire en cas de succès, zéro sinon.
MemorySize() peut être utilisé pour obtenir la taille de l'élément téléchargé. Le tampon de mémoire doit être libéré avec FreeMemory() une fois fini.
Si #PB_HTTP_Asynchronous a été spécifié, la fonction renvoie la valeur 'ConnexionHttp' nécessaire pour HTTPProgress() , AbortHTTP() et FinishHTTP() .
Avec #PB_HTTP_Asynchronous, FinishHTTP() doit être appelé que le téléchargement s'est terminé avec succès ou non.

Remarques

Lors d'un téléchargement asynchrone, il est possible d'obtenir l'adresse du tampon mémoire avec FinishHTTP() .
Sous Linux, 'libcurl' doit être installé pour que cette commande fonctionne (déjà installé dans la plupart des distributions Linux).

Exemple

```
1  *Buffer = ReceiveHTTPMemory("http://www.purebasic.com/index.php")
2  If *Buffer
3    Taille = MemorySize(*Buffer)
4    Debug "Contenu: " + PeekS(*Buffer, Taille, #PB_UTF8|#PB_ByteLength)
5    FreeMemory(*Buffer)
6  Else
7    Debug "Le téléchargement a échoué"
8  EndIf
```

Voir aussi

URLEncoder()

OS Supportés

Tous

102.11 HTTPRequest

Syntaxe

```
Resultat = HTTPRequest(Type,
    URL$ [, Data$ [, Options
    [, EnTetes()]]])
```

Description

Envoie une requête HTTP avec des données textuelles optionnelles.

Arguments

Type Le type de la requête. Peut être l'une des valeurs suivantes :

```
#PB_HTTP_Get    : Requête
 GET    (le paramètre
 'Data$' sera ignoré)
#PB_HTTP_Post   : Requête
 POST   (le paramètre
 'Data$' sera envoyé si
 spécifié)
#PB_HTTP_Put    : Requête
 PUT    (le paramètre
 'Data$' sera envoyé si
 spécifié)
#PB_HTTP_Patch  : Requête
 PATCH  (le paramètre
 'Data$' sera envoyé si
 spécifié)
#PB_HTTP_Delete : Requête
 DELETE (le paramètre
 'Data$' sera envoyé si
 spécifié)
```

URL$ L'URL à interroger.

Data$ (optionnel) Les données textuelles à envoyer (envoyées au format UTF-8).

Options (optionnel) Ce peut être une combinaison des valeurs suivantes :

```
#PB_HTTP_Asynchronous:
 Téléchargement de
 manière asynchrone.
#PB_HTTP_NoRedirect  : Ne
 pas suivre les
 redirections
 automatiques.
#PB_HTTP_NoSSLCheck  : Ne
 pas vérifier si le
 certificat SSL est
 valide (peut être utile
 à des fins de test).
#PB_HTTP_HeadersOnly :
 Affiche les en-têtes
 seuls.
```

```
#PB_HTTP_WeakSSL      :
 Support des serveurs
 anciens.
#PB_HTTP_Debug        :
 Imprimer des
 informations dans la
 console de debogage.
```

EnTetes() (optionnel) Une map d'en-têtes supplémentaires, sous forme de chaînes de caractères.
Exemple :

```
1  NewMap Header$()
2  Header$("Content-Type") =
    "text/plain"
3  Header$("User-Agent") =
    "Firefox 54.0"
4  Header$("NoParamHeader") =
    ""
```

Valeur de retour

Renvoie l'identifiant de la requête HTTP si l'appel a été initialisé avec succès, zéro sinon .

Remarques

Sous Linux, vous devez installer 'libcurl' pour que cette commande fonctionne (la plupart des distributions Linux l'ont déjà).
Si des données binaires doivent être envoyées, vous pouvez utiliser HTTPRequestMemory() ().
Cette commande est conçue pour gérer facilement REST comme une API Web.
HTTPInfo() peut être utilisé pour obtenir des informations sur la requête.
Si #PB_HTTP_Asynchronous a été spécifié, HTTPProgress() peut être utilisé et AbortHTTP() doit être utilisé.
HTTPMemory() peut être utilisé pour obtenir le résultat sous forme de tampon brut (le tampon brut doit être libéré avec FreeMemory()).
FinishHTTP() doit toujours être appelé pour terminer une demande HTTP initialisée avec succès, même si l'appel était synchrone.

Exemple

```
1  HttpRequest =
    HTTPRequest(#PB_HTTP_Get,
    "https://www.google.com")
2  If HttpRequest
```

```
    Debug "Status: " +
    HTTPInfo(HTTPRequest,
    #PB_HTTP_StatusCode)
    Debug "Réponse: " +
    HTTPInfo(HTTPRequest,
    #PB_HTTP_Response)

    FinishHTTP(HTTPRequest)
  Else
    Debug "La requête a
    échoué"
  EndIf
```

Exemple : Avec en-têtes

```
  NewMap Header$()
  Header$("Content-Type") =
    "plaintext"
  Header$("User-Agent") =
    "Firefox 54.0"

  HttpRequest =
    HTTPRequest(#PB_HTTP_Get,
    "https://www.google.com",
    "", 0, Header$())
  If HttpRequest
    Debug "Status: " +
    HTTPInfo(HTTPRequest,
    #PB_HTTP_StatusCode)
    Debug "Réponse: " +
    HTTPInfo(HTTPRequest,
    #PB_HTTP_Response)

    FinishHTTP(HTTPRequest)
  Else
    Debug "La requête a
    échoué"
  EndIf
```

Voir aussi

URLEncoder() , AbortHTTP()

OS Supportés

Tous

102.12 HTTPRequestMemory

Syntaxe

```
Resultat =
    HTTPRequestMemory(Type,
    URL$ [, *Data, TailleData
    [, Options [, EnTetes()]]])
```

Description

Envoie une requête HTTP avec des données binaires optionnelles.

Arguments

Type Le type de la requête. Peut être l'une des valeurs suivantes :

```
#PB_HTTP_Get    : GET
 request (le paramètre
 '*Data' sera ignoré)
#PB_HTTP_Post   : POST
 request (le paramètre
 '*Data' sera envoyé si
 spécifié)
#PB_HTTP_Put    : PUT
 request (le paramètre
 '*Data' sera envoyé si
 spécifié)
#PB_HTTP_Patch  : PATCH
 request (le paramètre
 '*Data' sera envoyé si
 spécifié)
#PB_HTTP_Delete: DELETE
 request (le paramètre
 '*Data' sera envoyé si
 spécifié)
```

URL$ L'URL à interroger.

***Data (optionnel)** Adresse mémoire du tampon (buffer) contenant les données à envoyer.

TailleData (optionnel) Taille en octets du tampon.

Options (optionnel) Ce peut être une combinaison des valeurs suivantes :

```
#PB_HTTP_Asynchronous:
 Téléchargement de
 manière asynchrone.
#PB_HTTP_NoRedirect   : Ne
 pas suivre les
 redirections
 automatiques.
#PB_Http_NoSSLCheck   : Ne
 pas vérifier si le
 certificat SSL est
 valide (peut être utile
 à des fins de test).
```

EnTetes() (optionnel) Une map d'en-têtes supplémentaires, sous forme de chaînes de caractères.
Exemple :

```
1  NewMap Header$()
2  Header$("Content-Type") =
    "octectstream"
```

```
3   Header$("User-Agent") =
     "Firefox 54.0"
4   Header$("NoParamHeader") =
     ""
```

Valeur de retour

Renvoie l'identifiant de la requête HTTP si l'appel a été initialisé avec succès, zéro sinon .

Remarques

Sous Linux, vous devez installer 'libcurl' pour que cette commande fonctionne (la plupart des distributions Linux l'ont déjà).
Si des données binaires doivent être envoyées, vous pouvez utiliser HTTPRequestMemory() ().
Cette commande est conçue pour gérer facilement REST comme une API Web.
HTTPInfo() peut être utilisé pour obtenir des informations sur la requête.
Si #PB_HTTP_Asynchronous a été spécifié, HTTPProgress() et AbortHTTP() peuvent être utilisés.
HTTPMemory() peut être utilisé pour obtenir le résultat sous forme de tampon brut (le tampon brut doit être libéré avec FreeMemory()).
FinishHTTP() doit toujours être appelé pour terminer une demande HTTP initialisée avec succès, même si l'appel était synchrone.

Exemple

```
1   HttpRequest =
     HTTPRequestMemory(#PB_HTTP_Get,
     "https://www.google.com")
2   If HttpRequest
3     Debug "Status: " +
     HTTPInfo(HTTPRequest,
     #PB_HTTP_StatusCode)
4     Debug "Réponse: " +
     HTTPInfo(HTTPRequest,
     #PB_HTTP_Response)
5
6     FinishHTTP(HTTPRequest)
7   Else
8     Debug "La requête a
     échoué"
9   EndIf
```

Voir aussi

HTTPRequest() , URLEncoder() , AbortHTTP()

OS Supportés

Tous

102.13 URLDecoder

Syntaxe

```
Resultat\$ = URLDecoder(URL$
    [, Format])
```

Description

Renvoie une URL décodée qui avait été encodée au format HTTP.

Arguments

URL$ L'URL à décoder.

Format (optionnel) Le format d'encodage de l'URL.
Peut être une des valeurs suivantes :

```
#PB_UTF8   (par défaut)
#PB_Ascii
```

Valeur de retour

Renvoie l'URL décodée.

Remarques

Une URL ne peut pas contenir certains caractères (comme la tabulation, l'espace, les lettres accentuées etc.) donc il est nécessaire de les encoder, principalement en utilisant le caractère d'échappement "%" suivi d'un chiffre. Si l'URL$ n'était pas encodée, cette fonction n'a aucun effet et renverra l'URL originale.
Voir ici.

Exemple

```
1  Debug
     URLDecoder("http://www.purebasic.com/test%20with%20space.php
2  ; Affichera
     "http://www.purebasic.com/test
     with space.php"
```

Voir aussi

URLEncoder()

OS Supportés

Tous

102.14 URLEncoder

Syntaxe

```
Resultat\$ = URLEncoder(URL$
    [, Format])
```

Description

Renvoie une URL encodée au format HTTP.

Arguments

URL$ L'URL à encoder.

Format (optionnel) Le format de l'URL avant encodage. Peut être une des valeurs suivantes :

```
#PB_UTF8   (par défaut)
#PB_Ascii
```

Valeur de retour

Renvoie l'URL encodée.

Remarques

Une URL ne peut pas contenir certains caractères (comme la tabulation, l'espace, les lettres accentuées etc.) donc il est nécessaire de les encoder, principalement en utilisant le caractère d'échappement "%" suivi d'un chiffre. Si l'URL$ n'était pas encodée, cette fonction n'a aucun effet et renverra l'URL originale.
Notez que cette fonction suit le standard RFC 3986 et donc certains caractères ne seront pas encodés, car ces caractères doivent rester en clair dans certaines URLs. Il s'agit des caractères suivants (liste non exhaustive) :

```
"-" | "_" | "." | "!" | "~"
  | "*" | "'" | "(" | ")" |
";" | "/" | "?" | ":" | "@"
  | "&" | "=" | "+" | "$" |
"," | """ | "#" | "%" |
```

Toutefois, si vous avez besoin de les coder, il vous faudra utiliser le tableau suivant : https ://www.w3schools.com/tags/ref_urlencode.asp ici.
Par exemple en UTF8 :

```
"-" -> %2D | "_" -> %5F |
"." -> %2E | "!" -> %21 |
"~" -> %7E |

"*" -> %2A | "'" -> %27 |
"(" -> %28 | ")" -> %29 |
";" -> %3B |
```

```
"/" -> %2F | "?" -> %3F |
":" -> %3A | "@" -> %40 |
"&" -> %26 |

"=" -> %3D | "+" -> %2B |
"$" -> %24 | "," -> %2C |
""" -> %22 |

"#" -> %23 | "%" -> %25 |
```

Exemple

```
1  Debug
     URLEncoder("http://www.purebasic.com/test
     with space.php")
2  ; Affichera
     "http://www.purebasic.com/test%20with%20space.php"
3
4  Debug
     URLEncoder("http://www.ok.com
     value=zzz ?yyy/")
5  ; Affichera
     "http://www.ok.com%20value=zzz%20?yyy/"
```

Voir aussi

URLDecoder()

OS Supportés

Tous

102.15 SetURLPart

Syntaxe

```
Resultat\$ = SetURLPart(URL$,
    Parametre$, Valeur$)
```

Description

Change une partie d'une URL.

Arguments

URL$ L'URL à modifier.
Une URL$ peut contenir des paramètres.
C'est utile quand un langage de script est
utilisé sur le serveur Web (comme PHP).
La syntaxe est la suivante :
http ://www.purebasic.com/index.php3 ?test=1.
Ici le paramètre se nomme "test" et sa
valeur associée est "1".
Les informations fournies dans "URL$"
doivent (au minimum), avoir la forme
suivante : "http ://www.purebasic.com"

Parametre$ La valeur à modifier.
Les paramètres ne sont pas sensibles à la casse.
De plus, il peut prendre l'une des valeurs prédéfinies suivantes pour accéder facilement à une partie standard de l'URL :

```
#PB_URL_Protocol  :
 Modifie le protocole de
 l'URL$
#PB_URL_Site      :
 Modifie le site de l'URL$
#PB_URL_Port      :
 Modifie le port de
 l'URL$ (s'il existe)
#PB_URL_Parameters:
 Modifie tous les
 paramètres de l'URL$
#PB_URL_Path      :
 Modifie le chemin de
 l'URL$
#PB_URL_User      :
 Modifie le nom
 d'utilisateur de l'URL$
 (s'il existe)
#PB_URL_Password  :
 Modifie le mot de passe
 de l'URL$
                    (s'il
 existe et si un
 utilisateur' existe
 aussi)
```

Valeur$ La valeur à affecter au paramètre donné, ou une partie d'une URL.

Valeur de retour

Renvoie l'URL modifiée.

Exemple

```
1  URL$ =
    "http://www.test.com/hello.php"
2
3  URL$ = SetURLPart(URL$,
    #PB_URL_Protocol, "ftp")
4  URL$ = SetURLPart(URL$,
    #PB_URL_Site,
    "www.purebasic.com")
5  URL$ = SetURLPart(URL$,
    #PB_URL_Port, "80")
6  URL$ = SetURLPart(URL$,
    #PB_URL_Path,
    "english/index.php")
7  URL$ = SetURLPart(URL$,
    #PB_URL_User, "user")
8  URL$ = SetURLPart(URL$,
    #PB_URL_Password, "pass")
```

```
9   URL$ = SetURLPart(URL$,
     "test", "1")
10  URL$ = SetURLPart(URL$,
     "ok", "2")
11
12  Debug URL$ ; Affichera
     "ftp://user:pass@www.purebasic.com:80/english/index.php?test
```

Voir aussi

GetURLPart() , URLEncoder()

OS Supportés

Tous

Chapitre 103

Image

Généralités

Les images sont des objets graphiques qui peuvent être affichés dans une fenêtre ou dans un gadget. Actuellement, PureBasic accepte les images aux formats BMP, Icône (.ico - seulement sous Windows) et tous les autres formats qui sont supportés via la bibliothèque ImagePlugin .
Le format PNG peut être utilisé pour afficher des images transparentes dans les gadgets , menus et toolbars . Sous Windows, les icônes peuvent aussi remplir ce rôle, mais l'utilisation des images PNG est recommandée, car elles sont supportées sur toutes les plateformes. La transparence des images BMP n'est pas prise en charge.

OS Supportés

Tous

103.1 AddImageFrame

Syntaxe

```
Resultat =
    AddImageFrame(#Image [,
    Index])
```

Description

Ajoute une nouvelle trame (frame = image dans une liste d'image) à l'image spécifiée.

Arguments

#**Image** L'image à utiliser.

Index (optionnel) L'index (à partir de 0) où sera inséré la trame.
Si non spécifié, la nouvelle trame sera ajoutée à la fin de la liste des images.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

La nouvelle trame aura les mêmes dimensions et la même profondeur de couleur que l'image.

Voir aussi

CreateImage() , RemoveImageFrame() , ImageFrameCount() , SetImageFrame() , GetImageFrame()

OS Supportés

Tous

103.2 RemoveImageFrame

Syntaxe

```
Resultat =
    RemoveImageFrame(#Image,
    Index)
```

Description

Retire une trame (frame = image dans une liste d'image) de l'image spécifiée.

Arguments

#Image L'image à utiliser.

Index L'index (à partir de 0) de la trame à retirer.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Si l'image n'est pas multi-trames, cette fonction n'a aucun effet.
ImageOutput() , ImageVectorOutput() , CopyImage() et GrabImage() fonctionne sur l'image courante.

Voir aussi

CreateImage() , AddImageFrame() , ImageFrameCount() , SetImageFrame() , GetImageFrame()

OS Supportés

Tous

103.3 GetImageFrame

Syntaxe

```
Index = GetImageFrame(#Image)
```

Description

Renvoie l'index de la trame (frame = image dans une liste d'image) en cours.

Arguments

#**Image** L'image à utiliser.

Valeur de retour

L'index de la trame en cours. (à partir de 0).
Renvoie toujours zéro si l'image n'est pas multi-trames.

Voir aussi

CreateImage() , AddImageFrame() , RemoveImageFrame() , ImageFrameCount() , GetImageFrame()

OS Supportés

Tous

103.4 SetImageFrame

Syntaxe

```
SetImageFrame(#Image, Index)
```

Description

Spécifie la trame courante.

Arguments

#**Image** L'image à utiliser.

Index L'index de la trame. (Commence à 0).

Valeur de retour

Aucune.

Voir aussi

CreateImage() , AddImageFrame() , RemoveImageFrame() , ImageFrameCount() , GetImageFrame()

OS Supportés

Tous

103.5 ImageFrameCount

Syntaxe

```
Resultat =
    ImageFrameCount(#Image)
```

Description

Renvoie le nombre de trames de l'image.

Arguments

#**Image** L'image à utiliser.

Valeur de retour

Renvoie le nombre de trames de l'image. Renvoie toujours 1 si l'image n'est pas multi-trames.

Voir aussi

CreateImage() , AddImageFrame() , RemoveImageFrame() , SetImageFrame() , GetImageFrame()

OS Supportés

Tous

103.6 GetImageFrameDelay

Syntaxe

```
Resultat =
    GetImageFrameDelay(#Image)
```

Description

Renvoie le délai d'affichage de la trame courante.

Arguments

#**Image** L'image à utiliser.

Valeur de retour

Renvoie le délai d'affichage (en millisecondes) de la trame en cours d'affichage.

Remarques

Chaque trame peut avoir son propre délai d'affichage.

Voir aussi

CreateImage() , AddImageFrame() , RemoveImageFrame() , SetImageFrame() , GetImageFrame() , SetImageFrameDelay()

OS Supportés

Tous

103.7 SetImageFrameDelay

Syntaxe

```
Resultat =
    SetImageFrameDelay(#Image,
    Delai)
```

Description

Définit le délai d'affichage de la trame courante.

Arguments

#**Image** L'image à utiliser.

Valeur de retour

Définit le délai d'affichage (en millisecondes) de la trame en cours d'affichage.

Remarques

Chaque trame peut avoir son propre délai d'affichage.

Voir aussi

CreateImage() , AddImageFrame() , RemoveImageFrame() , SetImageFrame() , GetImageFrame() , GetImageFrameDelay()

OS Supportés

Tous

103.8 CatchImage

Syntaxe

```
Resultat = CatchImage(#Image,
    *AdresseMemoire [, Taille])
```

Description

Charge une image à partir de l'emplacement mémoire spécifié.

Arguments

#Image Le numéro d'identification de l'image.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

***AdresseMemoire** L'adresse mémoire où se trouve l'image à charger.

Taille (optionnel) La taille de l'image en octets.
La taille est facultative car le chargeur (loader) peut déterminer quand arrêter la lecture, à partir de l'image. Il est cependant conseillé de prévoir une taille lors du chargement d'images inconnues, car le chargeur peut ensuite gérer correctement les images corrompues (sans préciser la taille de l'image, une image corrompue peut faire planter le programme).

Valeur de retour

Renvoie une valeur non nulle si l'image a été chargée avec succès , zéro sinon. Si #PB_Any a été spécifié comme paramètre #Image alors le numéro généré automatiquement est renvoyé dans 'Resultat' en cas de succès.

Remarques

La taille maximale de l'image est fonction du système d'exploitation et de la quantité de mémoire disponible. Si la mémoire disponible est suffisante alors une image de 8192x8192 pixels au moins peut être traitée par tous les systèmes d'exploitation pris en charge par PureBasic.
Quand une image est chargée, elle est convertie soit en 24 bits (si la profondeur de l'image est inférieure ou égale à 24 bits) soit en 32 bits (si l'image a un canal alpha). La transparence des images BMP n'est pas prise en charge.

Quand une image n'est plus utile, elle peut être détruite avec la commande FreeImage() pour libérer de la mémoire.
Cette commande est utile pour inclure des images (logo etc..) directement dans l'exécutable avec la commande IncludeBinary . Néanmoins, n'abusez pas de cette fonctionnalité car une image incluse dans un exécutable consomme automatiquement de la mémoire supplémentaire puisque tout l'exécutable, y compris les images, est chargé en mémoire.
L'image peut être au format BMP, icône (.ico, seulement supporté sous Windows) ou dans un des formats supportés par la bibliothèque ImagePlugin. Sous Windows, si le format de l'image est une icône (.ico), les commandes suivantes ne seront pas disponibles pour la manipuler : SaveImage() et ImageOutput() .
Les commandes suivantes peuvent être utilisées pour activer automatiquement le support des formats d'images suivants :
UseJPEGImageDecoder()
UseJPEG2000ImageDecoder()
UsePNGImageDecoder()
UseTIFFImageDecoder()
UseTGAImageDecoder()
UseGIFImageDecoder()

Exemple

```
1   If OpenWindow(0, 0, 0, 256, 256, "CatchImage", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2     If CatchImage(0, ?Image_Start, ?Image_End - ?Image_Start)
3       ImageGadget(0, 0, 0, ImageWidth(0), ImageHeight(0), ImageID(0))
4       Repeat : Until WaitWindowEvent() = #PB_Event_CloseWindow
5     EndIf
6   EndIf
7
8   DataSection
9     Image_Start:
10      IncludeBinary #PB_Compiler_Home+"Examples\Sources\Data\Background.bmp"
11    Image_End:
12  EndDataSection
```

Notes : Le " ?" est un pointeur vers un label. Vous trouverez plus d'informations au sujet des pointeurs et des accès mémoire ici

Voir aussi

CreateImage() , LoadImage() , FreeImage() , ImagePlugin library

OS Supportés

Tous

103.9 CopyImage

Syntaxe

```
Resultat = CopyImage(#Image1, #Image2)
```

Description

Copie une image.

Arguments

#Image1 Le numéro de l'image à copier.

#Image2 Le numéro de la copie. #PB_Any peut être utilisé pour générer automatiquement ce numéro. Note : Si l'#Image2 existait déjà, l'ancienne image est automatiquement supprimée.

Valeur de retour

Renvoie une valeur non nulle si l'image a été copiée avec succès, zéro sinon. Si #PB_Any a été spécifié comme paramètre #Image2 alors le numéro généré automatiquement est renvoyé dans 'Resultat' en cas de succès.

Remarques

Si l'image est multi-image (multi-trame), l'image courante sera utilisée pour la copie.

Exemple

```
1  If OpenWindow(0, 0, 0, 622, 256, "CopyImage", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2    ImageGadget(0, 0, 0, 256, 256, LoadImage(0, #PB_Compiler_Home+"Examples\Sources\Data\Background.bmp"))
3    ButtonGadget(1, 266, 100, 90, 30, "Copie ->")
4    ImageGadget(2, 366, 0, 256, 256, 0)
5
6    Repeat
```

```
7       Event =
    WaitWindowEvent()
8       If Event =
    #PB_Event_Gadget
9           Select EventGadget()
10            Case 1
11              CopyImage(0,1) ;
    Crée l'image 1 et copie
    image 0 dedans.
12              SetGadgetState(2,ImageID(1))
    ; Affichage
13          EndSelect
14      EndIf
15    Until Event =
    #PB_Event_CloseWindow
16  EndIf
```

Voir aussi

GrabImage() , FreeImage()

OS Supportés

Tous

103.10 CreateImage

Syntaxe

```
Resultat =
    CreateImage(#Image,
    Largeur, Hauteur [,
    Profondeur [,
    CouleurDeFond]])
```

Description

Crée une nouvelle image vierge (fond noir).

Arguments

#Image Le numéro d'identification de l'image.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Largeur, Hauteur Les dimensions de la nouvelle image.
À la fois la largeur et la hauteur doivent être supérieures à zéro.

Profondeur (optionnel) La profondeur de couleur de la nouvelle image.
Les valeurs valides sont : 24 et 32.
La valeur par défaut est 24 bits, si la profondeur n'est pas spécifiée.

CouleurDeFond (optionnel) La couleur de fond RGB() utilisée.

Cette couleur peut être réglée sur #PB_Image_Transparent pour créer une image avec le canal alpha fixé pour une transparence totale.
Cela n'a d'effet que sur les images 32 bits.
Si non spécifié alors la couleur par défaut est le noir.

Valeur de retour

Renvoie une valeur non nulle si l'image a été créée avec succès et zéro sinon. Si #PB_Any a été spécifié comme paramètre #Image alors le numéro généré automatiquement est renvoyé dans 'Resultat' en cas de succès.

Remarques

La taille maximale de l'image est fonction du système d'exploitation et de la quantité de mémoire disponible. Si la mémoire disponible est suffisante alors une image de 8192x8192 peut être traitée par tous les systèmes d'exploitation pris en charge par PureBasic.
Avec l'image créée , vous pouvez notamment utiliser les commandes suivantes :
StartDrawing() avec ImageOutput() pour dessiner dans l'image.
StartVectorDrawing() avec ImageVectorOutput() pour faire du dessin vectoriel.
CopyImage() pour créer une copie de l'image.
GrabImage() pour créer une autre image en copiant une zone définie dans l'image.
DrawImage() avec ImageID() pour dessiner l'image sur la surface de dessin en cours.
ImageGadget() pour afficher l'image dans une application fenêtrée.
ButtonImageGadget() pour créer un bouton avec image dans une application fenêtrée.

Exemple

```
1  If OpenWindow(0, 0, 0, 256,
     256, "CreateImage",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2     ImageGadget(0, 0, 0, 256,
     256, 0)
3
4     If CreateImage(0, 256,
     256, 32, RGB(255,255,255))
5        StartDrawing(ImageOutput(0))
6        Ellipse(70, 50, 30, 10,
     RGB(0,0,255))
```

```
7       Ellipse(186, 50, 30,
    10, RGB(0,0,255))
8       LineXY(128, 80, 128,
    150, RGB(255, 0, 255))
9       RoundBox(70, 200, 110,
    30, 20, 20, RGB(255, 0, 0))
10      StopDrawing()
11     EndIf
12
13     SetGadgetState(0,
    ImageID(0)) ; Affichage
14
15     Repeat : Until
    WaitWindowEvent() =
    #PB_Event_CloseWindow
16    EndIf
```

Voir aussi

LoadImage() , CatchImage() , FreeImage()

OS Supportés

Tous

103.11 EncodeImage

Syntaxe

```
*Resultat =
    EncodeImage(#Image [,
    ImagePlugin [, Options [,
    Profondeur]]])
```

Description

Encode une image dans une mémoire tampon.

Arguments

#**Image** L'image à encoder.

ImagePlugin (optionnel) Le format de l'image à encoder. Il peut être une des valeurs suivantes :

```
#PB_ImagePlugin_BMP       :
  Encoder l'image en BMP
  (Par défaut)
#PB_ImagePlugin_JPEG      :
  Encoder l'image en JPEG
  (UseJPEGImageEncoder()
doit être utilisé)
#PB_ImagePlugin_JPEG2000 :
  Encoder l'image en
  JPEG2000
  (UseJPEG2000ImageEncoder()
doit être utilisé)
```

```
#PB_ImagePlugin_PNG       :
 Encoder l'image en PNG
 (UsePNGImageEncoder()
doit être utilisé)
```

Options (optionnel) Qualité de l'image pour #PB_ImagePlugin_JPEG et #PB_ImagePlugin_JPEG2000 :
De 0 = Qualité médiocre à 10 = Qualité maximale.
Qualité par défaut = 7, si aucune option n'est spécifiée.
Quand une image est codée en utilisant une palette (1, 4 ou 8), la constante suivante peut faire partie de la combinaison :

```
#PB_Image_FloydSteinberg:
 Applique un filtre
 Floyd-Steinberg.
```

Profondeur (optionnel) La profondeur dans lequel vous souhaitez enregistrer l'image.
Les valeurs valides sont 1, 4, 8, 24 et 32.
La valeur par défaut est la profondeur de l'image originale.
Pour l'instant, seul l'encodeur PNG supporte le format d'image palettisée (1, 4 ou 8 bits).

Valeur de retour

Renvoie un pointeur vers une mémoire tampon nouvellement allouée contenant l'image encodée, ou zéro si l'encodeur a échoué.
MemorySize() peut être utilisé pour obtenir la taille de l'image encodée.
FreeMemory() doit être utilisé pour libérer la mémoire tampon après utilisation.

Exemple

```
1  UsePNGImageEncoder()
2  UsePNGImageDecoder()
3  If OpenWindow(0, 0, 0, 426,
    128, "EncodeImage",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
4   ImageGadget(0, 0, 0, 128,
   128, LoadImage(0,
   #PB_Compiler_Home+"Examples\Sources\Data\GeeBee2.bmp"))
5   ButtonGadget(1, 138, 50,
   150, 30, "Encodage PNG 2
   bits ->")
6   ImageGadget(2, 298, 0,
   128, 128, 0)
7
```

```
8     *ptr=EncodeImage(0
      ,#PB_ImagePlugin_PNG,
      #PB_Image_FloydSteinberg,
      2) ; Encodage 2 bits
9
10    Repeat
11      Event =
      WaitWindowEvent()
12      If Event =
      #PB_Event_Gadget
13        Select EventGadget()
14          Case 1
15            CatchImage(2,
      *ptr)          ; L'image
      est en mémoire
16            SetGadgetState(2,ImageID(2))
      ; Affichage
17        EndSelect
18      EndIf
19    Until Event =
      #PB_Event_CloseWindow
20    EndIf
```

Voir aussi

LoadImage() , ImagePlugin library

OS Supportés

Tous

103.12 FreeImage

Syntaxe

```
FreeImage(#Image)
```

Description

Supprime une image et libère la mémoire associée.

Arguments

#**Image** L'image à supprimer.
Si #PB_All est spécifié, toutes les images restantes seront libérées.

Valeur de retour

Aucune.

Remarques

Toutes les images restantes sont automatiquement supprimées quand le programme se termine.

Exemple

```
If OpenWindow(0, 0, 0, 426, 128, "FreeImage", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  ImageGadget(0, 0, 0, 128, 128, LoadImage(0, #PB_Compiler_Home+"Examples\Sources\Data\GeeBee2.bmp"))
  ButtonGadget(1, 138, 50, 150, 30, "Couper l'image en 2 ->")
  ImageGadget(2, 298, 0, 128, 128, 0)

  Repeat
    Event = WaitWindowEvent()
    If Event = #PB_Event_Gadget
      Select EventGadget()
        Case 1
          If IsImage(0)
          GrabImage(0, 1, 0, 0, 128, 64); Coupe l'image
          SetGadgetState(2, ImageID(1)) ; Affichage
          SetGadgetState(0, 0) ; Efface l'image d'origine
          FreeImage(0)
          ; Detruit l'image d'origine
        EndIf

      EndSelect
    EndIf
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

CreateImage() , LoadImage() , CatchImage() , CopyImage() , GrabImage()

OS Supportés

Tous

103.13 GrabImage

Syntaxe

```
Resultat = GrabImage(#Image1, #Image2, X, Y, Largeur, Hauteur)
```

Description

Crée une nouvelle image avec la zone sélectionnée de l'#Image1.

Arguments

#Image1 L'image source.

#Image2 Le numéro de la nouvelle image. #PB_Any peut être utilisé pour générer automatiquement ce numéro.
Note : Si l'#Image2 existait déjà, l'ancienne image est automatiquement supprimée.

X, Y, Largeur, Hauteur La position et la taille de la zone à copier.

Valeur de retour

Renvoie une valeur non nulle si l'image a été créée avec succès et zéro sinon.
Si #PB_Any a été spécifié comme paramètre #Image2 alors le numéro généré automatiquement est renvoyé dans 'Resultat' en cas de succès.

Remarques

Si l'image est multi-image (multi-trame), l'image courante sera utilisée pour la création.

Exemple

```
1   If OpenWindow(0, 0, 0, 426, 128, "GrabImage", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2     ImageGadget(0, 0, 0, 128, 128, LoadImage(0, #PB_Compiler_Home+"Examples\Sources\Data\GeeBee2.bmp"))
3     ButtonGadget(1, 138, 50, 150, 30, "Couper l'image en 2 ->")
4     ImageGadget(2, 298, 0, 128, 128, 0)
5
6     Repeat
7       Event = WaitWindowEvent()
8       If Event = #PB_Event_Gadget
9         Select EventGadget()
10          Case 1
11            GrabImage(0, 1, 0, 0, 128, 64); Coupe l'image
```

```
12              SetGadgetState(2,
    ImageID(1)) ; Affichage
13          EndSelect
14        EndIf
15      Until Event =
    #PB_Event_CloseWindow
16    EndIf
```

Voir aussi

CreateImage() , LoadImage() , CatchImage() , CopyImage()

OS Supportés

Tous

103.14 ImageDepth

Syntaxe

```
Resultat = ImageDepth(#Image
    [, Options])
```

Description

Renvoie la profondeur de couleur d'une image.

Arguments

#**Image** L'image à utiliser.

Options (optionnel) Peut prendre une des valeurs suivantes :

```
#PB_Image_InternalDepth:
 Renvoie la profondeur de
 l'image une fois décodée
 (défaut). Valeurs
 valides:

 24 (24 bits soit 16
 millions de couleurs)

 32 (32 bits soit 16
 millions de couleurs + 8
 bits de canal alpha)
#PB_Image_OriginalDepth:
 Renvoie la profondeur
 originale de l'image
 avant le décodage.
 Valeurs valides:

 1 (1 bit  soit 2
 couleurs, image
 monochrome)
```

-

```
4 (4 bits soit 16
couleurs)
```

-

```
8 (8 bits soit 256
couleurs)
```

-

```
16 (16 bits soit 65536
couleurs)
```

-

```
24 (24 bits soit 16
millions de couleurs)
```

-

```
32 (32 bits soit 16
millions de couleurs + 8
bits de canal alpha)
```

Valeur de retour

Renvoie l'une des valeurs de profondeur possibles décrites ci-dessus.

Remarques

Quand une image est chargée, elle est convertie soit en 24 bits (si la profondeur de l'image est inférieure ou égale à 24 bits) soit en 32 bits (si l'image a un canal alpha).

Exemple

```
1  LoadImage(0,
     #PB_Compiler_Home +
     "Examples\Sources\Data\GeeBee2.bmp")
2
3  Debug "La profondeur de
     l'image une fois décodée"
4  Debug ImageDepth(0,
     #PB_Image_InternalDepth)
5
6  Debug "La profondeur
     originale de l'image avant
     le décodage"
7  Debug ImageDepth(0,
     #PB_Image_OriginalDepth)
```

Voir aussi

ImageWidth() , ImageHeight()

OS Supportés

Tous

103.15 ImageFormat

Syntaxe

```
Resultat = ImageFormat(#Image)
```

Description

Renvoie le format d'une image.

Arguments

#**Image** L'image à utiliser.

Valeur de retour

Renvoie le format d'origine de l'image. Il peut être une des valeurs suivantes :

```
#PB_ImagePlugin_JPEG
#PB_ImagePlugin_JPEG2000
#PB_ImagePlugin_PNG
#PB_ImagePlugin_TGA
#PB_ImagePlugin_TIFF
#PB_ImagePlugin_BMP
#PB_ImagePlugin_ICON
#PB_ImagePlugin_GIF
```

Si l'image n'a pas été chargée avec l'un de ces formats, la fonction renverra zéro. C'est le cas pour les images créées avec CreateImage() ou GrabImage() .

Exemple

```
1  LoadImage(0,
     #PB_Compiler_Home +
     "Examples\Sources\Data\GeeBee2.bmp")
2
3  Select ImageFormat(0)
4    Case #PB_ImagePlugin_JPEG
5       Debug "JPEG"
6    Case
     #PB_ImagePlugin_JPEG2000
7       Debug "JPEG2000"
8    Case #PB_ImagePlugin_PNG
9       Debug "PNG"
10   Case #PB_ImagePlugin_TGA
11      Debug "TGA"
12   Case #PB_ImagePlugin_TIFF
13      Debug "TIFF"
14   Case #PB_ImagePlugin_BMP
15      Debug "BMP"
16   Case #PB_ImagePlugin_ICON
17      Debug "ICON "
18   Default
19      Debug "Inconnu"
20 EndSelect
```

Voir aussi

LoadImage() , CreateImage() , CatchImage() , GrabImage() , ImagePlugin library

OS Supportés

Tous

103.16 ImageHeight

Syntaxe

```
Hauteur = ImageHeight(#Image)
```

Description

Renvoie la hauteur d'une image.

Arguments

#**Image** L'image à utiliser.

Valeur de retour

Renvoie la hauteur de l'image en pixels.

Exemple

```
1  LoadImage(0,
     #PB_Compiler_Home +
     "Examples\Sources\Data\GeeBee2.bmp")
2
3  Debug "Hauteur de l'image"
4  Debug ImageHeight(0)
5
6  Debug "Largeur de l'image"
7  Debug ImageWidth(0)
```

Voir aussi

ImageWidth() , ImageDepth()

OS Supportés

Tous

103.17 ImageID

Syntaxe

```
Resultat = ImageID(#Image)
```

Description

Renvoie l'identifiant système d'une image.

Arguments

#**Image** L'image à utiliser.

Valeur de retour

Renvoie le handle de l'image (envoyé par le système d'exploitation).

Exemple

```
1  hImg=LoadImage(0,
     #PB_Compiler_Home +
     "Examples\Sources\Data\GeeBee2.bmp")
2
3  Debug ImageID(0)
4  Debug hImg
```

Voir aussi

DrawImage() , ImageGadget() , ButtonImageGadget() , CanvasGadget()

OS Supportés

Tous

103.18 ImageOutput

Syntaxe

```
Resultat = ImageOutput(#Image)
```

Description

Renvoie l'identifiant OutputID nécessaire pour les opérations de rendu 2D .

Arguments

#**Image** L'image qui reçoit le dessin.

Valeur de retour

Renvoie l'identifiant 'OutputID' ou zéro si le dessin n'est pas possible.
Cette valeur doit être transmise directement à la fonction StartDrawing() pour commencer l'opération de dessin. La valeur de retour n'est valide que pour une seule opération de dessin et ne peut pas être réutilisée. Chaque StartDrawing() nécessite un nouvel appel à ImageOutput().

Remarques

A la place de cette fonction, la commande ImageVectorOutput() peut être utilisée pour effectuer du dessin vectoriel sur l'image.
Cette fonction ne peut pas être utilisée avec les fichiers icône (*.ico).
Si l'image est multi-image (multi-trame), l'image courante sera utilisée.

Exemple

```
If OpenWindow(0, 0, 0, 256, 256, "CreateImage", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  ImageGadget(0, 0, 0, 256, 256, 0)

  If CreateImage(0, 256, 256, 32, RGB(255,255,255))
    StartDrawing(ImageOutput(0))
    Ellipse(70, 50, 30, 10, RGB(0,0,255))
    Ellipse(186, 50, 30, 10, RGB(0,0,255))
    LineXY(128, 80, 128, 150, RGB(255, 0, 255))
    RoundBox(70, 200, 110, 30, 20, 20, RGB(255, 0, 0))
    StopDrawing()
  EndIf

  SetGadgetState(0,ImageID(0)) ; Affichage

  Repeat : Until WaitWindowEvent() = #PB_Event_CloseWindow
EndIf
```

Voir aussi

StartDrawing() , ImageVectorOutput()

OS Supportés

Tous

103.19 ImageVectorOutput

Syntaxe

```
Resultat = ImageVectorOutput(#Image [, UniteDeMesure])
```

Description

Renvoie le numéro d'identification OutputID d'une image afin d'effectuer des opérations de dessin vectoriel .

Arguments

#**Image** L'image à utiliser.

UniteDeMesure (optionnel) Spécifie l'unité utilisée pour mesurer les distances sur le dessin.

```
#PB_Unit_Pixel      : Les valeurs sont mesurées en pixels (Par défaut)(ou point (dots) pour les imprimantes)
#PB_Unit_Point      : Les valeurs sont mesurées en points (1/72 pouce = 25.4/72 mm = 0,352 778 mm)
#PB_Unit_Inch       : Les valeurs sont mesurées en pouces (25,4 millimètres)
#PB_Unit_Millimeter: Les valeurs sont mesurées en millimètres (0,039 370 pouce)
```

Valeur de retour

Renvoie l'identifiant ouputID ou zéro si le dessin n'est pas possible.
Cette valeur doit être transmise directement à la fonction StartVectorDrawing() pour lancer l'opération de dessin.
La valeur de retour n'est valable que pour une seule opération de dessin et ne peut pas être réutilisée.

Remarques

Si l'image est multi-image (multi-trame), l'image courante sera utilisée.

Exemple

```
1  StartVectorDrawing(ImageVectorOutput(#Image, #PB_Unit_Millimeter))
2    ; code de dessin ici ...
3  StopVectorDrawing()
```

Remarques

Cette fonction ne peut pas être utilisée avec les fichiers icône (*.ico).

Voir aussi

StartVectorDrawing() , ImageOutput()

OS Supportés

Tous

103.20 ImageWidth

Syntaxe

```
Largeur = ImageWidth(#Image)
```

Description

Renvoie la largeur d'une image.

Arguments

#**Image** L'image à utiliser.

Valeur de retour

Renvoie la largeur de l'image en pixels.

Exemple

```
1  LoadImage(0,
     #PB_Compiler_Home +
     "Examples\Sources\Data\GeeBee2.bmp")
2
3  Debug "Hauteur de l'image"
4  Debug ImageHeight(0)
5
6  Debug "Largeur de l'image"
7  Debug ImageWidth(0)
```

Voir aussi

ImageHeight() , ImageDepth()

OS Supportés

Tous

103.21 IsImage

Syntaxe

```
Resultat = IsImage(#Image)
```

Description

Teste si une image est correctement initialisée.

Arguments

#**Image** L'image à tester.

Valeur de retour

Renvoie une valeur non nulle si l'image est une image valide et zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.

Exemple

```
1   If OpenWindow(0, 0, 0, 426, 128, "IsImage", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2     ImageGadget(0, 0, 0, 128, 128, LoadImage(0, #PB_Compiler_Home+"Examples\Sources\Data\GeeBee2.bmp"))
3     ButtonGadget(1, 138, 50, 150, 30, "Couper l'image en 2 ->")
4     ImageGadget(2, 298, 0, 128, 128, 0)
5
6     Repeat
7       Event = WaitWindowEvent()
8       If Event = #PB_Event_Gadget
9         Select EventGadget()
10          Case 1
11            If IsImage(0) ; L'image existe-t-elle ?
12              GrabImage(0, 1, 0, 0, 128, 64); Coupe l'image
13              SetGadgetState(2, ImageID(1)) ; Affichage
14              SetGadgetState(0, 0) ; Efface l'image d'origine
15              FreeImage(0)
                ; Detruit l'image d'origine
16            EndIf
17
18        EndSelect
19      EndIf
20    Until Event = #PB_Event_CloseWindow
21  EndIf
```

Voir aussi

CreateImage() , LoadImage() , CatchImage() , CopyImage() , GrabImage()

OS Supportés

Tous

103.22 LoadImage

Syntaxe

```
Resultat = LoadImage(#Image,
    Fichier$ [, Options])
```

Description

Charge une image contenue dans fichier.

Arguments

#Image Le numéro d'identification de l'image à charger.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Fichier$ Le chemin et le nom du fichier à charger.
Si le nom de fichier ne contient pas de chemin complet, le chemin est relatif.
Voir répertoire courant .

Options (optionnel) Ce paramètre n'a pour l'instant pas d'utilité. S'il est spécifié, 0 doit être utilisé pour assurer une future compatibilité.

Valeur de retour

Renvoie une valeur non nulle si l'image a été chargée avec succès et zéro sinon. Si #PB_Any a été spécifié comme paramètre #Image alors le numéro généré automatiquement est renvoyé dans 'Resultat' en cas de succès.

Remarques

La taille maximale de l'image est fonction du système d'exploitation et de la quantité de mémoire disponible. Si la mémoire disponible est suffisante alors une image de 8192x8192 peut être traitée par tous les systèmes d'exploitation pris en charge par PureBasic.
Quand une image est chargée, elle est convertie soit en 24 bits (si la profondeur de l'image est inférieure ou égale à 24 bits) soit en 32 bits (si l'image a un canal alpha). La

transparence des images BMP n'est pas prise en charge.
Une image chargée peut être libérée en utilisant la fonction FreeImage() .
L'image peut être au format BMP, icône (.ico, seulement supporté sous Windows) ou dans un des formats supportés par la bibliothèque ImagePlugin .
Les commandes suivantes peuvent être utilisées pour activer automatiquement davantage de formats d'images :
UseJPEGImageDecoder()
UseJPEG2000ImageDecoder()
UsePNGImageDecoder()
UseTIFFImageDecoder()
UseTGAImageDecoder()
UseGIFImageDecoder()
Avec l'image chargée, vous pouvez notamment utiliser les commandes suivantes :
StartDrawing() avec ImageOutput() pour dessiner dans l'image.
StartVectorDrawing() avec ImageVectorOutput() pour faire du dessin vectoriel dans l'image.
CopyImage() pour créer une copie de l'image.
GrabImage() pour créer une autre image en copiant une zone définie dans l'image.
DrawImage() avec ImageID() pour dessiner l'image sur la surface de dessin en cours.
DrawAlphaImage() avec ImageID() pour dessiner l'image. (avec ses composantes alpha) sur la surface de dessin en cours.
ImageGadget() pour afficher l'image dans une application fenêtrée.
ButtonImageGadget() pour créer un bouton avec image dans une application fenêtrée.

Exemple

```
1  If OpenWindow(0, 0, 0, 128, 128, "FreeImage", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2
3    LoadImage(0, #PB_Compiler_Home + "Examples\Sources\Data\GeeBee2.bmp") ; Charge l'image depuis un fichier
4    ImageGadget(0, 0, 0, 128, 128, ImageID(0)) ; Affichage
5
6    Repeat : Until WaitWindowEvent() = #PB_Event_CloseWindow
7  EndIf
```

Voir aussi

CreateImage() , CatchImage() , CopyImage() , GrabImage() , ImagePlugin library

OS Supportés

Tous

103.23 ResizeImage

Syntaxe

```
Resultat =
    ResizeImage(#Image,
    Largeur, Hauteur [, Mode])
```

Description

Redimensionne une image.

Arguments

#Image L'image à redimensionner.

Largeur, Hauteur Les nouvelles dimensions de l'image.
À la fois la largeur et la hauteur doivent être supérieures à zéro.
#PB_Ignore peut être spécifié pour la largeur ou la hauteur, ainsi la valeur ne sera pas modifiée.

Mode (optionnel) Peut prendre une des valeurs suivantes :

```
#PB_Image_Smooth:
 Redimensionne l'image
 avec interpolation (mode
 par défaut).
#PB_Image_Raw    :
 Redimensionne l'image
 sans interpolation (peut
 donner des résultats peu
 satisfaisants).
```

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi et zéro sinon.

Remarques

Cette fonction modifie le handle de l'image utilisée. Par conséquent, il doit être

ré-affecté par exemple à un ImageGadget() avec SetGadgetState() .
Cette fonction ne fonctionne pas avec des icônes (.ico) ni avec les images multi-frame. La limite de la taille de l'image qui peut être traitée est fonction du système d'exploitation et de la quantité de mémoire disponible. Si la mémoire disponible est suffisante alors une image de 8192x8192 peut être traitée par tous les systèmes d'exploitation pris en charge par PureBasic.

Exemple

```
1   If OpenWindow(0, 0, 0, 426, 128, "ResizeImage", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2     ImageGadget(0, 0, 0, 128, 128, LoadImage(0, #PB_Compiler_Home+"Examples\Sources\Data\GeeBee2.bmp"))
3     ButtonGadget(1, 138, 50, 150, 30, "Réduire l'image ->")
4     ImageGadget(2, 298, 0, 128, 128, 0)
5
6     Repeat
7       Event = WaitWindowEvent()
8       If Event = #PB_Event_Gadget
9         Select EventGadget()
10          Case 1
11            CopyImage(0,1)
12            ResizeImage(1, 64, 64); Coupe l'image
13            SetGadgetState(2, ImageID(1)) ; Affichage
14        EndSelect
15      EndIf
16    Until Event = #PB_Event_CloseWindow
17  EndIf
```

Voir aussi

ImageWidth() , ImageHeight()

OS Supportés

Tous

103.24 SaveImage

Syntaxe

```
Resultat = SaveImage(#Image,
    NomFichier$ [, ImagePlugin
    [, Options [,
    Profondeur]]])
```

Description

Enregistre une image sur un disque.

Arguments

#Image L'image à enregistrer.

NomFichier$ Le nom du fichier.
Si le nom de fichier ne contient pas de chemin complet alors le chemin est relatif. Voir répertoire courant .

ImagePlugin (optionnel) Le format de l'image enregistrée :

```
#PB_ImagePlugin_BMP       :
  Enregistre l'image en
  BMP (défaut).
#PB_ImagePlugin_JPEG      :
  Enregistre l'image en
  JPEG
  (UseJPEGImageEncoder()
doit être utilisé)
#PB_ImagePlugin_JPEG2000  :
  Enregistre l'image en
  JPEG2000
  (UseJPEG2000ImageEncoder()
doit être utilisé)
#PB_ImagePlugin_PNG       :
  Enregistre l'image en
  PNG
  (UsePNGImageEncoder()
doit être utilisé)
```

Options (optionnel) N'est utilisé qu'avec les plugin JPEG et JPEG 2000. Il est possible de régler la qualité d'une image à l'aide d'une valeur allant de 0 (plus mauvaise qualité) à 10 (qualité maximale), la qualité par défaut étant fixée à 7 si aucune option n'est spécifiée avec l'encodeur JPEG ou JPEG 2000. Quand une image est enregistrée en utilisant une palette (1, 4 ou 8), l'option suivante est disponible :

```
#PB_Image_FloydSteinberg:
Appliquer un filtre
Floyd-Steinberg.
```

Profondeur (optionnel) La profondeur de couleur de l'image lors de son enregistrement.
Les valeurs valides sont 1, 4, 8, 24 et 32. La valeur par défaut est la profondeur originale de l'image. Pour l'instant, seuls les encodeurs BMP et PNG supportent le mode palettisé (1, 4 ou 8 bits).

Valeur de retour

Renvoie une valeur non nulle si l'opération a réussi et zéro sinon.

Exemple

```
1  LoadImage(0,
     #PB_Compiler_Home+"Examples\Sources\Data\GeeBee2.bmp")
     ; Charge l'image depuis un
     fichier
2  SaveImage(0,
     GetTemporaryDirectory() +
     "ok.bmp") ; Enregistre
```

Voir aussi

ImageDepth() , ImagePlugin library

OS Supportés

Tous

Chapitre 104

ImagePlugin

Généralités

PureBasic supporte différents formats d'images grâce à un système de plugins dynamiques. Seul l'encodeur ou le décodeur nécessaire sera intégré au fichier exécutable, réduisant ainsi considérablement sa taille.
Par exemple, si vous avez seulement besoin du format JPEG, seule la routine concernant ce format sera utilisée.
Une autre caractéristique intéressante est l'auto-détection du format de fichier, si plusieurs décodeurs sont utilisés.
Les commandes suivantes supportent les plugins d'images : LoadImage() , CatchImage() , SaveImage() , LoadSprite() CatchSprite() , et SaveSprite() .

OS Supportés

Tous

104.1 UseGIFImageDecoder

Syntaxe

```
UseGIFImageDecoder()
```

Description

Active le support du format GIF (Graphics Interchange Format) pour les commandes CatchImage() , LoadImage() , CatchSprite() et LoadSprite() .
Les fonctions LoadImage() () et CatchImage() sont les seules a prendre en charge les GIF multi-trames (Comprenant plusieurs images).

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Tous les formats sont pris en charge, y compris le format progressif et entrelacé.

Voir aussi

LoadImage() , CatchImage() , LoadSprite() , CatchSprite()

OS Supportés

Tous

104.2 UseJPEGImageDecoder

Syntaxe

```
UseJPEGImageDecoder()
```

Description

Active le support du format JPEG (Join Picture Expert Group) pour les commandes CatchImage() , LoadImage() , CatchSprite() et LoadSprite() .

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Toutes les variantes du JPEG sont supportées, y compris la forme progressive. Le format de compression JPEG est dit 'destructif', ce qui veut dire que de l'information (plus ou moins essentielle) est perdue lors du processus de compression, permettant par la même occasion de diminuer considérablement la taille du fichier.

Voir aussi

UseJPEGImageEncoder() , LoadImage() , CatchImage() , LoadSprite() , CatchSprite()

OS Supportés

Tous

104.3 UseJPEGImageEncoder

Syntaxe

```
UseJPEGImageEncoder()
```

Description

Active le support du format JPEG (Join Picture Expert Group) pour les commandes SaveImage() et SaveSprite() .

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Le format de compression JPEG est dit 'destructif', ce qui veut dire que de l'information (plus ou moins essentielle) est perdue lors du processus de compression, permettant par la même occasion de diminuer considérablement la taille du fichier.

Voir aussi

UseJPEGImageDecoder() , SaveImage() , SaveSprite()

OS Supportés

Tous

104.4 UseJPEG2000ImageDecoder

Syntaxe

```
UseJPEG2000ImageDecoder()
```

Description

Active le support du format JPEG 2000 pour les commandes CatchImage() , LoadImage() , CatchSprite() et LoadSprite() .

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Le format de compression JPEG 2000 est dit 'destructif', ce qui veut dire que de l'information (plus ou moins essentielle) est perdue lors du processus de compression, permettant par la même occasion de diminuer considérablement la taille du fichier.

Voir aussi

UseJPEG2000ImageEncoder() , LoadImage() , CatchImage() , LoadSprite() , CatchSprite()

OS Supportés

Tous

104.5 UseJPEG2000ImageEncoder

Syntaxe

```
UseJPEG2000ImageEncoder()
```

Description

Active le support du format JPEG 2000 pour les commandes SaveImage() et SaveSprite() .

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Le format de compression JPEG 2000 est dit 'destructif', ce qui veut dire que de l'information (plus ou moins essentielle) est perdue lors du processus de compression, permettant par la même occasion de diminuer considérablement la taille du fichier.
Cet encodeur supporte les images 32-bit avec canal aplha (transparence).

Voir aussi

UseJPEG2000ImageDecoder() , SaveImage() , SaveSprite()

OS Supportés

Tous

104.6 UsePNGImageDecoder

Syntaxe

```
UsePNGImageDecoder()
```

Description

Active le support du format PNG (Portable Network Graphic) pour les commandes CatchImage() , LoadImage() , CatchSprite() et LoadSprite() .

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Toutes les variantes du PNG sont supportées, y compris la forme progressive. Le format PNG utilise un algorithme de compression non-destructif, ce qui veut dire que l'intégralité de l'information est conservée. C'est actuellement le meilleur des formats non-destructif.

Voir aussi

UsePNGImageEncoder() , LoadImage() , CatchImage() , LoadSprite() , CatchSprite()

OS Supportés

Tous

104.7 UsePNGImageEncoder

Syntaxe

```
UsePNGImageEncoder()
```

Description

Active le support du format PNG (Portable Network Graphic) pour les commandes SaveImage() et SaveSprite() .

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Le format PNG utilise un algorithme de compression non-destructif, ce qui veut dire que l'intégralité de l'information est conservée. C'est actuellement le meilleur des formats non-destructif.
Cet encodeur supporte les images 32-bit avec canal aplha (transparence).

Voir aussi

UsePNGImageDecoder() , SaveImage() , SaveSprite()

OS Supportés

Tous

104.8 UseTGAImageDecoder

Syntaxe

```
UseTGAImageDecoder()
```

Description

Active le support du format TGA (Targa) pour les commandes CatchImage() , LoadImage() , CatchSprite() et LoadSprite() .

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Le format TGA utilise un algorithme de compression non-destructif, ce qui veut dire que l'intégralité de l'information est conservée. La taille des images TGA est souvent très élévée car l'algorithme de compression (quand il est utilisé) est peu performant (RLE).
Si une couche alpha (transparence) est présente dans l'image, elle ne sera pas prise en compte (pour l'instant).

Voir aussi

LoadImage() , CatchImage() , LoadSprite() , CatchSprite()

OS Supportés

Tous

104.9 UseTIFFImageDecoder

Syntaxe

```
UseTIFFImageDecoder()
```

Description

Active le support du format TIFF pour les commandes CatchImage() , LoadImage() , CatchSprite() et LoadSprite() .

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Le format TIFF est très complexe et regroupe différents algorithmes de compressions tels que le JPEG, LZW, etc... Toutes les variantes du TIFF sont supportées. La taille des images est relativement importante.

Voir aussi

LoadImage() , CatchImage() , LoadSprite() , CatchSprite()

OS Supportés

Tous

Chapitre 105

Joint

Généralités

Les articulations, comme les charnières, les glissières, sont utilisées pour définir un lien entre deux entités .
Ce lien permet d'obtenir un comportement automatique lorsque l'une des entités est mise en mouvement.
Elles peuvent être utilisées pour simuler une interaction de façon réaliste entre deux objets, comme par exemple : Une porte qui tourne sur ses gonds, un train qui glisse sur ses rails etc.
InitEngine3D() doit être appelé avec succès avant d'utiliser les fonctions communes.

OS Supportés

Tous

105.1 EnableHingeJointAngularMotor

Syntaxe

```
EnableHingeJointAngularMotor(#Charniere,
    Activer, Vitesse,
    ImpulsionMax)
```

Description

Active le moteur angulaire d'une charnière.

Arguments

#**Charniere** La charnière à utiliser.

Activer Active ou désactive le moteur angulaire de la charnière.

```
#True : Le moteur est
  activé,
#False: Le moteur est
  désactivé.
```

Vitesse La vitesse que le moteur doit produire.

ImpulsionMax L'impulsion maximale que le moteur doit produire.

Valeur de retour

Aucune.

Voir aussi

HingeJoint()

OS Supportés

Tous

105.2 HingeJointMotorTarget

Syntaxe

```
HingeJointMotorTarget(#Charniere,
    Angle, Vitesse)
```

Description

Définit le moteur d'une charnière.

Arguments

#**Charniere** La charnière à utiliser.

Angle L'angle (en degrés) que le moteur doit atteindre.

Vitesse La vitesse que le moteur doit produire.

Valeur de retour

Aucune.

Remarques

La charnière doit être crée au préalable avec HingeJoint()

Voir aussi

HingeJoint()

OS Supportés

Tous

105.3 FreeJoint

Syntaxe

```
FreeJoint(#Charniere)
```

Description

Libère une charnière et toute la mémoire associée.

Arguments

#Charniere la charnière à libérer.
Si #PB_All est spécifié, toutes les charnières restantes seront libérées.

Valeur de retour

Aucune.

Remarques

la charnière ne doit pas être utilisée après l'appel de cette fonction (ex : en utilisant son numéro avec les autres fonctions de cette bibliothèque).
Toutes les charnières restantes sont automatiquement libérées lorsque le programme se termine.

Voir aussi

PointJoint() , HingeJoint() , ConeTwistJoint() , SliderJoint()

OS Supportés

Tous

105.4 IsJoint

Syntaxe

```
Resultat = IsJoint(#Charniere)
```

Description

Teste si une charnière est valide et correctement initialisée.

Arguments

#Charniere La charnière à tester.

Valeur de retour

Renvoie une valeur non nulle si la charnière est valide, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est une bonne façon de s'assurer qu'une charnière est prête à l'emploi.

OS Supportés

Tous

105.5 GenericJoint

Syntaxe

```
Resultat =
    GenericJoint(#Charniere,
    EntiteID, TransformX,
    TransformY, TransformZ,
    EntiteID2, TransformX2,
    TransformY2, TransformZ2)
```

Description

Crée une nouvelle charnière basée sur un ou deux points.

Arguments

#Charniere Le numéro d'identification de la nouvelle charnière.
PB_Any # peut être utilisé pour générer automatiquement ce numéro.

EntiteID Le numéro d'identification de l'entité associée à la charnière.
Pour obtenir un identifiant valide, utiliser EntityID() .

TransformX, TransformY, TransformZ La transformation à appliquer sur la première entité.

EntiteID2 (optionnel) La deuxième entité associée à la charnière.
Si ce paramètre n'est pas spécifié, une seule charnière est créée entre le point de pivotement et l'entité.
Si ce paramètre est spécifié, la première entité est fixée à la seconde entité.
Pour obtenir un identifiant valide, utiliser EntityID() .

TransformX2, TransformY2, TransformZ2 (optionnel) La transformation à appliquer sur la seconde entité.

Valeur de retour

Renvoie zéro si la charnière ne peut être créée.

Si #PB_Any est utilisé à la place de '#Charniere', le nouveau numéro de charnière sera renvoyé dans 'Resultat'.

Remarques

GetJointAttribute() et SetJointAttribute() peuvent être utilisées avec les attributs suivants pour modifier le comportement de la charnière :

```
#PB_Joint_EnableSpring:
 Activation du ressort
 #True = ON, #False = OFF
 (par défaut)
#PB_Joint_Stiffness   :
 Raideur entre 1 et 10000
#PB_Joint_Damping     :
 Amortissement entre 0 et 1
 (0 = très fort
 amortissement).
#PB_Joint_Position    :
 Position de l'axe
#PB_Joint_NoLimit     :
 Position libre de l'axe
#PB_Joint_LowerLimit  :
 Limite inférieure
#PB_Joint_UpperLimit  :
 Limite supérieure
```

Voir aussi

FreeJoint() , GetJointAttribute() , SetJointAttribute()

OS Supportés

Tous

105.6 PointJoint

Syntaxe

```
Resultat =
    PointJoint(#Charniere,
    EntiteID, PivotX, PivotY,
    PivotZ, [EntiteID2,
    PivotX2, PivotY2, PivotZ2])
```

Description

Crée une nouvelle charnière basée sur un ou deux points.

Arguments

#Charniere Le numéro d'identification de la nouvelle charnière.

PB_Any # peut être utilisé pour générer automatiquement ce numéro.

EntiteID Le numéro d'identification de l'entité associée à la charnière.
Pour obtenir un identifiant valide, utiliser EntityID() .

PivotX, PivotY, PivotZ La coordonnée du point de pivot de la charnière, par rapport au centre de l'entité.

EntiteID2 (optionnel) La deuxième entité associée à la charnière.
Si ce paramètre n'est pas spécifié, une seule charnière est créée entre le point de pivotement et l'entité.
Si ce paramètre est spécifié, la première entité est fixée à la seconde entité.
Pour obtenir un identifiant valide, utiliser EntityID() .

PivotX2, PivotY2, PivotZ2 (optionnel) La coordonnée du point de pivot de la charnière, par rapport au centre de la seconde entité.

Valeur de retour

Renvoie zéro si la charnière ne peut être créée.
Si #PB_Any est utilisé à la place de '#Charniere', le nouveau numéro de charnière sera renvoyé dans 'Resultat'.

Remarques

GetJointAttribute() et SetJointAttribute() peuvent être utilisées avec les attributs suivants pour modifier le comportement de la charnière :

```
#PB_PointJoint_Tau     :
 Valeur Tau de la charnière
#PB_PointJoint_Damping:
 Valeur d'amortissement de
 la charnière
```

Voir aussi

FreeJoint() , GetJointAttribute() , SetJointAttribute()

OS Supportés

Tous

105.7 HingeJoint

Syntaxe

```
Resultat =
    HingeJoint(#Charniere,
    EntiteID, PivotX, PivotY,
    PivotZ, AxeX, AxeY, AxeZ,
    EntiteID2, PivotX2,
    PivotY2, PivotZ2, AxeX2,
    AxeY2, AxeZ2)
```

Description

Crée une nouvelle charnière entre deux entités.

Arguments

#Charniere Le numéro d'identification de la nouvelle charnière.
PB_Any # peut être utilisé pour générer automatiquement ce numéro.

EntiteID La première entité associée à la charnière.
Pour obtenir un identifiant valide, utiliser EntityID() .

PivotX, PivotY, PivotZ La coordonnée du point de pivot de la charnière, par rapport au centre de l'entité.

AxeX, AxeY, AxeZ L'orientation de l'axe de la charnière.

EntiteID2 La deuxième entité associée à la charnière.
Pour obtenir un identifiant valide, utiliser EntityID() .

PivotX2, PivotY2, PivotZ2 La coordonnée du second point de pivotement de la charnière, par rapport au centre de la seconde entité.

AxeX2, AxeY2, AxeZ2 L'orientation de l'axe de la charnière.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any est utilisé à la place de '#Charniere', le nouveau numéro sera renvoyé dans 'Resultat'.

Remarques

Une charnière peut être utilisée pour simuler une porte, le mouvement d'un pont, etc.
GetJointAttribute() et SetJointAttribute() peuvent être utilisés avec les attributs suivants pour modifier le comportement de la charnière :

```
#PB_HingeJoint_LowerLimit :
 Limite inférieure de la
 charnière
#PB_HingeJoint_UpperLimit :
 Limite supérieure de la
 charnière
```

Voir aussi

FreeJoint() , GetJointAttribute() , SetJointAttribute() , EnableHingeJointAngularMotor()

OS Supportés

Tous

105.8 ConeTwistJoint

Syntaxe

```
Resultat =
    ConeTwistJoint(#Charniere,
    EntiteID, TransformationX,
    TransformationY,
    TransformationZ,
    EntiteID2,
    TransformationX2,
    TransformationY2,
    TransformationZ2)
```

Description

Crée une nouvelle charnière de type cône de torsion (articulation) entre deux entités.

Arguments

#**Charniere** Le numéro d'identification de la nouvelle charnière.
PB_Any # peut être utilisé pour générer automatiquement ce numéro.

EntiteID La première entité associée à la charnière.
Pour obtenir un identifiant valide, utiliser EntityID() .

TransformX, TransformY, TransformZ La valeur de la transformation à appliquer sur la première entité.

EntiteID2 La deuxième entité associée à la charnière.
Pour obtenir un identifiant valide, utiliser EntityID() .

TransformX2, TransformY2, TransformZ2 La valeur de la transformation à appliquer sur la seconde entité.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any est utilisé à la place de '#Charniere', le nouveau numéro sera renvoyé dans 'Resultat'.

Remarques

Un cône de torsion peut être utilisé pour attacher les bras ou les jambes au corps, etc. GetJointAttribute() et SetJointAttribute() peuvent être utilisés avec les attributs suivants pour modifier le comportement de la charnière :

```
#PB_ConeTwistJoint_SwingSpan
 : Première amplitude de
 balancement de la
 charnière.
#PB_ConeTwistJoint_SwingSpan2:
 Seconde amplitude de
 balancement de la
 charnière.
#PB_ConeTwistJoint_TwistSpan
 : Amplitude de torsion de
 la charnière.
```

Voir aussi

FreeJoint() , GetJointAttribute() , SetJointAttribute()

OS Supportés

Tous

105.9 SliderJoint

Syntaxe

```
Resultat =
   SliderJoint(#Charniere,
   EntiteID, TransformationX,
   TransformationY,
   TransformationZ,
   EntiteID2,
   TransformationX2,
   TransformationY2,
   TransformationZ2)
```

Description

Crée une nouvelle charnière de type glissière ou curseur entre deux entités.

Arguments

#Charniere Le numéro d'identification de la nouvelle charnière.
PB_Any # peut être utilisé pour générer automatiquement ce numéro.

EntiteID La première entité associée à la charnière.
Pour obtenir un identifiant valide, utiliser EntityID() .

TransformX, TransformY, TransformZ La valeur de la transformation de la première entité.

EntiteID2 La deuxième entité associée à la charnière.
Pour obtenir un identifiant valide, utiliser EntityID() .

TransformX2, TransformY2, TransformZ2 La valeur de la transformation de la seconde entité.

Valeur de retour

Renvoie zéro si la charnière ne peut être créée.
Si #PB_Any est utilisé à la place de '#Charniere', le nouveau numéro sera renvoyé dans 'Resultat'.

Remarques

La glissière peut être utilisée pour déplacer une entité avec contrainte sur une surface plane, etc.
GetJointAttribute() et SetJointAttribute() peuvent être utilisés avec les attributs suivants pour modifier le comportement de la charnière :

```
#PB_SliderJoint_LowerLimit
  : Limite inférieure de la
  charnière.
#PB_SliderJoint_UpperLimit
  : Limite supérieure de la
  charnière.
```

Voir aussi

FreeJoint() , GetJointAttribute() , SetJointAttribute()

OS Supportés

Tous

105.10 GetJointAttribute

Syntaxe

```
Resultat =
    GetJointAttribute(#Charniere,
    Attribut)
```

Description

Obtenir la valeur d'un attribut d'une charnière.

Arguments

#**Charniere** La charnière à utiliser.

Attribut L'attribut à obtenir.

Valeur de retour

Renvoie la valeur de l'attribut spécifié ou zéro si la charnière ne prend pas en charge l'attribut.

Remarques

Cette fonction est disponible pour toutes les charnières qui supportent les attributs.
Voir les commandes individuelles pour les attributs communs :
- PointJoint()
- ConeTwistJoint()
- HingeJoint()
- SliderJoint()

Voir aussi

SetJointAttribute()

OS Supportés

Tous

105.11 SetJointAttribute

Syntaxe

```
SetJointAttribute(#Charniere,
    Attribut, Valeur [, Axe])
```

Description

Régle la valeur de l'attribut d'une charnière.

Arguments

#**Charniere** La charnière à utiliser.

Attribut L'attribut à définir.

Valeur Valeur de l'attribut à définir.

Axe (optionnel) L'axe à utiliser.
A n'utiliser qu'avec GenericJoint() dont les axes sont indexés comme cela :

```
0, 1 et 2: translation
 suivant les axes (x, y,
 z)
3, 4 et 5: rotation
 suivant les axes (x, y,
 z)
```

Valeur de retour

Aucune.

Remarques

Cette fonction est disponible pour toutes les charnières qui supportent les attributs.
Voir les commandes individuelles pour les attributs communs :
- GenericJoint()
- PointJoint()
- ConeTwistJoint()
- HingeJoint()
- SliderJoint()

Voir aussi

GetJointAttribute()

OS Supportés

Tous

Chapitre 106

Joystick

Généralités

PureBasic permet la gestion complète des joysticks connectés à l'ordinateur.
Tous les joysticks et les pads sont acceptés, même les manettes multi-boutons, triggers, etc.

Remarques

Sur Windows l'interface est assurée par la technologie DirectX et sur Linux avec SDL.

OS Supportés

Tous

106.1 InitJoystick

Syntaxe

```
Resultat = InitJoystick()
```

Description

Initialise l'environnement de gestion des joysticks.

Arguments

Aucun.

Valeur de retour

Renvoie le nombre de joysticks disponibles.

Remarques

Vous devez appeler cette fonction avant toute autre fonction de cette bibliothèque.

Voir aussi

ExamineJoystick()

OS Supportés

Tous

106.2 ExamineJoystick

Syntaxe

```
Resultat =
    ExamineJoystick(#Joystick)
```

Description

Met à jour l'état d'un joystick.

Arguments

#**Joystick** Le numéro du joystick à utiliser.
Le premier numéro de joystick commence à 0.

Valeur de retour

Renvoie une valeur non nulle si le joystick est prêt.

Remarques

Met à jour l'état actuel d'un joystick qui pourra être interrogé dans un second temps avec les commandes JoystickAxisX() , JoystickAxisY() , JoystickAxisZ() et JoystickButton() .

Voir aussi

JoystickButton() , JoystickAxisX() , JoystickAxisY() , JoystickAxisZ() .

OS Supportés

Tous

106.3 JoystickAxisX

Syntaxe

```
Resultat =
    JoystickAxisX(#Joystick [,
    Pad [, Mode]]))
```

Description

Renvoie l'état de l'axe X d'un joystick.

Arguments

#**Joystick** Le numéro du joystick à utiliser.
Le premier joystick commence à 0.

Pad (optionnel) Le pad à utiliser si la manette dispose de plusieurs pads.
Le premier pad a pour indice 0.

Mode (optionnel) Peut être l'une des valeurs suivante :

```
#PB_Absolute: -1 :
 Mouvement vers la gauche
               0 : Aucun
 mouvement
               1 :
 Mouvement vers la droite
 (par défaut)
#PB_Relative: Renvoie une
 valeur entre -1000
 (gauche) et 1000 (droit).
             Si la
 manette de jeu ne
 supporte pas le
 mouvement relatif, le
 résultat sera -1000, 0
 ou 1000.
```

Valeur de retour

Renvoie la valeur de l'axe X qui dépend du mode utilisé.

Remarques

La fonction ExamineJoystick() doit être appelée avant cette fonction pour mettre à jour l'état du joystick.

Voir aussi

ExamineJoystick() , InitJoystick() , JoystickAxisY() , JoystickAxisZ()

OS Supportés

Tous

106.4 JoystickAxisY

Syntaxe

```
Resultat =
   JoystickAxisY(#Joystick [,
   Pad [, Mode]]))
```

Description

Renvoie l'état de l'axe Y d'un joystick.

Arguments

#**Joystick** Le numéro du joystick à utiliser.
Le premier joystick commence à 0.

Pad (optionnel) Le pad à utiliser si la manette dispose de plusieurs pads.
Le premier pad a pour indice 0.

Mode (optionnel) Peut être l'une des valeurs suivantes :

```
#PB_Absolute: -1 :
 Mouvement vers le haut
               0 : Aucun
 mouvement
               1 :
 Mouvement vers le bas
 (par défaut)
#PB_Relative: Renvoie une
 valeur entre -1000
 (haut) et 1000 (bas).
             Si la
 manette de jeu ne
 supporte pas le
 mouvement relatif, le
 résultat sera -1000, 0
 ou 1000.
```

Valeur de retour

Renvoie la valeur de l'axe Y qui dépend du mode utilisé.

Remarques

La fonction ExamineJoystick() doit être appelée avant cette fonction pour mettre à jour l'état du joystick.

Voir aussi

ExamineJoystick() , InitJoystick() , JoystickAxisX() , JoystickAxisZ()

OS Supportés

Tous

106.5 JoystickAxisZ

Syntaxe

```
Resultat =
   JoystickAxisZ(#Joystick [,
   Pad [, Mode]]))
```

Description

Renvoie l'état de l'axe Z d'un joystick.

Arguments

#**Joystick** Le numéro du joystick à utiliser.
Le premier joystick commence à 0.

Pad (optionnel) Le pad à utiliser si la manette dispose de plusieurs pads.
Le premier pad a pour indice 0.

Mode (optionnel) Peut être l'une des valeurs suivantes :

```
#PB_Absolute: -1 :
 Mouvement vers l'arrière
               0 : Aucun
 mouvement
               1 :
 Mouvement vers l'avant
 (par défaut)
#PB_Relative: Renvoie une
 valeur entre -1000 et
 1000.
             Si la
 manette de jeu ne
 supporte pas le
 mouvement relatif, le
 résultat sera -1000, 0
 ou 1000.
```

Valeur de retour

Renvoie la valeur de l'axe Z qui dépend du mode utilisé.

Remarques

La fonction ExamineJoystick() doit être appelée avant cette fonction pour mettre à jour l'état du joystick.

Voir aussi

ExamineJoystick() , InitJoystick() , JoystickAxisX() , JoystickAxisY()

OS Supportés

Tous

106.6 JoystickName

Syntaxe

```
Resultat\$ =
    JoystickName(#Joystick)
```

Description

Renvoie le nom d'un joystick.

Arguments

#**Joystick** Le numéro du joystick à utiliser.
Le premier joystick commence à 0.

Valeur de retour

Renvoie le nom de la manette.

Remarques

Cette fonction peut être utile si plusieurs manettes sont connectées, afin d'identifier la bonne.

Voir aussi

InitJoystick()

OS Supportés

Tous

106.7 JoystickButton

Syntaxe

```
Resultat =
    JoystickButton(#Joystick,
    Bouton)
```

Description

Renvoie l'état d'un bouton d'un joystick.

Arguments

#**Joystick** Le numéro du joystick.
Le premier joystick commence à 0.

Bouton Le bouton à tester.
L'index du premier bouton est 1.

Valeur de retour

Renvoie une valeur non nulle si le bouton est pressé, zéro sinon.

Remarques

La fonction ExamineJoystick() doit être appelée avant cette fonction pour mettre à jour l'état des boutons.
Un nombre quelconque de boutons peuvent être pressés simultanément.

Voir aussi

ExamineJoystick() , InitJoystick()

OS Supportés

Tous

Chapitre 107

Json

Généralités

La bibliothèque JSON fournit des fonctions pour parcourir (parse), créer et modifier des données au format JSON.
JSON (JavaScript Object Notation) est un format léger d'échange de données pris en charge par de nombreux langages de programmation. Vous trouverez une introduction à JSON ici.
Cette bibliothèque comprend et produit le format JSON tel que défini par RFC-7159.

OS Supportés

Tous

107.1 AddJSONElement

Syntaxe

```
Resultat =
    AddJSONElement(JSONValeur
    [, Index])
```

Description

Ajoute un nouvel élément à un tableau JSON de type #PB_JSON_Array.

Arguments

JSONValeur La valeur JSON à ajouter. Doit être du type #PB_JSON_Array.

Index (optionnel) L'index auquel la nouvelle valeur est introduite dans la matrice.
Si l'index est en dehors des limites du tableau, la nouvelle valeur est introduite soit au début (avec un Index <0) soit à la fin du tableau.
Si ce paramètre n'est pas spécifié, la nouvelle valeur est ajoutée à la fin du tableau.

Valeur de retour

Renvoie l'adresse de la valeur JSON ajoutée.
La valeur nouvellement ajoutée est initialement de type #PB_JSON_Null.

Exemple

```
If CreateJSON(0)

  ; transformation d'une collection de données vide en un tableau
  Tableau = SetJSONArray(JSONValue(0))

  ; ajout de 5 nombres entiers à la fin du tableau
  For i = 1 To 5
    AdresseNombre = AddJSONElement(Tableau)
    SetJSONInteger(AdresseNombre, i)
  Next i

  ; insertion de "Hello" à l'index 1
  AdresseTexte = AddJSONElement(Tableau, 1)
  SetJSONString(AdresseTexte, "Hello")

  Debug ComposeJSON(0)
EndIf

; Résultat : [1, "Hello", 2, 3, 4, 5]
; Remarque : Un tableau est toujours entre crochets "[" "]"
```

Voir aussi

SetJSONArray() , RemoveJSONElement() , ResizeJSONElements() , ClearJSONElements() , GetJSONElement() , JSONArraySize() , JSONType()

OS Supportés

Tous

107.2 AddJSONMember

Syntaxe

```
Resultat =
    AddJSONMember(JSONValeur,
    Cle$)
```

Description

Ajoute un nouveau membre (couple nom/valeur) à un objet JSON de type #PB_JSON_Object.

Arguments

JSONValeur La valeur JSON à ajouter.
Doit être du type #PB_JSON_Object.

Cle$ La clé du nouveau membre.
Si un membre avec la même clé existe dans l'objet, il sera remplacé.

Valeur de retour

Renvoie l'adresse du membre ajouté.
La valeur nouvellement ajoutée est initialement de type #PB_JSON_Null.

Remarques

Si un membre avec la clé spécifiée existe déjà, il sera remplacé.

Exemple

```
1   If CreateJSON(0)
2     Objet =
    SetJSONObject(JSONValue(0))
3
4     AdressePrenom =
    AddJSONMember(Objet,
    "Prénom")
5     SetJSONString(AdressePrenom,
    "Jean")
6
7     AdresseNom =
    AddJSONMember(Objet, "Nom")
8     SetJSONString(AdresseNom,
    "Dupond")
9
10    Debug ComposeJSON(0)
11  EndIf
12
13    ; Résultat : {"Nom":
    "Dupond", "Prénom": "Jean"}
14    ; Remarque : Un objet
    JSON, est un ensemble de
    couples nom/valeur non
    ordonnés.
15    ;            Un objet
    commence par { (accolade
    gauche) et se termine par
    } (accolade droite).
```

```
16    ;            Chaque nom
      est suivi de :
      (deux-points) et les
      couples nom/valeur sont
      séparés par une ,
      (virgule).
```

Voir aussi

SetJSONObject() , RemoveJSONMember() , ClearJSONMembers() , GetJSONMember() , ExamineJSONMembers() , JSONObjectSize() , JSONType()

OS Supportés

Tous

107.3 CatchJSON

Syntaxe

```
Resultat = CatchJSON(#JSON,
    *Memoire, Taille [,
    Option])
```

Description

Parcourt (parse) les données JSON d'une mémoire tampon.

Arguments

#JSON Numéro d'identification du nouveau JSON.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

***Memoire** Un emplacement de mémoire accessible en lecture.

Taille Longueur (en octets) de l'emplacement en mémoire.

Option (optionnel)
#PB_JSON_NoCase : Les données JSON seront parcourues sans tenir compte de la casse.
La valeur par défaut est d'être sensible à la casse.

Valeur de retour

Renvoie une valeur non nulle si les données JSON ont été parcourues correctement, zéro sinon.
Si #PB_Any a été utilisé comme paramètre #JSON alors le nombre généré est renvoyé en cas de succès.

Remarques

Le contenu de la mémoire tampon doit être codé au format UTF-8.
La fonction JSONValue() peut être utilisée pour accéder à la valeur JSON après l'analyse.
En cas d'erreur, les fonctions JSONErrorMessage() , JSONErrorLine() et JSONErrorPosition() peuvent être utilisées pour obtenir plus d'informations sur l'erreur.
Pour parcourir les données JSON directement à partir d'une chaîne, la fonction ParseJSON() peut être utilisée à la place.
JSON est un format de données sensible à la casse. Cependant, dans certaines situations, telles que l'utilisation de structures de désérialisation avec ExtractJSONStructure() ou des commandes similaires, il peut être utile de traiter des objets JSON de façon insensible à la casse. L'option #PB_JSON_NoCase permet de traiter les clés de chaque membre de l'objet de façon insensible à la casse.

Voir aussi

CreateJSON() , ParseJSON() , LoadJSON() , JSONValue() , FreeJSON() , JSONErrorMessage() , JSONErrorLine() , JSONErrorPosition() , ExportJSON()

OS Supportés

Tous

107.4 ClearJSONElements

Syntaxe

```
ClearJSONElements(JSONValeur)
```

Description

Efface tous les éléments d'un tableau JSON de type #PB_JSON_Array.

Arguments

JSONValeur La valeur JSON.
Doit être du type #PB_JSON_Array.

Valeur de retour

Aucune.

Exemple

```
ParseJSON(0, "[1, 2, 3, 4]")

; efface le tableau et
  ajoute un nouveau texte
ClearJSONElements(JSONValue(0))
SetJSONString(AddJSONElement(JSONValue(0)),
  "Hello")

Debug ComposeJSON(0)

; Résultat : ["Hello"]
```

Voir aussi

SetJSONArray() , AddJSONElement() , RemoveJSONElement() , ResizeJSONElements() , GetJSONElement() , JSONArraySize() , JSONType()

OS Supportés

Tous

107.5 ClearJSONMembers

Syntaxe

```
ClearJSONMembers(JSONValeur)
```

Description

Efface tous les membres d'un objet JSON de type #PB_JSON_Object.

Arguments

JSONValeur La valeur JSON.
Doit être du type #PB_JSON_Object.

Valeur de retour

Aucune.

Exemple

```
Donnees$ = "{ " + Chr(34) +
  "x" + Chr(34) + ": 10, " +
  Chr(34) + "y" + Chr(34) +
  ": 20 }"
Debug Donnees$
ParseJSON(0, Donnees$)

```

```
5   ; efface les membres existant puis en ajoute un nouveau
6   ClearJSONMembers(JSONValue(0))
7   SetJSONString(AddJSONMember(JSONValue(0), "Hello"), "le monde")
8
9   Debug ComposeJSON(0)
10
11  ; Résultat : { "x": 10, "y": 20 }
12  ;            {"Hello": "le monde"}
```

Voir aussi

SetJSONObject() , AddJSONMember() , RemoveJSONMember() , GetJSONMember() , ExamineJSONMembers() , JSONObjectSize() , JSONType()

OS Supportés

Tous

107.6 ComposeJSON

Syntaxe

```
Resultat\$ =
    ComposeJSON(#JSON [, Option])
```

Description

Compose des données JSON dans une chaîne de caractères.

Arguments

#JSON Le JSON à composer.

Option (optionnel) Avec la valeur #PB_JSON_PrettyPrint, la chaîne de caractères composée contiendra les sauts de ligne et les espaces supplémentaires pour une meilleure lisibilité. L'espace blanc supplémentaire n'est pas reconnu par le format JSON. Le résultat aura le même sens pour un lecteur JSON avec ou sans cette option.

Valeur de retour

Les données JSON dans une chaîne de caractères.

Remarques

Une chaîne de caractères peut être parcourue en données JSON en utilisant la fonction ParseJSON() .
La chaîne de caractères produite a le format des chaînes dans l'exécutable (Ascii ou Unicode). JSON est généralement encodé en UTF-8, donc lors de l'écriture de la chaîne dans un fichier par exemple ou lors d'un envoi à une autre application, il est conseillé de convertir la chaîne en UTF-8 avant de le faire.

Exemple

```
1   If CreateJSON(0)
2     ObjetPersonne = SetJSONObject(JSONValue(0))
3     SetJSONString(AddJSONMember(ObjetPersonne, "Prénom"), "Jean")
4     SetJSONString(AddJSONMember(ObjetPersonne, "Nom"), "Dupond")
5     SetJSONInteger(AddJSONMember(ObjetPersonne, "Age"), 42)
6
7     Debug ComposeJSON(0, #PB_JSON_PrettyPrint)
8   EndIf
9
10    ; Résultat : {
11    ;              "Nom"    : "Dupond",
12    ;              "Prénom": "Jean",
13    ;              "Age"    : 42
14    ;            }
```

Voir aussi

SaveJSON() , ExportJSON() , ExportJSONSize() , ParseJSON()

OS Supportés

Tous

107.7 CreateJSON

Syntaxe

```
Resultat = CreateJSON(#JSON
    [, Option])
```

Description

Crée une nouvelle collection de données JSON vide.

Arguments

#JSON Le numéro d'identification de la nouvelle collection JSON.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Option (optionnel)
#PB_JSON_NoCase : Les données JSON seront traitées de façon insensible à la casse.
La valeur par défaut est d'être sensible à la casse.

Valeur de retour

Renvoie une valeur non nulle si la collection JSON a été créée, zéro sinon.
Si #PB_Any a été utilisé pour paramètre #JSON alors le nombre généré est renvoyé en cas de succès.

Remarques

A la création, les données contiennent une valeur JSON de type #PB_JSON_Null. La fonction JSONValue() peut être utilisée pour changer ce type.
JSON est un format de données sensible à la casse. Cependant, dans certaines situations, telles que l'utilisation de structures de désérialisation avec ExtractJSONStructure() ou des commandes similaires, il peut être utile de traiter des objets JSON de façon insensible à la casse. L'option #PB_JSON_NoCase permet de traiter les clés de chaque membre de l'objet de façon insensible à la casse.

Exemple

```
1   If CreateJSON(0)
2     ObjetPersonne = SetJSONObject(JSONValue(0))
3     SetJSONString(AddJSONMember(ObjetPersonne, "Prénom"), "Jean")
4     SetJSONString(AddJSONMember(ObjetPersonne, "Nom"), "Dupond")
5     SetJSONInteger(AddJSONMember(ObjetPersonne, "Age"), 42)
6
7     Debug ComposeJSON(0, #PB_JSON_PrettyPrint)
8   EndIf
9
10    ; Résultat : {
11    ;              "Nom"    : "Dupond",
12    ;              "Prénom":  "Jean",
```

```
13    ;                "Age"    : 42
14    ;              }
```

Voir aussi

CatchJSON() , LoadJSON() , ParseJSON() , JSONValue() , FreeJSON()

OS Supportés

Tous

107.8 ExamineJSONMembers

Syntaxe

```
Resultat =
    ExamineJSONMembers(JSONValeur)
```

Description

Commence à examiner les membres d'une valeur JSON de type #PB_JSON_Object.

Arguments

JSONValeur La valeur JSON à examiner. La valeur doit être de type #PB_JSON_Object.

Valeur de retour

Renvoie une valeur non nulle si l'objet peut être énuméré, zéro sinon.

Remarques

Les membres qui ont été examinés avec NextJSONMember() , JSONMemberKey() et JSONMemberValue() renvoient une valeur non nulle si les données JSON ont été parcourues correctement, zéro sinon.

Exemple

```
1   Donnees$ = "{ " + Chr(34) +
      "x" + Chr(34) + ": 10, " +
2                     Chr(34) +
      "y" + Chr(34) + ": 20, " +
3                     Chr(34) +
      "z" + Chr(34) + ": 30 }"
4
5   ParseJSON(0, Donnees$)
6   ObjetValeur = JSONValue(0)
7
8   If
      ExamineJSONMembers(ObjetValeur)
```

```
9       While
        NextJSONMember(ObjetValeur)
10        Debug
        JSONMemberKey(ObjetValeur)
        + " = " +
        GetJSONInteger(JSONMemberValue(ObjetValeur))
11      Wend
12    EndIf
13
14    ; Résultat :    x = 10
15    ;               y = 20
16    ;               z = 30
```

Voir aussi

NextJSONMember() , JSONMemberKey() , JSONMemberValue() , GetJSONMember() , SetJSONObject() , JSONType()

OS Supportés

Tous

107.9 ExportJSON

Syntaxe

```
Resultat = ExportJSON(#JSON,
    *Memoire, Taille [,
    Option])
```

Description

Exporte les données JSON dans un emplacement mémoire.

Arguments

#JSON Le JSON à exporter.

***Memoire** Un emplacement mémoire accessible en écriture.

Taille La taille de l'emplacement mémoire. Si la taille n'est pas assez grande pour contenir toutes les données JSON, la fonction remplira l'emplacement de mémoire avec des données, mais ensuite elle renverra un code erreur.
La fonction ExportJSONSize() peut être utilisée pour déterminer la taille nécessaire.

Option (optionnel) Avec la valeur #PB_JSON_PrettyPrint, la chaîne de caractères composée contiendra les sauts de ligne et les espaces supplémentaires pour une meilleure lisibilité. L'espace blanc supplémentaire n'est pas reconnu par le format JSON. Le résultat aura le

même sens pour un lecteur JSON avec ou sans cette option.

Valeur de retour

Renvoie le nombre d'octets écrits dans l'emplacement mémoire en cas de succès, zéro sinon.

Remarques

Les données JSON sont exportées au format UTF-8.

Voir aussi

ExportJSONSize() , ComposeJSON() , SaveJSON() , CatchJSON()

OS Supportés

Tous

107.10 ExportJSONSize

Syntaxe

```
Resultat =
    ExportJSONSize(#JSON [,
    Options])
```

Description

Renvoie la taille en octets nécessaire pour exporter avec succès les données JSON dans une mémoire tampon avec les options spécifiées.

Arguments

#JSON Le JSON à exporter.

Option (optionnel) Les mêmes options que celles utilisées avec ExportJSON() . La seule valeur autorisée est #PB_JSON_PrettyPrint.

Valeur de retour

Le nombre d'octets nécessaire pour exporter les données au format JSON.

Voir aussi

ExportJSON() , ComposeJSON() , SaveJSON()

OS Supportés

Tous

107.11 ExtractJSONArray

Syntaxe

```
ExtractJSONArray(JSONValeur,
    Tableau())
```

Description

Extrait des éléments de la valeur JSON de type #PB_JSON_Array dans le Tableau() spécifié.

Arguments

JSONValeur La valeur JSON.
La valeur doit être de type #PB_JSON_Array.

Tableau() Le tableau à remplir avec les éléments de JSON.
Le tableau sera redimensionné pour avoir la même taille que la valeur JSON. Tout le contenu précédent sera perdu.

Valeur de retour

Aucune.

Remarques

L'extraction est réalisée de manière récursive si le tableau a un type structure. Si la valeur JSON contient des éléments qui n'ont pas le bon type pour remplir le Tableau(), ils seront ignorés et l'élément de tableau correspondant sera laissé vide.
Si Tableau() a plus d'une dimension alors les données JSON sont supposées être un tableau de tableaux pour représenter les données multidimensionnelles. Voir l'exemple ci-dessous pour plus de détails.

Exemple

```
1  ParseJSON(0, "[1, 3, 5, 7,
     9]")
2
3  Dim a(0)
4  ExtractJSONArray(JSONValue(0),
     a())
5
6  For i = 0 To ArraySize(a())
7    Debug a(i)
8  Next i
```

```
9
10    ; Résultat : 1
11    ;            3
12    ;            5
13    ;            7
14    ;            9
```

Exemple

```
1   ParseJSON(0, "[[0, 1, 2],
      [3, 4, 5], [6, 7, 8]]")
2
3   Dim a(0, 0)
4   ExtractJSONArray(JSONValue(0),
      a())
5
6   For x = 0 To 2
7     For y = 0 To 2
8       Debug a(x, y)
9     Next y
10  Next x
11
12  ; Résultat : 0
13  ;            1
14  ;            2
15  ;            3
16  ;            4
17  ;            5
18  ;            6
19  ;            7
20  ;            8
```

Voir aussi

ExtractJSONList() , ExtractJSONMap() , ExtractJSONStructure() , InsertJSONArray() , InsertJSONList() , InsertJSONMap() , InsertJSONStructure() , SetJSONArray() , JSONType()

OS Supportés

Tous

107.12 ExtractJSONList

Syntaxe

```
ExtractJSONList(JSONValeur,
    Liste())
```

Description

Extrait des éléments de la valeur JSON de type #PB_JSON_Array dans la Liste() spécifiée. La liste sera redimensionnée pour contenir le nombre d'éléments contenus dans la valeur JSON.

Arguments

JSONValeur La valeur JSON.
La valeur doit être de type #PB_JSON_Array.

Liste() La liste à remplir avec les éléments de JSON.
La liste sera redimensionnée pour avoir la même taille que la valeur JSON.
Tout le contenu précédent de la liste sera perdu.

Valeur de retour

Aucune.

Remarques

L'extraction est réalisée de manière récursive si la liste a un type 'Structure'.
Si la valeur JSON contient des éléments qui n'ont pas le bon type pour correspondre à la liste(), ils seront ignorés et l'élément de liste correspondant sera laissé vide.

Exemple

```
1   Donnees$ = "[ {" + Chr(34)
     + "x" + Chr(34) + ": 10, "
     + Chr(34) + "y" + Chr(34)
     + ": 20}, " +
2                "{" + Chr(34) +
     "x" + Chr(34) + ": 30, " +
     Chr(34) + "y" + Chr(34) +
     ": 50}, " +
3                "{" + Chr(34) +
     "x" + Chr(34) + ": -5, " +
     Chr(34) + "y" + Chr(34) +
     ": 100} ]"
4
5   Structure Localisation
6     x.l
7     y.l
8   EndStructure
9
10  NewList
     Localisations.Localisation()
11
12  ParseJSON(0, Donnees$)
13  ExtractJSONList(JSONValue(0),
     Localisations())
14
15  ForEach Localisations()
16    Debug
     Str(Localisations()\x) +
     ", " +
     Str(Localisations()\y)
17  Next
```

```
18
19    ; Résultat : 10, 20
20    ;            30, 50
21    ;            -5, 100
```

Voir aussi

ExtractJSONArray() , ExtractJSONMap() , ExtractJSONStructure() , InsertJSONArray() , InsertJSONList() , InsertJSONMap() , InsertJSONStructure() , SetJSONArray() , JSONType()

OS Supportés

Tous

107.13 ExtractJSONMap

Syntaxe

```
ExtractJSONMap(JSONValeur,
    Map())
```

Description

Extrait les membres de la valeur JSON de type #PB_JSON_Object dans la Map() spécifiée. La Map sera redimensionnée pour le nombre d'éléments contenu dans la valeur JSON.

Arguments

JSONValeur La valeur JSON.
La valeur doit être de type #PB_JSON_Object.

Map() La Map à remplir avec les éléments de JSON.
La Map sera redimensionnée pour avoir la même taille que la valeur JSON.
Tout le contenu précédent de la map sera perdu.

Valeur de retour

Aucune.

Remarques

L'extraction est effectuée de manière récursive si la Map a un type 'Structure'.
Si la valeur JSON contient des membres qui n'ont pas le bon type pour correspondre avec la Map(), ils seront ignorés et l'élément correspondant de la Map sera laissé vide.

Exemple

```
Donnees$ = "{" + Chr(34) + "activé" + Chr(34) + ": 1, " +
                    Chr(34) + "désactivé" + Chr(34) + ": 1, " +
                    Chr(34) + "visible" + Chr(34) + ": 0 }"
ParseJSON(0, Donnees$)

NewMap Options()
ExtractJSONMap(JSONValue(0), Options())

Debug Options("activé")
Debug Options("visible")

; Résultat : 1
;            0
```

Voir aussi

ExtractJSONArray() , ExtractJSONList() , ExtractJSONStructure() , InsertJSONArray() , InsertJSONList() , InsertJSONMap() , InsertJSONStructure() , SetJSONObject() , JSONType()

OS Supportés

Tous

107.14 ExtractJSONStructure

Syntaxe

```
ExtractJSONStructure(JSONValeur, *Memoire, Structure [, Options])
```

Description

Extrait les membres de la valeur JSON de type #PB_JSON_Object dans de la structure spécifiée, en mémoire.

Arguments

JSONValeur La valeur JSON.
La valeur doit être de type #PB_JSON_Object.

***Memoire** L'adresse de la mémoire de la structure à remplir.

Structure Le type de la structure à remplir.

Options (optionnel) Avec #PB_JSON_NoClear, la structure ne sera pas effacée avant l'extraction des données JSON : Si les données JSON ne spécifient pas de champ de structure, la valeur du champ actuel sera conservée. Sans #PB_JSON_NoClear, la structure entière sera effacée avant l'extraction des données à partir de JSON.

Valeur de retour

Aucune.

Remarques

La structure sera vidée de tout contenu précédent avant d'extraire les valeurs de JSON à moins que #PB_JSON_NoClear soit défini.
L'extraction est réalisée de manière récursive si la structure contient d'autres structures, des tableaux, des listes ou des maps.
Si la valeur JSON contient des membres qui n'ont pas le bon type pour correspondre à un membre de la structure, ils seront ignorés et l'élément de structure correspondante est laissé vide.
Tous les caractères '*' ou '$' sont supprimés des noms des membres de la structure avant de les comparer aux membres de l'objet JSON. Ainsi, une clé de membre ne doit pas inclure ces caractères.
La comparaison des clés de membre avec les noms des membres de la structure est réalisée de façon sensible à la casse. Si les données #JSON ont été créées ou parcourues avec l'option #PB_JSON_NoCase, la comparaison est effectuée suivant la casse.

Exemple

```
1  Structure Personne
2    Nom$
3    Age.l
4    List Livres.s()
5  EndStructure
6
7  Donnees$ = "{" + Chr(34) +
     "Nom" + Chr(34) + ": " +
     Chr(34) + "Jean Dupond" +
     Chr(34) + ", " +
8                    Chr(34) +
     "Age" + Chr(34) + ": 42, "
     +
```

```
                    Chr(34) +
    "Livres" + Chr(34) + ": ["
    +
                    Chr(34) +
    "Investir pour les nuls" +
    Chr(34) + ", " +
                    Chr(34) +
    "Dépenser pour les nuls
    ;)" + Chr(34) + "] }"

  ParseJSON(0, Donnees$)
  ExtractJSONStructure(JSONValue(0),
   @P.Personne, Personne)

  Debug P\Nom$
  Debug P\Age
  Debug ListSize(P\Livres())

  ; Résultat : Jean Dupond
  ;            42
  ;            2
```

Voir aussi

ExtractJSONArray() , ExtractJSONList() , ExtractJSONMap() , InsertJSONArray() , InsertJSONList() , InsertJSONMap() , InsertJSONStructure() , SetJSONObject() , JSONType()

OS Supportés

Tous

107.15 FreeJSON

Syntaxe

```
FreeJSON(#JSON)
```

Description

Libère les données JSON.

Arguments

#JSON Les données JSON à libérer. Si #PB_All est spécifié, tous les objets JSON restants sont libérés.

Valeur de retour

Aucune.

Remarques

Tous les objets JSON restants sont automatiquement libérés lorsque le programme se termine.

Voir aussi

IsJSON() , CreateJSON() , ParseJSON() , LoadJSON() , CatchJSON()

OS Supportés

Tous

107.16 GetJSONBoolean

Syntaxe

```
Resultat =
    GetJSONBoolean(JSONValeur)
```

Description

Renvoie la valeur booléenne d'une valeur JSON de type #PB_JSON_Boolean.

Arguments

JSONValeur La valeur JSON.
La valeur doit être de type #PB_JSON_Boolean.

Valeur de retour

La valeur booléenne #True ou #False.

Remarques

Une valeur JSON peut être réglée sur une valeur booléenne avec SetJSONBoolean() .

Exemple

```
1  ParseJSON(0, "[true, true,
     false]")
2
3  Debug
     GetJSONBoolean(GetJSONElement(JSONValue(0),
     0))
4  Debug
     GetJSONBoolean(GetJSONElement(JSONValue(0),
     1))
5  Debug
     GetJSONBoolean(GetJSONElement(JSONValue(0),
     2))
6
7  ; Résultat : 1
8  ;            1
9  ;            0
```

Voir aussi

SetJSONBoolean() , GetJSONDouble() , GetJSONElement() , GetJSONFloat() , GetJSONInteger() , GetJSONMember() , GetJSONString() , GetJSONQuad() , JSONType()

OS Supportés

Tous

107.17 GetJSONDouble

Syntaxe

```
Resultat.d =
    GetJSONDouble(JSONValeur)
```

Description

Renvoie un nombre décimal double précision ('Double') d'une valeur JSON de type #PB_JSON_Number.
Une valeur JSON peut être égale à un nombre en utilisant SetJSONDouble() , SetJSONFloat() , SetJSONInteger() ou SetJSONQuad() .

Arguments

JSONValeur La valeur JSON.
La valeur doit être de type #PB_JSON_Number.

Valeur de retour

Le nombre décimal de type 'Double'.

Exemple

```
1  ParseJSON(0, "[1, 1.23,
     1.23e-3]")
2
3  Debug
     GetJSONDouble(GetJSONElement(JSONValue(0),
     0))
4  Debug
     GetJSONDouble(GetJSONElement(JSONValue(0),
     1))
5  Debug
     GetJSONDouble(GetJSONElement(JSONValue(0),
     2))
6
7  ; Résultat : 1.0
8  ;            1.23
9  ;            0.00123
```

Voir aussi

SetJSONDouble() , SetJSONFloat() , SetJSONInteger() , SetJSONQuad() , GetJSONBoolean() , GetJSONElement() , GetJSONFloat() , GetJSONInteger() , GetJSONMember() , GetJSONString() , GetJSONQuad() , JSONType()

OS Supportés

Tous

107.18 GetJSONElement

Syntaxe

```
Resultat =
    GetJSONElement(JSONValeur,
    Index)
```

Description

Renvoie l'élément de tableau JSON à l'index donné d'une valeur JSON de type #PB_JSON_Array.

Arguments

JSONValeur La valeur JSON.
La valeur doit être de type #PB_JSON_Array.

Index L'indice de l'élément de tableau à renvoyer.
L'indice doit être compris entre 0 et JSONArraySize() - 1.

Valeur de retour

L'adresse de la valeur JSON à l'index de tableau spécifié.
Si la donnée 'Index' est hors de portée, le résultat est zéro.

Exemple

```
1  ParseJSON(0, "[1, 2, 3, 4,
     5]")
2
3  For i = 0 To
     JSONArraySize(JSONValue(0))
     - 1
4     Debug
     GetJSONInteger(GetJSONElement(JSONValue(0),
     i))
5  Next i
6
7  ; Resultat : 1
```

```
8    ;                2
9    ;                3
10   ;                4
11   ;                5
```

Voir aussi

SetJSONArray() , AddJSONElement() , RemoveJSONElement() , ResizeJSONElements() , ClearJSONElements() , JSONArraySize() , JSONType()

OS Supportés

Tous

107.19 GetJSONFloat

Syntaxe

```
Resultat.f =
    GetJSONFloat(JSONValeur)
```

Description

Renvoie un nombre décimal simple précision ('Float') d'une valeur JSON de type #PB_JSON_Number.
Une valeur JSON peut être égale à un nombre en utilisant SetJSONDouble() , SetJSONFloat() , SetJSONInteger() ou SetJSONQuad() .

Arguments

JSONValeur La valeur JSON.
La valeur doit être de type #PB_JSON_Number.

Valeur de retour

Le nombre décimal simple précision ('Float')

Exemple

```
1   ParseJSON(0, "[1, 1.23,
      1.23e-3]")
2
3   Debug
      GetJSONFloat(GetJSONElement(JSONValue(0),
      0))
4   Debug
      GetJSONFloat(GetJSONElement(JSONValue(0),
      1))
```

```
5   Debug
      GetJSONFloat(GetJSONElement(JSONValue(0),
      2))
6
7   ; Resultat : 1.0
8   ;
      1.23000001907349
9   ;             0.0012300000526
```

Voir aussi

SetJSONDouble() , SetJSONFloat() , SetJSONInteger() , SetJSONQuad() , GetJSONBoolean() , GetJSONDouble() , GetJSONElement() , GetJSONInteger() , GetJSONMember() , GetJSONString() , GetJSONQuad() , JSONType()

OS Supportés

Tous

107.20 GetJSONInteger

Syntaxe

```
Resultat =
    GetJSONInteger(JSONValeur)
```

Description

Renvoie un nombre entier 'Integer' d'une valeur JSON de type #PB_JSON_Number.
Une valeur JSON peut être égale à un nombre en utilisant SetJSONDouble() , SetJSONFloat() , SetJSONInteger() ou SetJSONQuad() .

Arguments

JSONValeur La valeur JSON.
La valeur doit être de type #PB_JSON_Number.

Valeur de retour

Le nombre entier 'Integer'.

Exemple

```
1   ParseJSON(0, "[1, 2, 3, 4,
      5]")
2
3   For i = 0 To
      JSONArraySize(JSONValue(0))
      - 1
```

```
4       Debug
        GetJSONInteger(GetJSONElement(JSONValue(0),
        i))
5     Next i
6
7     ; Resultat :
8     ;1
9     ;2
10    ;3
11    ;4
12    ;5
```

Voir aussi

SetJSONDouble() , SetJSONFloat() , SetJSONInteger() , SetJSONQuad() , GetJSONBoolean() , GetJSONDouble() , GetJSONElement() , GetJSONFloat() , GetJSONMember() , GetJSONString() , GetJSONQuad() , JSONType()

OS Supportés

Tous

107.21 GetJSONMember

Syntaxe

```
Resultat =
    GetJSONMember(JSONValeur,
    Cle$)
```

Description

Renvoie le membre de l'objet JSON en fonction de la clé donnée de type #PB_JSON_Object.

Arguments

JSONValeur La valeur JSON.
La valeur doit être de type #PB_JSON_Object.

Cle$ La clé de l'élément à renvoyer.
La clé est sensible à la casse à moins que l'option #PB_JSON_NoCase n'ait été spécifiée lors de la création ou de l'analyse des données JSON.

Valeur de retour

L'adresse de la valeur JSON en fonction de la clé spécifiée.
Si la clé n'existe pas dans l'objet alors le résultat est zéro.

Exemple

```
Input$ = "{ " + Chr(34) + "x" + Chr(34) + ": 10, " + Chr(34) + "y" + Chr(34) + ": 20, " + Chr(34) + "z" + Chr(34) + ": 30 }"

ParseJSON(0, Input$)

Debug GetJSONInteger(GetJSONMember(JSONValue(0), "x"))
Debug GetJSONInteger(GetJSONMember(JSONValue(0), "y"))
Debug GetJSONInteger(GetJSONMember(JSONValue(0), "z"))

; Resultat : 10
;            20
;            30
```

Voir aussi

SetJSONObject() , AddJSONMember() , RemoveJSONMember() , ClearJSONMembers() , ExamineJSONMembers() , JSONObjectSize() , JSONType()

OS Supportés

Tous

107.22 GetJSONString

Syntaxe

```
Resultat\$ =
    GetJSONString(JSONValeur)
```

Description

Renvoie la chaîne de caractères de la valeur JSON de type #PB_JSON_String.

Arguments

JSONValeur La valeur JSON.
La valeur doit être de type #PB_JSON_String.

Valeur de retour

La chaîne contenue dans la valeur JSON.

Exemple

```
ParseJSON(0, Chr(34) + "Le renard rapide saute par dessus le chien paresseux" + Chr(34))

Debug GetJSONString(JSONValue(0))

; Resultat : Le renard rapide saute par dessus le chien paresseux
```

Voir aussi

SetJSONString() , GetJSONBoolean() , GetJSONDouble() , GetJSONElement() , GetJSONFloat() , GetJSONInteger() , GetJSONMember() , GetJSONQuad() , JSONType()

OS Supportés

Tous

107.23 GetJSONQuad

Syntaxe

```
Resultat.q =
    GetJSONQuad(JSONValeur)
```

Description

Renvoie un entier ('Quad') de la valeur JSON de type #PB_JSON_Number. Une valeur JSON peut être égale à un nombre en utilisant SetJSONDouble() , SetJSONFloat() , SetJSONInteger() ou SetJSONQuad() .

Arguments

JSONValeur La valeur JSON.
La valeur doit être de type #PB_JSON_Number.

Valeur de retour

Le nombre entier ('Quad').

Exemple

```
1   ParseJSON(0, "[1, 2, 3, 4,
      5]")
2
3   For i = 0 To
      JSONArraySize(JSONValue(0))
      - 1
4     Debug
      GetJSONQuad(GetJSONElement(JSONValue(0),
      i))
5   Next i
6
7   ; Resultat : 1
8   ;            2
9   ;            3
10  ;            4
11  ;            5
```

Voir aussi

SetJSONDouble() , SetJSONFloat() , SetJSONInteger() , SetJSONQuad() , GetJSONBoolean() , GetJSONDouble() , GetJSONElement() , GetJSONFloat() , GetJSONInteger() , GetJSONMember() , GetJSONString() , JSONType()

OS Supportés

Tous

107.24 InsertJSONArray

Syntaxe

```
InsertJSONArray(JSONValeur,
    Tableau())
```

Description

Insère un tableau dans la valeur JSON donné.
La valeur JSON sera modifiée en type #PB_JSON_Array.

Arguments

JSONValeur La valeur JSON.
Le contenu de la valeur sera remplacée par le contenu de Tableau().

Tableau() Le tableau à insérer dans la valeur JSON.

Valeur de retour

Aucune.

Remarques

Si Tableau() a plus d'une dimension, la valeur JSON sera remplie avec un tableau de tableaux pour représenter les données multidimensionnelles. Voir l'exemple ci-dessous pour plus de détails.

Exemple

```
1   Dim Couleurs.s(3)
2   Couleurs(0) = "rouge"
3   Couleurs(1) = "jaune"
4   Couleurs(2) = "vert"
5   Couleurs(3) = "bleu"
6
7   If CreateJSON(0)
8     InsertJSONArray(JSONValue(0),
     Couleurs())
9     Debug ComposeJSON(0)
10  EndIf
11
12  ; Resultat : ["rouge",
     "jaune", "vert", "bleu"]
```

Exemple

```
1   Dim matrice(2, 2)
2   matrice(0, 0) = 1
3   matrice(1, 1) = 1
4   matrice(2, 2) = 1
5
6   If CreateJSON(0)
7     InsertJSONArray(JSONValue(0),
     matrice())
8     Debug ComposeJSON(0)
9   EndIf
10
11    ; Resultat : [[1, 0, 0],
    [0, 1, 0], [0, 0, 1]]
```

Voir aussi

InsertJSONList() , InsertJSONMap() , InsertJSONStructure() , ExtractJSONArray() , ExtractJSONList() , ExtractJSONMap() , ExtractJSONStructure() ,

OS Supportés

Tous

107.25 InsertJSONList

Syntaxe

```
InsertJSONList(JSONValeur,
    Liste())
```

Description

Insère une liste dans la valeur JSON donné. La valeur JSON sera modifiée en type #PB_JSON_Array.

Arguments

JSONValeur La valeur JSON.
Le contenu de la valeur sera remplacé par le contenu de la liste().

Liste() La liste à insérer dans la valeur JSON.

Valeur de retour

Aucune.

Exemple

```
1   NewList Noms.s()
2   AddElement(Noms()): Noms()
     = "Jean"
3   AddElement(Noms()): Noms()
     = "Jeanne"
4   AddElement(Noms()): Noms()
     = "Jim"
5
6   If CreateJSON(0)
7     InsertJSONList(JSONValue(0),
     Noms())
8     Debug ComposeJSON(0,
     #PB_JSON_PrettyPrint)
9   EndIf
10
11  ; Resultat :
12  ;[
13  ;   "Jean",
14  ;   "Jeanne",
15  ;   "Jim"
16  ;]
```

Voir aussi

InsertJSONArray() , InsertJSONMap() , InsertJSONStructure() , ExtractJSONArray() , ExtractJSONList() , ExtractJSONMap() , ExtractJSONStructure() ,

OS Supportés

Tous

107.26 InsertJSONMap

Syntaxe

```
InsertJSONMap(JSONValeur,
    Map())
```

Description

Insère une Map dans la valeur JSON donnée.
La valeur JSON sera modifiée en type #PB_JSON_Object.

Arguments

JSONValeur La valeur JSON.
Le contenu de la valeur sera remplacé par le contenu de la Map().

Map() La Map à insérer dans la valeur JSON.

Valeur de retour

Aucune.

Exemple

```
1   NewMap Couleurs()
2   Couleurs("rouge") = $0000FF
3   Couleurs("vert")  = $00FF00
4   Couleurs("bleu")  = $FF0000
5
6   If CreateJSON(0)
7     InsertJSONMap(JSONValue(0),
    Couleurs())
8     Debug ComposeJSON(0,
    #PB_JSON_PrettyPrint)
9   EndIf
10
11  ; Resultat :
12  ;{
13  ;   "bleu" : 16711680,
14  ;   "vert" : 65280,
15  ;   "rouge": 255
16  ;}
```

Voir aussi

InsertJSONArray() , InsertJSONList() , InsertJSONStructure() , ExtractJSONArray() , ExtractJSONList() , ExtractJSONMap() , ExtractJSONStructure() ,

OS Supportés

Tous

107.27 InsertJSONStructure

Syntaxe

```
InsertJSONStructure(JSONValeur,
    *Memoire, Structure)
```

Description

Insère le contenu de la structure présente en mémoire dans la valeur JSON donnée. La valeur JSON sera changé en type #PB_JSON_Object et contient un membre pour chaque membre de la structure.

Arguments

JSONValeur La valeur JSON. Le contenu de la valeur sera remplacé par le contenu de la structure.

***Memoire** L'adresse de la structure à insérer dans la valeur JSON.

Structure Le type de la structure à insérer.

Valeur de retour

Aucune.

Exemple

```
1   Structure Personne
2     Prenom$
3     Nom$
4     Age.l
5     List Livres.s()
6   EndStructure
7
8   Define P.Personne
9   P\Prenom$ = "Jean"
10  P\Nom$  = "Dupond"
11  P\Age        = 42
12  AddElement(P\Livres()):
     P\Livres() = "Investir
     pour les nuls"
13  AddElement(P\Livres()):
     P\Livres() = "English pour
     les nuls"
14  AddElement(P\Livres()):
     P\Livres() = "Dépenser
     pour les nuls ;)"
15
16  If CreateJSON(0)
17    InsertJSONStructure(JSONValue(0),
     @P, Personne)
18    Debug ComposeJSON(0,
     #PB_JSON_PrettyPrint)
```

```
19  EndIf
20
21  ; Resultat :
22  ;{
23  ; "Nom"    : "Dupond",
24  ; "Livres": [
25  ;    "Investir pour les
     nuls",
26  ;    "English pour les nuls",
27  ;    "Dépenser pour les nuls
     ;)"
28  ;  ],
29  ; "Prenom": "Jean",
30  ; "Age"    : 42
31  ;}
```

Voir aussi

InsertJSONArray() , InsertJSONList() , InsertJSONMap() , ExtractJSONArray() , ExtractJSONList() , ExtractJSONMap() , ExtractJSONStructure() ,

OS Supportés

Tous

107.28 IsJSON

Syntaxe

```
Resultat = IsJSON(#JSON)
```

Description

Teste si le numéro #JSON donné représente des données JSON valides et correctement initialisées.

Arguments

#JSON La valeur JSON.

Valeur de retour

Non nulle si #JSON est valide, zéro sinon.

Remarques

Cette fonction est solide et peut être utilisée avec n'importe quelle valeur.

Voir aussi

CreateJSON() , CatchJSON() , LoadJSON() , ParseJSON() , FreeJSON()

OS Supportés

Tous

107.29 JSONArraySize

Syntaxe

```
Resultat =
    JSONArraySize(JSONValeur)
```

Description

Renvoie le nombre d'éléments dans une valeur JSON de type #PB_JSON_Array.

Arguments

JSONValeur La valeur JSON.
La valeur doit être de type #PB_JSON_Array.

Valeur de retour

Le nombre d'éléments dans le tableau JSON.

Exemple

```
1  ParseJSON(0, "[1, 2, null,
     true]")
2  Debug
     JSONArraySize(JSONValue(0))
3
4  ; Resultat : 4
```

Voir aussi

SetJSONArray() , AddJSONElement() , RemoveJSONElement() , ResizeJSONElements() , ClearJSONElements() , GetJSONElement() , JSONType()

OS Supportés

Tous

107.30 JSONErrorLine

Syntaxe

```
Resultat = JSONErrorLine()
```

Description

Renvoie le numéro de ligne dans l'entrée JSON de la dernière opération qui a échoué lors du parcours des données avec ParseJSON() , CatchJSON() ou LoadJSON() .

Arguments

Aucun.

Valeur de retour

Le numéro de la ligne (base 1) de la dernière erreur.

Voir aussi

JSONErrorPosition() , JSONErrorMessage() , ParseJSON() , CatchJSON() , LoadJSON()

OS Supportés

Tous

107.31 JSONErrorMessage

Syntaxe

```
Resultat\$ =
    JSONErrorMessage()
```

Description

Renvoie un message décrivant la cause de la dernière opération qui a échoué avec ParseJSON() , CatchJSON() ou LoadJSON() .

Arguments

Aucun.

Valeur de retour

Le message d'erreur en anglais.

Exemple

```
1  If ParseJSON(0, "[1, 2, 3
     4]")
2    ; utilisation des données
3  Else
4    Debug JSONErrorMessage()
5  EndIf
6
```

```
7 |   ; Resultat : Unexpected
    |     character
```

Voir aussi

JSONErrorLine() , JSONErrorPosition() , ParseJSON() , CatchJSON() , LoadJSON()

OS Supportés

Tous

107.32 JSONErrorPosition

Syntaxe

```
Resultat = JSONErrorPosition()
```

Description

Renvoie la position du caractère dans la ligne de la dernière erreur avec ParseJSON() , CatchJSON() ou LoadJSON() .

Arguments

Aucun.

Valeur de retour

La position du caractère (base 1) de la dernière erreur à l'intérieur de la ligne rapportée par JSONErrorLine() .

Voir aussi

JSONErrorLine() , JSONErrorMessage() , ParseJSON() , CatchJSON() , LoadJSON()

OS Supportés

Tous

107.33 JSONMemberKey

Syntaxe

```
Resultat\$ =
    JSONMemberKey(JSONValeur)
```

Description

Après un appel à NextJSONMember() , renvoie la clé du membre de l'objet en cours d'examen de la valeur spécifiée JSON de type #PB_JSON_Object

Arguments

JSONValeur La valeur JSON.
La valeur doit être de type #PB_JSON_Object et en cours d'examen par ExamineJSONMembers() .

Valeur de retour

La clé de l'élément courant de l'objet JSON.

Exemple

Voir ExamineJSONMembers() .

Voir aussi

ExamineJSONMembers() , NextJSONMember() , JSONMemberValue()

OS Supportés

Tous

107.34 JSONMemberValue

Syntaxe

```
Resultat =
    JSONMemberValue(JSONValeur)
```

Description

Après un appel à NextJSONMember() , renvoie l'adresse du membre de l'objet en cours d'examen de la valeur spécifiée JSON de type #PB_JSON_Object

Arguments

JSONValeur La valeur JSON.
La valeur doit être de type #PB_JSON_Object et en cours d'examen par ExamineJSONMembers() .

Valeur de retour

L'adresse du membre actuel de l'objet JSON.

Exemple

Voir ExamineJSONMembers() .

Voir aussi

ExamineJSONMembers() , NextJSONMember() , JSONMemberKey()

OS Supportés

Tous

107.35 JSONObjectSize

Syntaxe

```
Resultat =
    JSONObjectSize(JSONValeur)
```

Description

Renvoie le nombre de membres dans une valeur JSON de type #PB_JSON_Object.

Arguments

JSONValeur La valeur JSON.
La valeur doit être de type #PB_JSON_Object.

Valeur de retour

Le nombre de membres dans l'objet JSON.

Exemple

```
Donnee$ = "{ " + Chr(34) + "x" + Chr(34) + ": 10, " + Chr(34) + "y" + Chr(34) + ": 20, " + Chr(34) + "z" + Chr(34) + ": 30 }"

ParseJSON(0, Donnee$)
Debug JSONObjectSize(JSONValue(0))

; Resultat : 3
```

Voir aussi

SetJSONObject() , AddJSONMember() , RemoveJSONMember() , ClearJSONMembers() , GetJSONMember() , ExamineJSONMembers() , JSONType()

OS Supportés

Tous

107.36 JSONType

Syntaxe

```
Resultat =
    JSONType(JSONValeur)
```

Description

Renvoie le type de la valeur JSON donné.

Arguments

JSONValeur La valeur JSON.

Valeur de retour

Peut être l'une des valeurs suivantes :

#PB_JSON_Null
: La valeur représente la valeur 'null' JSON.

#PB_JSON_String
: La valeur contient une chaîne de caractères.
GetJSONString() peut être utilisé pour lire la chaîne de caractères.

#PB_JSON_Number
: La valeur contient un nombre.
GetJSONDouble() , GetJSONFloat() , GetJSONInteger() ou GetJSONQuad() peuvent être utilisées pour lire le nombre.

#PB_JSON_Boolean
: La valeur contient un booléen.
GetJSONBoolean() peut être utilisé pour lire la valeur booléenne.

#PB_JSON_Array
: La valeur contient un tableau d'éléments JSON.
JSONArraySize() renvoie la taille du tableau.
GetJSONElement() peut être utilisé pour obtenir un élément du tableau.
AddJSONElement() , RemoveJSONElement() , ResizeJSONElements() ou ClearJSONElements() peuvent être utilisés pour modifier le tableau.

#PB_JSON_Object
: La valeur contient un objet (un ensemble de paires clé/valeur).
JSONObjectSize() renvoie le nombre de membres dans l'objet.
GetJSONMember() renvoie la valeur d'un membre.
ExamineJSONMembers() peut être utilisé pour examiner les valeurs d'un membre. AddJSONMember() , RemoveJSONMember() ou ClearJSONMembers() peuvent être utilisés pour modifier l'objet.

Exemple

```
; Une procédure qui accepte n'importe quelle valeur JSON et renvoie une chaîne
;
Procedure.s GetAnyValue(Value)
  Select JSONType(Value)
    Case #PB_JSON_Null: ProcedureReturn "L'élément null"
    Case #PB_JSON_String: ProcedureReturn GetJSONString(Value)
    Case #PB_JSON_Number: ProcedureReturn StrD(GetJSONDouble(Value))
    Case #PB_JSON_Boolean: ProcedureReturn Str(GetJSONBoolean(Value))
    Case #PB_JSON_Array: ProcedureReturn "Tableau"
    Case #PB_JSON_Object: ProcedureReturn "Objet"
  EndSelect
EndProcedure

ParseJSON(0, "[1, 2, true, null, " + Chr(34) + "hello" + Chr(34) + "]")
For i = 0 To JSONArraySize(JSONValue(0)) - 1
  Debug GetAnyValue(GetJSONElement(JSONValue(0), i))
Next i

; Resultat : 1
;            2
;            1
;            L'élément null
;            hello
```

Voir aussi

JSONValue() , SetJSONArray() , SetJSONBoolean() , SetJSONDouble() , SetJSONFloat() , SetJSONInteger() , SetJSONNull() , SetJSONObject() , SetJSONString() , SetJSONQuad()

OS Supportés

Tous

107.37 JSONValue

Syntaxe

```
Resultat = JSONValue(#JSON)
```

Description

Renvoie la valeur des données #JSON spécifiés.
Le type de la valeur peut être vérifié avec JSONType() .

Arguments

#JSON Les données JSON.

Valeur de retour

La valeur JSON.
Le résultat n'est jamais nul pour un ensemble de données #JSON valides.

Remarques

Chaque donnée #JSON contient exactement une valeur JSON (contenant des valeurs éventuellement imbriquées).
Les données #JSON nouvellement créées avec CreateJSON() contiennent une valeur de type #PB_JSON_Null.
Le type de la valeur JSON ou de son contenu peut être modifié avec l'une des fonctions suivantes :
- SetJSONArray() : Modifie la valeur en un tableau (vide)
- SetJSONBoolean() : Modifie la valeur en un booléen
- SetJSONDouble() : Modifie la valeur en un nombre 'Double'
- SetJSONFloat() : Modifie la valeur en un nombre 'Float'
- SetJSONInteger() : Modifie la valeur en un nombre entier
- SetJSONNull() : Modifie la valeur en une valeur 'null'
- SetJSONObject() : Modifie la valeur en un objet (vide)
- SetJSONString() : Modifie la valeur en une chaîne de caractère
- SetJSONQuad() : Modifie la valeur en un nombre 'Quad'

Exemple

```
1  ParseJSON(0, Chr(34) + "Le
     renard rapide saute par
     dessus le chien paresseux"
     + Chr(34))
```

```
2
3   Debug
      GetJSONString(JSONValue(0))
4
5   ; Resultat : Le renard
      rapide saute par dessus le
      chien paresseux
```

Voir aussi

JSONType()

OS Supportés

Tous

107.38 LoadJSON

Syntaxe

```
Resultat = LoadJSON(#JSON,
    Fichier$ [, Options])
```

Description

Parcourt (parse) les données JSON à partir d'un fichier.
Le contenu du fichier devrait être encodé au format UTF-8.
Les fichiers avec un autre codage de caractères ne peuvent pas être lus par cette commande.
La fonction JSONValue() peut être utilisée pour accéder à la valeur(s) JSON après le parse.

Arguments

#JSON Un numéro pour identifier la nouvelle JSON.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Fichier$ Le nom du fichier au format UTF-8 contenant les données JSON.

Options (optionnel)
#PB_JSON_NoCase : Insensible à la casse.
La valeur par défaut est d'être sensible à la casse.

Valeur de retour

Renvoie une valeur non nulle si les données JSON ont été analysées correctement, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #JSON alors le nombre généré est renvoyé en cas de succès.

Remarques

En cas d'erreur, les fonctions JSONErrorMessage() , JSONErrorLine() et JSONErrorPosition() peuvent être utilisées pour obtenir plus d'informations sur l'erreur.
JSON est un format de données sensible à la casse. Cependant, dans certaines situations, telles que les structures de désérialisation avec ExtractJSONStructure() ou des commandes similaires, il peut être utile de traiter des objets JSON de façon insensible à la casse. L'option #PB_JSON_NoCase permet de traiter tous les clés des membres de l'objet comme insensible à la casse.

Voir aussi

CreateJSON() , CatchJSON() , ParseJSON() , JSONValue() , FreeJSON() , JSONErrorMessage() , JSONErrorLine() , JSONErrorPosition() , SaveJSON()

OS Supportés

Tous

107.39 NextJSONMember

Syntaxe

```
Resultat =
    NextJSONMember(JSONValeur)
```

Description

Après un appel à ExamineJSONMembers() , cette fonction est utilisée pour parcourir tous les membres de la valeur JSON spécifiée de type #PB_JSON_Object. JSONMemberKey() et JSONMemberValue() peuvent être utilisés pour obtenir des informations sur l'élément courant.

Arguments

JSONValeur La valeur JSON.
La valeur doit être de type #PB_JSON_Object et ExamineJSONMembers() doit avoir été appelé sur cette valeur.

Valeur de retour

Renvoie une valeur non nulle si un autre membre JSON a été trouvé.
Si le résultat est zéro, alors il n'y a plus de membres JSON à examiner.

Exemple

Voir ExamineJSONMembers() .

Voir aussi

ExamineJSONMembers() ,
JSONMemberKey() , JSONMemberValue()

OS Supportés

Tous

107.40 ParseJSON

Syntaxe

```
Resultat = ParseJSON(#JSON,
    Entree$ [, Options])
```

Description

Parcourt (parse) les données JSON à partir d'une chaîne de caractères.
La fonction JSONValue() peut être utilisée pour accéder à la valeur(s) JSON après le parse.

Arguments

#JSON Le numéro d'identification de la nouvelle JSON.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Entree$ La chaîne de caractères contenant les données JSON à analyser.

Options (optionnel)
#PB_JSON_NoCase : Insensible à la casse.
La valeur par défaut est d'être sensible à la casse.

Valeur de retour

Renvoie une valeur non nulle si les données JSON ont été analysées correctement, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #JSON alors le nombre généré est renvoyé en cas de succès.

Remarques

En cas d'erreur, les fonctions JSONErrorMessage() , JSONErrorLine() et JSONErrorPosition() peuvent être utilisées pour obtenir plus d'informations sur l'erreur.

Pour parcourir les données JSON directement à partir d'une mémoire tampon, utiliser plutôt la fonction CatchJSON() .
JSON est un format de données sensible à la casse. Cependant, dans certaines situations, telles que les structures de désérialisation avec ExtractJSONStructure() ou des commandes similaires, il peut être utile de traiter des objets JSON de façon insensible à la casse. L'option #PB_JSON_NoCase permet de traiter tous les clés des membres de l'objet comme insensible à la casse.

Exemple

```
If ParseJSON(0, "[1, 2, 3, 4, 5]")
  For i = 0 To JSONArraySize(JSONValue(0)) - 1
    Debug GetJSONInteger(GetJSONElement(JSONValue(0), i))
  Next i
Else
  Debug JSONErrorMessage()
EndIf

; Resultat : 1
;            2
;            3
;            4
;            5
```

Voir aussi

CreateJSON() , CatchJSON() , LoadJSON() , JSONValue() , FreeJSON() , JSONErrorMessage() , JSONErrorLine() , JSONErrorPosition() , ExportJSON()

OS Supportés

Tous

107.41 RemoveJSONElement

Syntaxe

```
RemoveJSONElement(JSONValeur, Index)
```

Description

Retire l'élément à l'index spécifié d'une valeur de JSON de type #PB_JSON_Array.

Arguments

JSONValeur La valeur JSON.
La valeur doit être de type #PB_JSON_Array.

Index L'indice de l'élément à supprimer.
La valeur doit être comprise entre 0 et JSONArraySize() - 1.

Valeur de retour

Aucune.

Exemple

```
1  ParseJSON(0, "[1, 2, 3, 4, 5]")
2  RemoveJSONElement(JSONValue(0), 2)
3  Debug ComposeJSON(0)
4
5  ; Resultat : [1, 2, 4, 5]
```

Voir aussi

SetJSONArray() , AddJSONElement() , ResizeJSONElements() , ClearJSONElements() , GetJSONElement() , JSONArraySize() , JSONType()

OS Supportés

Tous

107.42 RemoveJSONMember

Syntaxe

```
RemoveJSONMember(JSONValeur, Cle$)
```

Description

Retire le membre dont la clé est spécifiée, d'une valeur de JSON de type #PB_JSON_Object.

Arguments

JSONValeur La valeur JSON.
La valeur doit être de type #PB_JSON_Object.

Key$ La clé du membre à supprimer.

Valeur de retour

Aucune.

Exemple

```
Donnee$ = "{ " + Chr(34) + "x" + Chr(34) + ": 10, " + Chr(34) + "y" + Chr(34) + ": 20, " + Chr(34) + "z" + Chr(34) + ": 30 }"

ParseJSON(0, Donnee$)
RemoveJSONMember(JSONValue(0), "x")
Debug ComposeJSON(0, #PB_JSON_PrettyPrint)


; Resultat : {
;             "y": 20,
;             "z": 30
;             }
```

Voir aussi

SetJSONObject() , AddJSONMember() , ClearJSONMembers() , GetJSONMember() , ExamineJSONMembers() , JSONObjectSize() , JSONType()

OS Supportés

Tous

107.43 ResizeJSONElements

Syntaxe

```
ResizeJSONElements(JSONValeur, Taille)
```

Description

Redimensionne une valeur JSON de type #PB_JSON_Array de sorte qu'il possède un nombre donné d'éléments.

Arguments

JSONValeur La valeur JSON.
La valeur doit être de type #PB_JSON_Array.

Taille La nouvelle taille du tableau.
Indique le nombre total d'éléments (pas l'indice de l'élément le plus élevé du tableau comme avec Dim).

Valeur de retour

Aucune.

Remarques

Si de nouveaux éléments sont ajoutés au tableau, ils auront le type #PB_JSON_Null.

Exemple

```
1   ParseJSON(0, "[1, 2, 3, 4,
      5]")
2
3   ResizeJSONElements(JSONValue(0),
      3)
4   Debug ComposeJSON(0)
5
6   ResizeJSONElements(JSONValue(0),
      10)
7   Debug ComposeJSON(0)
8
9   ; Resultat : [1, 2, 3]
10  ;             [1, 2, 3,
      null, null, null, null,
      null, null, null]
```

Voir aussi

SetJSONArray() , AddJSONElement() , RemoveJSONElement() , ClearJSONElements() , GetJSONElement() , JSONArraySize() , JSONType()

OS Supportés

Tous

107.44 SaveJSON

Syntaxe

```
Resultat = SaveJSON(#JSON,
    NomFichier$ [, Options])
```

Description

Sauvegarde les données JSON dans un fichier.
Le fichier sera encodé en UTF-8 sans en-tête BOM (byte-order mark).

Arguments

#JSON Le JSON à enregistrer.

Options (optionnel)
#PB_JSON_PrettyPrint : Les données enregistrées contiennent les sauts de ligne et les espaces pour une meilleure lisibilité. L'espace blanc supplémentaire n'est pas

significatif pour le format JSON. La sortie sera correctement interprétée avec un lecteur JSON avec ou sans cette option.

Valeur de retour

Renvoie une valeur non nulle si le fichier a été enregistré avec succès.
S'il y a une erreur lors de l'enregistrement du fichier, le résultat est zéro.

Voir aussi

ComposeJSON() , ExportJSON() , ExportJSONSize() , LoadJSON()

OS Supportés

Tous

107.45 SetJSONArray

Syntaxe

```
Resultat =
    SetJSONArray(JSONValeur)
```

Description

Change le type de la valeur JSON en #PB_JSON_Array.
Le tableau n'aura pas d'éléments (même si la valeur précédemment contenait des éléments de tableau).

Arguments

JSONValeur La valeur JSON.

Valeur de retour

Renvoie la valeur tableau ou zéro sinon.

Exemple

```
1  If CreateJSON(0)
2    ValeurTableau =
     SetJSONArray(JSONValue(0))
3    SetJSONString(AddJSONElement(ValeurTableau),
     "hello")
4    SetJSONString(AddJSONElement(ValeurTableau),
     "le Monde")
5
6    Debug ComposeJSON(0)
7  EndIf
8
9  ; Resultat : ["hello", "le
     Monde"]
```

Voir aussi

AddJSONElement() , RemoveJSONElement() , ResizeJSONElements() , ClearJSONElements() , GetJSONElement() , JSONArraySize() , JSONType()

OS Supportés

Tous

107.46 SetJSONBoolean

Syntaxe

```
SetJSONBoolean(JSONValeur,
    Valeur)
```

Description

Change le type de la valeur JSON en #PB_JSON_Boolean et stocke la valeur booléenne donnée.

Arguments

JSONValeur La valeur JSON.

Valeur La valeur booléenne à stocker. Une valeur non nulle est stockée sous forme #True, une valeur de 0 est stockée sous forme #False.

Valeur de retour

Aucune.

Exemple

```
1  If CreateJSON(0)
2    ValeurTableau =
     SetJSONArray(JSONValue(0))
3    SetJSONBoolean(AddJSONElement(ValeurTableau),
     #True)
4    SetJSONBoolean(AddJSONElement(ValeurTableau),
     #False)
5
6    Debug ComposeJSON(0)
7  EndIf
8
9  ; Resultat : [true, false]
```

Voir aussi

GetJSONBoolean() , SetJSONArray() , SetJSONDouble() , SetJSONFloat() , SetJSONInteger() , SetJSONNull() , SetJSONObject() , SetJSONString() , SetJSONQuad()

OS Supportés

Tous

107.47 SetJSONDouble

Syntaxe

```
SetJSONDouble(JSONValeur,
    Valeur.d)
```

Description

Change le type de la valeur JSON en #PB_JSON_Number et stocke la valeur 'Double' donnée.

Arguments

JSONValeur La valeur JSON.

Valeur.d La valeur à stocker.

Valeur de retour

Aucune.

Remarques

Notez que JSON ne permet pas les valeurs spéciales en virgule flottante +Infinity, -Infinity ou Nan (Not a number).
Si ces valeurs sont utilisées avec cette fonction, elles seront remplacées par la donnée 'null' lorsque les données seront sauvegardées ou encodées.
Les fonctions IsInfinity() ou IsNaN() peuvent être utilisées pour détecter ce cas.

Exemple

```
1  If CreateJSON(0)
2    ValeurTableau =
     SetJSONArray(JSONValue(0))
3    SetJSONDouble(AddJSONElement(ValeurTableau),
     1.23)
4    SetJSONDouble(AddJSONElement(ValeurTableau),
     4.56)
5
6    Debug ComposeJSON(0)
7  EndIf
8
9    ; Resultat : [1.23, 4.56]
```

Voir aussi

GetJSONDouble() , SetJSONArray() , SetJSONBoolean() , SetJSONFloat() , SetJSONInteger() , SetJSONNull() , SetJSONObject() , SetJSONString() , SetJSONQuad()

OS Supportés

Tous

107.48 SetJSONFloat

Syntaxe

```
SetJSONFloat(JSONValeur,
    Valeur.f)
```

Description

Change le type de la valeur JSON en #PB_JSON_Number et stocke la valeur 'Float' donnée.

Arguments

JSONValeur La valeur JSON.

Valeur.f La valeur à stocker.

Valeur de retour

Aucune.

Remarques

Notez que JSON ne permet pas les valeurs spéciales en virgule flottante +Infinity, -Infinity ou Nan (Not a number).
Si ces valeurs sont utilisées avec cette fonction, elles seront remplacées par la donnée 'null' lorsque les données seront sauvegardées ou encodées.
Les fonctions IsInfinity() ou IsNaN() peuvent être utilisées pour détecter ce cas.

Exemple

```
1  If CreateJSON(0)
2    ValeurTableau =
     SetJSONArray(JSONValue(0))
3    SetJSONFloat(AddJSONElement(ValeurTableau),
     1.23)
4    SetJSONFloat(AddJSONElement(ValeurTableau),
     4.56)
5
6    Debug ComposeJSON(0)
7  EndIf
```

```
8
9     ; Resultat : [1.2300000191,
        4.5599999428]
```

Voir aussi

GetJSONFloat() , SetJSONArray() , SetJSONBoolean() , SetJSONDouble() , SetJSONInteger() , SetJSONNull() , SetJSONObject() , SetJSONString() , SetJSONQuad()

OS Supportés

Tous

107.49 SetJSONInteger

Syntaxe

```
SetJSONInteger(JSONValeur,
    Valeur)
```

Description

Change le type de la valeur JSON en #PB_JSON_Number et stocke la valeur 'Integer' donnée.

Arguments

JSONValeur La valeur JSON.

Valeur La valeur à stocker.

Valeur de retour

Aucune.

Exemple

```
1   If CreateJSON(0)
2     ValeurTableau =
      SetJSONArray(JSONValue(0))
3     SetJSONInteger(AddJSONElement(ValeurTableau),
      1)
4     SetJSONInteger(AddJSONElement(ValeurTableau),
      2)
5     SetJSONInteger(AddJSONElement(ValeurTableau),
      3)
6
7     Debug ComposeJSON(0)
8   EndIf
9
10  ; Resultat : [1, 2, 3]
```

Voir aussi

GetJSONInteger() , SetJSONArray() , SetJSONBoolean() , SetJSONDouble() , SetJSONFloat() , SetJSONNull() , SetJSONObject() , SetJSONString() , SetJSONQuad()

OS Supportés

Tous

107.50 SetJSONNull

Syntaxe

```
SetJSONNull(JSONValeur)
```

Description

Efface la valeur JSON et transforme le type en #PB_JSON_Null.

Arguments

JSONValeur La valeur JSON.

Valeur de retour

Aucune.

Exemple

```
ParseJSON(0, "[1, 2, 3, 4, 5]")
SetJSONNull(GetJSONElement(JSONValue(0), 2))
SetJSONNull(GetJSONElement(JSONValue(0), 3))
Debug ComposeJSON(0)

; Resultat : [1, 2, null, null, 5]
```

Voir aussi

SetJSONArray() , SetJSONBoolean() , SetJSONDouble() , SetJSONFloat() , SetJSONInteger() , SetJSONObject() , SetJSONString() , SetJSONQuad()

OS Supportés

Tous

107.51 SetJSONObject

Syntaxe

```
Resultat =
    SetJSONObject(JSONValeur)
```

Description

Change le type de la valeur JSON en #PB_JSON_Object.
L'objet n'a pas de membres (même si la valeur précédemment contenait des membres).

Arguments

JSONValeur La valeur JSON.

Valeur de retour

Renvoie la valeur objet ou zéro sinon.

Exemple

```
If CreateJSON(0)
  ValeurObjet = SetJSONObject(JSONValue(0))
  SetJSONInteger(AddJSONMember(ValeurObjet, "x"), 10)
  SetJSONInteger(AddJSONMember(ValeurObjet, "y"), 20)
  SetJSONInteger(AddJSONMember(ValeurObjet, "z"), 30)

  Debug ComposeJSON(0, #PB_JSON_PrettyPrint)
EndIf

; Resultat : {
;               "x": 10,
;               "y": 20,
;               "z": 30
;            }
```

Voir aussi

AddJSONMember() , RemoveJSONMember() , ClearJSONMembers() , GetJSONMember() , ExamineJSONMembers() , JSONObjectSize() , JSONType()

OS Supportés

Tous

107.52 SetJSONQuad

Syntaxe

```
SetJSONQuad(JSONValeur,
    Valeur.q)
```

Description

Change le type de la valeur JSON en #PB_JSON_Number et stocke la valeur 'Quad' donnée.

Arguments

JSONValeur La valeur JSON.

Valeur.q La valeur à stocker.

Valeur de retour

Aucune.

Exemple

```
1   If CreateJSON(0)
2     ValeurTableau =
     SetJSONArray(JSONValue(0))
3     SetJSONQuad(AddJSONElement(ValeurTableau),
     1)
4     SetJSONQuad(AddJSONElement(ValeurTableau),
     2)
5     SetJSONQuad(AddJSONElement(ValeurTableau),
     3)
6
7     Debug ComposeJSON(0)
8   EndIf
9
10  ; Resultat : [1, 2, 3]
```

Voir aussi

GetJSONQuad() , SetJSONArray() , SetJSONBoolean() , SetJSONDouble() , SetJSONFloat() , SetJSONInteger() , SetJSONNull() , SetJSONObject() , SetJSONString()

OS Supportés

Tous

107.53 SetJSONString

Syntaxe

```
SetJSONString(JSONValeur,
    Texte$)
```

Description

Change le type de la valeur JSON en #PB_JSON_String et stocke la valeur chaîne de caractères donnée.

Arguments

JSONValeur La valeur JSON.

Texte$ La chaîne de caractères à stocker.

Valeur de retour

Aucune.

Exemple

```
If CreateJSON(0)
  ValeurTableau = SetJSONArray(JSONValue(0))
  SetJSONString(AddJSONElement(ValeurTableau), "avec les caractères" + Chr(13) + Chr(10) + "d'échappement retour chariot et retour à la ligne")
  SetJSONString(AddJSONElement(ValeurTableau), "avec le caractère \ antislash")

  Debug ComposeJSON(0)
EndIf

; Resultat : ["avec les caractères\r\nd'échappement retour chariot et retour à la ligne", "avec le caractère \\ antislash"]
```

Voir aussi

GetJSONString() , SetJSONArray() , SetJSONBoolean() , SetJSONDouble() , SetJSONFloat() , SetJSONInteger() , SetJSONNull() , SetJSONObject() , SetJSONQuad()

OS Supportés

Tous

Chapitre 108

Keyboard

Généralités

PureBasic propose un accès simple et rapide au clavier qui ne doit être utilisé que pour des jeux ou applications qui nécessitent une gestion directe, exlusive et très rapide de celui-ci.
Pour les applications normales, il convient d'utiliser AddKeyboardShortcut() .
Sous Windows, l'interface est assurée par la technologie DirectX. Une version récente de DirectX 9 doit être installée (voir ici : DirectX 9 runtime installer).

OS Supportés

Tous

108.1 InitKeyboard

Syntaxe

```
Resultat = InitKeyboard()
```

Description

Initialise l'environnement propre à la gestion du clavier.

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle si l'initialisation a réussi, zéro sinon.

Remarques

Vous devez appeler cette fonction avant tout appel de fonctions de cette bibliothèque.

Voir aussi

ExamineKeyboard() , KeyboardMode()

OS Supportés

Tous

108.2 ExamineKeyboard

Syntaxe

```
Resultat = ExamineKeyboard()
```

Description

Met à jour l'état du clavier.

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle si l'état du clavier a changé, zéro sinon.

Remarques

Cette fonction doit être appelée avant d'utiliser les commandes KeyboardInkey() , KeyboardPushed() et KeyboardReleased() .

Voir aussi

InitKeyboard() , KeyboardMode() , KeyboardInkey() , KeyboardPushed() , KeyboardReleased()

OS Supportés

Tous

108.3 KeyboardInkey

Syntaxe

```
Resultat\$ = KeyboardInkey()
```

Description

Renvoie le dernier caractère tapé sur le clavier.

Arguments

Aucun.

Valeur de retour

Renvoie le dernier caractère tapé sur le clavier, très utile pour interagir facilement avec l'utilisateur (nom dans les highscores, console dans le jeu, etc.).

Exemple

```
If InitSprite() And InitKeyboard() And OpenScreen(800,600,32,"")
 Repeat
   FlipBuffers()
   ClearScreen(RGB(0, 0, 0))

   ExamineKeyboard()

   ; Efface le dernier caractère si la touche 'Back' est appuyée
   ;
   If KeyboardReleased(#PB_Key_Back)
     FullText$ = Left(FullText$, Len(FullText$)-1)
   Else
     resultat$=KeyboardInkey()
     If FindString("1234567890'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyzéèçàù", resultat$) ; Ajouter vos propres caractères valides ici
       FullText$ + resultat$
     EndIf ; Ajouter le nouveau texte à l'actuel (le cas échéant)
   EndIf

   ; Affiche le résultat
   ;
   If StartDrawing(ScreenOutput())
     DrawingMode(1)
     FrontColor(RGB(128, 255, 0))
     DrawText(20, 20, "Ecrivez un petit texte...:")
     DrawText(20, 40, FullText$)
     StopDrawing()
   EndIf
 Until KeyboardPushed(#PB_Key_Escape)
```

```
29  EndIf
```

Voir aussi

ExamineKeyboard() , KeyboardPushed() , KeyboardReleased()

OS Supportés

Tous

108.4 KeyboardMode

Syntaxe

```
KeyboardMode(Mode)
```

Description

Modifie le comportement du clavier.

Arguments

Mode Peut être une combinaison de :

```
#PB_Keyboard_Qwerty
          : Le clavier
 ignore l'agencement des
 touches propres au
 langage de l'utilisateur
 et utilise toujours
 l'agencement QWERTY
 (mode par défaut).
#PB_Keyboard_International
  : Le clavier utilise
 l'agencement des touches
 propres au langage de
 l'utilisateur (utile
 pour les claviers non
 QWERTY).
#PB_Keyboard_AllowSystemKeys:
 Les touches systèmes de
 l'OS ne sont pas
 bloquées (comme Win+R
 etc.). Cette option peut
 être gênante en mode
 plein écran

                                si
 l'utilisateur presse
 l'une de ces touches
 accidentellement.
```

Valeur de retour

Aucune.

Remarques

Cette commande affecte le résultat de KeyboardPushed() et KeyboardReleased() .

Voir aussi

ExamineKeyboard() , InitKeyboard() , KeyboardPushed() , KeyboardReleased()

OS Supportés

Windows

108.5 KeyboardPushed

Syntaxe

```
Resultat =
    KeyboardPushed(ToucheID)
```

Description

Teste l'état enfoncé ou non d'une touche du clavier.

Arguments

ToucheID L'identifiant PureBasic de la touche à tester.
Valeurs possibles :

```
 #PB_Key_All    ; Toutes
les touches sont testées.
                ; Utile
pour coder un "Appuyer
sur une touche..."
 #PB_Key_1
 #PB_Key_2
 #PB_Key_3
 #PB_Key_4
 #PB_Key_5
 #PB_Key_6
 #PB_Key_7
 #PB_Key_8
 #PB_Key_9
 #PB_Key_0

 #PB_Key_A
 #PB_Key_B
 #PB_Key_C
 #PB_Key_D
 #PB_Key_E
 #PB_Key_F
 #PB_Key_G
 #PB_Key_H
 #PB_Key_I
 #PB_Key_J
 #PB_Key_K
```

```
#PB_Key_L
#PB_Key_M
#PB_Key_N
#PB_Key_O
#PB_Key_P
#PB_Key_Q
#PB_Key_R
#PB_Key_S
#PB_Key_T
#PB_Key_U
#PB_Key_V
#PB_Key_W
#PB_Key_X
#PB_Key_Y
#PB_Key_Z

#PB_Key_Escape
#PB_Key_Minus
#PB_Key_Equals
#PB_Key_Back
#PB_Key_Tab
#PB_Key_LeftBracket
#PB_Key_RightBracket
#PB_Key_Return
#PB_Key_LeftControl
#PB_Key_SemiColon
#PB_Key_Apostrophe
#PB_Key_Grave
#PB_Key_LeftShift
#PB_Key_BackSlash
#PB_Key_Comma
#PB_Key_Period
#PB_Key_Slash
#PB_Key_RightShift
#PB_Key_Multiply
#PB_Key_LeftAlt
#PB_Key_Space
#PB_Key_Capital
#PB_Key_F1
#PB_Key_F2
#PB_Key_F3
#PB_Key_F4
#PB_Key_F5
#PB_Key_F6
#PB_Key_F7
#PB_Key_F8
#PB_Key_F9
#PB_Key_F10
#PB_Key_F11
#PB_Key_F12
#PB_Key_NumLock
#PB_Key_Scroll
#PB_Key_Pad0
#PB_Key_Pad1
#PB_Key_Pad2
#PB_Key_Pad3
#PB_Key_Pad4
#PB_Key_Pad5
#PB_Key_Pad6
```

```
#PB_Key_Pad7
#PB_Key_Pad8
#PB_Key_Pad9
#PB_Key_Add
#PB_Key_Subtract
#PB_Key_Decimal
#PB_Key_PadEnter
#PB_Key_RightControl
#PB_Key_PadComma
#PB_Key_Divide
#PB_Key_RightAlt
#PB_Key_Pause
#PB_Key_Home
#PB_Key_Up
#PB_Key_Down
#PB_Key_Left
#PB_Key_Right
#PB_Key_End
#PB_Key_PageUp
#PB_Key_PageDown
#PB_Key_Insert
#PB_Key_Delete
```

Valeur de retour

Renvoie une valeur non nulle si la touche est pressée, zéro sinon.

Remarques

- Un nombre quelconque de touches peut être pressé simultanément.
- Si vous voulez tester si une touche spécifique a été appuyée puis relâchée, utiliser la commande KeyboardReleased() .
- La fonction ExamineKeyboard() doit être appelée avant pour mettre à jour l'état du clavier.
- Le comportement du clavier peut être changé avec la commande KeyboardMode() .

Exemple

```
If InitSprite() And InitKeyboard() And OpenScreen(800,600,16,"")
 Repeat
   FlipBuffers()

   If StartDrawing(ScreenOutput())
     DrawText(0, 0, "Appuyez sur la touche [Echap] pour quitter")
     StopDrawing()
   EndIf

```

```
10        ExamineKeyboard()
11        If
      KeyboardPushed(#PB_Key_Escape)
            ; Appuyez sur la touche
      [Echap] pour quitter
12          End
13        EndIf
14      ForEver
15    EndIf
```

Voir aussi

ExamineKeyboard() , KeyboardInkey() , KeyboardReleased()

OS Supportés

Tous

108.6 KeyboardReleased

Syntaxe

```
Resultat =
    KeyboardReleased(ToucheID)
```

Description

Teste l'état relaché ou non d'une touche du clavier.

Arguments

ToucheID L'identifiant PureBasic de la touche à tester.
Valeurs possibles :

```
    #PB_Key_All    ; Toutes
    les touches sont testées.
                   ; Utile
    pour coder un "Appuyer
    sur une touche..."
    #PB_Key_1
    #PB_Key_2
    #PB_Key_3
    #PB_Key_4
    #PB_Key_5
    #PB_Key_6
    #PB_Key_7
    #PB_Key_8
    #PB_Key_9
    #PB_Key_0

    #PB_Key_A
    #PB_Key_B
    #PB_Key_C
    #PB_Key_D
    #PB_Key_E
```

```
#PB_Key_F
#PB_Key_G
#PB_Key_H
#PB_Key_I
#PB_Key_J
#PB_Key_K
#PB_Key_L
#PB_Key_M
#PB_Key_N
#PB_Key_O
#PB_Key_P
#PB_Key_Q
#PB_Key_R
#PB_Key_S
#PB_Key_T
#PB_Key_U
#PB_Key_V
#PB_Key_W
#PB_Key_X
#PB_Key_Y
#PB_Key_Z

#PB_Key_Escape
#PB_Key_Minus
#PB_Key_Equals
#PB_Key_Back
#PB_Key_Tab
#PB_Key_LeftBracket
#PB_Key_RightBracket
#PB_Key_Return
#PB_Key_LeftControl
#PB_Key_SemiColon
#PB_Key_Apostrophe
#PB_Key_Grave
#PB_Key_LeftShift
#PB_Key_BackSlash
#PB_Key_Comma
#PB_Key_Period
#PB_Key_Slash
#PB_Key_RightShift
#PB_Key_Multiply
#PB_Key_LeftAlt
#PB_Key_Space
#PB_Key_Capital
#PB_Key_F1
#PB_Key_F2
#PB_Key_F3
#PB_Key_F4
#PB_Key_F5
#PB_Key_F6
#PB_Key_F7
#PB_Key_F8
#PB_Key_F9
#PB_Key_F10
#PB_Key_F11
#PB_Key_F12
#PB_Key_NumLock
#PB_Key_Scroll
#PB_Key_Pad0
```

```
#PB_Key_Pad1
#PB_Key_Pad2
#PB_Key_Pad3
#PB_Key_Pad4
#PB_Key_Pad5
#PB_Key_Pad6
#PB_Key_Pad7
#PB_Key_Pad8
#PB_Key_Pad9
#PB_Key_Add
#PB_Key_Subtract
#PB_Key_Decimal
#PB_Key_PadEnter
#PB_Key_RightControl
#PB_Key_PadComma
#PB_Key_Divide
#PB_Key_RightAlt
#PB_Key_Pause
#PB_Key_Home
#PB_Key_Up
#PB_Key_Down
#PB_Key_Left
#PB_Key_Right
#PB_Key_End
#PB_Key_PageUp
#PB_Key_PageDown
#PB_Key_Insert
#PB_Key_Delete
```

Valeur de retour

Renvoie une valeur non nulle si la touche a été appuyée puis relâchée, zéro sinon.

Exemple

```
1   If InitSprite() And
     InitKeyboard() And
     OpenScreen(800,600,16,"")
2     Paused = #False
3     Repeat
4       FlipBuffers()
5
6       If
    StartDrawing(ScreenOutput())
7
8         ExamineKeyboard()
9         If
    KeyboardReleased(#PB_Key_P)
10          If Paused = #False
11            Paused = #True
12          Else
13            Paused = #False
14          EndIf
15        EndIf
16
17        DrawingMode(0)
18
```

```
19          If Paused = #False
20            DrawText(20, 20,
    "Le programme tourne...
            ")
21          Else
22            DrawText(20, 20,
    "Le programme est à
    l'arrêt...")
23          EndIf
24
25          StopDrawing()
26        EndIf
27      Until
    KeyboardPushed(#PB_Key_Escape)
28    EndIf
```

Voir aussi

ExamineKeyboard() , KeyboardInkey() , KeyboardPushed()

OS Supportés

Tous

Chapitre 109

Library

Généralités

Les bibliothèques sont des composants partagés du système d'exploitation contenant des fonctions spécifiques à l'intention des programmeurs. Par exemple, une bibliothèque peut contenir des commandes pour traiter et manipuler facilement des images. Chaque système d'exploitation a ses propres bibliothèques partagées pour faciliter la vie du programmeur. Avec PureBasic, il est possible d'utiliser ces bibliothèques tierces facilement et dynamiquement !
La raison pour laquelle les bibliothèques sont si importantes, c'est qu'elles évitent au programmeur d'avoir à reprogrammer sans cesse les mêmes routines. Elle sont conçues pour pouvoir être utilisées par n'importe quel programme, en utilisant un minimum de mémoire (si par exemple 10 programmes utilisent la même bibliothèque, elle ne sera chargée qu'une seule fois en mémoire).
Un autre avantage est la possibilité de faire évoluer un programme sans changer son exécutable (en mettant à jour uniquement la bibliothèque).
Sous Windows, ces bibliothèques sont bien connues sous le nom de 'DLL' (Dynamically Linked Library).

OS Supportés

Tous

109.1 CloseLibrary

Syntaxe

```
CloseLibrary(#Bibliotheque)
```

Description

Ferme une bibliotheque et libère la mémoire associée.

Arguments

#Bibliotheque Le numéro de la bibliothèque.
Si #PB_All est spécifié, toutes les autres bibliothèques seront fermées.

Valeur de retour

Aucune.

Remarques

Toutes les bibliothèques ouvertes sont automatiquement fermées quand le programme se termine.

Voir aussi

OpenLibrary()

OS Supportés

Tous

109.2 CallCFunction

Syntaxe

```
Resultat =
    CallCFunction(#Bibliotheque,
    NomFonction$ [,Parametre1
    [, Parametre2...]])
```

Description

Appelle une fonction d'une bibliothèque à la manière du language C pour les paramètres.

Arguments

#Bibliotheque La bibliothèque à utiliser.

NomFonction$ Le nom de la fonction à appeler (sensible à la casse).

Parametre1, Parametre2... (optionnel) Les paramètres de la fonction.
Le nombre de paramètres doivent correspondre aux paramètres de la fonction appelée.
Le nombre maximum de paramètres pris en charge est de 20.

Valeur de retour

Renvoie la valeur de retour de la fonction appelée ou zéro si la bibliothèque ne contient pas la fonction demandée.

Remarques

Attention, cette fonction n'est utile que si la commande appelée a été déclarée avec la convention 'CDECL' ce qui n'est pas le standard sous Windows. La plupart des DLL sous Windows nécessite l'utilisation de CallFunction() . Par contre sous linux, toutes les fonctions des bibliothèques partagées sont en 'CDECL'.
Pour appeler une fonction qui utilise la convention d'appel 'stdcall', utilisez la fonction CallFunction() .
Cette fonction n'est pas très flexible car elle n'accepte pas les paramètres de type double et quad, et ne peut pas renvoyer des valeurs de type double, quad ou float. Il est vivement conseillé d'utiliser les prototypes à la place.

Voir aussi

CallFunction() , GetFunction() , prototypes

OS Supportés

Tous

109.3 CallCFunctionFast

Syntaxe

```
Resultat =
    CallCFunctionFast(*PointeurFonction
    [,Parametre1 [,
    Parametre2...]])
```

Description

Appelle une fonction directement, en utilisant son adresse.
La fonction doit utiliser l'appel cdecl (la convention utilisée par le langage C).

Arguments

***PointeurFonction** L'adresse de la fonction à appeler.
Son pointeur est obtenu par GetFunction() , GetFunctionEntry() ou par LibraryFunctionAddress() .

L'utilisation de cette fonction est la méthode la plus rapide pour appeler des fonctions d'une bibliothèque, en particulier lorsque les résultats ont été obtenus avec GetFunction() ou LibraryFunctionAddress(). Cela est dû au fait que cette fonction n'est pas tenue de rechercher le nom de la fonction de bibliothèque.

Parametre1, Parametre2... (optionnel)
Les paramètres de la fonction.
Le nombre maximum de paramètres est 20.

Le nombre de paramètres autorisés est quelconque mais doit correspondre exactement aux besoins de la fonction. Par exemple, si une fonction nécessite 2 paramètres, alors 2 paramètres doivent être passés même si les valeurs de ces 2 paramètres sont nulles.

Valeur de retour

Renvoie la valeur de retour de la fonction appelée.

Remarques

Attention, cette fonction n'est utile que si la commande appelée a été déclarée avec la convention 'CDECL' ce qui n'est pas le standard sous Windows. La plupart des DLL sous Windows nécessite l'utilisation de CallFunction() . Par contre sous linux, toutes les fonctions des bibliothèques partagées sont en 'CDECL'.
Pour appeler une fonction qui utilise la convention d'appel stdcall, utilisez la fonction CallFunction() .
Note : Cette fonction n'est pas très flexible car elle n'accepte pas les paramètres de type string, float, double et quad, et ne peut pas renvoyer des valeurs de type string, float, double ou quad. Il est vivement conseillé d'utiliser les prototypes .

Exemple

```
1  ProcedureC Function1()
2      Debug "J'appelle la
     Fonction par son nom"
3  EndProcedure
4
5  NewMap *FuncPtr()
6  *FuncPtr("Function1") =
     @Function1()
7
```

```
8
9   CallCFunctionFast(*FuncPtr("Function1"))
```

Voir aussi

GetFunction() , CallFunctionFast() , prototypes

OS Supportés

Tous

109.4 CallFunction

Syntaxe

```
Resultat =
    CallFunction(#Bibliotheque,
    NomFonction$ [,Parametre1
    [, Parametre2...]])
```

Description

Appelle une fonction d'une bibliothèque (en utilisant son nom).

Arguments

#Bibliotheque La bibliothèque à utiliser.

NomFonction$ Le nom de la fonction à appeler (sensible à la casse).

Parametre1, Parametre2... (optionnel) Les paramètres de la fonction.
Le nombre de paramètres doivent correspondre aux paramètres de la fonction appelée.
Le nombre maximum de paramètres pris en charge est de 20.

Valeur de retour

Renvoie la valeur de retour de la fonction appelée, zéro sinon.

Remarques

Pour appeler une fonction qui utilise la convention 'cdecl', utiliser la fonction CallCFunction() .
Cette fonction n'est pas très flexible car elle n'accepte pas les paramètres de type string, float, double et quad, et ne peut pas renvoyer des valeurs de type string, float, double ou quad. Il est vivement conseillé d'utiliser les prototypes à la place.

Voir aussi

CallCFunction() , GetFunction() , prototypes

OS Supportés

Tous

109.5 CallFunctionFast

Syntaxe

```
Resultat =
    CallFunctionFast(*PointeurFonction
    [,Parametre1 [,
    Parametre2...]])
```

Description

Appelle une fonction directement, en utilisant son adresse.
La fonction doit utiliser l'appel stdcall (la plupart des DLL de Windows).

Arguments

***PointeurFonction** L'adresse de la fonction à appeler.
Son pointeur est obtenu par GetFunction() , GetFunctionEntry() ou par LibraryFunctionAddress() .

L'utilisation de cette fonction est la méthode la plus rapide pour appeler des fonctions d'une bibliothèque, en particulier lorsque les résultats ont été obtenus avec GetFunction() ou LibraryFunctionAddress(). Cela est dû au fait que cette fonction n'est pas tenue de rechercher le nom de la fonction de bibliothèque.

Parametre1, Parametre2... (optionnel) Les paramètres de la fonction.
Le nombre maximum de paramètres est 20.

Le nombre de paramètres doit correspondre exactement aux besoins de la fonction. Par exemple, si une fonction nécessite 2 paramètres, alors 2 paramètres doivent être passés même si les valeurs de ces 2 paramètres sont nulles.

Valeur de retour

Renvoie la valeur de retour de la fonction appelée.

Remarques

Pour appeler une fonction qui utilise la convention d'appel 'cdecl', utiliser la fonction CallCFunctionFast() .
Note : Cette fonction n'est pas très flexible car elle n'accepte pas les paramètres de type string, float, double et quad, et ne peut pas renvoyer des valeurs de type string, float, double ou quad. Il est vivement conseillé d'utiliser les prototypes à la place.

Voir aussi

CallCFunctionFast() , GetFunction() , prototypes

OS Supportés

Tous

109.6 CountLibraryFunctions

Syntaxe

```
Resultat =
    CountLibraryFunctions(#Bibliotheque)
```

Description

Renvoie le nombre de fonctions contenues dans une bibliothèque.

Arguments

#Bibliotheque La bibliothèque à utiliser.

Valeur de retour

Renvoie le nombre de fonctions disponibles dans la bibliothèque.

Remarques

La bibliothèque doit être ouverte avec OpenLibrary() avant de pouvoir utiliser cette commande.

OS Supportés

Windows, Linux

109.7 ExamineLibraryFunctions

Syntaxe

```
Resultat =
    ExamineLibraryFunctions(#Bibliotheque)
```

Description

Commence l'énumeration des fonctions contenues dans une bibliothèque.

Arguments

#Bibliotheque La bibliothèque à utiliser.

Valeur de retour

Renvoie une valeur non nulle si les fonctions peuvent être examinées, zéro sinon.

Remarques

La bibliothèque doit être ouverte avec OpenLibrary() avant de pouvoir utiliser cette commande.
L'énumeration peut continuer avec NextLibraryFunction() et les commandes LibraryFunctionName() et LibraryFunctionAddress() permettant alors de récuperer le nom et l'adresse de chaque fonction.

Voir aussi

NextLibraryFunction() , LibraryFunctionAddress() , LibraryFunctionName()

OS Supportés

Windows, Linux

109.8 GetFunction

Syntaxe

```
Resultat =
    GetFunction(#Bibliotheque,
    NomFonction$)
```

Description

Teste la présence d'une fonction dans une bibliothèque.

Arguments

#Bibliotheque La bibliothèque à utiliser.

NomFonction$ Le nom de la fonction à appeler (sensible à la casse).

Valeur de retour

Renvoie l'adresse de la fonction dans la bibliothèque en cas de succès ou zéro si la bibliothèque ne contient pas une fonction portant ce nom.

Remarques

La bibliothèque doit préalablement être ouverte avec OpenLibrary() .
La fonction peut être appelée par son adresse en utilisant un prototype .
Les fonctions CallFunctionFast() et CallCFunctionFast() peuvent également être utilisées pour cela, mais l'utilisation des prototypes est recommandée car ils sont plus souples.
Voir les prototypes pour un exemple.

Voir aussi

GetFunctionEntry() , CallFunctionFast() , CallCFunctionFast() , prototypes

OS Supportés

Tous

109.9 GetFunctionEntry

Syntaxe

```
Resultat =
    GetFunctionEntry(#Bibliotheque,
    Index)
```

Description

Teste si un numéro de fonction est présent dans une bibliothèque.
Cela permet de tester la présence d'une fonction par sa position dans la bibliothèque plutôt que par son nom.

Arguments

#Bibliotheque La bibliothèque à utiliser.

Index Un nombre représentant l'index de la fonction dans la bibliothèque.
La première fonction est à l'index 1.

Valeur de retour

Renvoie l'adresse de la fonction en cas de succès ou zéro si la bibliothèque ne contient pas de fonction à cet index.

Remarques

La bibliothèque doit préalablement être ouverte avec OpenLibrary() ,
Cette commande peut être utile pour accéder aux fonctions API non documentées, qui n'ont pas de nom de fonctions attribués.
La fonction peut être appelée par son adresse en utilisant un prototype .
Les fonctions CallFunctionFast() et CallCFunctionFast() peuvent également être utilisées pour cela, mais l'utilisation des prototypes est recommandée car ils sont plus souples.
Voir les prototypes pour un exemple.

Voir aussi

GetFunction() , CallFunctionFast() , CallCFunctionFast() , prototypes

OS Supportés

Windows

109.10 IsLibrary

Syntaxe

```
Resultat =
    IsLibrary(#Bibliotheque)
```

Description

Teste si une bibliothèque est correctement initialisée.

Arguments

#Bibliotheque La bibliothèque à tester.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage.
C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.

Voir aussi

OpenLibrary() , CloseLibrary()

OS Supportés

Tous

109.11 LibraryFunctionAddress

Syntaxe

```
Resultat =
    LibraryFunctionAddress()
```

Description

Renvoie l'adresse de la fonction de la bibliothèque en cours d'examen avec les commandes ExamineLibraryFunctions() et NextLibraryFunction() .

Arguments

Aucun.

Valeur de retour

Renvoie l'adresse en mémoire de la fonction de la bibliothèque, qui peut être utilisée avec les commandes CallFunctionFast() ou CallCFunctionFast() .
Les fonctions CallFunctionFast() et CallCFunctionFast() peuvent également être utilisées pour cela, mais l'utilisation des prototypes est recommandée car ils sont plus souples.
Voir les prototypes pour un exemple.

Voir aussi

ExamineLibraryFunctions() , NextLibraryFunction() , LibraryFunctionName() , CallFunctionFast() , CallCFunctionFast()

OS Supportés

Windows, Linux

109.12 LibraryFunctionName

Syntaxe

```
Resultat\$ =
    LibraryFunctionName()
```

Description

Renvoie le nom de la fonction de la bibliothèque en cours d'examen avec les commandes ExamineLibraryFunctions() et NextLibraryFunction() .

Arguments

Aucun.

Valeur de retour

Renvoie le nom de la fonction de la bibliothèque, qui peut être utilisé avec les commandes CallFunction() ou CallCFunction() .

Voir aussi

ExamineLibraryFunctions() , NextLibraryFunction() , LibraryFunctionAddress()

OS Supportés

Windows, Linux

109.13 LibraryID

Syntaxe

```
Resultat =
    LibraryID(#Bibliotheque)
```

Description

Renvoie l'identifiant unique du système (souvent appelé 'handle') d'une bibliothèque.

Arguments

#Bibliotheque La bibliothèque à tester.

Valeur de retour

Renvoie l'identifiant système de la bibliothèque.

OS Supportés

Tous

109.14 NextLibraryFunction

Syntaxe

```
Resultat =
    NextLibraryFunction()
```

Description

Cette fonction ne peut être utilisée qu'après l'appel avec succès de la commande ExamineLibraryFunctions() . Elle permet d'examiner une à une les fonctions contenues dans la bibliothèque. Le nom et l'adresse de chaque fonction peuvent être obtenus avec les commandes LibraryFunctionName() et LibraryFunctionAddress() .

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle si la fonction suivante a été trouvée ou zéro s'il n'y a pas d'autres fonctions qui seront examinées.

Voir aussi

ExamineLibraryFunctions() , LibraryFunctionName() , LibraryFunctionAddress()

OS Supportés

Windows, Linux

109.15 OpenLibrary

Syntaxe

```
Resultat =
    OpenLibrary(#Bibliotheque,
    NomFichier$)
```

Description

Ouvre une bibliothèque partagée afin que ses fonctions puissent être consultées.

Arguments

#Bibliotheque La bibliothèque à tester. #PB_Any peut être utilisé en tant que paramètre pour générer automatiquement ce numéro.

NomFichier$ Le nom du fichier de la bibliothèque à charger.

Si le nom de fichier ne comporte pas de chemin, alors le système d'exploitation recherche la bibliothèque dans ses dossiers systèmes, le répertoire des applications et le répertoire courant.

Valeur de retour

Renvoie une valeur non nulle si la bibliothèque a été ouverte avec succès, zéro sinon.
Si #PB_Any a été utilisé à la place de #Bibliothèque alors le numéro généré pour le bibliothèque est retourné en cas de succès.

Voir aussi

CloseLibrary() , GetFunction() , prototypes

OS Supportés

Tous

Chapitre 110

Light

Généralités

Les lumières sont des composants essentiels des mondes 3D. Comme dans le monde réel, il est possible de créer toutes sortes d' éclairages comme spots, etc et de modifier leurs attributs tels que la couleur, la luminosité, la direction du faisceau, etc. InitEngine3D() doit être appelé avec succès avant de pouvoir utiliser les commandes relatives aux lumières.

OS Supportés

Tous

110.1 CopyLight

Syntaxe

```
Resultat =
    CopyLight(#Lumiere,
    #NouvelleLumiere)
```

Description

Crée une copie d'une lumière.

Arguments

#Lumiere La lumière à copier.

#NouvelleLumiere Numéro de la nouvelle lumière.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.
Tous les attributs de la #Lumiere (couleur, position, ...) sont dupliqués.

Remarques

Si le numéro #NouvelleLumiere est déjà créé alors il est automatiquement supprimé et remplacé par le nouveau.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé comme paramètre alors le numéro est renvoyé dans 'Resultat'.

Voir aussi

CreateLight() , FreeLight()

OS Supportés

Tous

110.2 CreateLight

Syntaxe

```
Resultat =
    CreateLight(#Lumiere,
    Couleur [, X.f, Y.f, Z.f
    [, Options]])
```

Description

Crée une nouvelle lumière.

Arguments

#Lumiere La lumière à utiliser.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Couleur Couleur de la nouvelle lumière.
RGB() peut être utilisée pour obtenir facilement la couleur désirée.

X.f, Y.f, Z.f (optionnel) Position de la nouvelle lumière.
La nouvelle lumière est créée par défaut à la position 0.0,0.0,0.0.

Options (optionnel)

```
      #PB_Light_Point
            : Créer un point
    de lumière (Par défaut)
    (la lumière est émise
    dans toutes les
    directions).
   #PB_Light_Directional   :
    Crée une lumière
    directionnelle.
   #PB_Light_Spot          :
    Crée une lumière de type
    spot.
                              SpotLightRange()
  peut être utilisé pour
    modifier le comportement
    de la lumière.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si la lumière était déjà créée alors elle est automatiquement supprimée et remplacée par la nouvelle.

Exemple : :

```
1  CreateLight(0,
     RGB(255,0,0)) ; Crée une
     lumière rouge
2
3  CreateLight(1,
     RGB(0,255,0),  0, 100.7,
     50) ; Crée une lumière
     verte à la position (0,
     100.7, 50)
```

Voir aussi

FreeLight() , HideLight()

OS Supportés

Tous

110.3 FreeLight

Syntaxe

```
FreeLight(#Lumiere)
```

Description

Supprime une lumière.

Arguments

#Lumiere La lumière à supprimer. Si #PB_All est spécifié, toutes les lumières restantes sont libérées.

Valeur de retour

Aucune.

Remarques

Toutes les lumières restantes sont automatiquement supprimées quand le programme se termine.

Voir aussi

CreateLight()

OS Supportés

Tous

110.4 HideLight

Syntaxe

```
HideLight(#Lumiere, Etat)
```

Description

Affiche ou cache une lumière.

Arguments

#**Lumiere** La lumière à utiliser.

Etat #True : la lumière est cachée
#False: la lumière est affichée

Valeur de retour

Aucune.

Voir aussi

CreateLight() , FreeLight()

OS Supportés

Tous

110.5 IsLight

Syntaxe

```
Resultat = IsLight(#Lumiere)
```

Description

Teste si une lumière est correctement initialisée.

Arguments

#**Lumiere** La lumière à tester.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.

Voir aussi

CreateLight() , LightID()

OS Supportés

Tous

110.6 GetLightColor

Syntaxe

```
Resultat =
    GetLightColor(#Lumiere,
    Type)
```

Description

Renvoie la couleur d'une lumière.

Arguments

#**Lumiere** La lumière à utiliser.

Type

```
      #PB_Light_DiffuseColor
    : Renvoie la valeur de
    la couleur de diffusion
    de la lumière (default)
   #PB_Light_SpecularColor:
    Renvoie la valeur de la
    couleur spéculaire de la
    lumière
```

Valeur de retour

Renvoie la valeur de la couleur au format RGB.

Voir aussi

SetLightColor()

OS Supportés

Tous

110.7 SetLightColor

Syntaxe

```
SetLightColor(#Lumiere, Type,
    Couleur)
```

Description

Modifie la valeur de la couleur d'une lumière.

Arguments

#**Lumiere** La lumière à utiliser.

Type
```
      #PB_Light_DiffuseColor
    : Change la valeur de la
    couleur de diffusion.
  #PB_Light_SpecularColor:
    Change la valeur de la
    couleur speculaire.
```

Couleur La nouvelle valeur de la couleur au format RGB.
Elle peut être facilement créée avec RGB() .

Valeur de retour

Aucune.

Voir aussi

GetLightColor

OS Supportés

Tous

110.8 SpotLightRange

Syntaxe

```
SpotLightRange(#Lumiere,
    AngleInterne, AngleExterne
    [, Attenuation])
```

Description

Modifie le comportement d'une lumière de type 'Spot'.

Arguments

#Lumiere La lumière à utiliser.

AngleInterne Angle interne de la lumière.

AngleExterne Angle externe de la lumière.

Attenuation (optionnel) Atténuation de la lumière en fonction de la distance.
Une valeur de 1 signifie une diminution linéaire.
Une valeur inférieure à 1 signifie une lente atténuation.
Une valeur supérieure à 1 signifie une rapide atténuation.

Valeur de retour

Aucune.

Remarques

Attention : La lumière doit être créé avec l'option #PB_Light_Spot.

Voir aussi

CreateLight()

OS Supportés

Tous

110.9 LightLookAt

Syntaxe

```
LightLookAt(#Lumiere, X.f,
    Y.f, Z.f)
```

Description

Modifie l'orientation d'une lumière dans le monde et pointe vers le point X, Y, Z.

Arguments

#Lumiere La lumière à utiliser.

X.f, Y.f, Z.f Coordonnées du point.

Valeur de retour

Aucune.

Voir aussi

LightDirection() , LightDirectionX() , LightDirectionY() , LightDirectionZ()

OS Supportés

Tous

110.10 DisableLightShadow

Syntaxe

```
DisableLightShadow(#Lumiere,
    Etat)
```

Description

Désactive ou active la projection des ombres.

Arguments

#**Lumiere** La lumière à utiliser.

Etat #True : L'ombre est désactivée.
#False: L'ombre est activée.

Valeur de retour

Aucune.

Voir aussi

LightAttenuation()

OS Supportés

Tous

110.11 MoveLight

Syntaxe

```
MoveLight(#Lumiere, X.f, Y.f,
    Z.f [, Mode])
```

Description

Déplace une lumière dans le monde 3D.

Arguments

#**Lumiere** La lumière à utiliser.

X.f, Y.f, Z.f Déplacement de la lumière. Le déplacement est relatif à la position actuelle de la lumière.

Mode (optionnel) Peut être une des valeurs suivantes :

```
#PB_Relative: Déplacement
 relatif, à partir de la
 position actuelle de la
 lumière (par défaut).
#PB_Absolute: Déplacement
 absolu à la position
 spécifiée.
```

combinée avec l'une des valeurs suivantes :

```
#PB_Local : Déplacement
 local.
#PB_Parent: Déplacement
 par rapport à la
 position du parent.
#PB_World : Déplacement
 par rapport au monde.
```

Valeur de retour

Aucune.

Voir aussi

LightX() , LightY() , LightZ() , RotateLight()

OS Supportés

Tous

110.12 LightDirection

Syntaxe

```
LightDirection(#Lumiere, X.f,
    Y.f, Z.f)
```

Description

Change la direction de la lumière.

Arguments

#Lumiere La lumière à utiliser.

X.f, Y.f, Z.f Direction du vecteur. Généralement une valeur pour X.f, Y.f, Z.f, comprise entre -1.0 et 1.0, sinon il sera automatiquement normalisé.

Valeur de retour

Aucune.

Remarques

La position de la lumière n'est pas modifiée.

Voir aussi

LightDirectionX() , LightDirectionY() , LightDirectionZ()

OS Supportés

Tous

110.13 LightDirectionX

Syntaxe

```
Resultat.f =
    LightDirectionX(#Lumiere
    [, Mode])
```

Description

Renvoie le vecteur de direction 'X' d'une lumière.

Arguments

#**Lumiere** La lumière à utiliser.

Mode (optionnel) Le mode d'obtention du vecteur de direction de la caméra. Peut être l'une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 direction de la caméra
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 direction de la caméra
 par rapport à son parent.
```

Valeur de retour

Renvoie le vecteur de direction 'X' de la lumière.
Cette valeur est toujours comprise entre -1.0 et 1.0.

Voir aussi

LightDirection() , LightDirectionY() , LightDirectionZ()

OS Supportés

Tous

110.14 LightDirectionY

Syntaxe

```
Resultat.f =
    LightDirectionY(#Lumiere
    [, Mode])
```

Description

Renvoie le vecteur de direction 'Y' d'une lumière.

Arguments

#**Lumiere** La lumière à utiliser.

Mode (optionnel) Le mode d'obtention du vecteur de direction de la caméra. Peut être l'une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 direction de la caméra
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 direction de la caméra
 par rapport à son parent.
```

Valeur de retour

Renvoie le vecteur de direction 'Y' de la lumière.
Cette valeur est toujours comprise entre -1.0 et 1.0.

Voir aussi

LightDirection() , LightDirectionX() , LightDirectionZ()

OS Supportés

Tous

110.15 LightDirectionZ

Syntaxe

```
Resultat.f =
    LightDirectionZ(#Lumiere
    [, Mode])
```

Description

Renvoie le vecteur de direction 'Z' d'une lumière.

Arguments

#**Lumiere** La lumière à utiliser.

Mode (optionnel) Le mode d'obtention du vecteur de direction de la caméra. Peut être l'une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 direction de la caméra
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 direction de la caméra
 par rapport à son parent.
```

Valeur de retour

Renvoie le vecteur de direction 'Z' de la lumière.
Cette valeur est toujours comprise entre -1.0 et 1.0.

Voir aussi

LightDirection() , LightDirectionX() , LightDirectionY()

OS Supportés

Tous

110.16 LightX

Syntaxe

```
Resultat = LightX(#Lumiere [,
    Mode])
```

Description

Renvoie la position actuelle d'une lumière dans le monde.

Arguments

#**Lumiere** La lumière à utiliser.

Mode (optionnel) Le mode d'obtention du vecteur de direction de la caméra. Peut être l'une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 direction de la caméra
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 direction de la caméra
 par rapport à son parent.
```

Valeur de retour

Renvoie la position de la lumière en 'X'.

Voir aussi

LightY() , LightZ() , MoveLight()

OS Supportés

Tous

110.17 LightY

Syntaxe

```
Resultat = LightY(#Lumiere [,
    Mode])
```

Description

Renvoie la position actuelle d'une lumière dans le monde.

Arguments

#**Lumiere** La lumière à utiliser.

Mode (optionnel) Le mode d'obtention du vecteur de direction de la caméra. Peut être l'une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 direction de la caméra
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 direction de la caméra
 par rapport à son parent.
```

Valeur de retour

Renvoie la position de la lumière en 'Y'.

Voir aussi

LightX() , LightZ() , MoveLight()

OS Supportés

Tous

110.18 LightZ

Syntaxe

```
Resultat = LightZ(#Lumiere [,
    Mode])
```

Description

Renvoie la position actuelle d'une lumière dans le monde.

Arguments

#**Lumiere** La lumière à utiliser.

Mode (optionnel) Le mode d'obtention du vecteur de direction de la caméra. Peut être l'une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 direction de la caméra
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 direction de la caméra
 par rapport à son parent.
```

Valeur de retour

Renvoie la position de la lumière en 'Z'.

Voir aussi

LightX() , LightY() , MoveLight()

OS Supportés

Tous

110.19 LightAttenuation

Syntaxe

```
LightAttenuation(#Lumiere,
    Champ, Attenuation.f)
```

Description

Change l'attenuation d'une lumière.

Arguments

#**Lumiere** La lumière à utiliser.

#**Champ** Le champ (dans l'unité du monde) au-delà duquel la lumière n'éclaire plus le monde.

Attenuation L'atténuation globale de la lumière.
Une valeur de 0.0 signifie aucune atténuation globale (l'atténuation est fonction du champ).
Elle peut être utilisée pour régler la luminosité de la lumière.

Valeur de retour

Aucune.

Voir aussi

SetLightColor()

OS Supportés

Tous

110.20 RotateLight

Syntaxe

```
RotateLight(#Lumiere, X.f,
    Y.f, Z.f [, Mode])
```

Description

Rotation d'une lumière

Arguments

#Lumiere La lumière à utiliser.

X.f, Y.f, Z.f Valeurs des angles de rotation.
Tous les angles sont en degrés.

Mode (optionnel)

```
      PB_Absolute: Rotation
    absolue (par défaut).
   PB_Relative: Rotation
    relative basée sur la
    rotation précédente.
```

Valeur de retour

Aucune.

Voir aussi

MoveLight()

OS Supportés

Tous

110.21 LightRoll

Syntaxe

```
Resultat.f =
    LightRoll(#Lumiere [,
    Mode])
```

Description

Renvoie le roulis d'une lumière.

Arguments

#**Lumiere** La lumière à utiliser.

Mode (optionnel) Peut être une des valeurs suivantes :

```
#True : La valeur brute,
  mais elle ne peut pas
  être utilisée avec
  RotateLight()
pour récupérer la même
  orientation (par défaut).
#False: Le roulis est
  ajusté, de sorte qu'il
  peut être réutilisé avec
  RotateLight()
pour récupérer la même
  orientation.
```

Valeur de retour

La valeur courante du roulis de la lumière. Valeur toujours comprise entre -180.0 et 180.0 degrés.

Voir aussi

LightYaw() , LightPitch()

OS Supportés

Tous

110.22 LightPitch

Syntaxe

```
Resultat.f =
    LightPitch(#Lumiere [,
    Mode])
```

Description

Renvoie le tangage d'une lumière.

Arguments

#**Lumiere** La lumière à utiliser.

Mode (optionnel) Peut être une des valeurs suivantes :

```
#True : La valeur brute,
  mais elle ne peut pas
  être utilisée avec
  RotateLight()
pour récupérer la même
  orientation (par défaut).
```

```
    #False: Le tangage est
      ajusté, de sorte qu'il
      peut être réutilisé avec
      RotateLight()
  pour récupérer la même
    orientation.
```

Valeur de retour

La valeur courante du tangage de la lumière.
Valeur toujours comprise entre -180.0 et 180.0 degrés.

Voir aussi

LightYaw() , LightRoll()

OS Supportés

Tous

110.23 LightYaw

Syntaxe

```
Resultat.f =
    LightYaw(#Lumiere [, Mode])
```

Description

Renvoie le lacet d'une lumière.

Arguments

#**Lumiere** La lumière à utiliser.

Mode (optionnel) Peut être une des valeurs suivantes :

```
    #True : La valeur brute,
      mais elle ne peut pas
      être utilisée avec
      RotateLight()
  pour récupérer la même
    orientation (par défaut).
    #False: Le lacet est
      ajusté, de sorte qu'il
      peut être réutilisé avec
      RotateLight()
  pour récupérer la même
    orientation.
```

Valeur de retour

La valeur courante du lacet de la lumière.
Valeur toujours comprise entre -180.0 et 180.0 degrés.

Voir aussi

LightPitch() , LightRoll()

OS Supportés

Tous

110.24 LightID

Syntaxe

```
LumiereID = LightID(#Lumiere)
```

Description

Renvoie l'identificateur système unique d'une lumière.

Arguments

#**Lumiere** La lumière à utiliser.

Valeur de retour

Renvoie le numéro de la lumière.

Voir aussi

CreateLight() , IsLight()

OS Supportés

Tous

Chapitre 111

List

Généralités

Les listes (aussi appelée listes chaînées) regroupent des objets qui sont alloués dynamiquement en fonction de vos besoins. Elles se présentent sous la forme de listes d'éléments totalement indépendants les uns des autres. Vous pouvez y ajouter un nombre infini d'éléments, insérer des éléments à une position choisie, en effacer, etc... Cette méthode de stockage d'informations est très pratique et très souple. C'est la meilleure solution pour gérer des données dont vous ne connaissez pas le nombre à l'avance.
Avant de travailler avec les listes, vous devez les déclarer. Cela se fait avec le mot-clef NewList . Les structures sont également fréquemment utilisées dans les listes.
Les listes peuvent être triées à l'aide des fonctions SortList() ou SortStructuredList() de la bibliothèque Sort et peuvent être aussi mélangées de façon aléatoire en utilisant RandomizeList() .
Pour analyser le contenu d'une liste, il est possible d'utiliser les boucles suivantes : For : Next , ForEach : Next , Repeat : Until ou While : Wend .
Le premier élément a la position 0, le deuxième la position 1, etc..
Les autres possibilités pour stocker des données multiples sont les Tableaux et les Maps .

; ; ; ; ; Tri & divers

SortList()
SortStructuredList()
RandomizeList()

OS Supportés

Tous

111.1 AddElement

Syntaxe

```
*Resultat =
    AddElement(Liste())
```

Description

Ajoute un nouvel élément après l'élément courant.

Arguments

Liste() La liste à utiliser.

Valeur de retour

Renvoie une valeur non nulle qui est l'adresse du nouvel élément si le nouvel élément a été ajouté, zéro sinon.
La valeur renvoyée est un pointeur sur le nouvel élément.

Remarques

Devient le premier élément si la liste était vide et il devient l'élément courant de la liste.

Exemple

```
1   ; La manière la plus simple
     d'utiliser AddElement
2   NewList simple.w()
3   AddElement(simple())     ;
     Crée le premier nouvel
     élément de la liste
4   simple() = 23
5
6   AddElement(simple())     ;
     La position courante est
     le premier élément, alors
     nous en ajoutons un à la
     deuxième position
7   simple() = 45
8
9
10  ; Ceci montre comment
     utiliser la valeur de
     retour de la fonction
     AddElement
11  NewList experimentes.l()
12  If
     AddElement(experimentes())
     <> 0
13    experimentes() = 12345
14  Else
```

```
15      MessageRequester("Erreur
        !", "Impossible d'allouer
        de la mémoire pour le
        nouvel élément",
        #PB_MessageRequester_OK)
16    EndIf
17
18
19    ; Une petite structure pour
        montrer l'utilisation du
        pointeur.
20    Structure Programmeur
21      Nom.s
22      Talent.b
23    EndStructure
24
25    NewList
        LesProgrammeurs.Programmeur()
         ; La liste qui stocke les
        éléments
26
27    *Element.Programmeur =
        AddElement(LesProgrammeurs())
28    If *Element<>0
29      *Element\Nom = "David"
30      *Element\Talent = 3    ;
        Celui-là, c'est un féru de
        PureBasic ! ;)
31    Else
32      MessageRequester("Erreur
        !", "Impossible d'allouer
        de la mémoire pour le
        nouvel élément",
        #PB_MessageRequester_OK)
33    EndIf
```

Voir aussi

InsertElement() , DeleteElement() , ClearList()

OS Supportés

Tous

111.2 ChangeCurrentElement

Syntaxe

```
ChangeCurrentElement(Liste(),
   *NouvelElement)
```

Description

Change l'élément courant de la liste.

Arguments

Liste() La liste à utiliser.

***NouvelElement** Le nouvel élément à placer à la position courante de la liste. C'est un pointeur vers un autre élément qui existe déjà dans la liste. Cette adresse doit être récupérée avec l'opérateur @ et le nom de la liste et pas autrement.

Valeur de retour

Aucune.

Remarques

Cette fonction est très utile pour mémoriser un élément et le réutiliser après avoir exécuté d'autres traitements

Exemple : Simple

```
1   *Ancien_Element =
     @MaListe()    ; On mémorise
     l'adresse de l'élément
     courant
2
3   ResetList(MaListe())
                ; Effectue une
     recherche de tous les
     éléments nommés
4   While
     NextElement(MaListe())    ;
     "Jean" et les change en "J"
5     If MaListe()\nom = "Jean"
6       MaListe()\nom = "J"
7     EndIf
8   Wend
9
10  ChangeCurrentElement(MaListe(),
     *Ancien_Element) ;
     Restitue l'ancien élément
     courant (mémorisé avant la
     recherche)
```

Exemple : Complet

```
1   NewList maList()
2
3   AddElement(maList())
4   maList() = 100
5
6   AddElement(maList())
7   maList() = 200
8   *element = @maList()
9
```

```
10  AddElement(maList())
11  maList() = 300
12
13  Debug maList()
      ; Affiche 300 (dernier
      élément)
14  ChangeCurrentElement(maList(),
      *element)    ; Restaure la
      position de la liste
15  Debug maList()
      ; Affiche 200
16
17  ForEach maList()
18    If @maList() = *element
19      Debug "élément: " +
      maList()           ;
      Affiche "élément: 200"
20    EndIf
21  Next
```

Voir aussi

SelectElement() , PushListPosition() , PopListPosition()

OS Supportés

Tous

111.3 ClearList

Syntaxe

```
ClearList(Liste())
```

Description

Efface tous les éléments d'une liste et libère la mémoire utilisée.

Arguments

Liste() La liste à utiliser.

Valeur de retour

Aucune.

Remarques

Après l'appel à cette fonction, la liste reste utilisable, mais elle ne contient plus d'éléments. PureBasic libèrera seulement la mémoire occupée par les éléments.

Si la liste a été utilisée pour stocker des objets dynamiques, il n'est pas possible de le détecter (en PureBasic ou dans un autre langage). Dans ce cas, il convient de libérer tous ces objets avant la destruction de la liste.

Exemple

```
1   NewList nombres.w()
2
3   ; Une petite boucle pour
     ajouter plusieurs éléments
     à la liste
4   For i=1 To 100
5     AddElement(nombres())
6     nombres() = i
7   Next
8
9   ; Preuve que les éléments
     ont été ajoutés à la liste
10  MessageRequester("Information",
     "Il y a
     "+Str(ListSize(nombres()))+"
     éléments dans la liste",
     #PB_MessageRequester_OK)
11
12  ; Effacer la liste et
     montrer que la liste est
     vraiment vide
13  ClearList(nombres())
14  MessageRequester("Information",
     "Il y a
     "+Str(ListSize(nombres()))+"
     élements dans la liste",
     #PB_MessageRequester_OK)
```

Voir aussi

DeleteElement() , FreeList()

OS Supportés

Tous

111.4 CopyList

Syntaxe

```
Resultat =
    CopyList(ListeSource(),
    ListeDestination())
```

Description

Copie tous les éléments d'une liste dans une autre.

Arguments

ListeSource() La liste à copier.

ListeDestination() La copie de la liste.
Tous les éléments présents avant la copie seront effacés.
Si les deux listes ne sont pas du même type (natif ou structuré) la copie ne se fera pas.

Valeur de retour

Renvoie une valeur non nulle si la copie a réussi, zéro sinon.

Exemple

```
1   NewList Amis$()
2   NewList AmisCopie$()
3
4   AddElement(Amis$())
5   Amis$() = "Jean"
6
7   AddElement(Amis$())
8   Amis$() = "Elise"
9
10  CopyList(Amis$(),
      AmisCopie$())
11
12  ForEach AmisCopie$()
13    Debug AmisCopie$()
14  Next
```

Voir aussi

CopyArray() , CopyMap()

OS Supportés

Tous

111.5 FreeList

Syntaxe

```
FreeList(Liste())
```

Description

Détruit la liste et libère toutes les ressources associées.

Arguments

Liste() La liste à détruire et ne peut être réutilisée, contrairement à ClearList() .

Valeur de retour

Aucune.

Remarques

Pour accéder à cette liste à nouveau, il faut la recréer avec NewList .

Voir aussi

ClearList()

OS Supportés

Tous

111.6 ListSize

Syntaxe

```
Resultat = ListSize(Liste())
```

Description

Renvoie le nombre d'éléments contenus dans la liste.

Arguments

Liste() La liste à utiliser.

Valeur de retour

Le nombre total d'éléments dans la liste.

Remarques

Cette fonction ne modifie pas l'élément courant.
Elle est très rapide (pas d'itération) et peut être utilisée sereinement pour savoir si une liste est vide ou non.

Exemple

```
1  NewList Compter.w()
2
3  ; Petite boucle pour
     ajouter quelques éléments
     à la liste.
4  For i=0 To 10
5    AddElement(Compter())
6    Compter() = i * 23
7  Next
8
```

```
9   ; Affiche le nombre
    d'éléments qu'il y a dans
    la liste. J'espère que
    vous avez pensé
10  ; à la même valeur que
    celle affichée par
    l'ordinateur ;)
11  MessageRequester("Information",
    "Il y a
    "+Str(ListSize(Compter()))+"
    éléments dans la liste",
    #PB_MessageRequester_OK)
```

Voir aussi

ListIndex()

OS Supportés

Tous

111.7 CountList

Syntaxe

```
Resultat = CountList(Liste())
```

Description

Attention
Cette fonction est dépréciée, elle sera peut-être supprimée dans une future version de PureBasic. Elle ne doit pas être utilisée dans du nouveau code.

Cette fonction a été remplacée par ListSize() .

OS Supportés

Tous

111.8 DeleteElement

Syntaxe

```
*Resultat =
    DeleteElement(Liste() [,
    Options])
```

Description

Supprime l'élément courant de la liste.

Arguments

Liste() La liste à utiliser.

Options (optionnel) Si ce paramètre est égal à 1 et que le premier élément est supprimé alors le nouvel élément courant sera le second élément.
Cette option veille à ce qu'il y ait toujours un élément en cours de validité après une suppression aussi longtemps qu'il y a encore des éléments de la liste.

Valeur de retour

Renvoie une valeur non nulle qui est l'adresse du nouvel élément courant.
Si la liste ne comporte aucun élément courant après la suppression, le résultat est 0.
La valeur renvoyée est un pointeur sur le nouvel élément.

Remarques

Fonctionnement : Après l'appel de cette fonction, le nouvel élément courant est celui qui précédait l'élément supprimé. Si l'élément supprimé était le premier, alors il n'y a plus d'élément courant (la position dans la liste est avant le premier élément, comme après un ResetList()) sauf si Option=1 car dans ce cas le nouvel élément serait celui qui était en seconde position.

Exemple

```
1   NewList personnes.s()
2
3   AddElement(personnes()) :
     personnes() = "Tom"
4   AddElement(personnes()) :
     personnes() = "Dick"
5   AddElement(personnes()) :
     personnes() = "Harry"
6   AddElement(personnes()) :
     personnes() = "Bob"
7
8   FirstElement(personnes())
         ; se déplace sur "Tom"
9   DeleteElement(personnes(),1)
      ; et le supprime.
     L'élément courant devient
     (paramètre 1 utilisé)
     celui qui suivait
10  MessageRequester("Information",
     "La première personne de
     la liste est
     "+personnes(),
     #PB_MessageRequester_OK)
```

```
11
12  LastElement(personnes())
        ; se déplace vers
     "Bob"
13  PreviousElement(personnes())
      ; se déplace vers "Harry"
14  DeleteElement(personnes())
        ; et le supprime.Il y a
     un élément avant Harry,
     alors il devient l'élément
     courant
15  MessageRequester("Information",
     "La personne actuellement
     pointée dans la liste est
     "+personnes(),
     #PB_MessageRequester_OK)
```

Voir aussi

AddElement() , InsertElement() , ClearList()

OS Supportés

Tous

111.9 FirstElement

Syntaxe

```
*Resultat =
    FirstElement(Liste())
```

Description

Le premier élément de la liste devient l'élément courant.

Arguments

Liste() La liste à utiliser.

Valeur de retour

Renvoie une valeur non nulle qui est l'adresse du premierr élément, zéro si le premier élément n'existe pas.
La valeur renvoyée est un pointeur sur le premier élément s'il existe.

Exemple

```
1  ; Un exemple d'utilisation
     simple
2  NewList nombres.w()
3
4  AddElement(nombres())
```

```
nombres() = 5
AddElement(nombres())
nombres() = 8

FirstElement(nombres())
MessageRequester("Information", "La valeur du premier élément est "+Str(nombres()), #PB_MessageRequester_OK)


; Un exemple qui utilise la valeur de retour
NewList nombres.w()

If FirstElement(nombres()) <> 0
  MessageRequester("Information", "La valeur du premier élément est "+Str(nombres()), #PB_MessageRequester_OK)
Else
  MessageRequester("Information", "La liste est vide", #PB_MessageRequester_OK)
EndIf

AddElement(nombres())
nombres() = 5
AddElement(nombres())
nombres() = 8

If FirstElement(nombres()) <> 0
  MessageRequester("Information", "La valeur du premier élément est "+Str(nombres()), #PB_MessageRequester_OK)
Else
  MessageRequester("Information", "La liste est vide", #PB_MessageRequester_OK)
EndIf


; Un exemple réservé aux programmeurs expérimentés
NewList nombres.w()

AddElement(nombres())
nombres() = 5
AddElement(nombres())
nombres() = 8

*Element.Word = FirstElement(nombres())
```

```
43  If *Element
44    MessageRequester("Information",
    "La valeur du premier
    élément est
    "+Str(*Element\w),
    #PB_MessageRequester_OK)
45  Else
46    MessageRequester("Information",
    "La liste est vide",
    #PB_MessageRequester_OK)
47  EndIf
```

Voir aussi

LastElement() , PreviousElement() , NextElement() , SelectElement() , ListIndex()

OS Supportés

Tous

111.10 InsertElement

Syntaxe

```
*Resultat =
    InsertElement(Liste())
```

Description

Insère un nouvel élément avant l'élément courant, ou au début de la liste si la liste est vide. Ce nouvel élément devient l'élément courant de la liste.

Arguments

Liste() La liste à utiliser.

Valeur de retour

Renvoie une valeur non nulle qui est l'adresse du nouvel élément si le nouvel élément a été créé, zéro sinon.
La valeur renvoyée par cette commande est un pointeur vers les données du nouvel élément.

Exemple

```
1  ; La manière la plus simple
    d'utiliser InsertElement
2  NewList simple.w()
3  InsertElement(simple())
    ; Crée le premier élément
    de la liste
```

```
4   simple() = 23
5
6   InsertElement(simple())
     ; La position courante est
     le premier élément, nous
     ajoutons cet élément au
     début de la liste
7   simple() = 45
     ; L'ancien premier élément
     est maintenant le second
     élément de la liste
8
9
10  ; Ceci montre comment
     utiliser la valeur de
     retour de InsertElement
11  NewList experimentes.l()
12  If
     InsertElement(experimentes())
     <> 0
13    experimentes() = 12345
14  Else
15    MessageRequester("Erreur
     !", "Impossible d'allouer
     de la mémoire pour le
     nouvel élément",
     #PB_MessageRequester_OK)
16  EndIf
17
18
19  ; Une petite structure pour
     démontrer la description
     "programmeurs
     expérimentés" (ci-dessus)
20  Structure Programmeur
21    Nom.s
22    Talent.b
23  EndStructure
24
25  NewList
     LesProgrammeurs.Programmeur()
      ; La liste pour stocker
     les éléments
26
27  *Element.Programmeur =
     InsertElement(LesProgrammeurs())
28  If *Element<>0
29    *Element\Nom = "Dave"
30    *Element\Talent = 3    ;
     Celui-là, c'est un féru de
     PureBasic ! ;)
31  Else
32    MessageRequester("Erreur
     !", "Impossible d'allouer
     de la mémoire pour le
     nouvel élément",
     #PB_MessageRequester_OK)
33  EndIf
```

Voir aussi

AddElement() , DeleteElement() , ClearList()

OS Supportés

Tous

111.11 LastElement

Syntaxe

```
*Resultat =
    LastElement(Liste())
```

Description

Le dernier élément de la liste devient l'élément courant.

Arguments

Liste() La liste à utiliser.

Valeur de retour

Renvoie l'adresse du dernier élément de la liste ou zéro s'il n'y a pas d'éléments.
La valeur renvoyée par cette commande est un pointeur vers les données du dernier élément s'il existe.

Exemple

```
1   ; Un exemple d'utilisation simple
2   NewList nombres.w()
3
4   AddElement(nombres())
5   nombres() = 5
6   AddElement(nombres())
7   nombres() = 8
8
9   LastElement(nombres())
10  MessageRequester("Information", "La valeur du dernier élément est "+Str(nombres()), #PB_MessageRequester_OK)
11
12
13  ; Un exemple qui utilise la valeur de retour
14  NewList nombres.w()
15
16  If LastElement(nombres()) <> 0
```

```
17      MessageRequester("Information",
        "La valeur du dernier
        élément est
        "+Str(nombres()),
        #PB_MessageRequester_OK)
18    Else
19      MessageRequester("Information",
        "La liste est vide",
        #PB_MessageRequester_OK)
20    EndIf
21
22    AddElement(nombres())
23    nombres() = 5
24    AddElement(nombres())
25    nombres() = 8
26
27    If LastElement(nombres())
        <> 0
28      MessageRequester("Information",
        "La valeur du dernier
        élément est
        "+Str(nombres()),
        #PB_MessageRequester_OK)
29    Else
30      MessageRequester("Information",
        "La liste est vide",
        #PB_MessageRequester_OK)
31    EndIf
32
33
34    ; Un exemple réservé aux
        programmeurs expérimentés
35    NewList nombres.w()
36
37    AddElement(nombres())
38    nombres() = 5
39    AddElement(nombres())
40    nombres() = 8
41
42    *Element.Word =
        LastElement(nombres())
43    If *Element
44      MessageRequester("Information",
        "La valeur du dernier
        élément est
        "+Str(*Element\w),
        #PB_MessageRequester_OK)
45    Else
46      MessageRequester("Information",
        "La liste est vide",
        #PB_MessageRequester_OK)
47    EndIf
```

Voir aussi

FirstElement() , PreviousElement() , NextElement() , SelectElement() , ListIndex()

OS Supportés

Tous

111.12 ListIndex

Syntaxe

```
Resultat = ListIndex(Liste())
```

Description

Renvoie la position de l'élément courant de la liste.

Arguments

Liste() La liste à utiliser.

Valeur de retour

La position de l'élément courant dans la liste.
Le premier élément a la position 0, le deuxième la position 1, etc..
Une valeur de -1 indique qu'il n'y a pas d'élément courant (soit la liste est vide, soit ResetList() a été utilisé).

Remarques

Cette commande est très rapide car elle utilise une valeur précalculée.

Exemple

```
1   NewList fruit.s()
2
3   AddElement(fruit()) :
     fruit() = "oranges"
4   AddElement(fruit()) :
     fruit() = "bananes"
5   AddElement(fruit()) :
     fruit() = "pommes"
6   AddElement(fruit()) :
     fruit() = "poires"
7
8   FirstElement(fruit())
9   MessageRequester("Fruit :
     "+fruit(), "Maintenant à
     la position
     "+Str(ListIndex(fruit())),
      #PB_MessageRequester_OK)
10
11  NextElement(fruit())
```

```
12  MessageRequester("Fruit :
     "+fruit(), "Maintenant à
     la position
     "+Str(ListIndex(fruit())),
      #PB_MessageRequester_OK)
13
14  NextElement(fruit())
15  MessageRequester("Fruit :
     "+fruit(), "Maintenant à
     la position
     "+Str(ListIndex(fruit())),
      #PB_MessageRequester_OK)
16
17  NextElement(fruit())
18  MessageRequester("Fruit :
     "+fruit(), "Maintenant à
     la position
     "+Str(ListIndex(fruit())),
      #PB_MessageRequester_OK)
```

Voir aussi

SelectElement() , ListSize()

OS Supportés

Tous

111.13 NextElement

Syntaxe

```
*Resultat =
    NextElement(Liste())
```

Description

L'élément suivant devient l'élément courant.

Arguments

Liste() La liste à utiliser.

Valeur de retour

Renvoie l'adresse de l'élément suivant en cas de succès ou zéro s'il n'y a pas d'élément suivant.
La valeur renvoyée par cette commande est un pointeur vers les données de l'élément suivant s'il existe.

Remarques

Passe au premier élément si ResetList() a été utilisé.

Exemple

```
1  NewList scores.w()
2
3  For i=1 To 10
4    AddElement(scores())
5    scores() = 100 - i
6  Next
7
8  ResetList(scores())
9  While NextElement(scores())
10   ; Ceci est correct car le
     premier appel à
     NextElement() va déplacer
     l'élément courant vers le
     premier élément de la liste
11    MessageRequester("Score",
     Str(scores()),
     #PB_MessageRequester_OK)
12 Wend
```

Voir aussi

ResetList() , PreviousElement() , FirstElement() , LastElement() , SelectElement() , ListIndex()

OS Supportés

Tous

111.14 PreviousElement

Syntaxe

```
*Resultat =
    PreviousElement(Liste())
```

Description

L'élément précédent devient l'élément courant.

Arguments

Liste() La liste à utiliser.

Valeur de retour

Renvoie l'adresse de l'élément précédent en cas de succès ou zéro s'il n'y a pas d'élément précédent.
La valeur renvoyée par cette commande est un pointeur vers les données de l'élément précédent s'il existe.

Exemple

```
1   NewList nombres.w()
2
3   For i=1 To 10
4     AddElement(nombres())
5     nombres() = i
6   Next
7
8   Repeat
9     MessageRequester("Nombre
      ", Str(nombres()),
      #PB_MessageRequester_OK)
10  Until
      PreviousElement(nombres())
      = 0
```

Voir aussi

NextElement() , FirstElement() , LastElement() , SelectElement() , ListIndex()

OS Supportés

Tous

111.15 ResetList

Syntaxe

```
ResetList(Liste())
```

Description

Change l'index de l'élément courant et le place avant le premier élément de la liste. Donc il n'y a plus d'éléments valides.

Arguments

Liste() La liste à utiliser.

Valeur de retour

Aucune.

Remarques

C'est particulièrement utile pour parcourir tous les éléments avec NextElement() .

Exemple

```
1   NewList Amis.s()
2
3   AddElement(Amis())
4   Amis() = "Arnaud"
5
6   AddElement(Amis())
7   Amis() = "Seb"
8
9   ResetList(Amis())
10  While NextElement(Amis())
11    Debug Amis() ; Affiche
      tous les éléments de la
      liste
12  Wend
```

Voir aussi

NextElement() , ListIndex()

OS Supportés

Tous

111.16 SelectElement

Syntaxe

```
*Resultat =
    SelectElement(Liste(),
    Position)
```

Description

Change l'élément courant par celui trouvé à la position spécifiée.

Arguments

Liste() La liste à utiliser.

Position La position du nouvel élément courant.
Le premier élément de la liste est à la position 0, le suivant à la position 1 et ainsi de suite. Assurez-vous de ne pas spécifier une position située en dehors de la liste (valeur négative ou supérieure à ListSize() -1) !

Valeur de retour

Renvoie un pointeur vers les données de l'élément sélectionné ou zéro si la position est hors limite.

Remarques

Ceci est très utile si vous souhaitez sauter à une position précise de la liste. Comme les listes n'utilisent pas d'index, un saut d'élément en élément est effectué jusqu'à la position cible recherchée. Si une commande plus rapide est nécessaire, utilisez la commande ChangeCurrentElement() .

Exemple

```
1  NewList MaListe.l()
2
3  AddElement(MaListe()) :
    MaListe() = 23
4  AddElement(MaListe()) :
    MaListe() = 56
5  AddElement(MaListe()) :
    MaListe() = 12
6  AddElement(MaListe()) :
    MaListe() = 73
7
8  SelectElement(MaListe(), 0)
9  MessageRequester("Position",
    "A la position 0, la
    valeur est
    "+Str(MaListe()),0)
10
11 SelectElement(MaListe(), 2)
12 MessageRequester("Position",
    "A la position 2, la
    valeur est
    "+Str(MaListe()),0)
13
14 SelectElement(MaListe(), 1)
15 MessageRequester("Position",
    "A la position 1, la
    valeur est
    "+Str(MaListe()),0)
16
17 SelectElement(MaListe(), 3)
18 MessageRequester("Position",
    "A la position 3, la
    valeur est
    "+Str(MaListe()),0)
```

Voir aussi

ChangeCurrentElement()

OS Supportés

Tous

111.17 SwapElements

Syntaxe

```
SwapElements(Liste(),
    *PremierElement,
    *DeuxiemeElement)
```

Description

Permute la place de 2 éléments de la liste.

Arguments

Liste() La liste à utiliser.

***PremierElement** Adresse du premier élément à échanger.
Vous pouvez récupérer cette adresse en utilisant l'opérateur @ sur le nom de la liste.

***DeuxiemeElement** Adresse du second élément à échanger.
Vous pouvez récupérer cette adresse en utilisant l'opérateur @ sur le nom de la liste.

Valeur de retour

Aucune.

Remarques

Cette fonction procure une façon rapide pour réorganiser ou trier une liste car les éléments ne sont pas déplacés.

Exemple

```
1   NewList Nombres()
2
3   For k=0 To 10
4     AddElement(Nombres())
5     Nombres() = k
6   Next
7
8   SelectElement(Nombres(), 3)
      ; Sélectionne le 4ème
      élément
9   *PremierElement = @Nombres()
10
11  SelectElement(Nombres(), 9)
      ; Sélectionne le 10ème
      élément
12  *DeuxiemeElement =
      @Nombres()
13
14  ; Echange le 4ème et le
      10ème
```

```
15    ;
16    SwapElements(Nombres(),
       *PremierElement,
       *DeuxiemeElement)
17
18    ; Prouvons-le
19    ;
20    ForEach Nombres()
21      Debug Nombres()
22    Next
```

Voir aussi

MoveElement()

OS Supportés

Tous

111.18 MoveElement

Syntaxe

```
MoveElement(Liste(), Location
    [, *ElementRelatif])
```

Description

Déplace l'élément courant vers une autre position dans la liste.

Arguments

Liste() La liste à utiliser.

Location L'emplacement de l'élément courant. Peut être une des valeurs suivantes :

```
#PB_List_First : Déplacer
 l'élément vers le début
 de la liste
#PB_List_Last  : Déplacer
 l'élément vers la fin de
 la liste
#PB_List_Before: Déplacer
 l'élément avant le
 *ElementRelatif
#PB_List_After : Déplacer
 l'élément après le
 *ElementRelatif
```

***ElementRelatif (optionnel)** Indique l'adresse d'un autre élément auprès duquel l'élément actuel doit être déplacé. Ce paramètre est requis lorsque le paramètre "Localisation" est #PB_List_Before ou #PB_List_After. Vous pouvez obtenir cette adresse en utilisant l'opérateur @ avec le nom de la liste.

Valeur de retour

Aucune.

Remarques

L'élément déplacé reste l'élément courant de la liste. Cette opération est rapide car la donnée elle-même n'est pas déplacée.

Exemple

```
1   NewList Nombres()
2
3   For k=0 To 10
4     AddElement(Nombres())
5     Nombres() = k
6   Next
7
8   SelectElement(Nombres(), 5)
9   *Relatif = @Nombres()
      ; l'adresse de l'élément 5
10
11  SelectElement(Nombres(), 0)
12  MoveElement(Nombres(),
     #PB_List_After, *Relatif)
     ; déplacement après
     l'élément 5
13
14  SelectElement(Nombres(), 10)
15  MoveElement(Nombres(),
     #PB_List_First)
     ; déplacement au début
16
17  ; Resultat
18  ;
19  ForEach Nombres()
20    Debug Nombres()
21  Next
```

Voir aussi

SwapElements()

OS Supportés

Tous

111.19 PushListPosition

Syntaxe

```
PushListPosition(Liste())
```

Description

Mémorise l'élément courant (s'il existe) afin qu'il puisse être restauré plus tard en utilisant PopListPosition() .

Arguments

Liste() La liste à utiliser.

Valeur de retour

Aucune.

Remarques

La position est mémorisée sur une structure de pile, donc plusieurs appels à cette fonction sont possibles.
Cette fonction peut être utilisée pour sauvegarder l'élément courant, donc une itération peut être faite sur la liste en utilisant NextElement() ou foreach et l'élément courant peut être restauré après l'itération en utilisant PopListPosition() .
Plusieurs appels à cette fonction peuvent être faits, aussi longtemps que chacun est équilibré avec un appel PopListPosition() correspondant.
Note : Il n'est pas permis de supprimer un élément qui a été mémorisé, en utilisant DeleteElement() ou ClearList() . Cela peut entraîner un plantage lors de l'appel de PopListPosition() parce que la mémoire n'est plus valide.

Exemple

```
1  NewList Nombres()
2  AddElement(Nombres()):
     Nombres() = 1
3  AddElement(Nombres()):
     Nombres() = 2
4  AddElement(Nombres()):
     Nombres() = 5
5  AddElement(Nombres()):
     Nombres() = 3
6  AddElement(Nombres()):
     Nombres() = 5
7  AddElement(Nombres()):
     Nombres() = 2
8
9  ; Une simple élimination de
     doublon
10 ;
11 ForEach Nombres()
12   Valeur = Nombres()
13   PushListPosition(Nombres())
```

```
14      While NextElement(Nombres())
15        If Nombres() = Valeur
16          DeleteElement(Nombres())
17        EndIf
18      Wend
19      PopListPosition(Nombres())
20    Next
21
22    ForEach Nombres()
23      Debug Nombres()
24    Next
```

Voir aussi

PopListPosition() , SelectElement() , ChangeCurrentElement() , NextElement() , PreviousElement() , ForEach

OS Supportés

Tous

111.20 PopListPosition

Syntaxe

```
PopListPosition(Liste())
```

Description

Restaure l'élément courant précédemment mise en mémoire avec PushListPosition() .

Arguments

Liste() La liste à utiliser.

Valeur de retour

Aucune.

Remarques

L'état de la liste sera le même que celui de l'appel correspondant à PushListPosition() . S'il n'y avait pas d'élément courant après un PushListPosition() alors il n'y a pas d'élément courant après cet appel aussi.

Exemple

```
1   NewList Nombres()
2   AddElement(Nombres()):
      Nombres() = 1
3   AddElement(Nombres()):
      Nombres() = 2
```

```
4   AddElement(Nombres()):
     Nombres() = 5
5   AddElement(Nombres()):
     Nombres() = 3
6   AddElement(Nombres()):
     Nombres() = 5
7   AddElement(Nombres()):
     Nombres() = 2
8
9   ; Une simple élimination de
     doublon
10  ;
11  ForEach Nombres()
12    Valeur = Nombres()
13    PushListPosition(Nombres())
14    While
     NextElement(Nombres())
15      If Nombres() = Valeur
16        DeleteElement(Nombres())
17      EndIf
18    Wend
19    PopListPosition(Nombres())
20  Next
21
22  ForEach Nombres()
23    Debug Nombres()
24  Next
```

Voir aussi

PushListPosition() , SelectElement() , ChangeCurrentElement() , NextElement() , PreviousElement() , ForEach

OS Supportés

Tous

111.21 MergeLists

Syntaxe

```
MergeLists(ListeSource(),
    ListeDestination() [,
    Location])
```

Description

Déplace tous les éléments de la liste ListeSource() vers la liste ListeDestination().

Arguments

ListeSource() La liste à partir de laquelle les éléments seront pris. Cette liste sera vide après l'opération.

ListeDestination() La liste vers laquelle les éléments seront déplacés. Cette liste contiendra les éléments des deux listes.

Location (optionnel) Indique l'emplacement où les éléments seront insérés dans la liste ListeDestination(). Peut être une des valeurs suivantes :

```
#PB_List_First : Insère
 les éléments au début de
 ListeDestination()
#PB_List_Last  : Ajoute
 des éléments à la fin de
 ListeDestination()
#PB_List_Before: Insère
 les éléments avant
 l'élément courant de
 ListeDestination()
#PB_List_After : Insère
 les éléments après
 l'élément courant de
 ListeDestination()
```

Valeur de retour

Aucune.

Remarques

Cette opération est rapide car les données elles-même ne sont pas déplacées.

Exemple

```
1   NewList A.s()
2   AddElement(A()): A() = "a0"
3   AddElement(A()): A() = "a1"
4   AddElement(A()): A() = "a2"
5   AddElement(A()): A() = "a3"
6
7   NewList B.s()
8   AddElement(B()): B() = "b0"
9   AddElement(B()): B() = "b1"
10  AddElement(B()): B() = "b2"
11  AddElement(B()): B() = "b3"
12
13  ; Insère les éléments de
     A() avant l'élément "b1"
     de B()
14  SelectElement(B(), 1)
15  MergeLists(A(), B(),
     #PB_List_Before)
16
17  ForEach B()
18    Debug B()
19  Next
```

Voir aussi

SplitList()

OS Supportés

Tous

111.22 SplitList

Syntaxe

```
SplitList(ListeSource(),
    ListeDestination() [,
    GarderElementCourant])
```

Description

Découpe une liste en deux.

Arguments

ListeSource() La liste à partir de laquelle les éléments seront déplacés. L'élément courant de la liste précise le point de division de la liste. S'il n'y a pas d'élément courant, alors tous les éléments restent dans ListeSource().

ListeDestination() La liste vers laquelle les éléments seront déplacés. Tous les éléments existants dans cette liste sont supprimés avant l'ajout des nouveaux éléments.

GarderElementCourant (optionnel) Indique si l'élément courant de ListeSource() reste dans ListeSource() ou s'il est déplacé vers ListeDestination().

```
#True  : L'élément courant
 reste dans ListeSource().
#False : L'élément courant
 est déplacé vers
 ListeDestination() (par
 défaut).
```

Valeur de retour

Aucune.

Remarques

Cette opération est rapide car les données elles-même ne sont pas déplacées.

Remarques

Si 'GarderElementCourant' est fixé à #True alors le nouvel élément courant dans ListeSource() sera l'élément précédent de la liste. S'il n'y a pas d'élément précédent alors la liste n'aura plus d'élément courant. La ListeDestination() n'aura aucun élément courant.

Exemple

```
1   NewList A()
2   NewList B()
3
4   For i = 0 To 10
5     AddElement(A())
6     A() = i
7   Next i
8
9   ; Coupe en deux la liste
      A() à l'élément 5 et
      déplace les éléments
      restants dans la liste B()
10  SelectElement(A(), 5)
11  SplitList(A(), B())
12
13
14  Debug " -- A() -- "
15  ForEach A()
16    Debug A()
17  Next
18
19  Debug " -- B() -- "
20  ForEach B()
21    Debug B()
22  Next
```

Voir aussi

MergeLists()

OS Supportés

Tous

Chapitre 112

Mail

Généralités

Les e-mails sont maintenant un moyen commun de communication entre personnes. Cette bibliothèque permet de créer des e-mails, avec ou sans pièces jointes, et de les envoyer à un ou plusieurs destinataires. Sous Linux, 'libcurl' doit être installé pour que certaines commandes MAIL fonctionnent (déjà installé dans la plupart des distributions Linux).

OS Supportés

Tous

112.1 AddMailAttachment

Syntaxe

```
Resultat =
    AddMailAttachment(#Courrier,
    Description$, Fichier$ [,
    TypeMime$])
```

Description

Ajoute un fichier en pièce jointe.

Arguments

#Courrier Le courrier à utiliser.

Description$ Le texte d'information qui apparait à côté du fichier.

Fichier$ Le fichier à joindre.
Si le nom du fichier ne contient pas de chemin complet, il est interprété par rapport au répertoire courant .
Une fois que la pièce jointe a été ajoutée, son contenu est entièrement copié dans le courrier et ne prend pas en compte les modifications ou même la suppression du fichier original.

TypeMime$ (optionnel) Le type du fichier joint.
Si ce paramètre est omis, l'extension de fichier sera utilisée pour déterminer le type MIME.
Ci-dessous, une liste des types MIME disponibles.
Si l'extension du fichier ne correspond à aucun des types MIME disponibles, alors le type "application/octet-stream" sera utilisé par défaut.

Valeur de retour

Renvoie une valeur non nulle si le fichier a été attaché avec succès, zéro sinon.

Remarques

Le nombre de pièces jointes qui peuvent être ajoutées n'est pas limité, mais la taille limite pour chaque pièce jointe est actuellement fixée à 100 Mo. La plupart des serveurs et des clients mails n'ont pas la capacité de traiter les pièces jointes de cette taille, il est donc conseillé que chaque pièce jointe aie une taille raisonnable.
Les types MIME disponibles sont :

```
application/acad          |
 AutoCAD    dwg
application/clariscad     |
 ClarisCAD  ccad
application/drafting      |
 MATRA Prelude drafting
 drw
application/dxf           |
 AutoCAD  dxf
application/i-deas        |
 SDRC I-deas  unv
application/iges          |
 Format d'échange CAO IGES
   igs,iges
application/oda           |
 ODA  oda
application/pdf           |
 Adobe Acrobat  pdf
application/postscript    |
 PostScript   ai,eps,ps
application/pro_eng       |
 ProEngineer  prt
application/rtf           |
 Rich text   rtf
application/set           |
 CAO SET  set
application/sla           |
 stéréolithographie   stl
application/solids        |
 MATRA Solids   dwg
```

```
application/step          |
 Données STEP  step
application/vda           |
 Surface   vda
application/x-mif         |
 Framemaker   mif
application/x-csh         |
 Script C-Shell (UNIX)   dwg
application/x-dvi         |
 texte dvi  dvi
application/hdf           |
 Données   hdf
application/x-latex       |
 LaTEX  latex
application/x-netcdf      |
 NetCDF   nc,cdf
application/x-sh          |
 Script Bourne Shell   dwg
application/x-tcl         |
 Script Tcl  tcl
application/x-tex         |
 Fichiers Tex  tex
application/x-texinfo     |
 eMacs  texinfo,texi
application/x-troff       |
 Troff  t,tr,troff
application/x-troff-man   |
 Troff/macro man  man
application/x-troff-me    |
 Troff/macro ME   me
application/x-troff-ms    |
 Troff/macro MS   ms
application/x-wais-source |
 Source Wais   src
application/x-bcpio       |
 CPIO binaire  bcpio
application/x-cpio        |
 CPIO Posix  cpio
application/x-gtar        |
 Tar GNU   gtar
application/x-shar        |
 Archives Shell  shar
application/x-sv4cpio     |
 CPIO SVR4n  sv4cpio
application/x-sv4crc      |
 CPIO SVR4 avec CRC  sc4crc
application/x-tar         |
 Archive tar   tar
application/x-ustar       |
 Archive tar Posix   man
application/zip           |
 Archive ZIP   man
audio/basic               |
 Audio   au,snd
audio/x-aiff              |
 Audio AIFF   aif,aiff,aifc
audio/x-wav               |
 Audio Wave   wav
image/gif                 |
```

```
 Images gif  man
image/ief                   |
 Images exchange format  ief
image/jpeg                  |
 Images Jpeg   jpg,jpeg,jpe
image/png                   |
 Images Png  png
image/tiff                  |
 Images Tiff   tiff,tif
image/x-cmu-raster          |
 Raster cmu  cmu
image/x-portable-anymap     |
 Anymap PBM   pnm
image/x-portable-bitmap     |
 Bitmap PBM   pbm
image/x-portable-graymap    |
 Graymap PBM  pgm
image/x-portable-pixmap     |
 Pixmap PBM   ppm
image/x-rgb                 |
 Image RGB   rgb
image/x-xbitmap             |
 Images Bitmap X   xbm
image/x-xpixmap             |
 Images Pixmap X   xpm
image/x-xwindowdump         |
 Images dump X Window  man
multipart/x-zip             |
 Archive zip  zip
multipart/x-gzip            |
 Archive GNU zip  gz,gzip
text/html                   |
 HTML   htm,html
text/plain                  |
 Texte sans mise en forme
 txt,g,h,c,cc,hh,m,f90
text/richtext               |
 Texte enrichi   rtx
text/tab-separated-value    |
 Texte avec séparation des
 valeurs  tsv
text/x-setext               |
 Texte Struct   etx
video/mpeg                  |
 Vidéos MPEG   mpeg,mpg,mpe
video/quicktime             |
 Vidéos QuickTime  qt,mov
video/msvideo               |
 Vidéos Microsoft Windows
 avi
video/x-sgi-movie           |
 Vidéos MoviePlayer  movie
```

Exemple

```
1  If CreateMail(0,
    "test@purebasic.com",
    "Salut")
```

```
2     If AddMailAttachment(0,
      "Attachment 1",
      OpenFileRequester("Choisissez
      un fichier", "", "", 0))
3        Debug "Pièce jointe
      correctement créée"
4     Else
5        Debug "Impossible de
      créer la pièce jointe "
6     EndIf
7   EndIf
```

Voir aussi

AddMailAttachmentData() , CreateMail()

OS Supportés

Tous

112.2 AddMailAttachmentData

Syntaxe

```
Resultat =
    AddMailAttachmentData(#Courrier,
    Description$, *Memoire,
    Taille [, TypeMime$])
```

Description

Ajoute le contenu d'une zone de mémoire comme pièce jointe.

Arguments

#Courrier Le courrier à utiliser.

Description$ Le texte d'information qui apparait sur la pièce jointe.

***Memoire** La zone de mémoire" qui contient la pièce jointe.
Une fois que la pièce jointe a été ajoutée, son contenu est entièrement copié dans le courrier et ne prend pas en compte les modifications ou même la suppression de la zone mémoire originale.

Taille La taille de la pièce jointe en octets.

TypeMime$ (optionnel) Le type du fichier joint.
Si ce paramètre est omis, l'extension de fichier sera utilisée pour déterminer le type MIME.
Ci-dessous, une liste des types MIME disponibles.
Si l'extension du fichier ne correspond à aucun des types MIME disponibles, alors le type "application/octet-stream" sera utilisé par défaut.

Valeur de retour

Renvoie une valeur non nulle si la pièce jointe a été créée avec succès, zéro sinon.

Remarques

Il est possible d'ajouter autant de pièces jointes que désiré, mais la taille d'une pièce jointe ne pourra pas dépasser 100 Mo. La plupart des serveurs et des clients mails ne pourront de toutes façons pas les gérer, donc il est conseillé de garder une taille raisonnable pour les pièces jointes.

Exemple

```
If CreateMail(0, "test@purebasic.com", "Salut")

  If AddMailAttachmentData(0, "Attachment 1", ?Salut, 5)
    Debug "Pièce jointe ajoutée"
  Else
    Debug "Impossible d'ajouter la pièce jointe"
  EndIf
EndIf

DataSection
  Salut:
  Data.b 'S', 'a', 'l', 'u', 't'
```

Voir aussi

AddMailAttachment() , CreateMail()

OS Supportés

Tous

112.3 AddMailRecipient

Syntaxe

```
AddMailRecipient(#Courrier, Adresse$, Options)
```

Description

Ajoute un destinataire.

Arguments

#**Courrier** Le courrier à utiliser.

Adresse$ L'adresse e-mail du destinataire. L'adresse doit avoir l'un des formats suivants :

```
"louis.dupond@domain.com"
"<louis.dupond@domain.com>"
"Louis Dupond
 <louis.dupond@domain.com>"
```

Options Le type de destinataire. Peut être une combinaison des valeurs suivantes :

```
#PB_Mail_To : Destinataire
 principal
#PB_Mail_Cc : Destinataire
 en copie conforme (Copie
 Carbone: Tous les
 destinataires sont
 visibles)
#PB_Mail_Bcc: Destinataire
 en copie invisible
 ('Blind carbon copy':
 Seul le destinataire est
 visible)
```

Valeur de retour

Aucune.

Exemple

```
1  If CreateMail(0,
    "test@purebasic.com",
    "Salut")
2    AddMailRecipient(0,
    "andre@purebasic.com",
    #PB_Mail_To) ; Andre est
    le destinataire principal
3    AddMailRecipient(0,
    "fred@purebasic.com",
    #PB_Mail_Cc)  ; Fred
    reçoit une copie du mail
4    AddMailRecipient(0,
    "timo@purebasic.com",
    #PB_Mail_Bcc) ; Timo
    reçoit une copie du mail
    aussi mais Andre et Fred
    ne le savent pas
5  EndIf
```

Voir aussi

RemoveMailRecipient() , CreateMail()

OS Supportés

Tous

112.4 CreateMail

Syntaxe

```
Resultat =
    CreateMail(#Courrier, De$,
    Objet$ [, Encodage])
```

Description

Crée un nouveau courrier.

Arguments

#Courrier Le numéro d'identification du nouveau courrier.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

De$ L'adresse de l'expéditeur.
L'adresse doit avoir l'un des formats suivants :

```
"louis.dupond@domain.com"
"<louis.dupond@domain.com>"
"Louis Dupond
 <louis.dupond@domain.com>"
```

Objet$ L'objet du courrier.

Encodage (optionnel) L'encodage du courrier. Peut prendre une des valeurs suivantes :

```
#PB_Ascii : Le contenu du
 courrier est encodé en
 ascii
#PB_UTF8  : Le contenu du
 courrier est encodé en
 UTF-8
(unicode)(valeur par défaut)
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé alors le numéro auto-généré est renvoyé en cas de succès.

Remarques

SetMailBody() , SetMailAttribute() , AddMailAttachment() et AddMailAttachmentData() permettent d'agir sur le contenu du courrier.
D'après le standard RFC 2822 les caractères #CRLF$ permettent de créer les sauts de ligne.
Sous Linux, 'libcurl' doit être installé pour que cette commande fonctionnent (déjà installé dans la plupart des distributions Linux).

Exemple

```
If CreateMail(0, "test@purebasic.com", "Salut")
  SetMailBody(0, "Ceci est un essai !" + #CRLF$ + "2ème ligne")
  Debug "Mail créé"
Else
  Debug "Impossible de créer le mail"
EndIf
```

Voir aussi

SetMailBody() , SetMailAttribute() , AddMailAttachment() , AddMailAttachmentData() , SendMail() , FreeMail()

OS Supportés

Tous

112.5 FreeMail

Syntaxe

```
FreeMail(#Courrier)
```

Description

Supprime un courrier et libère la mémoire associée.

Arguments

#Courrier Le courrier à utiliser.
Si #PB_All est spécifié, tous les mails restants seront libérés.

Valeur de retour

Aucune.

Remarques

Tous les courriers restants sont automatiquement supprimés quand le programme se termine.

Voir aussi

CreateMail()

OS Supportés

Tous

112.6 GetMailAttribute

Syntaxe

```
Resultat\$ =
    GetMailAttribute(#Courrier,
    Attribut)
```

Description

Renvoie l'attribut d'un courrier.

Arguments

#Courrier Le courrier à utiliser.

Attribut Peut être une des valeurs suivantes :

```
#PB_Mail_From    : Renvoie
 l'adresse de retour
 définie par CreateMail()
```

.

```
#PB_Mail_Subject: Renvoie
 l'objet défini par
 CreateMail()
```

.

```
#PB_Mail_XMailer: Renvoie
 le champ 'X-Mailer'
 (s'il existe)
#PB_Mail_Date    : Renvoie
 le champ 'Date' (s'il
 existe)
#PB_Mail_Custom : Renvoie
 les champs utilisateurs
 (s'ils existent)
```

Valeur de retour

Renvoie l'attribut dans une chaîne de caractères. Une chaîne vide est renvoyée si l'attribut n'existe pas.

Remarques

SetMailAttribute() peut être utilisé pour modifier les attributs.

Exemple

```
1  If CreateMail(0,
    "test@purebasic.com",
    "Salut")
```

```
2     Debug GetMailAttribute(0,
      #PB_Mail_From)     ;
      Affiche
      "test@purebasic.com"
3     Debug GetMailAttribute(0,
      #PB_Mail_Subject) ;
      Affiche "Salut"
4   EndIf
```

Voir aussi

SetMailAttribute() , CreateMail()

OS Supportés

Tous

112.7 GetMailBody

Syntaxe

```
Resultat\$ =
    GetMailBody(#Courrier)
```

Description

Renvoie le contenu d'un courrier.

Arguments

#**Courrier** Le courrier à utiliser.

Valeur de retour

Renvoie une chaîne de caractères qui est le contenu du courrier, préalablement défini par SetMailBody() .

Exemple

```
1   If CreateMail(0,
      "test@purebasic.com",
      "Salut")
2     SetMailBody(0, "C'est le
      contenu")
3     Debug GetMailBody(0) ;
      Affiche "C'est le contenu"
4   EndIf
```

Voir aussi

SetMailBody() , CreateMail()

OS Supportés

Tous

112.8 IsMail

Syntaxe

```
Resultat = IsMail(#Courrier)
```

Description

Teste si le courrier est correctement initialisé.

Arguments

#**Courrier** Le courrier à tester.

Valeur de retour

Renvoie une valeur non nulle si le courrier est valide.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'un courrier est prêt à être utilisé.

Voir aussi

CreateMail() , FreeMail()

OS Supportés

Tous

112.9 MailProgress

Syntaxe

```
Resultat =
    MailProgress(#Courrier)
```

Description

Renvoie la progression de l'envoi d'un courrier.

Arguments

#**Courrier** Le courrier à utiliser.

Valeur de retour

Renvoie le nombre d'octets déjà envoyés ou une des valeurs suivantes :

```
#PB_Mail_Connected: L'envoi
 du courrier est dans sa
 phase d'initialisation.
#PB_Mail_Finished : Le
 courrier a été
 correctement envoyé.
#PB_Mail_Error    : Le
 courrier n'a pas pu être
 envoyé, une erreur est
 survenue.
```

Voir aussi

SendMail()

OS Supportés

Tous

112.10 RemoveMailRecipient

Syntaxe

```
RemoveMailRecipient(#Courrier,
    [, Addresse$ [, Options])
```

Description

Supprimer un destinataire.

Arguments

#Courrier Le courrier à utiliser.

Addresse$ (optionnel) L'adresse du destinataire à supprimer.
L'adresse doit être dans un format correct ou le mail ne sera pas envoyé et doit correspondre à une adresse à partir d'un appel à AddMailRecipient() .
S'il n'est pas spécifié, tous les destinataires sont retirés de l'e-mail.

Options (optionnel) La ou les catégories du ou des destinataires. Peut être une combinaison des valeurs suivantes :

```
#PB_Mail_To :
 Destinataire(s)
 principal(aux).
#PB_Mail_Cc :
 Destinataire(s) en copie
 conforme (Copie Carbone:
 Tous les destinataires
 sont visibles).
```

```
#PB_Mail_Bcc:
 Destinataire(s) en copie
 invisible ('Blind carbon
 copy': Seul le
 destinataire est
 visible).
```

S'il n'est pas spécifié, toutes les catégories sont retirées pour l'adresse indiquée.

Valeur de retour

Aucune.

Exemple

```
If CreateMail(0,
 "test@votreadresse.com",
 "Salut")
  AddMailRecipient(0,
 "andre@purebasic.com",
 #PB_Mail_To) ; Andre est
 le destinataire principal
  AddMailRecipient(0,
 "fred@purebasic.com",
 #PB_Mail_Cc)  ; Fred
 reçoit une copie du mail
  AddMailRecipient(0,
 "timo@purebasic.com",
 #PB_Mail_Bcc) ; Timo
 reçoit une copie du mail
 aussi mais Andre et Fred
 ne le savent pas

  ; Assurez-vous que Fred
 soit retiré de toutes les
 destinations :-)
  RemoveMailRecipient(0,
 "fred@purebasic.com")
EndIf
```

Voir aussi

AddMailRecipient()

OS Supportés

Tous

112.11 SendMail

Syntaxe

```
Resultat =
    SendMail(#Courrier, Smtp$
    [, Port [, Options [,
    Identifiant$,
    MotdePasse$]]])
```

Description

Envoie un courrier.

Arguments

#Courrier Le courrier à envoyer.

Smtp$ L'adresse du serveur d'envoi de courrier.

Port (optionnel) Le port du serveur mail (par défaut 25).

Options (optionnel) Peut être l'une des valeurs suivantes :

```
#PB_Mail_Asynchronous:
 envoie le courrier en
 arrière-plan. Utiliser
 MailProgress()
pour suivre la progression.
#PB_Mail_UseSSL      :
 envoie le courrier avec
 TLS/SSL (le serveur doit
 supporter ce protocole).
```

Identifiant$, MotdePasse$ (optionnel) L'utilisateur et le mot de passe utilisé pour l'authentification SMTP, si le serveur les requiert.

Valeur de retour

Renvoie une valeur non nulle si le courrier a été envoyé avec succès, zéro sinon.

Exemple : Simple SMTP

```
1  ; Note: changez l'adresse
    de destination et le smtp
    pour avoir un exemple
    fonctionnel.
2  ;
3  If CreateMail(0,
    "test@votreadresse.com",
    "Salut")
4    AddMailRecipient(0,
    "votreadresse@domaine.com",
    #PB_Mail_To)
5
6    Debug SendMail(0,
    "smtp.votresmtpici.com")
7  EndIf
```

Exemple : Utiliser GMail (TLS)

```
1  ; Assurez-vous d'utiliser
    le bon login et le bon mot
    de passe
```

```
2   ;
3   Identifiant$ = "votrelogin"
4   If CreateMail(0,
     Identifiant$ +
     "@gmail.com", "Hello")
5     AddMailRecipient(0,
     "votreadresse@gmail.com",
     #PB_Mail_To)
6
7     Debug SendMail(0,
     "smtp.gmail.com", 465,
     #PB_Mail_UseSSL,
     Identifiant$, "motdepasse")
8   EndIf
```

Voir aussi

CreateMail() , MailProgress()

OS Supportés

Tous

112.12 SetMailAttribute

Syntaxe

```
SetMailAttribute(#Courrier,
    Attribut, Valeur$)
```

Description

Modifie l'attribut d'un courrier.

Arguments

#**Courrier** Le courrier à utiliser.

Attribut Peut être une des valeurs suivantes :

- #PB_Mail_From : Change l'adresse de retour définie par CreateMail()
- #PB_Mail_Subject: Change l'objet défini par CreateMail()
- #PB_Mail_XMailer: Change le champ 'X-Mailer' (par défaut: aucun)
- #PB_Mail_Date : Change le champ 'Date' (par défaut la date de l'ordinateur)
- #PB_Mail_Custom : Change les champs utilisateurs (peuvent être sur plusieurs lignes)

Valeur$ Le nouvel attribut.

Valeur de retour

Aucune.

Exemple

```
1  If CreateMail(0,
     "test@purebasic.com",
     "Salut")
2     SetMailAttribute(0,
     #PB_Mail_XMailer,
     "PureMailer")
3     Debug GetMailAttribute(0,
     #PB_Mail_XMailer) ;
     Affichera "PureMailer"
4  EndIf
```

Voir aussi

GetMailAttribute() , CreateMail()

OS Supportés

Tous

112.13 SetMailBody

Syntaxe

```
SetMailBody(#Courrier,
    CorpsMessage$)
```

Description

Modifie le contenu (corps) d'un courrier.

Arguments

#Courrier Le courrier à utiliser.

CorpsMessage$ Le nouveau texte du corps de message.

Valeur de retour

Aucune.

Remarques

Selon le standard RFC 2822 un saut de ligne dans un e-mail utilise #CRLF$. GetMailBody() est disponible pour récupérer le contenu (corps) d'un courrier.

Exemple

```
1  If CreateMail(0,
     "test@purebasic.com",
     "Salut")
2    SetMailBody(0, "C'est le
     contenu")
3    Debug GetMailBody(0) ;
     Affiche "C'est le contenu"
4  EndIf
```

Voir aussi

GetMailAttribute() , CreateMail()

OS Supportés

Tous

Chapitre 113

Map

Généralités

Les maps (aussi connues sous la dénomination 'table de hachage' ou 'dictionnaire') sont des structures utilisées pour stocker des données qui sont allouées dynamiquement.
C'est une collection d'éléments qui sont complètement indépendants les uns des autres.
Il est possible d'ajouter autant d'éléments que désiré (limité uniquement par la quantité de mémoire disponible) et on consulte les éléments à l'aide d'une clé.
Ce type de gestion de données est très utile quand un accès rapide à un élément quelconque est requis. L'ordre d'insertion des éléments n'est pas conservé (contrairement à une liste), une map ne peut donc pas être triée.
Avant de travailler avec les maps, il faut préalablement les déclarer. Cela se fait avec le mot-clef NewMap . Les structures sont également fréquemment utilisées dans les maps.
Pour analyser le contenu d'une map, il est possible d'utiliser les boucles suivantes : For : Next , ForEach : Next , Repeat : Until ou While : Wend .
Les autres possibilités pour stocker des données sont les Tableaux et les Listes .

OS Supportés

Tous

113.1 AddMapElement

Syntaxe

```
Resultat =
    AddMapElement(Map(), Cle$
    [, Options])
```

Description

Ajoute un nouvel élément vide.

Arguments

Map() La map à utiliser.

Cle$ La clé du nouvel élément.

Options (optionnel) Peut être une des valeurs suivantes :

```
#PB_Map_ElementCheck  :
 Vérifie si un élément
 existe déjà avec la même
 clé et le remplace
 (défaut).
#PB_Map_NoElementCheck:
 Pas de vérification,
 donc si un élément
 existe déjà avec la même
 clé, il sera perdu et
 inaccessible.
                        Sa
 mémoire ne sera libérée
 que si ClearMap()
est utilisé. Ce mode est
 plus rapide mais plus
 dangereux.
                       A
 utiliser avec précaution.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
La valeur renvoyée est un pointeur vers le nouvel élément.

Remarques

Ce nouvel élément devient l'élément courant.
Cette fonction n'est pas obligatoire pour gérer les maps, car les éléments sont ajoutés automatiquement quand une valeur est affectée à une nouvelle clé.

Exemple

```
1  NewMap Pays.s()
2
3  ; Manière normale d'ajouter
    un élément
4  Pays("US") = "United State"
5
6  ; Et maintenant en
    utilisant 'AddMapElement()'
7  AddMapElement(Pays(), "FR")
```

```
  Pays() = "France"

  ForEach Pays()
    Debug Pays()
  Next
```

Voir aussi

DeleteMapElement() , ClearMap() , MapSize()

OS Supportés

Tous

113.2 ClearMap

Syntaxe

```
ClearMap(Map())
```

Description

Efface tous les éléments et libère la mémoire associée.

Arguments

Map() La map à utiliser.

Valeur de retour

Aucune.

Remarques

Après cette opération, la map est toujours utilisable, mais elle ne contient plus d'éléments.
PureBasic libèrera seulement la mémoire occupée par les éléments. Si la map a été utilisée pour stocker des objets dynamiques, il n'est pas possible de le détecter (en PureBasic ou dans un autre langage). Dans ce cas, il convient de libérer tous ces objets avant la destruction de la liste.

Exemple

```
  NewMap Pays.s()

  Pays("FR") = "France"
  Pays("US") = "United States"

  ; La preuve que des
    éléments ont été ajoutés à
    la map
```

```
7   MessageRequester("Information",
     "Il y a
     "+Str(MapSize(Pays()))+"
     éléments dans la map")
8
9   ; Efface les éléments de la
     map
10  ClearMap(Pays())
11  MessageRequester("Information",
     "Il y a
     "+Str(MapSize(Pays()))+"
     éléments dans la map")
```

Voir aussi

AddMapElement() , DeleteMapElement()

OS Supportés

Tous

113.3 CopyMap

Syntaxe

```
Resultat =
    CopyMap(MapSource(),
    MapDestination())
```

Description

Copie une Map.

Arguments

MapSource() La map à copier.

MapDestination() La map copiée.
Tous les éléments présents dans la 'MapDestination()' seront effacés. Après une copie réussie, les deux maps seront identiques.

Valeur de retour

Renvoie une valeur non nulle en cas de succès ou zéro sinon. Si les deux maps ne sont pas du même type (natif ou structuré) la copie ne se fera pas.

Exemple

```
1   NewMap Age()
2   NewMap AgeCopie()
3
4   Age("Jean")  = 15
5   Age("Elise") = 30
```

```
6
7   CopyMap(Age(), AgeCopie())
8
9   Debug AgeCopy("Jean")
10  Debug AgeCopy("Elise")
```

Voir aussi

CopyArray() , CopyList()

OS Supportés

Tous

113.4 FreeMap

Syntaxe

```
FreeMap(Map())
```

Description

Détruit une Map et libère toutes les ressources associées.

Arguments

Map() La map à utiliser.

Valeur de retour

Aucune.

Remarques

Pour accéder à cette map à nouveau, NewMap doit être appelé.

Voir aussi

ClearMap()

OS Supportés

Tous

113.5 MapSize

Syntaxe

```
Resultat = MapSize(Map())
```

Description

Renvoie le nombre d'éléments d'une Map.

Arguments

Map() La map à utiliser.

Valeur de retour

Renvoie le nombre d'éléments contenus dans la Map() spécifiée.

Remarques

Cette fonction ne modifie pas l'élément courant. Elle est très rapide (elle utilise une valeur précalculée) et peut être utilisée sereinement pour savoir si une map est vide ou non.

Exemple

```
1  NewMap Pays.s()
2
3  Pays("FR") = "France"
4  Pays("US") = "United States"
5
6  ; Affichera '2'
7  Debug "Taille de la map: "
     + Str(MapSize(Pays()))
```

OS Supportés

Tous

113.6 DeleteMapElement

Syntaxe

```
Resultat =
    DeleteMapElement(Map() [,
    Cle$])
```

Description

Efface un élément.

Arguments

Map() La map à utiliser.

Cle$ La clé de l'élément.
Si ce paramètre n'est pas spécifié alors c'est l'élément courant qui sera supprimé.

Valeur de retour

La valeur renvoyée est un pointeur vers le nouvel élément.
Si la liste ne comporte aucun élément courant après la suppression, le résultat est zéro.

Remarques

Après cet appel, le nouvel élément courant est l'élément précédent (celui qui précéde l'élément supprimé), qui est un élément arbitraire, car une map n'est pas triée. Si cet élément n'existe pas (en d'autres termes, vous avez supprimé le premier élément de la map) alors il n'y a plus d'élément en cours, comme après un ResetMap() . S'il n'y avait qu'un seul élément dans la carte lorsque vous l'avez supprimé, il ne vous reste plus aucun élément en cours !
Si le paramètre optionnel "Cle$" est spécifié, il n'y aura pas d'élément courant après cet appel. N'utilisez donc pas ce paramètre si la commande est utilisée dans une boucle ForEach : Next !

Exemple

```
1   NewMap Pays.s()
2
3   Pays("US") = "United States"
4   Pays("FR") = "France"
5   Pays("DE") = "Allemagne"
6
7   ; Supprime un pays
8   DeleteMapElement(Pays(),
      "FR")
9
10  ForEach Pays()
11    Debug Pays()
12  Next
```

Voir aussi

AddMapElement() , ClearMap() , MapSize()

OS Supportés

Tous

113.7 FindMapElement

Syntaxe

```
Resultat =
    FindMapElement(Map(), Cle$)
```

Description

Change l'élément courant.

Arguments

Map() La map à utiliser.
Cle$ La clé de l'élément.

Valeur de retour

Renvoie une valeur non nulle en cas de succès ou zéro sinon.
La valeur renvoyée est un pointeur vers l'élément courant.

Exemple

```
1   NewMap Pays.s()
2
3   Pays("US") = "United States"
4   Pays("FR") = "France"
5   Pays("DE") = "Allemagne"
6
7   If FindMapElement(Pays(),
     "US")
8     Debug "'US' est dans la
     liste des Pays."
9   Else
10    Debug "'US' n'est PAS
     dans la liste des Pays !"
11  EndIf
12
13  If FindMapElement(Pays(),
     "UK")
14    Debug "'UK' est dans la
     liste des Pays."
15  Else
16    Debug "'UK' n'est PAS
     dans la liste des Pays !"
17  EndIf
```

Voir aussi

AddMapElement() , DeleteMapElement() , MapKey()

OS Supportés

Tous

113.8 MapKey

Syntaxe

```
Resultat\$ = MapKey(Map())
```

Description

Renvoie la clé de l'élément courant.

Arguments

Map() La map à utiliser.

Valeur de retour

Renvoie la clé de l'élément courant.
S'il n'y a pas d'élément courant, une chaîne de caractères vide est renvoyée.

Exemple

```
1  NewMap Pays.s()
2
3  Pays("US") = "United States"
4  Pays("FR") = "France"
5  Pays("DE") = "Allemagne"
6
7  ForEach Pays()
8    Debug MapKey(Pays())
9  Next
```

Voir aussi

ResetMap() , NextMapElement()

OS Supportés

Tous

113.9 NextMapElement

Syntaxe

```
Resultat =
    NextMapElement(Map())
```

Description

Passe de l'élément courant à l'élément suivant.

Arguments

Map() La map à utiliser.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
La valeur renvoyée est un pointeur vers le nouvel élément.

Remarques

Si vous avez précédemment fait appel à ResetMap() alors l'élément courant est le premier élément.

Exemple

```
NewMap Pays.s()

Pays("US") = "United States"
Pays("FR") = "France"
Pays("DE") = "Allemagne"

ResetMap(Pays())
While NextMapElement(Pays())
  Debug Pays()
Wend
```

Voir aussi

ResetMap() , MapKey()

OS Supportés

Tous

113.10 ResetMap

Syntaxe

```
ResetMap(Map())
```

Description

Change l'élément courant et le place avant le premier élément de la map. Donc il n'y a plus d'élément courant valide.

Arguments

Map() La map à utiliser.

Valeur de retour

Aucune.

Remarques

C'est particulièrement utile pour parcourir tous les éléments avec NextMapElement() .

Exemple

Exemple :

```
NewMap Pays.s()

Pays("US") = "United States"
Pays("FR") = "France"
Pays("DE") = "Allemagne"

ResetMap(Pays())
```

```
8   While NextMapElement(Pays())
9     Debug Pays()
10  Wend
```

Voir aussi

NextMapElement()

OS Supportés

Tous

113.11 PushMapPosition

Syntaxe

```
PushMapPosition(Map())
```

Description

Mémorise l'élément courant (s'il existe).
Il pourra être restauré à l'aide de PopMapPosition() .

Arguments

Map() La map à utiliser.

Valeur de retour

Aucune.

Remarques

La position est sauvegardée dans une structure de pile, donc plusieurs appels à cette fonction sont possibles.
Cette fonction peut être utilisée pour sauvegarder l'élément courant, donc une itération peut être utilisée sur la map en utilisant NextMapElement() ou foreach et l'élément courant peut être restauré après l'itération en utilisant PopMapPosition() .
Plusieurs appels peuvent être faits à cette fonction, aussi longtemps que chacun est équilibré avec un appel PopMapPosition() correspondant.
Note : Il n'est pas permis de supprimer un élément qui a été mémorisé, en utilisant DeleteMapElement() ou ClearMap() . Cela peut entraîner un plantage lors de l'appel de PopMapPosition() parce que la mémoire n'est plus valide.

Exemple

```
1   NewMap Nombres()
2   Nombres("A") = 1
3   Nombres("B") = 2
4   Nombres("C") = 5
5   Nombres("D") = 3
6   Nombres("E") = 2
7   Nombres("F") = 5
8
9   ; Une simple élimination de
     doublon
10  ;
11  ForEach Nombres()
12    Value = Nombres()
13    PushMapPosition(Nombres())
14    While
     NextMapElement(Nombres())
15      If Nombres() = Value
16        DeleteMapElement(Nombres())
17      EndIf
18    Wend
19    PopMapPosition(Nombres())
20  Next
21
22  ForEach Nombres()
23    Debug Nombres()
24  Next
```

Voir aussi

PopMapPosition() , FindMapElement() , NextMapElement() , ResetMap() , ForEach

OS Supportés

Tous

113.12 PopMapPosition

Syntaxe

```
PopMapPosition(Map())
```

Description

Restaure l'élément courant précédemment mis en mémoire avec PushMapPosition() .

Arguments

Map() La map à utiliser.

Valeur de retour

Aucune.

Remarques

L'état de la map sera le même que celui de l'appel correspondant à PushMapPosition() . S'il n'y avait pas d'élément courant après un PushMapPosition() alors il n'y a pas d'élément courant après cet appel aussi.

Exemple

```
1   NewMap Nombres()
2   Nombres("A") = 1
3   Nombres("B") = 2
4   Nombres("C") = 5
5   Nombres("D") = 3
6   Nombres("E") = 2
7   Nombres("F") = 5
8
9   ; Une simple élimination de
     doublon
10  ;
11  ForEach Nombres()
12    Value = Nombres()
13    PushMapPosition(Nombres())
14    While
     NextMapElement(Nombres())
15      If Nombres() = Value
16        DeleteMapElement(Nombres())
17      EndIf
18    Wend
19    PopMapPosition(Nombres())
20  Next
21
22  ForEach Nombres()
23    Debug Nombres()
24  Next
```

Voir aussi

PushMapPosition() , FindMapElement() , NextMapElement() , ResetMap() , ForEach

OS Supportés

Tous

Chapitre 114

Material

Généralités

Les matières ou matériaux sont composés d'une ou plusieurs textures et parfois de couleurs.
Elles sont utilisées par les objets 3D tels que les 'Entités', les 'Billboards' et les 'Particules'.
Chaque matière regroupe un grand nombre de propriétés tel que l'éclairage, la couleur spéculaire et de réfraction, etc, ce qui permet à une matière de ressembler à un revêtement complexe comme le verre, l'eau, le bois...
InitEngine3D() doit être correctement initialisé avant de pouvoir utiliser ces commandes.
Les mots 'matière' et 'matériau' sont considérés ici comme synonymes afin d'alléger la lecture.

OS Supportés

Tous

114.1 AddMaterialLayer

Syntaxe

```
AddMaterialLayer(#Matiere,
    TextureID [, Mode [,
    IndexCoordonneeTexture]])
```

Description

Ajoute une nouvelle couche à une matière et lui affecte la texture spécifiée.

Arguments

#Matiere La matière à utiliser.

TextureID La texture à utiliser.

Un numéro 'TextureID' valide est obtenu facilement à l'aide de la commande TextureID() .

Mode (optionnel)

```
    #PB_Material_Add
                : Combine
  la nouvelle couche en
  ajoutant (Add) la valeur
  des pixels (donc le noir
  est complètement
  transparent)
#PB_Material_AddSigned
          : Effectue une
  opération 'Add' signée
  sur les pixels du calque
  précédent (la couleur
  noire est comme
  transparente)
#PB_Material_Substract
          : Effectue une
  opération 'Substract'
  signée sur les pixels du
  calque précédent (la
  couleur noire est comme
  transparente)
#PB_Material_Replace
           : Combine la
  nouvelle couche en
  remplaçant (Replace) les
  pixels
#PB_Material_AlphaBlend
         : Utilise le
  canal alpha de la
  texture (la texture doit
  être en TGA ou PNG) et
  la combine avec la
  couche précédente
#PB_Material_Modulate
           : Combine la
  nouvelle couche sur la
  précédente en modulant
  la valeur des pixels
  (Modulate)
#PB_Material_ModulateX2
         : Effectue une
  opération "Multiplier"
  sur les pixels du calque
  précédent et éclaircit
  ensuite (x2)
#PB_Material_ModulateX4
         : Effectue une
  opération "Multiplier"
  sur les pixels du calque
  précédent et éclaircit
  ensuite (x4)
#PB_Material_BlendDiffuseAlpha
  : Utilise le calque du
  canal Alpha de la
  texture et le mélange
```

```
    avec le calque précédent
    avec diffusion
  #PB_Material_BlendCurrentAlpha
    : Utilise le calque du
    canal Alpha de l'étape
    en cours et le mélange
    avec le calque précédent
```

IndexCoordonneeTexture (optionnel)
Index à utiliser pour les coordonnées de la texture (par défaut l'index = 0).

Valeur de retour

Aucune.

Voir aussi

CountMaterialLayers() , RemoveMaterialLayer()

OS Supportés

Tous

114.2 CopyMaterial

Syntaxe

```
Resultat =
    CopyMaterial(#Matiere,
    #NouvelleMatiere)
```

Description

Crée une nouvelle matière qui est la copie exacte d'une matière spécifiée.

Arguments

#Matiere La matière à copier.

#NouvelleMatiere La copie, c'est une nouvelle matière.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si la #NouvelleMatière était déjà créée, elle est automatiquement libérée et remplacée par la nouvelle.

Voir aussi

CreateMaterial() , FreeMaterial()

OS Supportés

Tous

114.3 CountMaterialLayers

Syntaxe

```
Resultat =
    CountMaterialLayers(#Matiere)
```

Description

Renvoie le nombre de couches d'une matière.

Arguments

#Matiere La matière à utiliser.

Valeur de retour

Renvoie le nombre de couches que contient la matière.

Voir aussi

AddMaterialLayer() , RemoveMaterialLayer()

OS Supportés

Tous

114.4 CreateMaterial

Syntaxe

```
Resultat =
    CreateMaterial(#Matiere,
    TextureID [, Couleur])
```

Description

Crée une nouvelle matière basée sur la texture spécifiée.

Arguments

#Matiere La nouvelle matière.

TextureID La texture à utiliser.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.
Un 'TextureID' valide ou #Null si la texture n'est pas nécessaire, est obtenu facilement à l'aide de la commande TextureID() .

Couleur (optionnel) La couleur à utiliser pour la couleur d'ambiance et diffuse. Une couleur RVB valide peut être créée avec RGB() .

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si la nouvelle matière était déjà créée, elle est automatiquement libérée et remplacée par la nouvelle.

Voir aussi

FreeMaterial() , IsMaterial() , MaterialID()

OS Supportés

Tous

114.5 CreateAnimatedMaterial

Syntaxe

```
Resultat =
    CreateAnimatedMaterial(#Matiere,
    Texture(), Duree.f)
```

Description

Crée une nouvelle matière animée à l'aide des textures spécifiées.

Arguments

#Matiere La nouvelle matière. #PB_Any peut être utilisé pour générer automatiquement ce numéro.

Texture() Un tableau de TextureID contenant les textures à utiliser pour l'animation. TextureID() peut être utilisé pour obtenir un identifiant de texture valide.

Duree La durée de l'animation en secondes. Il peut s'agir d'un nombre fractionnaire, par exemple : 1,5 soit une seconde et demie. Une fois l'animation terminée, elle recommencera automatiquement.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Matiere, le nombre généré est renvoyé en cas de succès.

Exemple

```
1  CreateTexture(0, 256, 256)
2  CreateTexture(1, 256, 256)
3  CreateTexture(2, 256, 256)
4
5  Dim Textures(2)
6  Textures(0) = TextureID(0)
7  Textures(1) = TextureID(1)
8  Textures(2) = TextureID(2)
9
10 ; Crée une matière animée
     avec 3 textures, qui
     jouera pendant 500 ms
11 ;
12 CreateAnimatedMaterial(0,
     Textures(), 0.5)
```

OS Supportés

Tous

114.6 CreateShader

Syntaxe

```
Resultat =
    CreateShader(ShaderID,
    ProgrammeVertex$,
    ProgrammeFragment$)
```

Description

Crée un nouveau shader en utilisant les programmes de vertex et de fragments spécifiés.

Arguments

ShaderID Un nombre entre 0 et 65536 identifie le nouveau shader.
Pour sélectionner l'un des shaders prédéfinis, utilisez l'une des valeurs suivantes comme 'ShaderID' (les valeurs ProgrammeVertex$ et ProgrammeFragment$ seront ignorées) :

```
#PB_Material_ColorShader
#PB_Material_PerpixelShader
#PB_Material_BumpShader
#PB_Material_SkyShader
#PB_Material_WaterShader
#PB_Material_WaterShaderRTT
#PB_Material_OceanShader
#PB_Material_PointSpriteSphereShader
```

ProgrammeVertex$ Le programme GLSL à utiliser pour les sommets (vertices).

ProgrammeFragment$ Le programme GLSL à utiliser pour les fragments.

Valeur de retour

Renvoie une valeur non nulle si le shader a été créé avec succès, zéro sinon.

OS Supportés

Tous

114.7 CreateShaderMaterial

Syntaxe

```
Resultat =
    CreateShaderMaterial(#Matiere,
    ShaderID)
```

Description

Crée une nouvelle matière basée sur le shader.

Arguments

#Matiere La nouvelle matière. #PB_Any peut être utilisé pour générer automatiquement ce numéro.

ShaderID L'identifiant du shader. CreateShader() peut être utilisée pour créer un nouveau shader.

Valeur de retour

Renvoie une valeur non nulle si la matière basée sur le shader a été créé avec succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Matiere, le nombre généré est renvoyé en cas de succès.

OS Supportés

Tous

114.8 MaterialShaderAutoParameter

Syntaxe

```
MaterialShaderAutoParameter(#Matiere,
    TypeProgramme,
    NomParametre$,
    TypeParametre, Extra.f)
```

Description

Défini une valeur de paramètre pour la matière basée sur le shader créé précédemment avec CreateShaderMaterial() .

Arguments

#**Matiere** La matière à utiliser.

TypeProgramme Type de programme. Il peut s'agir de l'une des valeurs suivantes :

```
#PB_Shader_Vertex   :
 Utiliser le programme
 vertex.
#PB_Shader_Fragment:
 Utiliser le programme
 vertex de fragment.
```

NomParametre$ Nom du paramètre à utiliser avec le programme GLSL.

TypeParametre Type de paramètre. Il peut s'agir de l'une des valeurs suivantes :

```
#PB_Shader_WorldMatrix
#PB_Shader_InverseWorldMatrix
#PB_Shader_TransposeWorldMatrix
#PB_Shader_InverseTransposeWorldMatrix
#PB_Shader_WorldMatrixArray3x4
#PB_Shader_WorldMatrixArray
#PB_Shader_WorldDualquaternionArray2x4
#PB_Shader_WorldScaleShearMatrixArray3x4
#PB_Shader_ViewMatrix
#PB_Shader_InverseViewMatrix
#PB_Shader_TransposeViewMatrix
#PB_Shader_InverseTransposeViewMatrix
#PB_Shader_ProjectionMatrix
#PB_Shader_InverseProjectionMatrix
#PB_Shader_TransposeProjectionMatrix
#PB_Shader_InverseTransposeProjectionMatrix
#PB_Shader_ViewprojMatrix
#PB_Shader_InverseViewprojMatrix
#PB_Shader_TransposeViewprojMatrix
#PB_Shader_InverseTransposeViewprojMatrix
#PB_Shader_WorldviewMatrix
#PB_Shader_InverseWorldviewMatrix
#PB_Shader_TransposeWorldviewMatrix
#PB_Shader_InverseTransposeWorldviewMatrix
#PB_Shader_WorldviewprojMatrix
#PB_Shader_InverseWorldviewprojMatrix
#PB_Shader_TransposeWorldviewprojMatrix
#PB_Shader_InverseTransposeWorldviewprojMatrix
#PB_Shader_RenderTargetFlipping
#PB_Shader_VertexWinding
#PB_Shader_FogColour
#PB_Shader_FogParams
#PB_Shader_SurfaceAmbientColour
```

```
#PB_Shader_SurfaceDiffuseColour
#PB_Shader_SurfaceSpecularColour
#PB_Shader_SurfaceEmissiveColour
#PB_Shader_SurfaceShininess
#PB_Shader_LightCount
#PB_Shader_AmbientLightColour
#PB_Shader_LightDiffuseColour
#PB_Shader_LightSpecularColour
#PB_Shader_LightAttenuation
#PB_Shader_SpotlightParams
#PB_Shader_LightPosition
#PB_Shader_LightPositionObjectSpace
#PB_Shader_LightPositionViewSpace
#PB_Shader_LightDirection
#PB_Shader_LightDirectionObjectSpace
#PB_Shader_LightDirectionViewSpace
#PB_Shader_LightDistanceObjectSpace
#PB_Shader_LightPowerScale
#PB_Shader_LightDiffuseColourPowerScaled
#PB_Shader_LightSpecularColourPowerScaled
#PB_Shader_LightDiffuseColourArray
#PB_Shader_LightSpecularColourArray
#PB_Shader_LightDiffuseColourPowerScaledArray
#PB_Shader_LightSpecularColourPowerScaledArray
#PB_Shader_LightAttenuationArray
#PB_Shader_LightPositionArray
#PB_Shader_LightPositionObjectSpaceArray
#PB_Shader_LightPositionViewSpaceArray
#PB_Shader_LightDirectionArray
#PB_Shader_LightDirectionObjectSpaceArray
#PB_Shader_LightDirectionViewSpaceArray
#PB_Shader_LightDistanceObjectSpaceArray
#PB_Shader_LightPowerScaleArray
#PB_Shader_SpotlightParamsArray
#PB_Shader_DerivedAmbientLightColour
#PB_Shader_DerivedSceneColour
#PB_Shader_DerivedLightDiffuseColour
#PB_Shader_DerivedLightSpecularColour
#PB_Shader_DerivedLightDiffuseColourArray
#PB_Shader_DerivedLightSpecularColourArray
#PB_Shader_LightNumber
#PB_Shader_LightCastsShadows
#PB_Shader_ShadowExtrusionDistance
#PB_Shader_CameraPosition
#PB_Shader_CameraPositionObjectSpace
#PB_Shader_TextureViewprojMatrix
#PB_Shader_TextureViewprojMatrixArray
#PB_Shader_TextureWorldviewprojMatrix
#PB_Shader_TextureWorldviewprojMatrixArray
#PB_Shader_SpotlightViewprojMatrix
#PB_Shader_SpotlightViewprojMatrixArray
#PB_Shader_SpotlightWorldviewprojMatrix
#PB_Shader_Custom
#PB_Shader_Time
#PB_Shader_Time0X
#PB_Shader_Costime0X
#PB_Shader_Sintime0X
#PB_Shader_Tantime0X
#PB_Shader_Time0XPacked
```

```
#PB_Shader_Time01
#PB_Shader_Costime01
#PB_Shader_Sintime01
#PB_Shader_Tantime01
#PB_Shader_Time01Packed
#PB_Shader_Time02pi
#PB_Shader_Costime02pi
#PB_Shader_Sintime02pi
#PB_Shader_Tantime02pi
#PB_Shader_Time02piPacked
#PB_Shader_FrameTime
#PB_Shader_Fps
#PB_Shader_ViewportWidth
#PB_Shader_ViewportHeight
#PB_Shader_InverseViewportWidth
#PB_Shader_InverseViewportHeight
#PB_Shader_ViewportSize
#PB_Shader_ViewDirection
#PB_Shader_ViewSideVector
#PB_Shader_ViewUpVector
#PB_Shader_Fov
#PB_Shader_NearClipDistance
#PB_Shader_FarClipDistance
#PB_Shader_PassNumber
#PB_Shader_PassIterationNumber
#PB_Shader_AnimationParametric
#PB_Shader_TexelOffsets
#PB_Shader_SceneDepthRange
#PB_Shader_ShadowSceneDepthRange
#PB_Shader_ShadowColour
#PB_Shader_TextureSize
#PB_Shader_InverseTextureSize
#PB_Shader_PackedTextureSize
#PB_Shader_TextureMatrix
#PB_Shader_LodCameraPosition
#PB_Shader_LodCameraPositionObjectSpace
#PB_Shader_LightCustom
```

Extra.f Valeur supplémentaire transmise au paramètre de programme.

Valeur de retour

Aucune.

OS Supportés

Tous

114.9 MaterialShaderParameter

Syntaxe

```
MaterialShaderParameter(#Matiere,
    TypeProgramme,
   NomParametre$,
   TypeParametre, v1.f, v2.f,
   v3.f, v4.f)
```

Description

Définit une valeur de paramètre pour la matière basée sur le shader créé précédemment avec CreateShaderMaterial() .

Arguments

#**Matiere** La matière à utiliser.

TypeProgramme Type de programme. Il peut s'agir de l'une des valeurs suivantes :

```
#PB_Shader_Vertex   :
 Utiliser le programme
 vertex.
#PB_Shader_Fragment:
 Utiliser le programme
 vertex de fragment.
```

NomParametre$ Nom du paramètre à utiliser avec le programme GLSL.

TypeParametre Type de paramètre. Il peut s'agir de l'une des valeurs suivantes :

```
#PB_Shader_Integer: Le
 paramètre utilise un
 type entier (les
 paramètres v2, v3 et v4
 sont ignorés)
#PB_Shader_Float  : Le
 paramètre utilise un
 type float (les
 paramètres v2, v3 et v4
 sont ignorés)
#PB_Shader_Vector3: Le
 paramètre utilise un
 type Vector3 (le
 paramètre v4 est ignoré)
#PB_Shader_Vector4: Le
 paramètre utilise un
 type Vector4
```

v1 Valeur du premier paramètre. Obligatoire pour tous les types de paramètres.

v2 Valeur du deuxième paramètre. Obligatoire pour les types #PB_Shader_Vector3 et #PB_Shader_Vector4.

v3 Valeur du troisième paramètre. Obligatoire pour les types #PB_Shader_Vector3 et #PB_Shader_Vector4.

v4 Valeur du quatrième paramètre. Obligatoire pour le type #PB_Shader_Vector4.

Valeur de retour

Aucune.

Remarques

Les shaders GLSL utilisent le type vec4 pour transmettre une couleur. Exemple :

```
MaterialShaderParam(0,
  #PB_Shader_Fragment, 4,
  Red(Couleur)/255,
  Green(Couleur)/255,
  Blue(Couleur)/255,
  Alpha(Couleur)/255)
```

OS Supportés

Tous

114.10 MaterialShaderTexture

Syntaxe

```
MaterialShaderTexture(#Matiere,
    TextureID1, TextureID2,
    TextureID3, TextureID4)
```

Description

Définit les textures à utiliser pour la matière basée sur le shader créé précédemment avec CreateShaderMaterial() .

Arguments

#Matiere La matière basée sur le shader à utiliser.

TextureID1 La première texture à utiliser. Utiliser TextureID() pour obtenir un TextureID valide, ou #Null si vous n'en avez pas besoin.

TextureID2 La seconde texture à utiliser. Utiliser TextureID() pour obtenir un TextureID valide, ou #Null si vous n'en avez pas besoin.

TextureID3 La troisième texture à utiliser.
Utiliser TextureID() pour obtenir un TextureID valide, ou #Null si vous n'en avez pas besoin.

TextureID4 La quatrième texture à utiliser.
Utiliser TextureID() pour obtenir un TextureID valide, ou #Null si vous n'en avez pas besoin.

Valeur de retour

Aucune.

OS Supportés

Tous

114.11 DisableMaterialLighting

Syntaxe

```
DisableMaterialLighting(#Matiere,
    Etat)
```

Description

Active ou désactive l'éclairage dynamique d'une matière.

Arguments

#**Matiere** La matière à utiliser.

Etat #True : L'éclairage dynamique est désactivé
#False: L'éclairage dynamique est activé

Valeur de retour

Aucune.

Remarques

L'objet qui utilisera cette matière ne sera pas affecté par les lumières dynamiques créées par CreateLight() .
L'éclairage d'une matière est activée par défaut quand une matière est créée.
Pour obtenir l'état courant du matériau, voir GetMaterialAttribute()

OS Supportés

Tous

114.12 FreeMaterial

Syntaxe

```
FreeMaterial(#Matiere)
```

Description

Libère et détruit une matière.

Arguments

#**Matiere** La matière à détruire.

Valeur de retour

Aucune.

Remarques

Les propriétés ainsi que la mémoire occupée par la matière sont libérées, donc cette matière ne peut plus être utilisée par un objet.
Attention : Tous les objets (entités, billboards, etc) qui utilisent cette matière devront être modifiés avant la destruction de la matière).
Toutes les matières sont automatiquement libérées quand le programme se termine.

Voir aussi

CreateMaterial()

OS Supportés

Tous

114.13 IsMaterial

Syntaxe

```
Resultat =
    IsMaterial(#Matiere)
```

Description

Teste si une matière est correctement initialisée.

Arguments

#**Matiere** La matière à utiliser.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage.
C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.

Voir aussi

CreateMaterial() , MaterialID()

OS Supportés

Tous

114.14 GetMaterialAttribute

Syntaxe

```
Resultat =
    GetMaterialAttribute(#Matiere,
    Attribut)
```

Description

Renvoie l'attribut de la matière.

Arguments

#**Matiere** La matière à utiliser.

Attribut L'attribut à obtenir. Peut être une des valeurs suivantes :

- #PB_Material_Shininess : La brillance tel que définie avec MaterialShininess()
- #PB_Material_TextureRotate : La rotation, en degré.
- #PB_Material_TextureUScale : La valeur uscale, voir ScaleMaterial()
- #PB_Material_TextureVScale : La valeur vscale, voir ScaleMaterial()
- #PB_Material_TextureUScroll: La valeur uscroll, voir ScrollMaterial()
- #PB_Material_TextureVScroll: La valeur vscroll, voir ScrollMaterial()
- #PB_Material_DepthCheck : L'état de contrôle de profondeur (activée ou désactivée).
- #PB_Material_DepthWrite : L'état d'écriture de profondeur (activée ou désactivée).
- #PB_Material_Lighting : L'éclairage, voir DisableMaterialLighting()

.
#PB_Material_ShadingMode : La teinte, voir MaterialShadingMode()
.
#PB_Material_CullingMode : Le mode d'élimination de la matière, voir MaterialCullingMode()
.

Valeur de retour

Renvoie la valeur de l'attribut spécifié. Les valeurs des canaux de couleur peuvent facilement être récupérées avec Red() , Green() et Blue() .

OS Supportés

Tous

114.15 GetMaterialColor

Syntaxe

```
Resultat =
    GetMaterialColor(#Matiere,
    Type)
```

Description

Renvoie la couleur d'une matière.

Arguments

#**Matiere** La matière à utiliser.

Type Le type de couleur à obtenir. Peut être une des valeurs suivantes :

```
#PB_Material_AmbientColor:
  La couleur utilisée par
 défaut, sans autre
 éclairage.
#PB_Material_DiffuseColor:
  La couleur que le
 matériau reflètera quand
 il sera éclairé de façon
 dynamique.
                          Par
 exemple, utiliser une
 couleur blanche se
 traduira par un
 éclairage normal
                          (toutes
 les couleurs sont
 réfléchies). Si vous
 utilisez un rouge, alors
 uniquement
```

```
                                les
    couleurs rouges du
    matériau seront
    réfléchies, ce qui
    donnera un matériau
                                entièrement
    rouge (ou noir s'il ne
    contient aucune couleur
    rouge).
  #PB_Material_SpecularColor:
    La couleur du matériau
    qu'il réfléchira quand
    il sera dynamiquement
    éclairé par
                                une
    lumière qui a une valeur
    de couleur spéculaire.
    Par exemple, utiliser une
                                couleur
    blanche se traduira par
    un éclairage normal
    (toutes les couleurs sont
                                réfléchies).
    Si vous utilisez un
    rouge, alors uniquement
    les couleurs rouges
                                du
    matériau seront
    réfléchies, résultant en
    un matériau entièrement
    rouge
                                (ou
    noir si le matériau ne
    contient aucune couleur
    rouge).
  #PB_Material_SelfIlluminationColor:
    La couleur du matériau
    qu'il émettra, même si
    aucune lumière ne
    l'atteint.
```

Valeur de retour

Renvoie la valeur de la couleur RGB.
Les valeurs des canaux de couleur peuvent facilement être récupérées avec Red() , Green() et Blue() .

OS Supportés

Tous

114.16 SetMaterialColor

Syntaxe

```
SetMaterialColor(#Matiere,
   Type, Couleur)
```

Description

Modifie la couleur de la matière.

Arguments

#**Matiere** La matière à utiliser.

Type Le type de couleur à obtenir. Peut être une combinaison des valeurs suivantes :

```
#PB_Material_AmbientColor:
  La couleur utilisée par
 défaut, sans autre
 éclairage.
#PB_Material_DiffuseColor:
  La couleur que le
 matériau reflètera quand
 il sera éclairé de façon
 dynamique.
                          Par
 exemple, utiliser une
 couleur blanche se
 traduira par un
 éclairage normal
                          (toutes
 les couleurs sont
 réfléchies). Si vous
 utilisez un rouge, alors
 uniquement
                          les
 couleurs rouges du
 matériau seront
 réfléchies, ce qui
 donnera un matériau
                          entièrement
 rouge (ou noir s'il ne
 contient aucune couleur
 rouge).
#PB_Material_SpecularColor:
 La couleur du matériau
 qu'il réfléchira quand
 il sera dynamiquement
 éclairé par
                          une
 lumière qui a une valeur
 de couleur spéculaire.
 Par exemple, utiliser une
                          couleur
 blanche se traduira par
 un éclairage normal
 (toutes les couleurs sont
                          réfléchies).
 Si vous utilisez un
 rouge, alors uniquement
 les couleurs rouges
                          du
 matériau seront
 réfléchies, résultant en
```

un matériau entièrement rouge (ou noir si le matériau ne contient aucune couleur rouge).
#PB_Material_SelfIlluminationColor: La couleur du matériau qu'il émettra, même si aucune lumière ne l'atteint.

Couleur Valeur de la couleur RGB. Une valeur RGB valide peut être créé avec RGB() .

Valeur de retour

Aucune.

OS Supportés

Tous

114.17 MaterialBlendingMode

Syntaxe

```
MaterialBlendingMode(#Matiere,
    Mode)
```

Description

Change la manière dont une matière sera mixée avec le monde 3D et donc la manière dont l'objet qui utilise cette matière sera rendue.

Arguments

#**Matiere** La matière à utiliser.

Mode #PB_Material_Add : Utilise le mode addition des pixels (Add) pour mixer l'objet avec la scène (donc la couleur noire (0,0,0) est considérée comme transparente).
#PB_Material_AlphaBlend: Utilise le canal alpha de la texture (TGA ou PNG) pour le mixage avec la scène.

```
    #PB_Material_Color      :
     Utilise la couleur
     transparente de la
     texture pour le mixage
     avec la scène.
```

Valeur de retour

Aucune.

Voir aussi

MaterialFilteringMode() ,
MaterialShadingMode()

OS Supportés

Tous

114.18 MaterialFilteringMode

Syntaxe

```
MaterialFilteringMode(#Matiere,
    Mode [,
    ValeurAnisotropicMax])
```

Description

Change le mode filtrage d'une matière.

Arguments

#Matiere La matière à utiliser.
Si #Matière est réglé sur #PB_Default alors le mode de filtrage utilisé pour la future matière est modifié.

Mode
```
    #PB_Material_None
              : Aucun filtrage
     n'est effectué, ce qui
     donne un résultat très
     pixelisé quand la
     matière est zoomée.
    #PB_Material_Bilinear     :
     Effectue un filtre
     bi-linéaire quand la
     texture zoome, ce qui
     donne une texture plus
     agréable, mais un peu
     floue.
    #PB_Material_Trilinear    :
     Effectue un filtre
     tri-linéaire quand la
     texture zoome, ce qui
     donne le meilleur rendu
     possible.
```

```
#PB_Material_Anisotropic :
 Règle la valeur maximale
 anisotropique.
 'ValeurAnisotropicMax'
 est généralement
 comprise entre 1 et 8.
```

ValeurAnisotropicMax (optionnel)
Valeur maximale du filtrage anisotropique avec l'option #PB_Material_Anisotropic.
Valeur généralement comprise entre 1 et 8.

Valeur de retour

Aucune.

Remarques

Quand une matière est créée, le filtre bi-lineaire est activé par défaut.
Utiliser un filtrage n'a plus un grand impact sur les performances vidéos car toutes les cartes graphiques modernes le font au niveau matériel.

Voir aussi

MaterialBlendingMode() , MaterialShadingMode()

OS Supportés

Tous

114.19 MaterialID

Syntaxe

```
Resultat =
    MaterialID(#Matiere)
```

Description

Renvoie l'identifiant (MaterialID) d'une matière.

Arguments

#Matiere La matière à utiliser.

Valeur de retour

Renvoie le numéro MaterialID de la matière. C'est l'identifiant unique de la matière requise par la commande SetEntityMaterial() .

Voir aussi

CreateMaterial() , MaterialID()

OS Supportés

Tous

114.20 MaterialShadingMode

Syntaxe

```
MaterialShadingMode(#Matiere,
    Mode)
```

Description

Change le mode d'ombrage d'une matière.

Arguments

#**Matiere** La matière à utiliser.

Mode Peut être l'une des valeurs suivantes :

```
#PB_Material_Flat      : La
 matière sera rendue en
 mode solide, l'ombrage
 est effectué face par
 face.
#PB_Material_Gouraud   :
 Effectue un ombrage
 progressif en utilisant
 l'algorithme de Gouraud
 (Par défaut).
#PB_Material_Phong     :
 Effectue un ombrage
 progressif en utilisant
 l'algorithme de Phong.
```

combinée avec l'une des valeurs suivantes :

```
#PB_Material_Solid     : La
 matière sera rendue en
 mode solide et texturé
 (par défaut).
#PB_Material_Wireframe: La
 matière sera rendue en
 mode fils de fer.
#PB_Material_Point     : La
 matière sera rendue en
 utilisant des points
 d'arêtes seulement.
```

Valeur de retour

Aucune.

Remarques

Pour obtenir l'état courant de ce mode, voir GetMaterialAttribute() .

Voir aussi

MaterialBlendingMode() , MaterialFilteringMode()

OS Supportés

Tous

114.21 MaterialCullingMode

Syntaxe

```
MaterialCullingMode(#Matiere,
    Mode)
```

Description

Modifie le mode élimination de la matière. Le 'culling' est le fait d'éliminer prématurément des parties de la scène qui ne seront pas visibles à l'écran, soit parce qu'elles sont cachées, soit parce qu'elles ne sont pas dans le champ de vision.

Arguments

#**Matiere** La matière à utiliser.

Mode Peut être l'une des valeurs suivantes :

```
#PB_Material_NoCulling
        : Pas
  d'élimination.
#PB_Material_ClockWiseCull
    : Elimination dans le
  sens horaire.
#PB_Material_AntiClockWiseCull:
  Elimination dans le sens
  anti-horaire.
```

Valeur de retour

Aucune.

Remarques

Pour obtenir l'état courant de ce mode, voir GetMaterialAttribute() .

OS Supportés

Tous

114.22 MaterialShininess

Syntaxe

```
MaterialShininess(#Matiere,
    Brillance [,
    CouleurSpeculaire])
```

Description

Modifie la brillance d'une matière.

Arguments

#Matiere La matière à utiliser.

Brillance Nouvelle brillance.

CouleurSpeculaire (optionnel) La couleur spéculaire à utiliser.
Une couleur RVB valide peut être créée avec RGB() .

Valeur de retour

Aucune.

Remarques

Pour obtenir l'état courant de ce mode, voir GetMaterialAttribute() .

Voir aussi

MaterialFog()

OS Supportés

Tous

114.23 MaterialTextureAliases

Syntaxe

```
MaterialTextureAliases(#Matiere,
    TextureID1, TextureID2,
    TextureID3, TextureID4)
```

Description

Définit les textures à utiliser dans un script de matériau.

Arguments

#Matiere La matière à utiliser.

TextureID1 Le TextureID() à utiliser pour le premier alias de texture (identifié comme 'texture_alias texture1' dans le script de matériau), ou zéro si aucune texture n'est nécessaire.

TextureID2 Le TextureID() à utiliser pour le second alias de texture (identifié comme 'texture_alias texture2' dans le script de matériau), ou zéro si aucune texture n'est nécessaire.

TextureID3 Le TextureID() à utiliser pour le troisième alias de texture (identifié comme 'texture_alias texture3' dans le script de matériau), ou zéro si aucune texture n'est nécessaire.

TextureID4 Le TextureID() à utiliser pour le quatrième alias de texture (identifié comme 'texture_alias texture4' dans le script de matériau), ou zéro si aucune texture n'est nécessaire.

Remarques

Cela permet d'utiliser le même script de matériau avec des textures dynamiques. Dans le script de matériau, vous devez remplacer "texture mytexture.jpg" par "texture_alias texture1" (ou "texture_alias texture2", "texture_alias texture3", "texture_alias texture4").

Valeur de retour

Aucune.

Voir aussi

GetScriptMaterial() , TextureID()

OS Supportés

Tous

114.24 GetScriptMaterial

Syntaxe

```
Resultat =
    GetScriptMaterial(#Matiere,
    Nom$)
```

Description

Renvoie un numéro de matière défini dans un fichier de script OGRE.

Arguments

#**Matiere** La matière à utiliser.

Nom$ Nom du fichier script OGRE.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro si la matière n'a pas été trouvée ou n'a pas pu être chargée.

Voir aussi

ReloadMaterial() , ResetMaterial()

OS Supportés

Tous

114.25 MaterialFog

Syntaxe

```
MaterialFog(#Matiere,
    Couleur, Intensite,
    DistanceDebut, DistanceFin)
```

Description

Crée un effet de brouillard sur la #Matière.

Arguments

#Matiere La matière à utiliser.

Couleur La couleur du brouillard. RGB() peut être utilisé pour obtenir une 'Couleur' valide.

Intensite Avec une valeur égale à zéro, l'effet de brouillard est retiré.

DistanceDebut Distance (en unités du monde) d'où le brouillard doit commencer.

DistanceFin Distance (en unités du monde) d'où le brouillard est totalement opaque.

Valeur de retour

Aucune.

Voir aussi

MaterialShininess()

OS Supportés

Tous

114.26 ReloadMaterial

Syntaxe

```
ReloadMaterial(NomMatiere$,
    NomFichierScript$,
    ParseScript)
```

Description

Recharge une matière à partir d'un script OGRE.

Arguments

NomMatiere$ Le nom de la matière à utiliser.

NomFichierScript$ Le nom du fichier script à utiliser.

ParseScript #True : Le 'NomFichierScript$' est analysé à nouveau pour obtenir des informations de mises à jour sur la matière.

Valeur de retour

Aucune.

Remarques

Ceci est utile lors de l'utilisation de matériaux (matières) personnalisé(e)s stocké(e)s dans des fichiers de script.
Un matériau peut être obtenu à partir des scripts OGRE par GetScriptMaterial() .

Voir aussi

GetScriptMaterial() , ResetMaterial()

OS Supportés

Tous

114.27 ResetMaterial

Syntaxe

```
ResetMaterial(TypeObjet)
```

Description

Réinitialise tous les matériaux pour les types d'objets spécifiés.

Arguments

TypeObjet
```
    #PB_Entity          :
  Réinitialisation des
  matériaux pour toutes
  les entités.
 #PB_ParticleEmitter:
  Réinitialisation des
  matériaux pour tous les
  émetteurs de particules.
 #PB_BillboardGroup :
  Réinitialisation des
  matériaux pour tous les
  Billboards.
```

Valeur de retour

Aucune.

Voir aussi

GetScriptMaterial() , ReloadMaterial()

OS Supportés

Tous

114.28 SetMaterialAttribute

Syntaxe

```
SetMaterialAttribute(#Matiere,
  Attribut, Valeur [,
  Couche])
```

Description

Définit la valeur de l'attribut spécifié au matériau.

Arguments

#**Matiere** La matière à utiliser.

Attribut L'attribut à définir. Il peut prendre l'une des valeurs suivantes :

```
  #PB_Material_DepthCheck
   : Active ou désactive le
   contrôle de la
   profondeur de la matière.
                          La
   valeur peut être #True
   (contrôle de profondeur
   activé) ou
                          #False
   (contrôle de profondeur
   désactivé).
```

#PB_Material_DepthWrite
: Active ou désactive
l'écriture de profondeur
pour le matériau.
La
valeur peut être #True
(profondeur écriture
activée) ou
#False
(profondeur écriture
désactivée).
#PB_Material_AlphaReject
: Active le rejet alpha
sur la texture (utile
pour les textures
semi-transparentes
comme les arbres, les
fenêtres, etc.).
#PB_Material_TAM
: Change le 'Mode
d'adressage de texture'
(TAM).
Il
peut s'agir de l'une des
valeurs suivantes
-
#PB_Material_WrapTAM :
Mode enveloppement
-
#PB_Material_MirrorTAM:
Mode miroir
-
#PB_Material_ClampTAM :
Mode serrage
-
#PB_Material_BorderTAM:
Mode frontière
#PB_Material_EnvironmentMap:
Active la Map (carte) de
l'environnement.
Il
peut s'agir de l'une des
valeurs suivantes
-
#PB_Material_NoMap
: Désactive
l'environnement
-
#PB_Material_PlanarMap
: Map de
l'environnement planaire
-
#PB_Material_CurvedMap
: Map de
l'environnement courbe
-
#PB_Material_ReflectionMap:
Map de l'environnement
de réflexion

-
#PB_Material_NormalMap
: Map de
l'environnement normal
#PB_Material_ProjectiveTexturing:
Active la texturation
projective pour ce
matériau. La valeur est
un numéro de #Camera à
utiliser.
#PB_Material_PointSprite
: Active ou désactive le
point sprite pour le
matériau.
Si
supérieur à zéro, la
valeur est la taille en
points/1000. Si zéro, la
taille en points est
désactivée
#PB_Material_DepthBias
:Active ou désactive le
bias de profondeur pour
le matériau.
Si
supérieur à zéro, la
valeur est le bias de
profondeur/1000. Si
zéro, le bias de
profondeur est désactivé

Valeur Valeur de l'attribut à définir.

Couche (optionnel) La couche à utiliser. La première couche commence à l'index zéro (si ce paramètre est omis, la couche zéro est utilisée).

Valeur de retour

Aucune.

Voir aussi

GetMaterialAttribute()

OS Supportés

Tous

114.29 ScrollMaterial

Syntaxe

```
ScrollMaterial(#Matiere, X,
   Y, Mode [, Couche])
```

Description

Déplace une couche d'une matière.

Arguments

#**Matiere** La matière à utiliser.

X, Y Les valeurs du déplacement.

Mode

```
    #PB_Material_Fixed    :
  La matière est déplacée
  de manière absolue, sans
  tenir compte des
  déplacements précédents.
#PB_Material_Animated: A
  chaque rendu, la couche
  est déplacée des valeurs
  x,y automatiquement
  (relatif au déplacement
  précédent).
```

Couche (optionnel) La couche à déplacer.
La première couche commence à l'index zéro. (Si ce paramètre est omis, la couche zéro est déplacée).

Valeur de retour

Aucune.

Remarques

Pour obtenir l'état courant de ce mode, voir GetMaterialAttribute() .

Voir aussi

RotateMaterial() , ScaleMaterial()

OS Supportés

Tous

114.30 RemoveMaterialLayer

Syntaxe

```
RemoveMaterialLayer(#Matiere)
```

Description

Supprime la dernière couche ajoutée sur une matière.

Arguments

#**Matiere** La matière à utiliser.

Valeur de retour

Aucune.

Voir aussi

AddMaterialLayer() ,
CountMaterialLayers()

OS Supportés

Tous

114.31 ScaleMaterial

Syntaxe

```
ScaleMaterial(#Matiere, X, Y
    [, Couche])
```

Description

Mise à l'échelle (agrandissement ou rapetissement) d'une matière.

Arguments

#**Matiere** La matière à utiliser.

X, Y Facteurs d'échelle (la taille est multipliée par ces valeurs) :
Une valeur de 1.0 signifie que la taille n'est pas modifiée
Une valeur comprise entre 0.0 et 1.0 de signifie que le matériau est réduit (par exemple : une échelle de 0.5 fera la moitié de la taille).

Couche (optionnel) La couche à utiliser. La première couche commence à l'index zéro. (Si ce paramètre est omis, la couche zéro est utilisée).

Valeur de retour

Aucune.

Remarques

Pour obtenir l'état courant de ce mode, voir GetMaterialAttribute() .

Voir aussi

RotateMaterial() , ScrollMaterial()

OS Supportés

Tous

114.32 RotateMaterial

Syntaxe

```
RotateMaterial(#Matiere,
    Angle, Mode [, Couche])
```

Description

Rotation d'une matière.

Arguments

#**Matiere** La matière à utiliser.

Angle Angle de rotation.

Mode

```
      #PB_Material_Fixed   :
    La matière est inclinée
    de manière absolue, sans
    tenir compte de
    l'inclinaison précédente.
  #PB_Material_Animated: A
    chaque rendu, la couche
    est automatiquement
    inclinée des valeurs x,y
                         (relatif
    à l'inclinaison
    précédente).
```

Couche La couche à déplacer.
La première couche commence à l'index zéro. (Si ce paramètre est omis, la couche zéro est utilisée).

Valeur de retour

Aucune.

Remarques

Pour obtenir l'état courant de ce mode, voir GetMaterialAttribute() .

Voir aussi

ScaleMaterial() , ScrollMaterial()

OS Supportés

Tous

114.33 MaterialAnimation

Syntaxe

```
MaterialAnimation(#Matiere,
    Texture$, NbImageAnimee,
    Temps.f)
```

Description

Ajoute une texture animée à la matière.

Arguments

#**Matiere** La matière à utiliser.

Texture$ Le nom du fichier texture à utiliser.

NbImageAnimee Le nombre d'images animées (frames) de la texture animée.

Temps.f (optionnel) Temps (en secondes) pour lire toute l'animation. Une fois entièrement joué, elle recommencera en boucle depuis le début.

Valeur de retour

Aucune.

Remarques

Une texture animée est composée de n'importe quel nombre de textures, toutes de la même taille, avec le numéro d'image ajouté avant l'extension dans leur nom de fichier.
Par exemple, si "test.jpg" est spécifié comme "Texture$" et "NbImageAnimee" est réglé sur 3, les textures "test_0.jpg", "test_1.jpg" et "test_2.jpg" seront chargées et utilisées sur la matière.
D'habitude, toutes les images sont contenues dans un fichier zip portant le même nom que texture (ex Texture.zip) et accessible par Add3DArchive() et Parse3DScripts() .

OS Supportés

Tous

Chapitre 115

Math

Généralités

La bibliothèque Math propose des fonctions mathématiques telles que Cos(), Sin(), Pow(), Log() etc. La plupart des commandes fonctionnent avec des nombres flottants (.f) ou des doubles (.d). **Si une commande est utilisée avec une valeur de type double (.d) en entrée ou en sortie, alors la version double précision de la commande sera automatiquement utilisée.**

OS Supportés

Tous

115.1 Abs

Syntaxe

```
Resultat.f(d) =
    Abs(Nombre.f(d))
```

Description

Renvoie la valeur absolue (non signée) d'un nombre flottant.

Arguments

Nombre.f ou Nombre.d Le nombre à virgule flottante (float ou double)

Valeur de retour

Renvoie la valeur absolue et **toujours positive**.

Remarques

Cette fonction ne peut être utilisée correctement qu'avec des nombres flottants jamais avec des nombres entiers.

Exemple

```
1  Debug Abs(3.14159)  ;
     Affiche '3.14159...'
2  Debug Abs(-3.14159) ;
     Attention, affiche
     '3.14159...'
3
4  pif.f = -3.14159
5  pid.d = -3.14159
6  Debug Abs(pif)  ; Renvoie
     un float
7  Debug Abs(pid)  ; Renvoie
     un double
```

Voir aussi

Sign()

OS Supportés

Tous

115.2 ACos

Syntaxe

```
Resultat.f(d) =
    ACos(Nombre.f(d))
```

Description

Renvoie l'arc cosinus d'un nombre.

Arguments

Nombre.f ou Nombre.d Le nombre à virgule flottante (float ou double)
Le nombre doit être compris entre -1.0 et 1.0 inclus.

Valeur de retour

Renvoie l'angle en radian.

Remarques

La valeur renvoyée peut être transformée en degré en utilisant la fonction Degree() .
C'est la fonction réciproque de Cos() .

Exemple

```
1  Debug ACos(1)  ; Affiche
     '0.0'
2  Debug Acos(-1) ; Affiche
     '3.14159...' (pi)
```

Voir aussi

Cos() , ACosH() , Degree()

OS Supportés

Tous

115.3 ACosH

Syntaxe

```
Resultat.f(d) =
    ACosH(Nombre.f(d))
```

Description

Renvoie l'arc cosinus hyperbolique d'un nombre.

Arguments

Nombre.f ou Nombre.d Le nombre à virgule flottante (float ou double)
Le nombre doit être supérieur ou égal à 1.

Valeur de retour

Renvoie l'angle hyperbolique.

Remarques

C'est la fonction réciproque de CosH() .

Exemple

```
1  Debug ACosH(1)
                    ; Affiche
     '0.0'
2  Debug Exp(ACosH(0.5 *
     Sqr(5))) ; Affiche
     '1.618033...' (le nombre
     d'or)
```

Voir aussi

CosH() , ACos()

OS Supportés

Tous

115.4 ASin

Syntaxe

```
Resultat.f(d) =
    ASin(Nombre.f(d))
```

Description

Renvoie l'arc sinus d'un nombre.

Arguments

Nombre.f ou Nombre.d Le nombre à virgule flottante (float ou double)
La valeur du nombre doit être comprise entre -1.0 et 1.0.

Valeur de retour

Renvoie l'angle en radian.

Remarques

La valeur renvoyée peut être transformée en degré en utilisant la fonction Degree() .
C'est la fonction réciproque de Sin() .

Exemple

```
1  Debug ASin(1) ; Affiche
     '1.570796...' (pi/2)
2  Debug ASin(0) ; Affiche
     '0.0'
```

Voir aussi

Sin() , ASinH() , Degree()

OS Supportés

Tous

115.5 ASinH

Syntaxe

```
Resultat.f(d) =
    ASinH(Nombre.f(d))
```

Description

Renvoie l'arc sinus hyperbolique d'un nombre.

Arguments

Nombre.f ou Nombre.d Le nombre à virgule flottante (float ou double)
L'intervalle de valeur n'est pas limité.

Valeur de retour

Renvoie l'angle hyperbolique.

Remarques

C'est la fonction réciproque de SinH() .

Exemple

```
1  Debug ASinH(0)          ;
     Affiche '0.0'
2  Debug Exp(ASinH(0.5)) ;
     Affiche '1.618033...' (Le
     nombre d'Or)
```

Voir aussi

SinH() , ASin()

OS Supportés

Tous

115.6 ATan

Syntaxe

```
Resultat.f(d) =
    ATan(Nombre.f(d))
```

Description

Renvoie l'arc tangente d'un nombre.

Arguments

Nombre.f ou Nombre.d Le nombre à virgule flottante (float ou double)
Le nombre peut prendre n'importe quelle valeur.

Valeur de retour

Renvoie l'angle en radian.

Remarques

La valeur renvoyée peut être transformée en degré en utilisant la fonction Degree() .
C'est la fonction réciproque de Tan() .

Exemple

```
1   Debug ATan(1) ; Affiche
      '0.785398...' (pi/4)
```

Voir aussi

Tan() , ATanH() , Degree()

OS Supportés

Tous

115.7 ATan2

Syntaxe

```
Resultat.f(d) = ATan2(X.f(d),
    Y.f(d))
```

Description

Renvoie l'angle formé par la droite qui passe par (X ;Y) et par l'origine (0 ;0) d'une part et par l'axe des abscisses (axe des X) d'autre part.

Arguments

X.f ou X.d, Y.f ou Y.d Nombres à virgule flottante (float ou double)

Valeur de retour

Renvoie l'angle en radian.
Le résultat est toujours compris entre - #PI et + #PI.
ATTENTION : Si X et Y sont égaux à zéro alors la commande renvoie zéro au lieu de lever une erreur de type 'division par zéro'.

Remarques

La valeur renvoyée peut être transformée en degré en utilisant la fonction Degree() .
C'est la fonction réciproque de Tan()
L'intérêt de cette fonction et qu'elle prend en compte le signe des deux coordonnées x, y et place l'angle dans le bon quadrant.
Ainsi, atan2(1, 1) = PI/4 et atan2(-1, -1) = -3 x PI/4 contrairement à ATan() qui renverrait PI/4 dans les deux cas.
C'est utile pour calculer des angles entre des lignes en 2D, ou pour transformer des coordonnées rectangulaires en coordonnées polaires.

Exemple

```
1  Debug ATan2(0, 0)     ;
     Affiche 0.0 au lieu de
     lever une alerte division
     par zéro.
2  Debug ATan2(1, 1)     ;
     Affiche #PI/4   (45 degrés)
3  Debug ATan2(-1, 1)    ;
     Affiche 3#PI/4  (135
     degrés)
4  Debug ATan2(-1, -1)   ;
     Affiche -3#PI/4 (5#PI/4,
     225 degrés)
5  Debug ATan2(1, -1)    ;
     Affiche -#PI/4  (7#PI/4
     315 degrés)
6  Debug ATan2(1, 0)     ;
     Affiche 0.0
```

Voir aussi

ATan() , Degree()

OS Supportés

Tous

115.8 ATanH

Syntaxe

```
Resultat.f(d) =
    ATanH(Nombre.f(d))
```

Description

Renvoie l'arc tangente hyperbolique d'un nombre.

Arguments

Nombre.f ou Nombre.d Le nombre à virgule flottante (float ou double) Doit être compris entre -1.0 et 1.0, en excluant -1.0 et 1.0.

Valeur de retour

Renvoie l'angle hyperbolique.

Remarques

C'est la fonction réciproque de TanH() .

Exemple

```
Debug Exp(ATanH(0.2 *
  Sqr(5))) ; Affiche
  '1.618033...' (le nombre
  d'or)
```

Voir aussi

TanH() , ATan()

OS Supportés

Tous

115.9 Cos

Syntaxe

```
Resultat.f(d) =
    Cos(Angle.f(d))
```

Description

Renvoie le cosinus d'un angle.

Arguments

Angle.f ou Angle.d L'angle en radians. (Voir float ou double)

Valeur de retour

Renvoie le cosinus de l'angle.
Toujours entre -1.0 and 1.0.

Remarques

Un angle en degré doit être converti en radian avec la fonction Radian() .
C'est la fonction réciproque de ACos() .

Exemple

```
Debug Cos(3.141593) ;
  Affiche '-1.0'
```

Voir aussi

ACos() , CosH() , Radian()

OS Supportés

Tous

115.10 CosH

Syntaxe

```
Resultat.f(d) =
    CosH(Angle.f(d))
```

Description

Renvoie le cosinus hyperbolique d'un angle.

Arguments

Angle.f ou Angle.d L'angle en radians.
(float ou double)

Valeur de retour

Renvoie le cosinus hyperbolique de l'angle spécifié en radian.
C'est un nombre 'float' ou 'double' (voir float ou double)
Toujours supérieur ou égal à 1.0.

Remarques

C'est la fonction réciproque de ACosH() .

Exemple

```
1   Debug CosH(0)   ; Affiche 1.0
```

Voir aussi

ACosH() , Cos()

OS Supportés

Tous

115.11 Degree

Syntaxe

```
Resultat.f(d) =
    Degree(Angle.f(d))
```

Description

Convertit un angle en radian, en degré.

Arguments

Angle.f ou Angle.d L'angle en radians.
(float ou double)

Valeur de retour

Renvoie l'angle en degré.
C'est un nombre 'float' ou 'double' (voir float ou double)

Remarques

Il n'y a aucune normalisation pour s'assurer que l'angle renvoyé est compris entre 0 et 360. Si l'entrée est supérieure à 2 x PI alors le résultat sera plus grand que 360. De même, une entrée négative se traduira par une sortie négative.
C'est la fonction réciproque de Radian() .

Exemple

```
1   Debug Degree(#PI/4) ;
      renvoie 45.0
```

Voir aussi

Radian()

OS Supportés

Tous

115.12 Exp

Syntaxe

```
Resultat.f(d) =
    Exp(Nombre.f(d))
```

Description

Renvoie l'exponentielle d'un nombre.

Arguments

Nombre.f ou Nombre.d Le nombre.
(Voir float ou double)

Valeur de retour

Renvoie la valeur de 'e' élevée à la puissance 'Nombre'.
C'est un nombre 'float' ou 'double' (voir float ou double)

Remarques

C'est la fonction réciproque de Log() .

Voir aussi

Log() , Pow()

OS Supportés

Tous

115.13 Infinity

Syntaxe

```
Resultat.f(d) = Infinity()
```

Description

Renvoie la valeur flottante spéciale représentant l'infini.

Arguments

Aucun.

Valeur de retour

Renvoie la valeur représentant l'infini positif.
C'est un nombre 'float' ou 'double' (voir float ou double) suivant la variable utilisée.

Remarques

L'infini négatif peut être calculé en utilisant "-Infinity()".
Infinity et l'infini négatif sont des valeurs particulières. Elles se comportent dans les calculs de la façon souhaitée, par exemple diviser par un nombre infini positif (sauf 0 ou infini) entraînera l'infini à nouveau.
La fonction IsInfinity() peut être utilisée pour vérifier si une valeur représente l'infini positif ou négatif.

Exemple

```
1  Debug IsInfinity(Infinity()
     / 1000) ; afiche 1.0
```

Voir aussi

IsInfinity() , NaN()

OS Supportés

Tous

115.14 Int

Syntaxe

```
Resultat = Int(Nombre.f(d))
```

Description

Renvoie la partie entière d'un nombre flottant.

Arguments

Nombre.f ou Nombre.d Le nombre à virgule flottante (float ou double)

Valeur de retour

Renvoie la partie entière sans arrondi (par troncature à l'unité).

Remarques

Cette fonction renvoie un integer .
Pour obtenir un quad, utiliser IntQ() .
Pour obtenir un arrondi, utiliser Round() .

Exemple

```
1  Debug Int(10.565)   ;
     Affiche '10'
2  Debug Int(10.1)     ;
     Affiche '10'
3  Debug Int(10.9)     ;
     Affiche '10'
4  Debug Int(-10.565)  ;
     Affiche '-10'
5  Debug Int(-10.1)    ;
     Affiche '-10'
6  Debug Int(-10.9)    ;
     Affiche '-10'
```

Voir aussi

IntQ() , Round()

OS Supportés

Tous

115.15 IntQ

Syntaxe

```
Resultat.q = IntQ(Nombre.f(d))
```

Description

Renvoie la partie entière d'un nombre flottant, au format quad.

Arguments

Nombre.f ou Nombre.d Le nombre à virgule flottante (float ou double)

Valeur de retour

Renvoie la partie entière sans arrondi (par troncature à l'unité).

Remarques

Cette fonction renvoie un quad . Pour obtenir un integer, utiliser la fonction Int() qui de plus, sera beaucoup plus rapide sur un système 32 bits.
Pour obtenir un arrondi, utiliser Round() .

Exemple

```
1  Debug IntQ(12345678901.565)
     ; Affiche '12345678901'
```

Voir aussi

Int() , Round()

OS Supportés

Tous

115.16 IsInfinity

Syntaxe

```
Resultat.f(d) =
    IsInfinity(Valeur.f(d))
```

Description

Renvoie une valeur non nulle si 'Valeur' représente un infini.

Arguments

Nombre.f ou Nombre.d Le nombre à virgule flottante suspecté d'être un infini. (float ou double)

Valeur de retour

Renvoie une valeur non nulle si la valeur représente l'infini positif ou négatif.

Remarques

Comparer les valeurs infinies n'est pas conseillé, car cela dépend de l'implémentation matérielle, si l'infini est égal à lui-même ou non. Les valeurs de l'infini négatif et positif peuvent être générées par la fonction Infinity() .

Exemple

```
1  Debug
     IsInfinity(Infinity())  ;
     infini
2  Debug IsInfinity(Log(0))
          ; -infini
3  Debug IsInfinity(1234.5)
          ; nombre fini
4  Debug IsInfinity(NaN())
           ; NaN (Attention,
     'Not a Number' n'est pas
     la même chose qu'un infini)
```

Voir aussi

Infinity() , IsNaN()

OS Supportés

Tous

115.17 IsNaN

Syntaxe

```
Resultat.f(d) =
    IsNaN(Valeur.f(d))
```

Description

Renvoie une valeur non nulle si 'Valeur' n'est pas un nombre (Not a Number).

Arguments

Nombre.f ou Nombre.d Le nombre à virgule flottante (float ou double)

Valeur de retour

Renvoie une valeur non nulle si la valeur n'est pas un nombre (Not a Number).

Remarques

Cette valeur est le résultat d'un calcul incorrect. Il peut aussi être généré à l'aide de la fonction NaN() .
NaN est une valeur spéciale. Tester sa valeur ne doit pas être fait avec une comparaison normale, car il y a de nombreuses valeurs pour NaN. Le fait que NaN soit considéré égal à lui-même dépend de l'implémentation matérielle.

Exemple

```
1  Debug IsNAN(NaN())        ;
     NaN
2  Debug IsNAN(Sqr(-1))      ;
     NaN
3  Debug IsNAN(1234.5)       ;
     un nombre
4  Debug IsNAN(Infinity())   ;
     Attention, l'infini n'est
     pas NaN
```

Voir aussi

NaN() , IsInfinity()

OS Supportés

Tous

115.18 Pow

Syntaxe

```
Resultat.f(d) =
    Pow(Nombre.f(d),
    Puissance.f(d))
```

Description

Renvoie un nombre élevé à une puissance.

Arguments

Nombre.f ou Nombre.d Le nombre à virgule flottante (float ou double)
La mantisse.

Puissance.f ou Puissance.d Le nombre à virgule flottante (float ou double)
L'exposant.
Si 'Nombre' est négatif alors l'exposant doit être un nombre entier.

Valeur de retour

Renvoie le 'Nombre' élevé à la puissance 'Puissance'.

Remarques

Attention, le symbole ' ^' n'est pas un opérateur de puissance et ne peut donc pas remplacer la fonction Pow().
Pour information, la racine n-ième de x s'extrait comme cela : Pow(x, 1/n)

Exemple

```
1  Debug Pow(2.0, 3.0)  ;
     Affiche '8.0'
2  Debug Pow(27.0, 1/3.0) ;
     Affiche '3.0', la racine
     cubique de 27
```

Voir aussi

Sqr() , Exp()

OS Supportés

Tous

115.19 Log

Syntaxe

```
Resultat.f(d) =
    Log(Nombre.f(d))
```

Description

Renvoie le logarithme népérien d'un nombre.

Arguments

Nombre.f ou Nombre.d Le nombre à virgule flottante (float ou double)

Valeur de retour

Renvoie le logarithme base 'e' (naturel). Souvent écrit sous la forme ln(x).

Remarques

C'est la fonction réciproque de Exp() .

Exemple

```
Debug Log(10)    ; Affiche '2.3025850929940459'
Debug Log(1)     ; Affiche '0.0'
Debug Log(0)     ; Affiche '-Infinity'
Debug Log(-0)    ; Affiche '-Infinity'
Debug Log(-10)   ; Affiche 'NaN'
```

Voir aussi

Exp() , Log10()

OS Supportés

Tous

115.20 Log10

Syntaxe

```
Resultat.f(d) =
    Log10(Nombre.f(d))
```

Description

Renvoie le logarithme base 10 d'un nombre.

Arguments

Nombre.f ou Nombre.d Le nombre à virgule flottante (float ou double)

Valeur de retour

Renvoie le logarithme en base 10.

Exemple

```
Debug Log10(10)    ; Affiche '1.0'
Debug Log10(1)     ; Affiche '0.0'
Debug Log10(0)     ; Affiche '-Infinity'
Debug Log10(-0)    ; Affiche '-Infinity'
Debug Log10(-10)   ; Affiche 'NaN'

;Astuce: Log10 permet de connaître le 'nombre de 0' d'une valeur.
```

```
8 | Debug Log10(1000)    ;
    Affiche '3.0'
9 | Debug Log10(0.001)   ;
    Affiche '-3.0'
```

Voir aussi

Log() , Exp()

OS Supportés

Tous

115.21 Mod

Syntaxe

```
Resultat.f(d) =
    Mod(Dividende.f(d),
    Diviseur.f(d))
```

Description

Renvoie le reste d'une division.

Arguments

Dividende.f ou Dividende.d Le nombre à virgule flottante à diviser, le dividende (float ou double)

Diviseur.f ou Diviseur.d Le nombre diviseur à virgule flottante (float ou double)

Valeur de retour

Renvoie le reste de la division de 'Nombre' par 'Diviseur'.

Remarques

Ceci est la version en virgule flottante de l'opérateur Modulo '%' pour les entiers.

Exemple

```
1 | Debug Mod(10,2)    ; 10/2
    donne 5 et reste 0 donc
    l'affichage est '0.0'
2 | Debug Mod(11,2)    ; Affiche
    '1.0'
3 | Debug Mod(-11,2)   ; Affiche
    '-1.0'
4 | Debug Mod(11,-2)   ; Affiche
    '1.0'
5 | Debug Mod(-11,-2)  ; Affiche
    '-1.0'
```

OS Supportés

Tous

115.22 NaN

Syntaxe

```
Resultat.f(d) = NaN()
```

Description

Renvoie la valeur flottante spéciale représentant 'Pas un Nombre' ('Not a Number').

Arguments

Aucun.

Valeur de retour

Renvoie NaN.

Remarques

Cette valeur est renvoyée lors d'un calcul invalide, comme le calcul de la racine carrée d'un nombre négatif.
NaN est une valeur spéciale. L'utilisation de NaN dans un calcul renverra de nouveau la valeur NaN. La fonction IsNaN() peut être utilisée pour vérifier si une variable a pour valeur NaN.

Exemple

```
1   Debug IsNaN(NaN() * 5 + 2)
      ; affiche 1.0
```

Voir aussi

IsNaN() , Infinity()

OS Supportés

Tous

115.23 Radian

Syntaxe

```
Resultat.f(d) =
    Radian(Angle.f(d))
```

Description

Convertit la valeur d'un angle en degré, en radian.

Arguments

Angle.f ou Angle.d L'angle en degrés.
(float ou double)

Valeur de retour

Renvoie l'angle en radian.

Remarques

Il n'y a aucune normalisation pour s'assurer que l'angle est compris entre 0 et 2xPI
Si l'entrée est supérieure à 360 alors le résultat sera supérieur à 2xPI.
De même, une entrée négative se traduira par une sortie négative.
L'opération réciproque est disponible à l'aide de la commande Degree() .

Exemple

```
1   Debug Radian(90) ; Affiche
      #PI/2
```

Voir aussi

Degree()

OS Supportés

Tous

115.24 Random

Syntaxe

```
Resultat = Random(Maximum [,
    Minimum])
```

Description

Renvoie un nombre aléatoire.

Arguments

Maximum La valeur maximale.
Doit être une valeur positive ou nulle et ne peut excéder le maximum positif de integer .

Minimum (optionnel) La valeur minimale.
Doit être une valeur positive ou nulle et ne peut excéder le maximum positif de integer .
Si elle est spécifiée, le résultat se situe entre la valeur minimale et la valeur maximale (les deux valeurs incluses).

Valeur de retour

Renvoie une valeur entre zéro ou une valeur minimale et une valeur maximale (toutes deux incluses).

Remarques

RandomSeed() peut être utilisé pour régénérer la table des nombres aléatoires. Cela peut être utile quand un programmeur veut toujours avoir la même table de nombres aléatoires dans le même ordre.
RandomData() peut être utilisé pour remplir une mémoire tampon avec des nombres aléatoires.
RandomizeArray() ou RandomizeList() peut être utilisé pour rendre aléatoire les éléments d'un tableau ou d'une liste.
Note : Cette commande utilise un générateur de nombre aléatoire conçu pour être très rapide, au détriment de la robustesse. Il convient parfaitement pour une utilisation générale, mais si les nombres générés sont utilisés à des fins de chiffrement, il est fortement conseillé d'utiliser CryptRandom() .

Exemple

```
Repeat
  DeAJouer = Random(6,1)  ; Renvoie une valeur entre 1 et 6, incluant 1 et 6
  Choix = MessageRequester("Lancer le dé", "Vous avez eu un " + DeAJouer + ", Relancer le dé ?", #PB_MessageRequester_YesNo)
Until Choix = #PB_MessageRequester_No
```

Exemple

```
; La valeur minimum doit être positive mais cette astuce permet d'utiliser un minimum négatif
Procedure Random2(Min, Max)
    ProcedureReturn Random(Max - Min) + Min
EndProcedure

For i = 1 To 10
    Debug Random2(-10, 5)
Next i
```

Voir aussi

RandomSeed() , RandomData() , CryptRandom() , RandomizeArray() , RandomizeList()

OS Supportés

Tous

115.25 RandomData

Syntaxe

```
RandomData(*Memoire, Longueur)
```

Description

Remplit une mémoire avec des données aléatoires.

Arguments

***Memoire** L'adresse de la mémoire tampon à remplir.

Longueur La taille de la mémoire tampon à remplir.

Valeur de retour

Aucune.

Remarques

Cette commande utilise la même racine que la commande Random() .
RandomSeed() peut être utilisé pour régénérer la table des nombres aléatoires.
Note : Cette commande utilise un générateur de nombre aléatoire conçu pour être très rapide, au détriment de la robustesse. Il convient parfaitement pour une utilisation générale, mais si les nombres générés sont utilisés à des fins de chiffrement, il est fortement conseillé d'utiliser CryptRandomData() .

Exemple

```
1  ; Crée une image avec un contenu aléatoire
2  ;
3  CreateImage(0, 200, 200)
4  If StartDrawing(ImageOutput(0))
5    *Buffer = DrawingBuffer()
6    Pitch   = DrawingBufferPitch()
```

```
7
8       RandomData(*Buffer,
        Pitch*200)
9
10      StopDrawing()
11    EndIf
12
13    OpenWindow(0, 0, 0, 200,
       200, "Image aléatoire ",
       #PB_Window_SystemMenu |
       #PB_Window_ScreenCentered)
14    ImageGadget(0, 0, 0, 200,
       200, ImageID(0))
15
16    Repeat
17    Until WaitWindowEvent() =
       #PB_Event_CloseWindow
```

Voir aussi

RandomSeed() , Random() , CryptRandomData()

OS Supportés

Tous

115.26 RandomSeed

Syntaxe

```
RandomSeed(Valeur)
```

Description

Change la table des nombres aléatoires renvoyée par Random() ou RandomData() .

Arguments

Valeur La nouvelle table pour le générateur de nombres aléatoires.

Valeur de retour

Aucune.

Remarques

A chaque fois qu'un programme PureBasic se lance, sa table est régénérée, ce qui assure d'avoir des nombres aléatoires totalement différents à chaque exécution. Ainsi RandomSeed() n'est utile que lorsque l'objectif est de générer des nombres aléatoires dans le même ordre à chaque fois que le programme est exécuté.

Exemple

```
1  RandomSeed(123456789)
2  Debug Random(1000)    ;
     Affiche 934
3  Debug Random(10000)   ;
     Affiche 7716
4  Debug Random(100000)  ;
     Affiche 57368
```

Voir aussi

Random() , RandomData() , RandomizeArray() , RandomizeList()

OS Supportés

Tous

115.27 Round

Syntaxe

```
Resultat.f(d) =
    Round(Nombre.f(d), Mode)
```

Description

Arrondit un nombre flottant selon le mode choisi.

Arguments

Nombre.f ou Nombre.d Le nombre à virgule flottante (float ou double)

Mode Peut prendre l'une des valeurs suivantes :

```
#PB_Round_Down    : Le
 nombre est arrondi par
 défaut.
#PB_Round_Up      : Le
 nombre est arrondi par
 excès.
#PB_Round_Nearest: Le
 nombre est arrondi à
 l'entier le plus proche
 (0.5 est arrondi à 1)
```

Valeur de retour

Le nombre arrondi.
C'est un nombre à virgule flottante (float ou double)

Remarques

Pour transformer un nombre à virgule flottante en un entier sans arrondi, utiliser Int() ou IntQ() .

Exemple

```
1   Debug Round(11.6,
      #PB_Round_Down)     ;
      Affiche '11.0'
2   Debug Round(-3.6,
      #PB_Round_Down)     ;
      Affiche '-4.0'
3
4   Debug Round(11.6,
      #PB_Round_Up)       ;
      Affiche '12.0'
5   Debug Round(-3.6,
      #PB_Round_Up)       ;
      Affiche '-3.0'
6
7   Debug Round(11.6,
      #PB_Round_Nearest)  ;
      Affiche '12.0'
8   Debug Round(11.4,
      #PB_Round_Nearest)  ;
      Affiche '11.0'
9   Debug Round(11.5,
      #PB_Round_Nearest)  ;
      Affiche '12.0'
10  Debug Round(-3.5,
      #PB_Round_Nearest)  ;
      Affiche '-4.0'
```

Voir aussi

Int() , IntQ()

OS Supportés

Tous

115.28 Sign

Syntaxe

```
Resultat = Sign(Nombre.f(d))
```

Description

Renvoie le signe d'un nombre sous la forme d'un nombre entier (Integer).

Arguments

Nombre.f ou Nombre.d Le nombre à virgule flottante (float ou double)

Valeur de retour

Renvoie une valeur Integer représentant le signe du 'Nombre' spécifié :

```
- Renvoie  0 si le 'Nombre'
  est 0.
- Renvoie  1 si le 'Nombre'
  est positif.
- Renvoie -1 si le 'Nombre'
  est négatif.
```

Exemple

```
1  Debug Sign(-7)    ; Affiche
     -1
2  Debug Sign(0)     ; Affiche
     0
3  Debug Sign(7)     ; Affiche
     1
```

Voir aussi

Abs()

OS Supportés

Tous

115.29 Sin

Syntaxe

```
Resultat.f(d) =
    Sin(Angle.f(d))
```

Description

Renvoie le sinus d'un angle.

Arguments

Angle.f ou Angle.d L'angle en radians. (float ou double)

Valeur de retour

Renvoie le sinus de l'angle spécifié, exprimé en radian.
Toujours compris entre -1.0 et 1.0.

Remarques

La fonction réciproque est ASin() .

Exemple

```
1  Debug Sin(1.5708) ; Affiche
     approximativement '1.0'
     (0.99999999999325373)
```

Voir aussi

ASin() , SinH() , Radian()

OS Supportés

Tous

115.30 SinH

Syntaxe

```
Resultat.f(d) =
    SinH(Angle.f(d))
```

Description

Renvoie le sinus hyperbolique d'un angle.

Arguments

Angle.f ou Angle.d L'angle en radians. (float ou double)

Valeur de retour

Renvoie le sinus hyperbolique de l'angle spécifié, exprimé en radian.

Remarques

SinH() est la fonction réciproque de ASinH() .

Exemple

```
1  Debug SinH(Log(1.618033)) ;
     Affiche approximativement
     0.5 (0.49999931679051052)
```

Voir aussi

ASinH() , Sin()

OS Supportés

Tous

115.31 Sqr

Syntaxe

```
Resultat.f(d) =
    Sqr(Nombre.f(d))
```

Description

Renvoie la racine carrée d'un nombre.

Arguments

Nombre.f ou Nombre.d Le nombre à virgule flottante (float ou double)

Valeur de retour

Renvoie la racine carrée du nombre spécifié.

Exemple

```
1  Debug Sqr(10)     ; Affiche
     3.1622776601683795
2  Debug Sqr(10.01)  ; Affiche
     3.1638584039112749
3  Debug Sqr(-10)    ; Affiche
     NaN
```

Voir aussi

Pow()

OS Supportés

Tous

115.32 Tan

Syntaxe

```
Resultat.f(d) =
    Tan(Angle.f(d))
```

Description

Renvoie la tangente d'un angle.

Arguments

Angle.f ou Angle.d L'angle en radians. (float ou double)

Valeur de retour

Renvoie la tangente de l'angle spécifié, exprimé en radian.

Remarques

La fonction réciproque est ATan() .

Exemple

```
1  Debug Tan(0.785398) ;
     Affiche '1.0'
     (0.99999967320515659)
2  Debug Tan(0.0)      ;
     Affiche '0.0'
```

Voir aussi

ATan() , ATan2() , TanH() , Radian()

OS Supportés

Tous

115.33 TanH

Syntaxe

```
Resultat.f(d) =
    TanH(Angle.f(d))
```

Description

Renvoie la tangente hyperbolique d'un angle.

Arguments

Angle.f ou Angle.d L'angle en radians. (float ou double)

Valeur de retour

Renvoie la tangente hyperbolique de l'angle spécifié, exprimé en radian.

Remarques

TanH() est la fonction réciproque de ATanH() .

Exemple

```
1  Debug TanH(Log(1.618033)) ;
     Affiche '0.447213' (1/5 *
     Sqr(5))
```

Voir aussi

ATanH() , Tan()

OS Supportés

Tous

Chapitre 116

Memory

Généralités

Il est parfois très utile d'avoir un accès direct à la mémoire vive (RAM) de l'ordinateur pour exécuter et accélérer des routines gourmandes en temps d'exécution. Cette bibliothèque vous permet d'allouer un nombre quelconque de zones mémoire et de les utiliser directement avec PureBasic.
Note : La manipulation incorrecte de zones mémoire peut conduire au plantage de l'ordinateur.

OS Supportés

Tous

116.1 AllocateMemory

Syntaxe

```
*Resultat =
    AllocateMemory(Taille [,
    Options])
```

Description

Alloue une zone mémoire contiguë remplie de zéros (caractère #Null).

Arguments

Taille La taille en octets de la nouvelle zone mémoire.

Options (optionnel) Peut être une des valeurs suivantes :

```
#PB_Memory_NoClear: Ne pas
 remplir la zone de
 mémoire avec des zéros.
 Cela peut aider à
 l'avoir plus rapidement
 si la mémoire allouée
 est utilisée
 immédiatement.
```

Valeur de retour

Renvoie l'adresse de la mémoire allouée (pointeur), zéro sinon.

Remarques

La commande FreeMemory() peut être utilisée pour renvoyer la mémoire allouée au système.
La commande ReAllocateMemory() peut être utilisée pour changer la taille de la zone allouée.
Toutes les zones de mémoire allouées sont automatiquement libérées lorsque les programmes se terminent.
Note : Si le programme se bloque à cette commande, c'est généralement dû à une corruption de la mémoire plus tôt dans le programme, après écriture, dans une zone en dehors de la zone de mémoire allouée. La cause de ce genre d'erreur peut être trouvée en utilisant le purificateur

Exemple

```
1  *MemoireID =
    AllocateMemory(5000)
2  If *MemoireID
3    Debug "Adresse de début
    de la zone mémoire de 5000
    octets :"
4    Debug *MemoireID
5    PokeS(*MemoireID, "Stocke
    cette chaîne dans la zone
    mémoire")
6    FreeMemory(*MemoireID)  ;
    sera également effectué
    automatiquement à la fin
    du programme
7  Else
8    Debug "Impossible
    d'allouer la mémoire
    demandée !"
9  EndIf
```

Exemple

```
1  Texte.s = "Salut !!!"
2
3  *mem =
    AllocateMemory(StringByteLength(Texte)
    + SizeOf(CHARACTER))
4  CopyMemory(@Texte, *mem,
    StringByteLength(Texte) +
    SizeOf(CHARACTER))
5  Debug MemorySize(*mem)
6  Debug PeekS(*mem)
```

```
7   FreeMemory(*mem)
```

Voir aussi

ReAllocateMemory() , FreeMemory() , MemorySize()

OS Supportés

Tous

116.2 AllocateStructure

Syntaxe

```
*Resultat.Structure =
    AllocateStructure(Structure)
```

Description

Alloue un nouvel objet de type structure dynamique.
Cette structure dynamique est correctement initialisée et prête à être utilisée, sans la nécessité d'appeler InitializeStructure() .
Pour accéder aux données de la structure, le pointeur associé à la 'Structure' spécifiée doit être utilisé.

Arguments

Structure Le nom de la structure utilisée pour créer le nouvel objet dynamique. La structure doit exister au préalable.

Valeur de retour

L'adresse (pointeur), de la nouvelle structure dynamique, zéro sinon.

Remarques

Cette commande est à la disposition des utilisateurs avancés et ne devrait pas être nécessaire dans la plupart des programmes. Il est souvent plus judicieux d'utiliser un tableau structuré, une liste ou une map pour stocker dynamiquement des objets structurés.
Utiliser FreeStructure() pour libérer la mémoire d'une structure dynamique.
Toutes les structures dynamiques sont automatiquement libérées lorsque le programme se termine.
Si le programme se bloque avec cette commande, c'est généralement le résultat d'une corruption de la mémoire plus tôt

dans le programme, en écrivant dans une zone à l'extérieur de la zone mémoire allouée. Il est possible de remonter à la cause de cette erreur en utilisant le purificateur .

Exemple

```
1   Structure Peuple
2     Nom$
3     List Amis$()
4   EndStructure
5
6   *DynamicPeuple.Peuple =
     AllocateStructure(Peuple)
7   *DynamicPeuple\Nom$ = "Fred"
8   AddElement(*DynamicPeuple\Amis$())
9   *DynamicPeuple\Amis$() =
     "Stef"
10
11  Debug *DynamicPeuple\Nom$
12  Debug *DynamicPeuple\Amis$()
13
14  FreeStructure(*DynamicPeuple)
```

Voir aussi

FreeStructure()

OS Supportés

Tous

116.3 CompareMemory

Syntaxe

```
Resultat =
    CompareMemory(*Memoire1,
    *Memoire2, Taille)
```

Description

Compare deux zones mémoire.

Arguments

***Memoire1, *Memoire2** Les adresses des deux mémoires à comparer.

Taille Le nombre d'octets à comparer.

Valeur de retour

Renvoie une valeur non nulle si les deux mémoires sont identiques, zéro sinon.

Exemple

```
*Memoire1 =
 AllocateMemory(5000)
PokeB(*Memoire1, 65)      ;
 Stocke cette valeur dans
 la zone mémoire)
*Memoire2 =
 AllocateMemory(50)
PokeB(*Memoire2, 65)
*Memoire3 =
 AllocateMemory(100)
PokeB(*Memoire3, 90)
  Debug
 CompareMemory(*Memoire1,
 *Memoire2, 1) ; Affiche 1
  Debug
 CompareMemory(*Memoire1,
 *Memoire3, 1) ; Affiche 0
FreeMemory(*Memoire1)
FreeMemory(*Memoire2)
FreeMemory(*Memoire3)
```

Voir aussi

AllocateMemory() ,
CompareMemoryString() , MemorySize()

OS Supportés

Tous

116.4 CompareMemoryString

Syntaxe

```
Resultat =
   CompareMemoryString(*Texte1,
   *Texte2 [, Mode [,
   Longueur [, Options]]])
```

Description

Compare deux chaînes de caractères situées en mémoire.

Arguments

***Texte1, *Texte2** Les adresses des deux chaînes de caractères à comparer.

Mode (optionnel) Peut prendre l'une des valeurs suivantes :

```
#PB_String_CaseSensitive :
 La recherche est
 sensible à  la casse
 (a=a) (Par défaut).
```

```
#PB_String_NoCase        :
 Comparaison insensible à
 la casse (a=A).
```

Longueur (optionnel) Le nombre de caractères à comparer.
Si ce paramètre ne est pas spécifié ou égal à -1 alors les chaînes sont comparées jusqu'à ce qu'un caractère #Null soit atteint.
Si les chaînes ne sont pas terminées par un caractère #Null, ce paramètre doit alors être spécifié.

Options (optionnel) Peut prendre l'une des valeurs suivantes :

```
#PB_Ascii   : Compare la
 chaîne de caractères au
 format ASCII
#PB_UTF8    : Compare la
 chaîne de caractères au
 format  UTF-8
#PB_Unicode: Compare la
 chaîne de caractères au
 format  Unicode
```

La valeur par défaut est #PB_Unicode.

Valeur de retour

Renvoie l'une des valeurs suivantes :

```
#PB_String_Equal   : Le
 'Texte1' est égal au
 'Texte2'.
#PB_String_Lower   : Le
 'Texte1' est inférieur au
 'Texte2'.
#PB_String_Greater: Le
 'Texte1' est supérieur au
 'Texte2'.
```

Exemple

```
1  Texte1$ = "OK"
2  Texte2$ = "KO"
3
4  Comp =
    CompareMemoryString(@Texte1$,
    @Texte2$)
5  Select Comp
6    Case #PB_String_Equal
7      Debug "Le 'Texte1' est
    égal au 'Texte2'."
8    Case #PB_String_Lower
9      Debug "Le 'Texte1' est
    inférieur au 'Texte2'."
10   Case #PB_String_Greater
11     Debug "Le 'Texte1' est
    supérieur au 'Texte2'."
```

```
12  EndSelect
```

Voir aussi

PokeS() , PeekS() , MemoryStringLength() , CopyMemoryString() , CompareMemory() , MemorySize()

OS Supportés

Tous

116.5 CopyMemory

Syntaxe

```
CopyMemory(*MemoireSource,
    *MemoireDestination,
    Taille)
```

Description

Copie une zone mémoire dans une autre.

Arguments

***MemoireSource** L'adresse de la zone mémoire à copier.

***MemoireDestination** L'adresse de la zone mémoire de destination.

Taille Le nombre d'octets à copier.

Valeur de retour

Aucune.

Remarques

Les zones mémoires source et destination ne devraient pas se chevaucher, dans ce cas, il est préférable d'utiliser MoveMemory() .

Exemple

```
1  Texte1$ = "OK"
2  Texte2$ = "KO"
3  Debug Texte1$ ; Affiche OK
4  Debug Texte2$ ; Affiche KO
5
6  CopyMemory(@Texte1$,
    @Texte2$, 4) ; en unicode,
    un caractère est codé sur
    2 octets
7
8  Debug Texte1$ ; Affiche OK
9  Debug Texte2$ ; Affiche OK
```

Voir aussi

MoveMemory() , CopyMemoryString() , AllocateMemory() , MemorySize()

OS Supportés

Tous

116.6 CopyMemoryString

Syntaxe

```
Resultat =
    CopyMemoryString(*Memoire
    [, @*MemoireDestination])
```

Description

Copie un texte d'une zone mémoire dans une autre, si elle est spécifiée, à la fin du tampon mémoire courant sinon.

Arguments

***Memoire** L'adresse de la chaîne de caractères à copier.
La chaîne doit se terminer par un caractère #Null.
La chaîne doit être dans le format de chaîne de PureBasic.

@*MemoireDestination (optionnel) Le pointeur sur une variable contenant l'adresse de la mémoire tampon de destination.
Après la copie, la variable *MemoireDestination pointera vers le caractère #Null à la fin de la chaîne copiée, donc un nouvel appel à cette fonction ajoutera la nouvelle chaîne à la précédente.
Si ce paramètre est omis, l'adresse de l'appel précédent est utilisé.

Valeur de retour

Renvoie *MemoireDestination.

Exemple

```
1  *Tampon =
     AllocateMemory(1000)
2  *Pointeur = *Tampon
3  CopyMemoryString("Salut",
     @*Pointeur)
4  CopyMemoryString(" le
     Monde")  ; Cette chaîne
     sera copiée juste après
     "Salut"
```

```
5  *Pointeur-2*SizeOf(CHARACTER)
     ; revient en arrière de 2
     caractères (sur le 'd' de
     'Monde')
6  CopyMemoryString("DE")
             ; les deux
     dernières lettres seront
     en majuscules
7  Debug PeekS(*Tampon)
```

Voir aussi

CopyMemory() , PeekS() , PokeS()

OS Supportés

Tous

116.7 FillMemory

Syntaxe

```
FillMemory(*Memoire, Taille
    [, Valeur [, Type]])
```

Description

Remplit une zone mémoire avec une valeur donnée.

Arguments

***Memoire** L'adresse de la zone mémoire à remplir.

Taille La taille en octets de la zone mémoire à remplir.

Valeur (optionnel) La valeur à écrire dans la zone de mémoire.
La valeur par défaut est la valeur #NULL (0).

Type (optionnel) Peut être une des constantes suivantes :

```
  #PB_Byte      : Type byte
  (1 octet signé) (défaut).
  #PB_Ascii     : Type ascii
 (1 octet non signé).
  #PB_Word      : Type word
  (2 octets signés).
  #PB_Unicode   : Type word
  (2 octets non signés).
  #PB_Character: Type
    caractère
 (2 octets non signés en
    unicode
).
  #PB_Long      : Type long
```

```
    (4 octets).
  #PB_Integer  : Type integer
   (4 octets dans un
   exécutable 32 bits, 8
   octets dans un
   exécutable 64 bits).
```

Valeur de retour

Aucune.

Exemple

```
1  *Tampon =
    AllocateMemory(500)
2
3  FillMemory(*Tampon, 500) ;
    Remplit 500 octets avec
    des zéros (vide la zone
    mémoire)
4  FillMemory(*Tampon, 500,
    $FF) ; Remplit 500 octets
    avec la valeur $FF
5  FillMemory(*Tampon, 500,
    $BADF00D, #PB_Long) ;
    Remplit 500 octets avec la
    valeur $BADF00D
```

Voir aussi

AllocateMemory() , MemorySize()

OS Supportés

Tous

116.8 FreeMemory

Syntaxe

```
FreeMemory(*Memoire)
```

Description

Libère une zone mémoire.

Arguments

***Memoire** L'adresse de la zone de mémoire à libérer.
Ce doit être la valeur renvoyée par AllocateMemory() ou ReAllocateMemory() .

Valeur de retour

Aucune.

Remarques

Si le programme se bloque à cette commande, même si l'entrée semble correcte, c'est généralement le résultat d'une corruption de mémoire à un moment plus tôt dans le programme en écrivant dans une zone à l'extérieur de la zone de mémoire allouée. Une telle erreur peut être analysée en utilisant le purificateur
Toutes les zones mémoire restantes sont automatiquement libérées quand le programme se termine.

Exemple

```
1  *MemoireID =
     AllocateMemory(5000)
2  If *MemoireID
3    Debug "Adresse de début
     de la zone mémoire de 5000
     octets :"
4    Debug *MemoireID
5    PokeS(*MemoireID, "Stocke
     cette chaîne dans la zone
     mémoire")
6    FreeMemory(*MemoireID)  ;
     sera également effectué
     automatiquement à la fin
     du programme
7  Else
8    Debug "Impossible
     d'allouer la mémoire
     demandée !"
9  EndIf
```

Voir aussi

AllocateMemory() , ReAllocateMemory()

OS Supportés

Tous

116.9 FreeStructure

Syntaxe

```
FreeStructure(*Structure)
```

Description

Libère la structure dynamique précédemment allouée avec AllocateStructure() .

Arguments

***Structure** L'adresse de la structure dynamique à libérer.
Ce doit être la valeur renvoyée par AllocateStructure() .

Valeur de retour

Aucune.

Remarques

Il n'est pas nécessaire d'appeler ClearStructure() avant de libérer la structure.
Si le programme se bloque avec cette commande, c'est généralement le résultat d'une corruption de la mémoire plus tôt dans le programme, en écrivant dans une zone à l'extérieur de la zone mémoire allouée. Il est possible de remonter à la cause de cette erreur en utilisant le purificateur .
Toutes les structures dynamiques sont automatiquement libérées à la fermeture du programme.

Exemple

```
1   Structure Peuple
2     Nom$
3     List Amis$()
4   EndStructure
5
6   *DynamicPeuple.Peuple =
     AllocateStructure(Peuple)
7   *DynamicPeuple\Nom$ = "Fred"
8   AddElement(*DynamicPeuple\Amis$())
9   *DynamicPeuple\Amis$() =
     "Stef"
10
11  Debug *DynamicPeuple\Nom$
12  Debug *DynamicPeuple\Amis$()
13
14  FreeStructure(*DynamicPeuple)
```

Voir aussi

AllocateStructure()

OS Supportés

Tous

116.10 MemorySize

Syntaxe

```
Resultat =
    MemorySize(*Memoire)
```

Description

Renvoie la taille d'une zone mémoire.

Arguments

***Memoire** L'adresse de la zone mémoire à utiliser.
Ce doit être la valeur renvoyée par AllocateMemory() ou ReAllocateMemory() .

Valeur de retour

Renvoie la taille de la zone mémoire, en octets.

Exemple

```
1  Texte1$="OK"
2  *Mem=AllocateMemory(100)
3  ;Debug MemorySize(@Texte1$)
     ; Erreur
4  Debug MemorySize(*Mem) ;
     Affiche 100
```

Exemple

```
1  Texte.s = "Salut !!!"
2
3  *mem =
     AllocateMemory(StringByteLength(Texte)
     + SizeOf(CHARACTER))
4  CopyMemory(@Texte, *mem,
     StringByteLength(Texte) +
     SizeOf(CHARACTER))
5  Debug MemorySize(*mem)
6  Debug PeekS(*mem)
7  FreeMemory(*mem)
```

Voir aussi

AllocateMemory() , ReAllocateMemory() , FreeMemory()

OS Supportés

Tous

116.11 MemoryStringLength

Syntaxe

```
Resultat =
    MemoryStringLength(*Memoire
    [, Options])
```

Description

Renvoie la longueur d'une chaîne de caractères en mémoire et terminée par un caractère #Null.

Arguments

***Memoire** L'adresse de la chaîne de caractères.

Options (optionnel) Le format à utiliser. Peut prendre l'une des valeurs suivantes :

```
    #PB_Ascii   : Format ASCII
    #PB_UTF8    : Format UTF-8
    #PB_Unicode: Format
     Unicode (par défaut,
     voir unicode
)
```

Combiné avec l'une des valeurs suivantes :

```
    #PB_ByteLength: Seulement
     avec l'option #PB_UTF8,
     le résultat représentera
     des octets (pas des
     caractères).
                    Cela peut
     être utile car UTF8 a
     des caractères de
     longueur variable.
```

Valeur de retour

Renvoie la longueur de la chaîne de caractères, en caractères excepté le caractère #Null de fin de chaîne.

Exemple

```
1  Texte1$ = "OK"
2  Debug
    MemoryStringLength(@Texte1$)
    ; Affiche 2
```

Voir aussi

PokeS() , PeekS() , AllocateMemory()

OS Supportés

Tous

116.12 MoveMemory

Syntaxe

```
MoveMemory(*MemoireSource,
    *MemoireDestination,
    Taille)
```

Description

Copie une zone mémoire dans une autre. Le chevauchement des deux zones mémoire est autorisé.

Arguments

***MemoireSource** L'adresse de la zone mémoire à copier.

***MemoireDestination** L'adresse de la zone mémoire de destination.

Taille Le nombre d'octets à copier.

Valeur de retour

Aucune.

Remarques

Contrairement à CopyMemory() , les zones mémoires source et destination peuvent se chevaucher. Néanmoins, MoveMemory() est plus lent que CopyMemory() , donc il est préférable de l'utiliser que si c'est justifié.

Exemple

```
1  Texte1$ = "OK"
2  Texte2$ = "KOKO"
3  Debug Texte1$ ; Affiche OK
4  Debug Texte2$ ; Affiche KOKO
5
6  MoveMemory(@Texte1$,
     @Texte2$ + 4, 4) ; en
     unicode, un caractère est
     codé sur 2 octets
7
8  Debug Texte1$ ; Affiche OK
9  Debug Texte2$ ; Affiche KOOK
```

Voir aussi

CopyMemory() , AllocateMemory() , MemorySize()

OS Supportés

Tous

116.13 ReAllocateMemory

Syntaxe

```
*Resultat =
    ReAllocateMemory(*Memoire,
    Taille [, Options])
```

Description

Redimensionne une zone mémoire.

Arguments

***Memoire** L'adresse de la zone mémoire à redimensionner.
Cette valeur doit être le résultat d'un appel à AllocateMemory() ou à ReAllocateMemory() .
Si ce paramètre est #Null, la commande agit comme AllocateMemory() et alloue une zone mémoire mais avec la nouvelle dimension.

Taille La nouvelle taille en octets.

Options (optionnel)

```
      #PB_Memory_NoClear: Ne
    remplit pas la zone
    mémoire étendue avec des
    zéros. Cela peut aider à
    l'avoir plus rapidement
    si la mémoire
                        étendue
    est utilisée
    immédiatement. Si la
    mémoire est rétrécie,
    cette option n'a aucun
    effet.
```

Valeur de retour

Renvoie la nouvelle adresse de la zone mémoire si elle peut être redimensionnée. Dans ce cas, l'ancienne adresse '*MemoireID' ne peut plus être utilisée. Si le redimensionnement a échoué (car il n'y a pas assez de mémoire disponible), le résultat est égal à zéro et le '*Memoire' est toujours valide avec la zone de mémoire existante et l'ancienne taille.

Remarques

Si la taille de la zone de mémoire est augmentée, les nouveaux octets sont tout

d'abord remplis de zéros à moins que #PB_Memory_NoClear ne soit indiqué. Si le programme se bloque à cette commande, même si l'entrée semble correcte, c'est généralement le résultat d'une altération de la mémoire qui s'est produite antérieurement dans le programme par l'écriture dans une zone à l'extérieur de la zone de mémoire allouée. Il est possible de trouver la cause de ce genre d'erreur en utilisant le purificateur
Tous les blocs de mémoire alloués restants sont automatiquement libérés lorsque le programme se termine.

Exemple

```
*MemoireID = AllocateMemory(1000)
PokeS(*MemoireID, "Stocke cette chaîne")
; faire quelque chose en plus avec ça ici...
;
*NouvelleMemoireID = ReAllocateMemory(*MemoireID, 2000) ; besoin de plus de mémoire
If *NouvelleMemoireID
  ; maintenant travailler avec *NouvelleMemoireID avec une taille de 2000 octets
  Debug "L'ancien contenu est toujours là :"
  Debug PeekS(*NouvelleMemoireID)
  ;
  FreeMemory(*NouvelleMemoireID)
Else
  ; le redimensionnement a échoué, continuons de travailler avec *MemoireID (de taille 1000 octets)
  ;
  FreeMemory(*MemoireID)
EndIf
```

Voir aussi

AllocateMemory() , FreeMemory() , MemorySize()

OS Supportés

Tous

116.14 PeekA

Syntaxe

```
Resultat.a = PeekA(*Memoire)
```

Description

Lit un caractère ascii en mémoire.

Arguments

***Memoire** L'adresse mémoire à lire.

Valeur de retour

Renvoie la valeur du caractère ascii .
Vous trouverez une table ASCII ici .

Remarques

Représente 1 octet non signé de 0 à + 255.

Exemple

```
*MemoireID =
 AllocateMemory(5000)
If *MemoireID
  PokeA(*MemoireID, 65)    ;
 Stocke cette valeur dans
 la zone mémoire")
  Debug PeekA(*MemoireID) ;
 Lit cette valeur depuis la
 zone mémoire")
  Debug
 Chr(PeekA(*MemoireID))
  FreeMemory(*MemoireID)
Else
  Debug "Impossible
 d'allouer la mémoire
 demandée !"
EndIf
```

Voir aussi

PokeA()

OS Supportés

Tous

116.15 PeekB

Syntaxe

```
Resultat.b = PeekB(*Memoire)
```

Description

Lit un octet (byte) en mémoire.

Arguments

***Memoire** L'adresse mémoire à lire.

Valeur de retour

Renvoie la valeur de l' octet .

Remarques

Reprsente 1 octet signé de -128 à +127.

Exemple

```
*MemoireID = AllocateMemory(5000)
If *MemoireID
  PokeB(*MemoireID, 65) ; Stocke cette valeur dans la zone mémoire")
  Debug PeekB(*MemoireID) ; Lit cette valeur depuis la zone mémoire")
  PokeB(*MemoireID+1, -65)
  Debug PeekB(*MemoireID + 1)
  Debug ~65+1             ; Complément à 2
  FreeMemory(*MemoireID)
Else
  Debug "Impossible d'allouer la mémoire demandée !"
EndIf
```

Voir aussi

PokeB()

OS Supportés

Tous

116.16 PeekC

Syntaxe

```
Resultat.c = PeekC(*Memoire)
```

Description

Lit un caractère en mémoire.

Arguments

***Memoire** L'adresse mémoire à lire.

Valeur de retour

Renvoie la valeur du caractère .

Remarques

Représente 2 octets non signés de 0 à +65 535 en mode unicode.

Exemple

```
*MemoireID = AllocateMemory(5000)
If *MemoireID
  PokeC(*MemoireID, 65)    ; Stocke cette valeur dans la zone mémoire")
  Debug PeekC(*MemoireID) ; Lit cette valeur depuis la zone mémoire")
  Debug Chr(PeekC(*MemoireID))
  FreeMemory(*MemoireID)
Else
  Debug "Impossible d'allouer la mémoire demandée !"
EndIf
```

Voir aussi

PokeC()

OS Supportés

Tous

116.17 PeekD

Syntaxe

```
Resultat.d = PeekD(*Memoire)
```

Description

Lit un double en mémoire.

Arguments

***Memoire** L'adresse mémoire à lire.

Valeur de retour

Renvoie la valeur du double .

Remarques

Représente 8 octets représentant un nombre à virgule flottante en double précision.

Exemple

```
1  *MemoireID =
     AllocateMemory(5000)
2  If *MemoireID
3    PokeD(*MemoireID,
     123456789.123456789) ;
     Stocke cette valeur dans
     la zone mémoire")
4    Debug PeekD(*MemoireID)
                    ; Lit cette
     valeur depuis la zone
     mémoire")
5    FreeMemory(*MemoireID)
6  Else
7    Debug "Impossible
     d'allouer la mémoire
     demandée !"
8  EndIf
```

Voir aussi

PokeD()

OS Supportés

Tous

116.18 PeekI

Syntaxe

```
Resultat.i = PeekI(*Memoire)
```

Description

Lit un entier 'integer' en mémoire.

Arguments

***Memoire** L'adresse mémoire à lire.

Valeur de retour

Renvoie la valeur de l' integer .

Remarques

Représente 4 octets signés de -2 147 483 648 à + 2 147 483 647 sur un exécutable 32 bits, 8 octets signés de -9 223 372 036 854 775 808 à + 9 223 372 036 854 775 807 sur un exécutable 64 bits.

Exemple

```
*MemoireID =
  AllocateMemory(5000)
If *MemoireID
  PokeI(*MemoireID,
  123456789.123456789)    ;
  Stocke cette valeur dans
  la zone mémoire")
  Debug PeekI(*MemoireID)
                  ; Lit
  cette valeur depuis la
  zone mémoire")
  FreeMemory(*MemoireID)
Else
  Debug "Impossible
  d'allouer la mémoire
  demandée !"
EndIf
```

Voir aussi

PokeI()

OS Supportés

Tous

116.19 PeekL

Syntaxe

```
Resultat.l = PeekL(*Memoire)
```

Description

Lit un long en mémoire.

Arguments

***Memoire** L'adresse mémoire à lire.

Valeur de retour

Renvoie la valeur de l' octet .

Remarques

Représente 4 octets signé de -2 147 483 648 à + 2 147 483 647.

Exemple

```
*MemoireID =
  AllocateMemory(5000)
If *MemoireID
```

```
3     PokeL(*MemoireID,
      123456789) ; Stocke cette
      valeur dans la zone
      mémoire")
4     Debug PeekL(*MemoireID)
          ; Lit cette valeur
      depuis la zone mémoire")
5     FreeMemory(*MemoireID)
6   Else
7     Debug "Impossible
      d'allouer la mémoire
      demandée !"
8   EndIf
```

Voir aussi

PokeL()

OS Supportés

Tous

116.20 PeekW

Syntaxe

```
Resultat.w = PeekW(*Memoire)
```

Description

Lit un mot (word) en mémoire.

Arguments

***Memoire** L'adresse mémoire à lire.

Valeur de retour

Renvoie la valeur du mot .

Remarques

Représente 2 octets signés de -32 768 à +32 767.

Exemple

```
1   *MemoireID =
      AllocateMemory(5000)
2   If *MemoireID
3     PokeW(*MemoireID, 32000)
          ; Stocke cette valeur
      dans la zone mémoire")
4     Debug PeekW(*MemoireID)
          ; Lit cette valeur
      depuis la zone mémoire")
```

```
    PokeW(*MemoireID+2,
    -32000)
    Debug PeekW(*MemoireID+2)
    Debug ~32000+1
                  ; Complément
    à deux
    FreeMemory(*MemoireID)
  Else
    Debug "Impossible
    d'allouer la mémoire
    demandée !"
  EndIf
```

Voir aussi

PokeW()

OS Supportés

Tous

116.21 PeekF

Syntaxe

```
Resultat.f = PeekF(*Memoire)
```

Description

Lit un float en mémoire.

Arguments

***Memoire** L'adresse mémoire à lire.

Valeur de retour

Renvoie la valeur du float .

Remarques

Représente 4 octets représentant un nombre à virgule flottante en simple précision.

Exemple

```
*MemoireID =
  AllocateMemory(5000)
If *MemoireID
  PokeF(*MemoireID,
  12345.12345)    ; Stocke
  cette valeur dans la zone
  mémoire")
  Debug PeekF(*MemoireID)
            ; Lit cette
  valeur depuis la zone
  mémoire")
```

```
5     FreeMemory(*MemoireID)
6   Else
7     Debug "Impossible
      d'allouer la mémoire
      demandée !"
8   EndIf
```

Voir aussi

PokeF()

OS Supportés

Tous

116.22 PeekQ

Syntaxe

```
Resultat.q = PeekQ(*Memoire)
```

Description

Lit un quad en mémoire.

Arguments

***Memoire** L'adresse mémoire à lire.

Valeur de retour

Renvoie la valeur du quad .

Remarques

Représente 8 octets signés de -9 223 372 036 854 775 808 à + 9 223 372 036 854 775 807.

Exemple

```
1   *MemoireID =
      AllocateMemory(5000)
2   If *MemoireID
3     PokeQ(*MemoireID,
      1234567890123456789)    ;
      Stocke cette valeur dans
      la zone mémoire")
4     Debug PeekQ(*MemoireID) ;
      Lit cette valeur depuis la
      zone mémoire")
5     FreeMemory(*MemoireID)
6   Else
7     Debug "Impossible
      d'allouer la mémoire
      demandée !"
8   EndIf
```

OS Supportés

Tous

116.23 PeekS

Syntaxe

```
Resultat\$ = PeekS(*Memoire
    [, Longueur [, Format]])
```

Description

Lit une chaîne de caractères en mémoire.

Arguments

***Memoire** L'adresse mémoire à lire.

Valeur de retour

Renvoie la valeur de la chaîne de caractères
.

Longueur (optionnel) Le nombre maximum de caractères à lire.
Si ce paramètre n'est pas spécifié ou s'il est égal à -1, il n'y a pas de maximum.
La chaîne est lue jusqu'a ce que le caractère #NULL soit rencontré ou si la longueur maximale est atteinte.

Format (optionnel) Le format à utiliser.
Peut prendre l'une des valeurs suivantes :

```
#PB_Ascii   : Format ASCII
#PB_UTF8    : Format UTF-8
#PB_Unicode: Format
 Unicode (par défaut)
```

Combiné avec l'une des valeurs suivantes :

```
#PB_ByteLength: Seulement
 avec l'option #PB_UTF8,
 le résultat représentera
 des octets (pas des
 caractères).
                Cela peut
 être utile car UTF8 a
 des caractères de
 longueur variable.
```

Valeur de retour

Renvoie la chaîne lue.

Exemple

```
*MemoireID =
  AllocateMemory(5000)
If *MemoireID
  PokeS(*MemoireID,
  "PureBasic hé")   ; Stocke
  cette valeur dans la zone
  mémoire")
  Debug PeekS(*MemoireID)
              ; Lit cette
  valeur depuis la zone
  mémoire")
  FreeMemory(*MemoireID)
Else
  Debug "Impossible
  d'allouer la mémoire
  demandée !"
EndIf
```

Voir aussi

PokeS() , MemoryStringLength() , CompareMemoryString() , CopyMemoryString()

OS Supportés

Tous

116.24 PeekU

Syntaxe

```
Resultat.u = PeekU(*Memoire)
```

Description

Lit un caractère unicode en mémoire.

Arguments

***Memoire** L'adresse mémoire à lire.

Valeur de retour

Renvoie la valeur du caractère unicode .

Remarques

Représente 2 octets non signés de 0 à + 65 535.

Exemple

```
*MemoireID =
 AllocateMemory(5000)
If *MemoireID
  PokeU(*MemoireID, 165)
  ; Stocke cette valeur dans
  la zone mémoire")
  Debug PeekU(*MemoireID)
  ; Lit cette valeur depuis
  la zone mémoire")
  Debug
  Chr(PeekU(*MemoireID))
  FreeMemory(*MemoireID)
Else
  Debug "Impossible
  d'allouer la mémoire
  demandée !"
EndIf
```

Voir aussi

PokeU()

OS Supportés

Tous

116.25 PokeA

Syntaxe

```
PokeA(*Memoire, Valeur.a)
```

Description

Ecrit un caractère ascii en mémoire.

Arguments

***Memoire** L'adresse mémoire où écrire.

Valeur.a La valeur à écrire.

Valeur de retour

Aucune.

Remarques

Représente 1 octet non signé de 0 à + 255.
Vous trouverez une table ASCII ici .

Exemple

```
*MemoireID =
  AllocateMemory(5000)
If *MemoireID
  PokeA(*MemoireID, 65)    ;
  Stocke cette valeur dans
  la zone mémoire")
  Debug PeekA(*MemoireID) ;
  Lit cette valeur depuis la
  zone mémoire")
  FreeMemory(*MemoireID)
Else
  Debug "Impossible
  d'allouer la mémoire
  demandée !"
EndIf
```

Voir aussi

PeekA()

OS Supportés

Tous

116.26 PokeB

Syntaxe

```
PokeB(*Memoire, Valeur.b)
```

Description

Ecrit un octet (byte) en mémoire.

Arguments

***Memoire** L'adresse mémoire où écrire.
Valeur.b La valeur à écrire.

Valeur de retour

Aucune.

Remarques

Représente 1 octet signé de -128 à +127.

Exemple

```
*MemoireID =
  AllocateMemory(5000)
If *MemoireID
  PokeB(*MemoireID, 65)
  ; Stocke cette valeur dans
  la zone mémoire")
```

```
    Debug PeekB(*MemoireID)
    ; Lit cette valeur depuis
    la zone mémoire")
    PokeB(*MemoireID+1, -65)
    Debug PeekB(*MemoireID+1)
    Debug ~65+1
    ; Complément à 2
    FreeMemory(*MemoireID)
  Else
    Debug "Impossible
    d'allouer la mémoire
    demandée !"
  EndIf
```

Voir aussi

PeekB()

OS Supportés

Tous

116.27 PokeC

Syntaxe

```
PokeC(*Memoire, Valeur.c)
```

Description

Ecrit un caractère en mémoire.

Arguments

***Memoire** L'adresse mémoire où écrire.

Valeur.c La valeur à écrire.

Valeur de retour

Aucune.

Remarques

Représente 2 octets non signés de 0 à +65 535 en mode unicode.

Exemple

```
*MemoireID =
  AllocateMemory(5000)
If *MemoireID
  PokeC(*MemoireID, 65)    ;
  Stocke cette valeur dans
  la zone mémoire")
  Debug PeekC(*MemoireID) ;
  Lit cette valeur depuis la
  zone mémoire")
```

```
    Debug
   Chr(PeekC(*MemoireID))
    FreeMemory(*MemoireID)
  Else
    Debug "Impossible
   d'allouer la mémoire
   demandée !"
  EndIf
```

Remarques

Représente 8 octets représentant un nombre à virgule flottante en double précision.

Voir aussi

PeekC()

OS Supportés

Tous

116.28 PokeD

Syntaxe

```
PokeD(*Memoire, Valeur.d)
```

Description

Ecrit un double en mémoire.

Arguments

***Memoire** L'adresse mémoire où écrire.

Valeurd. La valeur à écrire.

Valeur de retour

Aucune.

Remarques

Représente 8 octets représentant un nombre à virgule flottante en double précision.

Exemple

```
*MemoireID =
 AllocateMemory(5000)
If *MemoireID
  PokeD(*MemoireID,
 123456789.123456789)     ;
 Stocke cette valeur dans
 la zone mémoire")
```

```
4     Debug PeekD(*MemoireID)
                  ; Lit cette
    valeur depuis la zone
    mémoire")
5     FreeMemory(*MemoireID)
6   Else
7     Debug "Impossible
    d'allouer la mémoire
    demandée !"
8   EndIf
```

Voir aussi

PeekD()

OS Supportés

Tous

116.29 PokeI

Syntaxe

```
PokeI(*Memoire, Valeur.i)
```

Description

Ecrit un entier (integer) en mémoire.

Arguments

***Memoire** L'adresse mémoire où écrire.

Valeur.i La valeur à écrire.

Valeur de retour

Aucune.

Remarques

Représente 4 octets signés de -2 147 483 648 à + 2 147 483 647 sur un exécutable 32 bits, 8 octets signés de -9 223 372 036 854 775 808 à + 9 223 372 036 854 775 807 sur un exécutable 64 bits.

Exemple

```
1   *MemoireID =
      AllocateMemory(5000)
2   If *MemoireID
3     PokeI(*MemoireID,
    123456789.123456789)    ;
    Stocke cette valeur dans
    la zone mémoire")
```

```
4     Debug PeekI(*MemoireID)
                  ; Lit
      cette valeur depuis la
      zone mémoire")
5     FreeMemory(*MemoireID)
6   Else
7     Debug "Impossible
      d'allouer la mémoire
      demandée !"
8   EndIf
```

Voir aussi

PeekI()

OS Supportés

Tous

116.30 PokeL

Syntaxe

```
PokeL(*Memoire, Valeur.l)
```

Description

Ecrit un long en mémoire.

Arguments

***Memoire** L'adresse mémoire où écrire.

Valeur.l La valeur à écrire.

Valeur de retour

Aucune.

Remarques

Représente 4 octets signé de -2 147 483 648 à + 2 147 483 647.

Exemple

```
1   *MemoireID =
      AllocateMemory(5000)
2   If *MemoireID
3     PokeL(*MemoireID,
      123456789)    ; Stocke
      cette valeur dans la zone
      mémoire")
4     Debug PeekL(*MemoireID) ;
      Lit cette valeur depuis la
      zone mémoire")
5     FreeMemory(*MemoireID)
```

```
6   Else
7     Debug "Impossible
      d'allouer la mémoire
      demandée !"
8   EndIf
```

Voir aussi

PeekL()

OS Supportés

Tous

116.31 PokeQ

Syntaxe

```
PokeQ(*Memoire, Valeur.q)
```

Description

Ecrit un quad en mémoire.

Arguments

***Memoire** L'adresse mémoire où écrire.
Valeur.q La valeur à écrire.

Valeur de retour

Aucune.

Remarques

Représente 8 octets signés de -9 223 372 036 854 775 808 à + 9 223 372 036 854 775 807.

Exemple

```
1   *MemoireID =
      AllocateMemory(5000)
2   If *MemoireID
3     PokeQ(*MemoireID,
      1234567890123456789)    ;
      Stocke cette valeur dans
      la zone mémoire")
4     Debug PeekQ(*MemoireID) ;
      Lit cette valeur depuis la
      zone mémoire")
5     FreeMemory(*MemoireID)
6   Else
7     Debug "Impossible
      d'allouer la mémoire
      demandée !"
8   EndIf
```

Voir aussi

PeekQ()

OS Supportés

Tous

116.32 PokeW

Syntaxe

```
PokeW(*Memoire, Valeur.w)
```

Description

Ecrit un mot (word) en mémoire.

Arguments

***Memoire** L'adresse mémoire où écrire.

Valeur.w La valeur à écrire.

Valeur de retour

Aucune.

Remarques

Représente 2 octets signés de -32 768 à +32 767.

Exemple

```
*MemoireID =
  AllocateMemory(5000)
If *MemoireID
  PokeW(*MemoireID, 32000)
      ; Stocke cette valeur
  dans la zone mémoire")
  Debug PeekW(*MemoireID)
      ; Lit cette valeur
  depuis la zone mémoire")
  PokeW(*MemoireID+2,
  -32000)
  Debug PeekW(*MemoireID+2)
  Debug ~32000+1
              ; Complément
  à deux
  FreeMemory(*MemoireID)
Else
  Debug "Impossible
  d'allouer la mémoire
  demandée !"
EndIf
```

Voir aussi

PeekW()

OS Supportés

Tous

116.33 PokeF

Syntaxe

```
PokeF(*Memoire, Valeur.f)
```

Description

Ecrit un float en mémoire.

Arguments

***Memoire** L'adresse mémoire où écrire.

Valeur.f La valeur à écrire.

Valeur de retour

Aucune.

Remarques

Représente 4 octets représentant un nombre à virgule flottante en simple précision.

Exemple

```
*MemoireID =
  AllocateMemory(5000)
If *MemoireID
  PokeF(*MemoireID,
  12345.12345)    ; Stocke
  cette valeur dans la zone
  mémoire")
  Debug PeekF(*MemoireID)
            ; Lit cette
  valeur depuis la zone
  mémoire")
  FreeMemory(*MemoireID)
Else
  Debug "Impossible
  d'allouer la mémoire
  demandée !"
EndIf
```

Voir aussi

PeekF()

OS Supportés

Tous

116.34 PokeS

Syntaxe

```
Resultat = PokeS(*Memoire,
    Texte$ [, Longueur [,
    Options]])
```

Description

Ecrit une chaîne de caractères, suivi d'un caractère #NULL de fin de chaîne en mémoire.

Arguments

***Memoire** L'adresse mémoire où écrire.

Texte$ La chaîne de caractères à écrire.

Longueur (optionnel) Longueur maximale (en caractères).
Si ce paramètre n'est pas spécifié ou égal à -1 alors toute la chaîne est écrite.
Le caractère #NULL de fin de chaîne qui est toujours écrit (sauf si #PB_String_NoZero est défini) n'est pas inclus dans ce décompte.

Options (optionnel) Peut prendre l'une des valeurs suivantes :

```
#PB_Ascii   : Format ASCII
#PB_UTF8    : Format UTF-8
#PB_Unicode: Format
 Unicode (Par défaut)
Combiné avec
#PB_String_NoZero: N'écrit
 pas le caractère #NULL
 de fin de chaîne.
```

Valeur de retour

Renvoie le nombre d'octets écrits, non compris le caractère #NULL de fin de chaîne.
Le nombre d'octets écrit diffère de la longueur de la chaîne en caractères si le format est #PB_UTF8 ou #PB_Unicode.

Exemple

```
1  *MemoireID =
    AllocateMemory(5000)
2  If *MemoireID
```

```
3     PokeS(*MemoireID,
      "PureBasic hé")   ; Stocke
      cette valeur dans la zone
      mémoire")
4     Debug PeekS(*MemoireID)
                ; Lit cette
      valeur depuis la zone
      mémoire")
5     FreeMemory(*MemoireID)
6   Else
7     Debug "Impossible
      d'allouer la mémoire
      demandée !"
8   EndIf
```

Voir aussi

PeekS() , CopyMemoryString()

OS Supportés

Tous

116.35 PokeU

Syntaxe

```
PokeU(*Memoire, Valeur.u)
```

Description

Ecrit un caractère unicode en mémoire.

Arguments

***Memoire** L'adresse mémoire où écrire.
Valeur.u La valeur à écrire.

Valeur de retour

Aucune.

Exemple

```
1   *MemoireID =
      AllocateMemory(5000)
2   If *MemoireID
3     PokeU(*MemoireID, 165)
      ; Stocke cette valeur dans
      la zone mémoire")
4     Debug PeekU(*MemoireID)
      ; Lit cette valeur depuis
      la zone mémoire")
5     Debug
      Chr(PeekU(*MemoireID))
6     FreeMemory(*MemoireID)
```

```
7   Else
8     Debug "Impossible
      d'allouer la mémoire
      demandée !"
9   EndIf
```

Voir aussi

PeekU()

OS Supportés

Tous

Chapitre 117

Menu

Généralités

Il est très simple de gérer des menus avec PureBasic, et vous pourrez bien sûr paramétrer tout ce dont vous avez besoin. Pour utiliser un menu, vous devez d'abord le créer avec CreateMenu() pour les menus normaux, ou CreatePopupMenu() pour les menus contextuels (pop-up).
MacOS :
Sur MacOS un menu n'est jamais affiché dans une fenêtre, mais toujours sur le bureau. Le menu en haut du bureau affiche les éléments de l'application qui a le focus. Il existe les événements de menu prédéfinis suivants #PB_Menu_Quit, #PB_Menu_About et #PB_Menu_Preferences présents dans le menu de chaque programme MacOS. Leurs valeurs sont négatives pour ne pas entrer en conflit avec les éléments de menu définis par ailleurs dans le programme qui eux sont signalés avec la fonction EventMenu() .

OS Supportés

Tous

117.1 CloseSubMenu

Syntaxe

```
CloseSubMenu()
```

Description

Ferme le sous menu.

Arguments

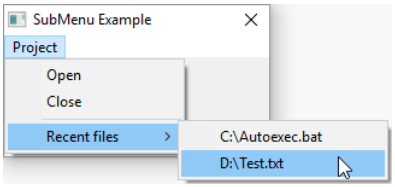
Aucun.

Valeur de retour

Aucune.

Exemple

```
1   If OpenWindow(0, 200, 200,
    220, 100, "Exemple
    SubMenu")
2    If CreateMenu(0,
    WindowID(0))
3       MenuTitle("Projet")
4         MenuItem(1, "Ouvrir")
5         MenuItem(2, "Fermer")
6         MenuBar()
7         OpenSubMenu("Fichiers
    Récents...") ; Crée un
    sous-menu
8           MenuItem( 3,
    "C:\Autoexec.bat")
9           MenuItem( 4,
    "D:\Teste.txt")
10        CloseSubMenu()
                          ;
    Termine la création du
    sous-menu
11   EndIf
12   Repeat : Until
    WaitWindowEvent()=#PB_Event_CloseWindow
13  EndIf
```



Voir aussi

OpenSubMenu()

OS Supportés

Tous

117.2 CreateMenu

Syntaxe

```
Resultat = CreateMenu(#Menu,
    FenetreID)
```

Description

Crée un nouveau menu vide sur la fenêtre spécifiée.

Arguments

#**Menu** Le numéro d'identification du menu.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

FenetreID La fenêtre qui accueille le nouveau menu.
Ce numéro peut être obtenu avec la fonction WindowID() .

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé le numéro du menu est renvoyé en cas de succès.

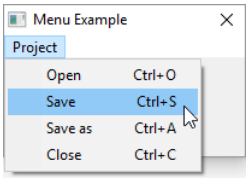
Remarques

Pour créer un menu avec des images, utilisez CreateImageMenu() .
Une fois créé, il devient le menu courant et il est alors possible d'utiliser les fonctions MenuTitle() , MenuItem() , MenuBar() , et OpenSubMenu() pour remplir le menu.
Pour gérer les évènements relatifs aux menus, voir la description des commandes suivantes :
WaitWindowEvent() (ou WindowEvent())
EventWindow()
EventMenu()

Exemple

```
1   UsePNGImageDecoder()
2
3   If OpenWindow(0, 200, 200,
     200, 100, "Image menu
     Example")
4     If CreateImageMenu(0,
     WindowID(0))     ; menu
     creation starts....
5       MenuTitle("Project")
6         MenuItem(1, "Open"
     +Chr(9)+"Ctrl+O",
     LoadImage(0,
     #PB_Compiler_Home +
     "examples/sources/Data/ToolBar/Open.png"))
7         MenuItem(2, "Save"
     +Chr(9)+"Ctrl+S",
     LoadImage(1,
     #PB_Compiler_Home +
     "examples/sources/Data/ToolBar/Save.png"))
8         MenuBar()
9         MenuItem(3, "Quit"
     +Chr(9)+"Ctrl+Q")
10    EndIf
```

```
11
12      Repeat : Until
        WaitWindowEvent() =
        #PB_Event_CloseWindow
13    EndIf
```



Voir aussi

CreateImageMenu() , CreatePopupMenu() , CreatePopupImageMenu() , FreeMenu() , MenuTitle() , MenuItem() , MenuBar() , OpenSubMenu()

OS Supportés

Tous

117.3 CreateImageMenu

Syntaxe

```
Resultat =
    CreateImageMenu(#Menu,
    FenetreID [, Options])
```

Description

Crée un nouveau menu image sur la fenêtre spécifiée.

Arguments

#Menu Le numéro d'identification du menu.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

FenetreID La fenêtre qui accueille le nouveau menu.
Ce numéro peut être obtenu avec la fonction WindowID() .

Options (optionnel) Peut être une combinaison de :

```
#PB_Menu_ModernLook:
 Active le nouveau look
 et affiche un dégradé
 (Windows seulement).
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé le numéro du menu est renvoyé en cas de succès.

Remarques

Une fois créé, il devient le menu courant et il est alors possible d'utiliser les fonctions MenuTitle() , MenuItem() , MenuBar() , et OpenSubMenu() pour remplir le menu.
Les dimensions des images sont de 16x16 pixels.
Pour gérer les évènements relatifs aux menus, voir la description des commandes suivantes :
WaitWindowEvent() (ou WindowEvent())
EventWindow()
EventMenu()

Exemple

```
1   If OpenWindow(0, 200, 200,
    220, 120, "Exemple de menu
    contextuel")
2    UsePNGImageDecoder()
                ; Pour utiliser
    les images png
3    If
    CreateImage(0,16,16,32)
        ; Obligatoirement aux
    dimensions: 16x16
4       StartDrawing(ImageOutput(0))
    ; Un petit dessin
5       Box(0,0,15,15,RGB(255,255,128))
6       DrawRotatedText(-7, 5,
    "->", 45, RGB(255,0,128))
7       StopDrawing()
8    EndIf
9                                   ;
    On peut aussi charger une
    icône toute prête.
10   If
    LoadImage(1,#PB_Compiler_Home
    +
    "Examples\Sources\Data\ToolBar\Open.png")
11   If CreateImageMenu(0,
    WindowID(0),
    #PB_Menu_ModernLook) ; La
    création du menu
    contextuel commence...
12      MenuTitle("Projet")
13        MenuItem(1, "Ouvrir"

    +Chr(9)+"Ctrl+O",ImageID(1))
    ; Ajout de l'icône
```

```
14        MenuItem(2,
          "Enregistrer"
          +Chr(9)+"Ctrl+E")
15        MenuItem(3,
          "Enregistrer
          sous"+Chr(9)+"Ctrl+R")
16        MenuItem(4, "Quitter"
          +Chr(9)+"Alt+F4",ImageID(0))
          ; Ajout de l'image perso
17        MenuBar()
18        OpenSubMenu("Fic&hiers
          récents")
19          MenuItem(5,
          "PureBasic.exe")
20          MenuItem(6,
          "Test.txt")
21        CloseSubMenu()
22
23      EndIf
24      Repeat
25        Event =
          WaitWindowEvent()
26        Select Event
                                ;
          Examine quel type
          d'évènement est survenu
          sur la fenêtre
27        Case
          #PB_Event_RightClick
            ; Le bouton droit
          de la souris a été clické
          =>
28          DisplayPopupMenu(0,WindowID(0))
          ; On affiche le menu
          contextuel
29
30        Case #PB_Event_Menu
            ; Un élément du menu
          a été sélectionné
31          Select EventMenu()
          ; On recupère le
          numéro de cet élement...
32            Case 1 : Debug
          "Menu : Ouvrir"
33            Case 2 : Debug
          "Menu : Enregistrer"
34            Case 3 : Debug
          "Menu : Enregistrer sous"
35            Case 4 : End
36            Case 5 : Debug
          "Menu : PureBasic.exe"
37            Case 6 : Debug
          "Menu : Text.txt"
38          EndSelect
39
40        EndSelect
41      Until Event =
          #PB_Event_CloseWindow
```

```
42    EndIf
43    EndIf
```

Voir aussi

CreateMenu() , CreatePopupMenu() , CreatePopupImageMenu() , FreeMenu() , MenuTitle() , MenuItem() , MenuBar() , OpenSubMenu()

OS Supportés

Tous

117.4 CreatePopupMenu

Syntaxe

```
Resultat =
    CreatePopupMenu(#Menu)
```

Description

Crée un nouveau menu contextuel (PopUp).

Arguments

#**Menu** Le numéro d'identification du menu.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé le numéro du menu est renvoyé en cas de succès.

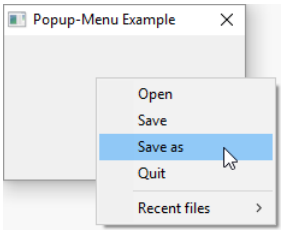
Remarques

Il est possible d'utiliser des images en créant un menu popup avec CreatePopupImageMenu.
Une fois créé, il devient le menu courant et il est alors possible d'utiliser les fonctions MenuTitle() , MenuItem() , MenuBar() , et OpenSubMenu() pour remplir le menu.
La fonction DisplayPopupMenu() sera ensuite utilisée pour afficher ce menu contextuel à l'écran.
Pour gérer les évènements relatifs aux menus, voir la description des commandes suivantes :
WaitWindowEvent() (ou WindowEvent())
EventWindow()
EventMenu()

Exemple

```
If OpenWindow(0, 200, 200, 220, 120, "Exemple de menu contextuel")
 If CreatePopupMenu(0)
       ; La création du menu contextuel commence...
   MenuTitle("Projet")
       ; Vous pouvez utiliser toutes les commandes relatives au menu,
    MenuItem(1, "Ouvrir")
   ; comme si c'était un menu normal...
    MenuItem(2, "Enregistrer")
    MenuItem(3, "Enregistrer sous")
    MenuItem(4, "Quitter")
    MenuBar()
    OpenSubMenu("Fic&hiers récents")
       MenuItem(5, "PureBasic.exe")
       MenuItem(6, "Test.txt")
    CloseSubMenu()

 EndIf
 Repeat
   Event = WaitWindowEvent()
   Select Event
                          ; Examine quel type d'évènement est survenu sur la fenêtre
    Case #PB_Event_RightClick
        ; Le bouton droit de la souris a été clické =>
       DisplayPopupMenu(0,WindowID(0)) ; On affiche le menu contextuel

    Case #PB_Event_Menu
     ; Un élément du menu a été sélectionné
       Select EventMenu()
     ; On recupère le numéro de cet élement...
         Case 1 : Debug "Menu : Ouvrir"
         Case 2 : Debug "Menu : Enregistrer"
```

```
25              Case 3 : Debug "Menu : Enregistrer sous"
26              Case 4 : End
27              Case 5 : Debug "Menu : PureBasic.exe"
28              Case 6 : Debug "Menu : Text.txt"
29            EndSelect
30
31        EndSelect
32      Until Event = #PB_Event_CloseWindow
33    EndIf
```



Voir aussi

CreatePopupImageMenu() , DisplayPopupMenu() , CreateMenu() , CreateImageMenu() , FreeMenu() , MenuTitle() , MenuItem() , MenuBar() , OpenSubMenu()

OS Supportés

Tous

117.5 CreatePopupImageMenu

Syntaxe

```
Resultat =
    CreatePopupImageMenu(#Menu
    [, Options])
```

Description

Crée un nouveau menu contextuel image.

Arguments

#Menu Le numéro d'identification du menu.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Options (optionnel) Peut être une combinaison de :

```
#PB_Menu_ModernLook:
 Active le nouveau look
 et affiche un dégradé
 (Windows seulement).
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé le numéro du menu est renvoyé en cas de succès.

Remarques

Une fois créé, il devient le menu courant et il est alors possible d'utiliser les fonctions MenuTitle() , MenuItem() , MenuBar() , et OpenSubMenu() pour remplir le menu.
Les dimensions des images sont de 16x16 pixels.
La fonction DisplayPopupMenu() sera ensuite utilisée pour afficher ce menu contextuel à l'écran.
Pour gérer les évènements relatifs aux menus, voir la description des commandes suivantes :
WaitWindowEvent() (ou WindowEvent())
EventWindow()
EventMenu()

Exemple

```
1   If OpenWindow(0, 200, 200,
    220, 120, "Exemple de menu
    contextuel")
2    UsePNGImageDecoder()
                ; Pour utiliser
    les images png
3    If
    CreateImage(0,16,16,32)
        ; Obligatoirement aux
    dimensions: 16x16
4       StartDrawing(ImageOutput(0))
    ; Un petit dessin
5       Box(0,0,15,15,RGB(255,255,128))
6       DrawRotatedText(-7, 5,
    "->", 45, RGB(255,0,128))
7       StopDrawing()
8    EndIf
9                                   ;
    On peut aussi charger une
    icône toute prête.
10   If
    LoadImage(1,#PB_Compiler_Home
    +
    "Examples\Sources\Data\ToolBar\Open.png")
11   If
    CreatePopupImageMenu(0,
    #PB_Menu_ModernLook) ; La
    création du menu
    contextuel commence...
12      ;MenuTitle("Projet")
            ; Décommenter pour
    voir la différence
```

```
13        MenuItem(1,
    "Ouvrir",ImageID(1)) ;
    Ajout de l'icône
14        MenuItem(2,
    "Enregistrer")
15        MenuItem(3,
    "Enregistrer sous")
16        MenuItem(4,
    "Quitter",ImageID(0)) ;
    Ajout de l'image perso
17        MenuBar()
18        OpenSubMenu("Fic&hiers
    récents")
19          MenuItem(5,
    "PureBasic.exe")
20          MenuItem(6,
    "Test.txt")
21        CloseSubMenu()
22
23     EndIf
24     Repeat
25       Event =
    WaitWindowEvent()
26       Select Event
                                ;
    Examine quel type
    d'évènement est survenu
    sur la fenêtre
27        Case
    #PB_Event_RightClick
            ; Le bouton droit
    de la souris a été clické
    =>
28          DisplayPopupMenu(0,WindowID(0))
    ; On affiche le menu
    contextuel
29
30        Case #PB_Event_Menu
         ; Un élément du menu
    a été sélectionné
31          Select EventMenu()
        ; On recupère le
    numéro de cet élement...
32            Case 1 : Debug
    "Menu : Ouvrir"
33            Case 2 : Debug
    "Menu : Enregistrer"
34            Case 3 : Debug
    "Menu : Enregistrer sous"
35            Case 4 : End
36            Case 5 : Debug
    "Menu : PureBasic.exe"
37            Case 6 : Debug
    "Menu : Text.txt"
38          EndSelect
39
40       EndSelect
41     Until Event =
    #PB_Event_CloseWindow
```

```
42    EndIf
43    EndIf
```

OS Supportés

Tous

117.6 DisplayPopupMenu

Syntaxe

```
DisplayPopupMenu(#Menu,
    FenetreID, [X, Y])
```

Description

Affiche un Menu contextuel (pop-up).

Arguments

#Menu Le menu contextuel, avec ou sans images, à afficher.

FenetreID La fenêtre qui accueille le nouveau menu contextuel.
Ce numéro peut être obtenu avec la fonction WindowID() .

X, Y (optionnel) Les coordonnées écran où apparaîtra le menu contextuel, en pixel depuis le coin en haut et à gauche de l'écran primaire.
Si ce paramètre n'est pas spécifié, le menu est affiché à la position de la souris.

Valeur de retour

Aucune.

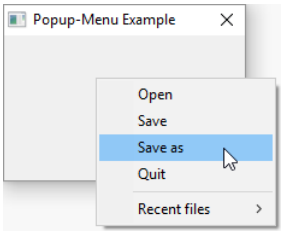
Remarques

Le menu contextuel se ferme automatiquement quand on clique sur un menu ou quand on clique en dehors du menu.

Exemple

```
1   If OpenWindow(0, 200, 200,
    220, 120, "Exemple de menu
    contextuel")
2     If CreatePopupMenu(0)
            ; La création du
    menu contextuel commence...
3       MenuTitle("Projet")
            ; Vous pouvez
    utiliser toutes les
    commandes relatives au
    menu,
```

```
4         MenuItem(1, "Ouvrir")
      ; comme si c'était un
    menu normal...
5         MenuItem(2,
    "Enregistrer")
6         MenuItem(3,
    "Enregistrer sous")
7         MenuItem(4, "Quitter")
8         MenuBar()
9         OpenSubMenu("Fic&hiers
    récents")
10          MenuItem(5,
    "PureBasic.exe")
11          MenuItem(6,
    "Test.txt")
12        CloseSubMenu()
13
14     EndIf
15     Repeat
16       Event =
    WaitWindowEvent()
17       Select Event
                              ;
    Examine quel type
    d'évènement est survenu
    sur la fenêtre
18        Case
    #PB_Event_RightClick
            ; Le bouton droit
    de la souris a été clické
    =>
19          DisplayPopupMenu(0,WindowID(0))
    ; On affiche le menu
    contextuel
20
21        Case #PB_Event_Menu
          ; Un élément du menu
    a été sélectionné
22          Select EventMenu()
        ; On recupère le
    numéro de cet élement...
23            Case 1 : Debug
    "Menu : Ouvrir"
24            Case 2 : Debug
    "Menu : Enregistrer"
25            Case 3 : Debug
    "Menu : Enregistrer sous"
26            Case 4 : End
27            Case 5 : Debug
    "Menu : PureBasic.exe"
28            Case 6 : Debug
    "Menu : Text.txt"
29          EndSelect
30
31       EndSelect
32     Until Event =
    #PB_Event_CloseWindow
33   EndIf
```



Voir aussi

CreatePopupMenu() ,
CreatePopupImageMenu()

OS Supportés

Tous

117.7 DisableMenuItem

Syntaxe

```
DisableMenuItem(#Menu,
    Element, Etat)
```

Description

Active ou désactive l'élément d'un menu.

Arguments

#Menu Le menu à utiliser.

Element Le numéro du menu à désactiver ou à réactiver.

Etat Le nouvel état d'activation du menu.

```
#False: Le menu est activé.
#True : Le menu est
 désactivé.
```

Valeur de retour

Aucune.

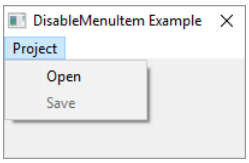
Remarques

Un menu désactivé apparaît généralement en grisé.

Exemple

```
1  If OpenWindow(0, 200, 200,
    200, 100, "Exemple
    DisableMenuItem")
2   If CreateMenu(0,
    WindowID(0))
3     MenuTitle("Projet")
```

```
4           MenuItem(1, "Ouvrir")
5           MenuItem(2,
        "Enregistrer")
6           DisableMenuItem(0,2,1)
          ; Désactive le second
        élément du menu
        ("Enregistrer")
7       EndIf
8
9       Repeat
10      Until WaitWindowEvent() =
        #PB_Event_CloseWindow
11    EndIf
```



Voir aussi

MenuItem() , SetMenuItemState() , SetMenuItemText()

OS Supportés

Tous

117.8 FreeMenu

Syntaxe

```
FreeMenu(#Menu)
```

Description

Supprime un menu et libère toutes ses ressources.

Arguments

#Menu Le menu à supprimer.
Si #PB_All est spécifié, tous les autres menus sont libérés.

Valeur de retour

Aucune.

Remarques

Tous les menus restants sont automatiquement supprimés quand le programme se termine.

Exemple

```
If OpenWindow(0, 200, 200, 300, 100, "Exemple FreeMenu")
 ButtonGadget(0,50,10,190,30,"Supprime le Menu")

 If CreateMenu(0, WindowID(0))
    MenuTitle("Projet")
    MenuItem(1, "Ouvrir")
 EndIf

 Repeat
    Event = WaitWindowEvent()
    If Event = #PB_Event_Gadget
      Select EventGadget()
        Case 0
          If IsMenu(0)  ; Le menu existe-t-il ?
            FreeMenu(0) ; Si oui alors on le supprime
          EndIf
      EndSelect
    EndIf
 Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

CreateMenu() , CreateImageMenu() , CreatePopupMenu() , CreatePopupImageMenu()

OS Supportés

Tous

117.9 GetMenuItemState

Syntaxe

```
Resultat = GetMenuItemState(#Menu, Element)
```

Description

Renvoie l'état coché d'un élément d'un menu.

Arguments

#Menu Le menu à utiliser.
Element Le numéro de l'élément.

Valeur de retour

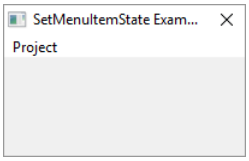
Renvoie une valeur non nulle si l'élément est coché (checké), zéro sinon.

Remarques

La coche peut être affichée ou enlevée grâce à la commande SetMenuItemState() .

Exemple

```
If OpenWindow(0, 200, 200, 300, 100, "Exemple GetMenuItemState")
 If CreateMenu(0, WindowID(0))
    MenuTitle("Projet")
      MenuItem(1, "Changé")
      SetMenuItemState(0,1,1)
    ; Affiche l'élément du menu comme étant coché.
 EndIf
 Repeat
    Evenement = WaitWindowEvent()
    ; Attente d'un évenement
    If Evenement = #PB_Event_Menu          ; Evenement de type 'Menu'
      If EventMenu() = 1
                ; Le premier élément du menu a été sélectionné
        If GetMenuItemState(0,1) = 1
          ; Etat actuel de l'élément = coché
          SetMenuItemState(0,1,0)
        ; On le décoche
        Else

    ; Etat actuel de l'élément = décoché
          SetMenuItemState(0,1,1)
        ; On le coche
        EndIf
      EndIf
    EndIf
 Until Evenement = #PB_Event_CloseWindow
EndIf
```



Voir aussi

SetMenuItemState() , GetMenuItemText() , MenuItem()

OS Supportés

Tous

117.10 GetMenuItemText

Syntaxe

```
Resultat\$ =
    GetMenuItemText(#Menu,
    Element)
```

Description

Renvoie le texte d'un menu.

Arguments

#**Menu** Le menu à utiliser.

Element L'élément du menu.

Valeur de retour

Renvoie le texte du menu déroulant.

Exemple

```
1   If OpenWindow(0, 200, 200,
     300, 100, "Exemple
     GetMenuItemText")
2   If CreateMenu(0,
     WindowID(0))
3     MenuTitle("Projet")
4     MenuItem(1, "Nouveau")
5     MenuItem(2, "Ouvrir")
6   EndIf
7
8   Repeat
9     Event = WaitWindowEvent()
10    If Event = #PB_Event_Menu
                ; un menu a été
     cliqué
11       Select EventMenu()
12         Case 1
13           ShowDebugOutput()
14           Debug " le 1er menu
     a été cliqué dont le texte
     est : " +
     GetMenuItemText(0,1)
15         Case 2
16           ShowDebugOutput()
```

```
17          Debug " le 2ième menu a été cliqué dont le texte est : " + GetMenuItemText(0,2)
18        EndSelect
19      EndIf
20    Until Event = #PB_Event_CloseWindow
21    EndIf
```

Voir aussi

SetMenuItemText() , MenuItem()

OS Supportés

Tous

117.11 GetMenuTitleText

Syntaxe

```
Resultat\$ =
    GetMenuTitleText(#Menu,
    Titre)
```

Description

Renvoie le texte d'un titre d'un menu.

Arguments

#Menu Le menu à utiliser.
Titre L'index du titre du menu.

Valeur de retour

Renvoie le texte de l'en-tête du menu.

Exemple

```
1   If OpenWindow(0, 200, 200, 300, 100, "Exemple GetMenuItemText")
2   If CreateMenu(0, WindowID(0))
3     MenuTitle("Projet")
4     MenuItem(1, "Nouveau")
5     MenuItem(2, "Ouvrir")
6   EndIf
7
8   Repeat
9     Event = WaitWindowEvent()
10    If Event = #PB_Event_Menu
              ; un menu a été cliqué
```

```
11        Select EventMenu()
12          Case 1
13            ShowDebugOutput()
14            Debug " le 1er menu
    a été cliqué qui
    appartient à : " +
    GetMenuTitleText(0,0)
15          Case 2
16            ShowDebugOutput()
17            Debug " le 2ième
    menu a été cliqué qui
    appartient à : " +
    GetMenuTitleText(0,0)
18        EndSelect
19      EndIf
20    Until Event =
      #PB_Event_CloseWindow
21    EndIf
```

Voir aussi

MenuTitle() , SetMenuTitleText()

OS Supportés

Tous

117.12 HideMenu

Syntaxe

```
HideMenu(#Menu, Etat)
```

Description

Cache ou affiche un menu.

Arguments

#**Menu** Le menu à utiliser.

Etat #True : Le menu est caché
#False : Le menu est affiché

Valeur de retour

Aucune.

Exemple : Cache un Menu

```
1   If OpenWindow(0, 200, 200,
      300, 100, "Exemple
      HideMenu")
2   ButtonGadget(10,70,10,150,30,"Cacher
      le menu")
```

```
  If CreateMenu(0,
   WindowID(0))
    MenuTitle("Projet")
    MenuItem(1, "Nouveau")
    MenuItem(2, "Ouvrir")
  EndIf

  Repeat
    Event = WaitWindowEvent()
    If Event =
   #PB_Event_Gadget
      Select EventGadget()
        Case 10
          HideMenu(0,#True) ;
   cache le menu
      EndSelect

    EndIf
  Until Event =
   #PB_Event_CloseWindow
  EndIf
```

Exemple : Affiche un Menu

```
If OpenWindow(0, 200, 200,
   300, 100, "Exemple
   HideMenu")
  ButtonGadget(0,70,10,150,30,"Affiche
   le menu")
  If CreateMenu(0,
   WindowID(0))
    MenuTitle("Projet")
    MenuItem(1, "Nouveau")
    MenuItem(2, "Ouvrir")
  EndIf

  HideMenu(0,#True) ; Cache
   le menu

  Repeat
    Event = WaitWindowEvent()
    If Event =
   #PB_Event_Gadget
      Select EventGadget()
        Case 0
        HideMenu(0,#False) ;
   Affiche le menu
      EndSelect

    EndIf
  Until Event =
   #PB_Event_CloseWindow
EndIf
```

Exemple : Change de Menu

```
If OpenWindow(0, 200, 200, 300, 100, "Exemple HideMenu")
ButtonGadget(0,70,10,150,30,"Change de menu")
If CreateMenu(0, WindowID(0))
  MenuTitle("Projet")
  MenuItem(1, "Nouveau")
  MenuItem(2, "Ouvrir")
EndIf
If CreateMenu(1, WindowID(0))
  MenuTitle("Menu")
  MenuItem(1, "Enregistrer")
  MenuItem(2, "Quitter")
EndIf

HideMenu(0,#False) ; Affiche le menu "Projet"
Repeat
  Event = WaitWindowEvent()
  If Event = #PB_Event_Gadget
    Select EventGadget()
      Case 0
        HideMenu(1,#False) ; Affiche le menu "Menu"
    EndSelect

  EndIf
Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

CreateMenu() , CreateImageMenu()

OS Supportés

Tous

117.13 IsMenu

Syntaxe

```
Resultat = IsMenu(#Menu)
```

Description

Teste si un menu est correctement initialisé.

Arguments

#**Menu** Le menu à utiliser.

Valeur de retour

Renvoie une valeur non nulle le le menu est valide, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage.

Exemple

```
If OpenWindow(0, 200, 200, 300, 100, "Exemple FreeMenu")
 ButtonGadget(0,50,10,190,30,"Supprime le Menu")

 If CreateMenu(0, WindowID(0))
    MenuTitle("Projet")
    MenuItem(1, "Ouvrir")
 EndIf

 Repeat
    Event = WaitWindowEvent()
    If Event = #PB_Event_Gadget
       Select EventGadget()
         Case 0
           If IsMenu(0)  ; Le menu existe-t-il ?
             FreeMenu(0) ; Si oui alors on le supprime
           EndIf
       EndSelect
    EndIf
 Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

CreateMenu() , CreatePopupMenu() , CreateImageMenu() , CreatePopupImageMenu()

OS Supportés

Tous

117.14 MenuBar

Syntaxe

```
MenuBar()
```

Description

Crée une barre de séparation horizontale dans un menu déroulant.

Arguments

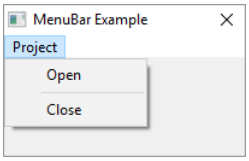
Aucun.

Valeur de retour

Aucune.

Exemple

```
1  If OpenWindow(0, 200, 200, 200, 100, "Exemple MenuBar")
2    If CreateMenu(0, WindowID(0))  ; la création du menu commence ici....
3      MenuTitle("Projet")
4        MenuItem(1, "Ouvrir")
5        MenuBar()                  ; la barre de séparation sera insérée ici
6        MenuItem(4, "Fermer")
7    EndIf
8    Repeat : Until WaitWindowEvent()=#PB_Event_CloseWindow
9  EndIf
```



Voir aussi

MenuTitle() , MenuItem() , OpenSubMenu() , CreateMenu() , CreatePopupMenu()

OS Supportés

Tous

117.15 MenuHeight

Syntaxe

```
Resultat = MenuHeight()
```

Description

Renvoie la hauteur de la barre de menu.

Arguments

Aucun.

Valeur de retour

Renvoie la hauteur, en pixels, de la barre de menu.

Remarques

Utile pour calculer la hauteur de la zone d'affichage (zone client) d'une fenêtre.
Linux & MacOS : Cette commande renverra toujours 0, car la barre de menu ne fait pas partie intégrante de la fenêtre (elle est toujours située dans la barre principale tout en haut de l'écran). Ainsi MenuHeight() peut être utilisé sous tous les OS pour ajuster la taille de la fenêtre en fonction de la hauteur de son menu.

Exemple

```
If OpenWindow(0, 200, 200, 300, 100, "Exemple MenuHeight")
 ButtonGadget(0,50,10,190,30,"Affiche la hauteur du Menu")

 If CreateMenu(0, WindowID(0))
    MenuTitle("Projet")
    MenuItem(1, "Ouvrir")
 EndIf

 Repeat
    Event = WaitWindowEvent()
    If Event = #PB_Event_Gadget
       Select EventGadget()
         Case 0
           ShowDebugOutput()
           Debug MenuHeight()
       EndSelect
    EndIf
 Until Event = #PB_Event_CloseWindow
EndIf
```

OS Supportés

Tous

117.16 MenuItem

Syntaxe

```
MenuItem(ElementID, Texte$ [,
    ImageID])
```

Description

Crée une nouvelle entrée dans un menu.

Arguments

ElementID Le numéro d'identification de la nouvelle entrée du menu.
Ce numéro sera utilisé dans les évènements Menu et renvoyé par EventMenu() , utilisé aussi dans certaines commande comme SetMenuItemState() .
Cette valeur doit être entre 0 et 65535.

Texte$ Le texte de l'élément.
Sous Windows, vous pouvez utiliser le caractère '&' pour souligner une lettre : "&Fichier" s'affichera ainsi : Fichier.

ImageID (optionnel) L'image qui sera affichée pour cet élément.
Le menu doit avoir été créé avec CreateImageMenu() ou CreatePopupImageMenu() .
'ImageID' peut être facilement obtenu avec ImageID() .

Valeur de retour

Aucune.

Remarques

Les dimensions des images sont de 16x16 pixels.
Si vous désirez associer un raccourci clavier (qui sera créé avec la commande AddKeyboardShortcut() , sauf sous MacOS) aligné sur le côté droit du menu (par exemple, "Sauvegarder Ctrl+S") alors vous pouvez utiliser le caractère 'tabulation' pour donner l'espacement correct. Le caractère 'Tabulation' a le code ASCII 9, donc il faudra utiliser la commande Chr(9) comme indiqué ci-dessous :

```
1  MenuItem(1, "&Ouvrir" +
     Chr(9) + "Ctrl+O")
```

Les raccourcis supportés sont :

```
1  - "Ctrl"  : touche
     'Contrôle'
2  - "Shift" : touche
     'Majuscule'
```

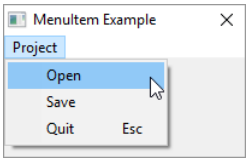
```
- "Alt"    : touche 'Alt'
- "Cmd"    : touche
  'Commande/Pomme' (MacOS
  seulement)
```

Ils peuvent être combinés entre eux à l'aide du caractère "+" : "Enregistrer sous" + Chr(9) + "Ctrl+Shift+S".
Sous MacOS, quand un raccourci est défini dans le menu, il n'est pas nécessaire de le redéclarer à l'aide de la fonction AddKeyboardShortcut() .
MacOS : les éléments 'Quitter', 'Préférences' et 'A propos' sont considérés comme spéciaux et doivent être placés dans le menu 'Application' pour avoir l'apparence et le comportement des applications MacOS. PureBasic fournit les constantes #PB_Menu_Quit, #PB_Menu_Preferences et #PB_Menu_About pour gérer ce genre d'éléments de menu. Lorsqu'une de ces constantes est détectée, l'élément n'est pas inséré à la place courante, mais dans le menu 'Application'. Si un raccourci a été associé, il est simplement ignoré et remplacé par le raccourci standard. Ces 3 constantes ne sont pas définies sur les autres systèmes d'exploitation, afin de permettre une numérotation flexible sur ces OS.

Exemple

```
;Note : Sous Windows XP et
 suivant, pour que le trait
 souligné apparaisse en
 permanence,
;décochez l'option :
 'Masquer les lettres
 soulignées pour la
 navigation au clavier
 jusqu'à ce que j'appuie
 sur la touche Alt'
;dans les propriétés du
 bureau puis affichage /
 Apparence / Effets.

If OpenWindow(0, 200, 200,
 200, 100, "Exemple
 MenuItem")
  If CreateMenu(0,
 WindowID(0))
    MenuTitle("Projet")
      MenuItem(1, "Ouvrir")
        ; Elément normal
      MenuItem(2,
 "&Enregistrer")  ; Elément
 avec une lettre soulignée.
```

```
10                                          ;
    Le trait souligné
    n'apparaîtra que si le
    menu est appelé avec F10
    ou ALT.
11      MenuItem(3,
    "Quitter"+Chr(9)+"ECHAP")
    ; Elément avec un
    raccourci clavier affiché
    sur la droite.
12    EndIf
13    Repeat : Until
    WaitWindowEvent()=#PB_Event_CloseWindow
14  EndIf
```



Voir aussi

MenuTitle() , MenuBar() , OpenSubMenu()

OS Supportés

Tous

117.17 MenuID

Syntaxe

```
Resultat = MenuID(#Menu)
```

Description

Renvoie l'identifiant système d'un menu.

Arguments

#**Menu** Le menu à utiliser.

Valeur de retour

Renvoie le numéro du menu.

Remarques

L'identifiant est parfois appelé 'Handle'.
Pour plus d'informations consultez le chapitre Numéros et Identifiants (Handles) .

Exemple

```
If OpenWindow(0, 200, 200, 300, 100, "Exemple MenuID")
ButtonGadget(0,70,10,150,30,"Affiche le MenuID")

hMenu= CreateMenu(0, WindowID(0))
  MenuTitle("Projet")
  MenuItem(1, "Ouvrir")

Repeat
  Event = WaitWindowEvent()
  If Event = #PB_Event_Gadget
    Select EventGadget()
      Case 0
        ShowDebugOutput()
        Debug MenuID(0)
        Debug hMenu
    EndSelect
  EndIf
Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

CreateMenu() , CreatePopupMenu() , CreateImageMenu() , CreatePopupImageMenu()

OS Supportés

Tous

117.18 MenuTitle

Syntaxe

```
MenuTitle(Titre$)
```

Description

Crée un nouveau titre de menu.

Arguments

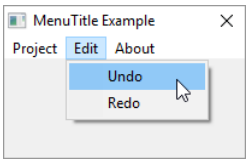
Titre$ Le texte du nouveau titre de menu. Sous Windows, vous pouvez utiliser le caractère & pour souligner une lettre si le thème graphique le permet, ainsi : "&Fichier" affichera une ligne de menu Fichier .

Remarques

MenuTitle() ne peut pas être utilisés dans les menus popup , car ils sont dépourvus d'en-têtes de menus.

Exemple

```
;Note : Sous Windows XP et suivant, pour que le trait souligné apparaisse en permanence,
;décochez l'option : 'Masquer les lettres soulignées pour la navigation au clavier jusqu'à ce que j'appuie sur la touche Alt'
;dans les propriétés du bureau puis affichage / Apparence / Effets.
If OpenWindow(0, 200, 200, 200, 100, "Exemple MenuTitle")
 If CreateMenu(0, WindowID(0))
    MenuTitle("Projet")
    ; Titre normal
      MenuItem(1, "Ouvrir")
      MenuItem(2, "Fermer")
    MenuTitle("&Edition")
       ; Titre avec une lettre soulignée. Le trait souligné n'apparaîtra que                                    ;
si le menu est appelé avec F10 ou ALT.
      MenuItem(3, "Défaire")
      MenuItem(4, "Refaire")
    MenuTitle("About")
       ; Titre seul
 EndIf
 Repeat : Until WaitWindowEvent()=#PB_Event_CloseWindow
EndIf
```



OS Supportés

Tous

117.19 OpenSubMenu

Syntaxe

```
OpenSubMenu(Texte$ [,
    ImageID])
```

Description

Crée un sous menu.

Arguments

Texte$ Le texte du nouveau sous menu. Vous pouvez utiliser le caractère & pour souligner une lettre, ainsi : "&Fichier" affichera une ligne de menu Fichier avec le F souligné.

ImageID (optionnel) L'image affichée pour cet élément. Le menu doit avoir été créé avec CreateImageMenu() ou CreatePopupImageMenu() . 'ImageID' peut être facilement obtenu avec ImageID() .

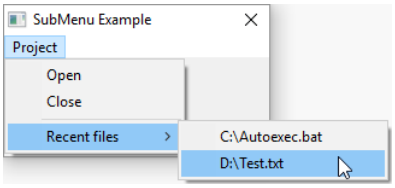
Remarques

Il n'est pas possible de renommer un OpenSubMenu aisément, sauf sous Windows qui lui, renvoie un numéro de menu.
Ex :
SubMenu = OpenSubMenu("Nouveau")
SetMenuItemText(0, SubMenu, "Ouvrir")
Sous Linux et MacOS vous devrez jongler entre plusieurs menus ou détruire et recréer le menu à convenance.

Exemple

```
1  If OpenWindow(0, 200, 200,
    220, 100, "Exemple
    SubMenu")
2     If CreateMenu(0,
    WindowID(0))
3        MenuTitle("Projet")
4          MenuItem(1, "Ouvrir")
5          MenuItem(2, "Fermer")
6          MenuBar()
7          OpenSubMenu("Fichiers
    Récents...") ; Crée un
    sous-menu
8             MenuItem( 3,
    "C:\Autoexec.bat")
9             MenuItem( 4,
    "D:\Teste.txt")
10         CloseSubMenu()
                          ;
    Termine la création du
    sous-menu
```

```
    EndIf
    Repeat : Until WaitWindowEvent()=#PB_Event_CloseWindow
  EndIf
```



Voir aussi

CloseSubMenu() , MenuTitle() , MenuItem() , MenuBar()

OS Supportés

Tous

117.20 SetMenuItemState

Syntaxe

```
SetMenuItemState(#Menu,
    Element, Etat)
```

Description

Change l'état coché d'un élément d'un menu.

Arguments

#**Menu** Le menu à utiliser.

Element Le menu qui sera coché.

Etat `#False` : La coche n'apparait pas.
`#True` : La coche apparait.

Valeur de retour

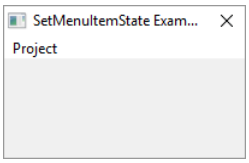
Aucune.

Remarques

GetMenuItemState() peut être utilisé pour récupérer l'état actuel de l'élément du menu.

Exemple

```
1  If OpenWindow(0, 200, 200,
   200, 100, "Exemple
   SetMenuItemState")
2   If CreateMenu(0,
   WindowID(0))
3      MenuTitle("Projet")
4        MenuItem(1, "Changé")
5        SetMenuItemState(0,1,1)
       ; Coche l'élément 1 du
   menu.
6   EndIf
7   Repeat : Until
   WaitWindowEvent()=#PB_Event_CloseWindow
8  EndIf
```



Voir aussi

GetMenuItemState()

OS Supportés

Tous

117.21 SetMenuItemText

Syntaxe

```
SetMenuItemText(#Menu,
    Element, Texte$)
```

Description

Change le texte d'un élément d'un menu.

Arguments

#Menu Le menu à utiliser.
Element Le numéro de l'élément.
Texte$ Le nouveau texte.

Valeur de retour

Aucune.

Exemple

```
If OpenWindow(0, 200, 200, 300, 100, "Exemple SetMenuItemText")
 ButtonGadget(0,70,10,150,30,"Changer le menu Ouvrir")
 If CreateMenu(0, WindowID(0))
   MenuTitle("Projet")
   MenuItem(1, "Ouvrir")
 EndIf

 Repeat
   Event = WaitWindowEvent()
   If Event = #PB_Event_Gadget
     Select EventGadget()
       Case 0
         SetMenuItemText(0,1,"Nouveau") ; Change le menu "Ouvrir" par "Nouveau"
     EndSelect
   EndIf
 Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

GetMenuItemText() , MenuItem()

OS Supportés

Tous

117.22 SetMenuTitleText

Syntaxe

```
SetMenuTitleText(#Menu, Titre, Texte$)
```

Description

Change le texte d'un titre d'un menu.

Arguments

#Menu Le menu à utiliser.

Titre Le numéro du titre du menu.

Texte$ Le nouveau texte.

Valeur de retour

Aucune.

Exemple

```
If OpenWindow(0, 200, 200, 300, 100, "Exemple SetMenuTitleText")
 ButtonGadget(0,70,10,150,30,"Changer le menu Projet")
 If CreateMenu(0, WindowID(0))
    MenuTitle("Projet")
    MenuItem(1, "Ouvrir")
 EndIf

 Repeat
    Event = WaitWindowEvent()
    If Event = #PB_Event_Gadget
       Select EventGadget()
         Case 0
          SetMenuTitleText(0,0,"Fichier") ; Change le menu "Projet" par "Fichier"
       EndSelect
    EndIf
 Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

GetMenuTitleText() , MenuTitle()

OS Supportés

Tous

117.23 BindMenuEvent

Syntaxe

```
BindMenuEvent(#Menu, Element, @Callback())
```

Description

Ajoute un événement du menu grâce à une procédure dite de 'callback'.

Arguments

#Menu Le menu à utiliser.

Element L'élément de menu.

@Callback() La procédure à appeler lorsque l'événement se produit. Elle doit être déclarée comme ceci :

```
1  Procedure EventHandler()
2    ; Du code ici...
3  EndProcedure
```

Les fonction de PureBasic comme EventGadget() , EventWindow() , EventMenu() , EventType() et EventData() sont disponibles pour obtenir plus d'informations sur l'événement.
Note : WindowEvent() et WaitWindowEvent() ne devraient jamais être appelées à l'intérieur du Callback(), car ça peut verrouiller un programme ou occasionner un comportement erroné.

Valeur de retour

Aucune.

Remarques

C'est un moyen supplémentaire pour gérer les événements dans PureBasic, qui fonctionne sans problème avec les habituelles commandes WindowEvent() et WaitWindowEvent() .
Cet évènement peut être supprimé avec UnbindMenuEvent() .

Exemple

```
1  Procedure TestHandler()
2    Debug "Evènement : Menu -Test-"
3  EndProcedure
4
5  Procedure QuitHandler()
6    Debug "Evènement : Menu -Quitter-"
7    End
8  EndProcedure
9
10 OpenWindow(0, 100, 100, 200, 50, "Test Clic", #PB_Window_SystemMenu)
11
12 CreateMenu(0, WindowID(0))
13   MenuTitle("Fichier")
14     MenuItem(0, "Test")
15     MenuItem(1, "Quitter")
16
17 BindMenuEvent(0, 0, @TestHandler())
18 BindMenuEvent(0, 1, @QuitHandler())
19
20 Repeat
```

```
21      Event = WaitWindowEvent()
22    Until Event =
       #PB_Event_CloseWindow
```

Voir aussi

BindGadgetEvent() , BindMenuEvent() , UnbindEvent() , WindowEvent() , WaitWindowEvent()

OS Supportés

Tous

117.24 UnbindMenuEvent

Syntaxe

```
UnbindMenuEvent(#Menu,
    Element, @Callback())
```

Description

Supprime un événement d'un menu provenant d'une procédure dite de 'callback' Si l'événement correspondant n'est pas trouvé, cette commande n'a aucun effet.

Arguments

#Menu Le menu à utiliser.

Element L'élément de menu.

@Callback() La procédure de 'Callback' à supprimer.

Valeur de retour

Aucune.

Exemple

```
1   Procedure TestHandler()
2     Debug "Evènement : Menu
      -Test-"
3   EndProcedure
4
5   Procedure QuitHandler()
6     Debug "Evènement : Menu
      -Quitter-"
7     End
8   EndProcedure
9
10  OpenWindow(0, 100, 100,
      200, 50, "Test Clic",
      #PB_Window_SystemMenu)
11
```

```
12  CreateMenu(0, WindowID(0))
13    MenuTitle("Fichier")
14      MenuItem(0, "Test")
15      MenuItem(1, "Quitter")
16
17  BindMenuEvent(0, 0,
     @TestHandler())
18  BindMenuEvent(0, 1,
     @QuitHandler())
19
20  UnbindMenuEvent(0, 1,
     @QuitHandler()) ; Supprime
     l'évènement -Quitter-
21
22  Repeat
23    Event = WaitWindowEvent()
24  Until Event =
     #PB_Event_CloseWindow
```

Voir aussi

BindEvent() , BindGadgetEvent() , BindMenuEvent() , WindowEvent() , WaitWindowEvent()

OS Supportés

Tous

Chapitre 118

Mesh

Généralités

Les Meshs (maillages en 3D) sont des objets 3D composés de triangles reliés entre eux pour donner une forme finale, toujours en 3D. Un Mesh peut posséder un squelette s'il est animé, permettant des animations réalistes et de qualité. Un Mesh ne peut pas être affiché directement dans le monde 3D, il doit être encapsulé dans une entity . InitEngine3D() doit être appelé avec succès avant de pouvoir utiliser les commandes relatives aux Meshs.

OS Supportés

Tous

118.1 CreateMesh

Syntaxe

```
Resultat = CreateMesh(#Mesh
    [, Type [, Mode])
```

Description

Créé un nouveau mesh complètement vide.

Arguments

#Mesh Le numéro d'identification du nouveau mesh.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Type (optionnel) Le type du nouveau mesh. Peut être une des valeurs suivantes :

```
#PB_Mesh_TriangleList : Le
 mesh est composé d'une
 liste de triangles (par
 défaut).
```

```
#PB_Mesh_TriangleStrip: Le
 mesh est composé d'une
 liste de triangles
 connectés (Les sommets
 sont partagés).
#PB_Mesh_TriangleFan  : Le
 mesh est composé d'une
 liste de triangles qui
 partagent le même sommet
 central .
#PB_Mesh_PointList    : Le
 mesh est composé d'une
 liste de points.
#PB_Mesh_LineList     : Le
 mesh est composé d'une
 liste de lignes.
#PB_Mesh_LineStrip    : Le
 mesh est composé d'une
 liste de lignes
 connectées (Les sommets
 sont partagés).
```

Mode (optionnel) Le mode du nouveau mesh. Peut être une des valeurs suivantes :

```
#PB_Mesh_Static : Une fois
 créé, le mesh ne peut
 plus être modifié avec
 les fonctions de mise à
 jour de mesh (par
 défaut).
#PB_Mesh_Dynamic: Une fois
 créé, le mesh peut être
 modifié avec les
 fonctions de mise à jour
 de mesh.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Mesh, alors la valeur de retour est le numéro d'identification, généré automatiquement.

Remarques

Après création, les commandes comme MeshVertexPosition() ou MeshFace() peuvent être utilisés pour le construire.
Si le mesh était déjà créé, il est automatiquement supprimé et remplacé par le nouveau.

OS Supportés

Tous

118.2 CreateDataMesh

Syntaxe

```
Resultat =
    CreateDataMesh(#Mesh,
    Array.MeshVertex() [,
    Mode])
```

Description

Crée un nouveau #Mesh à partir d'un tableau à 2 dimensions du type MeshVertex.

Arguments

#Mesh Le numéro d'identification du nouveau mesh.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Array.MeshVertex() Le tableau à 2 dimensions de type MeshVertex.
La structure MeshVertex est définie comme suit :

```
1   Structure MeshVertex
2     x.f
3     y.f
4     z.f
5     NormalX.f
6     NormalY.f
7     NormalZ.f
8     TangentX.f
9     TangentY.f
10    TangentZ.f
11    u.f
12    v.f
13    Couleur.l
14  EndStructure
```

Mode (optionnel) La façon de lier les sommets. Il peut s'agir de l'une des valeurs suivantes :

```
#PB_Mesh_DiagonalRegular1
      : Les diagonales
 sont alignées de le même
 sens
#PB_Mesh_DiagonalRegular2
      : Les diagonales
 sont alignées dans
 l'autre sens
#PB_Mesh_DiagonalAlternate
     : Les diagonales
 sont une fois dans un
 sens et une fois dans
 l'autre sens (alterné)
#PB_Mesh_DiagonalShortestLength:
 Les diagonales sont
 liées entre les deux
 sommets les plus proches
```

```
#PB_Mesh_DiagonalClosestNormal
 : Les diagonales sont
 liées entre les deux
 normales les plus
 proches (produit
 scalaire). Meilleur mode
 mais nécessite des
 normales.
```

Remarques

Si #Mesh a déjà été créé, il est libéré et remplacé par le nouveau.
Cette commande permet de créer un Mesh plus rapidement que d'utiliser CreateMesh() avec un tableau ad-hoc.

OS Supportés

Tous

118.3 CopyMesh

Syntaxe

```
Resultat = CopyMesh(#Mesh,
    #NouveauMesh)
```

Description

Copie un mesh.

Arguments

#Mesh Le numéro d'identification du mesh à copier.

#NouveauMesh Le numéro d'identification du nouveau mesh. Si #PB_Any est utilisé à la place de '#NouveauMesh' alors le numéro du nouveau mesh sera renvoyé dans 'Resultat'.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Si le #NouveauMesh était déjà créé, il est automatiquement supprimé et remplacé par le nouveau.
Les meshes dynamiques créés avec l'option #PB_Mesh_Dynamic, ne sont pas autorisés.

Voir aussi

CreateMesh()

OS Supportés

Tous

118.4 FreeMesh

Syntaxe

```
FreeMesh(#Mesh)
```

Description

Supprime un mesh.

Arguments

Mesh Le mesh à utiliser, précédemment créé à l'aide de la commande CreateMesh() ou LoadMesh() .
Si #PB_All est spécifié, tous les meshes restants sont libérés.

Valeur de retour

Aucune.

Remarques

Tous les meshes restants sont automatiquement supprimés quand le programme se termine.

OS Supportés

Tous

118.5 IsMesh

Syntaxe

```
Resultat = IsMesh(#Mesh)
```

Description

Teste si un mesh est correctement initialisé.

Arguments

Mesh Le mesh à utiliser.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.

Voir aussi

CreateMesh()

OS Supportés

Tous

118.6 LoadMesh

Syntaxe

```
Resultat = LoadMesh(#Mesh,
    NomFichier$)
```

Description

Charge un mesh à partir d'unu fichier au format '.mesh'.

Arguments

#Mesh Le numéro d'identification du nouveau mesh.
Si #PB_Any est utilisé alors le numéro du nouveau mesh sera renvoyé dans 'Resultat'.

NomFichier$ Nom du fichier contenant le mesh.
Avant de charger un mesh, une archive doit être spécifiée avec Add3DArchive() .

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Mesh, alors la valeur de retour est le numéro d'identification généré automatiquement.

Remarques

Le mesh doit nécessairement être au format OGRE '.mesh'.
Un outil en ligne de commande est disponible à cette adresse assimp pour convertir de nombreux formats 3D dans le format OGRE, y compris les matériaux et les animations. Il peut être téléchargé ici :

OgreAssimpConverter.zip (Windows uniquement). Certains problèmes ont été signalés avec l'ombre de meshes convertis, si cela arrive, utiliser OgreMeshUpdater.exe sur le mesh nouvellement converti et ça devrait le réparer.
Il est possible d'utiliser des "exporters" pour : Milkshape, Lightwave, Blender or 3DS Max.
Plus d'informations sur le site d' OGRE.

Voir aussi

CreateMesh() , FreeMesh()

OS Supportés

Tous

118.7 MeshID

Syntaxe

```
Resultat = MeshID(#Mesh)
```

Description

Renvoie l'identifiant d'un mesh.

Arguments

#Mesh Le numéro d'identification du nouveau mesh.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Valeur de retour

Renvoie le numéro d'identification unique du #Mesh.

Remarques

Cet identifiant est notamment nécessaire pour la commande CreateEntity() .

Voir aussi

CreateMesh()

OS Supportés

Tous

118.8 GetMeshData

Syntaxe

```
Resultat = GetMeshData(#Mesh,
    SousMesh, TableauData(),
    Options, PremierIndex,
    DernierIndex)
```

Description

Obtenir les données internes du mesh, comme les sommets, les faces, etc.
(Les meshes dynamiques, créés avec l'option #PB_Mesh_Dynamic, ne sont pas supportés).

Arguments

#Mesh Le mesh à utiliser.

SousMesh Le SousMesh à utiliser. L'indice du premier SousMesh commence à 0 (mesh principal).

TableauData() Le tableau pour recevoir les données. Il doit être un tableau de type "MeshVertex" ou "MeshFace" en fonction des paramètres 'Options'.

Options Indique le type de données qui doivent être récupérées. Peut être l'une des valeurs suivantes :

```
#PB_Mesh_Vertex:
 TableauData() est un
 tableau de type
 "MeshVertex".
#PB_Mesh_Face  :
 TableauData() est un
 tableau de type
 "MeshFace".
```

En combinaison avec :

```
#PB_Mesh_UVCoordinate :
 Obtenir les informations
 de coordonnées UV
 (seulement pour l'option
 #PB_Mesh_Vertex)
#PB_Mesh_Normal       :
 Obtenir les informations
 de la 'normale'
 (seulement pour l'option
 #PB_Mesh_Vertex)
#PB_Mesh_Color        :
 Obtenir les informations
 de couleurs (seulement
 pour l'option
 #PB_Mesh_Vertex)
#PB_Mesh_Tangent      :
 Obtenir les informations
 de tangente (seulement
```

```
pour l'option
#PB_Mesh_Vertex)
```

Les structures "MeshVertex" et "MeshFace" sont définis comme suit :

```
Structure MeshVertex
  x.f
  y.f
  z.f
  NormalX.f ; qu'avec
 l'option #PB_Mesh_Normal
  NormalY.f ;
  NormalZ.f ;
  TangentX.f
  TangentY.f
  TangentZ.f
  u.f       ; qu'avec
 l'option
 #PB_Mesh_UVCoordinate
  v.f       ;
  Color.l   ; qu'avec
 l'option #PB_Mesh_Color
EndStructure

Structure MeshFace
  Index.l
EndStructure
```

PremierIndex, DernierIndex Premier et dernier index pour obtenir les données.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon. En cas de succès, TableauData() a été redimensionnée et contient les informations du mesh.

Voir aussi

SetMeshData()

OS Supportés

Tous

118.9 SetMeshData

Syntaxe

```
Resultat = SetMeshData(#Mesh,
   SousMesh, TableauData(),
   Options, PremierIndex,
   DernierIndex)
```

Description

Changer les données internes du mesh, comme les sommets, les faces, etc.

(Les meshes dynamiques, créés avec l'option #PB_Mesh_Dynamic, ne sont pas supportés).

Arguments

#Mesh Le mesh à utiliser.

SousMesh Le SousMesh à utiliser.
L'indice du premier SousMesh commence à 0 (mesh principal).

TableauData() Le tableau qui contient les données.
Ce doit être un tableau de type "MeshVertex" ou "MeshFace" en fonction des paramètres 'Options'.

Options Indique le type de données qui doivent être envoyées. Peut être l'une des valeurs suivantes :

```
#PB_Mesh_Vertex:
 TableauData() est un
 tableau de type
 "MeshVertex".
#PB_Mesh_Face  :
 TableauData() est un
 tableau de type
 "MeshFace".
```

En combinaison avec :

```
#PB_Mesh_UVCoordinate :
 Les informations de
 coordonnées UV
 (seulement pour l'option
 #PB_Mesh_Vertex)
#PB_Mesh_Normal       :
 Les informations de la
 'normale' (seulement
 pour l'option
 #PB_Mesh_Vertex)
#PB_Mesh_Color        :
 Les informations de
 couleurs (seulement pour
 l'option #PB_Mesh_Vertex)
```

Les structures "MeshVertex" et "MeshFace" sont définis comme suit :

```
Structure MeshVertex
  x.f
  y.f
  z.f
  NormalX.f ; qu'avec
 l'option #PB_Mesh_Normal
  NormalY.f ;
  NormalZ.f ;
  TangentX.f
  TangentY.f
  TangentZ.f
  u.f       ; qu'avec
 l'option
 #PB_Mesh_UVCoordinate
```

```
    v.f        ;
    Color.l    ; qu'avec
   l'option #PB_Mesh_Color
  EndStructure

  Structure MeshFace
    Index.l
  EndStructure
```

PremierIndex, DernierIndex Premier et dernier index des données.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Voir aussi

GetMeshData()

OS Supportés

Tous

118.10 BuildMeshShadowVolume

Syntaxe

```
BuildMeshShadowVolume(#Mesh)
```

Description

Crée le volume d'ombre d'un mesh.

Arguments

#**Mesh** Le mesh à utiliser.

Valeur de retour

Aucune.

Remarques

Nécessaire si le mesh doit jeter des ombres. Cela doit être fait une fois la création du mesh complètement terminée, ou l'ombre ne correspondra pas au mesh.

Voir aussi

CreateMesh()

OS Supportés

Tous

118.11 CreateLine3D

Syntaxe

```
Resultat =
    CreateLine3D(#Mesh, X, Y,
    Z, Couleur, X2, Y2, Z2,
    Couleur2)
```

Description

Crée un mesh ligne 3D.

Arguments

#Mesh Le numéro d'identification du nouveau mesh.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Z Les coordonnées du premier point de la ligne, en unité Monde.

Couleur La couleur du premier point.
La fonction RGB() peut être utilisée pour obtenir une valeur valide.

X2, Y2, Z2 Les coordonnées du second point de la ligne, en unité Monde.

Couleur2 La couleur du second point.
Si elle est différente de la couleur du premier point alors un gradient sera créé entre les deux points.
La fonction RGB() peut être utilisée pour obtenir une valeur valide.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Mesh alors la valeur de retour est le numéro d'identification, généré automatiquement.

Remarques

La ligne est un objet fil de fer qui peut être utilisé pour faciliter le débogage. Pour changer la position de la ligne, il suffit de la créer à nouveau.

Exemple

```
1   InitEngine3D()
2   InitSprite()
3
4   OpenWindow(0, 0, 0, 640,
     480, "Exemple de Ligne
     3D", #PB_Window_SystemMenu
     |
     #PB_Window_ScreenCentered)
```

```
5   OpenWindowedScreen(WindowID(0),
     0, 0, 640, 480, 0, 0, 0)
6
7   ; Lumière
8   CreateLight(#PB_Any,
     RGB(25, 25, 180), -5, 10,
     5, #PB_Light_Point)
9
10  ; Camera
11  CreateCamera(0, 0, 0, 100,
     100)
12  MoveCamera(0, 2, 1, 3,
     #PB_Absolute | #PB_Local)
13  CameraLookAt(0, 0, 0, 0)
14
15  ; Créer la ligne et
     l'attache à la scène
16  CreateLine3D(0, 0, 0, 0,
     RGB(255, 0, 0), 1, 1, 1,
     RGB(0, 0, 255))
17  CreateEntity(0, MeshID(0),
     #PB_Material_None)
18
19  Repeat
20    RenderWorld()
21    FlipBuffers()
22  Until WaitWindowEvent(1) =
     #PB_Event_CloseWindow
```

Voir aussi

FreeMesh() , CreateSphere() , CreateMesh() , CreateCube() , CreatePlane() , CreateCylinder()

OS Supportés

Tous

118.12 CreateCube

Syntaxe

```
Resultat = CreateCube(#Mesh,
    Taille)
```

Description

Crée un mesh cube.

Arguments

#Mesh Le numéro d'identification du nouveau mesh.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Taille Taille du cube dans l'unité du monde.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Mesh alors la valeur de retour est le numéro d'identification, généré automatiquement.

Exemple

```
InitEngine3D()
InitSprite()

OpenWindow(0, 0, 0, 640, 480, "Exemple de Cube 3D", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
OpenWindowedScreen(WindowID(0), 0, 0, 640, 480, 0, 0, 0)

; Lumière
CreateLight(#PB_Any, RGB(25, 25, 180), -5, 10, 5, #PB_Light_Point)

; Camera
CreateCamera(0, 0, 0, 100, 100)
MoveCamera(0, 2, 1, 3, #PB_Absolute | #PB_Local)
CameraLookAt(0, 0, 0, 0)

; Créer le cube et l'attache à la scène
CreateCube(0, 1)
CreateEntity(0, MeshID(0), #PB_Material_None)

Repeat
  RenderWorld()
  FlipBuffers()
Until WaitWindowEvent(1) = #PB_Event_CloseWindow
```

Voir aussi

FreeMesh() , CreateSphere() , CreateMesh() , CreateCylinder() , CreatePlane() , CreateLine3D()

OS Supportés

Tous

118.13 CreateSphere

Syntaxe

```
Resultat =
    CreateSphere(#Mesh,
    Rayon.f [NbSegments,
    NbAnneaux])
```

Description

Crée un mesh sphère.

Arguments

#Mesh Le numéro d'identification du nouveau mesh.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Rayon.f Rayon de la sphère dans l'unité du monde.

NbSegments (optionnel) Nombre de segments de la sphère (16 par défaut). Les segments sont les lignes verticales de la sphère. Plus le nombre de segments est grand et plus la sphère sera réaliste mais la vitesse de rendu en sera affectée s'ils sont en trop grand nombre.

NbAnneaux (optionnel) Nombre d'anneaux de la sphère (16 par défaut). Les anneaux sont des lignes horizontales de la sphère. Plus le nombre d'anneaux est grand et plus la sphère sera réaliste mais la vitesse de rendu en sera affectée s'ils sont en trop grand nombre.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Mesh alors la valeur de retour est le numéro d'identification, généré automatiquement.

Exemple

```
1  InitEngine3D()
2  InitSprite()
3
4  OpenWindow(0, 0, 0, 640,
    480, "Exemple de Sphère
    3D", #PB_Window_SystemMenu
    |
    #PB_Window_ScreenCentered)
5  OpenWindowedScreen(WindowID(0),
    0, 0, 640, 480, 0, 0, 0)
6
```

```
7   ; Lumière
8   CreateLight(#PB_Any,
     RGB(25, 25, 180), -5, 10,
     5, #PB_Light_Point)
9
10  ; Camera
11  CreateCamera(0, 0, 0, 100,
     100)
12  MoveCamera(0, 2, 1, 3,
     #PB_Absolute | #PB_Local)
13  CameraLookAt(0, 0, 0, 0)
14
15  ; Créer la sphère et
     l'attache à la scène
16  CreateSphere(0, 1)
17  CreateEntity(0, MeshID(0),
     #PB_Material_None)
18
19  Repeat
20    RenderWorld()
21    FlipBuffers()
22  Until WaitWindowEvent(1) =
     #PB_Event_CloseWindow
```

Voir aussi

FreeMesh() , CreateCylinder() , CreateMesh() , CreateCube() , CreatePlane() , CreateLine3D()

OS Supportés

Tous

118.14 CreateTube

Syntaxe

```
Resultat = CreateTube(#Mesh,
    RayonExterne.f,
    RayonInterne.f, Hauteur.f
    [, NbSegmentsBase,
    NbSegmentsHauteur)
```

Description

Crée un mesh tube.

Arguments

#Mesh Le numéro d'identification du nouveau mesh.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

RayonExterne.f Rayon externe du tube dans l'unité du monde.

RayonInterne.f Rayon interne du tube dans l'unité du monde.

Hauteur.f Hauteur du tube dans l'unité du monde.

NbBaseSegments (optionnel) Nombre de segments de la base du tube (16 par défaut).

NbHeightSegments (optionnel) Nombre de segments dans la hauteur du tube (1 par défaut).

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Mesh alors la valeur de retour est le numéro d'identification, généré automatiquement.

Exemple

```
1   InitEngine3D()
2   InitSprite()
3
4   OpenWindow(0, 0, 0, 640,
     480, "Exemple de Tube 3D",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
5   OpenWindowedScreen(WindowID(0),
     0, 0, 640, 480, 0, 0, 0)
6
7   ; Lumière
8   CreateLight(#PB_Any,
     RGB(25, 25, 180), -5, 10,
     5, #PB_Light_Point)
9
10  ; Caméra
11  CreateCamera(0, 0, 0, 100,
     100)
12  MoveCamera(0, 2, 4, 3,
     #PB_Absolute | #PB_Local)
13  CameraLookAt(0, 0, 0, 0)
14
15  ; Création du tube et
     application dans la scène
16  CreateTube(0, 0.5, 0.4, 2)
17  CreateEntity(0, MeshID(0),
     #PB_Material_None)
18
19  Repeat
20    RenderWorld()
21    FlipBuffers()
22  Until WaitWindowEvent(1) =
     #PB_Event_CloseWindow
```

Voir aussi

FreeMesh() , CreateCylinder() , CreateMesh() , CreateCube() , CreatePlane() , CreateLine3D()

OS Supportés

Tous

118.15 CreateTorus

Syntaxe

```
Resultat = CreateTorus(#Mesh,
    Rayon.f, RayonSection.f,
    Hauteur.f [,
    NbSegmentsSection,
    NbSegmentsCercle)
```

Description

Crée un mesh tore.

Arguments

#Mesh Le numéro d'identification du nouveau mesh.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Rayon.f Rayon du tore dans l'unité du monde.

RayonSection.f Rayon de la section du tore dans l'unité du monde.

Hauteur.f Hauteur du tore dans l'unité du monde.

NbSegmentsSection (optionnel) Nombre de segments utilisés pour la section du tore (16 par défaut).

NbSegmentsCercle (optionnel) Nombre de segments utilisés pour la cercle du tore (16 par défaut).

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Mesh alors la valeur de retour est le numéro d'identification, généré automatiquement.

Exemple

```
InitEngine3D()
InitSprite()

OpenWindow(0, 0, 0, 640, 480, "Exemple de tore 3D", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
OpenWindowedScreen(WindowID(0), 0, 0, 640, 480, 0, 0, 0)

; Lumière
CreateLight(#PB_Any, RGB(25, 25, 180), -5, 10, 5, #PB_Light_Point)

; Caméra
CreateCamera(0, 0, 0, 100, 100)
MoveCamera(0, 2, 4, 3, #PB_Absolute | #PB_Local)
CameraLookAt(0, 0, 0, 0)

; Création du tore et application à la scène
CreateTorus(0, 1, 0.3)
CreateEntity(0, MeshID(0), #PB_Material_None)

Repeat
  RenderWorld()
  FlipBuffers()
Until WaitWindowEvent(1) = #PB_Event_CloseWindow
```

Voir aussi

FreeMesh() , CreateCylinder() , CreateMesh() , CreateCube() , CreatePlane() , CreateLine3D()

OS Supportés

Tous

118.16 CreateCapsule

Syntaxe

```
Resultat = CreateCapsule(#Mesh, Rayon.f, Hauteur.f [, NbAnneaux, NbSegments, NbSegmentsHauteur)
```

Description

Crée un mesh capsule.

Arguments

#Mesh Le numéro d'identification du nouveau mesh.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Rayon.f Rayon de la capsule dans l'unité du monde.

Hauteur.f Hauteur de la capsule dans l'unité du monde.

NbAnneaux (optionnel) Nombre d'anneaux utilisés pour créer la capsule (8 par défaut).

NbSegments (optionnel) Nombre de segments utilisés pour créer la capsule (16 par défaut).

NbSegmentsHauteur (optionnel) Nombre de segments dans la hauteur utilisés pour créer la capsule (1 par défaut).

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Mesh alors la valeur de retour est le numéro d'identification, généré automatiquement.

Exemple

```
1   InitEngine3D()
2   InitSprite()
3
4   OpenWindow(0, 0, 0, 640,
     480, "Exemple de Capsule
     3D", #PB_Window_SystemMenu
     |
     #PB_Window_ScreenCentered)
5   OpenWindowedScreen(WindowID(0),
     0, 0, 640, 480, 0, 0, 0)
6
7   ; Lumière
8   CreateLight(#PB_Any,
     RGB(25, 25, 180), -5, 10,
     5, #PB_Light_Point)
9
10  ; Caméra
11  CreateCamera(0, 0, 0, 100,
     100)
12  MoveCamera(0, 2, 0, 5,
     #PB_Absolute | #PB_Local)
13  CameraLookAt(0, 0, 0, 0)
14
15  ; Création de la capsule et
     application à la scène
16  CreateCapsule(0, 1, 1)
```

```
17    CreateEntity(0, MeshID(0),
       #PB_Material_None)
18
19    Repeat
20      RenderWorld()
21      FlipBuffers()
22    Until WaitWindowEvent(1) =
       #PB_Event_CloseWindow
```

Voir aussi

FreeMesh() , CreateCylinder() , CreateMesh() , CreateCube() , CreatePlane() , CreateLine3D()

OS Supportés

Tous

118.17 CreateIcoSphere

Syntaxe

```
Resultat =
    CreateIcoSphere(#Mesh,
    Rayon.f [, Iterations)
```

Description

Crée un mesh sphère icosaédrique

Arguments

#Mesh Le numéro d'identification du nouveau mesh.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Rayon.f Rayon de la sphère icosaédrique dans l'unité du monde.

Iterations (optionnel) Nombre d'itérations utilisés pour créer la sphère icosaédrique (2 par défaut).

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Mesh alors la valeur de retour est le numéro d'identification, généré automatiquement.

Exemple

```
InitEngine3D()
InitSprite()

OpenWindow(0, 0, 0, 640, 480, "Exemple de sphère icosaédrique", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
OpenWindowedScreen(WindowID(0), 0, 0, 640, 480, 0, 0, 0)

; Lumière
CreateLight(#PB_Any, RGB(25, 25, 180), -5, 10, 5, #PB_Light_Point)

; Caméra
CreateCamera(0, 0, 0, 100, 100)
MoveCamera(0, 2, 0, 5, #PB_Absolute | #PB_Local)
CameraLookAt(0, 0, 0, 0)

; Création de la sphère icosaédrique et application à la scène
CreateIcoSphere(0, 1)
CreateEntity(0, MeshID(0), #PB_Material_None)

Repeat
  RenderWorld()
  FlipBuffers()
Until WaitWindowEvent(1) = #PB_Event_CloseWindow
```

Voir aussi

FreeMesh() , CreateCylinder() , CreateMesh() , CreateCube() , CreatePlane() , CreateLine3D()

OS Supportés

Tous

118.18 CreateCone

Syntaxe

```
Resultat = CreateCone(#Mesh, Rayon.f, Hauteur.f [, NbSegmentsBase, NbSegmentsHauteur])
```

Description

Crée un mesh cône.

Arguments

#Mesh Le numéro d'identification du nouveau mesh.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Rayon.f Rayon du cône dans l'unité du monde.

Hauteur.f Hauteur du cône dans l'unité du monde.

NbSegmentsBase (optionnel) Nombre de segments utilisé pour la base du cône (16 par défaut).

NbSegmentsHauteur (optionnel) Nombre de segments utilisé pour la hauteur du cône (1 par défaut).

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Mesh alors la valeur de retour est le numéro d'identification, généré automatiquement.

Exemple

```
1   InitEngine3D()
2   InitSprite()
3
4   OpenWindow(0, 0, 0, 640,
     480, "Exemple Cône ",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
5   OpenWindowedScreen(WindowID(0),
     0, 0, 640, 480, 0, 0, 0)
6
7   ; Lumière
8   CreateLight(#PB_Any,
     RGB(25, 25, 180), -5, 10,
     5, #PB_Light_Point)
9
10  ; Camera
11  CreateCamera(0, 0, 0, 100,
     100)
12  MoveCamera(0, 2, 1, 3,
     #PB_Absolute | #PB_Local)
13  CameraLookAt(0, 0, 0, 0)
14
15  ; Création du cône et
     inclusion dans la scène
16  CreateCone(0, 0.5, 1)
17  CreateEntity(0, MeshID(0),
     #PB_Material_None)
18
19  Repeat
20    RenderWorld()
```

```
21      FlipBuffers()
22    Until WaitWindowEvent(1) =
       #PB_Event_CloseWindow
```

Voir aussi

FreeMesh() , CreateSphere() , CreateCylinder() , CreateMesh() , CreateCube() , CreatePlane() , CreateLine3D()

OS Supportés

Tous

118.19 CreateCylinder

Syntaxe

```
Resultat =
    CreateCylinder(#Mesh,
    Rayon.f, Hauteur.f [,
    NbSegmentsBase,
    NbSegmentsHauteur,
    Fermeture])
```

Description

Crée un mesh cylindre.

Arguments

#Mesh Le numéro d'identification du nouveau mesh.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Rayon.f Rayon du cylindre dans l'unité du monde.

Hauteur.f Hauteur du cylindre dans l'unité du monde.

NbSegmentsBase (optionnel) Nombre de segments utilisé pour la base du cylindre (16 par défaut).

NbSegmentsHauteur (optionnel) Nombre de segments utilisé pour la hauteur du cylindre (1 par défaut).

Fermeture (optionnel) `#True : Cylindre fermé en haut et en bas (par défaut)`
`#False : Cylindre ouvert en haut et en bas (tuyau)`

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Mesh alors la valeur de retour est le numéro d'identification, généré automatiquement.

Exemple

```
InitEngine3D()
InitSprite()

OpenWindow(0, 0, 0, 640, 480, "Exemple de Cylindre 3D", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
OpenWindowedScreen(WindowID(0), 0, 0, 640, 480, 0, 0, 0)

; Lumière
CreateLight(#PB_Any, RGB(25, 25, 180), -5, 10, 5, #PB_Light_Point)

; Camera
CreateCamera(0, 0, 0, 100, 100)
MoveCamera(0, 2, 1, 3, #PB_Absolute | #PB_Local)
CameraLookAt(0, 0, 0, 0)

; Créer le cylindre et l'attache à la scène
CreateCylinder(0, 0.5, 1)
CreateEntity(0, MeshID(0), #PB_Material_None)

Repeat
  RenderWorld()
  FlipBuffers()
Until WaitWindowEvent(1) = #PB_Event_CloseWindow
```

Voir aussi

FreeMesh() , CreateSphere() , CreateMesh() , CreateCube() , CreatePlane() , CreateLine3D() , CreateCone()

OS Supportés

Tous

118.20 CreatePlane

Syntaxe

```
Resultat = CreatePlane(#Mesh,
    TailleParcelleX,
    TailleParcelleZ,
    NbParcelleX, NbParcelleZ,
    NbRepetitionTextureX,
    NbRepetitionTextureZ)
```

Description

Crée un mesh plan.

Arguments

#Mesh Le numéro d'identification du nouveau mesh.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

TailleParcelleX Taille en 'X' d'une Parcelle du plan, dans l'unité monde. La parcelle est le composant de base d'un plan. Un plan peut être composé de nombreuses parcelles pour le rendre plus grand et permettre une déformation.

TailleParcelleZ Taille en 'Z' d'une parcelle du plan, dans l'unité monde. La parcelle est le composant de base d'un plan. Un plan peut être composé de nombreuses parcelles pour le rendre plus grand et permettre une déformation.

NbParcelleX Nombre de parcelles utilisées dans l'axe X du plan.

NbParcelleZ Nombre de parcelles utilisées dans l'axe Z du plan.

NbRepetitionTextureX Nombre de fois que la texture sera répétée sur l'axe X. Pour appliquer la texture entière sur tout l'axe X, il suffit d'utiliser 1.

NbRepetitionTextureZ Nombre de fois que la texture sera répétée sur l'axe Z. Pour appliquer la texture entière sur tout l'axe Z, il suffit d'utiliser 1.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Mesh alors la valeur de retour est le numéro d'identification, généré automatiquement.

Exemple

```
InitEngine3D()
InitSprite()

OpenWindow(0, 0, 0, 640, 480, "Exemple de plan en 3D", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
OpenWindowedScreen(WindowID(0), 0, 0, 640, 480, 0, 0, 0)

; Lumière
CreateLight(#PB_Any, RGB(25, 25, 180), -5, 10, 5, #PB_Light_Point)

; Camera
CreateCamera(0, 0, 0, 100, 100)
MoveCamera(0, 2, 1, 3, #PB_Absolute | #PB_Local)
CameraLookAt(0, 0, 0, 0)

; Créer le plan et l'attache à la scène
CreatePlane(0, 2, 2, 1, 1, 0, 0)
CreateEntity(0, MeshID(0), #PB_Material_None)

Repeat
  RenderWorld()
  FlipBuffers()
Until WaitWindowEvent(1) = #PB_Event_CloseWindow
```

Voir aussi

FreeMesh() , CreateSphere() , CreateMesh() , CreateCube() , CreatePlane() , CreateLine3D()

OS Supportés

Tous

118.21 AddSubMesh

Syntaxe

```
AddSubMesh([Type])
```

Description

Ajoute un nouveau sous-mesh dans le mesh courant créé avec CreateMesh() .

Arguments

Type (optionnel) Le type du nouveau sous-mesh. Peut être une des valeurs suivantes :

```
#PB_Mesh_TriangleList : Le
 sous-mesh sera composé
 d'une liste de triangles
 (par défaut).
#PB_Mesh_TriangleStrip: Le
 sous-mesh sera composé
 d'une liste de triangles
 connectés (Les sommets
 sont partagés).
#PB_Mesh_TriangleFan  : Le
 sous-mesh sera composé
 d'une liste de triangles
 qui partagent le même
 sommet central .
#PB_Mesh_PointList    : Le
 sous-mesh sera composé
 d'une liste de points.
#PB_Mesh_LineList     : Le
 sous-mesh sera composé
 d'une liste de lignes.
#PB_Mesh_LineStrip    : Le
 sous-mesh sera composé
 d'une liste de lignes
 connectées (Les sommets
 sont partagés).
```

Valeur de retour

Aucune.

Remarques

Un mesh peut avoir n'importe quel nombre de sous-meshes.
La position d'un sous-mesh est relative à la position du mesh.
Une fois qu'un sous-mesh est créé, utiliser les commandes suivantes pour le construire : MeshVertexPosition() , MeshFace() et MeshIndex() .

Voir aussi

FreeMesh() , CreateMesh() , MeshVertexPosition() , MeshFace()

OS Supportés

Tous

118.22 MeshIndexCount

Syntaxe

```
Resultat =
    MeshIndexCount(#Mesh [,
    SousMesh])
```

Description

Retourne le nombre des index dans le mesh.

Arguments

Mesh Le mesh à utiliser.

SousMesh (optionnel) S'il est spécifié, il renverra le nombre d'index dans le SousMesh.
L'indice du premier SousMesh est 0 (mesh principal).

Valeur de retour

Renvoie le nombre des index dans le mesh, ou zéro si le mesh ou le SousMesh n'existe pas.

Voir aussi

CreateMesh() , LoadMesh() , MeshVertexCount()

OS Supportés

Tous

118.23 MeshVertexCount

Syntaxe

```
Resultat =
    MeshVertexCount(#Mesh [,
    SousMesh])
```

Description

Renvoie le nombre de sommets du mesh.

Arguments

Mesh Le mesh à utiliser.

SousMesh (optionnel) S'il est spécifié, renvoie le nombre de sommets du sous-mesh spécifié.
L'indice du premier submesh est 0 (mesh principal).

Valeur de retour

Renvoie le nombre de sommets du mesh ou zéro si le mesh, ou le sous-mesh, n'existe pas.

Voir aussi

CreateMesh() , LoadMesh() , MeshIndexCount()

OS Supportés

Tous

118.24 UpdateMeshBoundingBox

Syntaxe

```
UpdateMeshBoundingBox(#Mesh)
```

Description

Met à jour la boîte englobante d'un mesh.

Arguments

Mesh Le mesh à utiliser.

Valeur de retour

Aucune.

Remarques

Si un mesh a été modifié manuellement, sa boîte englobante doit être recalculée, en particulier si le mesh est utilisé pour les collisions. Le cadre de sélection est la plus petite boîte possible capable de contenir tout le mesh.

Voir aussi

CreateMesh()

OS Supportés

Tous

118.25 UpdateMesh

Syntaxe

```
UpdateMesh(#Mesh, SousMesh)
```

Description

Démarre la mise à jour du mesh

Arguments

Mesh Le mesh à utiliser.

SousMesh L'indice du sous-mesh à modifier.
Le premier indice est 0 (mesh principal).

Valeur de retour

Aucune.

Remarques

Mise à jour du mesh pour modifier en temps réel ses sommets et d'autres valeurs.
Le mesh doit être créé avec l'option #PB_Mesh_Dynamic.
Une fois les modifications terminées, FinishMesh() doit être appelé.
Le mesh peut utiliser les commandes suivantes pour modifier ses propriétés : MeshIndex() , MeshFace() , MeshVertexPosition() , MeshVertexNormal() , MeshVertexTangent() , MeshVertexColor() et MeshVertexTextureCoordinate() .

Voir aussi

CreateMesh() , MeshIndex() , MeshFace() , MeshVertexPosition() , MeshVertexNormal() , MeshVertexTangent() , MeshVertexColor() , MeshVertexTextureCoordinate()

OS Supportés

Tous

118.26 MeshIndex

Syntaxe

```
MeshIndex(Index)
```

Description

Ajoute ou met à jour un seul sommet du mesh en cours de création avec CreateMesh() ou mis à jour avec UpdateMesh() .

Arguments

Index L'index du sommet.

Valeur de retour

Aucune.

Remarques

Se comporte comme la commande MeshFace() mais avec un nombre arbitraire de sommets. Lorsque vous utilisez le mode de #PB_Mesh_LineList ou #PB_Mesh_LineStrip, il y a seulement deux sommets par lignes, donc MeshIndex() doit être utilisé dans ce cas.

Voir aussi

UpdateMesh() , MeshIndex() , MeshFace() , MeshVertexPosition() , MeshVertexNormal() , MeshVertexTangent() , MeshVertexColor() , MeshVertexTextureCoordinate()

OS Supportés

Tous

118.27 MeshRadius

Syntaxe

```
Resultat = MeshRadius(#Mesh)
```

Description

Renvoie le rayon de la plus petite sphère capable de contenir le mesh.

Arguments

Mesh Le mesh à utiliser.

Valeur de retour

Renvoie le rayon du mesh.

Voir aussi

CreateMesh()

OS Supportés

Tous

118.28 MeshVertex

Syntaxe

```
MeshVertex(X, Y, Z, U.f, V.f,
    Couleur [, NormalX,
    NormalY, NormalZ])
```

Description

Ajouter un sommet au mesh en cours créé précédemment avec CreateMesh() .

Arguments

X, Y, Z La position du nouveau sommet.

U.f La coordonnée U.
Cette valeur est la coordonnée horizontale 'X' dans la texture où le sommet doit être mappé.
Cette valeur est généralement comprise entre 0 et 1, 0 étant l'origine et 1 l'extrémité (voir MeshVertexTextureCoordinate() pour plus d'informations).

V.f La coordonnée V.
Cette valeur est la coordonnée horizontale 'Y' dans la texture où le sommet doit être mappé.
Cette valeur est généralement comprise entre 0 et 1, 0 étant l'origine et 1 l'extrémité (voir MeshVertexTextureCoordinate() pour plus d'informations).

Couleur Couleur du vertex.
Cette couleur peut être au format RGB ou RGBA (voir MeshVertexColor() pour plus d'informations).

NormalX, NormalY, NormalZ Le vecteur normal (voir MeshVertexNormal() pour plus d'informations).

Valeur de retour

Aucune.

Remarques

Des attributs spécifiques au sommet nouvellement créé peuvent être ajoutés avec MeshVertexTangent() .
Pour créer une nouvelle face, utilisez MeshFace() .

Voir aussi

CreateMesh() , MeshFace() , MeshVertexNormal() , MeshVertexTangent() , MeshVertexColor() , MeshVertexTextureCoordinate()

OS Supportés

Tous

118.29 MeshVertexPosition

Syntaxe

```
MeshVertexPosition(X, Y, Z)
```

Description

Ajoute un nouveau sommet au mesh en cours, précédemment créé avec CreateMesh() .

Arguments

X, Y, Z La position du nouveau sommet.

Valeur de retour

Aucune.

Remarques

Pour ajouter des attributs spécifiques au sommet nouvellement créé, utiliser les commandes suivantes : MeshVertexNormal() , MeshVertexTangent() , MeshVertexColor() et MeshVertexTextureCoordinate() .
Si plusieurs attributs doivent être spécifiés, vous pouvez utiliser MeshVertex() .
Pour créer une nouvelle face, utiliser MeshFace() .

Voir aussi

CreateMesh() , MeshFace() , MeshVertexNormal() , MeshVertexTangent() , MeshVertexColor() , MeshVertexTextureCoordinate() , , MeshVertex()

OS Supportés

Tous

118.30 MeshVertexNormal

Syntaxe

```
MeshVertexNormal(X, Y, Z)
```

Description

Définit l'information concernant la 'normale' du sommet en cours, précédemment ajouté avec MeshVertexPosition() ou MeshVertex() .

Arguments

X, Y, Z La valeur du vecteur normale.

Valeur de retour

Aucune.

Remarques

Le vecteur 'normal' (perpendiculaire au plan) est utilisé pour calculer l'éclairage sur un objet. Pour calculer automatiquement le vecteur 'normal' une fois que le mesh est créé, utiliser NormalizeMesh() .

Voir aussi

CreateMesh() , MeshVertexPosition() , MeshVertexColor() , MeshVertexTextureCoordinate() , NormalizeMesh() , MeshVertex()

OS Supportés

Tous

118.31 MeshVertexTangent

Syntaxe

```
MeshVertexTangent(X, Y, Z)
```

Description

Définit la tangente au sommet en cours, précédemment ajouté avec MeshVertexPosition() .

Arguments

X, Y, Z Le vecteur tangent.

Valeur de retour

Aucune.

Remarques

Le vecteur tangent est principalement utilisé dans les scripts de shader. Pour calculer automatiquement le vecteur tangent une fois que le mesh est créé, utilisez BuildMeshTangents() .

Voir aussi

CreateMesh() , MeshVertexPosition() , MeshVertexNormal() , MeshVertexColor() , MeshVertexTextureCoordinate() , BuildMeshTangents() , MeshVertex()

OS Supportés

Tous

118.32 MeshVertexColor

Syntaxe

```
MeshVertexColor(Couleur)
```

Description

Définit des informations de couleur pour du sommet en cours, précédemment ajouté avec MeshVertexPosition() ou MeshVertex() .

Arguments

Couleur Couleur du sommet.
Cette couleur peut être au format RGB ou RGBA .

Valeur de retour

Aucune.

Remarques

Pour avoir un effet, le matériau associé au mesh doit être défini avec SetMaterialColor (#Material, #PB_Material_AmbientColor, -1) et AmbientColor() () défini sur une valeur positive.
Le vertex doit avoir été créé avant d'utiliser cette fonction.

Voir aussi

CreateMesh() , MeshVertexPosition() , MeshVertexNormal() , MeshVertexTangent() , MeshVertexTextureCoordinate() , MeshVertex()

OS Supportés

Tous

118.33 MeshVertexTextureCoordinate

Syntaxe

```
MeshVertexTextureCoordinate(U.f,
    [V.f [, W.f]])
```

Description

Définit l'information UVW pour le sommet en cours, précédemment ajouté avec MeshVertexPosition() ou MeshVertex() .

Arguments

U.f La valeur U.
Cette valeur est la position X dans la texture où le sommet devrait être. Cette valeur est généralement comprise entre 0 et 1, où 0 est l'origine en X et 1 est la fin en X de la texture.

V.f (optionnel) La valeur V.
Cette valeur est la position Y dans la texture où le sommet devrait être. Cette valeur est généralement comprise entre 0 et 1, où 0 est l'origine en Y et 1 est la fin en Y de la texture.

W.f (optionnel) La valeur W (pour les textures cubiques).
Cette valeur est la position Z dans la texture où le sommet devrait être mappé. Cette valeur est généralement comprise entre 0 et 1, où 0 est l'origine en Z de la texture et 1 est la fin en Z de la texture.

Valeur de retour

Aucune.

Remarques

Les coordonnées UVW sont utilisées pour appliquer la texture sur le mesh.

Voir aussi

CreateMesh() , MeshVertexPosition() , MeshVertexNormal() , MeshVertexTangent() , MeshVertexColor() , MeshVertex()

OS Supportés

Tous

118.34 MeshFace

Syntaxe

```
MeshFace(Sommet1 , Sommet2 ,
    Sommet3 [, Sommet4]))
```

Description

Ajoute ou met à jour une nouvelle face du mesh courant précédemment créé avec CreateMesh() .

Arguments

Sommet1 Le premier indice de sommet utilisé pour créer la face.

Sommet2 Le deuxième indice de sommet utilisé pour créer la face.

Sommet3 Le troisième indice de sommet utilisé pour créer la face.

Sommet4 (optionnel) Le quatrième indice de sommet utilisé pour créer la face, ce qui se traduira par un carré (quad).

Valeur de retour

Aucune.

Remarques

Les sommets spécifiés doivent exister.
Le premier indice de sommet débute à 0.
La face créée est un triangle ou un carré.
MeshIndex() peut être utilisé si le nombre de sommets est supérieur à quatre.

Voir aussi

CreateMesh() , MeshVertexPosition() , MeshVertex()

OS Supportés

Tous

118.35 FinishMesh

Syntaxe

```
FinishMesh(Mode)
```

Description

Termine la création du mesh courant démarré avec CreateMesh() .

Arguments

```
Mode    - #True : Le mesh
    est converti en un mesh
    statique et ne sera plus
    modifiable.
  - #False: Le mesh sera
    toujours modifiable avec
    UpdateMesh()
.
              Les meshes
    statiques sont plus
    rapides que les meshes
    dynamiques.
```

Valeur de retour

Aucune.

Voir aussi

CreateMesh()

OS Supportés

Tous

118.36 NormalizeMesh

Syntaxe

```
NormalizeMesh(#Mesh [,
    SousMesh])
```

Description

Normalise le mesh ou le sous-mesh.

Arguments

Mesh Le mesh à utiliser.

SousMesh (optionnel) S'il est spécifié, il normalise le sous-mesh.
Le premier indice de sous-mesh est 0 (mesh principal).

Valeur de retour

Aucune.

Remarques

Il calcule automatiquement le vecteur 'normal' pour tous les sommets du mesh ou du sous-mesh spécifié.
Les meshes dynamiques, créés avec l'option #PB_Mesh_Dynamic, ne sont pas supportés.

Voir aussi

CreateMesh() , MeshVertexNormal()

OS Supportés

Tous

118.37 BuildMeshTangents

Syntaxe

```
BuildMeshTangents(#Mesh)
```

Description

Calcule automatiquement les vecteurs tangents à tous les sommets du mesh spécifié.
(Les meshes dynamiques, créés avec l'option #PB_Mesh_Dynamic, ne sont pas supportés.)

Arguments

Mesh Le mesh à utiliser.

Valeur de retour

Aucune.

Voir aussi

CreateMesh() , MeshVertexTangent()

OS Supportés

Tous

118.38 AddMeshManualLOD

Syntaxe

```
AddMeshManualLOD(#Mesh,
    #MeshLOD, Distance.f)
```

Description

Ajoute un nouveau niveau de détail (LOD) au mesh.

Arguments

Mesh Le mesh à utiliser.

MeshLOD Le mesh à utiliser lorsque la distance de la caméra est atteinte.

Distance.f La distance minimale de la caméra où le #MeshLOD doit être utilisé à la place de #Mesh.

Valeur de retour

Aucune.

Remarques

Le #Mesh sera automatiquement remplacé par #MeshLOD (qui est souvent une version simplifiée de #Mesh, avec moins de détails) lorsqu'il est affiché au-dessus de la distance spécifiée de la caméra. Plusieurs mesh LOD peuvent être utilisées pour le même #Mesh en fonction de la distance.

Voir aussi

CreateMesh() , BuildMeshLOD()

OS Supportés

Tous

118.39 BuildMeshLOD

Syntaxe

```
BuildMeshLOD(#Mesh, NbLOD,
    Distance.f,
    ValeurReduction.f)
```

Description

Construit automatiquement un ou plusieurs niveaux de détail (LOD) pour le mesh.

Arguments

Mesh Le mesh à utiliser.

NbLOD Nombre de LOD nécessaires pour ce mesh.

Distance Distance minimale par rapport à la caméra où le premier mesh LOD sera utilisé à la place de #Mesh.
Pour les meshs LOD suivants, la distance sera calculée à l'aide de la formule suivante : 'Distance / SqrF (1-ValeurReduction)'.

ValeurReduction.f La réduction à appliquer, entre 0 (pas de réduction) et 1 (réduction de 100%).
Exemple : CreateMeshLodLevels(#Mesh, 3, 100, 0.75)
- La première réduction du mesh d'origine commence à partir de 100 unités de la caméra, nombre de sommets divisé par 2 (réduction de 75%).
- La deuxième réduction du mesh dorigine commence à partir de 200 unités

de la caméra, nombre de sommets divisé par 16.
- La troisième réduction du mesh dorigine commence à partir de 400 unités de la caméra, nombre de sommets divisé par 64.

Valeur de retour

Aucune.

Remarques

Le #Mesh sera automatiquement remplacé par un maillage moins complexe lorsqu'il sera affiché au-dessus de la distance spécifiée par rapport à la caméra.
Si des meshs LOD plus précis sont requis, vous pouvez utiliser AddMeshManualLOD() .

Voir aussi

CreateMesh() , AddMeshManualLOD()

OS Supportés

Tous

118.40 SaveMesh

Syntaxe

```
SaveMesh(#Mesh, Fichier$)
```

Description

Enregistre le mesh.

Arguments

Mesh Le mesh à sauvegarder.

Fichier$ Spécifie le nom de fichier et le chemin d'accès au nouveau fichier de mesh. Si le nom de fichier ne comporte pas de chemin d'accès complet, il est interprété par rapport au dossier courant .

Valeur de retour

Aucune.

Remarques

Le mesh sauvegardé peut être rechargé avec la commande LoadMesh() .

Voir aussi

CreateMesh() , LoadMesh()

OS Supportés

Tous

118.41 SetMeshMaterial

Syntaxe

```
SetMeshMaterial(#Mesh,
    MateriauID [, SousMesh])
```

Description

Définit le matériau par défaut du mesh.

Arguments

Mesh Le mesh à utiliser.

MateriauID Spécifie le matériau à utiliser par défaut pour le mesh. Pour obtenir un 'MateriauID' valide, utiliser MaterialID() .

SousMesh (optionnel) S'il est spécifié, le matériau ne sera appliqué que sur le sous-mesh. Le premier indice de sous-mesh est 0 (mesh principal).

Valeur de retour

Aucune.

Voir aussi

CreateMesh() , LoadMesh()

OS Supportés

Tous

118.42 SubMeshCount

Syntaxe

```
Resultat = SubMeshCount(#Mesh)
```

Description

Renvoie le nombre de sous-mesh du mesh.

Valeur de retour

Renvoie le nombre de sous-mesh du mesh.

Voir aussi

CreateMesh() , LoadMesh() , AddSubMesh()

OS Supportés

Tous

118.43 TransformMesh

Syntaxe

```
TransformMesh(#Mesh, X, Y, Z,
    EchelleX, EchelleY,
    EchelleZ, RotationX,
    RotationY, RotationZ [,
    SousMesh])
```

Description

Transforme le mesh en fonction des paramètres indiqués.

Arguments

Mesh Le mesh à transformer.

X Nouvelle position 'X' du mesh, par rapport à son noeud.
Si le paramètre 'SousMesh' est spécifié, il indique la nouvelle position 'X' du sous-mesh par rapport à son parent.

Y Nouvelle position 'Y' du mesh, par rapport à son noeud.
Si le paramètre 'SousMesh' est spécifié, il indique la nouvelle position 'Y' du sous-mesh par rapport à son parent.

Z Nouvelle position 'Z' du mesh, par rapport à son noeud.
Si le paramètre 'SousMesh' est spécifié, il indique la nouvelle position 'Z' du sous-mesh par rapport à son parent.

EchelleX Applique un facteur d'échelle sur l'axe X du mesh.
Si le paramètre 'SousMesh' est spécifié, le facteur d'échelle est appliqué au sous-mesh.

EchelleY Applique un facteur d'échelle sur l'axe Y du mesh.
Si le paramètre 'SousMesh' est spécifié, le facteur d'échelle est appliqué au sous-mesh.

EchelleZ Applique un facteur d'échelle sur l'axe Z du mesh.
Si le paramètre 'SousMesh' est spécifié, le facteur d'échelle est appliqué au sous-mesh.

RotationX Applique une rotation, en degrés, sur l'axe X du mesh.
Si le paramètre 'SousMesh' est spécifié, la rotation est appliquée au sous-mesh.

RotationY Applique une rotation, en degrés, sur l'axe Y du mesh.
Si le paramètre 'SousMesh' est spécifié, la rotation est appliquée au sous-mesh.

RotationZ Applique une rotation, en degrés, sur l'axe Z du mesh.
Si le paramètre 'SousMesh' est spécifié, la rotation est appliquée au sous-mesh.

SousMesh (optionnel) S'il est spécifié, la transformation ne sera appliquée que sur le sous-mesh. Le premier indice de sous-mesh est 0 (mesh principal).

Valeur de retour

Renvoie le nombre de sous-mesh du mesh.

Remarques

Les meshes dynamiques créés avec l'option #PB_Mesh_Dynamic, ne sont pas autorisés.

Voir aussi

CreateMesh() , LoadMesh()

OS Supportés

Tous

Chapitre 119

Mouse

Généralités

PureBasic permet un plein accès aux souris connectées à l'ordinateur. Les souris standards sont supportées jusqu'à trois boutons. Cette bibliothèque est optimisée et utilise des fonctions de très bas niveau particulièrement efficaces pour le développement des jeux.
N'utilisez pas cette bibliothèque pour des applications classiques, dans ce cas, WindowMouseX() , WindowMouseY() et EventType() doivent être utilisées.
Sous Windows, c'est la technologie DirectX qui est utilisée. Une version récente de DirectX 9 doit être installée (voir ici : DirectX 9 runtime installer).

OS Supportés

Tous

119.1 InitMouse

Syntaxe

```
Resultat = InitMouse()
```

Description

Initialise l'environnement Souris.

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle si une souris est disponible, zéro sinon.

Remarques

Vous devez appeler cette fonction avant tout usage des autres commandes de la bibliothèque.
Cette commande tente d'ouvrir DirectX (v3.0 pour la compatibilité NT4.0 ou v7.0 ou plus sinon). Si la fonction échoue (Resultat = 0), cela peut donc provenir de l'absence de DirectX sur votre ordinateur ou d'une version de DirectX trop ancienne. Sous Windows, une version récente de DirectX 9 doit être installé (voir ici : DirectX 9 runtime installer).

Exemple

```
1  ; Initialisation du monde 2D
2  InitSprite()
3  InitMouse()
4
5  ; Ouverture de la fenêtre
6  OpenWindow(0,0,0,800,600,"Souris - Cliquer et utiliser la molette...",#PB_Window_ScreenCentered|#PB_Window_SystemMenu)
7  OpenWindowedScreen(WindowID(0),0,0,800,600,1,0,0)
8
9  ;Création du curseur de la souris
10 LoadSprite(0, #PB_Compiler_Home + "examples/sources/Data/PureBasic.bmp")
     ; Load nice small Logo
11
12 ; Déplacement de la souris
13 MouseLocate(300,200)
14
15 ; Gestion de la fenêtre et de l'écran
16 Repeat
17   Repeat      ; Gestion de la fenêtre
18     Event  = WindowEvent()
19     Select Event
20       Case #PB_Event_CloseWindow
21         End
22     EndSelect
23   Until Event = 0
24
25 ExamineMouse() ; Etat de la souris
26 x = MouseX()   ; Position en x de la souris
27 y = MouseY()   ; Position en y de la souris
28
29 ; Affichage du curseur
```

```
30  DisplaySprite(0,
     x-SpriteWidth(0)/2,
     y-SpriteHeight(0)/2)
31
32  ; Molette de la souris =
     Rotation du curseur
33  tiks = MouseWheel()
34  RotateSprite(0, tiks,
     #PB_Relative)
35
36  ; Affichage de l'état de la
     souris
37  StartDrawing(ScreenOutput())
38  DrawText(0,5, "X=" + Str(x)
     + "   Y=" + Str(y) + "
     DeltaX " +
     Str(MouseDeltaX()) + "
     DeltaY= " +
     Str(MouseDeltaY()))
39  If
     MouseButton(#PB_MouseButton_Left)
     <> 0
40    DrawText(0,30, "Bouton
     Gauche",
     RGB(255,0,0),RGB(255,255,0))
41  Else
42    DrawText(0,30, "Bouton
     Gauche", RGB(255,255,255))
43  EndIf
44  If
     MouseButton(#PB_MouseButton_Middle)
     <> 0
45    DrawText(150,30, "Bouton
     Central",
     RGB(255,0,0),RGB(255,255,0))
46  Else
47    DrawText(150,30, "Bouton
     Central", RGB(255,255,255))
48  EndIf
49  If
     MouseButton(#PB_MouseButton_Right)
     <> 0
50    DrawText(300,30, "Bouton
     Droit",
     RGB(255,0,0),RGB(255,255,0))
51  Else
52    DrawText(300,30, "Bouton
     Droit", RGB(255,255,255))
53  EndIf
54  If tiks <> 0
55    DrawText(450,30,
     "Molette",
     RGB(255,0,0),RGB(255,255,0))
56  Else
57    DrawText(450,30,
     "Molette",
     RGB(255,255,255))
58  EndIf
```

```
59   DrawText(0,60,".: Libérer la souris: Clic sur bouton gauche et droit :.")
60   StopDrawing()
61
62   ; Si clic sur bouton gauche ET droit alors la souris est libérée de l'écran courant
63   If MouseButton(#PB_MouseButton_Left) And MouseButton(#PB_MouseButton_Right)
64     ReleaseMouse(1)
65     oui=6
66     quitter=MessageRequester("Info !", "Voulez-vous quitter le programme ?",#PB_MessageRequester_YesNo)
67     If quitter=oui
68       End
69     EndIf
70   EndIf
71
72   FlipBuffers()
73   ClearScreen(RGB(0,0,0))
74
75   ForEver
76   End
```

Voir aussi

ExamineMouse()

OS Supportés

Tous

119.2 ExamineMouse

Syntaxe

```
Resultat = ExamineMouse()
```

Description

Met à jour l'état de la souris.

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle si l'état de la souris a changé, zéro sinon.

Remarques

Cette commande doit être utilisée avant MouseDeltaX() , MouseDeltaY() , MouseX() , MouseY() ou MouseButton() . Le premier appel à ExamineMouse() capture la souris dans l'écran (surface accélérée) en cours, aussi bien en plein écran (Screen) que dans un écran fenêtré (WindowedScreen).
La souris est à nouveau disponible pour lensemble du système après avoir appelé ReleaseMouse(#True) ou après la fin de ce programme.

Exemple

```
1   ; Initialisation du monde 2D
2   InitSprite()
3   InitMouse()
4
5   ; Ouverture de la fenêtre
6   OpenWindow(0,0,0,800,600,"Souris - Cliquer et utiliser la molette...",#PB_Window_ScreenCentered|#PB_Window_SystemMenu)
7   OpenWindowedScreen(WindowID(0),0,0,800,600,1,0,0)
8
9   ;Création du curseur de la souris
10  LoadSprite(0, #PB_Compiler_Home + "examples/sources/Data/PureBasic.bmp")
      ; Load nice small Logo
11
12  ; Déplacement de la souris
13  MouseLocate(300,200)
14
15  ; Gestion de la fenêtre et de l'écran
16  Repeat
17    Repeat      ; Gestion de la fenêtre
18      Event  = WindowEvent()
19      Select Event
20        Case #PB_Event_CloseWindow
21          End
22      EndSelect
23    Until Event = 0
24
25  ExamineMouse() ; Etat de la souris
26  x = MouseX()   ; Position en x de la souris
27  y = MouseY()   ; Position en y de la souris
28
29  ; Affichage du curseur
```

```
30  DisplaySprite(0,
     x-SpriteWidth(0)/2,
     y-SpriteHeight(0)/2)
31
32  ; Molette de la souris =
     Rotation du curseur
33  tiks = MouseWheel()
34  RotateSprite(0, tiks,
     #PB_Relative)
35
36  ; Affichage de l'état de la
     souris
37  StartDrawing(ScreenOutput())
38  DrawText(0,5, "X=" + Str(x)
     + "   Y=" + Str(y) + "
     DeltaX " +
     Str(MouseDeltaX()) + "
     DeltaY= " +
     Str(MouseDeltaY()))
39  If
     MouseButton(#PB_MouseButton_Left)
     <> 0
40    DrawText(0,30, "Bouton
     Gauche",
     RGB(255,0,0),RGB(255,255,0))
41  Else
42    DrawText(0,30, "Bouton
     Gauche", RGB(255,255,255))
43  EndIf
44  If
     MouseButton(#PB_MouseButton_Middle)
     <> 0
45    DrawText(150,30, "Bouton
     Central",
     RGB(255,0,0),RGB(255,255,0))
46  Else
47    DrawText(150,30, "Bouton
     Central", RGB(255,255,255))
48  EndIf
49  If
     MouseButton(#PB_MouseButton_Right)
     <> 0
50    DrawText(300,30, "Bouton
     Droit",
     RGB(255,0,0),RGB(255,255,0))
51  Else
52    DrawText(300,30, "Bouton
     Droit", RGB(255,255,255))
53  EndIf
54  If tiks <> 0
55    DrawText(450,30,
     "Molette",
     RGB(255,0,0),RGB(255,255,0))
56  Else
57    DrawText(450,30,
     "Molette",
     RGB(255,255,255))
58  EndIf
```

```
59   DrawText(0,60,".: Libérer la souris: Clic sur bouton gauche et droit :.")
60   StopDrawing()
61
62   ; Si clic sur bouton gauche ET droit alors la souris est libérée de l'écran courant
63   If MouseButton(#PB_MouseButton_Left) And MouseButton(#PB_MouseButton_Right)
64     ReleaseMouse(1)
65     oui=6
66     quitter=MessageRequester("Info !", "Voulez-vous quitter le programme ?",#PB_MessageRequester_YesNo)
67     If quitter=oui
68       End
69     EndIf
70   EndIf
71
72   FlipBuffers()
73   ClearScreen(RGB(0,0,0))
74
75   ForEver
76   End
```

Voir aussi

InitMouse() , MouseDeltaX() , MouseDeltaY() , MouseX() , MouseY() , MouseButton()

OS Supportés

Tous

119.3 MouseButton

Syntaxe

```
Resultat = MouseButton(Bouton)
```

Description

Teste si un bouton est enfoncé.

Arguments

Bouton
#PB_MouseButton_Left : Teste si le bouton gauche de la souris est enfoncé

```
#PB_MouseButton_Right :
 Teste si le bouton droit
 de la souris est enfoncé
#PB_MouseButton_Middle:
 Teste si le bouton du
 milieu de la souris est
 enfoncé
```

Valeur de retour

Renvoie une valeur non nulle si le bouton spécifié est appuyé, zéro sinon.

Remarques

On peut appuyer sur plusieurs boutons simultanément.
ExamineMouse() doit être appelé avant cette fonction pour mettre l'état des boutons à jour.

Exemple

```
1  ; Initialisation du monde 2D
2  InitSprite()
3  InitKeyboard()
4  InitMouse()
5
6   ; Ouverture de la fenêtre
7   OpenWindow(0,0,0,800,600,"Souris
     -
     MouseButton",#PB_Window_ScreenCentered|#PB_Window_SystemMenu
8   OpenWindowedScreen(WindowID(0),0,0,800,600,1,0,0)
9
10  ; Gestion de la fenêtre et
     de l'écran
11  Repeat
12   Repeat          ; Gestion
     de la fenêtre
13     Event  = WindowEvent()
14   Until Event = 0
15
16   ExamineMouse() ; Etat de
     la souris
17
18   ; Affichage de l'état des
     boutons de la souris
19   StartDrawing(ScreenOutput())
20   DrawText(10,10, "Appuyer
     sur une touche du clavier
     pour quitter.",
     RGB(255,255,0))
21   DrawText(200,180, "Cliquez
     sur le bouton gauche,
     milieu ou droit de la
     souris.",
     RGB(255,0,0),RGB(255,255,0))
```

```
22      If MouseButton(#PB_MouseButton_Left) <> 0
23        DrawText(200,230, "Bouton Gauche", RGB(255,0,0),RGB(255,255,0))
24      Else
25        DrawText(200,230, "Bouton Gauche", RGB(255,255,255))
26      EndIf
27      If MouseButton(#PB_MouseButton_Middle) <> 0
28        DrawText(350,230, "Bouton Central", RGB(255,0,0),RGB(255,255,0))
29      Else
30        DrawText(350,230, "Bouton Central", RGB(255,255,255))
31      EndIf
32      If MouseButton(#PB_MouseButton_Right) <> 0
33        DrawText(500,230, "Bouton Droit", RGB(255,0,0),RGB(255,255,0))
34      Else
35        DrawText(500,230, "Bouton Droit", RGB(255,255,255))
36      EndIf
37      StopDrawing()
38
39      FlipBuffers()
40      ClearScreen(RGB(0,0,0))
41
42      ExamineKeyboard()
43    Until KeyboardPushed(#PB_Key_All)
44    End
```

Voir aussi

ExamineMouse()

OS Supportés

Tous

119.4 MouseDeltaX

Syntaxe

```
Resultat = MouseDeltaX()
```

Description

Teste le déplacement horizontal de la souris.

Arguments

Aucun.

Valeur de retour

Renvoie le déplacement horizontal (en pixels) de la souris depuis le dernier appel de cette fonction.

Remarques

Le résultat peut être positif ou négatif selon que le déplacement s'est effectué vers la droite ou vers la gauche depuis le dernier appel.
ExamineMouse() doit être appelé avant cette fonction pour mettre la position courante de la souris à jour.

Exemple

```
; Initialisation du monde 2D
InitSprite()
InitKeyboard()
InitMouse()

 ; Ouverture de la fenêtre
 OpenWindow(0,0,0,800,600,"Souris - MouseDeltaX",#PB_Window_ScreenCentered|#PB_Window_SystemMenu
 OpenWindowedScreen(WindowID(0),0,0,800,600,1,0,0)

 ; Gestion de la fenêtre et de l'écran
 Repeat
  Repeat          ; Gestion de la fenêtre
    Event  = WindowEvent()
  Until Event = 0

  ExamineMouse() ; Etat de la souris

  ; Affichage de l'état des boutons de la souris
  StartDrawing(ScreenOutput())
  DrawText(10,10, "Appuyer sur une touche du clavier pour quitter.",
  RGB(255,255,0))
  DrawText(250,180, "Bougez la souris.",
  RGB(255,0,0),RGB(255,255,0))

```

```
23    DrawText(250,230, "DeltaX
      " + Str(MouseDeltaX()),
      RGB(255,255,255))
24    DrawText(250,260, "DeltaY
      " + Str(MouseDeltaY()),
      RGB(255,255,255))
25
26    StopDrawing()
27
28    FlipBuffers()
29    ClearScreen(RGB(0,0,0))
30
31    ExamineKeyboard()
32  Until
       KeyboardPushed(#PB_Key_All)
33  End
```

Voir aussi

ExamineMouse() , MouseDeltaY()

OS Supportés

Tous

119.5 MouseDeltaY

Syntaxe

```
Resultat = MouseDeltaY()
```

Description

Teste le déplacement vertical de la souris.

Arguments

Aucun.

Valeur de retour

Renvoie le déplacement vertical (en pixels) de la souris depuis le dernier appel de cette fonction.

Remarques

Le résultat peut être positif ou négatif selon que le déplacement s'est effectué vers le haut ou vers le bas depuis le dernier appel. ExamineMouse() doit être appelé avant cette fonction pour mettre la position courante de la souris à jour.

Exemple

```
; Initialisation du monde 2D
InitSprite()
InitKeyboard()
InitMouse()

; Ouverture de la fenêtre
OpenWindow(0,0,0,800,600,"Souris - MouseDeltaY",#PB_Window_ScreenCentered|#PB_Window_SystemMenu
OpenWindowedScreen(WindowID(0),0,0,800,600,1,0,0)

; Gestion de la fenêtre et de l'écran
Repeat
  Repeat         ; Gestion de la fenêtre
    Event  = WindowEvent()
  Until Event = 0

  ExamineMouse() ; Etat de la souris

  ; Affichage de l'état des boutons de la souris
  StartDrawing(ScreenOutput())
  DrawText(10,10, "Appuyer sur une touche du clavier pour quitter.",
  RGB(255,255,0))
  DrawText(250,180, "Bougez la souris.",
  RGB(255,0,0),RGB(255,255,0))

  DrawText(250,230, "DeltaX " + Str(MouseDeltaX()),
  RGB(255,255,255))
  DrawText(250,260, "DeltaY " + Str(MouseDeltaY()),
  RGB(255,255,255))

  StopDrawing()

  FlipBuffers()
  ClearScreen(RGB(0,0,0))

  ExamineKeyboard()
Until KeyboardPushed(#PB_Key_All)
End
```

Voir aussi

ExamineMouse() , MouseDeltaX()

OS Supportés

Tous

119.6 MouseLocate

Syntaxe

```
MouseLocate(X, Y)
```

Description

Change la position absolue (en pixels) de la souris dans l'écran courant.

Arguments

X, Y La nouvelle position du curseur.

Valeur de retour

Aucune.

Remarques

Principalement utilisé avec les commandes MouseX() et MouseY() .

Exemple

```
; Initialisation du monde 2D
InitSprite()
InitKeyboard()
InitMouse()

; Curseur de la souris
CreateImage(0,20,20)
StartDrawing(ImageOutput(0))
Circle(10, 10, 10, RGB(255, 255, 0))
StopDrawing()

; Ouverture de la fenêtre
OpenWindow(0,0,0,800,600,"Souris - MouseLocate",#PB_Window_ScreenCentered|#PB_Window_SystemMenu
OpenWindowedScreen(WindowID(0),0,0,800,600,1,0,0)

; Gestion de la fenêtre et de l'écran
Repeat
 Repeat          ; Gestion de la fenêtre
   Event  = WindowEvent()
 Until Event = 0

 ExamineMouse() ; Etat de la souris

 ; Affichage de l'état des boutons de la souris
 StartDrawing(ScreenOutput())
```

```
26    DrawText(10,10, "Appuyer
      sur une touche du clavier
      pour quitter.",
      RGB(255,255,0))
27    DrawText(100,180, "Bougez
      la souris puis cliquez
      pour remettre le curseur
      au centre de l'écran.",
      RGB(255,0,0),RGB(255,255,0))
28
29    DrawText(MouseX(),
      MouseY(),
      "["+Chr(164)+"]",
      RGB(255,255,0))
30
31    If
      MouseButton(#PB_MouseButton_Left)
32      MouseLocate(400, 300)
33      DrawText(MouseX(),
      MouseY(),
      "["+Chr(164)+"]",
      RGB(255,255,0))
34    EndIf
35
36    StopDrawing()
37
38    FlipBuffers()
39    ClearScreen(RGB(0,0,0))
40
41    ExamineKeyboard()
42  Until
      KeyboardPushed(#PB_Key_All)
43  End
```

Voir aussi

ExamineMouse() , MouseX() , MouseY()

OS Supportés

Tous

119.7 MouseWheel

Syntaxe

```
Resultat = MouseWheel()
```

Description

Teste la molette centrale de la souris.

Arguments

Aucun.

Valeur de retour

Renvoie le nombre de ”ticks” (unité pas-à-pas du système) réalisé par la molette de la souris depuis le dernier appel de cette fonction.

Remarques

La valeur est positive si la molette a été déplacée vers l'avant et négative si la molette a été déplacée vers l'arrière. ExamineMouse() doit être appelé avant cette fonction pour mettre les informations sur la souris à jour.

Exemple

```
; Initialisation du monde 2D
InitSprite()
InitKeyboard()
InitMouse()

 ; Ouverture de la fenêtre
 OpenWindow(0,0,0,800,600,"Souris - Utiliser la molette",#PB_Window_ScreenCentered|#PB_Window_SystemMenu)
 OpenWindowedScreen(WindowID(0),0,0,800,600,1,0,0)

 ; Gestion de la fenêtre et de l'écran
 Repeat
  Repeat          ; Gestion de la fenêtre
    Event  = WindowEvent()
  Until Event = 0

  ExamineMouse() ; Etat de la souris

 ; Molette de la souris = Rotation du curseur
  tiks = MouseWheel()
  ; Affichage de l'état des boutons de la souris
  StartDrawing(ScreenOutput())
  DrawText(10,10, "Appuyer sur une touche du clavier pour quitter.",
  RGB(255,255,0))
  DrawText(250,180,
  "Utilisez la molette centrale de la souris.",
  RGB(255,0,0),RGB(255,255,0))
   If tiks > 0
    DrawText(250,230,
  "Molette vers le haut" ,
  RGB(255,0,0),
  RGB(255,255,0))
```

```
26      Delay(100)
27    ElseIf tiks < 0
28      DrawText(450,230,
      "Molette vers le bas"  ,
      RGB(255,0,0),
      RGB(255,255,0))
29      Delay(100)
30    Else
31      DrawText(350,230,
      "Molette au repos" ,
      RGB(255,255,255))
32    EndIf
33
34    StopDrawing()
35
36    FlipBuffers()
37    ClearScreen(RGB(0,0,0))
38
39    ExamineKeyboard()
40  Until
      KeyboardPushed(#PB_Key_All)
41  End
```

Voir aussi

ExamineMouse()

OS Supportés

Windows

119.8 MouseX

Syntaxe

```
Resultat = MouseX()
```

Description

Renvoie la position horizontale de la souris.

Arguments

Aucun.

Valeur de retour

Renvoie la position horizontale actuelle (en pixels) de la souris sur l'écran actif.

Remarques

ExamineMouse() doit être appelé avant cette fonction pour mettre la position courante de la souris à jour.

Voir aussi

ExamineMouse() , MouseY() , MouseLocate()

Exemple

```
; Initialisation du monde 2D
InitSprite()
InitKeyboard()
InitMouse()

; Ouverture de la fenêtre
OpenWindow(0,0,0,800,600,"Souris - MouseX",#PB_Window_ScreenCentered|#PB_Window_SystemMenu)
OpenWindowedScreen(WindowID(0),0,0,800,600,1,0,0)

; Gestion de la fenêtre et de l'écran
Repeat
 Repeat          ; Gestion de la fenêtre
   Event  = WindowEvent()
 Until Event = 0

 ExamineMouse() ; Etat de la souris

 ; Affichage de l'état des boutons de la souris
 StartDrawing(ScreenOutput())
 DrawText(10,10, "Appuyer sur une touche du clavier pour quitter.",
 RGB(255,255,0))
 DrawText(300,180, "Bougez la souris.",
 RGB(255,0,0),RGB(255,255,0))

 DrawText(MouseX(),
 MouseY(),
 "["+Chr(164)+"]",
 RGB(255,255,0))

 DrawText(250,230, "X= " +
 Str(MouseX()) ,
 RGB(255,255,255))
 DrawText(450,230, "Y="  +
 Str(MouseY()) ,
 RGB(255,255,255))

 StopDrawing()

 FlipBuffers()
 ClearScreen(RGB(0,0,0))

 ExamineKeyboard()
```

```
34   Until
      KeyboardPushed(#PB_Key_All)
35   End
```

OS Supportés

Tous

119.9 MouseY

Syntaxe

```
Resultat = MouseY()
```

Description

Renvoie la position verticale de la souris.

Arguments

Aucun.

Valeur de retour

Renvoie la position verticale actuelle (en pixels) de la souris sur l'écran actif.

Remarques

ExamineMouse() doit être appelé avant cette fonction pour mettre la position courante de la souris à jour.

Exemple

```
1    ; Initialisation du monde 2D
2    InitSprite()
3    InitKeyboard()
4    InitMouse()
5
6    ; Ouverture de la fenêtre
7    OpenWindow(0,0,0,800,600,"Souris
      -
      MouseY",#PB_Window_ScreenCentered|#PB_Window_SystemMenu)
8    OpenWindowedScreen(WindowID(0),0,0,800,600,1,0,0)
9
10   ; Gestion de la fenêtre et
      de l'écran
11   Repeat
12    Repeat          ; Gestion
      de la fenêtre
13      Event  = WindowEvent()
14    Until Event = 0
15
16    ExamineMouse() ; Etat de
      la souris
17
```

```
18    ; Affichage de l'état des
      boutons de la souris
19    StartDrawing(ScreenOutput())
20    DrawText(10,10, "Appuyer
      sur une touche du clavier
      pour quitter.",
      RGB(255,255,0))
21    DrawText(300,180, "Bougez
      la souris.",
      RGB(255,0,0),RGB(255,255,0))
22
23    DrawText(MouseX(),
      MouseY(),
      "["+Chr(164)+"]",
      RGB(255,255,0))
24
25    DrawText(250,230, "X= " +
      Str(MouseX()) ,
      RGB(255,255,255))
26    DrawText(450,230, "Y="  +
      Str(MouseY()) ,
      RGB(255,255,255))
27
28    StopDrawing()
29
30    FlipBuffers()
31    ClearScreen(RGB(0,0,0))
32
33    ExamineKeyboard()
34  Until
      KeyboardPushed(#PB_Key_All)
35  End
```

Voir aussi

ExamineMouse() , MouseX() , MouseLocate()

OS Supportés

Tous

119.10 ReleaseMouse

Syntaxe

```
ReleaseMouse(Etat)
```

Description

Capture ou libère la souris pour rendre son usage possible dans le système d'exploitation.

Arguments

Etat #True : La souris est libérée de l'écran
#False : La souris est capturée dans l'écran.

Valeur de retour

Aucune.

Remarques

On utilise typiquement cette fonction après avoir vérifié le résultat de la commande IsScreenActive() .

Exemple

```
1
2   ; Quelques variables
3   MargeG = 20
4   MargeH = 20
5   LargeurEcran = 440
6   HauteurEcran = 440
7
8   ; Initialisation du monde 2D
9   InitSprite()
10  InitMouse()
11
12  ; Ouverture de la fenêtre et de l'écran
13  OpenWindow(0,0,0,650,480,"Capture/Libération de la souris",#PB_Window_ScreenCentered|#PB_Window_SystemMenu)
14  ButtonGadget(1,500,440,120,24,"Cliquez !")
15  OpenWindowedScreen(WindowID(0),20,20,LargeurEcran,HauteurEcra
16
17  ; Gestion de la fenêtre et de l'écran
18  Repeat
19
20    Repeat ; Gestion de la fenêtre
21      Event  = WindowEvent()
22      Select Event
23        Case #PB_Event_Gadget
24          If EventGadget() = 1     ; Si Clic sur le Bouton "Cliquez !"
25            MessageRequester("Attention","Bouton cliqué !")
26          EndIf
27        Case #PB_Event_CloseWindow ; Si fermeture de la fenêtre
28          End
29      EndSelect
30
```

```
31      ; Affichage de la position de la souris dans la fenêtre
32      SetWindowTitle(0, "Capture/Libération de la souris X= " + Str(mx) + " Y= "+ Str(my))
33    Until Event =  0
34
35    ; Gestion de la capture de la souris dans l'écran noir
36      If inScreen = #True
          ; Si la souris est dans l'écran noir...
37        If MouseX() > LargeurEcran-2 Or MouseY() > HauteurEcran-2 Or MouseX() < 1 Or MouseY() <1
38            ReleaseMouse(#True)
        ; ...et si elle s'approche des bords de l'écran alors on libère la souris
39            inScreen = #False
40        EndIf
41      Else
42        mx = WindowMouseX(0)
          ; Sinon, si la souris entre dans l'écran noir...
43        my = WindowMouseY(0)
44        If mx < LargeurEcran + MargeG And mx > MargeG And my > MargeH And my < MargeH + HauteurEcran
45          ReleaseMouse(#False)
        ; ... alors on capture la souris
46          MouseLocate(mx-MargeG,my-MargeH)
47          inScreen = #True
48        EndIf
49    EndIf
50
51    ; Affichage de l'écran noir
52    ClearScreen(0)
53    StartDrawing(ScreenOutput())
54        DrawText(150,200,"Souris relachée")
55        DrawText(180,230,"X= ")
56        DrawText(180,260,"Y= ")
57    StopDrawing()
58    If inScreen  ; Si la souris est dans l'écran noir...
59        ExamineMouse()
```

```
60          StartDrawing(ScreenOutput())
61          FrontColor(RGB(255,255,0))
62          DrawText(150,200,"Souris
    capturée" )
63          DrawText(180,230,"X=
    "+ Str(MouseX()))
64          DrawText(180,260,"Y=
    "+ Str(MouseY()))
65          DrawText(MouseX(),
    MouseY(), "["+Chr(164)+"]")
66          StopDrawing()
67       EndIf
68       FlipBuffers()
69    ForEver
```

Voir aussi

ExamineMouse() , IsScreenActive()

OS Supportés

Tous

Chapitre 120

Movie

Généralités

PureBasic propose des commandes simples mais très efficaces pour intégrer la diffusion d'une vidéo dans une application ou un jeu.
Windows : Dans la mesure où la technologie DirectX est utilisée, tous les types de vidéos ou même de médias peuvent être diffusés à partir de cette bibliothèque : AVI, MPG, DivX, etc.
MacOS X : La technologie QuickTime est utilisée et tous les types de médias (selon les plugins installés) peuvent être diffusés avec cette bibliothèque : AVI, MPG, DivX etc.
Note : Sur certains OS, les fichiers musicaux peuvent également être joués avec cette bibliothèque, mais ce n'est pas officiellement supporté et cela peut provoquer des dysfonctionnements.
Il vaut mieux utiliser la bibliothèque Sound pour cela.
Sous Windows, une version récente de DirectX 9 doit être installée (voir ici : DirectX 9 runtime installer).

OS Supportés

Tous

120.1 FreeMovie

Syntaxe

```
FreeMovie(#Video)
```

Description

Libère une vidéo et toutes les ressources associées.

Arguments

#**Video** La vidéo à libérer.

Si #PB_All est spécifié, toutes les vidéos restantes sont libérées.

Valeur de retour

Aucune.

Remarques

Toutes les vidéos restantes sont automatiquement libérées quand le programme se termine.

Voir aussi

IsMovie() , LoadMovie()

OS Supportés

Tous

120.2 InitMovie

Syntaxe

```
Resultat = InitMovie()
```

Description

Initialise l'environnement Vidéo pour un usage futur.

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Cette fonction doit être exécutée avant tout autre commande de cette bibliothèque. Cette commande tente d'ouvrir DirectX (v3.0 pour la compatibilité NT4.0 ou v7.0 ou plus sinon). Si la fonction échoue (Resultat = 0), cela peut donc provenir de l'absence de DirectX sur votre ordinateur ou d'une version de DirectX trop ancienne. Sous Windows, une version récente de DirectX 9 doit être installé (voir ici : DirectX 9 runtime installer).

Voir aussi

LoadMovie()

OS Supportés

Tous

120.3 IsMovie

Syntaxe

```
Resultat = IsMovie(#Video)
```

Description

Teste si une vidéo est correctement initialisée.

Arguments

#**Video** La vidéo à utiliser.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage.

Voir aussi

LoadMovie()

OS Supportés

Tous

120.4 LoadMovie

Syntaxe

```
Resultat = LoadMovie(#Video,
    Fichier$)
```

Description

Charge une vidéo depuis un fichier.

Arguments

#**Video** Le numéro d'identification de la vidéo.
PB_Any # peut être utilisé pour générer automatiquement ce numéro.

Fichier$ Le nom et le chemin du fichier vidéo.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Voir aussi

PlayMovie() , MovieInfo() , MovieLength() , MovieWidth() , MovieHeight() , FreeMovie() , InitMovie()

OS Supportés

Tous

120.5 MovieAudio

Syntaxe

```
MovieAudio(#Video, Volume,
    Balance)
```

Description

Contrôle le flux audio d'un vidéo, en temps réel.

Arguments

#**Video** La vidéo à utiliser.

Volume Le volume sonore entre 0 (muet) et 100 (volume à fond).

Balance La balance stéréo entre -100 et 100.

```
-100 : Tout le son sur le
 canal gauche et rien sur
 le canal droit
   0 : Mode normal
 100 : Tout le son sur le
 canal droit et rien sur
 le canal gauche
```

Valeur de retour

Aucune.

Voir aussi

PlayMovie()

OS Supportés

Windows, MacOS X

120.6 MovieHeight

Syntaxe

```
Resultat = MovieHeight(#Video)
```

Description

Renvoie la hauteur de la vidéo.

Arguments

#Video La vidéo à utiliser.

Valeur de retour

Renvoie la hauteur de la vidéo, en pixels. Renvoie -1 si le flux vidéo est absent ou incompatible, cependant le flux audio peut être joué.

Voir aussi

MovieWidth() , MovieLength() , MovieInfo()

OS Supportés

Tous

120.7 MovieInfo

Syntaxe

```
Resultat = MovieInfo(#Video,
    Options)
```

Description

Renvoie des informations additionnelles concernant une vidéo.

Arguments

#Video La vidéo à utiliser.

Options Valeur supportée actuellement :

```
0: Renvoie le nombre de
 trames par seconde
 (*1000).
```

Valeur de retour

Renvoie la valeur spécifiée en fonction du paramètre 'Options'.

Voir aussi

MovieLength() , MovieWidth() , MovieHeight()

OS Supportés

Windows, MacOS X

120.8 MovieLength

Syntaxe

```
Resultat = MovieLength(#Video)
```

Description

Renvoie la longueur (en trames) d'une vidéo.

Arguments

#**Video** La vidéo à utiliser.

Valeur de retour

Renvoie la longueur de la vidéo en nombre de trames.

Voir aussi

MovieInfo() , MovieWidth() , MovieHeight()

OS Supportés

Windows, MacOS X

120.9 MovieSeek

Syntaxe

```
Resultat = MovieSeek(#Video,
    Trame.q)
```

Description

Déplace la position courante d'une vidéo à une trame spécifiée.

Arguments

#**Video** La vidéo à utiliser.

Trame.q La trame (frame) à atteindre.

Valeur de retour

Aucune.

Voir aussi

MovieStatus()

OS Supportés

Windows, MacOS X

120.10 MovieStatus

Syntaxe

```
Resultat.q =
    MovieStatus(#Video)
```

Description

Obtient le statut d'une vidéo.

Arguments

#**Video** La vidéo à utiliser.

Valeur de retour

Renvoie l'une des valeurs suivantes :

```
   -1: La vidéo est en pause.
    0: La vidéo est stoppée
  > 0: La vidéo est en cours
   d'exécution.
       La valeur renvoyée est
   le numéro de la trame en
   cours d'affichage.
```

Voir aussi

MovieSeek()

OS Supportés

Tous

120.11 MovieWidth

Syntaxe

```
Resultat = MovieWidth(#Video)
```

Description

Renvoie la largeur d'une vidéo.

Arguments

#**Video** La vidéo à utiliser.

Valeur de retour

Renvoie la largeur d'une vidéo, en pixels. Renvoie -1 si le flux vidéo est absent ou incompatible, cependant le flux audio peut être joué.

Voir aussi

MovieWidth() , MovieLength() , MovieInfo()

OS Supportés

Tous

120.12 PauseMovie

Syntaxe

```
PauseMovie(#Video)
```

Description

Met en pause une vidéo.

Arguments

#**Video** La vidéo à utiliser.

Valeur de retour

Aucune.

Remarques

L'exécution peut être reprise en utilisant la commande ResumeMovie() .

Voir aussi

PlayMovie() , ResumeMovie() , StopMovie()

OS Supportés

Tous

120.13 PlayMovie

Syntaxe

```
Resultat = PlayMovie(#Video,
    Fenetre)
```

Description

Joue une vidéo dans une fenêtre spécifiée.

Arguments

#**Video** La vidéo à lire.

Fenetre La fenêtre dans laquelle sera lue la vidéo.
Cette valeur peut être facilement obtenue en utilisant la fonction WindowID() .
Il est aussi possible de jouer une vidéo en plein écran, il suffit d'utiliser le résultat de la commande ScreenID() dans 'Fenetre'.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Veillez à charger la vidéo au préalable avec LoadMovie() .
La commande ResizeMovie() peut être utilisée pour dimensionner et déplacer la vidéo dans cette fenêtre (pour ne pas utiliser la fenêtre entière par exemple).

Voir aussi

LoadMovie() , StopMovie() , MovieWidth() , MovieHeight() , MovieLength() , MovieInfo()

OS Supportés

Tous

120.14 ResizeMovie

Syntaxe

```
ResizeMovie(#Video, X, Y,
    Largeur, Hauteur)
```

Description

Déplace et redimensionne la zone d'affichage d'une vidéo.

Arguments

#**Video** La vidéo à utiliser.

X, Y, Largeur, Hauteur Les nouvelles coordonnées et dimensions de la vidéo, en pixels.

Valeur de retour

Aucune.

Voir aussi

PlayMovie() , MovieWidth() , MovieHeight()

OS Supportés

Tous

120.15 ResumeMovie

Syntaxe

```
ResumeMovie(#Video)
```

Description

Reprend l'exécution d'une vidéo après un appel à la commande PauseMovie() .

Arguments

#**Video** La vidéo à utiliser.

Valeur de retour

Aucune.

Voir aussi

PauseMovie() , PlayMovie() , StopMovie()

OS Supportés

Tous

120.16 StopMovie

Syntaxe

```
StopMovie(#Video)
```

Description

Arrête une vidéo.

Arguments

#**Video** La vidéo à utiliser.

Valeur de retour

Aucune.

Voir aussi

PlayMovie() , PauseMovie() , ResumeMovie()

OS Supportés

Tous

Chapitre 121

Music

Généralités

PureBasic peut jouer de la musique sous forme de modules musicaux (.mod, .xm etc.) pour ajouter un fond sonore à vos jeux ou à vos applications. Les modules musicaux sont bien connus des demo-makers car c'est un moyen simple et efficace de créer un environnement sonore. Les outils utilisés pour créer les modules musicaux s'appellent des 'trackers' (ProTracker, FastTracker, Impulse Tracker...). Par rapport aux fichiers wav et mp3, les modules musicaux ont l'avantage d'être très compacts et d'une longueur illimitée tout en consommant très peu de ressources processeur. Il est aussi possible de sélectionner en temps réel différentes parties du morceau pour s'adapter, par exemple, à chaque phase d'un jeu. Il est bien entendu possible de mixer un son classique et un module musical en les jouant simultanément.
La bibliothèque ModPlug est utilisée et permet une reproduction sonore très fidèle tout en supportant de nombreux formats musicaux différents (XM, S3M, ...).
Avant de pouvoir utiliser les commandes relatives aux modules musicaux, il est nécessaire d'appeler avec succès la commande InitSound() .

OS Supportés

Tous

121.1 CatchMusic

Syntaxe

```
Resultat = CatchMusic(#Music,
    *Memoire, Taille)
```

Description

Charge un module musical déjà situé en mémoire.

Arguments

#Music Le numéro d'identification du module musical.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

***Memoire** L'adresse mémoire (du buffer) où se trouve le module.

Taille Taille du tampon (buffer) en mémoire.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Si le module #Music était déjà créé, il est automatiquement supprimé et remplacé par le nouveau.
Cette commande est utile pour inclure des modules musicaux directement dans l'exécutable avec la commande IncludeBinary . Néanmoins, n'abusez pas de cette fonctionnalité car un module musical inclus dans un exécutable consomme automatiquement de la mémoire supplémentaire (car tout l'exécutable, y compris les modules, sont chargés en mémoire).

Exemple

```
1   CatchMusic(0, ?Music,
     ?MusicEnd-?Music)
2   End
3
4   DataSection
5     Music:
6       IncludeBinary "Music.xm"
7     MusicEnd:
8   EndDataSection
```

Voir aussi

LoadMusic() , PlayMusic()

OS Supportés

Tous

121.2 FreeMusic

Syntaxe

```
FreeMusic(#Music)
```

Description

Arrête l'exécution et supprime un module musical de la mémoire.

Arguments

#Music Le module musical à libérer.
Si #PB_All est spécifié, tous les modules musicaux restants sont libérés.

Valeur de retour

Aucune.

Remarques

Tous les modules musicaux restants sont automatiquement supprimés quand le programme se termine.

Voir aussi

LoadMusic() , CatchMusic()

OS Supportés

Tous

121.3 GetMusicPosition

Syntaxe

```
Resultat =
    GetMusicPosition(#Music)
```

Description

Renvoie la position en cours.

Arguments

#Music Le module musical à utiliser.

Valeur de retour

Renvoie la position dans le module musical en cours de lecture.
La première plage commence à 0.

Voir aussi

SetMusicPosition()

OS Supportés

Tous

121.4 GetMusicRow

Syntaxe

```
Resultat = GetMusicRow(#Music)
```

Description

Renvoie la ligne en cours.

Arguments

#**Music** Le module musical à utiliser.

Valeur de retour

Renvoie la ligne dans le module musical en cours de lecture.
La première ligne commence à 0.

Voir aussi

SetMusicPosition()

OS Supportés

Tous

121.5 IsMusic

Syntaxe

```
Resultat = IsMusic(#Music)
```

Description

Teste si un module musical est correctement initialisé.

Arguments

#**Music** Le module musical à tester.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.

Voir aussi

LoadMusic() , CatchMusic()

OS Supportés

Tous

121.6 LoadMusic

Syntaxe

```
Resultat = LoadMusic(#Music,
    Fichier$)
```

Description

Charge en mémoire un module musical depuis un fichier.

Arguments

#Music Le numéro d'identification du module musical.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Fichier$ Le nom et le chemin du fichier musical à charger.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Voir aussi

CatchMusic() , PlayMusic()

OS Supportés

Tous

121.7 MusicVolume

Syntaxe

```
Resultat =
    MusicVolume(#Music, Volume)
```

Description

Change le volume général en temps réel.

Arguments

#**Music** Le module musical à utiliser.

Volume Le nouveau volume sonore entre 0 et 100.
Utile pour les fondus enchaînés (fade in/fade out).

Valeur de retour

Aucune.

OS Supportés

Tous

121.8 PlayMusic

Syntaxe

```
PlayMusic(#Music)
```

Description

Joue un module musical.

Arguments

#**Music** Le module musical à jouer.

Valeur de retour

Aucune.

Voir aussi

LoadMusic() , CatchMusic() , StopMusic()

OS Supportés

Tous

121.9 SetMusicPosition

Syntaxe

```
SetMusicPosition(#Music,
    Position)
```

Description

Change la position.

Arguments

#**Music** Le module musical à utiliser.

Position Change la position du module musical par la plage spécifiée.
La première plage commence à 0.

Valeur de retour

Aucune.

Voir aussi

GetMusicPosition()

OS Supportés

Tous

121.10 StopMusic

Syntaxe

```
StopMusic(#Music)
```

Description

Arrête la lecture.

Arguments

#Music Le module musical à utiliser. Si '#Music' est égal à #PB_All, alors tous les modules musicaux en cours de lecture sont arrêtés.

Valeur de retour

Aucune.

Voir aussi

PlayMusic()

OS Supportés

Tous

Chapitre 122

Network

Généralités

Les réseaux sont utilisés dans le monde entier pour faciliter la communication et le transfert d'informations entre ordinateurs. PureBasic supporte le protocole d'échange de données officiel d'Internet appelé TCP/IP en version IPv4 et IPv6. Cela vous permet d'écrire des applications ou des jeux basés sur ce protocole et le modèle 'client-serveur'. Avec les fonctions de cette bibliothèque, il vous est possible de créer toutes sortes d'applications de type Internet (navigateurs, serveur Web, client FTP...) ou des jeux multijoueurs.

OS Supportés

Tous

122.1 CloseNetworkConnection

Syntaxe

```
CloseNetworkConnection(Connexion)
```

Description

Ferme une connexion.

Arguments

Connexion La connexion à fermer.
Ceci est le résultat soit d'un appel à OpenNetworkConnection() ou à EventClient() .

Valeur de retour

Aucune.

Remarques

Si cette commande est utilisée en mode client, le serveur recevra un événement #PB_NetworkEvent_Disconnect" (utilisateur déconnecté).
Si cette commande est utilisée en mode serveur, la connexion sera fermée sans aucune notification pour le client. Quand un serveur reçoit un événement #PB_NetworkEvent_Disconnect, la connexion client associée est automatiquement fermée.
CloseNetworkConnection() ne doit pas être appelé dans ce cas.
Toutes les connexions restant ouvertes sont automatiquement fermées quand le programme se termine.

Voir aussi

OpenNetworkConnection() , EventClient() , CloseNetworkServer()

OS Supportés

Tous

122.2 ConnectionID

Syntaxe

```
Resultat =
    ConnectionID(Connexion)
```

Description

Renvoie l'identifiant système unique d'une connexion.

Arguments

Connexion La connexion à tester.
Ceci est le résultat soit d'un appel à OpenNetworkConnection() ou à EventClient() .

Valeur de retour

Renvoie l'identifiant système.
Ce résultat est parfois aussi appelé 'Handle'. Regardez le chapitre Numéros et Identifiants (Handles) pour plus d'informations.

Voir aussi

ServerID()

OS Supportés

Tous

122.3 ServerID

Syntaxe

```
Resultat = ServerID(#Serveur)
```

Description

Renvoie l'identifiant système unique d'un serveur.

Arguments

#**Serveur** Le serveur à tester.

Valeur de retour

Renvoie l'identifiant système. Ce résultat est parfois aussi appelé 'Handle'. Regardez le chapitre Numéros et Identifiants (Handles)" pour plus d'informations.

Voir aussi

ConnectionID()

OS Supportés

Tous

122.4 CloseNetworkServer

Syntaxe

```
CloseNetworkServer(#Serveur)
```

Description

Ferme un serveur.

Arguments

#**Server** Le serveur à fermer.

Valeur de retour

Aucune.

Remarques

Tous les clients connectés à ce serveur sont automatiquement déconnectés. Le port est libéré et peut être réutilisé par une autre application ou par un autre serveur.

Voir aussi

CreateNetworkServer() ,
CloseNetworkConnection()

OS Supportés

Tous

122.5 CreateNetworkServer

Syntaxe

```
Resultat =
    CreateNetworkServer(#Serveur,
    Port [, Mode [,
    IPLocale$]])
```

Description

Crée un nouveau serveur réseau sur l'ordinateur local sur le port spécifié.

Arguments

#**Serveur** Le numéro d'identification du nouveau serveur.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Port Le port à utiliser par le serveur.
Les valeurs du port peuvent aller de 6000 à 7000 (ce sont les valeurs recommandées, mais elles peuvent aller en réalité de 0 à 65535).

Mode (optionnel) Peut être l'une des valeurs suivantes :

```
#PB_Network_TCP: Le
 serveur utilisera le
 protocole TCP (mode par
 défaut).
#PB_Network_UDP: Le
 serveur utilisera le
 protocole UDP.
```

Combiné à l'une des valeurs suivantes (en utilisant le symbole ||) :

```
#PB_Network_IPv4: Crée un
 serveur en utilisant
 IPv4 (par défaut).
#PB_Network_IPv6: Crée un
 serveur en utilisant
 IPv6.
```

IPLocale$ (optionnel) L'adresse IP locale à lier au serveur.

Par défaut, le serveur fonctionne sur toutes les interfaces locales disponibles, et accepte toutes leurs

connexions. Il peut être utile de restreindre le serveur à une seule interface (par exemple, "127.0.0.1") pour éviter les tentatives de connexion à partir d'autres interfaces. Sous Windows, la liaison à l'hôte local permet d'éviter de déclencher le pare-feu intégré.

Valeur de retour

Renvoie une valeur non nulle si le serveur a été créé avec succès ou zéro si la création a échoué (par exemple parce que le port est en cours d'utilisation).
Si #PB_Any a été utilisé comme paramètre #Serveur alors le nombre généré est renvoyé en cas de succès.

Remarques

Un nombre quelconque de serveurs peuvent tourner simultanément sur la même machine, mais jamais avec le même numéro de port et le même protocole (#PB_Network_TCP et #PB_Network_UDP).
Il est possible de créer deux serveurs utilisant le même port, l'un utilisant IPv4 et l'autre en utilisant IPv6.
NetworkServerEvent() peut être utilisé pour surveiller les nouveaux événements serveur.

Voir aussi

OpenNetworkConnection() ,
CloseNetworkServer() ,
NetworkServerEvent() ,

OS Supportés

Tous

122.6 ExamineIPAddresses

Syntaxe

```
Resultat =
    ExamineIPAddresses([Format])
```

Description

Démarre l'examen des adresses IP valables sur l'ordinateur local. NextIPAddress() sera utilisé pour retrouver chaque adresse.

Arguments

Format (optionnel) Le format de l'adresse IP à examiner.
Peut être une des valeurs suivantes :

```
#PB_Network_IPv4: Examine les adresses IPv4 (par défaut).
#PB_Network_IPv6: Examine les adresses IPv6. Les adresses renvoyées doivent être libérées avec FreeIP() après utilisation.
```

Valeur de retour

Renvoie une valeur non nulle si l'examen a réussi, zéro sinon.

Exemple : IPv4

```
1  If ExamineIPAddresses()
2    Repeat
3      IP = NextIPAddress()
4      If IP
5        Debug "IPv4: " + IPString(IP)
6      EndIf
7    Until IP = 0
8  EndIf
```

Exemple : IPv6

```
1  If ExamineIPAddresses(#PB_Network_IPv6)
2    Repeat
3      IP = NextIPAddress()
4      If IP
5        Debug "IPv6: " + IPString(IP, #PB_Network_IPv6)
6        FreeIP(IP)
7      EndIf
8    Until IP = 0
9  EndIf
```

Voir aussi

NextIPAddress()

OS Supportés

Tous

122.7 FreeIP

Syntaxe

```
FreeIP(AdresseIP)
```

Description

Libère une adresse IPv6.

Arguments

AdresseIP : L'adresse IPv6 à libérer.

Valeur de retour

Aucune.

Remarques

Cette fonction fonctionne uniquement avec des adresses IPv6 renvoyées par MakeIPAddress() , NextIPAddress() et GetClientIP() . Elle ne doit pas être utilisée avec des adresses IPv4.

Voir aussi

NextIPAddress() , MakeIPAddress() , GetClientIP()

OS Supportés

Tous

122.8 HostName

Syntaxe

```
Resultat\$ = HostName()
```

Description

Renvoie le nom de l'ordinateur local.

Arguments

Aucun.

Valeur de retour

Renvoie le nom d'hôte.

OS Supportés

Tous

122.9 IPString

Syntaxe

```
Resultat\$ =
    IPString(AdresseIP [,
    Format])
```

Description

Renvoie la représentation chaîne d'une adresse IP.

Arguments

AdresseIP L'adresse IP.
Pour IPv6, cette adresse doit être le résultat de MakeIPAddress() , NextIPAddress() ou GetClientIP() .

Format (optionnel) Le format de l'adresse IP à convertir.
Peut être une des valeurs suivantes :

```
#PB_Network_IPv4:
 Convertit une adresse
 IPv4 (par défaut).
#PB_Network_IPv6:
 Convertit une adresse
 IPv6.
```

Valeur de retour

Renvoie l'adresse IP en tant que chaîne. Sous la forme "127.0.0.1" pour IPv4 ou " : :1" pour IPv6.

Voir aussi

MakeIPAddress() , IPAddressField()

OS Supportés

Tous

122.10 IPAddressField

Syntaxe

```
Resultat =
    IPAddressField(AdresseIP,
    Champ [, Format])
```

Description

Renvoie la valeur du champ de l'adresse IP spécifiée.

Arguments

AdresseIP L'adresse IP.
Pour IPv6, cette adresse doit être créée avec MakeIPAddress() .

Champ Le champ à renvoyer.
Cette valeur peut être une valeur comprise entre 0 et 3.
(0 étant la valeur la plus à gauche, 3 étant la plus à droite) pour IPv4 et de 0 à 7 pour IPv6.

Format (optionnel) Le format de l'adresse IP.
Peut être une des valeurs suivantes :

```
#PB_Network_IPv4: Une
 adresse IPv4 (par
 défaut).
#PB_Network_IPv6: Une
 adresse IPv6.
```

Valeur de retour

Renvoie la valeur du champ spécifié, dans la plage de 0 à 255.

Remarques

Cette commande est utile quand elle est utilisée en conjonction avec :
- IPAddressGadget()
- MakeIPAddress()

Voir aussi

MakeIPAddress() , IPString() , IPAddressGadget()

OS Supportés

Tous

122.11 MakeIPAddress

Syntaxe

```
Resultat =
   MakeIPAddress(Champ0,
   Champ1, Champ2, Champ3 [,
   Champ4, Champ5, Champ6,
   Champ7])
```

Description

Renvoie la valeur numérique équivalente à une adresse IP.

Arguments

Champ0, Champ1, Champ2, Champ3
Les différents champs de l'adresse. Chaque champ a une valeur comprise entre 0 et 255.

Champ4, Champ5, Champ6, Champ7 (optionnel)
Les autres champs pour l'adresse IPv6. Chaque champ a une valeur comprise entre 0 et 255. Lorsque ces champs sont spécifiés, une adresse IPv6 est créée. Lorsque l'adresse n'est plus nécessaire, elle doit être libérée manuellement avec FreeIP() . Une adresse IPv4 ne doit pas être libérée avec FreeIP() .

Valeur de retour

Renvoie l'adresse IP.

Remarques

Cette commande est utile quand elle est utilisée en conjonction avec :
- IPAddressGadget()

Voir aussi

IPString() , IPAddressField()

OS Supportés

Tous

122.12 EventServer

Syntaxe

```
Resultat = EventServer()
```

Description

Cette fonction n'est nécessaire que côté serveur et permet de savoir sur quel serveur les données ont été reçues et ainsi de pouvoir gérer plusieurs serveurs simultanément dans le même programme.

Arguments

Aucun.

Valeur de retour

Renvoie le numéro du serveur qui a provoqué l'événement.

Voir aussi

NetworkServerEvent() , EventClient()

OS Supportés

Tous

122.13 EventClient

Syntaxe

```
Resultat = EventClient()
```

Description

Cette fonction n'est nécessaire que côté serveur et permet de savoir quel est la connexion du client qui a envoyé les données.

Arguments

Aucun.

Valeur de retour

Renvoie la connexion du client qui a provoqué l'événement.

Remarques

Les commandes GetClientIP() et GetClientPort() peuvent servir à collecter plus d'informations à propos du client qui a émis les données.

Voir aussi

NetworkServerEvent() , GetClientIP() , GetClientPort()

OS Supportés

Tous

122.14 GetClientIP

Syntaxe

```
Resultat = GetClientIP(Client)
```

Description

Renvoie l'adresse IP du client.

Arguments

Client Le client pour lequel vous souhaitez obtenir l'adresse IP.

Valeur de retour

Renvoie l'adresse IP du client.

Remarques

Cette commande doit être appelée après EventClient() .
La commande GetClientPort() est aussi disponible pour connaître le port du client.
IPString() peut être utilisé pour convertir l'adresse IP dans une chaîne.
Si la connexion est une connexion IPv6, l'adresse renvoyée doit être libérée avec FreeIP() après utilisation.

Voir aussi

GetClientPort() , IPString() , EventClient()

OS Supportés

Tous

122.15 GetClientPort

Syntaxe

```
Resultat =
    GetClientPort(Client)
```

Description

Renvoie le port du client.

Arguments

Client Le client pour lequel vous souhaitez obtenir le port.

Valeur de retour

Renvoie le port du client.

Remarques

Cette commande doit être appelée après EventClient() .

Voir aussi

GetClientIP() , EventClient()

OS Supportés

Tous

122.16 NetworkClientEvent

Syntaxe

```
Resultat =
    NetworkClientEvent(Connexion)
```

Description

Vérifie si un événement s'est produit sur une connexion réseau créée par OpenNetworkConnection() .

Arguments

Connexion La connexion à tester.

Valeur de retour

Renvoie une des valeurs suivantes :

```
  #PB_NetworkEvent_None
   : Il ne s'est rien passé.
  #PB_NetworkEvent_Data
   : Des données ont été
   reçues (à lire avec
   ReceiveNetworkData()
)
  #PB_NetworkEvent_Disconnect:
   Le client a été déconnecté
   (la connexion est fermée).
```

Voir aussi

ReceiveNetworkData() , NetworkServerEvent()

OS Supportés

Tous

122.17 NetworkServerEvent

Syntaxe

```
Resultat =
    NetworkServerEvent([#Serveur])
```

Description

Vérifie si un événement s'est produit sur l'un des serveurs du réseau.

Arguments

#**Serveur (optionnel)** Le serveur à surveiller.
Quand cette option est utilisée, seuls les évènements provenant de ce serveur sont utilisés, tous les autres évènements sont laissés de côté.

Arguments

Aucun.

Valeur de retour

Renvoie une des valeurs suivantes :

```
  #PB_NetworkEvent_None
   : Il ne s'est rien passé.
  #PB_NetworkEvent_Connect
   : Un nouveau client s'est
   connecté au serveur (non
   disponible avec les
   connexions
   #PB_Network_UDP).
  #PB_NetworkEvent_Data
   : Des données ont été
   reçues (à lire avec
   ReceiveNetworkData()
).
  #PB_NetworkEvent_Disconnect:
   Un client s'est déconnecté
   du serveur. Sa connexion
   associée est
   automatiquement fermée,
                                CloseNetworkConnection()
 ne doit pas être appelée
   pour ce client.
                                (Non
   disponible avec les
   connexions #PB_Network_UDP)
```

Remarques

Le serveur qui a reçu l'événement peut être déterminé avec EventServer() . Le client qui a provoqué l'événement peut être déterminé avec EventClient() .

Voir aussi

ReceiveNetworkData() , EventServer() , EventClient() , CreateNetworkServer()

OS Supportés

Tous

122.18 NextIPAddress

Syntaxe

```
Resultat = NextIPAddress()
```

Description

Renvoie l'adresse IP suivante de la machine locale pendant un examen avec ExamineIPAddresses() .

Arguments

Aucun.

Valeur de retour

Renvoie l'adresse IP suivante sous forme numérique. Si le résultat est zéro, alors il n'y a pas d'autres adresses IP à examiner. Si ExamineIPAddresses() est appelé avec le format #PB_Network_IPv6, les adresses IP renvoyées doivent être libérées avec FreeIP() après utilisation.

Exemple : IPv4

```
1  If ExamineIPAddresses()
2    Repeat
3      IP = NextIPAddress()
4      If IP
5        Debug "IPv4: " +
       IPString(IP)
6      EndIf
7    Until IP = 0
8  EndIf
```

Exemple : IPv6

```
1  If
     ExamineIPAddresses(#PB_Network_IPv6)
2    Repeat
3      IP = NextIPAddress()
4      If IP
5        Debug "IPv6: " +
       IPString(IP,
       #PB_Network_IPv6)
6        FreeIP(IP)
7      EndIf
8    Until IP = 0
9  EndIf
```

Voir aussi

ExamineIPAddresses() , IPString()

OS Supportés

Tous

122.19 OpenNetworkConnection

Syntaxe

```
Resultat =
    OpenNetworkConnection(NomServeur$,
    Port [, Mode [, TempsMax
    [, IPLocal$ [,
    PortLocal]]]])
```

Description

Ouvre une connexion réseau sur le serveur spécifié.

Arguments

NomServeur$ Le serveur.
Cela peut être une adresse IP ou un nom complet (ex : "127.0.0.1" ou "ftp.home.net").

Port Le port du serveur.

Mode (optionnel) Peut être une des valeurs suivantes :

```
#PB_Network_TCP: La
 connexion utilisera le
 protocole réseau TCP
 (default)
#PB_Network_UDP: La
 connexion utilisera le
 protocole réseau UDP.
                La
 connexion ne sera pas
 explicitement créée, car
 UDP est un protocole
 sans connexion, mais il
 faudra ajouter
                une
 entrée dans la pile de
 gestion PureBasic pour
 lui permettre d'envoyer
 des données via UDP en
 utilisant
                les
 fonctions régulières
 SendNetworkData()
(et connexes) .
```

Combinée à l'une des valeurs suivantes (en utilisant le symbole ||) :

```
#PB_Network_IPv4: Ouvre la
 connexion en utilisant
 IPv4 (par défaut).
```

```
#PB_Network_IPv6: Ouvre la
 connexion en utilisant
 IPv6.
```

TempsMax (optionnel) Le temps maximal (en millisecondes) avant d'abandonner la tentative de connexion. Il ne devrait pas être réglé sur une valeur trop faible (inférieur à 5000 millisecondes), car l'initialisation d'une connexion sur le réseau peut prendre un certain temps.

IPLocal$ (optionnel) L'adresse IP locale liée à la connexion.

PortLocal (optionnel) Le port local lié à la connexion.
Par défaut, un port local aléatoire est automatiquement choisi pour la nouvelle connexion, mais il peut être remplacé par ce paramètre.

Valeur de retour

Renvoie un identifiant de connexion pour la connexion ouverte si la connexion a été établie correctement, zéro sinon.

Voir aussi

NetworkClientEvent() , SendNetworkData() , ReceiveNetworkData() , CloseNetworkConnection()

OS Supportés

Tous

122.20 ReceiveNetworkData

Syntaxe

```
Resultat =
    ReceiveNetworkData(Connexion,
    *MemoireTampon,
    LongueurMemoireTampon)
```

Description

Reçoit une donnée brute du client spécifié. Cette fonction peut être utilisée par une application Serveur ou Client.

Arguments

Connexion La connexion.
Du côté serveur, 'Connexion' est le client qui a envoyé les données (peut être facilement obtenue avec EventClient()).

Du côté client, 'Connexion' est renvoyé par OpenNetworkConnection() .

***MemoireTampon** L'adresse de la mémoire tampon qui réceptionne les données.

LongueurMemoireTampon La taille de la mémoire tampon qui réceptionne les données.

Valeur de retour

Renvoie le nombre d'octets reçus.
Si 'Resultat' est égal à 'LongueurMemoireTampon', alors il reste des données en attente à lire.
Si une erreur s'est produite pendant la connexion (lien mort, connexion fermée par le serveur, etc), 'Resultat' sera égal à -1.

Remarques

Elle doit être appelée uniquement après avoir reçu un événement #PB_NetworkEvent_Data.
Avec les connexions UDP, le maximum pour 'LongueurMemoireTampon' est 2048.
Avec les connexions TCP, le maximum pour 'LongueurMemoireTampon' est 65536.

Voir aussi

NetworkClientEvent() , NetworkServerEvent() , SendNetworkData()

OS Supportés

Tous

122.21 SendNetworkData

Syntaxe

```
Resultat =
    SendNetworkData(Connexion,
    *MemoireTampon,
    LongueurMemoireTampon)
```

Description

Envoie des données brutes au client spécifié. Cette fonction peut être utilisée par une application Serveur ou Client.

Arguments

Connexion La connexion.
Du côté serveur, 'Connexion' est le client qui a reçu les données.

Du côté client, 'Connexion' est renvoyé par OpenNetworkConnection() .

***MemoireTampon** L'adresse de la mémoire tampon d'envoi des données.

LongueurMemoireTampon La taille de la mémoire tampon d'envoi des données.

Valeur de retour

Renvoie le nombre d'octets qui ont été effectivement envoyés.
S'il n'est pas égal à 'LongueurMemoireTampon', la mémoire tampon de réception de l'utilisateur est probablement pleine.
Si rien n'a pu être envoyé alors 'Resultat' sera égal à -1.

Remarques

Avec les connexions UDP, le maximum pour 'LongueurMemoireTampon' est 2048.
Avec les connexions TCP, le maximum pour 'LongueurMemoireTampon' est 65536.

Voir aussi

SendNetworkString()

OS Supportés

Tous

122.22 SendNetworkString

Syntaxe

```
Resultat =
    SendNetworkString(Connexion,
    Texte$ [, Format])
```

Description

Envoie une chaîne de caractères au client spécifié. Cette fonction peut être utilisée par une application Serveur ou Client.

Arguments

Connexion La connexion.
Du côté serveur, 'Connexion' est le client qui a reçu les données.
Du côté client, 'Connexion' est renvoyé par OpenNetworkConnection() .

Texte$ La chaîne de caractères à envoyer.

Format (optionnel) Le format de chaîne à utiliser lors de l'envoi.
Peut être une des valeurs suivantes :

```
#PB_Ascii   : Envoie les
 chaînes au format ASCII
#PB_UTF8    : Envoie les
 chaînes au format UTF8
 (Par défaut)
#PB_Unicode: Envoie les
 chaînes au format unicode
```

Valeur de retour

Renvoie le nombre d'octets qui a été envoyé.

Remarques

SendNetworkString() fournit une solution rapide pour envoyer rapidement des chaînes de caractères.
La chaîne sera envoyée en tant que donnée brute (sans le caractère NULL de terminaison), aussi peut-elle être reçue en utilisant ReceiveNetworkData() , après que NetworkServerEvent() / NetworkClientEvent() ait renvoyé #PB_NetworkEvent_Data).
En mode unicode la chaîne est envoyée en UTF-8, qui est indépendant du processeur (contrairement à UTF-16).
Il n'existe pas de fonction ReceiveNetworkString().

Voir aussi

SendNetworkData() , ReceiveNetworkData()

OS Supportés

Tous

Chapitre 123

Node

Généralités

Les noeuds sont des conteneurs qui peuvent être utilisés pour grouper des objets comme des entités , des sons 3D , des caméras , des billboards , des émetteurs de particules et également d'autres noeuds.
Une fois qu'un objet est attaché à un noeud, sa position et ses mouvements sont relatifs à la position du noeud.
Les noeuds permettent une gestion hiérarchique des objets.
InitEngine3D() doit être appelé avec succès avant de pouvoir utiliser les commandes relatives aux noeuds.

OS Supportés

Tous

123.1 AttachNodeObject

Syntaxe

```
AttachNodeObject(#Noeud,
    ObjetID)
```

Description

Attache un objet à un noeud.

Arguments

#Noeud Le noeud à utiliser.

ObjetID Le numéro de l'objet.
Cet objet peut être :

```
   - Une entité : Utiliser
     EntityID()
  à la place de 'ObjetID'
   - Un son 3D  : Utiliser
     SoundID3D()
  à la place de 'ObjetID'
```

```
  - Une caméra : Utiliser
    CameraID()
à la place de 'ObjetID'
 - Une lumière: Utiliser
    LightID()
à la place de 'ObjectID'.
 - Un noeud   : Utiliser
    NodeID()
à la place de 'ObjetID'.
 - Mesh       : Utiliser
    MeshID()
à la place de 'ObjectID'
    (supporté seulement avec
    l'option
    #PB_Mesh_Dynamic).
 - Un groupe de billboards
    : BillboardGroupID()
à la place de 'ObjetID'
 - Un émetteur de
    particules:
    ParticleEmitterID()
à la place de 'ObjetID'.
```

Valeur de retour

Aucune.

Voir aussi

DetachNodeObject()

OS Supportés

Tous

123.2 DetachNodeObject

Syntaxe

```
DetachNodeObject(#Noeud,
    ObjetID)
```

Description

Détache un objet précédemment attaché à un noeud.

Arguments

#Noeud Le noeud à utiliser.

ObjetID Le numéro de l'objet.
Cet objet peut être :

```
  - Une entité : Utiliser
    EntityID()
à la place de 'ObjetID'
 - Un son 3D  : Utiliser
    SoundID3D()
```

```
à la place de 'ObjetID'
 - Une caméra : Utiliser
   CameraID()
à la place de 'ObjetID'
 - Une lumière: Utiliser
   LightID()
à la place de 'ObjectID'.
 - Un noeud   : Utiliser
   NodeID()
à la place de 'ObjetID'.
 - Un groupe de billboards
   : BillboardGroupID()
à la place de 'ObjetID'
 - Un émetteur de
   particules:
   ParticleEmitterID()
à la place de 'ObjetID'.
```

Valeur de retour

Aucune.

Voir aussi

AttachNodeObject()

OS Supportés

Tous

123.3 CreateNode

Syntaxe

```
Resultat = CreateNode(#Noeud
    [, X, Y, Z])
```

Description

Crée un nouveau noeud.

Arguments

#Noeud Le noeud à créer.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Z (optionnel) Les coordonnées du nouveau noeud.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si le noeud était déjà créé alors il sera automatiquement supprimé et remplacé par le nouveau.

Voir aussi

IsNode() , NodeID()

OS Supportés

Tous

123.4 NodeID

Syntaxe

```
NoeudID = NodeID(#Noeud)
```

Description

Renvoie l'identifiant unique d'un noeud.

Arguments

#**Noeud** Le noeud à utiliser.

Valeur de retour

Renvoie le numéro du noeud.

Remarques

Cette fonction est très utile quand une fonction d'une autre bibliothèque nécessite l'identifiant d'un noeud.

Voir aussi

CreateNode() , IsNode()

OS Supportés

Tous

123.5 NodeLookAt

Syntaxe

```
NodeLookAt(#Noeud, X, Y, Z [,
    DirectionX.f,
    DirectionY.f,
    DirectionZ.f])
```

Description

Change l'orientation d'un noeud dans le monde.

Arguments

#**Noeud** Le noeud à utiliser.

X, Y, Z Le noeud pointe vers la nouvelle position 'X, Y, Z' (dans l'unité du monde).

DirectionX.f, DirectionY.f, DirectionZ.f (optionnel) La direction en X, Y et Z du vecteur du noeud (valeur comprise entre -1.0 et 1.0).

Valeur de retour

Aucune.

Remarques

La position du noeud n'est pas modifiée.

OS Supportés

Tous

123.6 NodeX

Syntaxe

```
Resultat = NodeX(#Noeud [,
    Mode])
```

Description

Renvoie la position en 'X' d'un noeud dans le Monde.

Arguments

#**Noeud** Le noeud à utiliser.

Mode (optionnel) Peut être une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 position absolue en 'X'
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 position relative à son
 parent en 'X'.
```

Valeur de retour

Renvoie la position 'X' du noeud dans le Monde.

Voir aussi

NodeY() , NodeZ() , MoveNode()

OS Supportés

Tous

123.7 NodeY

Syntaxe

```
Resultat = NodeY(#Noeud [,
    Mode])
```

Description

Renvoie la position relative en 'Y' d'un noeud dans le Monde.

Arguments

#**Noeud** Le noeud à utiliser.

Mode (optionnel) Peut être une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
  position absolue en 'Y'
  dans le monde (par
  défaut).
#PB_Relative: Renvoie la
  position relative à son
  parent en 'Y'.
```

Valeur de retour

Renvoie la position 'Y' du noeud dans le Monde.

Voir aussi

NodeX() , NodeZ() , MoveNode()

OS Supportés

Tous

123.8 NodeZ

Syntaxe

```
Resultat = NodeZ(#Noeud [,
    Mode])
```

Description

Renvoie la position relative en 'Z' d'un noeud dans le Monde.

Arguments

#**Noeud** Le noeud à utiliser.

Mode (optionnel) Peut être une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 position absolue en 'Z'
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 position relative à son
 parent en 'Z'.
```

Valeur de retour

Renvoie la position 'Z' du noeud dans le Monde.

Voir aussi

NodeX() , NodeY() , MoveNode()

OS Supportés

Tous

123.9 FreeNode

Syntaxe

```
FreeNode(#Noeud)
```

Description

Supprime un noeud.

Arguments

#**Noeud** Le noeud à utiliser.
Si #PB_All est spécifié, tous les nuds restants sont libérés.

Valeur de retour

Aucune.

Remarques

La mémoire qui lui était associée est libérée et le noeud ne peut plus être utilisé.
Par contre les objets qui lui étaient attachés ne sont pas supprimés et peuvent être réutilisés.
Tous les noeuds restants sont automatiquement supprimés quand le programme se termine.

Voir aussi

CreateNode() , NodeID() , IsNode()
CreateNode()

OS Supportés

Tous

123.10 IsNode

Syntaxe

```
Resultat = IsNode(#Noeud)
```

Description

Teste si un noeud est correctement initialisé.

Arguments

#**Noeud** Le noeud à utiliser.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.
Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage.

Voir aussi

CreateNode() , NodeID() , FreeNode()

OS Supportés

Tous

123.11 MoveNode

Syntaxe

```
MoveNode(#Noeud, X, Y, Z [,
    Mode])
```

Description

Déplace un noeud dans le monde 3D.

Arguments

#**Noeud** Le noeud à utiliser.

X, Y, Z Nouvelles coordonnées du noeud.

Mode (optionnel) Peut être une des valeurs suivantes :

```
#PB_Relative: Déplacement
 relatif, à partir de la
 position actuelle du
 noeud (par défaut).
#PB_Absolute: Déplacement
 absolu à la position
 spécifiée.
```

combinée avec l'une des valeurs suivantes :

```
#PB_Local : Déplacement
 local.
#PB_Parent: Déplacement
 par rapport à la
 position du parent.
#PB_World : Déplacement
 par rapport au monde.
```

Valeur de retour

Aucune.

Remarques

Le déplacement est relatif à la position actuelle d'un noeud.

Voir aussi

NodeX() , NodeY() , NodeZ() , RotateNode()

OS Supportés

Tous

123.12 RotateNode

Syntaxe

```
RotateNode(#Noeud, X, Y, Z [,
    Mode])
```

Description

Effectue une rotation d'un noeud en fonction de la valeur des angles X, Y, Z.

Arguments

#**Noeud** Le noeud à utiliser.

X, Y, Z Angles de rotation en X, Y, Z. Tous les angles sont en degrés de 0° à 360°.

Mode (optionnel)

```
      PB_Absolute: Rotation
    absolue (par défaut).
  PB_Relative: Rotation
    relative basée sur la
    rotation précédente de
    l'entité.
```

Valeur de retour

Aucune.

Voir aussi

NodePitch() , NodeRoll() , NodeYaw() , MoveNode()

OS Supportés

Tous

123.13 ScaleNode

Syntaxe

```
ScaleNode(#Noeud, X, Y, Z [,
    Mode])
```

Description

Change la taille d'un noeud.

Arguments

#**Noeud** Le noeud à utiliser.

X, Y, Z Les facteurs d'échelle sur les trois axes.

Mode (optionnel) Peut être une des valeurs suivantes :

```
  #PB_Relative: Facteur
    d'échelle relatif, sur
    la base de la taille
    initiale (par défaut).
    L'utilisation de 1.0
    pour facteur d'échelle
    conservera la taille
    inchangée.
  #PB_Absolute: Facteur
    d'échelle absolue, dans
    l'unité du monde.
```

Valeur de retour

Aucune.

Remarques

Lorsque vous utilisez le mode #PB_Relative, alors la taille du noeud sera multiplié par la valeur donnée pour obtenir la nouvelle taille.

OS Supportés

Tous

123.14 NodeFixedYawAxis

Syntaxe

```
NodeFixedYawAxis(#Noeud,
    Activer [, VecteurX.f,
    VecteurY.f, VecteurZ.f])
```

Description

Changer l'axe de lacet du nud.

Arguments

#Noeud Le noeud à utiliser.

#Noeud Le noeud à utiliser.

Activer Active ou désactive l'utilisation d'un axe de lacet personnalisé.

```
#True : Un nouvel axe
 vecteur doit être
 spécifié.
#False: La caméra
 effectuera un lacet
 autour de son propre axe
 Y.
```

VecteurX.f, VecteurY.f, VecteurZ.f (optionnel) Direction du vecteur du nouvel axe de lacet (valeur comprise entre -1.0 et 1.0). Le paramètre "Activer" doit être réglé sur #True pour avoir un effet.

Valeur de retour

Aucune.

Remarques

Le comportement par défaut d'un noeud est un lacet autour de son propre axe Y.

Voir aussi

NodeYaw()

OS Supportés

Tous

123.15 NodeRoll

Syntaxe

```
Resultat.f = NodeRoll(#Noeud
    [, Mode])
```

Description

Renvoie le roulis d'un noeud.

Arguments

#**Noeud** Le noeud à utiliser.

Mode (optionnel) Peut être une des valeurs suivantes :

```
    #True : La valeur brute,
      mais elle ne peut pas
      être utilisée avec
      RotateNode()
   pour récupérer la même
      orientation (par défaut).
    #False: Le roulis est
      ajusté, de sorte qu'il
      peut être réutilisé avec
      RotateNode()
   pour récupérer la même
      orientation.
```

Valeur de retour

La valeur courante du roulis du noeud. Valeur toujours comprise entre -180.0 et 180.0 degrés.

Voir aussi

NodeYaw() , NodePitch()

OS Supportés

Tous

123.16 NodePitch

Syntaxe

```
Resultat.f =
    NodePitch(#Noeud[, Mode])
```

Description

Renvoie le tangage d'un noeud.

Arguments

#**Noeud** Le noeud à utiliser.

Mode (optionnel) Peut être une des valeurs suivantes :

```
    #True : La valeur brute,
      mais elle ne peut pas
      être utilisée avec
      RotateNode()
   pour récupérer la même
      orientation (par défaut).
    #False: Le tangage est
      ajusté, de sorte qu'il
      peut être réutilisé avec
      RotateNode()
   pour récupérer la même
      orientation.
```

Valeur de retour

La valeur courante du tangage du noeud. Valeur toujours comprise entre -180.0 et 180.0 degrés.

Voir aussi

NodeYaw() , NodeRoll()

OS Supportés

Tous

123.17 NodeYaw

Syntaxe

```
Resultat.f = NodeYaw(#Noeud
    [, Mode])
```

Description

Renvoie le lacet d'un noeud.

Arguments

#**Noeud** Le noeud à utiliser.

Mode (optionnel) Peut être une des valeurs suivantes :

```
 #True : La valeur brute,
  mais elle ne peut pas
  être utilisée avec
  RotateNode()
pour récupérer la même
  orientation (par défaut).
 #False: Le lacet est
  ajusté, de sorte qu'il
  peut être réutilisé avec
  RotateNode()
pour récupérer la même
  orientation.
```

Valeur de retour

La valeur courante du lacet du noeud. Valeur toujours comprise entre -180.0 et 180.0 degrés.

Voir aussi

NodePitch() , NodeRoll()

OS Supportés

Tous

Chapitre 124

NodeAnimation

Généralités

Cette bibliothèque d'animation des noeuds permet de définir un chemin à suivre pour un nud, avec une image-clé (key frame) prédéfinie et l'exécution d'une interpolation avec lissage.
En conséquence, un objet comme une caméra peut être fixée à ce noeud et être déplacée facilement le long du chemin. (effet de travelling)
InitEngine3D() doit être appelé avec succès avant d'utiliser les fonctions d'animation des noeuds.

OS Supportés

Tous

124.1 CreateNodeAnimation

Syntaxe

```
Resultat =
    CreateNodeAnimation(#NodeAnimation,
    NoeudID, Duree,
    Interpolation,
    InterpolationRotation)
```

Description

Crée une nouvelle animation.

Arguments

#NodeAnimation Le numéro d'identification de l'animation du noeud. #PB_Any peut être utilisé pour générer automatiquement ce numéro.

NoeudID Le noeud à déplacer automatiquement.
NodeID() peut être utilisé pour obtenir un identifiant de noeud valide.

Duree La durée de la nouvelle animation (en millisecondes).

Interpolation Le type d'interpolation à appliquer entre les points :

```
#PB_NodeAnimation_Linear:
 Chacun des points sera
 relié entre eux à l'aide
 d'une ligne droite,
                          ce
 qui peut entraîner un
 brusque changement de
 direction.
#PB_NodeAnimation_Spline:
 Chacun des points sera
 relié entre eux à l'aide
 d'une courbe (spline),
                          ce
 qui conduit un
 changement de direction
 en douceur, mais c'est
 plus lent.
```

InterpolationRotation Le type d'interpolation pour une rotation à appliquer entre les points.
Peut être l'une des valeurs suivantes :

```
#PB_NodeAnimation_LinearRotation
   : Interpolation
 linéaire : chaque point
 sera relié ensemble
                                  par
 une ligne droite, ce qui
 peut occasionner une
 rotation saccadée.
#PB_NodeAnimation_SphericalRotation:
 Interpolation
 Sphérique:chaque point
 sera relié ensemble
                                  par
 une courbe, ce qui
 donnera une rotation
 plus lisse, mais c'est
 aussi plus lent.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any est utilisé, le nouveau numéro sera renvoyé dans 'Resultat'.

Remarques

Une animation d'un noeud n'existe pas physiquement dans le Monde 3D, il s'agit d'un chemin (piste ou track) virtuel sur lequel se déplace un noeud et son objet attaché à lui.

Voir aussi

FreeNodeAnimation()

OS Supportés

Tous

124.2 FreeNodeAnimation

Syntaxe

```
FreeNodeAnimation(#NodeAnimation)
```

Description

Libère une animation et toute la mémoire associée.

Arguments

#NodeAnimation L'animation à libérer. Si #PB_All est spécifié, toutes les animations restantes sont libérées.

Valeur de retour

Aucune.

Remarques

Toutes les animations restantes sont automatiquement libérées lorsque le programme se termine.

Remarques

Son numéro ne doit pas être utilisé avec les autres fonctions de cette bibliothèque après l'appel de cette fonction, à moins de la créer à nouveau.

Voir aussi

CreateNodeAnimation()

OS Supportés

Tous

124.3 CreateNodeAnimationKeyFrame

Syntaxe

```
CreateNodeAnimationKeyFrame(#NodeAnimation,
    Temps, X, Y, Z)
```

Description

Crée une nouvelle image-clé dans une animation.

Arguments

#**NodeAnimation** L'animation à utiliser.

Temps Le temps dans l'animation où l'on choisit l'image-clé (en millisecondes). Cette valeur doit être comprise entre zéro et la 'Durée' définie dans CreateNodeAnimation() .

X, Y, Z La nouvelle position de l'image-clé dans le monde.

Valeur de retour

Aucune.

Remarques

Une image-clé est un point dans le monde à un moment précis.
Lorsque l'animation sera jouée, elle suivra chaque image-clé et donc le déplacement de point en point. Le déplacement sera interpolé pour respecter la contrainte de temps.
Par exemple, si la première image-clé est définie au temps 0, la seconde au temps 1000 millisecondes, et la troisième à 3000, le passage de la première à la seconde durera 1000 millisecondes, et le passage de la seconde à la troisième durera 2000 millisecondes. L'animation globale durera 3000 millisecondes.

Voir aussi

CreateNodeAnimation()

OS Supportés

Tous

124.4 GetNodeAnimationKeyFrameTime

Syntaxe

```
Resultat =
    GetNodeAnimationKeyFrameTime(#NodeAnimation,
    ImageCle)
```

Description

Renvoie le temps d'un image-clé.

Arguments

#**NodeAnimation** L'animation à utiliser.

#**ImageCle** L'indice de l'image-clé.
L'indice de la première image-clé est égal à zéro.

Valeur de retour

Renvoie le temps de l'image-clé (en millisecondes) de l'animation, ou zéro si l'image-clé n'existe pas.

Voir aussi

CreateNodeAnimationKeyFrame()

OS Supportés

Tous

124.5 SetNodeAnimationKeyFramePosition

Syntaxe

```
SetNodeAnimationKeyFramePosition(#NodeAnimation,
    ImageCle, X, Y, Z)
```

Description

Modifie la position de l'image-clé d'une animation.

Arguments

#**NodeAnimation** L'animation à utiliser.

#**ImageCle** L'indice de l'image-clé.
L'indice de la première image-clé est égal à zéro.
Si l'image-clé n'existe pas, cette fonction n'a aucun effet.

X, Y, Z La nouvelle position de l'image-clé dans le monde.

Valeur de retour

Aucune.

Voir aussi

CreateNodeAnimationKeyFrame()

OS Supportés

Tous

124.6 GetNodeAnimationKeyFrameX

Syntaxe

```
Resultat =
    GetNodeAnimationKeyFrameX(#NodeAnimation,
    ImageCle)
```

Description

Renvoie la position en 'X' de l'image-clé d'une animation.

Arguments

#NodeAnimation L'animation à utiliser.

#ImageCle L'indice de l'image-clé. L'indice de la première image-clé est égal à zéro.

Valeur de retour

Renvoie la position en 'X' de l'image-clé dans le monde ou zéro si l'image-clé n'existe pas.

Voir aussi

CreateNodeAnimationKeyFrame() , GetNodeAnimationKeyFrameY() , GetNodeAnimationKeyFrameZ()

OS Supportés

Tous

124.7 GetNodeAnimationKeyFrameY

Syntaxe

```
Resultat =
    GetNodeAnimationKeyFrameY(#NodeAnimation,
    ImageCle)
```

Description

Renvoie la position en 'Y' de l'image-clé d'une animation.

Arguments

#NodeAnimation L'animation à utiliser.

#ImageCle L'indice de l'image-clé. L'indice de la première image-clé est égal à zéro.

Valeur de retour

Renvoie la position en 'Y' de l'image-clé dans le monde ou zéro si l'image-clé n'existe pas.

Voir aussi

CreateNodeAnimationKeyFrame() , GetNodeAnimationKeyFrameX() , GetNodeAnimationKeyFrameZ()

OS Supportés

Tous

124.8 GetNodeAnimationKeyFrameZ

Syntaxe

```
Resultat =
    GetNodeAnimationKeyFrameZ(#NodeAnimation,
    ImageCle)
```

Description

Renvoie la position en 'Z' de l'image-clé d'une animation.

Arguments

#**NodeAnimation** L'animation à utiliser.

#**ImageCle** L'indice de l'image-clé. L'indice de la première image-clé est égal à zéro.

Valeur de retour

Renvoie la position en 'Z' de l'image-clé dans le monde ou zéro si l'image-clé n'existe pas.

Voir aussi

CreateNodeAnimationKeyFrame() , GetNodeAnimationKeyFrameX() , GetNodeAnimationKeyFrameY()

OS Supportés

Tous

124.9 SetNodeAnimationKeyFrameRotation

Syntaxe

```
SetNodeAnimationKeyFrameRotation(#NodeAnimation,
    ImageCle, X, Y, Z [, W,
    Mode])
```

Description

Modifie la rotation de l'image-clé d'une animation.

Arguments

#NodeAnimation L'animation à utiliser.

#ImageCle L'indice de l'image-clé.
L'indice de la première image-clé est égal à zéro.
Si l'image-clé n'existe pas, cette fonction n'a aucun effet.

X, Y, Z Rotation 'X, Y, Z' de l'image-clé dans le monde. Les valeurs dépendent du mode sélectionné.

W (optionnel) Rotation 'W' de l'image-clé dans le monde. (Utilisé uniquement avec #PB_Orientation_Quaternion et #PB_Orientation_Direction).

Mode (optionnel) Le mode de rotation.
Peut avoir l'une des valeurs suivantes :

- #PB_Orientation_PitchYawRoll: 'X' (tangage), 'Y' (lacet), 'Z' (roulis), appliqués dans cet ordre (par défaut).
- #PB_Orientation_Quaternion : 'X', 'Y', 'Z', 'W' pour les valeurs du quaternion
- #PB_Orientation_Direction : 'X', 'Y', 'Z' pour le vecteur de direction et 'W' pour la rotation (roulis).

Valeur de retour

Aucune.

Voir aussi

CreateNodeAnimationKeyFrame() ,
GetNodeAnimationKeyFramePitch() ,
GetNodeAnimationKeyFrameYaw() ,
GetNodeAnimationKeyFrameRoll()

OS Supportés

Tous

124.10 GetNodeAnimationKeyFramePitch

Syntaxe

```
Resultat =
    GetNodeAnimationKeyFramePitch(#NodeAnimation,
    ImageCle)
```

Description

Renvoie le tangage de l'image-clé d'une animation.

Arguments

#**NodeAnimation** L'animation à utiliser.

#**ImageCle** L'indice de l'image-clé.
L'indice de la première image-clé est égal à zéro.

Valeur de retour

Renvoie le tangage actuel de l'image-clé ou zéro si l'image clé n'existe pas.

Voir aussi

CreateNodeAnimationKeyFrame() , GetNodeAnimationKeyFrameYaw() , GetNodeAnimationKeyFrameRoll()

OS Supportés

Tous

124.11 GetNodeAnimationKeyFrameYaw

Syntaxe

```
Resultat =
    GetNodeAnimationKeyFrameYaw(#NodeAnimation,
    ImageCle)
```

Description

Renvoie le lacet de l'image-clé d'une animation.

Arguments

#**NodeAnimation** L'animation à utiliser.

#**ImageCle** L'indice de l'image-clé.
L'indice de la première image-clé est égal à zéro.

Valeur de retour

Renvoie le lacet actuel de l'image-clé ou zéro si l'image clé n'existe pas.

Voir aussi

CreateNodeAnimationKeyFrame() , GetNodeAnimationKeyFramePitch() , GetNodeAnimationKeyFrameRoll()

OS Supportés

Tous

124.12 GetNodeAnimationKeyFrameRoll

Syntaxe

```
Resultat =
    GetNodeAnimationKeyFrameRoll(#NodeAnimation,
    ImageCle)
```

Description

Renvoie le roulis de l'image-clé d'une animation.

Arguments

#**NodeAnimation** L'animation à utiliser.

#**ImageCle** L'indice de l'image-clé. L'indice de la première image-clé est égal à zéro.

Valeur de retour

Renvoie le roulis actuel de l'image-clé ou zéro si l'image clé n'existe pas.

Voir aussi

CreateNodeAnimationKeyFrame() , GetNodeAnimationKeyFramePitch() , GetNodeAnimationKeyFrameYaw()

OS Supportés

Tous

124.13 SetNodeAnimationKeyFrameScale

Syntaxe

```
SetNodeAnimationKeyFrameScale(#NodeAnimation,
    ImageCle, X.f, Y.f, Z.f)
```

Description

Modifie le facteur d'échelle de l'image-clé d'une animation.

Arguments

#NodeAnimation L'animation à utiliser.

#ImageCle L'indice de l'image-clé.
L'indice de la première image-clé est égal à zéro.
Si l'image-clé n'existe pas alors cette fonction n'a pas d'effet.

X.f, Y.f, Z.f Le nouveau facteur d'échelle de l'image-clé.
Avec une valeur de 1.0, aucune mise à l'échelle ne sera appliquée sur l'axe concerné.

Valeur de retour

Aucune.

Remarques

Le facteur d'échelle est appliqué au noeud associé à l'animation.
Le facteur d'échelle va redimensionner le noeud en multipliant sa taille par le facteur donné :
- Un facteur d'échelle de 1.0 n'affectera pas la taille du noeud
- Un facteur d'échelle de 2.0 permettra de doubler la taille du noeud
- Un facteur d'échelle de 0,5 permettra la rédction de moitié du noeud

Voir aussi

CreateNodeAnimationKeyFrame() , GetNodeAnimationKeyFramePitch() , GetNodeAnimationKeyFrameYaw() , GetNodeAnimationKeyFrameRoll()

OS Supportés

Tous

124.14 AddNodeAnimationTime

Syntaxe

```
AddNodeAnimationTime(#NodeAnimation,
    Temps)
```

Description

Ajouter du temps à une animation.

Arguments

#**NodeAnimation** L'animation à utiliser.

Temps Le temps à ajouter (en millisecondes) à l'animation.

Valeur de retour

Aucune.

Voir aussi

StartNodeAnimation()

OS Supportés

Tous

124.15 StartNodeAnimation

Syntaxe

```
StartNodeAnimation(#NodeAnimation
    [, Options])
```

Description

Lance une animation.

Arguments

#**NodeAnimation** L'animation à utiliser.

Options

```
#PB_NodeAnimation_Once:
Ne joue l'animation
qu'une seule fois.
```

Valeur de retour

Aucune.

Remarques

L'animation démarre toujours depuis le début.
Par défaut, l'animation boucle automatiquement lorsque sa fin est atteinte. NodeAnimationStatus() peut être utilisé pour détecter la fin de l'animation.

Voir aussi

StopNodeAnimation() ,
NodeAnimationStatus() ,
AddNodeAnimationTime()

OS Supportés

Tous

124.16 StopNodeAnimation

Syntaxe

```
StopNodeAnimation(#NodeAnimation)
```

Description

Arrête une animation.

Arguments

#**NodeAnimation** L'animation à stopper.

Valeur de retour

Aucune.

Voir aussi

StartNodeAnimation()

OS Supportés

Tous

124.17 NodeAnimationStatus

Syntaxe

```
Resultat =
    NodeAnimationStatus(#NodeAnimation)
```

Description

Renvoie le statut d'une animation.

Arguments

#**NodeAnimation** L'animation à utiliser.

Valeur de retour

```
  #PB_NodeAnimation_Stopped:
   L'animation est à l'arrêt
   (ou a pris fin).
  #PB_NodeAnimation_Started:
   L'animation est en cours
   d'exécution.
```

Voir aussi

StartNodeAnimation() , StopNodeAnimation()

OS Supportés

Tous

124.18 GetNodeAnimationTime

Syntaxe

```
Resultat =
    GetNodeAnimationTime(#NodeAnimation)
```

Description

Renvoie le temps courant d'une animation.

Arguments

#**NodeAnimation** L'animation à utiliser.

Valeur de retour

Le temps actuel de l'animation (en millisecondes) ou zéro si l'animation ne fonctionne pas.

Voir aussi

StartNodeAnimation() ,
AddNodeAnimationTime() ,
SetNodeAnimationTime()

OS Supportés

Tous

124.19 SetNodeAnimationTime

Syntaxe

```
SetNodeAnimationTime(#NodeAnimation,
    Temps)
```

Description

Modifie le temps courant d'une animation .

Arguments

#**NodeAnimation** L'animation à utiliser.

Temps Le temps absolu (en millisecondes).

Valeur de retour

Aucune.

Remarques

Il s'agit d'une position temporelle absolue.
Pour changer le temps, utilisez
AddNodeAnimationTime() .

Voir aussi

StartNodeAnimation() ,
AddNodeAnimationTime() ,
GetNodeAnimationTime()

OS Supportés

Tous

124.20 GetNodeAnimationLength

Syntaxe

```
Duree =
    GetNodeAnimationLength(#NodeAnimation)
```

Description

Renvoie la durée d'une animation.

Arguments

#**NodeAnimation** L'animation à utiliser.

Valeur de retour

Renvoie la durée de l'animation (en
millisecondes).

Voir aussi

StartNodeAnimation() ,
SetNodeAnimationLength()

OS Supportés

Tous

124.21 SetNodeAnimationLength

Syntaxe

```
SetNodeAnimationLength(#NodeAnimation,
    Duree)
```

Description

Change la durée d'une animation.

Arguments

#**NodeAnimation** L'animation à utiliser.

Animation$ Le nom de l'animation.
Les animations sont stockées dans l'objet mesh de manière sensible à la casse (ex : "Marche" sera une animation différente de "marche").
Si l'animation n'est pas trouvée ou le mesh n'a pas de squelette alors la fonction n'aura aucun effet.

Duree La nouvelle durée de l'animation (en millisecondes).

Voir aussi

StartNodeAnimation() ,
GetNodeAnimationLength()

OS Supportés

Tous

124.22 GetNodeAnimationWeight

Syntaxe

```
Resultat.f =
    GetNodeAnimationWeight(#NodeAnimation)
```

Description

Renvoie le poids d'une animation.

Arguments

#**NodeAnimation** L'animation à utiliser.

Valeur de retour

Le poids actuel de l'animation (valeur comprise entre 0.0 et 1.0).

```
0 : L'animation est sans
  effet.
1 : L'animation joue
  pleinement.
```

Remarques

Le poids est utile lorsque vous jouez plusieurs animations à la fois.
Par exemple, pour faire une transition en douceur d'une animation à l'autre, il est possible de réduire progressivement le poids de la première animation et d'augmenter le poids de la deuxième animation.

Voir aussi

StartNodeAnimation()

OS Supportés

Tous

124.23 SetNodeAnimationWeight

Syntaxe

```
SetNodeAnimationWeight(#NodeAnimation,
    Poids.f)
```

Description

Modifie le poids d'une animation.

Arguments

#NodeAnimation L'animation à utiliser.

Poids Le nouveau poids de l'animation (valeur comprise entre 0.0 et 1.0).

```
0 : L'animation est sans
  effet.
1 : L'animation joue
  pleinement.
```

Valeur de retour

Aucune.

Remarques

Le poids est utile lorsque vous jouez plusieurs animations à la fois.
Par exemple, pour faire une transition en douceur d'une animation à l'autre, il est possible de réduire progressivement le poids de la première animation et d'augmenter le poids de la deuxième animation.

Voir aussi

StartNodeAnimation()

OS Supportés

Tous

Chapitre 125

OnError

Généralités

Cette bibliothèque permet de détecter les erreurs d'exécution (crashs de programmes) de manière similaire au débogueur PureBasic , mais sans le désavantage de la taille plus importante et de la vitesse d'exécution réduite inhérente à l'utilisation du débogueur. Cela permet aux programmes livrés à l'utilisateur final de collecter des informations importantes lors de dysfonctionnements et de les renvoyer au développeur.
Le débogueur PureBasic est quand même le meilleur outil pour trouver les bugs lors de la phase de développement, car il fournit des informations bien plus détaillées sur le programme (comme la valeur des variables) ainsi que des outils visuels pour trouver rapidement l'origine des bugs.
Note : Si cette bibliothèque et le débogueur PureBasic sont activés en même temps, toutes les erreurs ne seront pas détectées par la bibliothèque OnError, car des vérifications sont effectuées par le débogueur avant même que le code ne soit exécuté.
Cette bibliothèque fournit des informations sur le nom du code source et la ligne à laquelle l'erreur est survenue à l'aide des commandes ErrorFile() et ErrorLine() , seulement si cette fonctionnalité a été activée lors de la compilation du programme (car elle induit une légère pénalité dans la vitesse d'exécution du programme pour retenir le numéro de la ligne en cours d'exécution). Pour l'activer, il faut cocher "Activer le numéro de ligne pour OnError" dans les options de compilation ou spécifier /LINENUMBERING (Windows) ou –linenumbering (Linux, MacOS X) en ligne de commande .

OS Supportés

Tous

125.1 OnErrorExit

Syntaxe

```
OnErrorExit()
```

Description

Termine l'exécution du programme après n'importe quelle erreur gérée par OnError.

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Le système peut afficher une fenêtre d'erreur ou un message dans la console avant de quitter.
Pour quitter un programme de manière silencieuse (sans message système), utiliser OnErrorCall() et terminer le programme à partir du handler.

Exemple

```
1  MessageRequester("Test
     OnError", "Début Test.")
2
3  OnErrorExit()
4  Pokes(10, "Salut le
     Monde.") ; Déclenche une
     erreur de type
     #PB_OnError_InvalidMemory
5                                ;
     et le programme se termine
6  MessageRequester("Test
     OnError", "Ceci ne devrait
     jamais s'afficher.")
```

OS Supportés

Tous

125.2 OnErrorCall

Syntaxe

```
OnErrorCall(@ErrorHandler())
```

Description

Spécifie une procédure à exécuter si une erreur survient. La procédure peut afficher des informations à propos de l'erreur en utilisant les commandes de cette bibliothèque. Le programme se terminera automatiquement dès que la procédure quittera.

Arguments

@ErrorHandler() Adresse d'une procédure sous la forme :

```
1  Procedure ErrorHandler()
2  ; Votre code ici
3  EndProcedure
```

Valeur de retour

Aucune.

Exemple

```
1  Procedure GestionErreur()
2    MessageRequester("Test
     OnError", "L'erreur
     suivante est arrivée: " +
     ErrorMessage())
3  EndProcedure
4
5  MessageRequester("Test
     OnError", "Début Test.")
6
7  OnErrorCall(@GestionErreur())
8  Pokes(10, "Salut le
     Monde.") ; Provoque une
     erreur
     #PB_OnError_InvalidMemory
9
10 MessageRequester("Test
     OnError", "Ceci ne devrait
     jamais s'afficher.")
```

OS Supportés

Tous

125.3 OnErrorGoto

Syntaxe

```
OnErrorGoto(?Etiquette)
```

Description

Si une erreur survient, le programme saute à l'étiquette (label) donnée. Une fois dans le label, des informations à propos de l'erreur pourront être affichées en utilisant les commandes de cette bibliothèque.

Arguments

?Etiquette L'adresse de l'étiquette sur laquelle renvoyer le programme après une erreur.

Valeur de retour

Aucune.

Remarques

La pile du programme ne sera pas sauvegardée avant le saut vers le label, donc les variables locales ne seront plus accessibles. Il n'est pas recommandé de vouloir continuer l'exécution d'un programme après une erreur, car le contexte ne sera plus correct. La meilleure pratique est de rassembler un maximum d'informations à propos de l'erreur et de quitter le programme.

Exemple

```
1   MessageRequester("Test
     OnError", "Début Test.")
2
3   OnErrorGoto(?GestionErreurs)
4   Pokes(10, "Salut le
     Monde.") ; Provoque une
     erreur
     #PB_OnError_InvalidMemory
5
6   MessageRequester("Test
     OnError", "Ceci ne devrait
     jamais s'afficher.")
7   End
8
9   GestionErreurs:
10  MessageRequester("Test
     OnError", "L'erreur
     suivante est arrivée: " +
     ErrorMessage())
11  End
```

OS Supportés

Tous

125.4 OnErrorDefault

Syntaxe

```
OnErrorDefault()
```

Description

Réinitialise l'action à exécuter lors d'une erreur en utilisant celle du système par défaut. En général c'est une fenêtre d'erreur qui s'affiche suivie de la fin du programme, mais pas toujours (certaines erreurs peuvent être ignorées). Pour être sûr de quitter le programme sur chaque erreur, utilisez OnErrorExit() .

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Si la bibliothèque OnError est utilisée dans une dll , la meilleure approche est d'initialiser la gestion d'erreur au début de chaque fonction publique de la dll, et de la réinitialiser avec cette commande à la fin de la fonction (pour éviter les interférences entre la bibliothèque OnError et la gestion d'erreur du programme utilisant la dll).

Exemple

```
1  Procedure ErrorHandler()
2    MessageRequester("Test
     OnError", "L'erreur
     suivante est arrivée: " +
     ErrorMessage())
3  EndProcedure
4
5  MessageRequester("Test
     OnError", "Début Test.")
6
7  OnErrorCall(@ErrorHandler())
8  OnErrorDefault()
                ; Commenter
     cet appel pour obtenir le
     gestionnaire à la place de
     l'erreur envoyée par le
     système
```

```
9  Pokes(10, "Salut le Monde.") ; Provoque une erreur
   #PB_OnError_InvalidMemory
10
11 MessageRequester("Test OnError", "Ceci ne devrait jamais s'afficher.")
```

OS Supportés

Tous

125.5 ErrorCode

Syntaxe

```
Resultat = ErrorCode()
```

Description

Renvoie le code de l'erreur qui est survenue.

Arguments

Aucun.

Valeur de retour

Renvoie une des valeurs suivantes :

```
#PB_OnError_InvalidMemory
          : Lecture ou écriture dans une zone protégée.
#PB_OnError_Floatingpoint
          : Erreur de calcul flottant.
#PB_OnError_Breakpoint
             : Point d'arrêt débogueur atteint (autres que ceux du PureBasic).
#PB_OnError_IllegalInstruction
     : Exécution d'une instruction invalide.
#PB_OnError_PriviledgedInstruction:
 Exécution d'une instruction privilégiée (système) non autorisée.
#PB_OnError_DivideByZero
           : Division par zéro (Windows seulement).
```

Linux et MacOS X génèrent #PB_OnError_Floatingpoint pour les erreurs de division par zéro.
De plus, chaque OS peut avoir plus de valeurs. Sous Windows, des erreurs personnalisées peuvent être génèrées avec RaiseError() .

Remarques

Cette commande renvoie une valeur valide uniquement si l'erreur est gérée par OnErrorCall() ou OnErrorGoto() .

OS Supportés

Tous

125.6 ErrorMessage

Syntaxe

```
Resultat\$ =
    ErrorMessage([CodeErreur])
```

Description

Renvoie un message d'erreur en anglais.

Arguments

CodeErreur (optionnel) Un code d'erreur spécifique.
Voir ErrorCode() pour les codes d'erreur disponibles.

Valeur de retour

Renvoie un message d'erreur en anglais correspondant au code erreur specifié. Si le paramètre est omis, la fonction renvoie le message de l'erreur qui est survenue.

OS Supportés

Tous

125.7 ErrorLine

Syntaxe

```
Resultat = ErrorLine()
```

Description

Renvoie le numéro de ligne dans le code source où l'erreur est survenue.

Arguments

Aucun.

Valeur de retour

Renvoie le numéro de ligne dans le code source où l'erreur est survenue.
Cette commande renvoie une valeur valide uniquement si l'erreur est gérée par OnErrorCall() ou OnErrorGoto() .

Remarques

Le suivi des lignes en temps réel a besoin d'être activé à la compilation pour que cette fonction renvoie la ligne actuelle. Pour l'activer, il faut cocher "Activer le numéro de ligne pour OnError" dans les options de compilation ou spécifier /LINENUMBERING (Windows) ou –linenumbering (Linux, MacOS X) en ligne de commande . Si ce support est désactivé, cette fonction renverra -1.

OS Supportés

Tous

125.8 ErrorFile

Syntaxe

```
Resultat\$ = ErrorFile()
```

Description

Renvoie le nom du fichier du code source où l'erreur est survenue.

Arguments

Aucun.

Valeur de retour

Renvoie le nom du fichier du code source où l'erreur est survenue.
Cette commande renvoie une valeur valide uniquement si l'erreur est gérée par OnErrorCall() ou OnErrorGoto() .

Remarques

Le suivi des lignes en temps réel a besoin d'être activé à la compilation pour que cette fonction renvoie la ligne actuelle. Pour l'activer, il faut cocher "Activer le numéro de ligne pour OnError" dans les options de compilation ou spécifier /LINENUMBERING (Windows) ou –linenumbering (Linux, MacOS X) en ligne

de commande . Si ce support est désactivé, cette fonction renverra "OnError line support disabled".

OS Supportés

Tous

125.9 ErrorAddress

Syntaxe

```
Resultat = ErrorAddress()
```

Description

Renvoie l'adresse mémoire de l'instruction assembleur qui a causé l'erreur.

Arguments

Aucun.

Valeur de retour

Renvoie l'adresse mémoire de l'instruction assembleur qui a causé l'erreur.
Cette commande renvoie une valeur valide uniquement si l'erreur est gérée par OnErrorCall() ou OnErrorGoto() .

OS Supportés

Tous

125.10 ErrorTargetAddress

Syntaxe

```
Resultat =
    ErrorTargetAddress()
```

Description

Après une erreur de type #PB_OnError_InvalidMemory, cette commande renvoie l'adresse mémoire qui a été lue ou écrite quand l'erreur est survenue.

Arguments

Aucun.

Valeur de retour

Après une erreur de type #PB_OnError_InvalidMemory, cette commande renvoie l'adresse mémoire qui a été lue ou écrite quand l'erreur est survenue. Cette commande n'a pas d'utilité pour les autres types d'erreur.

OS Supportés

Tous

125.11 ErrorRegister

Syntaxe

```
Resultat =
    ErrorRegister(Registre)
```

Description

Renvoie le contenu du registre spécifié du microprocesseur lorsque l'erreur est survenue.

Arguments

Registre Le registre à afficher.
Les registres disponibles dépendent du CPU pour lequel le programme est compilé. Les valeurs suivantes sont disponibles :
x86 :

```
#PB_OnError_EAX
#PB_OnError_EBX
#PB_OnError_ECX
#PB_OnError_EDX
#PB_OnError_EBP
#PB_OnError_ESI
#PB_OnError_EDI
#PB_OnError_ESP
#PB_OnError_Flags
```

x64 :

```
#PB_OnError_RAX
#PB_OnError_RCX
#PB_OnError_RDX
#PB_OnError_RBX
#PB_OnError_RSP
#PB_OnError_RBP
#PB_OnError_RSI
#PB_OnError_RDI
#PB_OnError_R8
#PB_OnError_R9
...
#PB_OnError_R15
#PB_OnError_Flags
```

Valeur de retour

Renvoie le contenu du registre spécifié du microprocesseur lorsque l'erreur est survenue.
Cette commande renvoie une valeur valide uniquement si l'erreur est gérée par OnErrorCall() ou OnErrorGoto() .

OS Supportés

Tous

125.12 RaiseError

Syntaxe

```
RaiseError(CodeErreur)
```

Description

Génère artificiellement une erreur du type spéficié et lance une procédure de gestion d'erreur ou termine le programme si elle n'existe pas. Le 'CodeErreur' pourra être récupéré dans la procédure de gestion d'erreur avec la commande ErrorCode() .

Arguments

CodeErreur Sous Windows, n'importe quel numéro d'erreur peut être utilisé (y compris ceux définis par l'utilisateur), c'est un nombre entre 0 et 268435455 (un sur 27 bits). Sous Linux ou MacOS X, seuls les numéros d'erreur suivants sont acceptés :

```
#PB_OnError_InvalidMemory
          : Lecture ou
  écriture dans une zone
  protégée.
#PB_OnError_Floatingpoint
          : Erreur de
  calcul flottant.
#PB_OnError_Breakpoint
             : Point
  d'arrêt débogueur
  atteint (autres que ceux
  du PureBasic).
#PB_OnError_IllegalInstruction
     : Exécution d'une
  instruction invalide.
#PB_OnError_PriviledgedInstruction:
  Exécution d'une
  instruction privilégiée
  (système) non autorisée.
```

Exemple

```
Procedure ErrorHandler()
  MessageRequester("Test OnError", "L'erreur suivante est arrivée: " + ErrorMessage())
EndProcedure

MessageRequester("OnError test", "Début Test.")

OnErrorCall(@ErrorHandler())
RaiseError(#PB_OnError_InvalidMemory)

MessageRequester("Test OnError", "Ceci ne devrait jamais s'afficher.")
```

Voir aussi

ErrorCode()

OS Supportés

Tous

125.13 ExamineAssembly

Syntaxe

```
Resultat =
    ExamineAssembly(*MemoireDébut
    [, *MemoireFin])
```

Description

Désassemblage des instructions à partir d'une adresse mémoire.

Arguments

***MemoireDébut** L'adresse en mémoire de la première instruction à désassembler. Le désassemblage se déroulera jusqu'à ce que NextInstruction() ne soit plus appelé ou si *MemoireFin est spécifié.

***MemoireFin (optionnel)** L'adresse en mémoire de la dernière instruction à désassembler et NextInstruction() reverra zéro.

Valeur de retour

Renvoie une valeur non nulle si le désassemblage est possible, zéro sinon.

Remarques

Important : Les commandes de désassemblage utilisent la bibliothèque Udis86 disassembler pour décoder les instructions. Elle est disponible sous license BSD qui peut être consultée ici . Si ExamineAssembly() et ses autres commandes relatives sont utilisées dans un programme, le texte de licence ci-dessus devra être inclus avec le logiciel.

Exemple

```
1   DisableDebugger ; Désactive le Debugger
2
3   Code_Start:
4     ; Placer le code à désassembler ici
5     a = (Random(100) * 5) + 2000
6   Code_End:
7
8   Texte$ = "Code désassemblé:" + Chr(13)
9   If ExamineAssembly(?Code_Start, ?Code_End)
10    While NextInstruction()
11      Texte$ + RSet(Hex(InstructionAddress()), SizeOf(Integer)*2, "0")
12      Texte$ + " " + InstructionString() + Chr(13)
13    Wend
14  EndIf
15
16  MessageRequester("Resultat", Texte$)
```

OS Supportés

Tous

125.14 NextInstruction

Syntaxe

```
Resultat = NextInstruction()
```

Description

Désassemble l'instruction suivante, après avoir appelé ExamineAssembly() .

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle si l'instruction a pu être désassemblée correctement, zéro sinon ou s'il n'y a plus d'instruction à décoder ('*AdresseFin' spécifiée à ExamineAssembly() a été atteinte).

Remarques

Les informations à propos de l'instruction décodée sont disponibles à l'aide des commandes InstructionString() et InstructionAddress() .

Voir aussi

ExamineAssembly() , InstructionAddress() , InstructionString()

OS Supportés

Tous

125.15 InstructionAddress

Syntaxe

```
Resultat =
    InstructionAddress()
```

Description

Renvoie l'adresse de l'instruction qui a été désassemblée avec NextInstruction() .

Arguments

Aucun.

Valeur de retour

Renvoie l'adresse de l'instruction qui a été désassemblée avec NextInstruction() .

Voir aussi

NextInstruction() , InstructionString()

OS Supportés

Tous

125.16 InstructionString

Syntaxe

```
Resultat\$ =
    InstructionString()
```

Description

Renvoie la représentation sous forme de texte de l'instruction qui a été désassemblée avec NextInstruction() .

Arguments

Aucun.

Valeur de retour

Renvoie une chaîne de caractères représentant l'instruction qui a été désassemblée avec NextInstruction() .

Voir aussi

NextInstruction() , InstructionAddress()

OS Supportés

Tous

Chapitre 126

Packer

Généralités

La bibliothèque Packer propose un ensemble de fonctions efficaces pour compresser / décompresser des données et gérer les archives compressées.
Plusieurs formats de compressions sont pris en charge, de Zip à Lzma.

OS Supportés

Tous

126.1 AddPackFile

Syntaxe

```
Resultat =
    AddPackFile(#Archive,
    Fichier$,
    FichierCompresser$)
```

Description

Ajoute et compresse un fichier dans une archive préalablement créée avec CreatePack() .

Arguments

#**Archive** L'archive à utiliser.

Fichier$ Le fichier à ajouter à l'archive.

FichierCompresser$ Le nom de fichier à utiliser dans l'archive.

Valeur de retour

Renvoie une valeur non nulle si le fichier a été ajouté avec succès à l'archive.
Si le fichier ne peut pas être compressé, il sera stocké tel quel dans l'archive.

Remarques

L'ajout d'un grand fichier peut prendre un temps assez long.

Voir aussi

CreatePack()

OS Supportés

Tous

126.2 AddPackMemory

Syntaxe

```
Resultat =
    AddPackMemory(#Archive,
    *MemoireTampon, Longueur ,
    FichierCompresser$)
```

Description

Ajoute et compresse une zone mémoire tampon dans une archive préalablement créée avec la commande CreatePack() .

Arguments

#Archive L'archive à utiliser.

***MemoireTampon** L'adresse de la mémoire tampon dont le contenu sera ajouté à l'archive.

Longueur La taille de la mémoire à ajouter à l'archive.

FichierCompresser$ Le nom de fichier à utiliser dans l'archive.

Valeur de retour

Renvoie une valeur non nulle si le contenu de la mémoire tampon a été ajouté avec succès à l'archive.
Si le contenu de la mémoire tampon ne peut pas être compressé, il sera stocké tel quel dans l'archive.

Voir aussi

CreatePack()

OS Supportés

Tous

126.3 ClosePack

Syntaxe

```
ClosePack(#Archive)
```

Description

Ferme un fichier archive.

Arguments

#Archive L'archive à fermer. Si #PB_All est spécifié, toutes les archives restantes sont fermées.

Valeur de retour

Aucune.

Remarques

Tous les fichiers archives ouverts sont automatiquement fermés quand le programme se termine.

Voir aussi

CreatePack()

OS Supportés

Tous

126.4 CompressMemory

Syntaxe

```
Resultat =
    CompressMemory(*MemoireTampon,
    Taille, *Sortie,
    SortieTaille [, Plugin [,
    Niveau]])
```

Description

Compresse le contenu d'une mémoire tampon dans une mémoire tampon de sortie.

Arguments

***MemoireTampon** L'adresse de la mémoire tampon de compression.

Taille La taille de la mémoire à compresser.

***Sortie** L'adresse de la mémoire tampon qui contiendra les données compressées.

SortieTaille La taille de la mémoire tampon qui contiendra les données compressées.

Plugin (optionnel) Le plug-in à utiliser, si plus d'un plug-in a été enregistré. Il peut être une des valeurs suivantes :

```
  #PB_PackerPlugin_BriefLZ:
   Utilisation de BriefLZ.
   UseBriefLZPacker()
 doit être appelé auparavant.
  #PB_PackerPlugin_Zip    :
   Utilisation de Zip.
   UseZipPacker()
 doit être appelé auparavant.
  #PB_PackerPlugin_Lzma   :
   Utilisation de Lzma.
   UseLzmaPacker()
 doit être appelé auparavant.
```

Niveau (optionnel) Le niveau de compression à utiliser.
Une valeur entière allant de 0 (taux de compression inférieur, compression plus rapide) à 9 (taux de compression plus élevé, compression plus lente).

Valeur de retour

Renvoie la taille compressée si le contenu de la mémoire tampon a été correctement compressé, zéro sinon.
Si les données du tampon ne peuvent pas être compressées, il renverra zéro (ex : les données déjà compressées ne peuvent généralement pas l'être davantage).

Remarques

La longueur du tampon de sortie doit être au moins aussi grande que celle du tampon de compression.

Voir aussi

UncompressMemory()

OS Supportés

Tous

126.5 ExaminePack

Syntaxe

```
Resultat =
    ExaminePack(#Archive)
```

Description

Commence l'examen du contenu d'une archive.

Arguments

#**Archive** L'archive à utiliser.

Valeur de retour

Renvoie une valeur non nulle si l'archive peut être examinée, zéro sinon.

Remarques

NextPackEntry() doit être appelé pour lister les entrées dans le fichier archive.
L'archive doit être préalablement ouverte avec OpenPack() .
Les archives créées avec CreatePack() ne peuvent pas être examinées.

Exemple

```
1   UseZipPacker()
2
3   ; Ouvre le fichier compressé
4   If OpenPack(0,
     "Mesfichierscompresses.zip")
5
6     ; Liste toutes les entrées
7     If ExaminePack(0)
8       While NextPackEntry(0)
9         Debug "Nom: " +
     PackEntryName(0) + ",
     Taille: " +
     PackEntrySize(0)
10      Wend
11    EndIf
12
13    ClosePack(0)
14  EndIf
```

Voir aussi

OpenPack() , NextPackEntry()

OS Supportés

Tous

126.6 NextPackEntry

Syntaxe

```
Resultat =
    NextPackEntry(#Archive)
```

Description

Va à l'entrée suivante dans le fichier archive.

Arguments

#**Archive** L'archive à utiliser.

Valeur de retour

Renvoie une valeur non nulle si l'archive contient une autre entrée, zéro sinon.

Remarques

ExaminePack() doit être appelé auparavant. Pour obtenir plus d'informations sur l'entrée en cours, voir PackEntrySize() , PackEntryType() et PackEntryName() . Pour décompresser l'entrée courante, voir UncompressPackMemory() ou UncompressPackFile() .

Voir aussi

OpenPack() , PackEntrySize() , PackEntryType() , PackEntryName() , UncompressPackMemory() , UncompressPackFile()

OS Supportés

Tous

126.7 PackEntryType

Syntaxe

```
Resultat =
    PackEntryType(#Archive)
```

Description

Renvoie le type de l'entrée courante trouvée avec NextPackEntry() .

Arguments

#**Archive** L'archive à utiliser.

Valeur de retour

Renvoie une des valeurs suivantes :

```
#PB_Packer_File      :
 L'entrée courante est un
 fichier.
#PB_Packer_Directory:
 L'entrée courante est un
 répertoire.
```

Voir aussi

OpenPack() , NextPackEntry() , PackEntrySize() , PackEntryName() .

OS Supportés

Tous

126.8 PackEntrySize

Syntaxe

```
Resultat =
    PackEntrySize(#Archive [,
    Mode])
```

Description

Renvoie la taille de l'entrée courante trouvée avec NextPackEntry() .

Arguments

#**Archive** L'archive à utiliser.

Mode (optionnel) Le type de taille à obtenir.
Peut être une des valeurs suivantes :

```
#PB_Packer_UncompressedSize:
 Renvoie la taille non
 compressée de l'entrée
 courante (par défaut).
#PB_Packer_CompressedSize
 : Renvoie la taille
 compressée de l'entrée
 courante.
                                Seulement
 avec les archives
 BriefLZ.
```

Valeur de retour

Renvoie la taille de l'entrée courante.

Voir aussi

OpenPack() , NextPackEntry() , PackEntryType() , PackEntryName() .

OS Supportés

Tous

126.9 PackEntryName

Syntaxe

```
Resultat\$ =
    PackEntryName(#Archive)
```

Description

Renvoie le nom de l'entrée courante trouvée avec NextPackEntry() .

Arguments

#**Archive** L'archive à utiliser.

Valeur de retour

Renvoie le nom de l'entrée courante.

Voir aussi

OpenPack() , NextPackEntry() , PackEntryType() , PackEntrySize() .

OS Supportés

Tous

126.10 CreatePack

Syntaxe

```
Resultat =
    CreatePack(#Archive,
    Fichier$ [, Plugin [,
    Niveau]])
```

Description

Créé un nouveau fichier archive vide.

Arguments

#**Archive** Un numéro pour identifier le nouveau fichier archive.
#PB_Any peut être utilisé en tant que paramètre pour générer automatiquement ce numéro.

Fichier$ Le nom du nouveau fichier archive.

Plugin (optionnel) Le plug-in à utiliser, si plus d'un plug-in a été enregistré. Peut être une des valeurs suivantes :

```
#PB_PackerPlugin_BriefLZ:
  utilisation de BriefLZ.
  UseBriefLZPacker()
doit être déclaré
  auparavant.
#PB_PackerPlugin_Zip    :
  utilisation de Zip.
  UseZipPacker()
doit être déclaré
  auparavant.
#PB_PackerPlugin_Lzma   :
  utilisation de Lzma
  (archive 7z).
  UseLzmaPacker()
doit être déclaré
  auparavant.
#PB_PackerPlugin_Tar    :
  utilisation de Tar.
  UseTarPacker()
doit être déclaré
  auparavant.
                                Il
  peut être combiné avec
  #PB_Packer_Gzip ou
  #PB_Packer_Bzip2.
```

Niveau (optionnel) Le niveau de compression à utiliser.
Une valeur entière allant de 0 (taux de compression inférieur, compression plus rapide) à 9 (taux de compression plus élevé, compression plus lente).

Valeur de retour

Renvoie une valeur non nulle si le fichier archive a été créé avec succès, zéro sinon. Si #PB_Any a été utilisé comme paramètre #Archive alors c'est le numéro de l'archive généré est renvoyé.

```
1  UseZipPacker()
2
3  ; Créer le fichier compressé
4  If CreatePack(0,
    "MesFichiersCompresses.zip")
5
6    ; Ajouter vos fichiers
7    AddPackFile(0,
    "Image1.bmp", "Image1.bmp")
8    AddPackFile(0,
    "Image2.bmp", "Image2.bmp")
9    AddPackFile(0,
    "mywave1.wav",
    "mywave1.wav")
10   AddPackFile(0,
    "mywave2.wav",
    "mywave2.wav")
11   ClosePack(0)
12 EndIf
```

Voir aussi

ClosePack()

OS Supportés

Tous

126.11 OpenPack

Syntaxe

```
Resultat = OpenPack(#Archive,
    Fichier$ [, Plugin])
```

Description

Ouvre un fichier Archive.

Arguments

#Archive Un numéro pour identifier le fichier archive.
#PB_Any peut être utilisé en tant que paramètre pour générer automatiquement ce numéro.

Fichier$ Le nom du fichier archive à ouvrir.

Plugin (optionnel) Le plug-in à utiliser, si plus d'un plug-in a été enregistré. Il peut être une des valeurs suivantes :

```
  #PB_PackerPlugin_BriefLZ:
   Utilisation de BriefLZ.
   UseBriefLZPacker()
doit être appelé auparavant.
  #PB_PackerPlugin_Zip     :
   Utilisation de Zip.
   UseZipPacker()
doit être appelé auparavant.
  #PB_PackerPlugin_Lzma    :
   Utilisation de Lzma.
   UseLzmaPacker()
doit être appelé auparavant.
  #PB_PackerPlugin_Tar     :
   Utilisation de Tar.
   UseTarPacker()
doit être appelé auparavant.
```

Valeur de retour

Renvoie une valeur non nulle si le fichier archive a été ouvert avec succès, zéro sinon. Si #PB_Any a été utilisé comme paramètre #Archive alors c'est le numéro de l'archive généré qui est renvoyé.

Remarques

Avant d'ouvrir un fichier archive, au moins un Plug-in doit être enregistré avec l'une des commandes suivantes : UseZipPacker() , UseLzmaPacker() , , UseTarPacker() , UseBriefLZPacker() . Une fois ouvert, le contenu de l'archive peut être listé avec ExaminePack() .

```
1   UseZipPacker()
2
3   ; Ouvre le fichier compressé
4   If OpenPack(0,
     "MesFichiersCompresses.zip")
5
6     ; Liste toutes les entrées
7     If ExaminePack(0)
8       While NextPackEntry(0)
9         Debug "Nom: " +
     PackEntryName(0) + ",
     Taille: " +
     PackEntrySize(0)
10        Wend
11    EndIf
12
13    ClosePack(0)
14  EndIf
```

Voir aussi

ClosePack() , ExaminePack()

OS Supportés

Tous

126.12 UncompressMemory

Syntaxe

```
Resultat =
    UncompressMemory(*MemoireTampon,
    Taille, *Sortie,
    SortieTaille [, Plugin])
```

Description

Décompresse le contenu de la mémoire tampon dans la mémoire tampon de sortie.

Arguments

***MemoireTampon** L'adresse de la mémoire tampon à décompresser.

Taille La taille de la mémoire à décompresser.

***Sortie** L'adresse de la mémoire tampon qui contiendra les données non compressées.

SortieTaille La taille de la mémoire tampon qui contiendra les données non compressées.
Elle doit être au moins aussi grande que la taille des données non compressées.

Plugin (optionnel) Le plug-in à utiliser, si plus d'un plug-in a été enregistré. Peut être une des valeurs suivantes :

```
  #PB_PackerPlugin_Zip     :
   Utilisation de Zip.
   UseZipPacker()
 doit être appelé auparavant.
  #PB_PackerPlugin_Lzma    :
   Utilisation de Lzma.
   UseLzmaPacker()
 doit être appelé auparavant.
  #PB_PackerPlugin_Jcalg1  :
   Utilisation de Jcalg1.
   UseJCALG1Packer()
 doit être appelé auparavant.
  #PB_PackerPlugin_BriefLZ:
   Utilisation de BriefLZ.
   UseBriefLZPacker()
 doit être appelé auparavant.
```

Valeur de retour

Renvoie la taille décompressée si le tampon a été correctement décompressé, zéro sinon.

Remarques

La longueur du tampon de sortie doit être au moins aussi grande que celle de la mémoire tampon à décompresser.

Voir aussi

CompressMemory()

OS Supportés

Tous

126.13 UncompressPackMemory

Syntaxe

```
Resultat =
   UncompressPackMemory(#Archive,
   *MemoireTampon, Taille [,
   FichierCompresser$])
```

Description

Décompresse dans une mémoire tampon, l'entrée courante en cours d'examen avec ExaminePack() et NextPackEntry() .

Arguments

#**Archive** L'archive à utiliser.

***MemoireTampon** L'adresse de la mémoire tampon pour décompresser une entrée de l'archive.

Taille La taille de la mémoire pour décompresser une entrée de l'archive.

FichierCompresser$ (optionnel) L'entrée à décompresser.
Si ce paramètre n'est pas spécifié, l'entrée courante en cours d'examen avec ExaminePack() et NextPackEntry() est décompressée.

Valeur de retour

Renvoie la taille décompressée si l'entrée de l'archive a bien été décompressée dans la mémoire tampon, zéro sinon.

Voir aussi

OpenPack() , ExaminePack() , NextPackEntry()

OS Supportés

Tous

126.14 UncompressPackFile

Syntaxe

```
Resultat =
    UncompressPackFile(#Archive,
    Fichier$ [,
    FichierCompresser$])
```

Description

Décompresse dans le nom de fichier spécifié, l'entrée courante de l'archive en cours d'examen avec ExaminePack() et NextPackEntry() .

Arguments

#**Archive** L'archive à utiliser.

Fichier$ Le nom de fichier de décompression de l'entrée courante de l'archive.

FichierCompresser$ (optionnel)
L'entrée à décompresser.
Si ce paramètre n'est pas spécifié, l'entrée courante examinée avec ExaminePack() et NextPackEntry() est décompressée.

Valeur de retour

Renvoie la taille décompressée si l'entrée de l'archive a bien été décompressée dans le nom du fichier, zéro sinon.

Remarques

Si le nom de fichier existe déjà, il sera effacé et remplacé par les nouvelles données non compressées.

Voir aussi

OpenPack() , ExaminePack() , NextPackEntry()

OS Supportés

Tous

126.15 UseZipPacker

Syntaxe

```
UseZipPacker()
```

Description

Active le compresseur, le décompresseur et le support des archives Zip.

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

L'archive créée sera compatible avec d'autres archives Zip au format 2.0.
La taille de l'archive créée peut aller jusqu'à 2GB.
Pour plus d'information : Wikipedia - Zip.

Voir aussi

OpenPack() , CreatePack() , CompressMemory() , UncompressMemory()

OS Supportés

Tous

126.16 UseLzmaPacker

Syntaxe

```
UseLzmaPacker()
```

Description

Active le compresseur, le décompresseur Lzma et le support des archives 7z.

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

La compression Lzma est considérée comme l'un des meilleurs algorithmes de compression disponible et polyvalent.
Il offre un très bon rapport compression / vitesse.
La compression peut être lente.
Pour plus d'informations : Wikipedia - Lzma.

Voir aussi

OpenPack() , CompressMemory() , UncompressMemory()

OS Supportés

Tous

126.17 UseTarPacker

Syntaxe

```
UseTarPacker()
```

Description

Active la compression, la décompression et le support des archives Tar.

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

La compression et la décompression des archives Tar sont généralement rapides.
Les compressions Bzip2 et Gzip sont toutes deux supportées.

Voir aussi

OpenPack() , CompressMemory() , UncompressMemory()

OS Supportés

Tous

126.18 UseBriefLZPacker

Syntaxe

```
UseBriefLZPacker()
```

Description

Active le compresseur, le décompresseur et le support des archives BriefLZ.

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Les archives créées sont un format personnalisé créé pour PureBasic.
La compression BriefLZ est très rapide et le fichier produit est très petit, il pourrait être le bon choix pour un programme qui a besoin d'une petite taille de l'exécutable.
La décompression est également très rapide, mais le rapport compression/vitesse n'est pas aussi bon qu'avec Zip ou Lzma.

Voir aussi

OpenPack() , CreatePack() , CompressMemory() , UncompressMemory()

OS Supportés

Tous

126.19 UseJCALG1Packer

Syntaxe

```
UseJCALG1Packer()
```

Description

Active le décompresseur JCALG1.

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Il s'agit d'un ancien algorithme qui a été utilisé dans les versions précédentes de PureBasic, il est toujours disponible pour permettre la compatibilité avec les anciens fichiers compressés.
La compression et l'archivage ne sont plus disponibles.
Il est disponible uniquement sous Windows x86 (32-bit).
Il est obsolète et n'est plus pris en charge.

Voir aussi

UncompressMemory()

OS Supportés

Windows

Chapitre 127

Particle

Généralités

Les systèmes de particules sont largement utilisés dans les scènes 3D pour simuler des objets aux comportements non prédictibles tels que la pluie, le feu, des explosions etc. PureBasic permet de créer un nombre quelconque d'émetteurs de particules fonctionnant de manière autonome. Chaque émetteur a sa forme propre et ses propriétés propres comme la vélocité, la vitesse, le taux d'émission et plus encore.
Les particules sont des plans en 2D (Billboards) qui font toujours face à la caméra et qui ont toutes la même taille.
InitEngine3D() doit être appelé avec succès avant l'utilisation des commandes Particle.

OS Supportés

Tous

127.1 CreateParticleEmitter

Syntaxe

```
Resultat =
    CreateParticleEmitter(#EmetteurParticule,
    Largeur, Hauteur,
    Profondeur, Type, [, X.f,
    Y.f, Z.f])
```

Description

Crée un émetteur de particules.

Arguments

#**EmetteurParticule** L'émetteur de particules à utiliser.

Le nouvel émetteur de particules. #PB_Any peut être utilisé pour générer automatiquement ce numéro.

Largeur, Hauteur, Profondeur
Dimensions du nouvel émetteur de particules.
La taille par défaut des futures particules peuvent être spécifiées avec la commande ParticleSize() .

Type
```
#PB_Particle_Point:
    L'émetteur est un point unique
#PB_Particle_Box  :
    L'émetteur est une boîte, avec une largeur, une hauteur et une profondeur
```

X.f, Y.f, Z.f (optionnel) Position du nouvel émetteur de particules.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any est utilisé pour le paramètre '#EmetteurParticule', le numéro de l'émetteur de particules sera renvoyé dans 'Resultat'.

Remarques

Si #EmetteurParticule a déjà été créé, sa place mémoire est automatiquement libérée et il est remplacé par le nouvel émetteur de particules.

Voir aussi

FreeParticleEmitter() , IsParticleEmitter() , ParticleEmitterID()

OS Supportés

Tous

127.2 IsParticleEmitter

Syntaxe

```
Resultat =
    IsParticleEmitter(#EmetteurParticule)
```

Description

Teste si un émetteur de particules est correctement initialisé.

Arguments

#EmetteurParticule L'émetteur de particules à utiliser.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.

Voir aussi

CreateParticleEmitter() , ParticleEmitterID()

OS Supportés

Tous

127.3 DisableParticleEmitter

Syntaxe

```
DisableParticleEmitter(#EmetteurParticule,
    Etat)
```

Description

Active ou désactive un émetteur de particules

Arguments

#**EmetteurParticule** L'émetteur de particules à utiliser.

Etat Peut prendre l'une des valeurs suivantes :

```
#True : Activé.
#False: Désactivé.
```

Valeur de retour

Aucune.

Voir aussi

CreateParticleEmitter()

OS Supportés

Tous

127.4 ParticleEmitterID

Syntaxe

```
Resultat =
    ParticleEmitterID(#EmetteurParticule)
```

Description

Renvoie l'identifiant unique d'un émetteur de particules.

Arguments

#**EmetteurParticule** L'émetteur de particules à utiliser.

Valeur de retour

Renvoie le numéro de l'émetteur.

Remarques

Cette fonction est très utile quand une fonction d'une autre bibliothèque nécessite l'identifiant d'un émetteur de particules.

Voir aussi

CreateParticleEmitter() , IsParticleEmitter()

OS Supportés

Tous

127.5 ParticleEmitterX

Syntaxe

```
Resultat =
    ParticleEmitterX(#EmetteurParticule
    [, Mode]))
```

Description

Renvoie la position en X dans le monde 3D d'un émetteur de particules.

Arguments

#**EmetteurParticule** L'émetteur de particules à utiliser.

Mode (optionnel) Peut prendre l'une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 position 'X' absolue
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 position 'X' par rapport
 à son parent.
```

Valeur de retour

Renvoie la position en X de l'émetteur, en fonction du mode choisi.

Voir aussi

ParticleEmitterY() , ParticleEmitterZ()

OS Supportés

Tous

127.6 ParticleEmitterY

Syntaxe

```
Resultat =
    ParticleEmitterY(#EmetteurParticule
    [, Mode]))
```

Description

Renvoie la position en Y dans le monde 3D d'un émetteur de particules.

Arguments

#**EmetteurParticule** L'émetteur de particules à utiliser.

Mode (optionnel) Peut être l'une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 position 'Y' absolue
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 position 'Y' par rapport
 à son parent.
```

Valeur de retour

Renvoie la position en Y de l'émetteur, en fonction du mode choisi.

Voir aussi

ParticleEmitterX() , ParticleEmitterZ()

OS Supportés

Tous

127.7 ParticleEmitterZ

Syntaxe

```
Resultat =
    ParticleEmitterZ(#EmetteurParticule
    [, Mode]))
```

Description

Renvoie la position en Z dans le monde 3D d'un émetteur de particules.

Arguments

#**EmetteurParticule** L'émetteur de particules à utiliser.

Mode (optionnel) Peut être l'une des valeurs suivantes :

```
#PB_Absolute: Renvoie la
 position 'Z' absolue
 dans le monde (par
 défaut).
#PB_Relative: Renvoie la
 position 'Z' par rapport
 à son parent.
```

Valeur de retour

Renvoie la position en Z de l'émetteur, en fonction du mode choisi.

Voir aussi

ParticleEmitterX() , ParticleEmitterY()

OS Supportés

Tous

127.8 ParticleEmitterAngle

Syntaxe

```
ParticleEmitterAngle(#EmetteurParticule,
    Angle.f)
```

Description

Modifie les angles des particules émises.
Toutes les particules auront le même angle.

Arguments

#**EmetteurParticule** L'émetteur de particules à utiliser.

Angle.f Le nouvel angle des particules, en degré.

Valeur de retour

Aucune.

OS Supportés

Tous

127.9 ParticleEmissionRate

Syntaxe

```
ParticleEmissionRate(#EmetteurParticule,
    Taux)
```

Description

Change le taux d'émission.

Arguments

#**EmetteurParticule** L'émetteur de particules à utiliser.

Taux Nouveau taux d'émission, en particules par seconde.

Valeur de retour

Aucune.

Voir aussi

ParticleEmitterDirection() , ParticleSize() , ParticleTimeToLive() , ParticleSpeedFactor()

OS Supportés

Tous

127.10 ParticleMaterial

Syntaxe

```
ParticleMaterial(#EmetteurParticule,
    MatiereID)
```

Description

Assigne une matière.

Arguments

#**EmetteurParticule** L'émetteur de particules à utiliser.

MatiereID Le numéro de la matière. Peut être facilement obtenu avec la commande MaterialID() .

Valeur de retour

Aucune.

Remarques

Cette matière sera utilisée par toutes les particules de cet émetteur.
Un émetteur peut seulement avoir une matière assignée à la fois.

Voir aussi

GetScriptParticleEmitter() , MaterialID()

OS Supportés

Tous

127.11 ParticleTimeToLive

Syntaxe

```
ParticleTimeToLive(#EmetteurParticule,
    TempsMininum, TempsMaximum)
```

Description

Change la durée de vie des particules.

Arguments

#**EmetteurParticule** L'émetteur de particules à utiliser.

TempsMininum, TempsMaximum Durée de vie minimum et maximum de chaque particule dans l'unité de temps du monde 3D.
La durée de vie des particules est une valeur aléatoire à l'intérieur de cette plage.

Valeur de retour

Aucune.

Voir aussi

ParticleVelocity() , ParticleEmissionRate() ParticleEmitterDirection() , ParticleSize() , ParticleSpeedFactor()

OS Supportés

Tous

127.12 ParticleVelocity

Syntaxe

```
ParticleVelocity(#EmetteurParticule,
    Mode, Valeur)
```

Description

Change la vitesse des particules.

Arguments

#**EmetteurParticule** L'émetteur de particules à utiliser.

Mode Peut être une des valeurs suivantes :

```
#PB_Particle_MinimumVelocity
 : Vélocité minimum (une
 valeur aléatoire sera
 choisie entre mini et
 maxi).
#PB_Particle_MaximumVelocity
 : Vélocité maximum (une
 valeur aléatoire sera
 choisie entre mini et
 maxi).
#PB_Particle_Velocity
        : Vélocité
 constante.
```

Valeur La valeur à donner au mode.

Valeur de retour

Aucune.

Voir aussi

ParticleTimeToLive() , ParticleEmitterDirection() , ParticleSize() , ParticleSpeedFactor() , ParticleEmissionRate()

OS Supportés

Tous

127.13 ParticleAcceleration

Syntaxe

```
ParticleAcceleration(#EmetteurParticule,
    X.f, Y.f, Z.f)
```

Description

Modifie le vecteur accélération des particules. Peut être utile pour simuler la gravité, le vent, etc.

Arguments

#**EmetteurParticule** L'émetteur de particules à utiliser.

X.f, Y.f, Z.f Le vecteur accélération (force) à appliquer à toutes les particules.

Valeur de retour

Aucune.

OS Supportés

Tous

127.14 ParticleSize

Syntaxe

```
ParticleSize(#EmetteurParticule,
    Largeur, Hauteur)
```

Description

Change les dimensions des particules.

Arguments

#**EmetteurParticule** L'émetteur de particules à utiliser.

Largeur, Hauteur Nouvelles dimensions des particules.

Valeur de retour

Aucune.

Remarques

Les particules sont des plans en 2D (Billboards) qui font toujours face à la Caméra.
Toutes les particules d'un même émetteur ont toujours la même taille.

Voir aussi

ParticleEmissionRate() ,
ParticleEmitterDirection()
ParticleTimeToLive() ,
ParticleEmitterDirection() ,
ParticleSpeedFactor() ,
ParticleEmissionRate()

OS Supportés

Tous

127.15 ParticleColorRange

Syntaxe

```
ParticleColorRange(#EmetteurParticule,
    PremiereCouleur,
    DerniereCouleur)
```

Description

Change la plage de couleurs des particules.

Arguments

#**EmetteurParticule** L'émetteur de particules à utiliser.

PremiereCouleur, DerniereCouleur Chaque particule émise obtiendra une valeur aléatoire dans la plage située entre 'PremiereCouleur' et 'DerniereCouleur'. La commande RGB() peut être utilisée pour obtenir rapidement une couleur.

Valeur de retour

Aucune.

Voir aussi

ParticleColorFader()

OS Supportés

Tous

127.16 ParticleColorFader

Syntaxe

```
ParticleColorFader(#EmetteurParticule,
    TauxRouge.f, TauxVert.f,
    TauxBleu.f, TauxAlpha.f))
```

Description

Change le niveau d'intensité des couleurs des particules (fader).

Arguments

#**EmetteurParticule** L'émetteur de particules à utiliser.

TauxRouge.f, TauxVert.f, TauxBleu.f, TauxAlpha.f Taux de chacune des couleurs Rouge, Vert, Bleu et de la transparence (alpha), par seconde. Exemples de valeurs et leurs effets :

```
     0: Aucun changement ne
   sera appliqué
    -1: Soustraira 256 à la
   composante couleur
   toutes les secondes.
     1: Ajoutera 256 à la
   composante couleur
   toutes les seconde.
 -0.25: Soustraira 64 à la
   composante couleur
   toutes les secondes.
```

Valeur de retour

Aucune.

Voir aussi

ParticleColorRange()

OS Supportés

Tous

127.17 FreeParticleEmitter

Syntaxe

```
FreeParticleEmitter(#EmetteurParticule)
```

Description

Supprime et libère la mémoire associée à un émetteur de particules.

Arguments

#**EmetteurParticule** L'émetteur de particules à supprimer.
Si #PB_All est spécifié, tous les émetteurs de particules restants sont libérés.

Valeur de retour

Aucune.

Remarques

Tous les émetteurs de particules restants sont automatiquement supprimés quand le programme se termine.

Voir aussi

CreateParticleEmitter()

OS Supportés

Tous

127.18 HideParticleEmitter

Syntaxe

```
HideParticleEmitter(#EmetteurParticule,
    Etat)
```

Description

Cache ou affiche un émetteur de particules.

Arguments

#**EmetteurParticule** L'émetteur de particules à utiliser.

Etat #True : L'émetteur est caché
#False: L'émetteur est affiché

Valeur de retour

Aucune.

Voir aussi

CreateParticleEmitter()

OS Supportés

Tous

127.19 MoveParticleEmitter

Syntaxe

```
MoveParticleEmitter(#EmetteurParticule,
    X.f, Y.f, Z.f [, Mode])
```

Description

Déplace un émetteur de particules dans le monde 3D.

Arguments

#**EmetteurParticule** L'émetteur de particules à déplacer.

X.f, Y.f, Z.f Valeur du déplacement. La nouvelle position de l'émetteur de particules.

Mode (optionnel) Peut être une des valeurs suivantes :

```
#PB_Relative: Déplacement
 relatif (par défaut).
#PB_Absolute: Déplacement
 absolu à la position
 spécifiée.
```

combinée avec l'une des valeurs suivantes :

```
#PB_Local : Déplacement
 local.
#PB_Parent: Déplacement
 par rapport à la
 position du parent.
#PB_World : Déplacement
 par rapport au monde.
```

Valeur de retour

Aucune.

Voir aussi

ParticleEmitterX() , ParticleEmitterY() , ParticleEmitterZ()

OS Supportés

Tous

127.20 ParticleEmitterDirection

Syntaxe

```
ParticleEmitterDirection(#EmetteurParticule,
   X.f, Y.f, Z.f)
```

Description

Change la direction d'émission de particules.

Arguments

#**EmetteurParticule** L'émetteur de particules à utiliser.

X.f, Y.f, Z.f L'émetteur pointera vers le point désigné par les coordonnées X,Y et Z.

Valeur de retour

Aucune.

Voir aussi

ParticleEmissionRate()
ParticleTimeToLive() ,
ParticleEmitterDirection() ,
ParticleSpeedFactor() ,
ParticleEmissionRate()

OS Supportés

Tous

127.21 ResizeParticleEmitter

Syntaxe

```
ResizeParticleEmitter(#Particle,
    #EmetteurParticule,
    Largeur, Hauteur,
    Profondeur)
```

Description

Redimensionne un émetteur de particules

Arguments

#**EmetteurParticule** L'émetteur de particules à utiliser.

Largeur, Hauteur, Profondeur Nouvelles dimensions de l'émetteur.

Valeur de retour

Aucune.

Voir aussi

MoveParticleEmitter() , ParticleSize()

OS Supportés

Tous

127.22 GetScriptParticleEmitter

Syntaxe

```
Resultat =
    GetScriptParticleEmitter(#EmetteurParticule,
    Nom$)
```

Description

Obtenir un émetteur de particules défini dans un fichier de script OGRE.

Arguments

#**EmetteurParticule** Le numéro du nouvel émetteur de particules à trouver. #PB_Any peut être utilisé pour générer automatiquement ce numéro.

Nom$ Le nom de l'émetteur de particules dans le fichier de script OGRE. Ce nom est sensible à la casse.

Valeur de retour

Renvoie zéro si l'émetteur de particules est introuvable ou ne peut pas être chargé à partir des fichiers de script.
Si #PB_Any a été utilisé alors le numéro auto-généré est renvoyé en cas de succès.

Valeur de retour

Aucune.

Voir aussi

CreateParticleEmitter()

OS Supportés

Tous

127.23 ParticleSpeedFactor

Syntaxe

```
ParticleSpeedFactor(#EmetteurParticule,
    FacteurVitesse.f)
```

Description

Modifie la vitesse d'émission.

Arguments

#**EmetteurParticule** L'émetteur de particules à utiliser.

FacteurVitesse.f Le nouveau facteur vitesse de l'émetteur de particules. Le facteur de vitesse par défaut est 1. Lorsque vous utilisez une valeur supérieure à 1 alors la vitesse d'émission est augmentée. Lors de l'utilisation d'une valeur inférieure à 1, la vitesse d'émission est diminuée.
Par exemple, un facteur vitesse de 4.5, multipliera la vitesse d'émission des particulese par 4,5. L'utilisation d'un facteur de vitesse de 0.5, divisera par deux la vitesse d'émission des particules .

Valeur de retour

Aucune.

Voir aussi

ParticleEmissionRate()
ParticleTimeToLive() ,
ParticleEmitterDirection() ,
ParticleEmissionRate()

OS Supportés

Tous

127.24 ParticleScaleRate

Syntaxe

```
ParticleScaleRate(#EmetteurParticule,
    TauxTailleParticule.f)
```

Description

Modifie la taille des particules au cours du temps.

Arguments

#**EmetteurParticule** L'émetteur de particules à utiliser.

TauxTailleParticule Le nouveau taux d'accroissement ou de réduction de la taille des particules au cours du temps.
Le taux par défaut est 0, ce qui signifie que la taille des particules ne varie pas au cours du temps .
Une valeur supérieure à 0 augmente la taille des particules au cours du temps (chaque seconde, la taille des particules sera augmentée du taux).
Une valeur inférieure à 0 réduit la taille des particules au cours du temps (chaque seconde, la taille des particules sera soustraite du taux).
Par exemple :
Un taux de 10 augmentera la taille de particules de 10 unités par seconde.
Un taux de -2 réduira la taille de particules de 2 unités par seconde.

Valeur de retour

Aucune.

OS Supportés

Tous

127.25 ParticleAngle

Syntaxe

```
ParticleAngle(#EmetteurParticule,
    AngleDepart.f, AngleFin.f
    [, VitesseAngleDepart.f,
    VitesseAngleFin.f])
```

Description

Modifie l'angle des particules une fois émisent.

Arguments

#EmetteurParticule L'émetteur de particules à utiliser.

AngleDepart.f, AngleFin.f La plage d'émission des particules (en degré), entre 0 et 360.
Par exemple, une plage de 45 à 90 n'émettra que des particules avec un angle aléatoire compris entre 45 et 90 degrés.

VitesseAngleDepart.f, VitesseAngleFin.f Spécifie la plage de vitesse de rotation des particules.
Une vitesse de rotation aléatoire sera choisie dans cette plage lorsqu'une nouvelle particule est émise.

Remarques

Pour éviter des problèmes d'affichage, le matériau de la particule doit être configuré pour être non répétable à l'aide de SetMaterialAttribute(Matiere, #PB_Material_TAM, #PB_Material_ClampTAM).

Valeur de retour

Aucune.

OS Supportés

Tous

Chapitre 128

Preference

Généralités

Un fichier "Préférence" contient des paramètres définis par l'utilisateur pour le programme. Il est stocké sur un disque et chargé à chaque lancement de ce programme, comme les fichiers '.INI' sous Windows par exemple. PureBasic offre la possibilité de créer un fichier de préférences hiérarchisées sous forme de groupes et facilement utilisable sur différentes plate-formes. Le format du fichier est en mode unicode UTF-8 avec BOM avec une préférence par ligne, utilisant la syntaxe 'Clé = Valeur'. Des groupes peuvent être créés pour une lecture plus simple.

OS Supportés

Tous

128.1 ClosePreferences

Syntaxe

```
ClosePreferences()
```

Description

Ferme un fichier de préférences préalablement ouvert avec la commande OpenPreferences() ou créé avec CreatePreferences() .

Arguments

Aucun.

Valeur de retour

Aucune.

Exemple

```
; Ouvrir un fichier préférence
OpenPreferences(#PB_Compiler_Home +"Examples\Sources\Data\test.pref")

; Choisir le groupe "Window"
PreferenceGroup("Window")

; Afficher la clé "w" de ce groupe
MessageRequester("Info", "w = " + ReadPreferenceLong("w", 0))

; Fermer le fichier préférence
ClosePreferences()
```

Voir aussi

CreatePreferences() , OpenPreferences()

OS Supportés

Tous

128.2 CreatePreferences

Syntaxe

```
Resultat = CreatePreferences(Fichier$ [, Options])
```

Description

Crée un nouveau fichier de préférences vide.
Si le fichier existe déjà, il est écrasé.

Arguments

Fichier$ Le nom du nouveau fichier de préférences.

Options (optionnel) Peut être une combinaison des valeurs suivantes :

```
#PB_Preference_NoSpace : Pas d'espaces autour du signe égal. Utile quand il s'agit d'un fichier de préférences externes qui n'accepte pas les espaces autour du signe égal.
```

```
#PB_Preference_GroupSeparator:
 Ajoute une ligne vide
 entre les groupes pour
 faciliter la lisibilité.
```

Valeur de retour

Renvoie une valeur non nulle si le fichier a été créé avec succès, zéro sinon.

Remarques

Une fois le fichier créé, les commandes telles que WritePreferenceString() par exemple permettent d'écrire des informations dedans. Une fois les valeurs écrites, il est possible de les relire avec ReadPreferenceString() ou une commande ad hoc.
Pour effacer une clé ou un groupe, utiliser RemovePreferenceKey() ou RemovePreferenceGroup() .
PreferenceGroup() permet de créer ou changer le groupe courant.
Une fois les opérations sur le fichier terminées, il faut utiliser ClosePreferences() pour mettre à jour, écrire et fermer le fichier préférences sur le disque.

Exemple

```
1  ; Créer un fichier
    preference nommé Setup.ini
2  CreatePreferences(GetTemporaryDirectory()+"Setup.ini")
3
4  ; Créer un groupe nommé
    "Window"
5  PreferenceGroup("Window")
6    WritePreferenceLong ("X",
    10)      ; X = 10
7    WritePreferenceLong ("Y",
    10)      ; Y = 10
8    WritePreferenceLong("W",
    800)      ; W = 800
9    WritePreferenceLong("H",
    600)      ; H = 600
10   WritePreferenceFloat("%",
    20)      ; % = 20.000000
11   WritePreferenceString("Titre",
    "PureNote") ; Titre =
    "PureNote"
12
13 ; Fermer le fichier
    préférence
14 ClosePreferences()
```

Voir aussi

ClosePreferences()

OS Supportés

Tous

128.3 ExaminePreferenceGroups

Syntaxe

```
Resultat =
    ExaminePreferenceGroups()
```

Description

Commence l'énumération de tous les groupes trouvés dans le fichier de préférence courant.

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

La commande NextPreferenceGroup() permet de passer au groupe suivant.

Exemple

```
1   ; Ouvrir un fichier
     préférence
2   OpenPreferences(#PB_Compiler_Home
     +"Examples\Sources\Data\test.pref")
3
4   ; Recherche des groupes
     dans le fichier préférence
5   ExaminePreferenceGroups()
6   While NextPreferenceGroup()
     ; Tant qu'il existe un
     groupe à afficher
7     MessageRequester("Groupes",
     PreferenceGroupName())   ;
     Afficher ce groupe
8   Wend
9
10  ; Fermer le fichier
     préférence
11  ClosePreferences()
```

Voir aussi

NextPreferenceGroup() ,
PreferenceGroupName()

OS Supportés

Tous

128.4 ExaminePreferenceKeys

Syntaxe

```
Resultat =
    ExaminePreferenceKeys()
```

Description

Commence l'énumération de toutes les clés trouvées dans le fichier de préférence courant.

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

La commande NextPreferenceKey() permet de passer à la clé suivante.

Exemple

```
1   ; Ouvrir un fichier
     préférence
2   OpenPreferences(#PB_Compiler_Home
     +"Examples\Sources\Data\test.pref")
3
4   ; Groupe à examiner
5   PreferenceGroup("Window")
6
7   ; Recherche des clés dans
     le groupe 'Window'
8   ExaminePreferenceKeys()
9   While  NextPreferenceKey()
     ; Tant qu'il existe une
     clé à afficher
10   MessageRequester("Clés du
     groupe 'Window'",
     PreferenceKeyName() + " =
     " + PreferenceKeyValue())
     ; Afficher la clé et sa
     valeur
```

```
11  Wend
12
13  ; Fermer le fichier
      préférence
14  ClosePreferences()
```

Voir aussi

ExaminePreferenceGroups() ,
NextPreferenceGroup() ,
NextPreferenceKey() ,
PreferenceKeyName() (),
PreferenceKeyValue() ()

OS Supportés

Tous

128.5 FlushPreferenceBuffers

Syntaxe

```
Resultat =
    FlushPreferenceBuffers()
```

Description

Veille à ce que toutes les modifications des préférences soient écrites sur le disque.

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle si les préférences ont été écrites sur le disque avec succès, zéro si une erreur survient (disque plein, erreur disque, etc).

Voir aussi

CreatePreferences() , OpenPreferences()

OS Supportés

Tous

128.6 NextPreferenceGroup

Syntaxe

```
Resultat =
    NextPreferenceGroup()
```

Description

Renvoie le groupe suivant.

Arguments

Aucun.

Valeur de retour

Récupère les informations concernant le groupe suivant dans l'énumération commencée avec ExaminePreferenceGroups() .

Remarques

Pour savoir le nom du groupe courant, utiliser PreferenceGroupName() .

Exemple

```
; Ouvrir un fichier préférence
OpenPreferences(#PB_Compiler_Home +"Examples\Sources\Data\test.pref")

; Recherche des groupes dans le fichier préférence
ExaminePreferenceGroups()
While NextPreferenceGroup() ; Tant qu'il existe un groupe à afficher
  MessageRequester("Groupes", PreferenceGroupName())  ; Afficher ce groupe
Wend

; Fermer le fichier préférence
ClosePreferences()
```

Voir aussi

ExaminePreferenceGroups() , PreferenceGroupName() , ExaminePreferenceKeys()

OS Supportés

Tous

128.7 NextPreferenceKey

Syntaxe

```
Resultat = NextPreferenceKey()
```

Description

Renvoie la clé suivante.

Arguments

Aucun.

Valeur de retour

Récupère les informations concernant la clé suivante dans l'énumération commencée avec ExaminePreferenceKeys() .

Remarques

Pour savoir le nom et la valeur de la clé courante, utiliser PreferenceKeyName() et PreferenceKeyValue() .

Exemple

```
; Ouvrir un fichier préférence
OpenPreferences(#PB_Compiler_Home +"Examples\Sources\Data\test.pref")

; Groupe à examiner
PreferenceGroup("Window")

; Recherche des clés dans le groupe 'Window'
ExaminePreferenceKeys()
While  NextPreferenceKey() ; Tant qu'il existe une clé à afficher
  MessageRequester("Clés du groupe 'Window'", PreferenceKeyName() + " = " + PreferenceKeyValue()) ; Afficher la clé et sa valeur
Wend

; Fermer le fichier préférence
ClosePreferences()
```

Voir aussi

ExaminePreferenceKeys() ,
PreferenceKeyName() ,
PreferenceKeyValue()

OS Supportés

Tous

128.8 PreferenceGroupName

Syntaxe

```
Resultat\$ =
    PreferenceGroupName()
```

Description

Renvoie le nom du groupe courant.

Arguments

Aucun.

Valeur de retour

Renvoie le nom du groupe courant dans l'énumération commencée avec ExaminePreferenceGroups() ou préalablement sélectionnée avec PreferenceGroup() .

Exemple

```
; Ouvrir un fichier préférence
OpenPreferences(#PB_Compiler_Home+"Examples\Sources\Data\test.pref")

; Recherche des groupes dans le fichier préférence
ExaminePreferenceGroups()
While NextPreferenceGroup() ; Tant qu'il existe un groupe à afficher
  MessageRequester("Groupes", PreferenceGroupName())  ; Afficher ce groupe
Wend

; Fermer le fichier préférence
ClosePreferences()
```

Voir aussi

ExaminePreferenceGroups()

OS Supportés

Tous

128.9 PreferenceKeyName

Syntaxe

```
Resultat\$ =
    PreferenceKeyName()
```

Description

Renvoie le nom de la clé courante.

Arguments

Aucun.

Valeur de retour

Renvoie le nom de la clé courante dans l'énumération commencée avec ExaminePreferenceKeys() .

Remarques

La commande PreferenceKeyValue() permet de récupérer la valeur de la clé.

Exemple

```
; Ouvrir un fichier préférence
OpenPreferences(#PB_Compiler_Home +"Examples\Sources\Data\test.pref")

; Groupe à examiner
PreferenceGroup("Window")

; Recherche des clés dans le groupe 'Window'
ExaminePreferenceKeys()
While  NextPreferenceKey() ; Tant qu'il existe une clé à afficher
  MessageRequester("Clés du groupe 'Window'", PreferenceKeyName() + " = " + PreferenceKeyValue()) ; Afficher la clé et sa valeur
Wend

; Fermer le fichier préférence
ClosePreferences()
```

Voir aussi

ExaminePreferenceKeys() , PreferenceKeyValue()

OS Supportés

Tous

128.10 PreferenceKeyValue

Syntaxe

```
Resultat\$ =
    PreferenceKeyValue()
```

Description

Renvoie la valeur de la clé courante.

Arguments

Aucun.

Valeur de retour

Renvoie la valeur de la clé courante dans l'énumération commencée avec ExaminePreferenceKeys() .

Remarques

La commande PreferenceKeyName() permet de récupérer le nom de la clé.

Exemple

```
; Ouvrir un fichier préférence
OpenPreferences(#PB_Compiler_Home +"Examples\Sources\Data\test.pref")

; Groupe à examiner
PreferenceGroup("Window")

; Recherche des clés dans le groupe 'Window'
ExaminePreferenceKeys()
While  NextPreferenceKey()
  ; Tant qu'il existe une clé à afficher
   MessageRequester("Clés du groupe 'Window'", PreferenceKeyName() + " = " + PreferenceKeyValue())
  ; Afficher la clé et sa valeur
Wend

; Fermer le fichier préférence
ClosePreferences()
```

Voir aussi

ExaminePreferenceKeys() , PreferenceKeyName()

OS Supportés

Tous

128.11 OpenPreferences

Syntaxe

```
Resultat =
    OpenPreferences(Fichier$
    [, Options])
```

Description

Ouvre un fichier de préférences.

Arguments

Fichier$ Le nom du fichier de préférences à ouvrir.

Options (optionnel) Peut être une combinaison des valeurs suivantes :

```
#PB_Preference_NoSpace
        : Pas d'espaces
  autour du signe égal.
                                    Utile
  quand il s'agit d'un
  fichier de préférences
  externes qui
                                    n'accepte
  pas les espaces autour
  de signe égal.
#PB_Preference_GroupSeparator:
  Ajoute une ligne vide
  entre les groupes pour
  faciliter la lisibilité.
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Si le fichier ne peut pas être ouvert, le programmeur peut malgré tout utiliser la commande de lecture, celle-ci renvoyant alors la valeur spécifiée par défaut. Ceci est très pratique pour initialiser en une fois les variables du programme. Les fonctions comme ReadPreferenceString() peuvent être utilisées pour lire les valeurs de préférences stockées dans le fichier.
Pour effacer une clé ou un groupe, utiliser RemovePreferenceKey() ou RemovePreferenceGroup() .
PreferenceGroup() permet de créer ou de changer le groupe courant.
Il est possible de modifier les valeurs existantes avec WritePreferenceString() ou une commande ad hoc.

Une fois les opérations sur le fichier terminées, il faut utiliser ClosePreferences() pour mettre à jour, écrire et fermer le fichier préférences sur le disque.

Exemple

```
; Ouvrir un fichier preférence
OpenPreferences(#PB_Compiler_Home +"Examples\Sources\Data\test.pref")

; Examen des Groupes
ExaminePreferenceGroups()
; Pour chaque groupe
While NextPreferenceGroup()
  texte$ = texte$ + PreferenceGroupName() + #LF$ ; On récupère son nom
; Examen des Clés pour le groupe en cours
 ExaminePreferenceKeys()
; Pour chaque clé
While  NextPreferenceKey()
  texte$ = texte$ + PreferenceKeyName() + " = " + PreferenceKeyValue() + #LF$ ; On récupère son nom et sa valeur
Wend
texte$ = texte$ +  #LF$
Wend

; Afficher tous les groupes et toutes les clés avec leur valeur
MessageRequester("test.pref",texte$)

; Fermer le fichier preférence
ClosePreferences()
```

Voir aussi

ClosePreferences()

OS Supportés

Tous

128.12 PreferenceGroup

Syntaxe

```
Resultat =
    PreferenceGroup(Nom$)
```

Description

Crée un nouveau groupe sous la forme : [Nom$] ou modifie le groupe courant dans le fichier de préférences.

Arguments

Nom$ Le nouveau nom du groupe.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Toutes les opérations de lecture ou d'écriture qui suivent ne concernent que ce groupe.
Si le groupe n'existe pas, il ne sera pas immédiatement créé. Il sera créé uniquement lorsqu'au moins une clé sera écrite dans ce groupe. Ceci permet d'utiliser PreferenceGroup() pour tester si un groupe existe sans créer plein de groupes vides.
Pour quitter un groupe, un "Nom$" vide peut être utilisé.

Exemple

```
1   ; Ouvrir un fichier
     préférence
2   OpenPreferences(#PB_Compiler_Home
     +"Examples\Sources\Data\test.pref")
3
4   ; Groupe à examiner
5   PreferenceGroup("Window")
6
7   ; Recherche des clés dans
     le groupe 'Window'
8   ExaminePreferenceKeys()
9   While  NextPreferenceKey()
     ; Tant qu'il existe une
     clé à afficher
10   MessageRequester("Clés du
     groupe 'Window'",
     PreferenceKeyName() + " =
     " + PreferenceKeyValue())
     ; Afficher la clé et sa
     valeur
11  Wend
12
13  ; Fermer le fichier
     préférence
14  ClosePreferences()
```

Voir aussi

PreferenceGroupName() ()

OS Supportés

Tous

128.13 PreferenceComment

Syntaxe

```
PreferenceComment(Texte$)
```

Description

Ecrit une ligne de commentaire dans le fichier préférences en cours de création.

Arguments

Texte$ Le nouveau commentaire à écrire.

Valeur de retour

Aucune.

Exemple

```
; Création du fichier prefs.txt
If CreatePreferences(GetTemporaryDirectory()+"Prefs.txt")

 PreferenceComment("Les coordonnées de la fenêtre") ; écriture d'un commentaire
 PreferenceComment("") ; écriture d'un commentaire (ligne vide)

 PreferenceGroup("Window")
   WritePreferenceLong("X", 100)
   WritePreferenceLong("Y", 125)

   ClosePreferences()

   RunProgram(GetTemporaryDirectory()+"Prefs.txt")
EndIf
```

OS Supportés

Tous

128.14 ReadPreferenceDouble

Syntaxe

```
Resultat.d =
    ReadPreferenceDouble(Clé$,
    ValeurParDefaut)
```

Description

Lit et renvoie le nombre Double associé à une clé.

Arguments

Clé$ Le nom de la clé à lire.
Si la commande PreferenceGroup() a été utilisée alors la recherche est limitée au groupe courant.

ValeurParDefaut la valeur par défaut à renvoyer si la clé n'existe pas ou si le fichier de préférences n'a pas été ouvert correctement (fichier absent par exemple).

Valeur de retour

Renvoie le nombre à virgule en double précision Double associée à la clé. Si ce nombre n'existe pas alors la valeur par défaut 'ValeurParDefaut' est renvoyée.

Exemple

```
1  ; Ouvrir un fichier
     préférence
2  OpenPreferences(#PB_Compiler_Home
     +"Examples\Sources\Data\test.pref")
3
4  ; Ouvrir le Groupe 'Window'
5  PreferenceGroup("Window")
6
7  ; Parcourir les clés
     jusqu'à la clé: w = 800
8  NextPreferenceKey() ; x = 0
9  NextPreferenceKey() ; y = 0
10 NextPreferenceKey() ; w =
     800
11
12 ; Lecture de la clé
13 cle$= PreferenceKeyName()
14
15 ; Affichage de la valeur de
     la clé sous différents
     formats
16   MessageRequester("Clé
     Integer",
     Str(ReadPreferenceInteger(cle$,0)))
```

```
17      MessageRequester("Clé
        Float",
        StrF(ReadPreferenceFloat(cle$,0),6))
18      MessageRequester("Clé
        Double",
        StrD(ReadPreferenceDouble(cle$,0),15))
19      MessageRequester("Clé
        Long",
        Str(ReadPreferenceLong(cle$,0)))
20      MessageRequester("Clé
        Quad",
        Str(ReadPreferenceQuad(cle$,0)))
21      MessageRequester("Clé
        String",
        ReadPreferenceString(cle$,"0"))
22
23    ; Fermer le fichier
        préférence
24    ClosePreferences()
```

Voir aussi

ReadPreferenceFloat() ,
ReadPreferenceInteger() ,
ReadPreferenceLong() ,
ReadPreferenceQuad() ,
ReadPreferenceString()

OS Supportés

Tous

128.15 ReadPreferenceFloat

Syntaxe

```
Resultat.f =
    ReadPreferenceFloat(Clé$,
    ValeurParDefaut)
```

Description

Lit et renvoie le nombre Float associé à une clé.

Arguments

Clé$ Le nom de la clé à lire.
Si la commande PreferenceGroup() a été utilisée alors la recherche est limitée au groupe courant.

ValeurParDefaut la valeur par défaut à renvoyer si la clé n'existe pas ou si le fichier de préférences n'a pas été ouvert correctement (fichier absent par exemple).

Valeur de retour

Renvoie le nombre à virgule en simple précision Float associée à la clé. Si ce nombre n'existe pas alors la valeur par défaut 'ValeurParDefaut' est renvoyée.

Exemple

```
; Ouvrir un fichier préférence
OpenPreferences(#PB_Compiler_Home +"Examples\Sources\Data\test.pref")

; Ouvrir le Groupe 'window'
PreferenceGroup("window")

; Parcourir les clés jusqu'à la clé: w = 800
NextPreferenceKey() ; x = 0
NextPreferenceKey() ; y = 0
NextPreferenceKey() ; w = 800

; Lecture de la clé
cle$= PreferenceKeyName()

; Affichage de la valeur de la clé sous différents formats
 MessageRequester("Clé Integer", Str(ReadPreferenceInteger(cle$,0)))
 MessageRequester("Clé Float", StrF(ReadPreferenceFloat(cle$,0),6))
 MessageRequester("Clé Double", StrD(ReadPreferenceDouble(cle$,0),15))
 MessageRequester("Clé Long", Str(ReadPreferenceLong(cle$,0)))
 MessageRequester("Clé Quad", Str(ReadPreferenceQuad(cle$,0)))
 MessageRequester("Clé String", ReadPreferenceString(cle$,"0"))

; Fermer le fichier préférence
ClosePreferences()
```

Voir aussi

ReadPreferenceDouble() ,
ReadPreferenceInteger() ,
ReadPreferenceLong() ,

ReadPreferenceQuad() ,
ReadPreferenceString()

OS Supportés

Tous

128.16 ReadPreferenceInteger

Syntaxe

```
Resultat.i =
    ReadPreferenceInteger(Clé$,
    ValeurParDefaut)
```

Description

Lit et renvoie le nombre entier Integer associé à une clé.

Arguments

Clé$ Le nom de la clé à lire.
Si la commande PreferenceGroup() a été utilisée alors la recherche est limitée au groupe courant.

ValeurParDefaut la valeur par défaut à renvoyer si la clé n'existe pas ou si le fichier de préférences n'a pas été ouvert correctement (fichier absent par exemple).

Valeur de retour

Renvoie le nombre entier Integer associée à la clé. Si ce nombre n'existe pas alors la valeur par défaut 'ValeurParDefaut' est renvoyée.

Exemple

```
1  ; Ouvrir un fichier
    préférence
2  OpenPreferences(#PB_Compiler_Home
    +"Examples\Sources\Data\test.pref")
3
4  ; Ouvrir le Groupe 'window'
5  PreferenceGroup("window")
6
7  ; Parcourir les clés
    jusqu'à la clé: w = 800
8  NextPreferenceKey() ; x = 0
9  NextPreferenceKey() ; y = 0
10 NextPreferenceKey() ; w =
    800
11
12 ; Lecture de la clé
13 cle$= PreferenceKeyName()
```

```
14
15    ; Affichage de la valeur de
       la clé sous différents
       formats
16     MessageRequester("Clé
       Integer",
       Str(ReadPreferenceInteger(cle$,0)))
17     MessageRequester("Clé
       Float",
       StrF(ReadPreferenceFloat(cle$,0),6))
18     MessageRequester("Clé
       Double",
       StrD(ReadPreferenceDouble(cle$,0),15))
19     MessageRequester("Clé
       Long",
       Str(ReadPreferenceLong(cle$,0)))
20     MessageRequester("Clé
       Quad",
       Str(ReadPreferenceQuad(cle$,0)))
21     MessageRequester("Clé
       String",
       ReadPreferenceString(cle$,"0"))
22
23    ; Fermer le fichier
       préférence
24    ClosePreferences()
```

Voir aussi

ReadPreferenceDouble() ,
ReadPreferenceFloat() ,
ReadPreferenceLong() ,
ReadPreferenceQuad() ,
ReadPreferenceString()

OS Supportés

Tous

128.17 ReadPreferenceLong

Syntaxe

```
Resultat.l =
    ReadPreferenceLong(Clé$,
    ValeurParDefaut)
```

Description

Lit et renvoie le nombre entier Long associé à une clé.

Arguments

Clé$ Le nom de la clé à lire.
Si la commande PreferenceGroup() a été utilisée alors la recherche est limitée au groupe courant.

ValeurParDefaut la valeur par défaut à renvoyer si la clé n'existe pas ou si le fichier de préférences n'a pas été ouvert correctement (fichier absent par exemple).

Valeur de retour

Renvoie le nombre entier Long associée à la clé. Si ce nombre n'existe pas alors la valeur par défaut 'ValeurParDefaut' est renvoyée.

Exemple

```
; Ouvrir un fichier préférence
OpenPreferences(#PB_Compiler_Home +"Examples\Sources\Data\test.pref")

; Ouvrir le Groupe 'window'
PreferenceGroup("window")

; Parcourir les clés jusqu'à la clé: w = 800
NextPreferenceKey() ; x = 0
NextPreferenceKey() ; y = 0
NextPreferenceKey() ; w = 800

; Lecture de la clé
cle$= PreferenceKeyName()

; Affichage de la valeur de la clé sous différents formats
 MessageRequester("Clé Integer", Str(ReadPreferenceInteger(cle$,0)))
 MessageRequester("Clé Float", StrF(ReadPreferenceFloat(cle$,0),6))
 MessageRequester("Clé Double", StrD(ReadPreferenceDouble(cle$,0),15))
 MessageRequester("Clé Long", Str(ReadPreferenceLong(cle$,0)))
 MessageRequester("Clé Quad", Str(ReadPreferenceQuad(cle$,0)))
 MessageRequester("Clé String", ReadPreferenceString(cle$,"0"))

; Fermer le fichier préférence
ClosePreferences()
```

Voir aussi

ReadPreferenceDouble() ,
ReadPreferenceFloat() ,
ReadPreferenceInteger() ,
ReadPreferenceQuad() ,
ReadPreferenceString()

OS Supportés

Tous

128.18 ReadPreferenceQuad

Syntaxe

```
Resultat.q =
    ReadPreferenceQuad(Clé$,
    ValeurParDefaut)
```

Description

Lit et renvoie le nombre entier Quad associé à une clé.

Arguments

Clé$ Le nom de la clé à lire.
Si la commande PreferenceGroup() a été utilisée alors la recherche est limitée au groupe courant.

ValeurParDefaut la valeur par défaut à renvoyer si la clé n'existe pas ou si le fichier de préférences n'a pas été ouvert correctement (fichier absent par exemple).

Valeur de retour

Renvoie le nombre entier Quad associée à la clé. Si ce nombre n'existe pas alors la valeur par défaut 'ValeurParDefaut' est renvoyée.

Exemple

```
1  ; Ouvrir un fichier
    préférence
2  OpenPreferences(#PB_Compiler_Home
    +"Examples\Sources\Data\test.pref")
3
4  ; Ouvrir le Groupe 'window'
5  PreferenceGroup("window")
6
7  ; Parcourir les clés
    jusqu'à la clé: w = 800
8  NextPreferenceKey() ; x = 0
9  NextPreferenceKey() ; y = 0
10 NextPreferenceKey() ; w =
    800
```

```
11
12    ; Lecture de la clé
13    cle$= PreferenceKeyName()
14
15    ; Affichage de la valeur de
       la clé sous différents
       formats
16     MessageRequester("Clé
      Integer",
      Str(ReadPreferenceInteger(cle$,0)))
17     MessageRequester("Clé
      Float",
      StrF(ReadPreferenceFloat(cle$,0),6))
18     MessageRequester("Clé
      Double",
      StrD(ReadPreferenceDouble(cle$,0),15))
19     MessageRequester("Clé
      Long",
      Str(ReadPreferenceLong(cle$,0)))
20     MessageRequester("Clé
      Quad",
      Str(ReadPreferenceQuad(cle$,0)))
21     MessageRequester("Clé
      String",
      ReadPreferenceString(cle$,"0"))
22
23    ; Fermer le fichier
       préférence
24    ClosePreferences()
```

Voir aussi

ReadPreferenceDouble() ,
ReadPreferenceFloat() ,
ReadPreferenceInteger() ,
ReadPreferenceLong() ,
ReadPreferenceString()

OS Supportés

Tous

128.19 ReadPreferenceString

Syntaxe

```
Resultat\$ =
    ReadPreferenceString(Clé$,
    ValeurParDefaut)
```

Description

Lit et renvoie la chaîne de caractères String associé à une clé.

Arguments

Clé$ Le nom de la clé à lire.

Si la commande PreferenceGroup() a été utilisée alors la recherche est limitée au groupe courant.

ValeurParDefaut la valeur par défaut à renvoyer si la clé n'existe pas ou si le fichier de préférences n'a pas été ouvert correctement (fichier absent par exemple).

Valeur de retour

Renvoie la chaîne de caractères String associée à la clé. Si ce nombre n'existe pas alors la valeur par défaut 'ValeurParDefaut' est renvoyée.

Exemple

```
; Ouvrir un fichier préférence
OpenPreferences(#PB_Compiler_Home +"Examples\Sources\Data\test.pref")

; Ouvrir le Groupe 'window'
PreferenceGroup("window")

; Parcourir les clés jusqu'à la clé: w = 800
NextPreferenceKey() ; x = 0
NextPreferenceKey() ; y = 0
NextPreferenceKey() ; w = 800

; Lecture de la clé
cle$= PreferenceKeyName()

; Affichage de la valeur de la clé sous différents formats
 MessageRequester("Clé Integer", Str(ReadPreferenceInteger(cle$,0)))
 MessageRequester("Clé Float", StrF(ReadPreferenceFloat(cle$,0),6))
 MessageRequester("Clé Double", StrD(ReadPreferenceDouble(cle$,0),15))
 MessageRequester("Clé Long", Str(ReadPreferenceLong(cle$,0)))
 MessageRequester("Clé Quad", Str(ReadPreferenceQuad(cle$,0)))
 MessageRequester("Clé String", ReadPreferenceString(cle$,"0"))

; Fermer le fichier préférence
```

```
24   ClosePreferences()
```

Voir aussi

ReadPreferenceDouble() ,
ReadPreferenceFloat() ,
ReadPreferenceInteger() ,
ReadPreferenceLong() ,
ReadPreferenceQuad()

OS Supportés

Tous

128.20 RemovePreferenceGroup

Syntaxe

```
RemovePreferenceGroup(Groupe$)
```

Description

Supprime un groupe et toutes ses clés.

Arguments

Groupe$ Le groupe à supprimer.

Valeur de retour

Aucune.

Exemple

```
1   ; Création du fichier
     prefs.txt
2   If
     CreatePreferences(GetTemporaryDirectory()+"Prefs.txt",
     #PB_Preference_GroupSeparator)
3
4     PreferenceGroup("Window")
5       WritePreferenceLong
    ("X", 100)
6       WritePreferenceLong
    ("Y", 125)
7       WritePreferenceString("Titre",
    "PureNote")
8
9       PreferenceGroup("event")
10      WritePreferenceString("Pourcentage",
    "Pourcentage")
11      WritePreferenceFloat("%",
    100)
12
13      ClosePreferences()
14
15      RunProgram(GetTemporaryDirectory()+"Prefs.txt")
```

```
16      EndIf
17
18      MessageRequester("Info",
        "Ok pour supprimer le
        groupe 'event' ")
19
20    ; Ouvrir un fichier
        préférence
21      OpenPreferences(GetTemporaryDirectory()+"Prefs.txt")
22
23    ; Supprimer le groupe
        'event'
24      RemovePreferenceGroup("event")
25
26      ClosePreferences()
27
28      RunProgram(GetTemporaryDirectory()+"Prefs.txt")
```

Voir aussi

PreferenceGroup() ,
ExaminePreferenceGroups()

OS Supportés

Tous

128.21 RemovePreferenceKey

Syntaxe

```
RemovePreferenceKey(Clé$)
```

Description

Supprime une clé et sa valeur associée.

Arguments

Clé$ La clé à supprimer.

Valeur de retour

Aucune.

Exemple

```
1   If
      CreatePreferences(GetTemporaryDirectory()+"Prefs.txt",
      #PB_Preference_GroupSeparator)
2      PreferenceGroup("Window")
3        WritePreferenceLong
      ("X", 100)
4        WritePreferenceLong
      ("Y", 125)
5        WritePreferenceString("Titre",
      "PureNote")
```

```
6       ClosePreferences()
7       RunProgram(GetTemporaryDirectory()+"Prefs.txt")
8       EndIf
9
10      MessageRequester("Info",
        "Ok pour supprimer la clé
        'Titre' ")
11
12    ; Ouvrir le fichier
        préférence
13      OpenPreferences(GetTemporaryDirectory()+"Prefs.txt")
14      PreferenceGroup("Window")
15
16    ; Supprimer la clé 'Titre'
17      RemovePreferenceKey("Titre")
18
19      ClosePreferences()
20      RunProgram(GetTemporaryDirectory()+"Prefs.txt")
```

Voir aussi

ExaminePreferenceKeys()

OS Supportés

Tous

128.22 WritePreferenceFloat

Syntaxe

```
WritePreferenceFloat(Clé$,
    Valeur.f)
```

Description

Crée ou change la paire Clé-Valeur sous la forme : 'Clé$ = Valeur'.
La valeur est de type nombre à virgule en simple précision Float

Arguments

Clé$ Le nom de la clé à écrire.
Si la clé existe, la valeur associée est remplacée par la nouvelle valeur.
Si PreferenceGroup() a été utilisé alors l'écriture est limitée au groupe courant.

Valeur Le nombre à virgule en simple précision à associer à la clé. Voir Float

Valeur de retour

Aucune.

Exemple

```
; Création du fichier prefs.txt
If CreatePreferences(GetTemporaryDirectory()+"Prefs.txt", #PB_Preference_GroupSeparator)
  PreferenceGroup("Window")
    WritePreferenceString("Titre", "PureNote")
    WritePreferenceLong("X", 100)
    WritePreferenceLong("Y", 125)
    WritePreferenceInteger("I", 1024)
    WritePreferenceQuad("Q", 9223372036854775807)
    WritePreferenceFloat("%", 20.10)
    WritePreferenceDouble("D", 0.0123456789 )
  ClosePreferences()
  RunProgram(GetTemporaryDirectory()+"Prefs.txt")
  EndIf
```

Voir aussi

WritePreferenceDouble() ,
WritePreferenceInteger() ,
WritePreferenceLong() ,
WritePreferenceQuad() ,
WritePreferenceString()

OS Supportés

Tous

128.23 WritePreferenceDouble

Syntaxe

```
WritePreferenceDouble(Clé$,
    Valeur.d)
```

Description

Crée ou change la paire Clé-Valeur sous la forme : 'Clé$ = Valeur'.
La valeur est de type nombre à virgule en double précision Double

Arguments

Clé$ Le nom de la clé à écrire.
Si la clé existe, la valeur associée est remplacée par la nouvelle valeur.

Si PreferenceGroup() a été utilisé alors l'écriture est limitée au groupe courant.

Valeur Le nombre à virgule en double précision à associer à la clé. Voir Double

Valeur de retour

Aucune.

Exemple

```
; Création du fichier prefs.txt
If CreatePreferences(GetTemporaryDirectory()+"Prefs.txt", #PB_Preference_GroupSeparator)
  PreferenceGroup("Window")
    WritePreferenceString("Titre", "PureNote")
    WritePreferenceLong ("X", 100)
    WritePreferenceLong ("Y", 125)
    WritePreferenceInteger("I", 1024)
    WritePreferenceQuad("Q", 9223372036854775807)
    WritePreferenceFloat("%", 20.10)
    WritePreferenceDouble("D", 0.0123456789 )
  ClosePreferences()
  RunProgram(GetTemporaryDirectory()+"Prefs.txt")
  EndIf
```

Voir aussi

WritePreferenceFloat() ,
WritePreferenceInteger() ,
WritePreferenceLong() ,
WritePreferenceQuad() ,
WritePreferenceString()

OS Supportés

Tous

128.24 WritePreferenceInteger

Syntaxe

```
WritePreferenceInteger(Clé$, Valeur)
```

Description

Crée ou change la paire Clé-Valeur sous la forme : 'Clé$ = Valeur'.

La valeur est de type entier Integer

Arguments

Clé$ Le nom de la clé à écrire.
Si la clé existe, la valeur associée est remplacée par la nouvelle valeur.
Si PreferenceGroup() a été utilisé alors l'écriture est limitée au groupe courant.

Valeur Le nombre entier à associer à la clé. Voir Integer

Valeur de retour

Aucune.

Exemple

```
1  ; Création du fichier prefs.txt
2  If CreatePreferences(GetTemporaryDirectory()+"Prefs.txt", #PB_Preference_GroupSeparator)
3    PreferenceGroup("Window")
4      WritePreferenceString("Titre", "PureNote")
5      WritePreferenceLong("X", 100)
6      WritePreferenceLong("Y", 125)
7      WritePreferenceInteger("I", 1024)
8      WritePreferenceQuad("Q", 9223372036854775807)
9      WritePreferenceFloat("%", 20.10)
10     WritePreferenceDouble("D", 0.0123456789 )
11   ClosePreferences()
12   RunProgram(GetTemporaryDirectory()+"Prefs.txt")
13   EndIf
```

Voir aussi

WritePreferenceFloat() ,
WritePreferenceDouble() ,
WritePreferenceLong() ,
WritePreferenceQuad() ,
WritePreferenceString()

OS Supportés

Tous

128.25 WritePreferenceLong

Syntaxe

```
WritePreferenceLong(Clé$,
    Valeur)
```

Description

Crée ou change la paire Clé-Valeur sous la forme : 'Clé$ = Valeur'.
La valeur est de type entier long Long

Arguments

Clé$ Le nom de la clé à écrire.
Si la clé existe, la valeur associée est remplacée par la nouvelle valeur.
Si PreferenceGroup() a été utilisé alors l'écriture est limitée au groupe courant.

Valeur Le nombre entier long à associer à la clé. Voir Long

Valeur de retour

Aucune.

Exemple

```
1   ; Création du fichier
     prefs.txt
2   If
     CreatePreferences(GetTemporaryDirectory()+"Prefs.txt",
     #PB_Preference_GroupSeparator)
3     PreferenceGroup("Window")
4       WritePreferenceString("Titre",
    "PureNote")
5       WritePreferenceLong
    ("X", 100)
6       WritePreferenceLong
    ("Y", 125)
7       WritePreferenceInteger("I",
    1024)
8       WritePreferenceQuad("Q",
    9223372036854775807)
9       WritePreferenceFloat("%",
    20.10)
10      WritePreferenceDouble("D",
    0.0123456789 )
11    ClosePreferences()
12    RunProgram(GetTemporaryDirectory()+"Prefs.txt")
13    EndIf
```

Voir aussi

WritePreferenceFloat() ,
WritePreferenceDouble() ,
WritePreferenceInteger() ,

WritePreferenceQuad() ,
WritePreferenceString()

OS Supportés

Tous

128.26 WritePreferenceQuad

Syntaxe

```
WritePreferenceQuad(Clé$,
    Valeur.q)
```

Description

Crée ou change la paire Clé-Valeur sous la forme : 'Clé$ = Valeur'.
La valeur est de type entier quad Quad

Arguments

Clé$ Le nom de la clé à écrire.
Si la clé existe, la valeur associée est remplacée par la nouvelle valeur.
Si PreferenceGroup() a été utilisé alors l'écriture est limitée au groupe courant.

Valeur Le nombre entier quad à associer à la clé. Voir Quad

Valeur de retour

Aucune.

Exemple

```
1   ; Création du fichier
     prefs.txt
2   If
     CreatePreferences(GetTemporaryDirectory()+"Prefs.txt",
     #PB_Preference_GroupSeparator)
3     PreferenceGroup("Window")
4       WritePreferenceString("Titre",
     "PureNote")
5       WritePreferenceLong
     ("X", 100)
6       WritePreferenceLong
     ("Y", 125)
7       WritePreferenceInteger("I",
     1024)
8       WritePreferenceQuad("Q",
     9223372036854775807)
9       WritePreferenceFloat("%",
     20.10)
10      WritePreferenceDouble("D",
     0.0123456789 )
11    ClosePreferences()
```

```
12      RunProgram(GetTemporaryDirectory()+"Prefs.txt")
13      EndIf
```

Voir aussi

WritePreferenceFloat() ,
WritePreferenceDouble() ,
WritePreferenceInteger() ,
WritePreferenceLong() ,
WritePreferenceString()

OS Supportés

Tous

128.27 WritePreferenceString

Syntaxe

```
WritePreferenceString(Clé$,
    Valeur$)
```

Description

Crée ou change la paire Clé-Valeur sous la forme : 'Clé$ = Valeur'.
La valeur est de type chaîne de caractères String

Arguments

Clé$ Le nom de la clé à écrire.
Si la clé existe, la valeur associée est remplacée par la nouvelle valeur.
Si PreferenceGroup() a été utilisé alors l'écriture est limitée au groupe courant.

Valeur La valeur est de type chaîne de caractères à associer à la clé. Voir String

Valeur de retour

Aucune.

Exemple

```
1   ; Création du fichier
     prefs.txt
2   If
     CreatePreferences(GetTemporaryDirectory()+"Prefs.txt",
     #PB_Preference_GroupSeparator)
3     PreferenceGroup("Window")
4       WritePreferenceString("Titre",
     "PureNote")
5       WritePreferenceLong
     ("X", 100)
6       WritePreferenceLong
     ("Y", 125)
```

```
7       WritePreferenceInteger("I",
    1024)
8       WritePreferenceQuad("Q",
    9223372036854775807)
9       WritePreferenceFloat("%",
    20.10)
10      WritePreferenceDouble("D",
    0.0123456789 )
11      ClosePreferences()
12      RunProgram(GetTemporaryDirectory()+"Prefs.txt")
13      EndIf
```

Voir aussi

WritePreferenceFloat() ,
WritePreferenceDouble() ,
WritePreferenceInteger() ,
WritePreferenceQuad() ,
WritePreferenceQuad()

OS Supportés

Tous

Chapitre 129

Printer

Généralités

Les imprimantes sont des périphériques essentiels pour transformer les informations numériques virtuelles en informations écrites. La plupart des logiciels nécessitent l'impression des informations sur papier pour être pleinement efficaces. PureBasic permet d'imprimer tous types de données depuis le texte de base jusqu'aux images dans toutes les résolutions.

OS Supportés

Tous

129.1 DefaultPrinter

Syntaxe

```
Resultat = DefaultPrinter()
```

Description

Sélectionne l'imprimante par défaut.

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon (pas d'imprimante par défaut disponible).

Remarques

Cette fonction doit être utilisée avant toute autre de la bibliothèque Printer.
Une fois la commande DefaultPrinter() terminée avec succès la commande StartPrinting() est utilisée pour démarrer l'impression.

Exemple

```
Resultat = DefaultPrinter()
If Resultat <> 0
  MessageRequester("Info","L'imprimante par défaut est "+ Resultat)
Else
  MessageRequester("Info","Pas d'imprimante par défaut.")
EndIf
```

Voir aussi

StartPrinting() , PrinterOutput()

OS Supportés

Tous

129.2 NewPrinterPage

Syntaxe

```
NewPrinterPage()
```

Description

Créé une nouvelle page vide.

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

La page précédente est envoyée à l'imprimante et ne peut plus être modifiée. Cette commande doit être appelée à l'intérieur d'un bloc StartDrawing() /StopDrawing() .

Exemple

```
If PrintRequester()
  If StartPrinting("Deux feuilles")
    If StartDrawing(PrinterOutput())
      DrawingMode(#PB_2DDrawing_Transparent)
      DrawText(10, 10, "Première page !", RGB(0, 0, 0))

```

```
7         ; Informe
      l'imprimante qu'elle doit
      commencer une nouvelle page
8         NewPrinterPage()
9
10        DrawText(10, 10,
      "Deuxième page !", RGB(0,
      0, 0))
11
12        StopDrawing()
13      EndIf
14
15      StopPrinting()
16    EndIf
17  EndIf
```

Voir aussi

StartDrawing() , StopDrawing()

OS Supportés

Tous

129.3 PrinterOutput

Syntaxe

```
Resultat = PrinterOutput()
```

Description

Renvoie l'identifiant OutputID de l'imprimante courante, à utiliser avec la commande StartDrawing() .

Arguments

Aucun.

Valeur de retour

Renvoie le numéro d'identification de l'imprimante en cours.

Remarques

L'impression sera effectuée en utilisant les opérations de dessin à base de pixels. Dessiner sur une imprimante en utilisant le dessin à base de pixels peut réduire la qualité d'impression et de plus, ajoute de la complexité supplémentaire parce que la résolution des imprimantes doit être prise en compte. Le dessin à base de vecteurs avec la fonction PrinterVectorOutput() devrait être préféré, car il fournit des fonctions indépendantes de la résolution du matériel, et permet une impression de haute qualité.

Exemple

```
StartDrawing(PrinterOutput())
  ; faire des trucs de dessin ici ...
StopDrawing()
```

Voir aussi

StartDrawing() , PrinterVectorOutput()

OS Supportés

Tous

129.4 PrinterVectorOutput

Syntaxe

```
Resultat =
    PrinterVectorOutput([UniteDeMesure])
```

Description

Renvoie le numéro d'identificationt OutputID de l'imprimante courante afin d'effectuer des opérations de dessin vectoriel, à utiliser avec StartVectorDrawing() .

Arguments

UniteDeMesure (optionnel) Spécifie l'unité utilisée pour mesurer les distances sur le dessin.

```
#PB_Unit_Pixel      : Les valeurs sont mesurées en pixels (Par défaut)(ou point (dots) pour les imprimantes)
#PB_Unit_Point      : Les valeurs sont mesurées en points (1/72 pouce = 25.4/72 mm = 0,352 778 mm)
#PB_Unit_Inch       : Les valeurs sont mesurées en pouces (25,4 millimètres)
#PB_Unit_Millimeter: Les valeurs sont mesurées en millimètres (0,039 370 pouce)
```

Valeur de retour

Le OutputID de l'imprimante en cours afin d'être utilisé avec la fonction StartVectorDrawing() .

Exemple

```
StartVectorDrawing(PrinterVectorOutput(#PB_Unit_Point))
  ; code de dessin ici ...
StopVectorDrawing()
```

Voir aussi

StartVectorDrawing() , PrinterOutput()

OS Supportés

Tous

129.5 PrintRequester

Syntaxe

```
Resultat = PrintRequester()
```

Description

Ouvre une boîte de dialogue pour sélectionner l'imprimante et ajuster les paramètres.

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon ou si l'utilisateur a interrompu la sélection en fermant la boîte de dialogue.

Remarques

Cette fonction doit être utilisée avant toute autre de la bibliothèque Printer.
Une fois la commande PrintRequester() terminée avec succès la commande StartPrinting() est utilisée pour démarrer l'impression.

Exemple

```
Resultat = PrintRequester()
If Resultat <> 0
  MessageRequester("Info",
  "Une imprimante a été
  sélectionnée : " +
  Resultat)
Else
```

```
    MessageRequester("Info",
    "Erreur ou Sélection
    annulée")
  EndIf
```

Voir aussi

StartPrinting()

OS Supportés

Tous

129.6 StartPrinting

Syntaxe

```
Resultat =
    StartPrinting(Impression$)
```

Description

Initialise l'imprimante et démarre l'impression.

Arguments

Impression$ Le nom qui apparaîtra dans la file d'attente et qui identifie la tâche d'impression.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Exemple

```
  If PrintRequester()

  If StartPrinting("Test")

      If
   StartDrawing(PrinterOutput())
        DrawingMode(#PB_2DDrawing_Transparent)
        DrawText(10, 10,
   "Première page !", RGB(0,
   0, 0))
        DrawingMode(#PB_2DDrawing_Outlined)
        Circle(500,500,500 ,
   RGB(255,0,0))
        StopDrawing()
      EndIf

      StopPrinting()
    EndIf

  EndIf
```

Voir aussi

StopPrinting()

OS Supportés

Tous

129.7 StopPrinting

Syntaxe

```
StopPrinting()
```

Description

Arrête toutes les opérations d'impression et envoie les données à l'imprimante.

Arguments

Aucun.

Valeur de retour

Aucune.

Exemple

```
If PrintRequester()

If StartPrinting("Test")

    If StartDrawing(PrinterOutput())
      DrawingMode(#PB_2DDrawing_Transparent)
      DrawText(10, 10, "Première page !", RGB(0, 0, 0))
      DrawingMode(#PB_2DDrawing_Outlined)
      Circle(500,500,500 , RGB(255,0,0))
      StopDrawing()
    EndIf

    StopPrinting()
  EndIf

EndIf
```

Voir aussi

StartPrinting()

OS Supportés

Tous

129.8 PrinterPageWidth

Syntaxe

```
Resultat = PrinterPageWidth()
```

Description

Renvoie la largeur de la zone de tracé.

Arguments

Aucun.

Valeur de retour

Renvoie la largeur de la zone d'impression en pixels.

Remarques

Ce nombre change en fonction de la valeur DPI de l'imprimante.
Cela signifie qu'un document imprimé en 75 DPI aura une largeur de zone d'impression 4 fois plus petite qu'un document imprimé en 150 DPI.
La valeur renvoyée représente la zone client, c'est la largeur disponible pour imprimer, sans compter les marges 'hardware' (qu'on ne peux pas changer) propres à chaque imprimante.

Exemple

```
1  If PrintRequester()
2    MessageRequester("Info",
     "Hauteur marge matérielle
     : " + PrinterPageHeight()
     + " Pixels")
3    MessageRequester("Info",
     "Largeur marge matérielle
     : " + PrinterPageWidth() +
     " Pixels")
4  Else
5    MessageRequester("Info",
     "Pas d'imprimante
     sélectionnée.")
6  EndIf
```

Voir aussi

PrinterPageHeight()

OS Supportés

Tous

129.9 PrinterPageHeight

Syntaxe

```
Resultat = PrinterPageHeight()
```

Description

Renvoie la hauteur de la zone de tracé.

Arguments

Aucun.

Valeur de retour

Renvoie la hauteur de la zone d'impression en pixels.

Remarques

Ce nombre change en fonction de la valeur DPI de l'imprimante.
Cela signifie qu'un document imprimé en 75 DPI aura une hauteur de zone d'impression 4 fois plus petite qu'un document imprimé en 150 DPI.
La valeur renvoyée représente la zone client, c'est la largeur disponible pour imprimer, sans compter les marges 'hardware' (qu'on ne peux pas changer) propres à chaque imprimante.

Exemple

```
1  If PrintRequester()
2    MessageRequester("Info",
     "Hauteur marge matérielle
     : " + PrinterPageHeight()
     + " Pixels")
3    MessageRequester("Info",
     "Largeur marge matérielle
     : " + PrinterPageWidth() +
     " Pixels")
4  Else
5    MessageRequester("Info",
     "Pas d'imprimante
     sélectionnée.")
6  EndIf
```

Voir aussi

PrinterPageWidth()

OS Supportés

Tous

Chapitre 130

Process

Généralités

Cette bibliothèque permet de récupérer des informations sur le programme en cours, ainsi que d'exécuter d'autres programmes et de communiquer avec eux. Elle fournit un accès indépendant à l'environnement et les paramètres des programmes, ainsi que l'entrée et la sortie standard.

OS Supportés

Tous

130.1 AvailableProgramOutput

Syntaxe

```
Resultat =
    AvailableProgramOutput(Programme)
```

Description

Renvoie le nombre d'octets disponibles en lecture sur la sortie du programme.

Arguments

Programme Le programme à utiliser.
Le programme doit avoir été lancé avec RunProgram() en spécifiant l'option #PB_Program_Read.

Valeur de retour

Le nombre d'octets disponibles en lecture sur la sortie du programme.

Remarques

La sortie peut être effectivement lue soit avec ReadProgramString() soit avec ReadProgramData() .

Cependant ReadProgramString() ou ReadProgramData() restent bloquée indéfiniment si le programme réclame l'intervention de l'utilisateur. Comme par exemple, renvoyer un choix oui/non au programme pour qu'il se poursuive.

Voir aussi

ReadProgramString() , ReadProgramData()

OS Supportés

Tous

130.2 CloseProgram

Syntaxe

```
CloseProgram(Programme)
```

Description

Ferme la connexion avec le programme lancé avec RunProgram() et libère toutes les ressources associées.

Arguments

Programme Le programme à utiliser. Le programme doit avoir été lancé avec RunProgram() .

Valeur de retour

Aucune.

Remarques

Cette commande ne force pas le programme à se terminer, seule la connexion avec lui est fermée. Pour forcer le programme à quitter, la commande KillProgram() est disponible. De plus, même si le programme se termine de façon normale, il est important d'appeler CloseProgram() pour libérer toutes les ressources.
Si le programme a été lancé avec l'option #PB_Program_Write, CloseProgram() enverra un signal EOF (fin de fichier) sur l'entrée standard du 'Programme'. Ce signal peut aussi être artificiellement provoqué sans fermer la connexion avec le programme en appelant WriteProgramData() avec la valeur spéciale #PB_Program_Eof.

Voir aussi

KillProgram() , RunProgram()

OS Supportés

Tous

130.3 CountProgramParameters

Syntaxe

```
Resultat =
    CountProgramParameters()
```

Description

Renvoie le nombre de paramètres qui ont été passés au programme.

Arguments

Aucun.

Valeur de retour

Renvoie le nombre de paramètres qui ont été passés au programme via la ligne de commande ou avec RunProgram() .

Remarques

ProgramParameter() peut être utilisé pour lire chaque paramètre.

Voir aussi

ProgramParameter()

OS Supportés

Tous

130.4 EnvironmentVariableName

Syntaxe

```
Resultat\$ =
    EnvironmentVariableName()
```

Description

Renvoie le nom de la variable d'environnement en cours d'énumération.

Arguments

Aucun.

Valeur de retour

Renvoie le nom de la variable d'environnement en cours d'énumération avec ExamineEnvironmentVariables() et NextEnvironmentVariable() .

Remarques

Pour récupérer la valeur de cette variable d'environnement, utiliser EnvironmentVariableValue() .

Voir aussi

ExamineEnvironmentVariables() , NextEnvironmentVariable() , EnvironmentVariableValue()

OS Supportés

Tous

130.5 EnvironmentVariableValue

Syntaxe

```
Resultat\$ =
    EnvironmentVariableValue()
```

Description

Renvoie la valeur de la variable d'environnement en cours d'énumération.

Arguments

Aucun.

Valeur de retour

Renvoie la valeur de la variable d'environnement en cours d'énumération avec ExamineEnvironmentVariables() et NextEnvironmentVariable() .

Remarques

Pour récupérer le nom de cette variable d'environnement, utiliser EnvironmentVariableName() .

Voir aussi

ExamineEnvironmentVariables() , NextEnvironmentVariable() , EnvironmentVariableName()

OS Supportés

Tous

130.6 ExamineEnvironmentVariables

Syntaxe

```
Resultat =
    ExamineEnvironmentVariables()
```

Description

Initialise l'énumération des variables d'environnement disponibles.

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Les commandes NextEnvironmentVariable() , EnvironmentVariableName() et EnvironmentVariableValue() sont disponibles pour récupérer les informations concernant chaque variable d'environnement.

Exemple

```
1   ; énumère toutes les
     variables d'environnement
     disponibles pour le
     programme
2   ;
3   OpenConsole()
4   If
     ExamineEnvironmentVariables()
5     While
     NextEnvironmentVariable()
6       PrintN(EnvironmentVariableName()
     + " = " +
     EnvironmentVariableValue())
7     Wend
8   EndIf
9   PrintN("")
10  PrintN("Appuyez sur
     [Entree] pour quitter.")
11  Input()
```

Voir aussi

NextEnvironmentVariable() ,
EnvironmentVariableName() ,
EnvironmentVariableValue()

OS Supportés

Tous

130.7 GetEnvironmentVariable

Syntaxe

```
Resultat\$ =
    GetEnvironmentVariable(Variable$)
```

Description

Renvoie le contenu (texte) d'une Variable d'environnement.

Arguments

Nom$ Le nom de la variable d'environnement.

Valeur de retour

Renvoie le contenu de la 'Variable$' d'environnement spécifiée. Si la variable d'environnement n'existe pas, une chaîne de caractères vide sera renvoyée.

Exemple

```
1  ; Affiche le contenu de la
     variable d'environnement
     "PATH"
2  ;
3  OpenConsole()
4  PrintN(GetEnvironmentVariable("PATH"))
5  PrintN("")
6  PrintN("Appuyez sur
     [Entree] pour quitter.")
7  Input()
```

Voir aussi

SetEnvironmentVariable()

OS Supportés

Tous

130.8 IsProgram

Syntaxe

```
Resultat =
    IsProgram(Programme)
```

Description

Teste si un programme préalablement lancé avec RunProgram() est correctement initialisé.

Arguments

Programme Le programme à tester.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage.

Voir aussi

RunProgram()

OS Supportés

Tous

130.9 KillProgram

Syntaxe

```
KillProgram(Programme)
```

Description

Termine immédiatement l'exécution du programme spécifié préalablement lancé avec RunProgram() .

Arguments

Programme Le programme à terminer.

Valeur de retour

Aucune.

Remarques

Cette commande termine l'exécution du programme, mais elle ne ferme pas la connexion avec le programme. CloseProgram() doit toujours être appelé pour libérer les ressources associées avec le programme.

Voir aussi

CloseProgram()

OS Supportés

Tous

130.10 NextEnvironmentVariable

Syntaxe

```
Resultat =
    NextEnvironmentVariable()
```

Description

Passe à la variable d'environnement suivante dans l'énumération commencée avec ExamineEnvironmentVariables() .

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle s'il existe encore des variables à lire, zéro sinon.

Remarques

Les commandes EnvironmentVariableName() et EnvironmentVariableValue() sont disponibles pour récupérer le nom et la valeur de la variable d'environnement courante.

Voir aussi

ExamineEnvironmentVariables() , EnvironmentVariableName() , EnvironmentVariableValue()

OS Supportés

Tous

130.11 ProgramExitCode

Syntaxe

```
Resultat =
    ProgramExitCode(Programme)
```

Description

Renvoie le code de retour d'un programme quand il s'est terminé.

Arguments

Programme Le programme à utiliser. Le programme doit avoir été lancé avec RunProgram() .

Valeur de retour

Le code de retour (exitcode) renvoyé quand le programme spécifié se ferme.

Remarques

Cette commande doit seulement être utilisée après que l'exécution du programme se soit réellement terminée. Les commandes ProgramRunning() ou WaitProgram() permettent de faire cette vérification.
Le code de retour permet de renvoyer une valeur numérique au programme qui l'a exécuté. C'est souvent utilisé pour signaler si une erreur est survenue dans le programme lancé. De plus, sur OSX et Linux le code de retour est compris entre 0 et 255.
Pour renvoyer un code de retour à partir d'un programme PureBasic, il faut utiliser le paramètre optionnel du mot-clef End :

```
1  End 1   ; renvoie le code
     de retour 1
```

OS Supportés

Tous

130.12 ProgramFilename

Syntaxe

```
Resultat$ = ProgramFilename()
```

Description

Renvoie le chemin complet et le nom du programme.

Arguments

Aucun.

Valeur de retour

Renvoie le chemin complet et le nom du programme.

Remarques

Utile pour savoir où le programme a été installé ou savoir le nom de l'exécutable. GetPathPart() et GetFilePart() permettent d'isoler respectivement le chemin et le nom du fichier à partir de 'Resultat$'.
Si cette commande est utilisée dans une DLL, cette commande renvoie le chemin et le nom de fichier de la DLL, pas celui du programme qui a chargé la DLL.

OS Supportés

Tous

130.13 ProgramID

Syntaxe

```
Resultat =
    ProgramID(Programme)
```

Description

Renvoie l'identifiant système unique d'un programme.

Arguments

Programme Le programme à utiliser. Le programme doit avoir été lancé avec RunProgram() .

Valeur de retour

Renvoie l'identifiant système unique d'un programme, souvent nommé ID ou PID.
Si l'identifiant ne peut être retourné, -1 sera renvoyé.
Cela se produit si RunProgram() est utilisé pour ouvrir un fichier dans un autre programme, comme RunProgram("Source.pb").
Note : la valeur renvoyée ici n'est pas un 'handle' (comme la plupart des autres commandes xxxID()). C'est le "Process ID", qui peut être lu dans le 'Gestionnaire des tâches'. Pour obtenir un 'Process handle', utiliser l'API OpenProcess_().

Note : La valeur renvoyée peut être fausse si le process est lancé par un wrapper comme c'est le cas avec Windows 10 et les applications UWP (Universal Windows Platform) lancées par ApplicationFrameHost.exe (AFH).

OS Supportés

Tous

130.14 ProgramParameter

Syntaxe

```
Resultat\$ =
    ProgramParameter([Index])
```

Description

Renvoie le paramètre suivant qui a été passé à l'exécutable a son lancement.

Arguments

Index (optionnel) Le paramètre à cet index est renvoyé.
Le premier paramètre commence à 0.

Valeur de retour

Le paramètre suivant, ou une chaîne vide si aucun paramètre n'est trouvé. Si index est spécifié, il renverra la valeur du paramètre spécifié.

Remarques

Cette commande est particulièrement utile avec les programmes en mode console , où l'utilisateur passe un ou plusieurs paramètres au démarrage du programme.
Note : Se baser sur le renvoi d'une chaîne vide pour détecter le dernier paramètre n'est pas forcément l'idéal car la commande renverra aussi une chaîne vide si "" est passé sur la ligne de commande.
Il est recommandé d'utiliser CountProgramParameters() et d'appeler ProgramParameter() autant de fois que nécessaire.

Exemple

```
1   MonProgramme.exe
     MonTexte.txt /RAPIDE "Mode
     Special"
```

La première fois que ProgramParameter() est appelé, la commande va renvoyer "MonTexte.txt", la seconde fois "/RAPIDE" et la troisième fois "Mode Special".

OS Supportés

Tous

130.15 ProgramRunning

Syntaxe

```
Resultat =
    ProgramRunning(Programme)
```

Description

Teste si un programme est toujours en cours d'exécution.

Arguments

Programme Le programme à utiliser. Le programme doit avoir été lancé avec RunProgram() .

Valeur de retour

Renvoie une valeur non nulle tant que le programme n'est pas encore terminé, zéro sinon.
Si le programme a été exécuté avec l'option #PB_Program_Read, il renverra une valeur non nulle aussi longtemps qu'il y a quelque chose à lire, même si le programme est déjà terminé.

Voir aussi

RunProgram()

OS Supportés

Tous

130.16 ReadProgramData

Syntaxe

```
Resultat =
    ReadProgramData(Programme,
    *Memoire, Taille)
```

Description

Lit les données provenant de la sortie standard (stdout) d'un programme.

Arguments

Programme Le programme à utiliser. Le programme doit avoir été lancé avec RunProgram() en spécifiant l'option #PB_Program_Read.

***Memoire** La mémoire tampon dans laquelle seront stockées les données lues. Un tampon de mémoire peut être créé avec AllocateMemory() .

Taille La taille des données à lire, en octets. Le tampon devra être suffisamment grand pour gérer cette taille.

Valeur de retour

Le nombre d'octets effectivement lus.

Remarques

La commande lit un nombre d'octets jusqu'à la valeur 'Taille', ou moins s'il n'y a pas assez de données à lire. Par contre, cette commande attendra jusqu'à ce qu'il y ait au moins un octet avant de quitter, donc la valeur 0 pour 'Resultat' n'est pas possible.
Note : cette commande bloque jusqu'à ce qu'il y ait des données à lire. Pour éviter ce blocage, la commande AvailableProgramOutput() peut être utilisée pour vérifier si il y a bien des données en attente.
Cependant la fonction reste bloquée indéfiniment si le programme réclame l'intervention de l'utilisateur. Comme par exemple, renvoyer un choix oui/non au programme pour qu'il se poursuive.

Voir aussi

ReadProgramString()

OS Supportés

Tous

130.17 ReadProgramError

Syntaxe

```
Resultat\$ =
    ReadProgramError(Programme
    [, Options])
```

Description

Lit une ligne de texte provenant de la sortie erreur (stderr) d'un programme.

Arguments

Programme Le programme à utiliser. Il doit être lancé avant en utilisant RunProgram() avec l'option #PB_Program_Error.

Options (optionnel) Le format de chaîne de caractères à utiliser pour lire les erreurs en sortie. Le format par défaut peut être affecté avec les options #PB_Program_Ascii, #PB_Program_Unicode et #PB_Program_UTF8 de la fonction RunProgram() . Peut être l'une des valeurs suivantes :

```
#PB_Ascii   : Lecture de la
 sortie d'erreur en ascii
#PB_UTF8    : Lecture de la
 sortie d'erreur en UTF8
 (Par défaut)
#PB_Unicode: Lecture de la
 sortie d'erreur en
 unicode
```

Valeur de retour

Le texte d'erreur, ou une chaîne vide s'il n'y a pas de sortie d'erreur.

Remarques

Contrairement à ReadProgramData() , cette commande n'est pas bloquante, si aucune donnée n'est disponible (une chaîne de caractères vide sera renvoyée).

Voir aussi

ReadProgramData()

OS Supportés

Tous

130.18 ReadProgramString

Syntaxe

```
Resultat\$ =
    ReadProgramString(Programme
    [, Options])
```

Description

Lit une ligne de texte provenant de la sortie standard (stdout) d'un programme.

Arguments

Programme Le programme à utiliser. Il doit être lancé avant en utilisant RunProgram() avec l'option #PB_Program_Read.

Options (optionnel) Le format de chaîne de caractères à utiliser pour lire le texte en sortie. Le format par défaut peut être affecté avec les options #PB_Program_Ascii, #PB_Program_Unicode et #PB_Program_UTF8 de la fonction RunProgram() . Peut être l'une des valeurs suivantes :

```
#PB_Ascii   : Lecture de la
 sortie d'erreur en ascii
#PB_UTF8    : Lecture de la
 sortie d'erreur en UTF8
 (Par défaut)
#PB_Unicode: Lecture de la
 sortie d'erreur en
 unicode
```

Valeur de retour

Une chaîne créée à partir du texte de sortie du programme.

Remarques

Cette commande attend et bloque jusqu'à ce qu'il y ait des données à lire. Pour éviter ce blocage, la commande AvailableProgramOutput() peut être utilisée pour vérifier si il y a bien des données en attente. Cette commande attend également qu'une ligne complète soit reçue. Si une lecture binaire est nécessaire, la commande ReadProgramData() peut être utilisée. Cependant la fonction reste bloquée indéfiniment si le programme réclame l'intervention de l'utilisateur. Comme par exemple, renvoyer un choix oui/non au programme pour qu'il se poursuive.

Voir aussi

ReadProgramData()

OS Supportés

Tous

130.19 RemoveEnvironmentVariable

Syntaxe

```
RemoveEnvironmentVariable(Nom$)
```

Description

Supprime la variable d'environnement spécifiée du bloc d'environnement du programme.

Arguments

Nom$ La variable d'environnement à supprimer.

Valeur de retour

Aucune.

OS Supportés

Tous

130.20 RunProgram

Syntaxe

```
Resultat =
    RunProgram(Fichier$ [,
    Parametres$,
    RepertoireCourant$ [,
    Options [,
    programmeEmetteur]]])
```

Description

Lance un programme externe.

Arguments

Fichier$ Le nom de l'exécutable, y compris son chemin.

Parametres$ (optionnel) Les paramètres de ligne de commande qui seront transmis au programme.

RepertoireCourant$ (optionnel) Le répertoire qui sera alors le répertoire courant pour le programme lancé.

Options (optionnel) Peut être l'une des combinaisons suivantes (en utilisant l'opérateur OR '||') :

```
#PB_Program_Wait    :
 Attend jusqu'à ce que le
 programme lancé se
 termine.
#PB_Program_Hide    : Lance
 le programme en mode
 invisible.
```

```
#PB_Program_Open    : Ouvre
 des canaux de
 communication entre le
 programme lancé et le
 programme PureBasic.
#PB_Program_Read    :
 Lecture possible sur la
 sortie standard (stdout).
#PB_Program_Write   :
 Ecriture possible sur
 l'entrée standard
 (stdin).
#PB_Program_Error   :
 Lecture possible sur la
 sortie d'erreur(stderr).
#PB_Program_Connect:
 Connecte la sortie d'un
 autre programme à
 l'entrée du programme
 PureBasic.
#PB_Program_Ascii   : Les
 opérations de
 lecture/écriture se font
 en ASCII.
#PB_Program_Unicode: Les
 opérations de
 lecture/écriture se font
 en Unicode.
#PB_Program_UTF8    : Les
 opérations de
 lecture/écriture se font
 en UTF8. (Par défaut)
```

Un programme exécuté avec l'option #PB_Program_Open doit toujours être fermé avec la commande CloseProgram() .
Les options 'Read', 'Write', 'Error' et 'Connect' nécessitent aussi l'option #PB_Program_Open.
Quand l'option #PB_Program_Connect est utilisée, un autre programme doit avoir été préalablement lancé avec les options #PB_Program_Open et #PB_Program_Read. Le numéro renvoyé par RunProgram() doit être passé dans le paramètre 'ProgrammeEmetteur'.

ProgrammeEmetteur (optionnel)
L'option #PB_Program_Connect est nécessaire, doit contenir le numéro renvoyé d'un autre programme démarré auparavant avec RunProgram() avec les options #PB_Program_Open et #PB_Program_Read.
La sortie du programme émetteur sera envoyée directement à l'entrée du programme maintenant exécuté. Plusieurs programmes peuvent être connectés de cette manière, pour "canaliser" (pipe) les

données via ce groupe de programmes connectés.

Remarques

Les commandes suivantes peuvent être utilisées lorsque l'option #PB_Program_Open est spécifiée :
- IsProgram() : teste si le numéro représente un programme lancé avec RunProgram().
- ProgramID() : renvoie l'identifiant système du programme.
- ProgramRunning() : teste si le programme est toujours en cours d'exécution.
- WaitProgram() : attend la fin du programme.
- KillProgram() : force le programme à quitter.
- ProgramExitCode() : renvoie le code de retour du programme.
- CloseProgram() : ferme les connexions du programme et libère les ressources.

Les commandes suivantes peuvent être utilisées par les programmes lancés avec les options #PB_Program_Read, #PB_Program_Write ou #PB_Program_Error :
- AvailableProgramOutput() : teste si des données sont disponibles.
- ReadProgramString() : lit une ligne à partir de la sortie standard du programme exécuté.
- ReadProgramData() : lit des données binaires à partir de la sortie standard du programme exécuté.
- ReadProgramError() : lit une ligne à partir de la sortie erreur du programme exécuté.
- WriteProgramString() : écrit une ligne sur l'entrée standard du programme exécuté.
- WriteProgramData() : écrit des données binaires sur l'entrée standard du programme exécuté.

Valeur de retour

Renvoie une valeur non nulle si le programme a été lancé avec succès, zéro sinon.
Si l'option #PB_Program_Open est utilisée, 'Resultat' contient le numéro qui identifie le nouveau programme lancé. ReadProgramString() ou ProgramExitCode() ou d'autres fonctions déjà mentionnées peuvent alors être utilisé pour obtenir des informations sur ce nouveau programme.

Exemple

```
; Exécute le compilateur PureBasic avec l'option -h et affiche la sortie
;
Compilateur = RunProgram(#PB_Compiler_Home+"compilers/pbcompiler", "-h", "", #PB_Program_Open | #PB_Program_Read)
Sortie$ = ""
If Compilateur
  While ProgramRunning(Compilateur)
    If AvailableProgramOutput(Compilateur)
      Sortie$ + ReadProgramString(Compilateur) + Chr(13)
    EndIf
  Wend
  Sortie$ + Chr(13) + Chr(13)
  Sortie$ + "Code de retour : " + Str(ProgramExitCode(Compilateur))

  CloseProgram(Compilateur) ; Ferme la connexion vers le programme
EndIf

MessageRequester("Sortie", Sortie$)
```

Sous Windows RunProgram() utilise l'application par défaut associée à un type de fichier. Par exemple : RunProgram("Test.html") ouvrira le navigateur par défaut configuré sur votre système.

OS Supportés

Tous

130.21 SetEnvironmentVariable

Syntaxe

```
SetEnvironmentVariable(Nom$, Valeur$)
```

Description

Crée une nouvelle variable d'environnement.

Arguments

Nom$ Le nom de la nouvelle variable.

Valeur$ La valeur associée à la variable.

Valeur de retour

Aucune.

Remarques

Si la variable d'environnement existait déjà, sa valeur est automatiquement remplacée. Les variables d'environnement sont automatiquement héritées par les programmes lancés avec RunProgram() , donc il est possible de passer un grand nombre d'informations au nouveau programme en utilisant cette méthode (le programme exécuté pourra utiliser GetEnvironmentVariable() pour lire le contenu des variables).

Voir aussi

GetEnvironmentVariable()

OS Supportés

Tous

130.22 WaitProgram

Syntaxe

```
Resultat =
    WaitProgram(Programme [,
    Minuteur])
```

Description

Arrête l'exécution du code jusqu'à ce que le Programme spécifié se termine, ou que le minuteur expire.

Arguments

Programme Le programme à utiliser. Le programme doit avoir été lancé avec RunProgram() .

Minuteur (optionnel) Le délai à utiliser, en millisecondes.

Valeur de retour

Renvoie une valeur non nulle si le programme a quitté, zéro si le délai a été atteint.

OS Supportés

Tous

130.23 WriteProgramData

Syntaxe

```
Resultat =
    WriteProgramData(Programme,
    *Memoire, Taille)
```

Description

Ecrit des données dans l'entrée standard (stdin) d'un programme.

Arguments

Programme Le programme à utiliser. Il doit être lancé avant avec RunProgram() avec l'option #PB_Program_Write.

***Memoire** La mémoire tampon qui contient les données à écrire. La valeur spéciale #PB_Program_Eof peut être passée dans le paramètre '*Memoire' pour que le programme reçoive un signal EOF (fin de fichier) ce qui indique qu'il n'y a plus de données à transmettre. Le paramètre 'Taille' est ignoré dans ce cas. Après avoir appelé WriteProgramData() avec cette valeur spéciale, plus rien ne peut être écrit dans l'entrée standard du programme.

Taille La taille des données à écrire.

Valeur de retour

Le nombre d'octet effectivement écrit.

Voir aussi

WriteProgramString() , WriteProgramStringN()

OS Supportés

Tous

130.24 WriteProgramString

Syntaxe

```
WriteProgramString(Programme,
    Texte$ [, Options])
```

Description

Ecrit un texte dans l'entrée standard (stdin) d'un programme.

Arguments

Programme Le programme à utiliser. Il doit être lancé avant avec RunProgram() avec l'option #PB_Program_Write.

Texte$ Le texte à écrire avec un caractère retour à la ligne à la fin. Pour inclure le saut de ligne, WriteProgramStringN() peut être utilisé. Pour écrire des données brutes, WriteProgramData() peut être utilisé. Cette fonction peut également être utilisée pour envoyer un EOF (fin de fichier) qui indique au programme qu'il n'y a plus de données.

Options (optionnel) Le format de la chaîne de caractères à utiliser. Le format par défaut peut être affecté avec les options #PB_Program_Ascii, #PB_Program_Unicode et #PB_Program_UTF8 de la fonction RunProgram() . Peut prendre l'une des valeurs suivantes

```
#PB_Ascii   : Écrit la
  chaîne en ascii
#PB_UTF8    : Écrit la
  chaîne en UTF8 (Par
  défaut)
#PB_Unicode: Écrit la
  chaîne en unicode
```

Valeur de retour

Aucune.

Voir aussi

WriteProgramStringN() , ReadProgramString()

OS Supportés

Tous

130.25 WriteProgramStringN

Syntaxe

```
WriteProgramStringN(Programme,
   Texte$ [, Options])
```

Description

Ecrit un texte dans l'entrée standard (stdin) d'un programme en ajoutant automatiquement un caractère retour à la ligne.

Arguments

Programme Le programme à utiliser. Il doit être lancé avant avec RunProgram() avec l'option #PB_Program_Write.

Texte$ Le texte à écrire avec un caractère retour à la ligne à la fin.
Pour écrire un texte sans retour de ligne, la commande WriteProgramString() est disponible.
Pour écrire autre chose que du texte dans l'entrée standard, la commande WriteProgramData() est disponible. Elle permet entre autre d'envoyer le signal #PB_Program_Eof (EOF, End Of File) qui indique au programme qu'il n'y a plus d'entrée.

Options (optionnel) Le format de la chaîne de caractères à utiliser.
Le format par défaut peut être affecté avec les options #PB_Program_Ascii, #PB_Program_Unicode et #PB_Program_UTF8 de la fonction RunProgram() .
Peut prendre l'une des valeurs suivantes

```
#PB_Ascii   : Écrit la
 chaîne en ascii
#PB_UTF8    : Écrit la
 chaîne en UTF8 (Par
 défaut)
#PB_Unicode: Écrit la
 chaîne en unicode
```

Valeur de retour

Aucune.

Voir aussi

WriteProgramString() , ReadProgramString()

OS Supportés

Tous

Chapitre 131

RegularExpression

Généralités

Les expressions régulières permettent de rechercher, extraire ou remplacer n'importe quel texte dans une chaîne de caractères qui correspond à un motif (pattern) donné. Ces motifs sont souvent assez difficiles à écrire et à appréhender, mais quand ils sont correctement utilisés ils rendent beaucoup de manipulations autour des chaînes de caractères plus aisées. De fait, cette bibliothèque n'est pas pour les néophytes et il est conseillé d'avoir de solides bases en PureBasic et en programmation en général pour l'utiliser.
Cette bibliothèque utilise PCRE, qui est une implémentation OpenSource des expressions régulières de Perl. Toutes les expressions régulières supportées par PCRE seront reconnues dans PureBasic. Pour avoir une liste complète des motifs et des arguments disponibles, se rendre sur la page internet de PCRE :
http ://www.pcre.org/pcre.txt
La license PCRE est consultable ici .
Important : La licence de PCRE est très permissive, et permet son utilisation gratuite dans n'importe quel type de projet (commercial ou non) à partir du moment où le texte de la licence est fourni avec l'application lors de sa distribution. Donc si vous utilisez cette bibliothèque, vous devez inclure un fichier contenant la licence mentionnée ci-dessus.

OS Supportés

Tous

131.1 CountRegularExpressionGroups

Syntaxe

```
Resultat =
    CountRegularExpressionGroups(#RegularExpression)
```

Description

Renvoie le nombre de groupes définis dans une expression régulière.
Les groupes d'expressions régulières peuvent être accessibles avec des fonctions comme RegularExpressionGroup() .

Arguments

#**ExpressionReguliere** L'expression régulière à utiliser.

Valeur de retour

Le nombre de groupes définis dans l'expression régulière.

Remarques

Les groupes dans une expression régulière sont définis en entourant une sous-expression avec des parenthèses "(" et ")".
Les groupes sont numérotés tels qu'ils apparaissent dans l'expression régulière de gauche à droite.
Le premier groupe a l'index 1.

Voir aussi

RegularExpressionGroup()

OS Supportés

Tous

131.2 CreateRegularExpression

Syntaxe

```
Resultat =
    CreateRegularExpression(#ExpressionReguliere,
    Motif$ [, Options])
```

Description

Crée une nouvelle expression régulière.

Arguments

#**ExpressionReguliere** L'identifiant de l'expression régulière.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

**Motif$** 'Motif$' est une chaîne de caractères qui contient la règle qui sera appliquée pour chercher, extraire ou remplacer du texte.

Options (optionnel) Peut prendre une des valeurs suivantes :

```
#PB_RegularExpression_DotAll
    : '.' prend tout en
 compte, y compris les
 retours à la ligne.
#PB_RegularExpression_Extended
  : Les espaces et les
 commentaires commençant
 par '#' sont ignorés.
#PB_RegularExpression_MultiLine
 : '^' et '$' prennent en
 compte les retours à la
 ligne dans les données.
#PB_RegularExpression_AnyNewLine:
 'CR', 'LF', et 'CRLF'
 sont considérés comme
 des retours à la ligne.
#PB_RegularExpression_NoCase
    : Insensible à la
 casse.
```

Valeur de retour

Renvoie une valeur non nulle si l'expression régulière a été créée avec succès, zéro sinon.

Remarques

Si une erreur a été détectée dans la définition du motif, la fonction renvoie zéro. Voir RegularExpressionError() .
Si une expression régulière n'est plus utilisée, il est possible de libérer ses ressources à l'aide de la fonction FreeRegularExpression() .

Exemple

```
1  ; Cette expression repérera
    chaque mot de 3 lettres
    qui commence par une
    minuscule,
2  ; suivi du caractère 'b' et
    qui se termine par une
    majuscule. ex: abC
3  ;
4  If
    CreateRegularExpression(0,
    "[a-z]b[A-Z]")
5    Debug
    MatchRegularExpression(0,
    "abC") ; Donnera 1
```

```
6     Debug
      MatchRegularExpression(0,
      "abc") ; Donnera 0
7   Else
8     Debug
      RegularExpressionError()
9   EndIf
```

Voir aussi

RegularExpressionError() ,
FreeRegularExpression()

OS Supportés

Tous

131.3 ExamineRegularExpression

Syntaxe

```
Resultat =
    ExamineRegularExpression(#ExpressionReguliere,
    Chaîne$)
```

Description

Démarre la comparaison entre une expression réglière et une chaîne de caractère. Chaque correspondance peut être trouvée en utilisant la fonction NextRegularExpressionMatch() . Chaque correspondance comprend : la chaîne correspondante, sa position, sa longueur et des groupes au sein de la correspondance peuvent être extraits avec les fonctions appropriées.

Arguments

#ExpressionReguliere L'expression régulière à utiliser.

Chaîne$ La chaîne de caractères à tester.

Valeur de retour

Renvoie une valeur non nulle si l'examen de la correspondance a été lancé avec succès, zéro sinon.

Exemple

```
1   ; On cherche chaque mot de
      3 lettres qui commence par
      une lettre minuscule,
```

```
  ; suivie par le caractère 'b' et qui se termine par une lettre majuscule. ex: abC
  ; Chaque match est affiché avec sa position dans la chaîne d'origine.
  ;
  If CreateRegularExpression(0, "[a-z]b[A-Z]")
    If ExamineRegularExpression(0, "abC ABc zbA abc")
      While NextRegularExpressionMatch(0)
        Debug "Match: " + RegularExpressionMatchString(0)
        Debug "    Position: " + Str(RegularExpressionMatchPosition(0))
        Debug "    Longueur: " + Str(RegularExpressionMatchLength(0))
      Wend
    EndIf
  Else
    Debug RegularExpressionError()
  EndIf
```

Voir aussi

NextRegularExpressionMatch() ,
RegularExpressionMatchString() ,
RegularExpressionMatchPosition() ,
RegularExpressionMatchLength()

OS Supportés

Tous

131.4 ExtractRegularExpression

Syntaxe

```
Resultat = ExtractRegularExpression(#ExpressionReguliere, Texte$, Tableau$())
```

Description

Extrait toutes les parties de la chaîne de caractères correspondant à une expression régulière vers un tableau.

Arguments

#ExpressionReguliere L'identifiant de l'expression régulière.

Texte$ Le texte à tester.

Tableau$() Il est automatiquement redimensionné pour contenir les éléments trouvés dans le 'Texte$'.

Valeur de retour

Le nombre d'éléments trouvés.

Exemple

```
; Cette expression repérera chaque mot de 3 lettres qui commence par une minuscule,
; suivi du caractère 'b' et qui se termine par une majuscule. ex: abC
;
If CreateRegularExpression(0, "[a-z]b[A-Z]")
  Dim Resultat$(0)
  Nb = ExtractRegularExpression(0, "abC ABc zbA abc", Resultat$())
  For k = 0 To Nb-1
    Debug Resultat$(k)
  Next
Else
  Debug RegularExpressionError()
EndIf
```

Voir aussi

CreateRegularExpression()

OS Supportés

Tous

131.5 FreeRegularExpression

Syntaxe

```
FreeRegularExpression(#ExpressionReguliere)
```

Description

Libère une expression régulière ainsi que la mémoire associée.

Arguments

#**ExpressionReguliere** L'expression régulière à supprimer.
Si #PB_All est spécifié, toutes les expressions régulières restantes sont libérées.

Valeur de retour

Aucune.

Remarques

Toutes les expressions régulières restantes sont automatiquement libérées quand le programme se termine.

Voir aussi

CreateRegularExpression()

OS Supportés

Tous

131.6 IsRegularExpression

Syntaxe

```
Resultat =
    IsRegularExpression(#ExpressionReguliere)
```

Description

Teste si une expression régulière est correctement initialisée.

Arguments

#**ExpressionReguliere** L'identifiant de l'expression régulière.

Valeur de retour

Revoie une valeur non nulle si l'expression régulière est valide.

Remarques

Cette fonction a été conçue pour accepter n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.

Voir aussi

CreateRegularExpression()

OS Supportés

Tous

131.7 MatchRegularExpression

Syntaxe

```
Resultat =
    MatchRegularExpression(#ExpressionReguliere,
    Texte$)
```

Description

Teste si une chaîne de caractères correspond à une expression régulière.

Arguments

#ExpressionReguliere L'identifiant de l'expression régulière.

Texte$ La chaîne de caractères à tester.

Valeur de retour

Renvoie une valeur non nulle s'il existe une ou plusieurs correspondances, zéro sinon.

Exemple

```
1   ; Cette expression repérera
     chaque mot de 3 lettres
     qui commence par une
     minuscule,
2   ; suivi du caractère 'b' et
     qui se termine par une
     majuscule. ex: abC
3   ;
4   If
     CreateRegularExpression(0,
     "[a-z]b[A-Z]")
5     If
     MatchRegularExpression(0,
     "abC ABc zbA abc")
6       Debug "Le texte
     correspond !"
7     Else
8       Debug "Aucun motif
     trouvé dans le texte"
9     EndIf
10  Else
11    Debug
     RegularExpressionError()
12  EndIf
```

Voir aussi

CreateRegularExpression()

OS Supportés

Tous

131.8 NextRegularExpressionMatch

Syntaxe

```
Resultat =
    NextRegularExpressionMatch(#ExpressionReguliere)
```

Description

Recherche d'une nouvelle correspondance après un appel à ExamineRegularExpression() .

Arguments

#**ExpressionReguliere** L'identifiant de l'expression régulière.

Valeur de retour

Renvoie une valeur non nulle s'il existe une correspondance, zéro sinon.

Remarques

Les fonctions suivantes peuvent être utilisées pour obtenir des informations sur la correspondance en cours :
- RegularExpressionMatchString() : Renvoie la chaîne correspondante
- RegularExpressionMatchPosition() : Renvoie la position
- RegularExpressionMatchLength() : Renvoie la longueur
- RegularExpressionGroup() : Extrait la chaîne de caractères d'un groupe
- RegularExpressionGroupPosition() : Renvoie la position d'un groupe
- RegularExpressionGroupLength() : Renvoie la longueur d'un groupe
- RegularExpressionNamedGroup() : Extrait la chaîne de caractères d'un groupe nommé
- RegularExpressionNamedGroupPosition() : Renvoie la position d'un groupe nommé
- RegularExpressionNamedGroupLength() : Renvoie la longueur d'un groupe nommé

Exemple

Voir ExamineRegularExpression() .

Voir aussi

ExamineRegularExpression()

OS Supportés

Tous

131.9 RegularExpressionMatchString

Syntaxe

```
Resultat\$ =
    RegularExpressionMatchString(#ExpressionReguliere)
```

Description

Renvoie la chaîne de caractères après un appel à ExamineRegularExpression() et à NextRegularExpressionMatch() .

Arguments

#**ExpressionReguliere** L'identifiant de l'expression régulière.

Valeur de retour

La sous-chaîne de caractères qui correspond à l'expression régulière.

Exemple

Voir ExamineRegularExpression() .

Voir aussi

ExamineRegularExpression() , NextRegularExpressionMatch() , RegularExpressionMatchPosition() , RegularExpressionMatchLength()

OS Supportés

Tous

131.10 RegularExpressionMatchPosition

Syntaxe

```
Resultat =
    RegularExpressionMatchPosition(#ExpressionReguliere)
```

Description

Renvoie la position dans la chaîne après un appel à ExamineRegularExpression() et à NextRegularExpressionMatch() .

Arguments

#**ExpressionReguliere** L'identifiant de l'expression régulière.

Valeur de retour

La position à l'intérieur de la chaîne originale.
Le premier caractère de la chaîne est à la position 1.

Exemple

Voir ExamineRegularExpression() .

Voir aussi

ExamineRegularExpression() , NextRegularExpressionMatch() , RegularExpressionMatchString() , RegularExpressionMatchLength()

OS Supportés

Tous

131.11 RegularExpressionMatchLength

Syntaxe

```
Resultat =
    RegularExpressionMatchLength(#ExpressionReguliere)
```

Description

Renvoie la longueur en caractères après un appel à ExamineRegularExpression() et à NextRegularExpressionMatch() .

Arguments

#**ExpressionReguliere** L'identifiant de l'expression régulière.

Valeur de retour

La longueur de la correspondance en caractères.

Exemple

Voir ExamineRegularExpression() .

Voir aussi

ExamineRegularExpression() , NextRegularExpressionMatch() , RegularExpressionMatchString() , RegularExpressionMatchPosition()

OS Supportés

Tous

131.12 RegularExpressionGroup

Syntaxe

```
Resultat\$ =
    RegularExpressionGroup(#ExpressionReguliere,
    Groupe)
```

Description

Extrait la chaîne de caractères d'un groupe après un appel à ExamineRegularExpression() et à NextRegularExpressionMatch() .

Arguments

#ExpressionReguliere L'identifiant de l'expression régulière.

Groupe L'index du groupe à extraire. Le premier groupe a pour index 1.

Valeur de retour

Renvoie la chaîne de caractères qui correspond au groupe d'expression regulière.

Remarques

Les Groupes dans une expression régulière sont définis en entourant une sous-expression avec des parenthèses "(" et ")".
Les groupes sont numérotés comme ils apparaissent dans l'expression régulière de gauche à droite. Le premier groupe a l'index 1.
La fonction CountRegularExpressionGroups() peut être utilisée pour connaître le nombre de groupes dans une expression régulière.
Une variante est l'utilisation de la fonction RegularExpressionNamedGroup() .

Exemple

```
1  ; Cette expression
   correspond à une couleur
   (avec la valeur rouge,
   vert et bleu)
2  ; Les couleurs sont
   regroupées avec des () et
   la valeur de la couleur
   est extraite
```

```
  ; dans le cas d'une correspondance.
  ;
  If CreateRegularExpression(0, "color=(red|green|blue)")
    If ExamineRegularExpression(0, "stype=bold, color=blue, margin=50")
      While NextRegularExpressionMatch(0)
        Debug "La couleur est " + RegularExpressionGroup(0, 1)
      Wend
    EndIf
  Else
    Debug RegularExpressionError()
  EndIf
```

Voir aussi

ExamineRegularExpression() ,
NextRegularExpressionMatch() ,
RegularExpressionGroupPosition() ,
RegularExpressionGroupLength() ,
RegularExpressionNamedGroup()

OS Supportés

Tous

131.13 RegularExpressionGroupPosition

Syntaxe

```
Resultat = RegularExpressionGroupPosition(#ExpressionReguliere, Groupe)
```

Description

Renvoie la position (dans la chaîne) du groupe après un appel à ExamineRegularExpression() et à NextRegularExpressionMatch() .

Arguments

#ExpressionReguliere L'identifiant de l'expression régulière.

Group L'index du groupe à extraire. Le premier groupe a pour index 1.

Valeur de retour

Renvoie la position du caractère du groupe au sein de la chaîne correspondante (pas dans la chaîne d'entrée originale !).
Ce premier caractère a la position 1.

Remarques

Les Groupes dans une expression régulière sont définis en entourant une sous-expression avec des parenthèses "(" et ")".
Les groupes sont numérotés comme ils apparaissent dans l'expression régulière de gauche à droite. Le premier groupe a l'index 1.
La fonction CountRegularExpressionGroups() peut être utilisée pour connaître le nombre de groupes dans une expression régulière.
Une variante est l'utilisation de la fonction RegularExpressionNamedGroupPosition() .

Voir aussi

ExamineRegularExpression() , NextRegularExpressionMatch() , RegularExpressionGroup() , RegularExpressionGroupLength() , RegularExpressionNamedGroupPosition()

OS Supportés

Tous

131.14 RegularExpressionGroupLength

Syntaxe

```
Resultat =
    RegularExpressionGroupLength(#ExpressionReguliere,
    Groupe)
```

Description

Renvoie la longueur du groupe après un appel à ExamineRegularExpression() et à NextRegularExpressionMatch() .

Arguments

#ExpressionReguliere L'identifiant de l'expression régulière.

Group L'index du groupe à extraire. Le premier groupe a pour index 1.

Valeur de retour

Renvoie la position du caractère du groupe.

Remarques

Les Groupes dans une expression régulière sont définis en entourant une sous-expression avec des parenthèses "(" et ")".
Les groupes sont numérotés comme ils apparaissent dans l'expression régulière de gauche à droite. Le premier groupe a l'index 1.
La fonction CountRegularExpressionGroups() peut être utilisée pour connaître le nombre de groupes dans une expression régulière.
Une variante est l'utilisation de la fonction RegularExpressionNamedGroupLength() .

Voir aussi

ExamineRegularExpression() , NextRegularExpressionMatch() , RegularExpressionGroup() , RegularExpressionGroupPosition() , RegularExpressionNamedGroupLength()

OS Supportés

Tous

131.15 RegularExpressionNamedGroup

Syntaxe

```
Resultat\$ =
    RegularExpressionNamedGroup(#ExpressionReguliere,
    GroupeNom$)
```

Description

Extrait la chaîne trouvée par un groupe nommé après un appel à ExamineRegularExpression() et à NextRegularExpressionMatch() .

Arguments

#ExpressionReguliere L'identifiant de l'expression régulière.

GroupeNom$ Le nom du groupe à extraire.
Le nom du groupe est sensible à la casse.

Valeur de retour

Renvoie la chaîne trouvée par le groupe d'expression régulière.

Remarques

Les groupes dans une expression régulière sont définis en entourant une sous-expression avec des parenthèses "(" et ")". Les groupes sont accessibles soit par leur index en utilisant RegularExpressionGroup() soit par leur nom en utilisant la syntaxe "(? <Nom>)" comme le montre l'exemple ci-dessous.

Exemple

```
; Cette expression correspond à une couleur (avec la valeur rouge, vert et bleu)
; Les couleurs sont regroupées avec des () et un nom "col" puis la valeur de la
; couleur est extraite dans le cas d'une correspondance.
;
If CreateRegularExpression(0, "color=(?<col>red|green|blue)")
  If ExamineRegularExpression(0, "stype=bold, color=blue, margin=50")
    While NextRegularExpressionMatch(0)
      Debug "La couleur est " + RegularExpressionNamedGroup(0, "col")
    Wend
  EndIf
Else
  Debug RegularExpressionError()
EndIf
```

Voir aussi

ExamineRegularExpression() , NextRegularExpressionMatch() , RegularExpressionNamedGroupPosition() , RegularExpressionNamedGroupLength() , RegularExpressionGroup()

OS Supportés

Tous

131.16 RegularExpressionNamedGroupPosition

Syntaxe

```
Resultat =
    RegularExpressionNamedGroupPosition(#ExpressionReguliere,
    GroupeNom$)
```

Description

Renvoie la position (au sein de la chaîne correspondante en cours) du groupe nommé spécifié après un appel à ExamineRegularExpression() et à NextRegularExpressionMatch() .

Arguments

#ExpressionReguliere L'identifiant de l'expression régulière.

GroupeNom$ Le nom du groupe à extraire.
Le nom du groupe est sensible à la casse.

Valeur de retour

Renvoie la position du caractère du groupe au sein de la chaîne correspondante (pas dans la chaîne d'entrée originale !).
Le premier caractère de la correspondance a la position 1.

Remarques

Les groupes dans une expression régulière sont définis en entourant une sous-expression avec des parenthèses "(" et ")". Les groupes sont accessibles soit par leur index en utilisant RegularExpressionGroup() soit par leur nom en utilisant la syntaxe "(?<Nom>)" comme le montre l'exemple RegularExpressionNamedGroup() .

Voir aussi

ExamineRegularExpression() , NextRegularExpressionMatch() , RegularExpressionNamedGroup() , RegularExpressionNamedGroupLength() , RegularExpressionGroupPosition()

OS Supportés

Tous

131.17 RegularExpressionNamedGroupLength

Syntaxe

```
Resultat =
    RegularExpressionNamedGroupLength(#ExpressionReguliere,
    GroupeNom$)
```

Description

Renvoie la longueur au sein de la chaîne correspondante en cours) du groupe nommé spécifié après un appel à ExamineRegularExpression() et à NextRegularExpressionMatch() .

Arguments

#ExpressionReguliere L'identifiant de l'expression régulière.

GroupeNom$ Le nom du groupe à extraire.
Le nom du groupe est sensible à la casse.

Valeur de retour

Renvoie la position du caractère du groupe.

Remarques

Les groupes dans une expression régulière sont définis en entourant une sous-expression avec des parenthèses "(" et ")". Les groupes sont accessibles soit par leur index en utilisant RegularExpressionGroup() soit par leur nom en utilisant la syntaxe "(? <Nom>)" comme le montre l'exemple RegularExpressionNamedGroup() .

Voir aussi

ExamineRegularExpression() , NextRegularExpressionMatch() , RegularExpressionNamedGroup() , RegularExpressionNamedGroupPosition() , RegularExpressionGroupLength()

OS Supportés

Tous

131.18 ReplaceRegularExpression

Syntaxe

```
Resultat\$ =
    ReplaceRegularExpression(#ExpressionReguliere,
    Texte$, TexteRemplace$)
```

Description

Remplace une chaîne de caractères par une autre.

Arguments

#ExpressionReguliere L'identifiant de l'expression régulière.

Texte$ La chaîne de caractères à remplacer.

TexteRemplace$ La chaîne de caractères de remplacement.

Valeur de retour

Renvoie la nouvelle chaîne de caractères.

Remarques

Les références arrières (généralement décrites comme \ 1, \ 2, etc) ne sont pas prises en charge.
ExtractRegularExpression() combinée à ReplaceString() devrait obtenir le comportement demandé.

Exemple

```
1  ; Cette expression repérera
     chaque mot de 3 lettres
     qui commence par une
     minuscule,
2  ; suivi du caractère 'b' et
     qui se termine par une
     majuscule. ex: abC
3  ;
4  If
     CreateRegularExpression(0,
     "[a-z]b[A-Z]")
5    Resultat$ =
     ReplaceRegularExpression(0,
     "abC ABc zbA abc", "---")
6    Debug Resultat$     ;
     Affichera "--- ABc --- abc"
7  Else
8    Debug
     RegularExpressionError()
9  EndIf
```

Voir aussi

CreateRegularExpression()

OS Supportés

Tous

131.19 RegularExpressionError

Syntaxe

```
Resultat\$ =
    RegularExpressionError()
```

Description

Renvoie sous forme textuelle (en anglais) l'erreur survenue lors du dernier appel à CreateRegularExpression() .

Arguments

Aucun.

Valeur de retour

Renvoie l'erreur.

Exemple

```
1  ; Un crochet '[' supplémentaire a été ajouté, donc il y a une erreur de syntaxe dans
2  ; l'expression régulière
3  ;
4  If CreateRegularExpression(0, "[a-z]b[[A-Z][]")
5    Debug "Succès"
6  Else
7    Debug RegularExpressionError()
8  EndIf
```

Voir aussi

CreateRegularExpression()

OS Supportés

Tous

Chapitre 132

Requester

Généralités

Les Requesters sont ces 'boîtes de dialogue' que l'on trouve dans pratiquement toutes les applications avec interface graphique. Elles sont très pratiques pour effectuer simplement des tâches de base parfois complexes (ouvrir un fichier, choisir une couleur ou une police ...) qui ont chacune leur boîte de dialogue spécialisée.

OS Supportés

Tous

132.1 ColorRequester

Syntaxe

```
Resultat =
    ColorRequester([Couleur])
```

Description

Ouvre une boîte de dialogue standard pour choisir une couleur 24 bits (3 octets).

Arguments

Couleur (optionnel) Couleur par défaut lors de l'ouverture de la boîte.

Valeur de retour

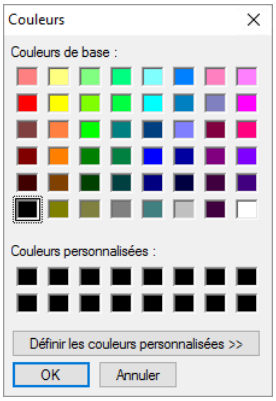
La couleur 24 bits (3 octets) sélectionnée ou -1 ou si l'utilisateur a annulé la sélection. Pour récupérer facilement la valeur de chacune des composantes, utiliser les commandes Red() , Green() et Blue() .

Remarques

La bibliothèque VectorDrawing fonctionne avec les couleurs 32 bits(4 octets), comprenant la transparence alpha. Vous trouverez ci-dessous un exemple de code pour ajouter la transparence à une couleur 24 bits.

Exemple

```
Couleur = ColorRequester()
If Couleur > -1
  a$ = "Vous avez sélectionné la couleur suivante:" + Chr(10)  ; Chr(10) est seulement nécessaire
  a$ + "Valeur 24 Bits: " + Str(Couleur)
  + Chr(10)  ; pour les retours à la ligne
  a$ + "Composante rouge: " + Str(Red(Couleur))
    + Chr(10)
  a$ + "Composante verte: " + Str(Green(Couleur))
  + Chr(10)
  a$ + "Composante bleue: " + Str(Blue(Couleur))
Else
  a$ = "La sélection a été annulée."
EndIf
MessageRequester("Information",a$,0)
End
```



OS Supportés

Tous

132.2 FontRequester

Syntaxe

```
Resultat =
    FontRequester(Police$,
    TaillePolice, Options [,
    Couleur [, Style]])
```

Description

Ouvre une boîte de dialogue standard pour choisir une police de caractères.

Arguments

Police$ Le nom de la police affichée par défaut lors de l'ouverture de la boîte de dialogue.

TaillePolice La taille de la police affichée par défaut lors de l'ouverture de la boîte de dialogue.

Options Peut être l'une des valeurs suivantes :

```
#PB_FontRequester_Effects
 : Affiche les options
 effets (Barré, Souligné,
 Couleur, etc) (Windows
 uniquement).
```

Couleur (optionnel) La couleur de la police affichée par défaut lors de l'ouverture de la boîte de dialogue. Utiliser RGB() pour obtenir une valeur de couleur valide.

Style (optionnel) Le style de la police affichée par défaut lors de l'ouverture de la boîte de dialogue.
Voir SelectedFontStyle() pour obtenir les styles disponibles.

Valeur de retour

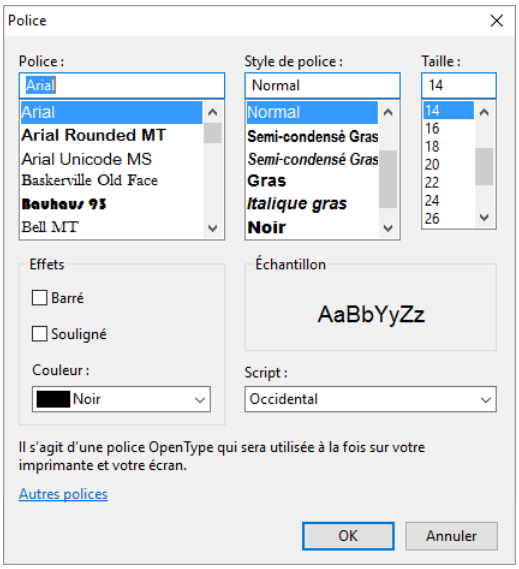
Renvoie une valeur non nulle si une police a été choisie, zéro si l'utilisateur a annulé la boîte de dialogue.

Remarques

Les fonctions SelectedFontColor() , SelectedFontName() , SelectedFontSize() et SelectedFontStyle() peuvent être utilisées après un appel réussi à cette fonction pour obtenir les informations sur la police sélectionnée.

Exemple

```
Police$ = "Arial"    ; Police initiale (peut aussi être nulle)
TaillePolice  = 14   ; Taille initiale (peut aussi être nulle)
Resultat = FontRequester(Police$, TaillePolice, #PB_FontRequester_Effects)
If Resultat
  Message$ = "Vous avez sélectionné la police suivante :"  + #LF$
   Message$ + "Nom :  " + SelectedFontName()        + #LF$
   Message$ + "Taille :  " + Str(SelectedFontSize())  + #LF$
   Message$ + "Couleur : " + Str(SelectedFontColor()) + #LF$
   If SelectedFontStyle() & #PB_Font_Bold
      Message$ + "Gras" + #LF$
   EndIf
   If SelectedFontStyle() & #PB_Font_StrikeOut
      Message$ + "Barré" + #LF$
   EndIf
   If SelectedFontStyle() & #PB_Font_Underline
      Message$ + "Souligné" + #LF$
   EndIf
Else
  Message$ = "La sélection a été annulée."
EndIf

MessageRequester("FontRequester", Message$, #PB_MessageRequester_Ok)
```



Voir aussi

SelectedFontColor() , SelectedFontName() , SelectedFontSize() , SelectedFontStyle()

OS Supportés

Tous

132.3 InputRequester

Syntaxe

```
Resultat\$ =
    InputRequester(Titre$,
    Message$, TexteParDefaut$
    [, Option])
```

Description

Ouvre une boîte de dialogue permettant la saisie de texte.

Arguments

Titre$ Titre de la boîte de dialogue.

Message$ Message affiché avant le champ de saisie.

TexteParDefaut$ Texte par défaut du champ de saisie.

Option (optionnel)

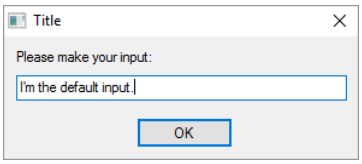
```
      #PB_InputRequester_Password:
    Définir le champ en type
    'mot de passe'.
                                Le
    texte affiché sera caché.
```

Valeur de retour

Renvoie le texte du champ de saisie, ou une chaîne de caractères vide si l'utilisateur a annulé la boîte.

Exemple

```
Saisie$ =
  InputRequester("InputRequester","Saisissez
  un texte :","Texte par
  défaut.")

If Saisie$ > ""
   a$ = "Vous avez écrit le
  texte suivant :" + Chr(10)
   ; Chr(10) nécessaire
  seulement
   a$ + Saisie$

  ; pour faire un saut de
  ligne
Else
   a$ = "La saisie a été
  annulée ou il n'y a pas de
  texte saisi."
EndIf
MessageRequester("Information",a$,0)
End
```



OS Supportés

Tous

132.4 MessageRequester

Syntaxe

```
Resultat =
    MessageRequester(Titre$,
    Message$ [, Options])
```

Description

Ouvre une boîte de dialogue permettant d'afficher un message destiné à fournir des informations à l'utilisateur. L'exécution du programme est interrompue tant que l'utilisateur n'a pas refermé la boîte.

Arguments

Titre$ Titre de la boîte de dialogue.

Message$ Message affiché dans la boîte de dialogue.

Option (optionnel) Peut être l'une des valeurs suivantes :

```
#PB_MessageRequester_Ok
          : Pour afficher
 seulement le bouton 'OK'
 (Par défaut)
#PB_MessageRequester_YesNo
       : Pour afficher
 les boutons 'Oui' et
 'Non'
#PB_MessageRequester_YesNoCancel
 : Pour afficher les
 boutons 'Oui', 'Non' et
 'Annuler'
```

Combiné avec l'une des valeurs suivantes :

```
#PB_MessageRequester_Info
   : Affiche une icône
 Info (I majuscule)
#PB_MessageRequester_Warning:
 Affiche une icône
 Attention (!)
#PB_MessageRequester_Error
  : Affiche une icône
 erreur (X)
```

Valeur de retour

Les valeurs de retour sont les suivantes :

```
#PB_MessageRequester_Yes
 : Le bouton 'Oui' a été
 pressé
#PB_MessageRequester_No
 : Le bouton 'Non' a été
 pressé
#PB_MessageRequester_Cancel
 : Le bouton 'Annuler' a
 été pressé
```

Exemple : MessageRequester simple (normalement utilisé uniquement pour afficher des informations)

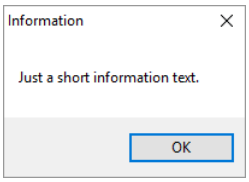
```
1  MessageRequester("MessageRequester
    Simple","Juste un petit
    texte
    d'information.",#PB_MessageRequester_Ok
    |
    #PB_MessageRequester_Info)
```

Exemple : MessageRequester avec boutons Oui / Non (habituellement utilisé pour les questions)

```
Resultat = MessageRequester("MessageRequester avec boutons Oui/Non","Veuillez faire votre choix :",#PB_MessageRequester_YesNo)
a$ = "Le résultat du MessageRequester était : "
If Resultat = 6      ; le bouton Oui a été choisi (Resultat = 6)
  a$ + "Oui"
Else                 ; le bouton Non a été choisi (Resultat = 7)
  a$ + "Non"
EndIf
MessageRequester("Information",a$,#PB_MessageRequester_Ok)
```

Exemple : MessageRequester avec boutons Oui / Non / Annuler (habituellement utilisé pour les questions)

```
Resultat = MessageRequester("MessageRequester avec boutons Oui / Non / Annuler","Veuillez faire votre choix :",#PB_MessageRequester_YesNoCancel)
a$ = "Le résultat du MessageRequester était : "
If Result = 6        ; le bouton Oui a été choisi (Resultat = 6)
  a$ + "Oui"
ElseIf Result = 7    ; le bouton Non a été choisi (Resultat = 7)
  a$ + "Non"
Else                 ; le bouton Annuler a été choisi ou la touche Echap a été appuyée (Resultat = 2)
  a$ + "Annuler"
EndIf
MessageRequester("Information",a$,#PB_MessageRequester_Ok)
```



OS Supportés

Tous

132.5 NextSelectedFileName

Syntaxe

```
Resultat\$ =
    NextSelectedFileName()
```

Description

Après un OpenFileRequester() utilisant la constante #PB_Requester_MultiSelection, cette instruction renvoie le fichier sélectionné suivant, s'il existe.

Arguments

Aucun.

Valeur de retour

Une chaîne de caractères contenant le nom du fichier suivant ou une chaîne vide s'il n'y a plus de fichier à lister.

Exemple

```
1   NomFichier$ =
      OpenFileRequester("Choisissez
      quelques
      fichiers","","",0,
      #PB_Requester_MultiSelection)
2
3   While NomFichier$
4     Debug NomFichier$
5     NomFichier$ =
      NextSelectedFileName()
6   Wend
```

OS Supportés

Tous

132.6 OpenFileRequester

Syntaxe

```
Resultat\$ =
    OpenFileRequester(Titre$,
    FichierParDefaut$,
    Filtre$, FiltrePosition [,
    Options])
```

Description

Ouvre une boîte de dialogue standard pour choisir un fichier.

Arguments

Titre$ Texte de titre de la boîte de dialogue, il remplace le titre par défaut.

FichierParDefaut$ Permet d'initialiser la boîte de dialogue avec un répertoire et un fichier par défaut.

Filtre$ Permet de n'afficher que les fichiers dont le nom et/ou l'extension de fin correspond au filtre.
Il doit avoir la forme des exemples suivants : '*.txt' pour des fichiers texte, '*.mus;*.mod' pour des fichiers musique ayant l'extension '.mus' ou '.mod'.
Chaque filtre est en fait une paire composée du label qui sera visible dans le requester et du filtre proprement dit. Le séparateur est le caractère '|' (Pipe).
Note : Les paramètres 'Filtre$' et 'FiltrePosition' n'ont aucun effet sur Mac OSX (tous les fichiers seront toujours affichés).

FiltrePosition Nombre qui précise le filtre utilisé par défaut.
Valeur entre 0 et le nombre total de filtres moins 1.
Une fois la boîte fermée, SelectedFilePattern() permet de récupérer la position du filtre choisi.
Note : Les paramètres 'Filtre$' et 'FiltrePosition' n'ont aucun effet sur Mac OSX (tous les fichiers seront toujours affichés).

Options (optionnel) Peut être une combinaison des valeurs suivantes :

```
  #PB_Requester_MultiSelection
   : Active la
   multi-sélection. (voir
   NextSelectedFileName()
)
```

Valeur de retour

Renvoie le chemin et le nom du ou des fichiers sélectionnés ou une chaîne de caractères vide si l'utilisateur a annulé la boîte.

Remarques

Le 'NomFichier$' renvoyé peut être découpé facilement en chemin, nom ou extension à l'aide des commandes suivantes : GetFilePart() , GetPathPart() et GetExtensionPart() .

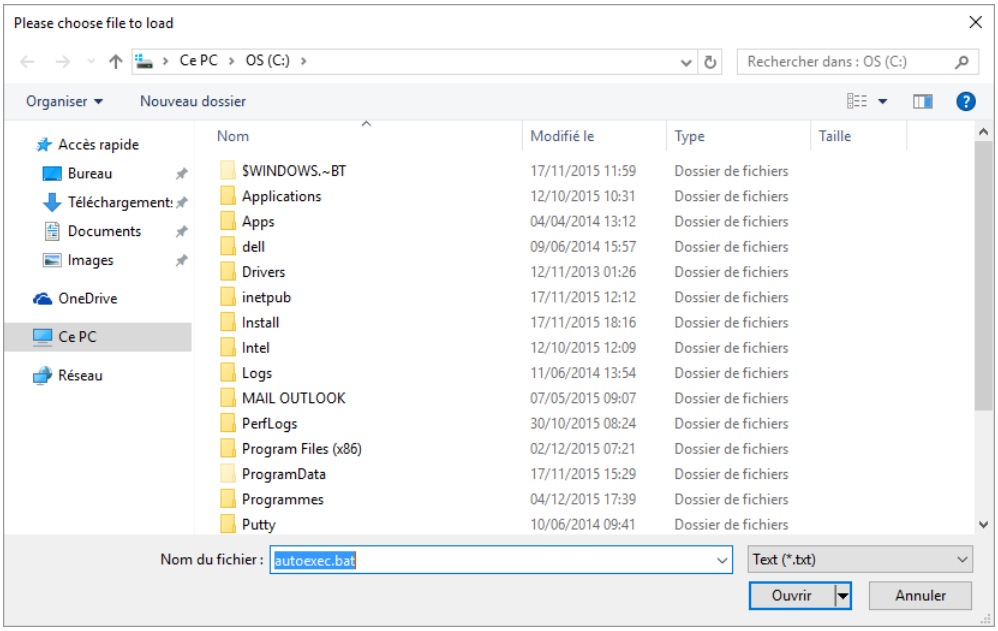
Exemple

```
"Fichiers Textes|*.txt"
            ; Un seul
 filtre avec comme label
 'Fichiers Textes'
"Fichiers
 Textes|*.txt;*.doc"      ;
 Un seul filtre avec *.txt
 et *.doc accepté
"Musiques|*.mp3|Video|*.avi;*.asf"
 ; 2 filtres (Musique et
 video)
```

Exemple

```
1  FichierParDefaut$ =
    "C:\autoexec.bat"   ;
    Répertoire et fichier par
    défaut qui seront affichés
2  ;  Avec la chaîne suivante
    nous allons définir les
    filtres ("|" comme
    séparateur) pour
    l'affichage de fichier :
3  ;  1er  : "Texte (*.txt)"
    comme nom, ".txt" et
    ".bat" comme extension
    autorisée
4  ;  2ème : "PureBasic
    (*.pb)" comme nom, ".pb"
    comme extension autorisée
5  ;  3ème : "Tous les
    fichiers (*.*)" comme nom,
    "*.*" comme extension
    autorisée, valide pour
    tous les fichiers
6  Filtre$ = "Texte
    (*.txt)|*.txt;*.bat|PureBasic
    (*.pb)|*.pb|Tous les
    fichiers (*.*)|*.*"
7  Filtre  = 0    ; utiliser
    par défaut le premier des
    trois filtres possibles
```

```
8   Fichier$ =
     OpenFileRequester("Choisissez
     un fichier à charger",
     FichierParDefaut$,
     Filtre$, Filtre)
9   If Fichier$
10    MessageRequester("Information",
     "Vous avez choisi le
     fichier suivant
     :"+Chr(10)+Fichier$, 0)
11  Else
12    MessageRequester("Information",
     "La sélection a été
     annulée.", 0)
13  EndIf
```



Voir aussi

NextSelectedFileName()

OS Supportés

Tous

132.7 PathRequester

Syntaxe

```
Resultat\$ =
    PathRequester(Titre$,
    CheminInitial$)
```

Description

Ouvre une boîte de dialogue standard de choix de chemin.

Arguments

Titre$ Texte de titre de la boîte de dialogue, il remplace le titre par défaut.

CheminInitial$ Permet d'initialiser la boîte de dialogue sur le chemin spécifié.

Remarques

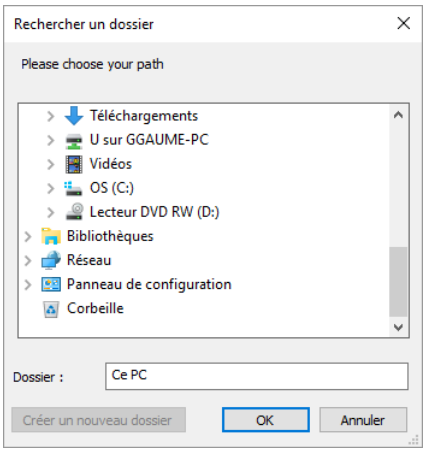
Sous Windows, le chemin est renvoyé avec un '\' de fin.
Sous Linux et OS X, le chemin est renvoyé avec un '/' de fin.

Valeur de retour

Renvoie le chemin sélectionné ou une chaîne de caractères vide si l'utilisateur a annulé la boîte.

Exemple

```
1  CheminInitial$ = "C:\"   ;
    Répertoire par défaut qui
    sera affiché (une chaîne
    vide est aussi possible)
2  Chemin$ =
    PathRequester("Choisissez
    un répertoire",
    CheminInitial$)
3  If Chemin$
4    MessageRequester("Information",
    "Vous avez choisi le
    répertoire suivant
    :"+Chr(10)+Chemin$, 0)
5  Else
6    MessageRequester("Information",
    "La sélection a été
    annulée.", 0)
7  EndIf
```



OS Supportés

Tous

132.8 SaveFileRequester

Syntaxe

```
Resultat\$ =
    SaveFileRequester(Titre$,
    FichierParDefaut$,
    Filtre$, FiltrePosition)
```

Description

Ouvre une boîte de dialogue standard pour enregistrer un fichier.

Arguments

Titre$ Texte de titre de la boîte de dialogue, il remplace le titre par défaut.

FichierParDefaut$ Permet d'initialiser la boîte de dialogue avec un répertoire et un fichier par défaut.

Filtre$ Permet de n'afficher que les fichiers dont le nom et/ou l'extension de fin correspond au filtre.
Il doit avoir la forme des exemples suivants : '*.txt' pour des fichiers texte, '*.mus ;*.mod' pour des fichiers musique ayant l'extension '.mus' ou '.mod'.
Chaque filtre est en fait une paire composée du label qui sera visible dans le requester et du filtre proprement dit. Le séparateur est le caractère '|' (Pipe).
Note : Les paramètres 'Filtre$' et 'FiltrePosition' n'ont aucun effet sur Mac OSX (tous les fichiers seront toujours affichés).

FiltrePosition Nombre qui précise le filtre utilisé par défaut.
Valeur entre 0 et le nombre total de filtres moins 1.
Une fois la boîte fermée, SelectedFilePattern() permet de récupérer la position du filtre choisi.
Note : Les paramètres 'Filtre$' et 'FiltrePosition' n'ont aucun effet sur Mac OSX (tous les fichiers seront toujours affichés).

Valeur de retour

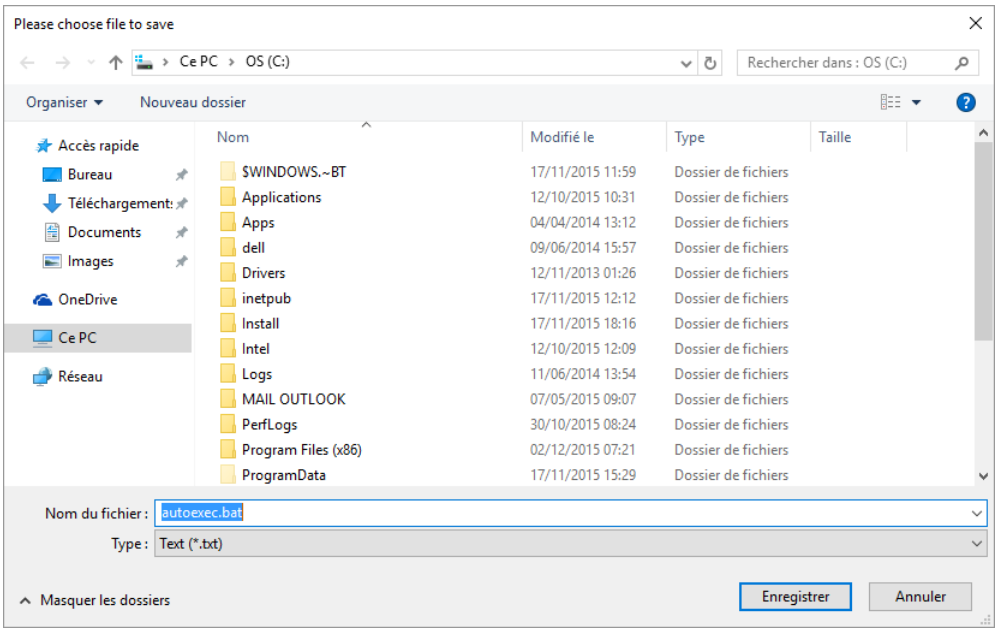
Renvoie le nom du fichier entré ou une chaîne de caractères vide si l'utilisateur a annulé la boîte.

Remarques

Le 'NomFichier$' renvoyé peut être découpé facilement en chemin, nom ou extension à l'aide des commandes suivantes : GetFilePart() , GetPathPart() et GetExtensionPart() .

Exemple

```
FichierParDefaut$ = "C:\autoexec.bat"   ; Répertoire et fichier par défaut qui seront affichés
;  Avec la chaîne suivante nous allons définir les filtres ("|" comme séparateur) pour l'affichage de fichier :
;  1er  : "Texte (*.txt)" comme nom, ".txt" et ".bat" comme extension autorisée
;  2ème : "PureBasic (*.pb)" comme nom, ".pb" comme extension autorisée
;  3ème : "Tous les fichiers (*.*)" comme nom, "*.*" comme extension autorisée, valide pour tous les fichiers
Filtre$ = "Texte (*.txt)|*.txt;*.bat|PureBasic (*.pb)|*.pb|Tous les fichiers (*.*)|*.*"
Filtre = 0    ; utiliser par défaut le premier des trois filtres possibles
Fichier$ = SaveFileRequester("Choisissez un fichier à sauvegarder", FichierParDefaut$, Filtre$, Filtre)
If Fichier$
  MessageRequester("Information", "Vous avez choisi ce fichier :"+Chr(10)+Fichier$, 0)
Else
  MessageRequester("Information", "La sélection a été annulée.", 0)
EndIf
```



OS Supportés

Tous

132.9 SelectedFilePattern

Syntaxe

```
Resultat =
    SelectedFilePattern()
```

Description

Renvoie l'index du filtre sélectionné lors de l'utilisation de la commande OpenFileRequester() ou SaveFileRequester() .

Arguments

Aucun.

Valeur de retour

Renvoie l'index du filtre sélectionné lors de l'utilisation de la commande OpenFileRequester() ou SaveFileRequester() .
L'index du premier filtre est 0.

Exemple

```
FichierParDefaut$ =
  "C:\autoexec.bat"     ;
  Répertoire et fichier par
  défaut
```

```
2  Filtre$ = "Texte (*.txt)|*.txt;*.bat|"   ; Premier filtre   (index = 0)
3  Filtre$ + "PureBasic (*.pb)|*.pb|"       ; Deuxième filtre  (index = 1)
4  Filtre$ + "Bmp (*.bmp)|*.bmp|" ; Troisième filtre (index = 2)
5  Filtre$ + "Jpeg (*.jpg)|*.jpg|"          ; Quatrième filtre (index = 3)
6  Filtre$ + "Tous les fichiers (*.*)|*.*"  ; Cinquième filtre (index = 4)
7  Filtre = 1  ; Le deuxième filtre sera utilisé par défaut.
8
9  ; Ouvre une boîte de dialogue pour choisir un fichier, vous pouvez changer de filtre et obtenir son index après validation
10 Fichier$ = OpenFileRequester("Choisissez un fichier", FichierParDefaut$, Filtre$, Filtre)
11 Index = SelectedFilePattern()
12 If Index > -1
13   MessageRequester("Information", "L'index du filtre sélectionné est le suivant : "+Str(Index), 0)
14 Else
15   MessageRequester("Information", "La sélection a été annulée.", 0)
16 EndIf
```

OS Supportés

Tous

132.10 SelectedFontColor

Syntaxe

```
Resultat = SelectedFontColor()
```

Description

Renvoie la couleur sélectionnée par l'utilisateur avec la fonction FontRequester() .

Arguments

Aucun.

Valeur de retour

Renvoie la couleur sélectionnée par l'utilisateur avec la fonction FontRequester() .
Pour récupérer facilement la valeur de chacune des composantes, utiliser les commandes Red() , Green() et Blue() .

Voir aussi

FontRequester()

OS Supportés

Windows

132.11 SelectedFontName

Syntaxe

```
Resultat\$ =
    SelectedFontName()
```

Description

Renvoie dans le nom de la police sélectionnée par l'utilisateur avec la fonction FontRequester() .

Arguments

Aucun.

Valeur de retour

Renvoie dans le nom de la police sélectionnée par l'utilisateur avec la fonction FontRequester() .
Ce nom peut être utilisé directement par la fonction LoadFont() .

Voir aussi

FontRequester()

OS Supportés

Tous

132.12 SelectedFontSize

Syntaxe

```
Resultat = SelectedFontSize()
```

Description

Renvoie la taille de la police sélectionnée par l'utilisateur avec FontRequester() .

Arguments

Aucun.

Valeur de retour

Renvoie la taille de la police sélectionnée par l'utilisateur avec FontRequester() .

Voir aussi

FontRequester()

OS Supportés

Tous

132.13 SelectedFontStyle

Syntaxe

```
Resultat = SelectedFontStyle()
```

Description

Renvoie le style de la police de caractères sélectionnée par l'utilisateur avec la fonction FontRequester() .

Arguments

Aucun.

Valeur de retour

Renvoie le style de la police de caractères sélectionnée par l'utilisateur avec la fonction FontRequester() .
'Style' peut contenir une ou plusieurs des valeurs suivantes :

```
#PB_Font_Bold        : Le
  style de la police est en
  'Gras'.
#PB_Font_Italic      : Le
  style de la police est en
  'Italique'.
```

```
#PB_Font_StrikeOut   : Le
 style de la police est en
 'Barré'.
#PB_Font_Underline   : Le
 style de la police est en
 'Souligné'.
```

Pour tester si une valeur est présente, il suffit d'utiliser l'opérateur '&' (And) :

```
1  If SelectedFontStyle() &
    #PB_Font_Bold
2    ; La police est en 'Gras'
3  EndIf
```

Voir aussi

FontRequester()

OS Supportés

Tous

Chapitre 133

Runtime

Généralités

Les objets dits "Runtime" (en cours d'exécution du programme) sont les objets d'un programme qui restent accessibles même lorsque le programme est compilé. L'accès à ces objets se fait par l'intermédiaire de leur référence qui est une chaîne de caractères créée à cet effet pendant la compilation.
Pour plus d'informations sur le concept "Runtime", voir Runtime .

OS Supportés

Tous

133.1 GetRuntimeInteger

Syntaxe

```
Resultat =
    GetRuntimeInteger(Objet$)
```

Description

Renvoie la valeur au format 'entier' ('integer') de l'objet runtime ou l'adresse de la procedure si l'objet runtime est une procedure.

Arguments

Objet$ Nom de l'objet.
Les objets suivants sont supportés :

```
- Variable : Le nom d'une
  'Variable' (insensible à
  la casse).
- Constante: Le nom d'une
  '#Constante' (insensible
  à la casse).
- Procedure: Le nom d'une
  'Procedure()'
  (insensible à la casse).
```

Lors d'un accès aux éléments publics d'un module , le nom du préfixe du module est obligatoire, même si UseModule est utilisé.

Valeur de retour

Renvoie la valeur 'entière' de l'objet, zéro si l'objet est introuvable.
Comme zéro est une valeur entière valide, IsRuntime() peut être utilisé pour déterminer si l'objet runtime existe vraiment. Si la variable est de type 'float' ou 'double', elle est automatiquement convertie en entier 'integer'.
Si l'objet runtime est une procédure, la fonction renvoie l'adresse de la procédure.

Voir aussi

SetRuntimeInteger() , IsRuntime()

OS Supportés

Tous

133.2 GetRuntimeDouble

Syntaxe

```
Resultat.d =
    GetRuntimeDouble(Objet$)
```

Description

Renvoie la valeur au format 'double' de l'objet runtime.

Arguments

Objet$ Nom de l'objet.
Les objets suivants sont supportés :

```
- Variable : Le nom d'une
  'Variable' (insensible à
  la casse).
- Constante: Le nom d'une
  '#Constante' (insensible
  à la casse).
- Procedure: Le nom d'une
  'Procedure()'
  (insensible à la casse).
```

Lors d'un accès aux éléments publics d'un module , le nom du préfixe du module est obligatoire, même si UseModule est utilisé.

Valeur de retour

Renvoie la valeur au format 'double' de l'objet ou zéro si l'objet est introuvable. Comme zéro est une valeur entière valide, IsRuntime() peut être utilisé pour déterminer si l'objet runtime existe vraiment. Si la variable est de type 'float' ou 'integer', elle est automatiquement convertie en entier 'double'.

Voir aussi

SetRuntimeDouble() , IsRuntime()

OS Supportés

Tous

133.3 GetRuntimeString

Syntaxe

```
Resultat\$ =
    GetRuntimeString(Objet$)
```

Description

Renvoie la valeur au format 'chaîne de caractères' ('string') de l'objet runtime.

Arguments

Objet$ Nom de l'objet.
Les objets suivants sont supportés :

```
- Variable : Le nom d'une
  'Variable' (insensible à
  la casse).
- Constante: Le nom d'une
  '#Constante' (insensible
  à la casse).
- Procedure: Le nom d'une
  'Procedure()'
  (insensible à la casse).
```

Lors d'un accès aux éléments publics d'un module , le nom du préfixe du module est obligatoire, même si UseModule est utilisé.

Valeur de retour

Renvoie la valeur au format 'chaîne de caractères' de l'objet ou une chaîne vide si l'objet est introuvable.
Comme une chaîne vide est une valeur de chaîne valide, IsRuntime() peut être utilisé pour déterminer si l'objet runtime existe vraiment.

Voir aussi

SetRuntimeString() , IsRuntime()

OS Supportés

Tous

133.4 IsRuntime

Syntaxe

```
Resultat = IsRuntime(Objet$)
```

Description

Vérifie si l'objet spécifié est bien un objet runtime .

Arguments

Objet$ Nom de l'objet.
Les objets suivants sont supportés :

```
- Variable : Le nom d'une
  'Variable' (insensible à
  la casse).
- Constante: Le nom d'une
  '#Constante' (insensible
  à la casse).
- Procedure: Le nom d'une
  'Procedure()'
  (insensible à la casse).
```

Lors d'un accès aux éléments publics d'un module , le nom du préfixe du module est obligatoire, même si UseModule est utilisé.

Valeur de retour

Renvoie une valeur non nulle si l'objet spécifié est un objet runtime , zéro sinon.

OS Supportés

Tous

133.5 SetRuntimeDouble

Syntaxe

```
SetRuntimeDouble(Objet$,
    Valeur.d)
```

Description

Modifie la valeur au format 'double' de l'objet runtime.

Arguments

Objet$ Nom de l'objet. Les objets suivants sont supportés :

```
- Variable : Le nom d'une
  'Variable' (insensible à
  la casse).
```

Lors d'un accès aux éléments publics d'un module , le nom du préfixe du module est obligatoire, même si UseModule est utilisé.

Valeur La nouvelle valeur au format 'double'.

Valeur de retour

Aucune.

Voir aussi

GetRuntimeDouble() , IsRuntime()

OS Supportés

Tous

133.6 SetRuntimeInteger

Syntaxe

```
SetRuntimeInteger(Objet$,
    Valeur.i)
```

Description

Modifie la valeur au format 'entier' ('integer') de l'objet runtime.

Arguments

Objet$ Nom de l'objet. Les objets suivants sont supportés :

```
- Variable : Le nom d'une
  'Variable' (insensible à
  la casse).
```

Lors d'un accès aux éléments publics d'un module , le nom du préfixe du module est obligatoire, même si UseModule est utilisé.

Valeur La nouvelle valeur au format 'integer'.

Valeur de retour

Aucune.

Voir aussi

GetRuntimeInteger() , IsRuntime()

OS Supportés

Tous

133.7 SetRuntimeString

Syntaxe

```
SetRuntimeString(Objet$,
    Valeur$)
```

Description

Modifie la valeur au format 'chaîne de caractères' ('string') de l'objet runtime.

Arguments

Objet$ Nom de l'objet. Les objets suivants sont supportés :

```
- Variable : Le nom d'une
  'Variable' (insensible à
  la casse).
```

Lors d'un accès aux éléments publics d'un module , le nom du préfixe du module est obligatoire, même si UseModule est utilisé.

Valeur$ La nouvelle valeur au format 'chaîne de caractères'.

Valeur de retour

Aucune.

Voir aussi

GetRuntimeInteger() , IsRuntime()

OS Supportés

Tous

Chapitre 134

Scintilla

Généralités

Scintilla est un composant libre d'édition de code source. Il est livré avec son propre code source et une licence autorisant son utilisation dans tout projet libre ou commercial. La licence peut être consultée ici . Le code source, ainsi que la documentation peuvent être trouvés sur le site de Scintilla.
Extrait du site de Scintilla : En plus des fonctionnalités classiques des composants d'édition de texte basique, Scintilla inclut des fonctionnalités particulièrement utiles lors de l'édition et du débogage de code source. Cela inclut la stylisation syntaxique, des indicateurs d'erreurs, l'auto-complétion et des indications sur les appels de fonctions. La marge de sélection peut contenir des marques similaires à celles utilisées dans les débogueurs pour indiquer les points d'arrêt et la ligne courante. Les choix stylistiques sont plus ouverts que la plupart des éditeurs, et permettent l'utilisation de différentes polices, de différentes tailles, les polices grasses et italiques, plusieurs couleurs de texte et de fond, etc...
Important : La licence de Scintilla requiert qu'une indication relative au droit d'auteur soit incluse dans tout logiciel utilisant Scintilla et que le texte de la licence soit fourni dans la documentation du logiciel. PureBasic intègre la bibliothèque Scintilla avec la bibliothèque gadget , si bien que des commandes telles que ResizeGadget() ou HideGadget() peuvent être utilisées avec un contrôle scintilla. De plus elle fournit la fonction ScintillaSendMessage() pour communiquer avec le contrôle pour l'exploiter à son plein potentiel. Toutes les structures et constantes sont définies de base dans PureBasic.

OS Supportés

Tous

134.1 InitScintilla

Syntaxe

```
Resultat =
    InitScintilla([NomBibliotheque$])
```

Description

Charge la dll Scintilla qui est nécessaire à l'utilisation des commandes Scintilla.

Arguments

NomBibliotheque$ (optionnel) Permet d'indiquer le chemin et le nom du fichier de la dll ('scintilla.dll' par défaut). La dll scintilla est incluse dans le répertoire "Compilers" du dossier PureBasic.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon. Sous MacOs et Linux, la valeur de retour est toujours #True car la bibliothèque est liée statiquement.

Remarques

Cette commande n'est présente que sur Windows, car sur les autres OS, la bibliothèque Scintilla est liée statiquement avec l'exécutable et n'a donc pas besoin d'être chargée ou distribuée avec le programme.

OS Supportés

Windows

134.2 ScintillaGadget

Syntaxe

```
Resultat =
    ScintillaGadget(#Scintilla,
    X, Y, Largeur, Hauteur,
    @Callback())
```

Description

Crée un nouveau contrôle d'édition scintilla.

Arguments

#Scintilla Le numéro d'identification du nouveau gadget scintilla.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y, Largeur, Hauteur La position et les dimensions du nouveau gadget.

@Callback() L'adresse de la procédure qui recevra les évènements émis par le gadget. Elle doit être définie de la façon suivante, avec 'Gadget' qui est le numéro du contrôle et *scinotify qui pointe vers une structure comportant les informations sur l'évènement :

```
1
2   ProcedureDLL
     ScintillaCallBack(Gadget,
     *scinotify.SCNotification)
3    ;
4    ; Votre code ici.
5    ;
6   EndProcedure
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Les événements suivants sont pris en charge par EventType() :

```
#PB_EventType_RightClick
```

InitScintilla() doit être appelé avec succès avant d'utiliser cette commande.
Les commandes spécifiques de Scintilla peuvent être envoyées au gadget avec la commande ScintillaSendMessage() .
En outre les commandes de gadgets communes comme ResizeGadget() ou HideGadget() peuvent aussi bien être utilisées par le contrôle.

Exemple

```
1   If OpenWindow(0, 0, 0, 400,
     90, "ScintillaGadget",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2
3       If InitScintilla()
4          ScintillaGadget(0,
    10, 10, 390, 70, 0)
5
6          ; Texte en rouge
```

```
7           ScintillaSendMessage(0,
    #SCI_STYLESETFORE, 0,
    RGB(255, 0, 0))
8
9           ; Définit le texte
    initial du ScintillaGadget
10          *Texte=UTF8("Hé,
    voici un simple
    ScintillaGadget avec du
    texte...")
11          ScintillaSendMessage(0,
    #SCI_SETTEXT, 0, *Texte)
12          FreeMemory(*Texte)
    ; Le tampon créé par
    UTF8() doit être libéré
    pour éviter les fuites de
    mémoire.
13
14          ; Ajout d'une
    deuxième ligne de texte
    précédée d'un saut de ligne
15          Texte$ = Chr(10) +
    "Seconde ligne."
16          *Texte=UTF8(Texte$)
17          ScintillaSendMessage(0,
    #SCI_APPENDTEXT,
    Len(Texte$), *Texte)
18          FreeMemory(*Texte)
19      EndIf
20
21      Repeat : Until
    WaitWindowEvent() =
    #PB_Event_CloseWindow
22   EndIf
```

OS Supportés

Tous

134.3 ScintillaSendMessage

Syntaxe

```
Resultat =
    ScintillaSendMessage(#Scintilla,
    Message [, Param [,
    LParam]])
```

Description

Envoie un message au gadget scintilla pour exécuter une tâche spécifique.

Arguments

#**Scintilla** Le numéro d'identification du gadget scintilla.

Message Le message à envoyer.
C'est un nombre entier sous la forme d'une constante prédéfinie qui commence par '#SCI_'.
Vous trouverez plus d'informations sur les messages possibles sur le site Scintilla.
Les constantes #SCI_[...] représentant les valeurs possibles pour le 'Message' sont déjà définies dans PB.

Param (optionnel) Le premier paramètre du message.
S'il est omis alors la valeur par défaut est zéro.

LParam (optionnel) Le second paramètre du message.
S'il est omis alors la valeur par défaut est zéro.

Valeur de retour

Renvoie le résultat du message envoyé.

OS Supportés

Tous

Chapitre 135

Screen

Généralités

Un Screen (écran) est une surface utilisée pour afficher du contenu qui requiert une accélération matérielle comme les sprites , ou des objets et des mondes 3D". Un écran peut être créé soit dans une fenêtre standard, soit en plein écran.
Windows : Par défaut, c'est DirectX9 qui utilise l'accélération matérielle, si elle est disponible.
Sous Windows, une version récente de DirectX 9 doit être installée (voir ici : DirectX 9 runtime installer).
En fonction des besoins, deux autres sous-systèmes sont également disponibles. Il s'agit de "OpenGL" et de "DirectX11". À utiliser dans le menu "Compilateur\Options du Compilateur \Option de Compilation\Bibliothèque sous-système".
Linux : OpenGL est utilisé pour gérer l'écran ce qui permet d'utiliser l'accélération matérielle.
MacOS X : OpenGL est utilisé pour gérer l'écran ce qui permet d'utiliser l'accélération matérielle.

OS Supportés

Tous

135.1 ChangeGamma

Syntaxe

```
ChangeGamma(IntensiteRouge,
    IntensiteVert,
    IntensiteBleu)
```

Description

Change la composante Gamma de l'écran en cours.

Arguments

IntensiteRouge, IntensiteVert, IntensiteBleu
Les nouvelles intensités.
Les valeurs valides sont comprises entre 0 et 255.

Valeur de retour

Aucune.

Remarques

ATTENTION : Cela ne fonctionne qu'en mode plein écran (pas en mode fenêtré). L'intensité des canaux Rouge, Vert et Bleu peut être modifiée individuellement.
Cette fonction peut être utilisée pour faire des fondus enchaînés (fade in/fade out) en mode plein écran, des splashs de couleurs, etc. S'il ne fait rien, le matériel ne supporte pas cette fonction (Pas d'émulation prévue, en raison du nombre élevé d'opérations nécessaires à effectuer).

Exemple

```
1  ;Initialisation du système
     d'affichage.
2   InitSprite()
3
4   ; Ouverture d'un écran
     800*600 32 bits, noir par
     défaut
5   OpenScreen(800, 600, 32,
     "Sprite")
6
7   For i=0 To 255
8   ;Ecran jaune
9   ClearScreen(RGB(255,255,i))
10  ;Changement de la
     composante bleue et
     affichage du résultat dans
     le buffer
11  ChangeGamma(255, 255, i)
12  ;Inversion des buffers =
     affichage sur l'écran
13  FlipBuffers()
14  Next i
15
16 ;Attendre 1 seconde
17  Delay(1000)
18
19 ;Fermer l'écran
20  CloseScreen()
```

OS Supportés

Windows (DirectX)

135.2 ClearScreen

Syntaxe

```
ClearScreen(Couleur)
```

Description

Efface la totalité de l'écran avec la couleur spécifiée.

Arguments

Couleur La couleur qui remplira l'écran. RGB() peut être utilisé pour obtenir une valeur de couleur valide.
Un tableau avec les couleurs communes est disponible ici .

Valeur de retour

Aucune.

Remarques

ClearScreen() doit toujours être appelé en dehors d'un bloc StartDrawing() : StopDrawing() .

Exemple

```
1  ;Initialisation du système d'affichage.
2  InitSprite()
3
4  ;Ouverture d'un écran 800*600 32 bits, noir par défaut
5  OpenScreen(800, 600, 32, "Sprite")
6
7  ; Attendre 3 secondes
8  Delay(3000)
9
10 ; L'écran est noir pendant 3s
11
12 ; L'écran est rouge mais dans l'autre buffer
13 ClearScreen(RGB(255,0,0))
14
15 ; Il faut inverser les buffers pour afficher l'écran rouge
16 FlipBuffers()
17
18 ; Attendre 3 secondes
19 Delay(3000)
20 ; L'écran est rouge pendant 1s
```

```
21
22  ; Fermer l'écran
23   CloseScreen()
```

OS Supportés

Tous

135.3 CloseScreen

Syntaxe

```
CloseScreen()
```

Description

Ferme l'écran en cours.

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

L'écran est fermé qu'il soit en mode fenêtré ou en mode plein écran.
Après la fermeture d'un écran, tous les sprites doivent être rechargés car le format de l'écran a été perdu et sa mémoire vidéo libérée.
Une application ou un jeu peut passer du plein écran au mode fenêtré à la volée sans aucun problème.

Exemple

```
1   CloseScreen()
```

Voir aussi

OpenScreen() , OpenWindowedScreen()

OS Supportés

Tous

135.4 FlipBuffers

Syntaxe

```
FlipBuffers()
```

Description

Les deux tampons, celui de l'écran visible (Front) et celui de l'écran invisible (Back) sont interchangés. La zone invisible est désormais visible et vice versa, ce qui permet de faire un effet de "double-buffering" et d'éviter le scintillement (flickering).

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

La fonction FlipBuffers() doit toujours être appelée en dehors d'un bloc StartDrawing() : StopDrawing() .
Un écran doit avoir été ouvert avec OpenScreen() ou OpenWindowedScreen() .
La manière dont les tampons sont interchangés (avec ou sans synchronisation) est fixé par OpenScreen() ou OpenWindowedScreen() .

Exemple

```
1  ;Initialisation du système d'affichage.
2  InitSprite()
3
4  ;Ouverture d'un écran 800*600 32 bits, noir par défaut
5  OpenScreen(800, 600, 32, "Sprite")
6  ;Par défaut, les deux buffers sont remplis par un fond noir
7  ;Appelons-les : Buffer_NonAffiché et Buffer_Affiché
8  ;Buffer_Affiché = Fond noir / Buffer_NonAffiché = Fond noir
9
10 Delay(1000) ; Attendre 1 seconde
```

```
11  ;L'écran est noir pendant 1s
12
13  ;ATTENTION : Toutes les
      actions 2D se font sur le
      Buffer_NonAffiché
14  ClearScreen(RGB(255,0,0))
15  ;Buffer_Affiché = Fond noir
      / Buffer_NonAffiché = Fond
      rouge
16
17  FlipBuffers() ; Il faut
      inverser les buffers pour
      afficher l'écran rouge
18  ;Buffer_Affiché = Fond rouge
      / Buffer_NonAffiché = Fond
      noir
19
20  Delay(1000) ; Attendre 1
      seconde
21  ;L'écran est rouge pendant 1s
22
23  ClearScreen(RGB(0,255,0))
24  ;Buffer_Affiché = Fond rouge
      / Buffer_NonAffiché = Fond
      vert
25
26  FlipBuffers(); Il faut
      inverser les buffers pour
      afficher l'écran vert
27  ;Buffer_Affiché = Fond vert
      / Buffer_NonAffiché = Fond
      rouge
28
29  Delay(1000) ; Attendre 1
      seconde
30  ;L'écran est vert pendant 1s
31
32  ClearScreen(RGB(0,0,255))
33  ;Buffer_Affiché = Fond vert
      / Buffer_NonAffiché = Fond
      bleu
34
35  FlipBuffers(); Il faut
      inverser les buffers pour
      afficher l'écran bleu
36  ;Buffer_Affiché = Fond bleu
      / Buffer_NonAffiché = Fond
      vert
37
38  Delay(1000) ; Attendre 1
      seconde
39  ;L'écran est bleu pendant 1s
40
41  FlipBuffers(); l'écran est
      vert !!!
42  ;Buffer_Affiché = Fond vert
      / Buffer_NonAffiché = Fond
      bleu
43
```

```
44 Delay(1000) ; Attendre 1
       seconde
45  ;L'écran est vert pendant 1s
46
47 CloseScreen()
```

Voir aussi

OpenScreen() , OpenWindowedScreen()

OS Supportés

Tous

135.5 IsScreenActive

Syntaxe

```
Resultat = IsScreenActive()
```

Description

Les jeux et les applications en mode plein écran programmés en PureBasic fonctionnent toujours en mode multitâche, pour éviter de bloquer tout le système. En d'autre termes, l'utilisateur peut à tout moment revenir sur son bureau. Si c'est le cas, cette commande le détectera en renvoyant zéro. Le programmeur devra alors prendre les mesures nécessaires comme libérer la souris avec ReleaseMouse() , mettre le jeu en pause, arrêter les sons et les musiques, etc...

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle si l'écran est actif, zéro sinon.

Remarques

Cette commande doit être obligatoirement placée après FlipBuffers() .

Voir aussi

OpenScreen() , OpenWindowedScreen() , ReleaseMouse() , FlipBuffers()

OS Supportés

Tous

135.6 ScreenID

Syntaxe

```
Resultat = ScreenID()
```

Description

Renvoie l'identifiant système de l'écran.

Arguments

Aucun.

Valeur de retour

L'identifiant de l'écran.
Sous Windows, l'écran n'est autre qu'une fenêtre, donc n'importe quelle commande nécessitant un 'WindowID' peut être utilisée (comme PlayMovie()).

OS Supportés

Windows (DirectX), Linux (SDL)

135.7 ScreenWidth

Syntaxe

```
Resultat = ScreenWidth()
```

Description

Renvoie la largeur de l'écran courant, précédemment ouvert avec OpenScreen() ou OpenWindowedScreen() .

Arguments

Aucun.

Valeur de retour

Renvoie la largeur de l'écran en cours, ou zéro si aucun écran est ouvert.

Exemple

```
1 InitSprite()
2
3  ; Ouverture d'un écran
     800*600 32 bits, noir par
     défaut
4 OpenScreen(800, 600, 32,
     "Sprite")
5
6 StartDrawing(ScreenOutput())
```

```
W=ScreenWidth()
H=ScreenHeight()
D=ScreenDepth()
DrawText(10,50,"Largeur de l'écran : " + Str(W))
DrawText(10,100,"Hauteur de l'écran : " + Str(H))
DrawText(10,150,"Profondeur des couleurs : " + Str(D))
StopDrawing()

;Inversion des buffers = affichage sur l'écran
FlipBuffers()

;Attendre 4 secondes
Delay(4000)

;Fermer l'écran
CloseScreen()
```

Voir aussi

OpenScreen() , OpenWindowedScreen() , ScreenHeight() , ScreenDepth()

OS Supportés

Tous

135.8 ScreenHeight

Syntaxe

```
Resultat = ScreenHeight()
```

Description

Renvoie la hauteur de l'écran courant, précédemment ouvert avec OpenScreen() ou OpenWindowedScreen() .

Arguments

Aucun.

Valeur de retour

Renvoie la hauteur de l'écran en cours, ou zéro si aucun écran est ouvert.

Exemple

Voir l'exemple de la fonction ScreenWidth()

Voir aussi

OpenScreen() , OpenWindowedScreen() , ScreenWidth() , ScreenDepth()

OS Supportés

Tous

135.9 ScreenDepth

Syntaxe

```
Resultat = ScreenDepth()
```

Description

Renvoie la profondeur de couleur de l'écran courant, précédemment ouvert avec OpenScreen() ou OpenWindowedScreen() .

Arguments

Aucun.

Valeur de retour

Renvoie la profondeur de couleur (entre 8 et 32) de l'écran en cours, ou zéro si aucun écran est ouvert.

Exemple

Voir l'exemple de la fonction ScreenWidth()

Voir aussi

OpenScreen() , OpenWindowedScreen() , ScreenWidth() , ScreenHeight()

OS Supportés

Tous

135.10 SetFrameRate

Syntaxe

```
SetFrameRate(ImagesParSeconde)
```

Description

Définit le nombre d'images par seconde affiché dans l'écran en cours.

Arguments

ImagesParSeconde Le nouveau taux d'affichage d'images.

Valeur de retour

Aucune.

Remarques

C'est particulièrement utile en mode fenêtré.
Cette fonction définit le nombre maximal de fois par seconde que la fonction FlipBuffers() est appelée.

Voir aussi

OpenScreen() , OpenWindowedScreen() , FlipBuffers()

OS Supportés

Tous

135.11 OpenScreen

Syntaxe

```
Resultat =
    OpenScreen(Largeur,
    Hauteur, Profondeur,
    Titre$ [, FlipMode [,
    TauxDeRafraichissement])
```

Description

Ouvre un nouvel écran.

Arguments

Largeur, Hauteur La résolution de l'écran, en pixels.
Attention, la résolution voulue doit pouvoir être supportée par le matériel. La fonction ExamineScreenModes() permet d'obtenir la liste des résolutions supportées par le matériel.

Profondeur Peut prendre l'une des valeurs suivantes :

```
16: 65536 couleurs,
 palette fixe
24: 16777216 couleurs,
 palette fixe
32: 16777216 couleurs,
 plus rapide que le mode
 24-bit, la transparence
 (alphablending) est
 autorisée.
```

Titre$ Le titre de l'application.
Sous Windows seulement. Le paramètre 'Titre$' sera affiché dans la barre des tâches lors d'un retour vers le bureau. Il est conseillé de choisir un nom approprié à l'application.

FlipMode (optionnel) Définit le mode de synchronisation utilisé avant l'échange des buffers.
(Vertical blank synchronization = Synchro verticale)').
Peut prendre l'une des valeurs suivantes :

```
#PB_Screen_NoSynchronization
  : Désactive la
synchronisation
#PB_Screen_WaitSynchronization
: Active la
synchronisation (mode
par défaut)
#PB_Screen_SmartSynchronization:
Active la
synchronisation en
essayant de préserver le
temps processeur,lorsque
le jeu
                                    est
en mode plein écran.
```

Attendre la synchronisation permet d'avoir des inversions de buffers parfaites, sans 'tearing' (ancienne et nouvelle image à moitié visible) ou autres effets parasites car l'inversion est effectuée lorsque que le faisceau lumineux est en dehors de la partie visible de l'écran. De plus, l'inversion se fait à un rythme régulier, en relation avec sa fréquence de rafraîchissement : par exemple, pour un écran en 60 Hz, l'inversion pourra se faire au maximum 60 fois par seconde (donc on aura un rythme régulier de 60 images/secondes).

TauxDeRafraichissement (optionnel)
Définit le taux de rafraîchissement (en Hz).
Si l'écran ne supporte pas ce taux de rafraîchissement, alors OpenScreen() échouera.
La fonction ExamineScreenModes() permet d'obtenir la liste des taux de rafraîchissement supportés par le matériel.
Note : Sous Windows, le taux de rafraîchissement peut être bloqué par le système ou par les pilotes vidéo. Le résultat de cette fonction n'est donc pas forcément fiable.

Valeur de retour

Renvoie une valeur non nulle si l'écran a été créé avec succès, zéro sinon.

Remarques

L'écran est créé avec deux buffers vidéo, ce qui permet l'affichage successif des deux buffers avec la fonction FlipBuffers() , particulièrement utile pour les jeux.
Pour ouvrir un écran à l'intérieur d'une fenêtre, il convient d'utiliser la commande OpenWindowedScreen() .
Il n'est pas possible d'utiliser la bibliothèque Requester dans un écran.

Exemple

```
; ATTENTION le retour vers le bureau (ALT+TAB) n'est pas géré dans cet exemple.

; Initialisation des sprites, du clavier et de la souris
If InitSprite() = 0 Or InitKeyboard() = 0 Or InitMouse() = 0
  MessageRequester("Erreur", "Impossible d'initialiser l'écran.")
  End
EndIf

; Ouverture de l'écran
If OpenScreen(800,600,32,"Exemple OpenScreen") = 0
  MessageRequester("Erreur", "Impossible d'ouvrir l'écran.")
  End
EndIf

; Curseur de la souris
CreateSprite(0,20,20,#PB_Sprite_PixelCollision)
StartDrawing(SpriteOutput(0))
Box(0, 0, 20, 20, RGB(255, 255, 255))
StopDrawing()

; Afficheur "Survoler pour quitter"
CreateSprite(1,200,100,#PB_Sprite_PixelCollision)
StartDrawing(SpriteOutput(1))
Box(0, 0, 200, 100, RGB(255, 255, 0))
DrawText(10,25,"Survoler pour quitter",RGB(255,255,255),
```

```
    RGB(255,0,0))
StopDrawing()

 ;Boucle principale
Repeat

  ;Effacer complètement
   l'écran et afficher un
   fond gris
  ClearScreen(RGB(128,128,128))

  ;On lit les évènements
   clavier et souris
  ExamineMouse()
  ExamineKeyboard()

  ;Position de la souris
  x = MouseX()
  y = MouseY()

  ;Affichage du curseur de la
   souris en temps réel
  DisplaySprite(0, X, Y)
  ;Affichage de l'afficheur
   "Survoler pour quitter"
  DisplaySprite(1, 500, 50)

  ;Détection de collision
   entre le curseur de la
   souris et l'afficheur
  If SpriteCollision(0, X, Y,
   1, 500, 50) <> 0
    End ; Si collision alors
   on ferme le programme
  EndIf

  ;Calcul du FPS  (image par
   seconde)
  FPS_Counter + 1
  If FPS_Counter >= FPS
    FPS = FPS_Counter * 1000
   / (ElapsedMilliseconds() -
   FPS_ElapsedTime)
    FPS_Counter = 0
    FPS_ElapsedTime =
   ElapsedMilliseconds()
  EndIf

  ;Affichage du FPS et du
   texte
  StartDrawing(ScreenOutput())
  DrawText(0, 0, "FPS : " +
   StrD(FPS, 1))
  DrawText(0, 75, "Appuyez
   sur ESC pour
   quitter",RGB(0,0,0),
   RGB(255,255,255))
  DrawText(0, 550, "Appuyez
   sur les touches fléchées
```

```
    du clavier tout en
    bougeant la
    souris",RGB(255,0,0),
    RGB(0,255,0))
  StopDrawing()

  ;Gestion des 4 touches
    fléchées du clavier
  If
    KeyboardPushed(#PB_Key_Right)
     RotateSprite(0,45,#PB_Relative)
  EndIf
  If
    KeyboardPushed(#PB_Key_Left)
     RotateSprite(0,1,#PB_Relative)
  EndIf
  If
    KeyboardPushed(#PB_Key_Up)
     ZoomSprite(0,100,100)
  EndIf
  If
    KeyboardPushed(#PB_Key_Down)
     ZoomSprite(0,20,20)
  EndIf

  ;Maintenant que tout est
    calculé et affiché dans le
    buffer invisible,
  ;on inverse les buffers
    pour rendre la scène
    visible à l'écran.
  FlipBuffers()

Until
    KeyboardPushed(#PB_Key_Escape)
    ;On quitte l'application
    en  appuyant sur la touche
    Echap (ESC)

; Idée   : Remplacer la
    ligne de code : If
    OpenScreen(800,600,32,"Exemple
    OpenScreen") = 0
; par    : If
    OpenScreen(800,600,32,"Exemple
    OpenScreen",
    #PB_Screen_NoSynchronization)
    = 0
; ou par : If
    OpenScreen(800,600,32,"Exemple
    OpenScreen",
    #PB_Screen_SmartSynchronization)
    = 0
; et observez le FPS ainsi
    que la qualité de l'image
    (bouger la souris)
```

Voir aussi

OpenWindowedScreen() , FlipBuffers()

OS Supportés

Tous

135.12 OpenWindowedScreen

Syntaxe

```
Resultat =
    OpenWindowedScreen(FenetreID,
    X, Y, Largeur, Hauteur [,
    RedimensionnementAuto,
    OffsetDroit, OffsetBas [,
    FlipMode]])
```

Description

Ouvre un écran dans une fenêtre standard.

Arguments

FenetreID La fenêtre qui contient l'écran. WindowID() peut être utilisé pour obtenir un identifiant valide.

X, Y La position de l'écran, en pixels, dans la fenêtre.

Largeur, Hauteur Les dimensions de l'écran, en pixels.

RedimensionnementAuto (optionnel)

```
      #True  : L'écran sera
    redimensionné et sont
    contenu sera zoomé quand
    les dimensions de la
    fenêtre changeront.
   #False : Pas de
    redimensionnement de
    l'écran (par défaut).
```

OffsetDroit, OffsetBas (optionnel) Définissent les marges droite et basse dans la fenêtre, en pixels. (Pour prendre en compte la barre d'état (statusbar) par exemple).

FlipMode (optionnel) Mode de synchronisation avant d'interchanger les buffers (Synchro Vertical).
Peut être une des valeurs suivantes :

```
    #PB_Screen_NoSynchronization
      : désactive la
    synchronisation
    #PB_Screen_WaitSynchronization
    : active la
    synchronisation (mode
    par défaut)
```

```
#PB_Screen_SmartSynchronization:
active la
synchronisation en
essayant de préserver le
temps processeur,
lorsque le jeu
                                    est
en mode plein écran.
```

Attendre la synchronisation permet d'avoir des inversions de buffers parfaites, sans 'tearing' (ancienne et nouvelle image à moitié visible) ou autres effets parasites car l'inversion est effectuée lorsque que le faisceau lumineux est en dehors de la partie visible de l'écran. De plus, l'inversion se fait à un rythme régulier, en relation avec sa fréquence de rafraîchissement : par exemple, pour un écran en 60 Hz, l'inversion pourra se faire au maximum 60 fois par seconde (donc on aura un rythme régulier de 60 images/secondes).

Valeur de retour

Renvoie une valeur non nulle si l'écran a été créé avec succès, zéro sinon.

Remarques

Il n'est pas possible de créer plus d'un écran fenêtré à la fois.
Les dimensions de l'écran ne peuvent être supérieures à celles de la fenêtre sinon des artéfacts apparaitront.
L'écran fenêtré emploie l'accélération matérielle de la même manière que le mode plein écran utilisant la commande OpenScreen() .
Comme une fenêtre est ouverte, les évènements de la fenêtre doivent être traitées avec WindowEvent() pour avoir un comportement correct. Tous les évènements doivent être traités avant de 'flipper' (interchanger) les tampons (voir les exemples ci-dessous et FlipBuffers()).
L'écran est créé avec deux buffers vidéo, ce qui permet l'affichage successif des deux buffers avec la fonction FlipBuffers() , particulièrement utile pour les jeux.

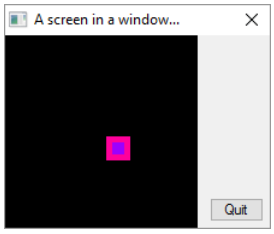
Exemple

Example 1 : Ecran fenêtré de taille fixe avec gadgets

```
1  If InitSprite() = 0
```

```
    MessageRequester("Erreur",
  "Impossible d'ouvrir
  l'écran & l'environnement
  nécessaire aux sprites !",
  0)
    End
  EndIf

  If OpenWindow(0, 0, 0, 220,
  160, "Un écran dans une
  fenêtre...",
  #PB_Window_SystemMenu |
  #PB_Window_ScreenCentered)
    ButtonGadget(0, 170, 135,
  45, 20, "Quitter")

    If
  OpenWindowedScreen(WindowID(0),
  0, 0, 160, 160)
      CreateSprite(0, 20, 20)
      If
  StartDrawing(SpriteOutput(0))
        Box(0, 0, 20, 20,
  RGB(255, 0, 155))
        Box(5, 5, 10, 10,
  RGB(155, 0, 255))
        StopDrawing()
      EndIf
    Else
      MessageRequester("Erreur",
  "Impossible d'ouvrir un
  écran dans la fenêtre!", 0)
      End
    EndIf
  EndIf

  direction = 2
  Repeat
    ; Il est très important
  de traiter tous les
  évènements restants dans
  la file d'attente à chaque
  tour
    ;
    Repeat
      Event = WindowEvent()

      Select Event
        Case #PB_Event_Gadget
          If EventGadget() = 0
            End
          EndIf

        Case
  #PB_Event_CloseWindow
          End
      EndSelect
    Until Event = 0

```

```
    FlipBuffers()
    ClearScreen(RGB(0, 0, 0))
    DisplaySprite(0, x, x)
    x + direction
    If x > 140 : direction =
   -2 : EndIf
    If x < 0   : direction =
   2 : EndIf
    Delay(1)
  ForEver
```

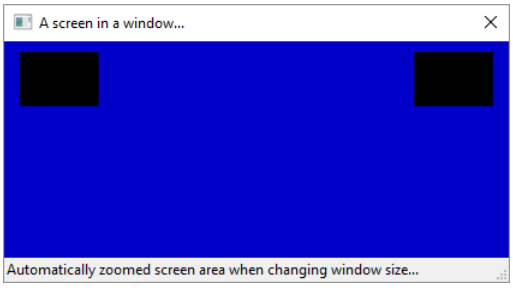


Exemple

Example 2 : Avec redimensionnement automatique et marge du bas

```
1  If InitSprite() = 0
2    MessageRequester("Erreur",
     "Impossible d'ouvrir
     l'écran & l'environnement
     nécessaire aux sprites !",
     0)
3    End
4  EndIf
5
6  If OpenWindow(0, 0, 0, 420,
     200, "Un écran dans une
     fenêtre...",
     #PB_Window_SystemMenu |
     #PB_Window_SizeGadget |
     #PB_Window_ScreenCentered)
7    CreateStatusBar(0,
     WindowID(0))
8      AddStatusBarField(420)
9
10   StatusBarText(0, 0, "Zoom
     automatique lors du
     changement de taille de la
     fenêtre...")
11
12 If
     OpenWindowedScreen(WindowID(0),
     0, 0,320, 200, #True, 0,
     20)
13
14   CreateSprite(0, 50, 50) ;
     Crée un sprite vide qui
     sera tout noir
15
16 Repeat
```

```
17      ; Il est très important de traiter tous les évènements restants dans la file d'attente à chaque tour
18      ;
19      Repeat
20        Event = WaitWindowEvent(10)
21
22        If Event = #PB_Event_CloseWindow
23          End
24        EndIf
25      Until Event = 0
26
27      FlipBuffers()
28      ClearScreen(RGB(0, 0, 200)) ; Un fond bleu
29
30      DisplaySprite(0, 10, 10) ; Affichez notre boîte noire dans le coin en haut à gauche
31      DisplaySprite(0, 260, 10) ; Affichez notre boîte noire dans le coin en haut à droite
32    ForEver
33
34      Else
35        MessageRequester("Erreur", "Impossible d'ouvrir l'écran fenêtré !", 0)
36      EndIf
37    EndIf
```



Pour plus de détail, voir cet exemple

Voir aussi

OpenScreen() , FlipBuffers()

OS Supportés

Tous

135.13 ScreenOutput

Syntaxe

```
Resultat = ScreenOutput()
```

Description

Renvoie l'identifiant de l'écran courant nécessaire aux opérations 2D.

Arguments

Aucun.

Valeur de retour

L'identifiant de l'écran en cours.

Remarques

Ne peut être utilisé que dans un bloc StartDrawing() / StopDrawing() de la bibliothèque 2D, 2DDrawing .
La mémoire allouée à ScreenOutput() est libérée avec StopDrawing().
Sous Linux et OS X, ScreenOutput() copie la totalité de l'écran tampon dans la mémoire principale pour effectuer des opérations de dessin 2D (OpenGL ne permet pas d'accéder directement au tampon). Par conséquent dessiner sur un écran peut être très lent et devrait être évité.
ScreenOutput() doit être appelé dans le même thread où OpenScreen() a été appelé.

Exemple

```
1  StartDrawing(ScreenOutput())
2    Box(0, 0, 200, 50,
     RGB(255, 255, 255))
3  StopDrawing()
```

OS Supportés

Tous

135.14 ExamineScreenModes

Syntaxe

```
Resultat =
    ExamineScreenModes()
```

Description

Démarre l'examen des modes d'écrans disponibles sur l'ordinateur.
La liste des modes d'écran peut être énumérée à l'aide de la commande NextScreenMode() .

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle si l'énumération des modes d'affichage est un succès, zéro sinon.

Exemple

```
1  InitSprite()
2
3  If ExamineScreenModes()
4    While NextScreenMode()
5      Debug
     Str(ScreenModeWidth())+"x"+Str(ScreenModeHeight())+"x"+Str(S
6    Wend
7  EndIf
```

Voir aussi

NextScreenMode()

OS Supportés

Tous

135.15 NextScreenMode

Syntaxe

```
Resultat = NextScreenMode()
```

Description

Cette commande doit être appelée après ExamineScreenModes() . Elle permet de lister les modes d'écran disponible un par un.
Les informations disponibles sur le mode d'écran en cours d'examen peuvent être récupérées à l'aide des commandes suivantes : ScreenModeWidth() , ScreenModeHeight() , ScreenModeDepth() et ScreenModeRefreshRate() .

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle s'il existe un mode suivant ou zéro s'il ne reste plus de mode à lister.

Exemple

```
1  InitSprite()
2
3  If ExamineScreenModes()
4    While NextScreenMode()
5      Debug
     Str(ScreenModeWidth())+"x"+Str(ScreenModeHeight())+"x"+Str(S
6    Wend
7  EndIf
```

Voir aussi

ExamineScreenModes() , ScreenModeWidth() , ScreenModeHeight() , ScreenModeDepth() , ScreenModeRefreshRate()

OS Supportés

Tous

135.16 ScreenModeDepth

Syntaxe

```
Resultat = ScreenModeDepth()
```

Description

Renvoie la profondeur de couleur (en bits) du mode d'écran actuellement listé par ExamineScreenModes() et NextScreenMode() .

Arguments

Aucun.

Valeur de retour

La profondeur peut être une des valeurs suivantes en fonction de la carte graphique : 8, 15, 16, 24 ou 32 bits.

Exemple

```
InitSprite()

If ExamineScreenModes()
  While NextScreenMode()
    Debug
   Str(ScreenModeWidth())+"x"+Str(ScreenModeHeight())+"x"+Str(S
  Wend
EndIf
```

Voir aussi

ExamineScreenModes() , NextScreenMode() , ScreenModeWidth() , ScreenModeHeight() , ScreenModeRefreshRate()

OS Supportés

Tous

135.17 ScreenModeHeight

Syntaxe

```
Resultat = ScreenModeHeight()
```

Description

Renvoie la hauteur (en pixels) du mode d'écran actuellement listé par ExamineScreenModes() et NextScreenMode() .

Arguments

Aucun.

Valeur de retour

Renvoie la hauteur (en pixels) du mode d'écran.

Exemple

```
InitSprite()

If ExamineScreenModes()
  While NextScreenMode()
    Debug
   Str(ScreenModeWidth())+"x"+Str(ScreenModeHeight())+"x"+Str(S
  Wend
EndIf
```

Voir aussi

ExamineScreenModes() , NextScreenMode() , ScreenModeWidth() , ScreenModeDepth() , ScreenModeRefreshRate()

OS Supportés

Tous

135.18 ScreenModeRefreshRate

Syntaxe

```
Resultat =
    ScreenModeRefreshRate()
```

Description

Renvoie le taux de rafraîchissement (en hertz) du mode d'écran actuellement listé par ExamineScreenModes() et NextScreenMode() .

Arguments

Aucun.

Valeur de retour

Renvoie le taux de rafraîchissement (en hertz) du mode d'écran.

Remarques

Sous OS X de nombreux portables et moniteurs ne supportent pas cette commande, qui renvoie souvent 0.

Exemple

```
1  InitSprite()
2
3  If ExamineScreenModes()
4    While NextScreenMode()
5      Debug
     Str(ScreenModeWidth())+"x"+Str(ScreenModeHeight())+"x"+Str(S
6    Wend
7  EndIf
```

Voir aussi

ExamineScreenModes() , NextScreenMode() , ScreenModeWidth() , ScreenModeHeight() , ScreenModeDepth()

OS Supportés

Tous

135.19 ScreenModeWidth

Syntaxe

```
Resultat = ScreenModeWidth()
```

Description

Renvoie la largeur (en pixels) du mode d'écran actuellement listé par ExamineScreenModes() et NextScreenMode() .

Arguments

Aucun.

Valeur de retour

Renvoie la largeur (en pixels) du mode d'écran.

Exemple

```
1  InitSprite()
2
3  If ExamineScreenModes()
4    While NextScreenMode()
5      Debug
     Str(ScreenModeWidth())+"x"+Str(ScreenModeHeight())+"x"+Str(S
6    Wend
7  EndIf
```

Voir aussi

ExamineScreenModes() , NextScreenMode() , ScreenModeHeight() , ScreenModeDepth() , ScreenModeRefreshRate()

OS Supportés

Tous

Chapitre 136

SerialPort

Généralités

Le port série aussi connu sous le nom de port RS-232 est toujours largement utilisé dans l'industrie, et ce en dépit de son âge : il a été créé en 1969. Sa simplicité permet de réaliser facilement des prototypes contrôlés par ordinateur. Cette bibliothèque offre un accès complet aux ports série disponibles. Il est conseillé de connaître la définition des termes relatifs aux ports série, pour plus d'informations : Wikipedia - RS232.

OS Supportés

Tous

136.1 AvailableSerialPortInput

Syntaxe

```
Resultat =
    AvailableSerialPortInput(#PortSerie)
```

Description

Renvoie le nombre d'octets restants dans le tampon d'entrée.

Arguments

#PortSerie Le port série à utiliser.

Voir aussi

AvailableSerialPortOutput()

OS Supportés

Tous

136.2 AvailableSerialPortOutput

Syntaxe

```
Resultat =
    AvailableSerialPortOutput(#PortSerie)
```

Description

Renvoie le nombre d'octets restants dans le tampon de sortie.

Arguments

#**PortSerie** Le port série à utiliser.

Valeur de retour

Renvoie le nombre d'octets restants dans le tampon de sortie.

Voir aussi

AvailableSerialPortInput()

OS Supportés

Tous

136.3 CloseSerialPort

Syntaxe

```
CloseSerialPort(#PortSerie)
```

Description

Ferme un port série.

Arguments

#**PortSerie** Le port série à utiliser.
Si #PB_All est spécifié, tous les ports série sont libérés.

Valeur de retour

Aucune.

Remarques

Tous les ports série ouverts sont fermés automatiquement quand le programme se termine.

Voir aussi

OpenSerialPort()

OS Supportés

Tous

136.4 GetSerialPortStatus

Syntaxe

```
Resultat =
    GetSerialPortStatus(#PortSerie,
    Attribut)
```

Description

Renvoie le statut d'un port série.

Arguments

#**PortSerie** Le port série à utiliser.

Attribut Peut prendre l'une des valeurs suivantes :

```
#PB_SerialPort_RI :
 Renvoie le statut du
 signal RI (0 ou 1)
#PB_SerialPort_DCD:
 Renvoie le statut du
 signal DCD (0 ou 1)
#PB_SerialPort_DSR:
 Renvoie le statut du
 signal DSR (0 ou 1)
#PB_SerialPort_CTS:
 Renvoie le statut du
 signal CTS (0 ou 1)
#PB_SerialPort_XonCharacter
 : Caractère 'Xon'
 utilisé pour le mode
 Xon/Xoff (entre 1 et 255)
#PB_SerialPort_XoffCharacter:
 Caractère 'Xoff' utilisé
 pour le mode Xon/Xoff
 (entre 1 et 255)
```

Valeur de retour

Renvoie le statut du port série en fonction de l'attribut demandé.

```
0 ou 1 pour l'attribut
 #PB_SerialPort_RI
0 ou 1 pour l'attribut
 #PB_SerialPort_DCD
0 ou 1 pour l'attribut
 #PB_SerialPort_DSR
0 ou 1 pour l'attribut
 #PB_SerialPort_CTS
Entre 1 et 255 pour
 l'attribut
 #PB_SerialPort_XonCharacter.
```

```
Entre 1 et 255 pour
 l'attribut
 #PB_SerialPort_XoffCharacter.
```

OS Supportés

Tous

136.5 IsSerialPort

Syntaxe

```
Resultat =
    IsSerialPort(#PortSerie)
```

Description

Teste si un port série est correctement initialisé.

Arguments

#**PortSerie** Le port série à utiliser.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Cette fonction a été conçue pour accepter n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.

Voir aussi

OpenSerialPort()

OS Supportés

Tous

136.6 SerialPortError

Syntaxe

```
Resultat =
    SerialPortError(#PortSerie)
```

Description

Renvoie l'erreur qui est survenue après ReadSerialPortData() , WriteSerialPortData() ou WriteSerialPortString() .

Arguments

#**PortSerie** Le port série à utiliser.

Valeur de retour

Peut être une combinaison des valeurs suivantes :

```
#PB_SerialPort_RxOver   :
 Un dépassement est survenu
 dans le tampon d'entrée
 (Il n'y a plus de
                          place
 dans le tampon d'entrée ou
 un caractère a été reçu
 apres un 'fin-de-fichier'
 (EOF)).
#PB_SerialPort_OverRun  :
 Un dépassement est survenu
 dans le tampon des
 caractères. Le prochain
 caractère est perdu.
#PB_SerialPort_RxParity :
 Le système a détecté une
 erreur de parité.
#PB_SerialPort_Frame    :
 Le système a détecté une
 erreur d'encapsulation des
 données.
#PB_SerialPort_Break    :
 Le système a détecté une
 condition d'arrêt.
#PB_SerialPort_TxFull   :
 L'application a essayé
 d'envoyer un caractère
 mais le tampon de sortie
 est plein.
#PB_SerialPort_IOE      :
 Une erreur d'Entrée/Sortie
 est survenue.
#PB_SerialPort_WaitingCTS
 : Indique que le système
 attend le signal CTS
 (clear-to-send) avant
 d'émettre.
#PB_SerialPort_WaitingDSR
 : Indique que le système
 attend le signal DSR
 (data-set-ready) avant
 d'émettre.
#PB_SerialPort_WaitingRLSD
 : Indique que le système
 attend le signal RLSD
 (receive-line-signal-detect)
 avant d'émettre.
#PB_SerialPort_XoffReceived:
 Indique que le système est
 en attente car le
 caractère XOFF a été reçu.
```

```
#PB_SerialPort_XoffSent
 : Indique que le système
 est en attente car le
 caractère XOFF a été
 transmis.
                              La
 transmission s'arrête
 quand le caractère XOFF
 est émis à un système qui
 attend le caractère XON.
#PB_SerialPort_EOFSent  :
 Indique que le caractère
 'fin-de-fichier' (EOF) a
 été reçu.
```

Voir aussi

ReadSerialPortData() ,
WriteSerialPortData() ,
WriteSerialPortString()

OS Supportés

Tous

136.7 SerialPortID

Syntaxe

```
SerialPortID =
   SerialPortID(#PortSerie)
```

Description

Renvoie l'identifiant système unique d'un port série.

Arguments

#**PortSerie** Le port série à utiliser.

Valeur de retour

Renvoie l'identifiant système unique du port série.

Remarques

Cette commande est utile quand une autre bibliothèque ou une commande de l'API a besoin d'accéder au port série.

Voir aussi

OpenSerialPort()

OS Supportés

Tous

136.8 OpenSerialPort

Syntaxe

```
Resultat =
    OpenSerialPort(#PortSerie,
    NomPortSerie$, Bauds,
    Parite, Data, Stop.f,
    HandshakeMode,
    TailleTamponEntree,
    TailleTamponSortie)
```

Description

Ouvre un port série.

Arguments

#PortSerie Le port série à utiliser. #PB_Any peut être utilisé pour générer automatiquement ce numéro.

NomPortSerie$ L'identifiant du port série, par exemple "COM1" sous Windows ou "/dev/ttyS0" sous Linux.

Bauds La vitesse de la connexion. Peut prendre une des valeurs suivantes :

```
 50, 75, 110, 150, 300,
600, 1200, 1800, 2400,
4800, 9600, 19200,
38400, 57600 ou 115200
```

Parite Le bit de parité utilisé. Peut être une des valeurs suivantes :

```
#PB_SerialPort_NoParity
 : Aucune parité
#PB_SerialPort_EvenParity
 : Pair
#PB_SerialPort_OddParity
 : Impair
#PB_SerialPort_MarkParity
 : Quand le bit de parité
 est présent mais pas
 utilisé et toujours posé
 à 1
#PB_SerialPort_SpaceParity:
 Quand le bit de parité
 est présent mais pas
 utilisé et toujours posé
 à 0
```

Data La longueur de chaque donnée en bits :

```
5 : Code Baudot (très
 ancien)
6 : Rare
7 : ASCII
8 : Octet
9 : Rare
```

Stop Le nombre de bits d'arrêts :

```
1, 1.5 ou 2
```

HandshakeMode Peut prendre une des valeurs suivantes :

```
#PB_SerialPort_NoHandshake
      : Pas de protocole
 d'initialisation
#PB_SerialPort_RtsHandshake
     : Pas de protocole
 d'initialisation mais
 RTS est mis à 1
#PB_SerialPort_RtsCtsHandshake
 : Protocole RTS/CTS
#PB_SerialPort_XonXoffHandshake:
 Protocole Xon/Xoff
```

TailleTamponEntree La taille du tampon d'entrée, en octets.

TailleTamponSortie La taille du tampon de sortie, en octets.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si le port série ne peut pas être ouvert alors il peut être déjà en cours d'utilisation, ou les paramètres ne sont pas corrects.

Exemple

```
1  If OpenSerialPort(0,
     "COM1", 300,
     #PB_SerialPort_NoParity,
     8, 1,
     #PB_SerialPort_NoHandshake,
     1024, 1024)
2    Debug "Succes"
3  Else
4    Debug "Echec"
5  EndIf
```

Voir aussi

CloseSerialPort()

OS Supportés

Tous

136.9 ReadSerialPortData

Syntaxe

```
Resultat =
    ReadSerialPortData(#PortSerie,
    *Memoire, Taille)
```

Description

Lit des données et les écrit en mémoire.

Arguments

#PortSerie Le port série à utiliser.

***Memoire** L'adresse mémoire où seront stockées les données reçues du port série.

Taille Le nombre d'octets à lire.

Valeur de retour

Le nombre d'octets réellement lu ou zéro si une erreur est apparue.
Ce nombre peut être inférieur au nombre demandé dans 'Taille'

Remarques

Cette fonction bloquera l'éxécution du programme jusqu'à l'arrivée des données. Pour vérifier que des données sont disponibles dans le port série, utiliser AvailableSerialPortInput() .

Voir aussi

AvailableSerialPortInput()

OS Supportés

Tous

136.10 SerialPortTimeouts

Syntaxe

```
SerialPortTimeouts(#PortSerie,
    RIT, RTTC, RTTM, WTTC,
    WTTM)
```

Description

Change les temps d'attentes par défaut.

Arguments

#**PortSerie** Le port série à utiliser.

RIT 'RIT' (Read Interval Timeout) : Indique le temps d'attente maximum, en millisecondes, entre l'arrivée de deux caractères sur la ligne de communication. La valeur par défaut est 100.

RTTC 'RTTC' (Read Total Timeout Constant) : Indique la constante, en millisecondes, utilisée pour calculer le temps d'attente maximum pour les opérations de lecture. Pour chaque opération de lecture, cette valeur est ajoutée au produit de 'RTTM' (Read Total Timeout Multiplier) et le nombre d'octets demandés.
La valeur par défaut est 100.

RTTM 'RTTM' (Read Total Timeout Multiplier) : Détermine le multiplicateur, en millisecondes, utilisé pour calculer le temps d'attente total pour les opérations de lecture. Pour chaque opération de lecture, cette valeur est multipliée par le nombre d'octets demandés.
La valeur par défaut est 10.

WTTC 'WTTC' (WriteTotalTimeoutConstant) : Indique la constante, en millisecondes, utilisée pour calculer le temps d'attente maximum pour les opérations d'écriture. Pour chaque opération d'écriture, cette valeur est ajoutée au produit de 'WTTM' (Write Total Timeout Multiplier) et le nombre d'octets devant être écrits.
La valeur par défaut est 10.

WTTM 'WTTM' (Write Total Timeout Multiplier) : Détermine le multiplicateur, en millisecondes, utilisé pour calculer le temps d'attente total pour les opérations d'écriture. Pour chaque opération d'écriture, cette valeur est multipliée par le nombre d'octets devant être écrits.
La valeur par défaut est 100.

Valeur de retour

Aucune.

Voir aussi

OpenSerialPort()

OS Supportés

Tous

136.11 SetSerialPortStatus

Syntaxe

```
SetSerialPortStatus(#PortSerie,
    Attribut, Valeur)
```

Description

Change le statut d'un port série.

Arguments

#PortSerie Le port série à utiliser.

Attribut Peut être l'une des valeurs suivantes :

```
#PB_SerialPort_DTR: Change
 le statut du signal DTR
 (0 ou 1)
#PB_SerialPort_RTS: Change
 le statut du signal RTS
 (0 ou 1)
#PB_SerialPort_TXD: Change
 le statut du signal TXD
 (0 ou 1)
#PB_SerialPort_XonCharacter
 : Caractère 'Xon'
 utilisé pour le mode
 Xon/Xoff (entre 1 et
 255). La valeur par
 défaut est $11.
#PB_SerialPort_XoffCharacter:
 Caractère 'Xoff' utilisé
 pour le mode Xon/Xoff
 (entre 1 et 255). La
 valeur par défaut est
 $13.
```

Valeur La valeur de l'attribut.

```
0 ou 1 pour l'attribut
 #PB_SerialPort_DTR
0 ou 1 pour l'attribut
 #PB_SerialPort_RTS
0 ou 1 pour l'attribut
 #PB_SerialPort_TXD
entre 1 et 255 pour
 l'attribut
 #PB_SerialPort_XonCharacter.
 La valeur par défaut est
 $11.
entre 1 et 255 pour
 l'attribut
 #PB_SerialPort_XoffCharacter.
 La valeur par défaut est
 $13.
```

Valeur de retour

Aucune.

Exemple

```
If OpenSerialPort(0, "COM1", 300, #PB_SerialPort_NoParity, 8, 1, #PB_SerialPort_XonXoffHandshake, 1024, 1024)
  SetSerialPortStatus(0, #PB_SerialPort_XonCharacter, 8)
  SetSerialPortStatus(0, #PB_SerialPort_XoffCharacter, 9)
  Debug "Succès"
Else
  Debug "Echec"
EndIf
```

Voir aussi

OpenSerialPort()

OS Supportés

Tous

136.12 WriteSerialPortData

Syntaxe

```
Resultat = WriteSerialPortData(#PortSerie, *Memoire, Taille)
```

Description

Envoie des données.

Arguments

#PortSerie Le port série à utiliser.

***Memoire** Le tampon (zone mémoire) où les données seront stockées.

Taille La taille du tampon.

Valeur de retour

Renvoie le nombre d'octets écrits sur le port série, ou zéro si l'opération a échoué.

Remarques

Pour connaître le nombre d'octets disponibles dans le tampon de sortie, utiliser AvailableSerialPortOutput() .

Voir aussi

OpenSerialPort() , WriteSerialPortString()

OS Supportés

Tous

136.13 WriteSerialPortString

Syntaxe

```
Resultat =
    WriteSerialPortString(#PortSerie,
    Texte$ [, Format])
```

Description

Envoie une chaîne de caractère.

Arguments

#PortSerie Le port série à utiliser.

Texte$ Le texte à écrire.

Format (optionnel) Peut être l'une des valeurs suivantes :

```
#PB_Ascii   : La chaîne
 sera écrite dans le
 format ASCII.
#PB_UTF8    : La chaîne
 sera écrite dans le
 format UTF8 (par défaut).
#PB_Unicode : La chaîne
 sera écrite dans le
 format unicode (UTF16).
```

Valeur de retour

Renvoie le nombre d'octets écrits sur le port série, ou zéro si l'opération a échoué.

Remarques

Pour connaître le nombre d'octets disponibles dans le tampon de sortie, utiliser AvailableSerialPortOutput() .

Voir aussi

OpenSerialPort() , WriteSerialPortData()

OS Supportés

Tous

Chapitre 137

Sort

Généralités

PureBasic propose des fonctions de tri optimisées pour trier les données numériques ou alphanumériques contenues dans les tableaux et les listes , soit en ordre croissant soit en ordre décroissant.
Les listes structurées sont triées avec la méthode Mergesort qui est de type stable, ce qui veut dire que si vous triez suivant un champs, les autres champs ne sont pas touchés, ainsi si vous commencez par trier une liste structurée par un champs "titre" et ensuite par un champs "album", vous obtiendrez une liste qui est triée par album et chaque album est trié par titres.
Par contre le tri des tableaux structurés ou non, utilise la méthode Quicksort qui est instable, c'est-à-dire que le tri sur la clé secondaire est perdu.
De plus, il est aussi possible de réorganiser les données d'un tableau ou d'une liste, de façon aléatoire.

OS Supportés

Tous

137.1 SortArray

Syntaxe

```
SortArray(Tableau(), Options
    [, Debut, Fin])
```

Description

Trie les données d'un tableau.

Arguments

Tableau() Le tableau à trier.

Options Peut être une combinaison des constantes suivantes :
#PB_Sort_Ascending : Trie le tableau par ordre croissant (les plus petites valeurs en tête)
#PB_Sort_Descending : Trie le tableau par ordre décroissant (les plus grandes valeurs en tête)
Et la constante suivante :
#PB_Sort_NoCase : Trie un tableau de chaînes de caractères en ne tenant pas compte de la casse (a=A, b=B etc..).
Par défaut le tri est sensible à la casse.

Debut, Fin (optionnel) Ne trie que la plage de 'Debut' à 'Fin'.

Valeur de retour

Aucune.

Remarques

- Le tableau doit être de l'un des types suivants :
byte, word, long, string ou float.
- Les tableaux multi-dimensionnels ne sont pas supportés.
- Les tableaux structurés sont triés à l'aide de la fonction SortStructuredArray()
Note : Si un tableau n'est pas totalement plein, les éléments vides seront placés en tête dans l'ordre croissant et en fin dans l'ordre décroissant.
Nombres NaN (not a number) ne sont pas acceptés lors du tri car ils produisent des résultats aléatoires.

Voir aussi

SortList() , SortStructuredArray() , SortStructuredList() , RandomizeArray() , RandomizeList() , Random() , RandomSeed()

OS Supportés

Tous

137.2 SortList

Syntaxe

```
SortList(Liste(), Options [,
    Debut, Fin])
```

Description

Trie les données d'une liste.

Arguments

Liste() La liste à trier.

Options Peut être une combinaison des constantes suivantes :
#PB_Sort_Ascending : Trie la liste par ordre croissant (les plus petites valeurs en tête)
#PB_Sort_Descending : Trie la liste par ordre décroissant (les plus grandes valeurs en tête)
Et la constante suivante :
#PB_Sort_NoCase : Trie la liste de chaînes de caractères en ne tenant pas compte de la casse (a=A, b=B etc..).
Par défaut le tri est sensible à la casse.

Debut, Fin (optionnel) Ne trie que la plage de 'Debut' à 'Fin'.
Le premier élément d'une list a la position 0, le deuxième la position 1, etc..

Valeur de retour

Aucune.

Remarques

- La liste peut être de l'un des types suivants : byte, word, long, string ou float.
- Les listes structurées sont triées à l'aide de la fonction SortStructuredList() .

Voir aussi

SortArray() , SortStructuredArray() , SortStructuredList() , RandomizeArray() , RandomizeList() , Random() , RandomSeed()

OS Supportés

Tous

137.3 SortStructuredArray

Syntaxe

```
SortStructuredArray(Tableau(),
    Options,
    OffsetOf(Structure\Champs),
    TypeOf(Structure\Champs)
    [, Debut, Fin])
```

Description

Trie un tableau de structures.

Arguments

Tableau() Le tableau de structures à trier selon les options souhaitées.
Le tableau doit avoir une structure associée.

Options Peut être une combinaison des constantes suivantes :
#PB_Sort_Ascending : Trie le tableau par ordre croissant (les plus petites valeurs en tête)
#PB_Sort_Descending : Trie le tableau par ordre décroissant (les plus grandes valeurs en tête)
Et la constante suivante :
#PB_Sort_NoCase : Trie le tableau de chaînes de caractères en ne tenant pas compte de la casse (a=A, b=B etc..).
Par défaut le tri est sensible à la casse.

OffsetOf(Structure\Champs)
OffsetOf() peut être utilisé pour obtenir l'offset du champ dans la structure associée au tableau.

TypeOf(Structure\Champs TypeOf() définit le type du champ de la structure qui sert pour le tri. Les types disponibles sont :

```
#PB_Byte      : Le champ de
 la structure est un
 octet (.b)
#PB_Word      : Le champ de
 la structure est un word
 (.w)
#PB_Long      : Le champ de
 la structure est un long
 (.l)
#PB_String    : Le champ de
 la structure est un
 string (.s ou $). Les
 strings fixes (fixed
 strings) ne sont pas
 supportées)
#PB_Float     : Le champ de
 la structure est un
 flottant (.f)
#PB_Double    : Le champ de
 la structure est un
 double (.d)
#PB_Quad      : Le champ de
 la structure est un quad
 (.q)
#PB_Character: Le champ de
 la structure est un
 caractère (.c)
#PB_Integer   : Le champ de
 la structure est un
 integer (.i)
#PB_Ascii     : Le champ de
```

```
la structure est un
caractère ascii (.a)
#PB_Unicode  : Le champ de
la structure est un
caractère unicode (.u)
```

Debut, Fin (optionnel) Ne trie que la plage de 'Debut' à 'Fin'.

Valeur de retour

Aucune.

Remarques

Notes : Si un tableau n'est pas totalement plein, les éléments vides seront placés en tête dans l'ordre croissant et en fin dans l'ordre décroissant.
Les chaînes fixes (fixed strings) ne sont pas supportées dans les commandes de tri.
Nombres NaN (not a number) ne sont pas acceptés lors du tri car ils produisent des résultats aléatoires.

Exemple

```
1   Structure Animal
2     Nom$
3     Vitesse.l
4   EndStructure
5
6   Dim Animaux.Animal(2)
7
8   Animaux(0)\Nom$ = "Tigre"
9   Animaux(0)\Vitesse = 10
10
11  Animaux(1)\Nom$ = "Jaguar"
12  Animaux(1)\Vitesse = 40
13
14  Animaux(2)\Nom$ = "Zèbre"
15  Animaux(2)\Vitesse = 30
16
17  ; Trie le tableau en
      fonction du champ
      'Nom$'qui est une chaîne
      de caractères (String)
18  ;
19  SortStructuredArray(Animaux(),
      #PB_Sort_Ascending,
      OffsetOf(Animal\Nom$),
      TypeOf(Animal\Nom$))
20
21  For k=0 To 2
22    Debug Animaux(k)\Nom$+" -
      Vitesse :
      "+Str(Animaux(k)\Vitesse)
23  Next
24
```

```
25    ; Trie le tableau en
        fonction du champ
        'Vitesse' qui est un long
26    ;
27    SortStructuredArray(Animaux(),
        #PB_Sort_Ascending,
        OffsetOf(Animal\Vitesse),
        TypeOf(Animal\Vitesse))
28
29    For k=0 To 2
30      Debug Animaux(k)\Nom$+" -
        Vitesse :
        "+Str(Animaux(k)\Vitesse)
31    Next
```

Voir aussi

SortArray() , SortList() , SortStructuredList() , RandomizeArray() , RandomizeList() , Random() , RandomSeed()

OS Supportés

Tous

137.4 SortStructuredList

Syntaxe

```
SortStructuredList(Liste(),
    Options,
    OffsetOf(Structure\Champs),
    TypeOf(Structure\Champs)
    [, Debut, Fin])
```

Description

Trie une liste de structures.

Arguments

Liste() La liste de structures à trier selon les options souhaitées.
La liste doit avoir une structure associée.

Options Peut être une combinaison des constantes suivantes :
#PB_Sort_Ascending : Trie la liste par ordre croissant (les plus petites valeurs en tête)
#PB_Sort_Descending : Trie la liste par ordre décroissant (les plus grandes valeurs en tête)
Et la constante suivante :
#PB_Sort_NoCase : Trie la liste de chaînes de caractères en ne tenant pas compte de la casse (a=A, b=B etc..).
Par défaut le tri est sensible à la casse.

OffsetOf(Structure\Champs)
OffsetOf() peut être utilisé pour obtenir l'offset du champ dans la structure associée à la liste.

TypeOf(Structure\Champs TypeOf() définit le type du champ de la structure qui sert pour le tri. Les types disponibles sont :

```
#PB_Byte      : Le champ de
 la structure est un
 octet (.b)
#PB_Word      : Le champ de
 la structure est un word
 (.w)
#PB_Long      : Le champ de
 la structure est un long
 (.l)
#PB_String    : Le champ de
 la structure est un
 string (.s ou $). Les
 strings fixe (fixed
 strings) ne sont pas
 supportées)
#PB_Float     : Le champ de
 la structure est un
 flottant (.f)
#PB_Double    : Le champ de
 la structure est un
 double (.d)
#PB_Quad      : Le champ de
 la structure est un quad
 (.q)
#PB_Character: Le champ de
 la structure est un
 caractère (.c)
#PB_Integer   : Le champ de
 la structure est un
 integer (.i)
#PB_Ascii     : Le champ de
 la structure est un
 caractère ascii (.a)
#PB_Unicode   : Le champ de
 la structure est un
 caractère unicode (.u)
```

Debut, Fin (optionnel) Ne trie que la plage de 'Debut' à 'Fin'.
Le premier élément d'une list a la position 0, le deuxième la position 1, etc..

Valeur de retour

Aucune.

Remarques

Notes : Les chaînes fixes (fixed strings) ne sont pas supportées dans les commandes de tri.

Exemple

```
Structure Animal
  Nom$
  Vitesse.l
EndStructure

NewList Animaux.Animal()

AddElement(Animaux())
Animaux()\Nom$ = "Tigre"
Animaux()\Vitesse = 10

AddElement(Animaux())
Animaux()\Nom$ = "Jaguar"
Animaux()\Vitesse = 40

AddElement(Animaux())
Animaux()\Nom$ = "Zèbre"
Animaux()\Vitesse = 30

; Trie la liste en fonction du champ 'Nom$'qui est une chaîne de caractères (String)
;
SortStructuredList(Animaux(), #PB_Sort_Ascending, OffsetOf(Animal\Nom$),TypeOf(Animal\Nom$))

ForEach Animaux()
  Debug Animaux()\Nom$+" - Vitesse : "+Str(Animaux()\Vitesse)
Next

; Trie la liste en fonction du champ 'Vitesse' qui est un long
;
SortStructuredList(Animaux(), #PB_Sort_Ascending, OffsetOf(Animal\Vitesse), TypeOf(Animal\Vitesse))

ForEach Animaux()
  Debug Animaux()\Nom$+" - Vitesse : "+Str(Animaux()\Vitesse)
Next
```

Voir aussi

SortArray() , SortList() , SortStructuredArray() , RandomizeArray() , RandomizeList() , Random() , RandomSeed()

OS Supportés

Tous

137.5 RandomizeArray

Syntaxe

```
RandomizeArray(Tableau() [,
    Debut, Fin])
```

Description

Réordonne les éléments d'un tableau dans un ordre aléatoire.

Arguments

Tableau() Le tableau à mettre dans un ordre aléatoire.

Debut, Fin (optionnel) L'index du premier et du dernier élément du tableau qui doivent être mis dans un ordre aléatoire. Si ces paramètres ne sont pas spécifiés alors la totalité du tableau sera mis dans un ordre aléatoire.

Valeur de retour

Aucune.

Remarques

Cette fonction utilise le générateur de nombres pseudo-aléatoire de la fonction Random() pour déterminer le nouvel ordre des éléments du tableau. C'est donc dépendant du RandomSeed() courant.

Voir aussi

SortArray() , SortStructuredArray() , RandomizeList() , Random() , RandomSeed()

OS Supportés

Tous

137.6 RandomizeList

Syntaxe

```
RandomizeList(Liste() [,
    Debut, Fin])
```

Description

Réordonne les éléments d'une liste dans un ordre aléatoire.

Arguments

Liste() La liste à mettre dans un ordre aléatoire.

Debut, Fin (optionnel) L'index du premier et du dernier élément de la Liste qui doivent être dans mis un ordre aléatoire. Si ces paramètres ne sont pas spécifiés alors la totalité du tableau sera mis dans un ordre aléatoire.
Le premier élément d'une list a la position 0, le deuxième la position 1, etc..

Valeur de retour

Aucune.

Remarques

Cette fonction utilise le générateur de nombres pseudo-aléatoire de la fonction Random() pour déterminer le nouvel ordre des éléments de la liste. C'est donc dépendant du RandomSeed() courant.

Voir aussi

SortList() , SortStructuredList() , RandomizeArray() , Random() , RandomSeed()

OS Supportés

Tous

Chapitre 138

Sound

Généralités

La bibliothèque "Son" de PureBasic permet d'ajouter simplement du son à une application ou à un jeu. Les commandes disponibles sont spécialement optimisées pour tirer le maximum du matériel. DirectX est utilisé sous Windows et la bibliothèque SDL est utilisée sous Linux.
Une version récente de DirectX 9 doit être installée (voir ici : DirectX 9 runtime installer).

OS Supportés

Tous

138.1 CatchSound

Syntaxe

```
Resultat = CatchSound(#Son,
    *Memoire [, Taille [,
    Options]])
```

Description

Charge un Son qui se trouve déjà en mémoire.
Les formats reconnus sont le WAV au format PCM (le format ADPCM n'est pas supporté) ou n'importe quel format supporté par la bibliothèque SoundPlugin .
UseFLACSoundDecoder()
UseOGGSoundDecoder()

Arguments

#Son Le numéro d'identification du nouveau Son.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

***Memoire** L'adresse de la mémoire où se trouve le Son.

Taille (optionnel) La taille en octets de la zone mémoire appelée tampon ou buffer qui contient le Son.
Attention, ce paramètre est facultatif avec les fichiers WAV, mais il est obligatoire pour les autres formats sonores.

Options (optionnel) Peut être une combinaison des valeurs suivantes :

```
#PB_Sound_Streaming:
 Active le support du
 streaming
                        Seulement
 pour les formats FLAC
et
                        OGG
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Exemple

```
1  CatchSound(0, ?Music)
2  End
3
4  DataSection
5    Music:
6    IncludeBinary "Sound.wav"
```

Remarques

Le " ?" est un pointeur sur une étiquette (label). Plus d'informations sur les pointeurs et les accès mémoire peuvent être trouvées dans le chapitre ici .

Voir aussi

LoadSound() , FreeSound() , PlaySound()

OS Supportés

Tous

138.2 GetSoundPosition

Syntaxe

```
Resultat =
   GetSoundPosition(#Son [,
   Mode [, Canal]])
```

Description

Renvoie la position courante.

Arguments

#Son Le son à utiliser.

Mode (optionnel) Peut être une des valeurs suivantes :

```
#PB_Sound_Frame       :
 Position en 'frame' (par
 défaut).
#PB_Sound_Millisecond:
 Position en
 millisecondes.
```

Canal (optionnel) Le canal à utiliser. C'est la valeur retournée par PlaySound() avec l'option #PB_Sound_MultiChannel.

Valeur de retour

Renvoie la position courante du son ou -1 si une erreur s'est produite.

Remarques

Les sons chargés avec l'option #PB_Sound_Streaming ne sont pas pris en charge.

Exemple

```
1  InitSound()            ;
     Initialisation des Sons
2  UseOGGSoundDecoder()   ;
     Utilisation des fichiers
     ogg
3
4  ; Chargement de 2 sons
     depuis 2 fichiers
5  LoadSound(0,
     #PB_Compiler_Home
     +"Examples\3D\Data\Siren.ogg")
6  LoadSound(1,
     #PB_Compiler_Home
     +"Examples\3D\Data\Roar.ogg")
7
8  ; La sirène joue
9  PlaySound(0)
10
11 ; Afficher la position
     pendant que la sirène joue
12  Repeat
13   Pos=GetSoundPosition(0,
     #PB_Sound_Millisecond)
14   Delay(100)    ; Attendre
     100 ms
15   Debug Pos     ; Afficher la
     position toutes les 100 ms
     environ
```

```
16    If Pos>1000  ; Arrêter
      après 1 seconde environ
17      Break
18    EndIf
19   ForEver
20
21  ; Puis les 2 sons sont joués
      en même temps
22    PlaySound(1)
23
24  MessageRequester("Info", "Ok
      pour quitter.")
25  End
```

Voir aussi

SetSoundPosition()

OS Supportés

Tous

138.3 SetSoundPosition

Syntaxe

```
SetSoundPosition(#Son,
    Position, [, Mode [,
    Canal]])
```

Description

Régle la position courante.

Arguments

#Son Le son à utiliser.

Position La nouvelle position.

Mode (optionnel) Peut être une des valeurs suivantes :

```
#PB_Sound_Frame       :
 Position en 'frame' (Par
 défaut).
#PB_Sound_Millisecond:
 Position en
 millisecondes.
```

Canal (optionnel) Le canal à utiliser. C'est la valeur retournée par PlaySound() avec l'option #PB_Sound_MultiChannel.

Valeur de retour

Aucune.

Remarques

Les sons chargés avec l'option #PB_Sound_Streaming ne sont pas pris en charge.

Exemple

```
1  InitSound()            ; Initialisation des Sons
2  UseOGGSoundDecoder()   ; Utilisation des fichiers ogg
3
4  ; Chargement d'un son depuis un fichier
5  LoadSound(0, #PB_Compiler_Home +"Examples\3D\Data\Siren.ogg")
6
7  ; Le son joue
8  PlaySound(0)
9
10 ; Mettre la position a 2 secondes
11 SetSoundPosition(0, 2000, #PB_Sound_Millisecond)
12
13 MessageRequester("Info", "Ok pour quitter.")
14 End
```

Voir aussi

GetSoundPosition()

OS Supportés

Tous

138.4 FreeSound

Syntaxe

```
FreeSound(#Son)
```

Description

Arrête et supprime de la mémoire un Son précédemment chargé avec la fonction LoadSound() ou CatchSound() .

Arguments

#Son Le Son à supprimer.
Si #PB_All est utilisé alors tous les Sons seront supprimés.

Valeur de retour

Aucune.

Remarques

Tous les sons restants sont automatiquement libérés quand le programme se termine.

Exemple

```
InitSound()            ; Initialisation des Sons
UseOGGSoundDecoder()   ; Utilisation des fichiers ogg

; Chargement d'un son depuis un fichier
LoadSound(0, #PB_Compiler_Home +"Examples\3D\Data\Siren.ogg")
; Le son joue en boucle
PlaySound(0, #PB_Sound_Loop)

MessageRequester("Info", "Ok pour quitter.")

FreeSound(0) ; Le son est libéré
End
```

Voir aussi

LoadSound()

OS Supportés

Tous

138.5 InitSound

Syntaxe

```
Resultat = InitSound()
```

Description

Initialise l'environnement sonore.

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Les principales causes d'échec sous Windows sont : DirectX non trouvé ou carte son non détectée.
(voir ici : DirectX 9 runtime installer).

Remarques

Cette fonction doit toujours être appelée avant toute autre fonction de la bibliothèque Sound.

Exemple

```
1  InitSound()            ;
     Initialisation des Sons
2  UseOGGSoundDecoder()   ;
     Utilisation des fichiers
     ogg
3
4  ; Chargement d'un son depuis
     un fichier
5  LoadSound(0,
     #PB_Compiler_Home
     +"Examples\3D\Data\Siren.ogg")
6  ; Le son joue en boucle
7  PlaySound(0, #PB_Sound_Loop)
8
9  MessageRequester("Info", "Ok
     pour quitter.")
10
11 End
```

OS Supportés

Tous

138.6 IsSound

Syntaxe

```
Resultat = IsSound(#Son)
```

Description

Teste si un Son est correctement initialisé.

Arguments

#**Son** Le Son à utiliser.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.

Exemple

```
1  If IsSound(0) = 0
2    MessageRequester("Info",
     "Le son n'est pas valide.")
3  EndIf
```

Voir aussi

FreeSound()

OS Supportés

Tous

138.7 LoadSound

Syntaxe

```
Resultat = LoadSound(#Son,
    NomFichier$ [, Options])
```

Description

Charge un Son en mémoire depuis un fichier.
Les formats reconnus sont le WAV au format PCM (le format ADPCM n'est pas supporté) ou n'importe quel format supporté par la bibliothèque SoundPlugin .
UseFLACSoundDecoder()
UseOGGSoundDecoder()

Arguments

#Son Le numéro d'identification du nouveau Son.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

NomFichier$ Le chemin et le nom du fichier sonore à charger.

Options (optionnel) Peut être une combinaison des valeurs suivantes :

```
#PB_Sound_Streaming:
 Active le support du
 streaming pour jouer le
 son
                      seulement
 pour les formats FLAC
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Exemple

```
1  InitSound()            ;
     Initialisation des Sons
2  UseOGGSoundDecoder()   ;
     Utilisation des fichiers
     ogg
3
4  ; Chargement d'un son depuis
     un fichier
5  LoadSound(0,
     #PB_Compiler_Home
     +"Examples\3D\Data\Siren.ogg")
6  ; Le son joue en boucle
7  PlaySound(0, #PB_Sound_Loop)
8
9  MessageRequester("Info", "Ok
     pour quitter.")
10
11 FreeSound(0) ; Le son est
     libéré
12 End
```

Voir aussi

CatchSound() , FreeSound() , PlaySound()

OS Supportés

Tous

138.8 PauseSound

Syntaxe

```
PauseSound(#Son [, Canal])
```

Description

Met en pause.

Arguments

#Son Le son à utiliser.
Si #PB_All est spécifié, tous les sons (et tous les canaux) sont mis en pause.

Canal (optionnel) Le canal à utiliser.
C'est la valeur retournée par PlaySound() avec l'option #PB_Sound_MultiChannel.

Valeur de retour

Aucune.

Remarques

Les sons chargés avec l'option #PB_Sound_Streaming ne sont pas pris en charge.

Exemple

```
InitSound()            ; Initialisation des Sons
UseOGGSoundDecoder()   ; Utilisation des fichiers ogg

; Chargement d'un son depuis un fichier
LoadSound(0, #PB_Compiler_Home +"Examples\3D\Data\Siren.ogg")
; Le son joue en boucle
PlaySound(0, #PB_Sound_Loop)

MessageRequester("Info", "Ok pour mettre en pause.")
PauseSound(0)  ; Pause

MessageRequester("Info", "Ok pour reprendre.")
ResumeSound(0)  ; Reprendre la lecture

MessageRequester("Info", "Ok pour quitter.")

FreeSound(0) ; Le son est libéré
End
```

Voir aussi

LoadSound() , ResumeSound()

OS Supportés

Tous

138.9 ResumeSound

Syntaxe

```
ResumeSound(#Son [, Canal])
```

Description

Reprend la lecture.

Arguments

#Son Le son à utiliser.
Si #PB_All est spécifié, tous les sons (et tous les canaux) reprennent la lecture.

Canal (optionnel) Le canal à utiliser.
C'est la valeur retournée par PlaySound() avec l'option #PB_Sound_MultiChannel.

Valeur de retour

Aucune.

Remarques

Les sons chargés avec l'option #PB_Sound_Streaming ne sont pas pris en charge.

Exemple

```
InitSound()            ; Initialisation des Sons
UseOGGSoundDecoder()   ; Utilisation des fichiers ogg

; Chargement d'un son depuis un fichier
LoadSound(0, #PB_Compiler_Home +"Examples\3D\Data\Siren.ogg")
; Le son joue en boucle
PlaySound(0, #PB_Sound_Loop)

MessageRequester("Info", "Ok pour mettre en pause.")
PauseSound(0)  ; Pause

MessageRequester("Info", "Ok pour reprendre.")
ResumeSound(0)  ; Reprendre la lecture

MessageRequester("Info", "Ok pour quitter.")

FreeSound(0) ; Le son est libéré
End
```

Voir aussi

LoadSound() , PauseSound()

OS Supportés

Tous

138.10 PlaySound

Syntaxe

```
Resultat = PlaySound(#Son [,
    Option [, Volume]])
```

Description

Joue un Son.

Arguments

#**Son** Le Son à jouer.

Option (optionnel) Peut être la combinaison des valeurs suivantes :

```
    #PB_Sound_Loop         :
     joue le son en boucle
     (infini)
    #PB_Sound_MultiChannel:
     joue le son en utilisant
     un nouveau canal au lieu
     de stopper le son
     précédent. Ceci permet
     d'utiliser
                           le
     même son et de le jouer
     plusieurs fois sur
     différents canaux.
     'Resultat' contiendra le
     nouveau canal alloué,
                           et
     pourra être spécifié
     avec SoundVolume()
  , etc...
```

Volume (optionnel) Permet de régler le volume initial du Son.
Les valeurs valides vont de 0 (aucun volume) à 100 (volume maximum).
La valeur par défaut est 100.

Valeur de retour

Renvoie le canal si l'option #PB_Sound_MultiChannel est utilisée.

Exemple

```
1 InitSound()           ;
    Initialisation des Sons
2 UseOGGSoundDecoder()  ;
    Utilisation des fichiers
    ogg
```

```
3
4   ; Chargement d'un son depuis
      un fichier
5   LoadSound(0,
      #PB_Compiler_Home
      +"Examples\3D\Data\Siren.ogg")
6   ; Le son joue en boucle
7   PlaySound(0, #PB_Sound_Loop,
      20)
8
9   MessageRequester("Info", "Ok
      pour quitter.")
10
11  FreeSound(0) ; Le son est
      libéré
12  End
```

Voir aussi

StopSound() , FreeSound() , PauseSound() , ResumeSound()

OS Supportés

Tous

138.11 GetSoundFrequency

Syntaxe

```
GetSoundFrequency(#Son, [,
    Canal])
```

Description

Renvoie la fréquence moyenne.

Arguments

#Son Le Son à utiliser.

Canal (optionnel) Le canal à utiliser. C'est la valeur renvoyée par PlaySound() avec l'option #PB_Sound_MultiChannel.

Valeur de retour

Renvoie la fréquence courante du son, en Hz.

Exemple

```
1   InitSound()           ;
      Initialisation des Sons
2   UseOGGSoundDecoder()  ;
      Utilisation des fichiers
      ogg
```

```
3
4   ; Chargement d'un son depuis
      un fichier
5   LoadSound(0,
      #PB_Compiler_Home
      +"Examples\3D\Data\Siren.ogg")
6   ; Le son joue en boucle
7   PlaySound(0, #PB_Sound_Loop,
      20)
8
9   MessageRequester("Info", "La
      fréquence moyenne est de "
      +
      Str(GetSoundFrequency(0))+"
      Hz")
10
11  MessageRequester("Info", "Ok
      pour quitter.")
12
13  FreeSound(0) ; Le son est
      libéré
14  End
```

Voir aussi

SetSoundFrequency()

OS Supportés

Windows

138.12 SetSoundFrequency

Syntaxe

```
SetSoundFrequency(#Son,
    Frequence [, Canal])
```

Description

Régle une fréquence en temps réel.

Arguments

#Son Le Son à utiliser.

Frequence La fréquence à utiliser.
Les valeurs valides sont comprises entre 1 000 Hz et 100 000 Hz.

Canal (optionnel) Le canal à utiliser.
C'est la valeur retournée par PlaySound() avec l'option #PB_Sound_MultiChannel.
Si #PB_All est spécifié, tous les canaux sont affectés.

Valeur de retour

Aucune.

Remarques

La nouvelle valeur de la fréquence est enregistrée pour le son, il n'est donc pas nécessaire de l'appeler à chaque fois.

Exemple

```
InitSound()           ; Initialisation des Sons
UseOGGSoundDecoder()  ; Utilisation des fichiers ogg

; Chargement d'un son depuis un fichier
LoadSound(0, #PB_Compiler_Home +"Examples\3D\Data\Siren.ogg")
; Le son joue en boucle
PlaySound(0, #PB_Sound_Loop, 20)

MessageRequester("Info", "La fréquence moyenne est de " + Str(GetSoundFrequency(0))+" Hz")
PauseSound(0)

SetSoundFrequency(0,16000)

ResumeSound(0)

MessageRequester("Info", "La fréquence moyenne est de " + Str(GetSoundFrequency(0))+" Hz")

FreeSound(0) ; Le son est libéré
End
```

Voir aussi

GetSoundFrequency()

OS Supportés

Windows

138.13 SoundStatus

Syntaxe

```
Resultat = SoundStatus(#Son
    [, Canal])
```

Description

Renvoie l'état d'un Son.

Arguments

#Son Le Son à utiliser.

Canal (optionnel) Le canal à utiliser. C'est la valeur renvoyée par PlaySound() avec l'option #PB_Sound_MultiChannel.

Valeur de retour

Peut être une des valeurs suivantes :

```
#PB_Sound_Stopped: Le son
 est arrêté.
#PB_Sound_Playing: Le son
 est joué.
#PB_Sound_Paused : Le son
 est en pause.
#PB_Sound_Unknown: Le son
 est dans un état inconnu
 (une erreur s'est produite
 lors de l'obtention de
 l'état).
```

Exemple

```
1   Procedure
     SelectStatus(Status)
2     Select Status
3       Case #PB_Sound_Stopped
4         MessageRequester("Info",
    "Le son est arrêté.")
5
6       Case #PB_Sound_Playing
7         MessageRequester("Info",
    "Le son est joué.")
8
9       Case  #PB_Sound_Paused
10        MessageRequester("Info",
    "Le son est en pause.")
11
12      Case #PB_Sound_Unknown
13        MessageRequester("Info",
    "Statut inconnu.")
14
15      Default
16        MessageRequester("Info",
    "Statut inconnu.")
17
18    EndSelect
19  EndProcedure
20
```

```
21 InitSound()            ;
     Initialisation des Sons
22 UseOGGSoundDecoder()   ;
     Utilisation des fichiers
     ogg
23
24 ; Chargement d'un son depuis
     un fichier
25 LoadSound(0,
     #PB_Compiler_Home
     +"Examples\3D\Data\Siren.ogg")
26 ; Le son joue en boucle
27 PlaySound(0, #PB_Sound_Loop,
     20)
28 SelectStatus(SoundStatus(0))
29
30 PauseSound(0)
31 SelectStatus(SoundStatus(0))
32
33 ResumeSound(0)
34 SelectStatus(SoundStatus(0))
35
36 StopSound(0)
37 SelectStatus(SoundStatus(0))
38
39 FreeSound(0) ; Le son est
     libéré
40 End
```

Voir aussi

PlaySound() , StopSound() , PauseSound() , ResumeSound()

OS Supportés

Tous

138.14 SoundPan

Syntaxe

```
SoundPan(#Son, Spacialisation
    [, Canal])
```

Description

Change la stéréo d'un Son en temps réel.

Arguments

#Son Le Son à utiliser.

Spacialisation La nouvelle valeur de spacialisation stéréo.
Elle devient la valeur par défaut pour ce #Son.

Les valeurs disponibles vont de -100 (tout le son à gauche) à 100 (tout le son à droite).
Si la stéréo est 0, alors le son est joué sur les deux haut-parleurs.

Canal (optionnel) Le canal à utiliser. C'est la valeur renvoyée par PlaySound() avec l'option #PB_Sound_MultiChannel.

Valeur de retour

Aucune.

Exemple

```
InitSound()           ; Initialisation des Sons
UseOGGSoundDecoder()  ; Utilisation des fichiers ogg

; Chargement d'un son depuis un fichier
LoadSound(0, #PB_Compiler_Home +"Examples\3D\Data\Siren.ogg")
; Le son joue en boucle
PlaySound(0, #PB_Sound_Loop, 20)

MessageRequester("Info", "Le son est joué en stéréo.")

SoundPan(0, -100)
MessageRequester("Info", "Le son est joué en mono sur le canal gauche.")

SoundPan(0, 100)
MessageRequester("Info", "Le son est joué en mono sur le canal droit.")

SoundPan(0, 0)
MessageRequester("Info", "Le son est joué en stéréo.")

FreeSound(0) ; Le son est libéré
End
```

OS Supportés

Windows, MacOS X

138.15 SoundLength

Syntaxe

```
SoundLength(#Son [, Mode])
```

Description

Renvoie la longueur d'un Son.

Arguments

#Son Le Son à utiliser.

Mode (optionnel) Peut être une des valeurs suivantes :

```
#PB_Sound_Frame       :
 Longueur en 'frame' (par
 défaut).
#PB_Sound_Millisecond:
 Longueur en
 millisecondes.
```

Valeur de retour

Renvoie la longueur du Son ou -1, si une erreur s'est produite.

Remarques

Les sons chargés avec l'option #PB_Sound_Streaming ne sont pas pris en charge.

Exemple

```
1  InitSound()           ;
     Initialisation des Sons
2  UseOGGSoundDecoder()  ;
     Utilisation des fichiers
     ogg
3
4  ; Chargement d'un son depuis
     un fichier
5  LoadSound(0,
     #PB_Compiler_Home
     +"Examples\3D\Data\Siren.ogg")
6  ; Le son joue en boucle
7  PlaySound(0, #PB_Sound_Loop,
     20)
8
9  MessageRequester("Info", "La
     durée du son est de "+
     Str(SoundLength(0)) + "
     frames.")
10
```

```
11 | MessageRequester("Info", "La
     durée du son est de "+
     Str(SoundLength(0,
     #PB_Sound_Millisecond)) +
     " ms.")
12 |
13 | FreeSound(0) ; Le son est
     libéré
14 | End
```

OS Supportés

Tous

138.16 SoundVolume

Syntaxe

```
SoundVolume(#Son, Volume [,
    Canal])
```

Description

Change le volume en temps réel.

Arguments

#Son Le Son à utiliser.
Si #PB_All est spécifié, tous les Sons (et tous les canaux) reprennent la lecture.

Volume Le nouveau volume.
Les valeurs valides sont comprises entre 0 (pas de volume) à 100 (volume maximal).

Canal (optionnel) Le canal à utiliser.
C'est la valeur retournée par PlaySound() avec l'option #PB_Sound_MultiChannel.
Si #PB_All est spécifié, tous les canaux sont affectés.

Valeur de retour

Aucune.

Exemple

```
1 | InitSound()           ;
    Initialisation des Sons
2 | UseOGGSoundDecoder()  ;
    Utilisation des fichiers
    ogg
3 |
4 | ; Chargement d'un son depuis
    un fichier
5 | LoadSound(0,
    #PB_Compiler_Home
    +"Examples\3D\Data\Siren.ogg")
```

```
; Le son joue en boucle
PlaySound(0, #PB_Sound_Loop, 20)

MessageRequester("Info", "Le volume sonore est à 20%")

SoundVolume(0, 80)
MessageRequester("Info", "Le volume sonore est à 80%")

FreeSound(0) ; Le son est libéré
End
```

Voir aussi

LoadSound()

OS Supportés

Tous

138.17 StopSound

Syntaxe

```
StopSound(#Son, [, Canal])
```

Description

Arrête la lecture.

Arguments

#Son Le Son à utiliser.
Si #PB_All est spécifié, tous les Sons (et tous les canaux) sont arrêtés.

Canal (optionnel) Le canal à utiliser.
C'est la valeur retournée par PlaySound() avec l'option #PB_Sound_MultiChannel.

Valeur de retour

Aucune.

Exemple

```
InitSound()           ; Initialisation des Sons
UseOGGSoundDecoder()  ; Utilisation des fichiers ogg

; Chargement d'un son depuis un fichier
```

```
5  LoadSound(0,
     #PB_Compiler_Home
     +"Examples\3D\Data\Siren.ogg")
6  ; Le son joue en boucle
7  PlaySound(0, #PB_Sound_Loop,
     20)
8
9  MessageRequester("Info", "Ok
     pour stopper le son")
10
11 StopSound(0)
12 MessageRequester("Info",
     "Son stoppé")
13
14 FreeSound(0) ; Le son est
     libéré
15 End
```

Voir aussi

PlaySound()

OS Supportés

Tous

Chapitre 139

Sound3D

Généralités

Cette bibliothèque permet d'ajouter des sons avec un effet 3D dans un monde en 3D. Un exemple simple est un son qui sera automatiquement atténué en fonction de la distance.
La syntaxe est similaire à la bibliothèque Sound .
Cette bibliothèque utilise un moteur graphique pour fonctionner ainsi la commande InitEngine3D() doit être appelée avec succès avant de pouvoir utiliser les commandes relatives aux sons 3D.

OS Supportés

Tous

139.1 FreeSound3D

Syntaxe

```
FreeSound3D(#Son3D)
```

Description

Arrête et supprime de la mémoire un son 3D.

Arguments

#Son3D Le son 3D à supprimer.
Si #PB_All est spécifié, tous les sons 3D restants sont libérés.

Valeur de retour

Aucune.

Remarques

Tous les sons restants sont automatiquement libérés quand le programme se termine.

Voir aussi

LoadSound3D()

OS Supportés

Tous

139.2 IsSound3D

Syntaxe

```
Resultat = IsSound3D(#Son3D)
```

Description

Teste si un son 3D est correctement initialisé.

Arguments

#**Son3D** Le son 3D à utiliser.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.

Voir aussi

LoadSound3D()

OS Supportés

Tous

139.3 LoadSound3D

Syntaxe

```
Resultat =
    LoadSound3D(#Son3D,
    NomFichier$ [, Option])
```

Description

Charge un fichier son 3D en mémoire.

Arguments

#**Son3D** Le numéro d'identification du son 3D chargé.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

NomFichier$ Le nom du fichier sonore, de type WAV ou OGG.

Option (optionnel)
`#PB_Sound_Streaming` : Active le support du streaming.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Le son 3D doit être en mono.
Un son 3D n'a pas de position.
Il peut être attaché à un noeud pour avoir sa propre position. Un son 3D est joué par rapport à l'emplacement d'écoute. Pour modifier l'emplacement d'écoute, utilisez SoundListenerLocate() .

Exemple

```
1    ; Initialisation du monde 3D
2    InitEngine3D()
3    InitSprite()
4    InitKeyboard()
5    InitMouse()
6    Add3DArchive(#PB_Compiler_Home+"Examples/3D/Data", #PB_3DArchive_FileSystem)
7    Add3DArchive(#PB_Compiler_Home+"Examples/3D/Data/Packs/skyb #PB_3DArchive_Zip)
8    Add3DArchive(#PB_Compiler_Home+"Examples/3D/Data/Textures", #PB_3DArchive_FileSystem)
9
10   ; Ouverture de la fenêtre
11   OpenWindow(0,0,0,1000,1000,"Le Son varie en fonction de la position de la planète",#PB_Window_ScreenCentered|#PB_Window_SystemMenu)
12   OpenWindowedScreen(WindowID(0),0,0,1000,1000,1,0,0)
13
14   ;Création du monde 3D
15   SkyBox("stevecube.jpg");
16   cam=CreateCamera(#PB_Any, 0, 0, 100, 100)
17   planete=CreateSphere(#PB_Any, 10)
18   tex_planete=LoadTexture(#PB_Any, "clouds.jpg")
```

```
19      material=CreateMaterial(#PB_Any,TextureID(tex_planete))
20      planeteE=CreateEntity(#PB_Any,MeshID(planete),MaterialID(ma
21
22      ; Chargement du fichier Son3d Wav ou Ogg, qui doit être mono
23      son3D= LoadSound3D(#PB_Any,"Roar.ogg",#PB_Sound3D_Loop)
24
25      SoundVolume3D(son3D, 20)        ; Volume sonore à 20%
26      SoundRange3D(son3D, 1, 100)      ; Zone d'émission du son
27      SoundCone3D(son3D, 90, 360, 20)  ; Facultatif, Cône sonore
28
29      ; Le Son est joué en boucle
30      PlaySound3D(son3D, #PB_Sound3D_Loop)
31
32      ; L'auditeur se déplace avec la caméra
33      SoundListenerLocate(CameraX(cam),CameraY(cam),CameraZ(cam))
34
35      ; Le Son et la planète sont liés,
36      ; ainsi si l'utilisateur déplace la planète, le son se déplace avec elle
37      noeud=CreateNode(#PB_Any, 0, 0, -75)
38      AttachNodeObject(noeud, SoundID3D(son3D))
39      AttachNodeObject(noeud, EntityID(planeteE))
40
41      ; Gestion de la fenêtre, du clavier et de la souris
42      Repeat
43        Repeat
44          Event  = WindowEvent()
45          Select Event
46            Case #PB_Event_CloseWindow
47          End
48
49          EndSelect
50        Until Event = 0
51
52        ExamineKeyboard()
53        ExamineMouse()
54
55        If MouseDeltaX()>0
56          MoveNode(noeud,0.5,0,0)
57        ElseIf MouseDeltaX()<0
```

```
58          MoveNode(noeud,-0.5,0,0)
59        EndIf
60        If MouseDeltaY()>0
61          MoveNode(noeud,0,-0.5,0)
62        ElseIf MouseDeltaY()<0
63          MoveNode(noeud,0,0.5,0)
64        EndIf
65        If
      MouseButton(#PB_MouseButton_Left)
      <>0
66          End
67        EndIf
68        If MouseButton(
      #PB_MouseButton_Right) <>0
69          MoveNode(noeud,0,0,-75,#PB_Absolute)
70        EndIf
71
72        If
      KeyboardPushed(#PB_Key_Escape)
73          quitter + 1
74        EndIf
75        RotateNode(noeud,0.3,0.4,0.5,#PB_Relative)
76        If
      KeyboardPushed(#PB_Key_Right)
77          MoveNode(noeud,0.5,0,0)
78        EndIf
79        If
      KeyboardPushed(#PB_Key_Left)
80          MoveNode(noeud,-0.5,0,0)
81        EndIf
82        If
      KeyboardPushed(#PB_Key_Up)
83          MoveNode(noeud,0,0.5,0)
84        EndIf
85        If
      KeyboardPushed(#PB_Key_Down)
86          MoveNode(noeud,0,-0.5,0)
87        EndIf
88        tiks = MouseWheel()
89        If tiks > 0
90          MoveNode(noeud,0,0,5)
91        EndIf
92        If tiks < 0
93          MoveNode(noeud,0,0,-5)
94        EndIf
95
96        ; Affichage de la scène
97        StartDrawing(WindowOutput(0))
98        DrawText(0,5,"X=" +
      Str(NodeX(noeud)) + " Y="
      + Str(NodeY(noeud)) + "
      Z="+Str(NodeZ(noeud)))
99        DrawText(0,30,"Souris
      ou Clavier: Haut, Bas,
      Droite, Gauche et Molette
      souris")
100       DrawText(0,60,".: Echap
      ou clic gauche pour
      quitter :.")
```

```
101         StopDrawing()
102
103         RenderWorld()
104
105         FlipBuffers()
106
107     Until quitter
108
109     End
```

Voir aussi

FreeSound3D() , IsSound3D()

OS Supportés

Tous

139.4 PlaySound3D

Syntaxe

```
PlaySound3D(#Son3D [, Option])
```

Description

Joue un son 3D.

Arguments

#**Son3D** Le son 3D à utiliser.

Option (optionnel)
```
#PB_Sound3D_Loop :
Joue le son en boucle.
```

Valeur de retour

Aucune.

Exemple

Voir LoadSound3D()

Voir aussi

StopSound3D() , LoadSound3D()

OS Supportés

Tous

139.5 SoundVolume3D

Syntaxe

```
SoundVolume3D(#Son3D, Volume)
```

Description

Change le volume en temps réel.

Arguments

#Son3D Le son 3D à utiliser.

Volume Le nouveau volume devient la valeur par défaut pour ce #Son3D. Les valeurs valides vont de 0 (muet) à 100 (volume maximum).

Valeur de retour

Aucune.

Exemple

Voir LoadSound3D()

Voir aussi

SoundCone3D() , SoundRange3D()

OS Supportés

Tous

139.6 StopSound3D

Syntaxe

```
StopSound3D(#Son3D)
```

Description

Arrête l'éxécution d'un son 3D (s'il était en cours de lecture).

Arguments

#Son3D Le son 3D à utiliser. Avec #PB_All tous les sons en cours de lecture sont arrêtés.

Valeur de retour

Aucune.

Voir aussi

PlaySound3D() , LoadSound3D()

OS Supportés

Tous

139.7 SoundID3D

Syntaxe

```
Son3DID = SoundID3D(#Son3D)
```

Description

Renvoie l'identifiant unique d'un son 3D.

Arguments

#**Son3D** Le son 3D à utiliser.

Valeur de retour

Renvoie le numéro du son 3D.

Voir aussi

IsSound3D()

OS Supportés

Tous

139.8 SoundRange3D

Syntaxe

```
SoundRange3D(#Son3D, Minimum,
    Maximum)
```

Description

Modifie le champ d'émission, en unité du monde 3D.

Arguments

#**Son3D** Le son 3D à utiliser.

Minimum, Maximum Le paramètre 'Minimum' indique la distance la plus proche à partir de laquelle le son sera entendu par l'auditeur.
Le paramètre 'Maximum' indique la distance la plus grande jusqu'à laquelle le son sera entendu.
Au delà de cette distance le son ne sera plus joué.
Entre ces deux valeurs, le son est atténué en fonction de la position de l'auditeur.

Valeur de retour

Aucune.

Exemple

Voir LoadSound3D()

Voir aussi

SoundListenerLocate() , SoundCone3D() , SoundVolume3D()

OS Supportés

Tous

139.9 SoundCone3D

Syntaxe

```
SoundCone3D(#Son3D,
    ConeInterieur.f,
    ConeExterieur.f,
    VolumeConeExterieur)
```

Description

Change l'angle du cône sonore pour créer un son 3D directionnel.

Arguments

#Son3D Le son 3D à utiliser.

ConeInterieur, ConeExterieur Angles intérieur et extérieur du cône, en degré. Valeurs allant de 0 à 360°.

VolumeConeExterieur Correspond au volume du son en dehors du cône. Valeur entre 0 et 100.

Valeur de retour

Aucune.

Exemple

Voir LoadSound3D()

Voir aussi

SoundRange3D()

OS Supportés

Tous

139.10 SoundListenerLocate

Syntaxe

```
SoundListenerLocate(X, Y, Z)
```

Description

Change l'emplacement de l'auditeur
(l'oreille) dans le monde 3D.

Arguments

X, Y, Z Nouvelle position de l'auditeur.

Valeur de retour

Aucune.

Exemple

Voir LoadSound3D()

Voir aussi

SoundRange3D() , SoundCone3D() ,
SoundVolume3D()

OS Supportés

Tous

Chapitre 140

SoundPlugin

Généralités

PureBasic supporte différents formats de fichiers sons grâce à un système de plugins dynamiques.
Seul l'encodeur ou le décodeur nécessaire sera intégré au fichier exécutable, réduisant ainsi considérablement sa taille.
Par exemple, si vous avez seulement besoin du format OGG, seule la routine concernant ce format sera utilisée.
Une autre caractéristique intéressante est l'auto-détection du format de fichier, si plusieurs décodeurs sont utilisés.
Les commandes suivantes supportent les plugins sonores : LoadSound() et CatchSound() .

OS Supportés

Tous

140.1 UseFLACSoundDecoder

Syntaxe

```
UseFLACSoundDecoder()
```

Description

Active le support du format FLAC (Free Lossless Audio Codec) pour les commandes CatchSound() et LoadSound() .

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Comme le format FLAC est dit 'non-destructif', la taille du fichier restera assez conséquente, surtout comparée au format OGG . Il permet tout de même de réduire considérablement la taille des fichiers tout en gardant l'intégralité des informations du fichier.
Pour plus d'informations : Wikipedia - FLAC.
Le streaming du son est supporté pour ce plugin.

Voir aussi

UseOGGSoundDecoder()

OS Supportés

Tous

140.2 UseOGGSoundDecoder

Syntaxe

```
UseOGGSoundDecoder()
```

Description

Active le support du format OGG (OGG Vorbis) pour les commandes CatchSound() et LoadSound() .

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Le format de compression OGG est dit 'destructif', ce qui veut dire que de l'information (non essentielle) est perdue lors du processus de compression, permettant par la même occasion de diminuer considérablement la taille du fichier. C'est actuellement l'un des meilleurs formats de compression de sa catégorie, surpassant de loin le format MP3 classique que ce soit au niveau de la qualité ou de la taille.
Pour plus d'informations : Wikipedia - OGG.
Le streaming du son est supporté pour ce plugin.

Voir aussi

UseFLACSoundDecoder()

OS Supportés

Tous

Chapitre 141

SpecialEffect

Généralités

La bibliothèque d'effets spéciaux permet d'appliquer facilement plusieurs effets en temps réel dans le monde 3D , comme le post-traitement, les effets de traînée que l'on appelle aussi 'rubans', les lensflares (torches), etc.

OS Supportés

Tous

141.1 CreateCompositorEffect

Syntaxe

```
Resultat =
    CreateCompositorEffect(#Effet,
    CameraID, NomEffet$)
```

Description

Crée un nouvel effet compositeur (compositor).

Arguments

#Effet Le numéro d'identification du nouvel effet.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

CameraID La caméra qui applique l'effet. Cet effet n'affecte que cette caméra. Pour obtenir un numéro 'CameraID' valide, utiliser CameraID() .

NomEffet$ Nom de l'effet comme décrit dans le fichier compositeur OGRE(généralement un fichier '.compositor').

Valeur de retour

Renvoie une valeur non nulle si l'effet a été créé avec succès, zéro sinon.
Si #PB_Any a été utilisé comme paramètre #Effet alors le nouveau numéro généré est retourné en cas de succès.

Remarques

Une fois créé, l'effet est immédiatement appliqué au rendu.
Il est possible de masquer l'effet avec HideEffect() .

Voir aussi

FreeEffect() , HideEffect()

OS Supportés

Tous

141.2 CreateRibbonEffect

Syntaxe

```
Resultat =
    CreateRibbonEffect(#Effet,
    MatiereID, NbChaines,
    NbElements, Longueur)
```

Description

Crée un nouvel effet de traînée (ruban).

Arguments

#Effet Le numéro d'identification du nouvel effet.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

MatiereID Le matériau à appliquer sur le ruban.
Pour obtenir un 'MatiereID' valide, utiliser MaterialID() .

NbChaines Nombre de chaînes composant le ruban.
Plus il y a de chaînes et plus le ruban sera précis.

NbElements Nombre d'éléments par chaînes.
Plus il y a d'éléments et plus le ruban sera précis.

Longueur Longueur maximale, en unité du monde, du ruban.
Une fois la longueur atteinte, le ruban disparaît complètement.

Valeur de retour

Renvoie une valeur non nulle si l'effet a été créé avec succès, zéro sinon.
Si #PB_Any a été utilisé comme paramètre #Effet alors le nouveau numéro généré est renvoyé en cas de succès.

Remarques

Attention, une fois créé, l'effet reste invisible.
AttachRibbonEffect() permet d'afficher l'effet.
Il est possible de masquer l'effet avec HideEffect() .
Il est possible de masquer l'effet avec HideEffect() .

Voir aussi

FreeEffect() , HideEffect()

OS Supportés

Tous

141.3 RibbonEffectWidth

Syntaxe

```
RibbonEffectWidth(#Effet,
    IndexChaine, Largeur,
    LargeurFondu))
```

Description

Modifie la largeur de la chaîne d'un ruban.

Arguments

#Effet L'effet à utiliser.
Cet effet doit être créé avec CreateRibbonEffect() .

IndexChaine L'indice de la chaîne.
Le premier indice commence à zéro.
Ce numéro d'index doit être inférieur au nombre de chaînes créées par CreateRibbonEffect() .

Largeur La nouvelle largeur de la chaîne, dans l'unité du monde.

LargeurFondu La largeur du fondu par seconde.
Chaque seconde, la largeur de la chaîne du ruban sera diminuée de cette valeur jusqu'à zéro.

Valeur de retour

Aucune.

Voir aussi

CreateRibbonEffect()

OS Supportés

Tous

141.4 AttachRibbonEffect

Syntaxe

```
AttachRibbonEffect(#Effet,
    NoeudID)
```

Description

Attache un ruban au noeud donné.

Arguments

#**Effet** L'effet à utiliser.
Cet effet doit être créé avec CreateRibbonEffect() .

NoeudID Le noeud attaché au ruban.
Un seul ruban peut être fixé à plusieurs noeuds.
Pour obtenir un ID de noeud valide, utiliser NodeID()

Valeur de retour

Aucune.

Voir aussi

CreateRibbonEffect() ,
DetachRibbonEffect()

OS Supportés

Tous

141.5 DetachRibbonEffect

Syntaxe

```
DetachRibbonEffect(#Effet,
    NoeudID)
```

Description

Détache le ruban du noeud donné.

Arguments

#**Effet** L'effet à utiliser.
Cet effet a été créé avec CreateRibbonEffect() .

NoeudID Le nud à détacher du ruban.
Si le ruban n'a pas été attaché au nud, la fonction n'a aucun effet.
Pour obtenir un ID de noeud valide, utiliser NodeID() .

Valeur de retour

Aucune.

Voir aussi

CreateRibbonEffect() ,
AttachRibbonEffect()

OS Supportés

Tous

141.6 CreateLensFlareEffect

Syntaxe

```
CreateLensFlareEffect(#Effet,
    CameraID, NoeudID,
    TailleBurst, TailleHalo,
    HaloMatiereID,
    CercleMatiereID,
    BurstMatiereID)
```

Description

Crée un nouvel effet de lensflare (flamme ou flamboiement).

Arguments

#**Effet** Le numéro d'identification du nouvel effet.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

CameraID La caméra qui utilise le lensflare.
Cela n'affectera que cette caméra.
Pour obtenir un 'CameraID' valide, utiliser CameraID() .

NoeudID Le nud parent qui gère le lensflare.
Pour obtenir un 'NoeudID' valide , utiliser NodeID() .

TailleBurst Taille du Burst du lensflare, dans l'unité du monde.

TailleHalo La taille du halo, dans l'unité du monde.

HaloMatiereID Le matériau du halo. Pour obtenir un ID de matériau valide, utiliser MaterialID() .

CercleMatiereID Le matériau du cercle. Pour obtenir un ID de matériau valide, utiliser MaterialID() .

BurstMatiereID Le matériau de salve. Pour obtenir un ID de matériau valide, utiliser MaterialID() .

Valeur de retour

Renvoie une valeur non nulle si l'effet a été créé avec succès, zéro sinon.
Si #PB_Any a été utilisé comme paramètre #Effet alors le nouveau numéro généré est renvoyé en cas de succès.

Remarques

Un lensflare est toujours attaché à un noeud, et sera affiché automatiquement en fonction de la position du nud relativement à la vue de la caméra.

Voir aussi

FreeEffect() , LensFlareEffectColor()

OS Supportés

Tous

141.7 LensFlareEffectColor

Syntaxe

```
LensFlareEffectColor(#Effet,
    TypeCouleur, Couleur)
```

Description

Change la couleur d'un effet lensflare (flamme ou flamboiement).

Arguments

#**Effet** L'effet à utiliser.
Cet effet a été créé avec CreateLensFlareEffect() .

TypeCouleur

```
      #PB_LensFlare_HaloColor
     : Change la couleur du
    halo.
```

```
#PB_LensFlare_CircleColor:
 Change la couleur du
 cercle.
#PB_LensFlare_BurstColor :
 Change la couleur du
 burst.
```

Couleur La nouvelle couleur du lensflare. Une couleur valide peut être obtenue par RGB() .

Valeur de retour

Aucune.

Voir aussi

CreateLensFlareEffect()

OS Supportés

Tous

141.8 FreeEffect

Syntaxe

```
FreeEffect(#Effet)
```

Description

Supprime un effet.

Arguments

#**Effet** L'effet à supprimer. Si #PB_All est spécifié, tous les effets restants sont supprimés.

Valeur de retour

Aucune.

Remarques

Une fois l'effet supprimé, il ne peut plus être utilisé.
Tous les effets restants sont automatiquement supprimés lorsque le programme se termine.

Voir aussi

CreateRibbonEffect() , CreateCompositorEffect()

OS Supportés

Tous

141.9 IsEffect

Syntaxe

```
Resultat = IsEffect(#Effet)
```

Description

Teste si un effet est un effet valide et correctement initialisé.

Arguments

#**Effet** L'effet à utiliser.

Valeur de retour

Renvoie une valeur non nulle si l'effet est un effet valide, zéro sinon.

Remarques

Cette fonction peut être utilisée avec n'importe quelle valeur.
C'est une bonne façon de s'assurer qu'un effet est prêt à l'emploi.

Voir aussi

CreateRibbonEffect() ,
CreateCompositorEffect()

OS Supportés

Tous

141.10 HideEffect

Syntaxe

```
HideEffect(#Effet, Etat)
```

Description

Masque ou affiche un effet.

Arguments

#**Effet** L'effet à utiliser.

Etat #True : L'effet sera caché.
#False : L'effet sera affiché.

Valeur de retour

Aucune.

Voir aussi

CreateRibbonEffect() ,
CreateCompositorEffect()

OS Supportés

Tous

141.11 CompositorEffectParameter

Syntaxe

```
CompositorEffectParameter(#Effet,
    TechniqueID, PassID,
    NomEffet$, DataType, *Data)
```

Description

Régle en temps réel les paramètres d'un effet.

Arguments

#**Effet** L'effet à utiliser.
L'effet doit être un effet compositeur créé par CreateCompositorEffect() .

TechniqueID Le TechniqueID de l'effet.

PassID Le PassID de l'effet.

NomEffet$ Le nom de l'effet.

DataType Le type de données du paramètre.

Data Les données du paramètre.
Cela dépend du type de données spécifié.

Valeur de retour

Aucune.

Voir aussi

CreateCompositorEffect()

OS Supportés

Tous

141.12 RibbonEffectColor

Syntaxe

```
RibbonEffectColor(#Effet,
    IndexChaine, Couleur,
    CouleurFondu)
```

Description

Définit les couleurs du ruban.

Arguments

#**Effet** L'effet à utiliser.
Un ruban doit être créé avec CreateRibbonEffect() .

IndexChaine L'indice de la chaîne.
Le premier indice commence à partir de zéro.
Cet indice doit être inférieur au nombre de chaînes créées avec CreateRibbonEffect() .

Couleur La couleur du ruban.
Cette couleur peut être au format RGB() ou RGBA() .

CouleurFondu La couleur de fondu du ruban est la couleur finale lorsque le ruban s'estompe.
Cette couleur peut être au format RGB() ou RGBA() .

Valeur de retour

Aucune.

Voir aussi

CreateRibbonEffect()

OS Supportés

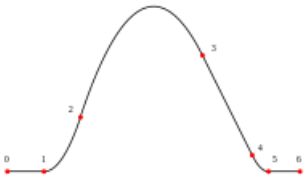
Tous

Chapitre 142

Spline

Généralités

Un ou une spline est une courbe qui contient un certain nombre de points qui sont interpolés de façon à donner une courbe très lisse. Le temps qu'il faut pour aller d'un point à un autre est toujours le même, indépendamment de la distance.
Une spline n'existe pas physiquement dans le monde 3D, c'est un objet virtuel qui peut être utilisé à des fins diverses, comme la création d'un chemin de poursuite (pathfinding), le lissage du mouvement d'un noeud (n'oubliez pas de consulter la bibliothèque NodeAnimation pour cela) et plus encore.
Spline est souvent prononcé à la française, sinon la prononciation anglaise est "splaïne" et quant à la traduction officelle en français, le mot "cerce" n'est quasiment jamais utilisé.
InitEngine3D() doit être appelé avec succès avant d'utiliser les fonctions splines.
Voir aussi l'article de wikipedia splines.



OS Supportés

Tous

142.1 CreateSpline

Syntaxe

```
Resultat =
    CreateSpline(#Spline)
```

Description

Crée une nouvelle courbe spline.

Arguments

#Spline Le numéro d'identification de la nouvelle spline.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si #PB_Any est utilisé pour le paramètre '#Spline' alors son numéro est renvoyé.

Remarques

Une spline n'existe pas physiquement dans le monde 3D, il s'agit d'un objet virtuel qui peut être utilisé à des différentes fins, la création d'un chemin de poursuite (path finding), le lissage du mouvement d'un noeud (n'oubliez pas de consulter NodeAnimation) et plus encore.
Pour calculer la position d'un point intermédiaire, voir ComputeSpline() .

Voir aussi

FreeSpline() , ComputeSpline() , AddSplinePoint()

OS Supportés

Tous

142.2 FreeSpline

Syntaxe

```
FreeSpline(#Spline)
```

Description

Libère une spline et toute la mémoire associée.

Arguments

#Spline La spline à utiliser.
Si #PB_All est spécifié, toutes les splines restantes sont libérées.

Valeur de retour

Aucune.

Remarques

La spline ne doit pas être utilisée après l'appel de cette fonction, (en utilisant son numéro, avec les autres fonctions de cette bibliothèque) à moins de la recréer. Toutes les splines restantes sont automatiquement libérées lorsque le programme se termine.

Voir aussi

CreateSpline()

OS Supportés

Tous

142.3 AddSplinePoint

Syntaxe

```
AddSplinePoint(#Spline, X, Y,
    Z)
```

Description

Ajoute un nouveau point à une spline.

Arguments

#Spline La spline à utiliser.

X, Y, Z La position du nouveau point dans le monde.

Valeur de retour

Aucune.

Remarques

Le temps qu'il faut pour aller d'un point à un autre est toujours le même, indépendamment de la distance.

Voir aussi

CreateSpline() , ComputeSpline()

OS Supportés

Tous

142.4 ClearSpline

Syntaxe

```
ClearSpline(#Spline)
```

Description

Efface une spline.

Arguments

#**Spline** La spline à effacer.

Valeur de retour

Aucune.

Remarques

Tous les points seront retirés de la spline.

Voir aussi

CreateSpline() , AddSplinePoint()

OS Supportés

Tous

142.5 CountSplinePoints

Syntaxe

```
Resultat =
    CountSplinePoints(#Spline)
```

Description

Renvoie le nombre de points d'une spline.

Arguments

#**Spline** La spline à utiliser.

Valeur de retour

Renvoie le nombre de points de la spline.

Voir aussi

CreateSpline() , AddSplinePoint()

OS Supportés

Tous

142.6 SplinePointX

Syntaxe

```
Resultat =
    SplinePointX(#Spline,
    IndexPoint)
```

Description

Renvoie la position 'X' d'un point d'une spline.

Arguments

#Spline La spline à utiliser.

IndexPoint L'index du point.
L'index du premier point est égal à zéro.
L'index doit être inférieur au résultat de CountSplinePoints() .

Valeur de retour

Renvoie la position 'X' du point.

Voir aussi

CreateSpline() , AddSplinePoint() , SplinePointY() , SplinePointZ()

OS Supportés

Tous

142.7 SplinePointY

Syntaxe

```
Resultat =
    SplinePointY(#Spline,
    IndexPoint)
```

Description

Renvoie la position 'Y' d'un point d'une spline.

Arguments

#Spline La spline à utiliser.

IndexPoint L'index du point.
L'index du premier point est égal à zéro.
L'index doit être inférieur au résultat de CountSplinePoints() .

Valeur de retour

Renvoie la position 'Y' du point.

Voir aussi

CreateSpline() , AddSplinePoint() , SplinePointX() , SplinePointZ()

OS Supportés

Tous

142.8 SplinePointZ

Syntaxe

```
Resultat =
    SplinePointZ(#Spline,
    IndexPoint)
```

Description

Renvoie la position 'Z' d'un point d'une spline.

Arguments

#**Spline** La spline à utiliser.

IndexPoint L'index du point.
L'index du premier point est égal à zéro. L'index doit être inférieur au résultat de CountSplinePoints() .

Valeur de retour

Renvoie la position 'Z' du point.

Voir aussi

CreateSpline() , AddSplinePoint() , SplinePointX() , SplinePointY()

OS Supportés

Tous

142.9 UpdateSplinePoint

Syntaxe

```
UpdateSplinePoint(#Spline,
    IndexPoint, X, Y, Z)
```

Description

Mise à jour d'un point d'une spline.

Arguments

#**Spline** La spline à utiliser.

IndexPoint L'index du point.
L'index du premier point est égal à zéro.
L'index doit être inférieur au résultat de CountSplinePoints() .

X, Y, Z La nouvelle position du point dans le monde.

Valeur de retour

Aucune.

Remarques

Chaque point sera également réparti autour de la spline, ce qui signifie que le temps est constant pour aller d'un point à un autre, indépendamment de la distance entre ces points.

Voir aussi

CreateSpline() , ComputeSpline()

OS Supportés

Tous

142.10 ComputeSpline

Syntaxe

```
ComputeSpline(#Spline,
    Deplacement.f)
```

Description

Calcule la nouvelle position d'un point après un déplacement sur une spline.

Arguments

#**Spline** La spline à utiliser.

Deplacement.f Le déplacement (décalage) dans la spline.
La valeur de décalage varie de 0.0 (début de la spline) à 1.0 (fin de la spline).

Valeur de retour

Aucune.

Remarques

Une fois que le point a été calculé, sa position est disponible avec SplineX() , SplineY() et SplineZ() .

Voir aussi

CreateSpline() , SplineX() , SplineY() , SplineZ()

OS Supportés

Tous

142.11 SplineX

Syntaxe

```
Resultat = SplineX(#Spline)
```

Description

Renvoie la position 'X' d'une spline dans le monde.

Arguments

#**Spline** La spline à utiliser.

Valeur de retour

Renvoie la position 'X' de la spline dans le monde après exécution de ComputeSpline() .

Voir aussi

ComputeSpline() , SplineY() , SplineZ()

OS Supportés

Tous

142.12 SplineY

Syntaxe

```
Resultat = SplineY(#Spline)
```

Description

Renvoie la position 'Y' d'une spline dans le monde.

Arguments

#**Spline** La spline à utiliser.

Valeur de retour

Renvoie la position 'Y' de la spline dans le monde après exécution de ComputeSpline() .

Voir aussi

ComputeSpline() , SplineX() , SplineZ()

OS Supportés

Tous

142.13 SplineZ

Syntaxe

```
Resultat = SplineZ(#Spline)
```

Description

Renvoie la position 'Z' d'une spline dans le monde.

Arguments

#**Spline** La spline à utiliser.

Valeur de retour

Renvoie la position 'Z' de la spline dans le monde après exécution de ComputeSpline() .

Voir aussi

ComputeSpline() , SplineX() , SplineY()

OS Supportés

Tous

Chapitre 143

Sprite

Généralités

Les 'Sprites' sont bien connus des joueurs sur ordinateur, ce sont des petites images parfois appelées 'brush' ou 'brushes' qui peuvent être affichées n'importe où sur l'écran. Les Sprites peuvent se déplacer au dessus d'un plan graphique en utilisant un mode transparent.
Encore mieux, PureBasic permet d'effectuer des effets en temps réel comme l'ombrage, l'alphablending, la coloration, le zoom, la rotation, tout cela en mode fenêtré ou en mode plein écran !
Après l'initialisation de l'écran et de l'environnement sprite via InitSprite() , vous pouvez commencer à ouvrir un plein écran ou un écran fenêtré .
DirectX 9 est utilisé pour gérer les sprites. Cela permet d'utiliser l'accélération matérielle si elle est disponible. Une version récente de DirectX 9 doit être installée (voir ici : DirectX 9 runtime installer).
Deux sous-systèmes supplémentaires, "OpenGL" et "DirectX11", sont également disponibles pour gérer les sprites.

OS Supportés

Tous

143.1 CatchSprite

Syntaxe

```
Resultat =
    CatchSprite(#Sprite,
    *AdresseMemoire [, Mode])
```

Description

Charge le sprite situé à l'adresse mémoire spécifiée.

Arguments

#**Sprite** Le numéro d'identification du nouveau sprite.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

***AdresseMemoire** L'adresse mémoire de l'image utilisée pour créer le sprite.

Mode (optionnel) Peut être une combinaison de (utiliser l'opérateur '||') :

```
#PB_Sprite_PixelCollision:
 Ajoute des informations
 spéciales pour gérer les
 collisions de pixels à
 travers
 SpritePixelCollision()
```

.

```
#PB_Sprite_AlphaBlending :
 Le sprite est créé avec
 un canal alpha
 (transparence),
 nécessaire pour utiliser
 DisplayTransparentSprite()
```

.

```
                              Le
 format de l'image doit
 le supporter (seulement
 PNG et TIFF pour
 l'instant).
```

Valeur de retour

Renvoie une valeur non nulle si le sprite a été créé avec succès, zéro sinon.

Remarques

Le sprite peut être au format BMP (non compressé = non RLE) ou dans un des formats supportés par la bibliothèque ImagePlugin .
UseJPEGImageDecoder()
UseJPEG2000ImageDecoder()
UsePNGImageDecoder()
UseTIFFImageDecoder()
UseTGAImageDecoder()
Avant de charger un sprite, un écran doit être ouvert à l'aide de la commande OpenScreen() ou OpenWindowedScreen() .
Un Sprite chargé peut être par la suite libéré par la fonction FreeSprite() .
La commande 'CatchSprite' est utile quand il est nécessaire d'inclure directement les sprites dans l'exécutable (en utilisant IncludeBinary). Cette méthode n'est pas idéale et devrait être évitée au maximum car en faisant ceci, les sprites prennent deux fois plus de mémoire (une première fois

dans l'exécutable, et une deuxième fois quand ils sont chargés).
Une autre utilisation souvent utilisée est de récupérer un sprite dans une banque de données compressée.
Le " ?" est un pointeur sur une étiquette. Plus d'informations sur les pointeurs et l'accès à la mémoire peuvent être trouvés ici .

Exemple

```
1  InitSprite()
2
3   OpenScreen(800,600,32,"Sprite")
4   DataSection
5     Pic:
6     IncludeBinary
     #PB_Compiler_Home
     +"Examples/Sources/Data/PureBasicLogo.bmp"
7   EndDataSection
8
9   CatchSprite(0,?Pic)
10
11  DisplaySprite(0, 210, 260)
12  FlipBuffers()
13  Delay(3000)
```

Voir aussi

CreateSprite() , LoadSprite()

OS Supportés

Tous

143.2 ClipSprite

Syntaxe

```
ClipSprite(#Sprite, X, Y,
    Largeur, Hauteur)
```

Description

Change la zone affichable d'un Sprite
Par exemple, si un sprite fait 100*100 pixels et que l'on définit une zone de 'clipping' X=10, Y=10, Largeur=20, Hauteur=20, alors seule la zone rectangulaire comprise entre ces coordonnées sera affichée. Le Sprite se comporte alors exactement comme un nouveau sprite de 20*20.

Arguments

#Sprite Le sprite à découper.

X, Y La position de départ de la découpe.

Largeur, Hauteur Les dimensions de la découpe, en pixels.

Valeur de retour

Aucune.

Remarques

Pour enlever la zone de découpe ('clipping') du Sprite, la constante #PB_Default doit être spécifiée dans 'X', 'Y', et/ou 'Largeur', 'Hauteur'.
Sur certaines anciennes cartes graphiques, ClipSprite() ne fonctionne pas correctement si la taille du sprite dépasse la taille de l'écran.

Exemple

```
1  InitSprite()
2
3  OpenScreen(800,600,32,"Sprite")
4
5  LoadSprite(0,#PB_Compiler_Home
       +"Examples/Sources/Data/PureBasicLogo.bmp")
6  LoadSprite(1,#PB_Compiler_Home
       +"Examples/Sources/Data/PureBasicLogo.bmp")
7
8  ClipSprite(1, 230, 0, 80, 68)
9
10 DisplaySprite(0, 210, 160)
11 DisplaySprite(1, 450, 260)
12 FlipBuffers()
13 Delay(3000)
```

Voir aussi

DisplaySprite() ,
DisplayTransparentSprite()

OS Supportés

Tous

143.3 CopySprite

Syntaxe

```
Resultat =
    CopySprite(#Sprite1,
    #Sprite2 [, Mode])
```

Description

Copie le #Sprite1 sur le #Sprite2.

Arguments

#Sprite1 Le numéro d'identifiant du sprite à copier.

#Sprite2 Un numéro d'identifiant du sprite de destination.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.
Si le #Sprite2 n'existe pas, il est créé.

Mode (optionnel) Peut être une combinaison de (utiliser '|')

```
#PB_Sprite_PixelCollision:
 Ajoute des informations
 spéciales pour gérer les
 collisions de pixels à
 travers
 SpritePixelCollision()
```

.

```
#PB_Sprite_AlphaBlending :
 Le sprite est créé avec
 un canal alpha
 (transparence),
 nécessaire pour utiliser
 DisplayTransparentSprite()
```

.

Valeur de retour

Renvoie une valeur non nulle si le sprite a été copié avec succès, zéro sinon.

Exemple

```
1  InitSprite()
2
3  OpenScreen(800,600,32,"Sprite")
4
5  LoadSprite(0,#PB_Compiler_Home
       +"Examples/Sources/Data/PureBasicLogo.bmp")
6
7   ;Copie du sprite 0 dans le
       sprite 1
8  CopySprite(0,1)
9
10 DisplaySprite(0, 210, 160)
11 DisplaySprite(1, 210, 260)
12 FlipBuffers()
13 Delay(3000)
```

Voir aussi

CreateSprite() , ClipSprite() , GrabSprite()

OS Supportés

Tous

143.4 CreateSprite

Syntaxe

```
Resultat =
    CreateSprite(#Sprite,
    Largeur, Hauteur [, Mode])
```

Description

Crée un nouveau Sprite.

Arguments

#**Sprite** Le numéro d'identifiant du nouveau sprite.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Largeur, Hauteur Les dimensions du nouveau sprite, en pixels.

Mode (optionnel) Peut être une combinaison de (utiliser '||')

```
#PB_Sprite_PixelCollision:
 Ajoute des informations
 spéciales pour gérer les
 collisions de pixels à
 travers
 SpritePixelCollision()
.
#PB_Sprite_AlphaBlending :
 Le sprite est créé avec
 un canal alpha
 (transparence),
 nécessaire pour utiliser
 DisplayTransparentSprite()
.
```

Valeur de retour

Renvoie une valeur non nulle si le sprite a été créé avec succès, zéro sinon.

Remarques

CreateSprite() doit être appelée dans le même thread où OpenScreen() a été appelé.

Exemple

```
1 InitSprite()
2
3 OpenScreen(800,600,32,"Sprite")
4
```

```
 ;Création du sprite
CreateSprite(0,200,200)
StartDrawing(SpriteOutput(0))
DrawingMode(#PB_2DDrawing_Gradient)
BackColor(RGB(255,255,255))
FrontColor(RGB(0,0,255))
CircularGradient(100, 100,
   100)
Circle(100, 100, 100)
CircularGradient(350, 100, 75)
Circle(300, 100, 100)
StopDrawing()

DisplaySprite(0, 210, 160)

FlipBuffers()
Delay(3000)
```

Voir aussi

SpriteOutput()

OS Supportés

Tous

143.5 DisplaySprite

Syntaxe

```
DisplaySprite(#Sprite, X, Y)
```

Description

Affiche un Sprite sur l'écran courant.

Arguments

#Sprite Le sprite à afficher.

X, Y Les coordonnées du sprite, en pixels, dans l' écran .

Valeur de retour

Aucune.

Remarques

Comme il n'y a pas de couleur transparente, cette fonction est plus rapide que DisplayTransparentSprite() .
Il est parfaitement autorisé d'afficher le sprite partiellement ou complètement hors de l'écran.

Exemple

```
InitSprite()

OpenScreen(800,600,32,"Sprite")

 ;Création du sprite
CreateSprite(0,200,200)
StartDrawing(SpriteOutput(0))
DrawingMode(#PB_2DDrawing_Gradient)
BackColor(RGB(255,255,255))
FrontColor(RGB(255,255,0))
CircularGradient(100, 100,
    100)
Circle(100, 100, 100)
CircularGradient(350, 100, 75)
Circle(300, 100, 100)
StopDrawing()

 ;Affichage du sprite
DisplaySprite(0, 210, 160)

FlipBuffers()
Delay(3000)
```

Voir aussi

CreateSprite() , DisplayTransparentSprite()

OS Supportés

Tous

143.6 DisplayTransparentSprite

Syntaxe

```
DisplayTransparentSprite(#Sprite,
    X, Y [, Intensite [,
    Couleur]])
```

Description

Affiche un Sprite avec une couleur de transparence sur l'écran courant.

Arguments

#Sprite Le sprite à afficher.

X, Y Les coordonnées du sprite, en pixels, dans l' écran .

Intensite (optionnel) Le niveau d'intensité d'affichage du sprite.
Les valeur valides sont comprises entre 0 (entièrement transparent) à 255 (complètement opaque).
La valeur par défaut est 255.

Couleur (optionnel) La couleur utilisée pour afficher le sprite.
Pour obtenir une couleur valide, utiliser RGB() .

Valeur de retour

Aucune.

Remarques

La couleur noire (RGB(0,0,0,)) est la couleur transparente par défaut, cette couleur ne sera donc pas affichée.
Elle peut être changée avec la commande TransparentSpriteColor() .
Il est parfaitement autorisé d'afficher le sprite partiellement ou complètement hors de l'écran.

Exemple

```
1  InitSprite()
2
3  OpenScreen(800,600,32,"Sprite")
4
5  LoadSprite(0,#PB_Compiler_Home
       +"Examples/Sources/Data/Geebee2.bmp")
6
7  DisplaySprite(0, 50, 160)
8  DisplayTransparentSprite(0,
       200, 160,128)
9  DisplayTransparentSprite(0,
       350,
       160,128,RGB(255,0,255))
10 DisplayTransparentSprite(0,
       500,
       160,255,RGB(255,0,255))
11 FlipBuffers()
12 Delay(3000)
```

Voir aussi

CreateSprite() , DisplaySprite()

OS Supportés

Tous

143.7 FreeSprite

Syntaxe

```
FreeSprite(#Sprite)
```

Description

Supprime un Sprite de la mémoire.

Arguments

#**Sprite** Le sprite à libérer.
Si #PB_All est spécifié, tous les sprites restants sont libérés.

Valeur de retour

Aucune.

Remarques

Tous les sprites restants sont automatiquement libérés quand le programme se termine.

Voir aussi

CreateSprite()

OS Supportés

Tous

143.8 GrabSprite

Syntaxe

```
Resultat =
    GrabSprite(#Sprite, X, Y,
    Largeur, Hauteur [, Mode])
```

Description

Capture une zone de l'écran et crée un nouveau Sprite avec son contenu.

Arguments

#**Sprite** Le numéro d'identifiant du nouveau sprite.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

X, Y Les coordonnées de la zone de capture, en pixels.

Largeur, Hauteur Les dimensions de la zone de capture, en pixels.

Mode (optionnel) Peut être une combinaison de (utiliser '||')

```
#PB_Sprite_PixelCollision:
 Ajoute des informations
 spéciales pour gérer les
 collisions de pixels à
 travers
 SpritePixelCollision()
```

.

```
#PB_Sprite_AlphaBlending :
 Le sprite est créé avec
 un canal alpha
 (transparence),
 nécessaire pour utiliser
 DisplayTransparentSprite()
```

.

Valeur de retour

Renvoie une valeur non nulle si la capture s'est faite avec succès, zéro sinon.

Remarques

GrabSprite() doit toujours être appelé en dehors d'un bloc StartDrawing() : StopDrawing() .

Exemple

```
1  InitSprite()
2
3  OpenScreen(800,600,32,"Sprite")
4
5   ;Création du sprite 0
6  CreateSprite(0,200,200)
7  StartDrawing(SpriteOutput(0))
8  Circle(100, 100, 100,
       RGB(255,255,0))
9  StopDrawing()
10
11  ;Affichage du sprite 0
12 DisplaySprite(0, 300, 200)
13
14  ;Capture d'une zone de
       l'écran et création du
       sprite 1
15 GrabSprite(1, 400, 300, 100,
       100)
16
17  ;Affichage du sprite 1
18 DisplaySprite(1, 500, 100)
19
20 FlipBuffers()
21
22 Delay(5000)
```

Voir aussi

CreateSprite() , ClipSprite() , DisplaySprite() , DisplayTransparentSprite()

OS Supportés

Tous

143.9 InitSprite

Syntaxe

```
Resultat = InitSprite()
```

Description

Initialise l'environnement nécessaire au fonctionnement des sprites. Vous devez placer cette fonction au début de votre code source si vous souhaitez utiliser les fonctions de la bibliothèque Sprite.

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle si l'initialisation s'est faite avec succès, zéro sinon.
Il est conseillé de tester cette valeur car en cas d'erreur d'initialisation, vous devez alors quitter le programme ou annuler tous les appels aux fonctions de la bibliothèque Sprite.
Cette fonction tente d'initialiser DirectX 9. Les causes probables d'un échec sont l'absence de DirectX ou une version trop ancienne.

OS Supportés

Tous

143.10 IsSprite

Syntaxe

```
Resultat = IsSprite(#Sprite)
```

Description

Teste si un Sprite est correctement initialisé.

Arguments

#**Sprite** Le sprite à tester.

Valeur de retour

Renvoie une valeur non nulle si le sprite est correctement initialisé, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.

Voir aussi

CreateSprite() , LoadSprite()

OS Supportés

Tous

143.11 LoadSprite

Syntaxe

```
Resultat =
    LoadSprite(#Sprite,
    Fichier$ [, Mode])
```

Description

Charge un Sprite en mémoire.

Arguments

#Sprite Le numéro d'identifiant du nouveau sprite.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Fichier$ Le chemin et le nom du fichier à utiliser pour créer le sprite.

Mode (optionnel) Peut être une combinaison de (utiliser '||')

```
#PB_Sprite_PixelCollision:
 Ajoute des informations
 spéciales pour gérer les
 collisions de pixels à
 travers
 SpritePixelCollision()
```

.

```
#PB_Sprite_AlphaBlending :
 Le sprite est créé avec
 un canal alpha
 (transparence, PNG et
 TIFF), nécessaire pour
 utiliser
 DisplayTransparentSprite()
```

.

Valeur de retour

Renvoie une valeur non nulle si le sprite a été chargé correctement, zéro sinon.

Remarques

Le fichier peut être une image au format BMP, ou dans un des formats supportés par la bibliothèque ImagePlugin .
UseJPEGImageDecoder()
UseJPEG2000ImageDecoder()
UsePNGImageDecoder()
UseTIFFImageDecoder()
UseTGAImageDecoder()
Avant de charger un sprite, un écran doit être ouvert à l'aide de la commande OpenScreen() ou OpenWindowedScreen() .
Les sprites ne devraient pas être plus grands que l'écran utilisé.
Utiliser de plus grands sprites peut éventuellement fonctionner sur certains matériels, et pas sur d'autres. Mieux vaut découper votre grand sprite en plusieurs petits.

Exemple

```
1 InitSprite()
2 OpenScreen(800,600,32,"Sprite")
3
4 LoadSprite(0,#PB_Compiler_Home
     +"Examples/Sources/Data/PureBasicLogo.bmp")
5
6 DisplaySprite(0, 200, 200)
7 FlipBuffers()
8 Delay(3000)
```

Voir aussi

CreateSprite() , DisplaySprite() , GrabSprite()

OS Supportés

Tous

143.12 SaveSprite

Syntaxe

```
Resultat =
    SaveSprite(#Sprite,
    Fichier$ [, ImagePlugin [,
    Options]])
```

Description

Enregistre un Sprite dans un fichier.

Arguments

#**Sprite** Le sprite à enregistrer.

Fichier$ Le chemin et le nom du fichier à utiliser.

ImagePlugin (optionnel) Peut prendre l'une des constantes suivantes :

```
#PB_ImagePlugin_BMP       :
  Enregistre l'image en
  BMP  (format par défaut)
#PB_ImagePlugin_JPEG      :
  Enregistre l'image en
  JPEG (la commande
  UseJPEGImageEncoder()
doit être utilisée)
#PB_ImagePlugin_JPEG2000 :
  Enregistre l'image en
  JPEG2000 (la commande
  UseJPEG2000ImageEncoder()
doit être utilisée)
#PB_ImagePlugin_PNG       :
  Enregistre l'image en
  PNG  (la commande
  UsePNGImageEncoder()
doit être utilisée)
```

Options (optionnel) Option de sauvegarde, dépendant du format d'enregistrement choisi.
Pour le moment, seul le réglage de la qualité de l'image est proposé.
Il est possible de régler la qualité d'une image à l'aide d'une valeur allant de 0 (plus mauvaise qualité) à 10 (qualité maximale). Seuls les plugins JPEG et JPEG2000 supportent cette option (la qualité par défaut est fixée à 7 si aucune option n'est spécifiée).

Valeur de retour

Renvoie une valeur non nulle si le sprite a été sauvegardé avec succès, zéro sinon.

Remarques

Par défaut, le format de sauvegarde est le BMP 24 bits.
Très utile avec la fonction GrabSprite() pour faire des copies d'écrans par exemple.

Exemple

```
1 InitSprite()
2 OpenScreen(800,600,32,"Sprite")
3 LoadSprite(0,#PB_Compiler_Home
    +"Examples/Sources/Data/PureBasicLogo.bmp")
4 DisplaySprite(0, 200, 200)
```

```
5
6   ;Capture de l'écran entier.
7   GrabSprite(1, 0, 0, 800, 600)
8
9   FlipBuffers()
10
11  ;Enregistrement de la
      capture dans un fichier
      BMP.
12  ;Vous pouvez choisir le
      chemin de l'enregistrement.
13  ;Ici c'est le dossier
      courant ou le dossier
      personnel de l'utilisateur.
14  SaveSprite(1, "Ecran.bmp")
15
16  Delay(3000)
```

Voir aussi

LoadSprite()

OS Supportés

Tous

143.13 SpriteCollision

Syntaxe

```
Resultat =
    SpriteCollision(#Sprite1,
    X1, Y1, #Sprite2, X2, Y2)
```

Description

Teste si deux Sprites se chevauchent.

Arguments

#Sprite1 Le premier sprite à tester.

X1, Y1 Coordonnées du premier sprite, en pixels.

#Sprite2 Le second sprite à tester.

X2, Y2 Coordonnées du second sprite, en pixels.

Valeur de retour

Renvoie une valeur non nulle si les sprites entrent en collision, zéro sinon.

Remarques

Cette fonction teste une zone rectangulaire, ce qui la rend rapide mais peu précise, mais cependant très utile dans des jeux d'arcade. Fonctionne aussi avec les sprites zoomés. Utiliser SpritePixelCollision() pour un test de collision précis.

Exemple

```
InitSprite()
OpenScreen(800,600,32,"Exemple OpenScreen")

 ;Sprite Flèche
CreateSprite(0,20,20,#PB_Sprite_PixelCollision)
StartDrawing(SpriteOutput(0))
DrawText(0,0,"-->",RGB(255,255,0),RGB(0,0,0))
StopDrawing()

 ;Sprite Cible
CreateSprite(1,100,100,#PB_Sprite_PixelCollision)
StartDrawing(SpriteOutput(1))
Circle(50,50,49,RGB(255,255,0))
Circle(50,50,40,RGB(0,255,0))
Circle(50,50,30,RGB(0,255,255))
Circle(50,50,20,RGB(0,0,255))
Circle(50,50,10,RGB(255,0,0))
StopDrawing()

 ;Sprite Boom
CreateSprite(2,100,100,#PB_Sprite_PixelCollision)
StartDrawing(SpriteOutput(2))
Circle(50,50,50,RGB(255,0,0))
DrawText(50-TextWidth("BOOM")/2,50-TextHeight("BOOM")/2,"BOOM",
StopDrawing()

 ;Coordonnée de départ en x de la flèche
x=100

 ;Boucle
While x < 800  ; Si la flèche manque la cible alors le programme se terminera quand x = 800

  ;La flèche avancera de 2 pixels vers la droite
  x=x+2

  ;Effacer complètement l'écran et afficher un fond noir
  ClearScreen(RGB(0,0,0))

  ;Affichage de la flèche
```

```
40    DisplaySprite(0, x, 250)
41    ;Affichage de la cible
42    DisplaySprite(1, 500, 200)
43
44    ;Détection de collision
       entre la flèche et la cible
45    If SpriteCollision(0, x,
       250, 1, 500, 200) <> 0
46      ;S'il y a collision alors
       on efface l'écran et on
       affiche BOOM
47      ClearScreen(RGB(0,0,0))
48      DisplaySprite(2, 500, 200)
49      ;On inverse les buffers
       pour rendre la scène
       visible à l'écran.
50      FlipBuffers()
51      Delay(2000)
52      ClearScreen(RGB(0,0,0))
53      End ; et puis on ferme le
       programme
54    EndIf
55
56
57    ;Maintenant que tout est
       calculé et affiché dans le
       buffer invisible,
58    ;On inverse les buffers
       pour rendre la scène
       visible à l'écran.
59    FlipBuffers()
60
61    Delay(10)
62
63  Wend
```

Voir aussi

SpritePixelCollision()

OS Supportés

Tous

143.14 SpriteDepth

Syntaxe

```
Resultat =
    SpriteDepth(#Sprite)
```

Description

Renvoie la profondeur de couleurs d'un Sprite.

Arguments

#**Sprite** Le sprite à utiliser.

Valeur de retour

La profondeur de couleurs du sprite, en bits.

Exemple

```
InitSprite()
OpenScreen(800,600,32,"Exemple OpenScreen 32 bits")

 ;Sprite Cible 32 bits car l'écran est 32 bits.
CreateSprite(0,100,100)
StartDrawing(SpriteOutput(0))
Circle(50,50,49,RGB(255,255,0))
Circle(50,50,40,RGB(0,255,0))
Circle(50,50,30,RGB(0,255,255))
Circle(50,50,20,RGB(0,0,255))
Circle(50,50,10,RGB(255,0,0))
StopDrawing()

 ;Affichage de la cible
DisplaySprite(0, 500, 200)

 ;Affichage d'un texte sur l'écran (sans sprite).
StartDrawing(ScreenOutput())
DrawText(50, 250, "Profondeur de couleurs du sprite : " + SpriteDepth(0) + " bits.")
StopDrawing()

 ;Maintenant que tout est calculé et affiché dans le buffer invisible,
 ;On inverse les buffers pour rendre la scène visible à l'écran.
FlipBuffers()

Delay(5000)
```

Voir aussi

SpriteWidth() , SpriteHeight()

OS Supportés

Tous

143.15 SpriteHeight

Syntaxe

```
Resultat =
    SpriteHeight(#Sprite)
```

Description

Renvoie la hauteur d'un Sprite.

Arguments

#**Sprite** Le sprite à utiliser.

Valeur de retour

La hauteur du sprite en pixels.

Exemple

```
InitSprite()
OpenScreen(800,600,32,"Exemple
    OpenScreen 32-bits")

 ;Sprite Cible 32-bits car
    l'écran est 32-bits.
CreateSprite(0,100,100)
StartDrawing(SpriteOutput(0))
Circle(50,50,49,RGB(255,255,0))
Circle(50,50,40,RGB(0,255,0))
Circle(50,50,30,RGB(0,255,255))
Circle(50,50,20,RGB(0,0,255))
Circle(50,50,10,RGB(255,0,0))
StopDrawing()

 ;Affichage de la cible
DisplaySprite(0, 500, 200)

 ;Affichage d'un texte sur
    l'écran (sans sprite).
StartDrawing(ScreenOutput())
DrawText(50, 250, "Hauteur du
    sprite : " +
    SpriteHeight(0) + "
    pixels.")
StopDrawing()

 ;Maintenant que tout est
    calculé et affiché dans le
    buffer invisible,
 ;On inverse les buffers pour
    rendre la scène visible à
    l'écran.
FlipBuffers()

Delay(5000)
```

Voir aussi

SpriteWidth() , SpriteDepth()

OS Supportés

Tous

143.16 SpriteID

Syntaxe

```
Resultat = SpriteID(#Sprite)
```

Description

Renvoie l'identifiant système d'un Sprite.

Arguments

#**Sprite** Le sprite à utiliser.

Valeur de retour

L'ID du sprite.
Ce résultat est parfois aussi appelé 'Handle'.
Voir le chapitre Numéros et Identifiants (Handles) pour plus d'informations.

Voir aussi

CreateSprite() , LoadSprite() , CatchSprite()

OS Supportés

Tous

143.17 SpritePixelCollision

Syntaxe

```
Resultat =
    SpritePixelCollision(#Sprite1,
    X1, Y1, #Sprite2, X2, Y2)
```

Description

Vérifie si deux Sprites se chevauchent.
#PB_Sprite_PixelCollision doit être spécifié lors de la création sprite.

Arguments

#Sprite1 Le premier sprite à tester.

X1, Y1 Les coordonnées du premier sprite, en pixels.

#Sprite2 Le second sprite à tester.

X2, Y2 Les coordonnées du second sprite, en pixels.

Valeur de retour

Renvoie une valeur non nulle si les deux sprites entrent en collision, zéro sinon.

Remarques

Cette fonction effectue une comparaison pixel par pixel sur les pixels transparents des deux sprites ce qui rend la routine très précise mais également relativement lente. Pour optimiser au maximum la comparaison, il convient d'enlever le plus possible la zone transparente autour du sprite, pour ne garder que sa dimension réelle. Fonctionne avec les sprites zoomés. Pour plus de rapidité, utiliser SpriteCollision() qui ne teste ques les bordures rectangulaires.
Attention, ne fonctionne pas avec les sprites ayant subi une rotation ou une transformation.

Exemple

```
1  InitSprite()
2  OpenScreen(800,600,32,"Exemple
       OpenScreen")
3
4   ;Sprite Balle
5  CreateSprite(0,30,30,#PB_Sprite_PixelCollision)
6  StartDrawing(SpriteOutput(0))
7  Circle(15,15,13,RGB(255,255,255))
8  Circle(15,15,10,RGB(128,128,128))
9  StopDrawing()
10
11  ;Sprite Cible
12 CreateSprite(1,100,100,#PB_Sprite_PixelCollision)
13 StartDrawing(SpriteOutput(1))
14 Circle(50,50,49,RGB(255,255,0))
15 Circle(50,50,40,RGB(0,255,0))
16 Circle(50,50,30,RGB(0,255,255))
17 Circle(50,50,20,RGB(0,0,255))
18 Circle(50,50,10,RGB(255,0,0))
19 StopDrawing()
20
21  ;Sprite Boom
22 CreateSprite(2,100,100,#PB_Sprite_PixelCollision)
23 StartDrawing(SpriteOutput(2))
```

```
Circle(50,50,50,RGB(255,0,0))
DrawText(50-TextWidth("BOOM")/2,50-TextHeight("BOOM")/2,"BOOM",
StopDrawing()

 ;Coordonnée de départ en x de la flèche
x=0

 ;Boucle
While x < 800  ; Si la balle manque la cible alors le programme se terminera quand x = 800

  ;La balle avancera de 4 pixels vers la droite
  x=x+4

  ;Effacer complètement l'écran et afficher un fond noir
  ClearScreen(RGB(0,0,0))

  ;Affichage de la flèche
  DisplaySprite(0, x, 240)
  ;Affichage de la cible
  DisplaySprite(1, 500, 200)

  ;Détection de collision entre la balle et la cible
  If SpritePixelCollision(0, x, 240, 1, 500, 200) <> 0
    ;S'il y a collision alors on efface l'écran et on affiche BOOM
    ClearScreen(RGB(0,0,0))
    DisplaySprite(2, 500, 200)
    ;On inverse les buffers pour rendre la scène visible à l'écran.
    FlipBuffers()
    Delay(2000)
    ClearScreen(RGB(0,0,0))
    End ; et puis on ferme le programme
  EndIf

  ;Maintenant que tout est calculé et affiché dans le buffer invisible,
  ;On inverse les buffers pour rendre la scène visible à l'écran.
  FlipBuffers()

  Delay(10)

Wend
```

Voir aussi

SpriteCollision()

OS Supportés

Tous

143.18 SpriteWidth

Syntaxe

```
Resultat =
    SpriteWidth(#Sprite)
```

Description

Renvoie la largeur d'un Sprite.

Arguments

#**Sprite** Le sprite à utiliser.

Valeur de retour

La largeur du sprite en pixels.

Exemple

```
1  InitSprite()
2  OpenScreen(800,600,32,"Exemple
       OpenScreen 32 bits")
3
4   ;Sprite Cible 32 bits car
       l'écran est 32 bits.
5  CreateSprite(0,100,100)
6  StartDrawing(SpriteOutput(0))
7  Circle(50,50,49,RGB(255,255,0))
8  Circle(50,50,40,RGB(0,255,0))
9  Circle(50,50,30,RGB(0,255,255))
10 Circle(50,50,20,RGB(0,0,255))
11 Circle(50,50,10,RGB(255,0,0))
12 StopDrawing()
13
14  ;Affichage de la cible
15 DisplaySprite(0, 500, 200)
16
17  ;Affichage d'un texte sur
       l'écran (sans sprite).
18 StartDrawing(ScreenOutput())
19 DrawText(50, 250, "Largeur du
       sprite : " +
       SpriteWidth(0) + "
       pixels.")
20 StopDrawing()
21
22  ;Maintenant que tout est
       calculé et affiché dans le
       buffer invisible,
```

```
23  ;On inverse les buffers pour
      rendre la scène visible à
      l'écran.
24  FlipBuffers()
25
26  Delay(5000)
```

Voir aussi

SpriteHeight() , SpriteDepth()

OS Supportés

Tous

143.19 SpriteOutput

Syntaxe

```
Resultat =
    SpriteOutput(#Sprite)
```

Description

Renvoie l'identifiant d'un Sprite nécessaire à la bibliothèque 2DDrawing .

Arguments

#**Sprite** Le sprite sur lequel on dessinera.

Valeur de retour

Renvoie l'OutputID du Sprite s'il est possible de dessiner dessus, zéro sinon.

Remarques

Il faut utiliser un bloc StartDrawing() / StopDrawing() pour effectuer les dessins 2D directement sur le Sprite.
La mémoire allouée avec SpriteOutput() est libérée avec StopDrawing() , OutputID ne peut donc pas être réutilisé.
SpriteOutput() doit être appelée dans le même thread où OpenScreen() a été appelé.

Exemple

```
1  ...
2  StartDrawing(SpriteOutput(#Sprite))
3    ; Dessin ici ...
4  StopDrawing()
```

OS Supportés

Tous

143.20 TransparentSpriteColor

Syntaxe

```
TransparentSpriteColor(#Sprite,
    Couleur)
```

Description

Change la couleur de transparence d'un Sprite (quand il est affiché avec DisplayTransparentSprite()).

Arguments

#Sprite Le sprite à utiliser.
Si #PB_Default est utilisé à la place de #Sprite, alors la couleur par défaut (Noir - RGB(0,0,0)) est remplacée par la couleur spécifiée qui devient la nouvelle couleur de transparence par défaut.
Elle sera utilisée par tous les sprites qui seront créés ou chargés ultérieurement (par LoadSprite() , CreateSprite() , etc...).

Couleur La nouvelle couleur de transparence.
RGB() peut être utilisé pour définir la valeur de 'Couleur'. Un tableau représentant les couleurs les plus communes est disponible ici .
La couleur de transparence par défaut est noire RGB(0,0,0).

Valeur de retour

Aucune.

Exemple

```
1  InitSprite()
2
3  OpenScreen(800,600,32,"Sprite")
4
5  LoadSprite(0,#PB_Compiler_Home
       +"Examples/Sources/Data/Geebee2.bmp")
6  DisplaySprite(0, 50, 160)
7  DisplayTransparentSprite(0,
       200, 160,128)
8  DisplayTransparentSprite(0,
       350,
       160,128,RGB(255,0,255))
9  DisplayTransparentSprite(0,
       500, 160,0,RGB(255,0,255))
10 DisplayTransparentSprite(0,
       500,
       160,255,RGB(255,0,255))
11
```

```
12 TransparentSpriteColor(0,
       RGB(255,0,255))
13 DisplaySprite(0, 50, 360)
14 DisplayTransparentSprite(0,
       200, 360,128)
15 DisplayTransparentSprite(0,
       350,
       360,128,RGB(255,0,255))
16 DisplayTransparentSprite(0,
       500, 360,0,RGB(255,0,255))
17 DisplayTransparentSprite(0,
       500,
       360,255,RGB(255,0,255))
18 FlipBuffers()
19 Delay(6000)
```

Voir aussi

DisplayTransparentSprite() , RGB()

OS Supportés

Tous

143.21 RotateSprite

Syntaxe

```
RotateSprite(#Sprite,
    Angle.f, Mode)
```

Description

Rotation d'un Sprite.

Arguments

#**Sprite** Le sprite à utiliser.

Angle.f Angle de rotation, en degrés, dans le sens des aiguilles d'une montre, de 0 à 360.

Mode 'Mode' peut prendre les valeurs suivantes :

```
#PB_Absolute: L'angle
  prend la valeur de
  'Angle'.
#PB_Relative: L'angle est
  ajouté à la valeur
  précédente.
```

Exemple

```
1 InitSprite()
2
3 OpenScreen(800,600,32,"Sprite")
```

```
4
5  LoadSprite(0,#PB_Compiler_Home
       +"Examples/Sources/Data/PureBasicLogo.bmp")
6
7  For i=0 To 2*360 Step 10
8    RotateSprite(0,
       i,#PB_Absolute)
9    DisplaySprite(0, 250, 260)
10   FlipBuffers()
11   Delay(100)
12 Next i
13
14 For i=2*360 To 0 Step -10
15   RotateSprite(0,
       i,#PB_Absolute)
16   DisplaySprite(0, 250, 260)
17   FlipBuffers()
18   Delay(100)
19 Next i
20
21 For i=0 To 360 Step 10
22   ClearScreen(RGB(0,0,0))
23   RotateSprite(0,
       i,#PB_Absolute)
24   DisplaySprite(0, 250, 260)
25   FlipBuffers()
26   Delay(100)
27 Next i
28
29 For i=0 To 360 Step 30
30   ClearScreen(RGB(0,0,0))
31   RotateSprite(0,
       i,#PB_Absolute)
32   DisplaySprite(0, 250, 260)
33   FlipBuffers()
34   Delay(100)
35 Next i
```

Voir aussi

TransformSprite() , ZoomSprite()

OS Supportés

Tous

143.22 SpriteBlendingMode

Syntaxe

```
SpriteBlendingMode(ModeSource,
    ModeDestination)
```

Description

Change la façon dont les couleurs des sprites sont mélangées avec le fond lors de l'utilisation de DisplayTransparentSprite() .

Arguments

ModeSource, ModeDestination Les modes source et destination peuvent prendre une valeur quelconque parmi les valeurs suivantes :

```
#PB_Sprite_BlendZero
#PB_Sprite_BlendOne
#PB_Sprite_BlendSourceColor
#PB_Sprite_BlendInvertSourceColor
#PB_Sprite_BlendDestinationColor
#PB_Sprite_BlendInvertDestinationColor
#PB_Sprite_BlendSourceAlpha
#PB_Sprite_BlendInvertSourceAlpha
#PB_Sprite_BlendDestinationAlpha
#PB_Sprite_BlendInvertDestinationAlpha
```

Les valeurs par défaut sont SpriteBlending-Mode(#PB_Sprite_BlendSourceAlpha, #PB_Sprite_BlendInvertSourceAlpha).

Valeur de retour

Aucune.

Exemple

```
;Structure contenant le nom et le mode de transparence
Structure ModeTransparence
  Nom$
  Numero.l
EndStructure

;Liste qui contient tous les "ModeSource"
NewList ModeSource.ModeTransparence()
AddElement(ModeSource())
ModeSource()\Nom$ = "#PB_Sprite_BlendZero"
ModeSource()\Numero =#PB_Sprite_BlendZero
AddElement(ModeSource())
ModeSource()\Nom$ = "#PB_Sprite_BlendOne"
ModeSource()\Numero =#PB_Sprite_BlendOne
AddElement(ModeSource())
ModeSource()\Nom$ = "#PB_Sprite_BlendSourceColor"
ModeSource()\Numero =#PB_Sprite_BlendSourceColor
AddElement(ModeSource())
ModeSource()\Nom$ = "#PB_Sprite_BlendInvertSourceColor"
ModeSource()\Numero =#PB_Sprite_BlendInvertSourceColor
```

```
21 AddElement(ModeSource())
22 ModeSource()\Nom$ =
       "#PB_Sprite_BlendDestinationColor"
23 ModeSource()\Numero
       =#PB_Sprite_BlendDestinationColor
24 AddElement(ModeSource())
25 ModeSource()\Nom$ =
       "#PB_Sprite_BlendInvertDestinationColor"
26 ModeSource()\Numero
       =#PB_Sprite_BlendInvertDestinationColor
27 AddElement(ModeSource())
28 ModeSource()\Nom$ =
       "#PB_Sprite_BlendSourceAlpha"
29 ModeSource()\Numero
       =#PB_Sprite_BlendSourceAlpha
30 AddElement(ModeSource())
31 ModeSource()\Nom$ =
       "#PB_Sprite_BlendInvertSourceAlpha"
32 ModeSource()\Numero
       =#PB_Sprite_BlendInvertSourceAlpha
33 AddElement(ModeSource())
34 ModeSource()\Nom$ =
       "#PB_Sprite_BlendDestinationAlpha"
35 ModeSource()\Numero
       =#PB_Sprite_BlendDestinationAlpha
36 AddElement(ModeSource())
37 ModeSource()\Nom$ =
       "PB_Sprite_BlendInvertDestinationAlpha"
38 ModeSource()\Numero
       =#PB_Sprite_BlendInvertDestinationAlpha
39
40  ;Liste qui contient tous les
       "ModeDestination"
41 NewList
       ModeDestination.ModeTransparence()
42 CopyList(ModeSource(),
       ModeDestination())
43
44  ;Initialisation des sprites
       et du clavier
45 InitSprite()
46 InitKeyboard()
47
48  ;Création de l'image de fond
       qui permettra de voir la
       transparence des sprites
49 CreateImage(1,800,600,32)
50 StartDrawing(ImageOutput(1))
51 Box(0,0,800,600,RGBA(255,255,255,255))
52 For i=1 To 1000
53   Ellipse(Random(800,1),
       Random(600,1),
       Random(8,1), Random(8,1) ,
       RGB(Random(255),Random(255),Random(255)))
54 Next i
55 StopDrawing()
56
57  ;Ouverture d'un écran
58 OpenScreen(800,600,32,"Sprite")
```

```
59
60  ;Sprite Cible.
61  CreateSprite(0,100,100,#PB_Sprite_AlphaBlending)
62  StartDrawing(SpriteOutput(0))
63  Circle(50,50,49,RGBA(255,255,0,255))
64  Circle(50,50,40,RGBA(0,255,0,128))
65  Circle(50,50,30,RGBA(0,255,255,128))
66  Circle(50,50,20,RGBA(0,0,255,128))
67  Circle(50,50,10,RGBA(255,0,0,64))
68  StopDrawing()
69
70  ;Pour toutes les
      combinaisons entre le
      ModeSource et le
      ModeDestination
71  ForEach ModeSource()
72    ForEach ModeDestination()
73
74      ;Le programme se termine
      si on appuie longtemps sur
      la touche Echap (ESC)
75      ExamineKeyboard()
76      If
      KeyboardPushed(#PB_Key_Escape)
77        End
78      EndIf
79
80      ;Affichage du fond et de
      quelques informations
81      StartDrawing(ScreenOutput())
82      DrawImage(ImageID(1),1,1)
83      DrawText(0, 0, "Appuyer 5
      secondes sur Echap pour
      quitter")
84      DrawText(50, 100,
      ModeSource()\Nom$+" /
      "+ModeDestination()\Nom$,
      RGB(255,255,255),
      RGB(255,0,0))
85      DrawText(50, 270,
      "Couleur transparente =
      Noire", RGB(255,255,0))
86      DrawText(50, 470,
      "Couleur transparente =
      Rouge", RGB(255,255,0))
87      StopDrawing()
88
89      ;TRANSPARENCE du sprite :
      Couleur noire par défaut
90      SpriteBlendingMode(ModeSource()\Numero,
      ModeDestination()\Numero)
91      DisplaySprite(0, 50, 160)
92
93      SpriteBlendingMode(ModeSource()\Numero,
      ModeDestination()\Numero)
94      TransparentSpriteColor(0,
      RGBA(0,0,0,128))
95
```

```
96      SpriteBlendingMode(ModeSource()\Numero,
        ModeDestination()\Numero)
97      DisplayTransparentSprite(0,
        200, 160,180)
98
99      SpriteBlendingMode(ModeSource()\Numero,
        ModeDestination()\Numero)
100     DisplayTransparentSprite(0,
        350,
        160,128,RGBA(255,0,0,180))
101
102     SpriteBlendingMode(ModeSource()\Numero,
        ModeDestination()\Numero)
103     DisplayTransparentSprite(0,
        500,
        160,0,RGBA(255,0,0,180))
104
105     SpriteBlendingMode(ModeSource()\Numero,
        ModeDestination()\Numero)
106     DisplayTransparentSprite(0,
        500,
        160,255,RGBA(255,0,0,180))
107
108     SpriteBlendingMode(ModeSource()\Numero,
        ModeDestination()\Numero)
109     DisplaySprite(0, 50, 360)
110     SpriteBlendingMode(ModeSource()\Numero,
        ModeDestination()\Numero)
111
112     ;TRANSPARENCE du sprite :
        Couleur rouge
113     TransparentSpriteColor(0,
        RGBA(255,0,0,180))
114
115     SpriteBlendingMode(ModeSource()\Numero,
        ModeDestination()\Numero)
116     DisplayTransparentSprite(0,
        200, 360,180)
117
118     SpriteBlendingMode(ModeSource()\Numero,
        ModeDestination()\Numero)
119     DisplayTransparentSprite(0,
        350, 360,180)
120
121     SpriteBlendingMode(ModeSource()\Numero,
        ModeDestination()\Numero)
122     DisplayTransparentSprite(0,
        500, 360,0)
123
124     SpriteBlendingMode(ModeSource()\Numero,
        ModeDestination()\Numero)
125     DisplayTransparentSprite(0,
        500, 360,255)
126
127     ;Affichage à l'écran
128     FlipBuffers()
129
130     ;Effacer le buffer
131     ClearScreen(RGBA(255,255,255,255))
```

```
132
133     ;Attendre 3 secondes
134     Delay(3000)
135
136   Next
137 Next
```

OS Supportés

Tous

143.23 SpriteQuality

Syntaxe

```
SpriteQuality(Qualite)
```

Description

Change la qualité de rendu d'un sprite.

Arguments

Qualite La qualité d'affichage du sprite.
Peut-être l'une des valeurs suivantes :

```
#PB_Sprite_NoFiltering
      : Pas de filtrage,
 plus rapide, mais laid
 après un zoom / rotation
 (par défaut).
#PB_Sprite_BilinearFiltering:
 Filtrage bilinéaire,
 plus lent mais propre
 lors de zoom / rotation.
```

Valeur de retour

Aucune.

Exemple

```
1
2
3  InitSprite()
4  OpenScreen(800,600,32,"Sprite")
5
6   ;Sprite Cible0
7  CreateSprite(0,100,100)
8  StartDrawing(SpriteOutput(0))
9  Circle(50,50,49,RGB(255,255,0))
10 Circle(50,50,40,RGB(0,255,0))
11 Circle(50,50,30,RGB(0,255,255))
12 Circle(50,50,20,RGB(0,0,255))
13 Circle(50,50,10,RGB(255,0,0))
14 StopDrawing()
15
```

```
 ;Sprite Cible1
CopySprite(0, 1)

 ;Boucle
For i=0 To 300

  ;Qualité normale pour Cible0
  SpriteQuality(#PB_Sprite_NoFiltering)
  ;Rotation du sprite pour
   apprécier la qualité
  RotateSprite(0, 0.5,
   #PB_Relative)
  ;Affichage du sprite
  DisplaySprite(0, 300, 160)

  ;Haute qualité pour Cible1
  SpriteQuality(#PB_Sprite_BilinearFiltering)
  ;Rotation du sprite pour
   apprécier la qualité
  RotateSprite(1, 0.5,
   #PB_Relative)
  ;Affichage du sprite
  DisplaySprite(1, 450, 160)

  ;Affichage à l'écran
  FlipBuffers()

  ;Effacer le buffer
  ClearScreen(RGB(0,0,0))

Next i
```

OS Supportés

Tous

143.24 TransformSprite

Syntaxe

```
TransformSprite(#Sprite, X1,
   Y1, [z1], X2, Y2, [Z2],
   X3, Y3, [Z3], X4, Y4, [Z4])
```

Description

Déforme un Sprite.

Arguments

X1, Y1 Coordonnées du premier point, en pixel.

X2, Y2 Coordonnées du deuxième point, en pixel.

X3, Y3 Coordonnées du troisième point, en pixel.

X4, Y4 Coordonnées du quatrième point, en pixel.

Z1, Z2, Z3, Z4 (optionnel) La coordonnée profondeur (en 'Z'), en pixel.

Valeur de retour

Aucune.

Remarques

Généralement utilisé pour effectuer des transformations en temps réel. Attention, comme un Sprite est composé de 2 vertex (2 triangles), la transformation peut donner un résultat très étrange...
Si l'un des paramètres optionnels 'Z' est spécifié, tous doivent être précisés.

```
;
; X1           X2
;    ---------
;    |       /|
;    |     /  |
;    |   /    |
;    | /      |
;    /--------
; x4           x3
;
```

Exemple

```
1  InitSprite()
2  OpenScreen(800,600,32,"Sprite")
3
4   ;Sprite Cible0
5  CreateSprite(0,100,100)
6  StartDrawing(SpriteOutput(0))
7  Circle(50,50,49,RGB(255,255,0))
8  Circle(50,50,40,RGB(0,255,0))
9  Circle(50,50,30,RGB(0,255,255))
10 Circle(50,50,20,RGB(0,0,255))
11 Circle(50,50,10,RGB(255,0,0))
12 StopDrawing()
13
14  ;Sprite Cible1
15 CopySprite(0, 1)
16
17  ;Boucle
18 For i=-100 To 100
19   ;Afficher Cible0 et Cible1
20   DisplaySprite(0, 200, 160)
21   DisplaySprite(1, 500, 160)
22
23   ;Déformation de Cible0 et
      Cible1
24   TransformSprite(0,i,i,100,0,100-i,100-i,0,100)
```

```
  TransformSprite(1,0,0,100-i,0+i,100-i,100-i,0,100)

  ;Affichage à l'écran
  FlipBuffers()

  ;Effacer le buffer
  ClearScreen(RGB(0,0,0))

  ;Attendre 100 msecondes
  Delay(100)

Next i
```

Voir aussi

ZoomSprite() , RotateSprite()

OS Supportés

Tous

143.25 ZoomSprite

Syntaxe

```
ZoomSprite(#Sprite, Largeur,
    Hauteur)
```

Description

Zoom un Sprite.

Arguments

Largeur Nouvelle largeur du sprite, en pixels.
Si #PB_Default est spécifié, la largeur initiale du sprite est rétablie.

Hauteur Nouvelle hauteur du sprite, en pixels.
Si #PB_Default est spécifié, la hauteur initiale du sprite est rétablie.

Valeur de retour

Aucune.

Exemple

```
InitSprite()
OpenScreen(800,600,32,"Sprite")

 ;Sprite Cible0
CreateSprite(0,100,100)
StartDrawing(SpriteOutput(0))
Circle(50,50,49,RGB(255,255,0))
```

```
Circle(50,50,40,RGB(0,255,0))
Circle(50,50,30,RGB(0,255,255))
Circle(50,50,20,RGB(0,0,255))
Circle(50,50,10,RGB(255,0,0))
StopDrawing()

 ;Sprite Cible1
CopySprite(0, 1)

 ;Boucle
For i=-100 To 100
  ;Afficher Cible0 et Cible1
  DisplaySprite(0, 200, 160)
  DisplaySprite(1, 500, 160)

  ;Zoom de Cible0 et Cible1
  ZoomSprite(0,100+i,100+i)
  ZoomSprite(1,100-i,100+i)

  ;Affichage à l'écran
  FlipBuffers()

  ;Effacer le buffer
  ClearScreen(RGB(0,0,0))

  ;Attendre 100 msecondes
  Delay(100)

Next i
```

Voir aussi

TransformSprite() , RotateSprite()

OS Supportés

Tous

Chapitre 144

StaticGeometry

Généralités

Une géométrie statique est une forme géométrique prédéfinie et pré-rendue, qui peut être très complexe et qui a toujours un rendu très rapide. Mais une fois créée, la forme géométrique ne peut plus être déplacée.
Elles peuvent être utilisées comme éléments de décors dans un paysage par exemple.
InitEngine3D() doit être appelé avec succès avant d'utiliser les fonctions géométriques statiques.

OS Supportés

Tous

144.1 FreeStaticGeometry

Syntaxe

```
FreeStaticGeometry(#Statique)
```

Description

Libére une forme géométrique statique.

Arguments

#Statique La forme à libérer.
Si #PB_All est spécifié, toutes les formes géométriqyes statiques restantes sont libérées.

Valeur de retour

Aucune.

Remarques

Toutes les formes géométriques statiques restantes sont automatiquement libérées lorsque le programme se termine.

Voir aussi

CreateStaticGeometry()

OS Supportés

Tous

144.2 IsStaticGeometry

Syntaxe

```
Resultat =
    IsStaticGeometry(#Statique)
```

Description

Teste si une forme géométrique statique est valide et correctement initialisée.

Arguments

#**Statique** La forme à utiliser.

Valeur de retour

Renvoie une valeur non nulle si #Statique est valide et initialisée, zéro sinon.

Remarques

C'est un bon moyen de vérifier que la forme géométrique statique est prête à l'emploi.

Voir aussi

CreateStaticGeometry()

OS Supportés

Tous

144.3 CreateStaticGeometry

Syntaxe

```
Resultat =
    CreateStaticGeometry(#Statique,
    Largeur, Hauteur,
    Longueur, ActiverOmbres)
```

Description

Crée une forme géométrique statique vide.

Arguments

#**Statique** Le numéro d'identification de la nouvelle forme géométrique statique. PB_Any # peut être utilisé pour générer automatiquement ce numéro.

Largeur, Hauteur, Longueur Dimensions (en unité monde) de la forme géométrique statique.

ActiverOmbres Active ou désactive les ombres dynamiques sur la forme.

```
#True : Ombres affichées.
#False: Ombres désactivées.
```

Valeur de retour

Renvoie une valeur non nulle si la forme géométrique statique a été créée avec succès, zéro sinon.
Si #PB_Any a été utilisé pour le paramètre #Statique alors le nombre généré est renvoyé en cas de succès.

Remarques

Si la nouvelle forme géométrique statique '#Statique' est chargée avec le même numéro, alors l'ancienne forme est automatiquement libérée lors de la création de la nouvelle.

Voir aussi

FreeStaticGeometry()

OS Supportés

Tous

144.4 AddStaticGeometryEntity

Syntaxe

```
AddStaticGeometryEntity(#Statique,
    EntitéID, X, Y, Z [,
    EchelleX, EchelleY,
    EchelleZ [, RotationX,
    RotationY, RotationZ [,
    RotationW, Mode]]])
```

Description

Ajoute une entité à une forme géométrique statique.

Arguments

#**Statique** La forme à utiliser.

EntitéID Le numéro d'identification EntitéID de l'entité à ajouter à la forme géométrique statique.

X, Y, Z La position de l'entité dans la forme géométrique statique.

EchelleX, EchelleY, EchelleZ (optionnel) Le facteur d'échelle à appliquer à l'entité.

RotationX, RotationY, RotationZ (optionnel) La rotation à appliquer à l'entité.

RotationW (optionnel) La rotation à appliquer à l'entité. (Seulement utilisé avec #PB_Orientation_Quaternion et #PB_Orientation_Direction).

Mode (optionnel) Le mode de rotation. Peut avoir l'une des valeurs suivantes :

- #PB_Orientation_PitchYawRoll: 'X' (tangage), 'Y' (lacet), 'Z' (roulis), appliqués dans cet ordre (par défaut).
- #PB_Orientation_Quaternion : 'X', 'Y', 'Z', 'W' pour les valeurs du quaternion
- #PB_Orientation_Direction : 'X', 'Y', 'Z' pour le vecteur de direction et 'W' pour la rotation (roulis).

Valeur de retour

Aucune.

Remarques

L'entité d'origine n'est pas modifiée par cette fonction et peut être libérée après l'ajout.
La même entité peut être ajoutée plusieurs fois.
Attention : Une fois que toutes les entités ont été ajoutées, la commande BuildStaticGeometry() doit être appelée pour générer la forme géométrique statique.

Voir aussi

CreateStaticGeometry() , BuildStaticGeometry()

OS Supportés

Tous

144.5 BuildStaticGeometry

Syntaxe

```
BuildStaticGeometry(#Statique)
```

Description

Construit une forme géométrique statique.

Arguments

#**Statique** La forme à utiliser.

Valeur de retour

Aucune.

Remarques

Une fois créée, une forme géométrique statique ne peut plus être modifiée.

Voir aussi

CreateStaticGeometry() , AddStaticGeometryEntity()

OS Supportés

Tous

Chapitre 145

StatusBar

Généralités

Une barre d'état ou 'StatusBar' est une barre horizontale située dans la partie inférieure d'une fenêtre et qui affiche des informations.
Cette barre est toujours visible et peut être séparée en plusieurs champs.

OS Supportés

Tous

145.1 AddStatusBarField

Syntaxe

```
AddStatusBarField(Largeur)
```

Description

Ajoute un champ à la barre d'état.

Arguments

Largeur La largeur du nouveau champ, en pixels.
Avec #PB_Ignore, le champ sera dimensionné automatiquement afin de remplir l'espace libre de la barre d'état.
Plusieurs champs peuvent avoir une largeur valant #PB_Ignore, dans ce cas l'espace libre sera partagé entre ces différents champs.

Valeur de retour

Aucune.

Remarques

Les nouveaux champs sont ajoutés à la suite (à droite) du dernier champ créé.

Les commandes suivantes permettent d'agir sur le contenu d'un champ de la barre d'état :
- StatusBarText()
- StatusBarImage()
- StatusBarProgress()

Exemple

```
If OpenWindow(0, 0, 0, 940, 60, "Barre d'état", #PB_Window_SystemMenu | #PB_Window_ScreenCentered | #PB_Window_SizeGadget)

ButtonGadget(1,10,3,300,25,"Ajouter un champ dans la barre d'état")
If CreateStatusBar(0, WindowID(0))
  AddStatusBarField(110)
EndIf

StatusBarText(0, 0, "Champ normal")

Repeat
  Event = WaitWindowEvent()

  Select Event

    Case #PB_Event_Gadget
      Select EventGadget()
        Case 1
          AddStatusBarField(110)
      EndSelect

  EndSelect
Until Event = #PB_Event_CloseWindow

EndIf
```

Voir aussi

StatusBarText() , StatusBarImage() , StatusBarProgress() , CreateStatusBar()

OS Supportés

Tous

145.2 CreateStatusBar

Syntaxe

```
Resultat =
    CreateStatusBar(#BarreEtat,
    FenetreID)
```

Description

Ajoute une barre d'état vide à une fenêtre donnée.

Arguments

#BarreEtat Le numéro d'identification de la barre d'état.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

FenetreID Le numéro d'identification de la fenêtre.
WindowID() peut être utilisée pour obtenir cette valeur.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Exemple

```
1   If OpenWindow(0, 0, 0, 940,
     50, "Barre d'état",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered
     | #PB_Window_SizeGadget)
2     If CreateStatusBar(0,
     WindowID(0))
3       AddStatusBarField(110)
4       AddStatusBarField(150)
5       AddStatusBarField(#PB_Ignore)
     ; dimensionne
     automatiquement ce champ
6       AddStatusBarField(300)
7     EndIf
8
9     StatusBarText(0, 0,
     "Champ normal")
10    StatusBarText(0, 1,
     "Champ sans bordure",
     #PB_StatusBar_BorderLess)
11    StatusBarText(0, 2,
     "Champ alignement droit",
     #PB_StatusBar_Right)
12    StatusBarText(0, 3,
     "Champ alignement centré",
     #PB_StatusBar_Center)
13
14    Repeat
15    Until WaitWindowEvent() =
     #PB_Event_CloseWindow
16  EndIf
```

Voir aussi

FreeStatusBar()

OS Supportés

Tous

145.3 FreeStatusBar

Syntaxe

```
FreeStatusBar(#BarreEtat)
```

Description

Efface et libère une barre d'état.

Arguments

#BarreEtat La barre d'état à libérer. Si #PB_All est spécifié, toutes les barres d'état sont libérées.

Valeur de retour

Aucune.

Remarques

Toutes les barres d'état restantes sont automatiquement libérées quand le programme se termine.

Voir aussi

CreateStatusBar()

OS Supportés

Tous

145.4 IsStatusBar

Syntaxe

```
Resultat =
    IsStatusBar(#BarreEtat)
```

Description

Teste si une barre d'état est correctement initialisée.

Arguments

#**BarreEtat** La barre d'état à utiliser.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage.

Voir aussi

CreateStatusBar()

OS Supportés

Tous

145.5 StatusBarImage

Syntaxe

```
StatusBarImage(#BarreEtat,
    Champ, ImageID [,
    Apparence])
```

Description

Ajoute ou remplace une image d'un champ.

Arguments

#**BarreEtat** La barre d'état à utiliser.

Champ Le champ à utiliser.
Le premier champ commence à zéro.

ImageID L'identifiant de l'image à utiliser.
Cet identifiant est aisément obtenu avec ImageID() .

Apparence (optionnel) Peut être une combinaison des valeurs suivantes (combinaison avec l'opérateur '||' (OR)) :

```
#PB_StatusBar_Raised     :
 Bordure en relief élevé
 (Sauf OS X et Windows
 avec thème graphique)
#PB_StatusBar_BorderLess:
 Sans bordure
#PB_StatusBar_Center     :
 Texte centré
#PB_StatusBar_Right      :
 Texte aligné à droite
```

Valeur de retour

Aucune.

Exemple

```
If OpenWindow(0, 0, 0, 340, 50, "Barre d'état et Image", #PB_Window_SystemMenu | #PB_Window_ScreenCentered | #PB_Window_SizeGadget)
  If CreateStatusBar(0, WindowID(0))
    AddStatusBarField(120)
    AddStatusBarField(170)
  EndIf

  UsePNGImageDecoder()

  If LoadImage(0, #PB_Compiler_Home + "Examples/Sources/Data/world.png")
    ;Changer le Chemin/NomFichier vers votre propre icône de 16x16 pixels.
    StatusBarImage(0, 0, ImageID(0))
  EndIf

  If LoadImage(1, #PB_Compiler_Home + "Examples/Sources/Data/Drive.bmp")
    ; Changer le Chemin/NomFichier vers votre propre icône de 16x16 pixels.
    StatusBarImage(0, 1, ImageID(1), #PB_StatusBar_Right)
  EndIf

  Repeat
  Until WaitWindowEvent() = #PB_Event_CloseWindow
EndIf

; Avez-vous remarqué la transparence des deux icônes ?
```

Voir aussi

StatusBarText() , StatusBarProgress() , CreateStatusBar()

OS Supportés

Tous

145.6 StatusBarID

Syntaxe

```
Resultat =
    StatusBarID(#BarreEtat)
```

Description

Renvoie l'identifiant système d'une barre d'état.

Arguments

#**BarreEtat** La barre d'état à utiliser.

Valeur de retour

Renvoie l'identifiant système unique de la barre d'état.
Cet identifiant est aussi connu sous le nom de 'Handle'.
Voir le chapitre Numéros et Identifiants (Handles) pour plus d'informations.

OS Supportés

Tous

145.7 StatusBarProgress

Syntaxe

```
StatusBarProgress(#BarreEtat,
    Champ, Valeur [, Apparence
    [, Min, Max]])
```

Description

Affiche une barre de progression dans un champ.

Arguments

#**BarreEtat** La barre d'état à utiliser.

Champ Le champ à utiliser.
Le premier champ commence à zéro.

Valeur La progression à afficher, entre un minimum et un maximum.
La mise à jour de la barre de progression se fait en appellant cette fonction de nouveau avec une nouvelle valeur de progression.

Apparence (optionnel) Peut être une combinaison des valeurs suivantes (combinaison avec l'opérateur '||' (OR)) :

```
#PB_StatusBar_Raised     :
 Bordure en relief élevé
 (sauf OS X et Windows
 avec thème graphique)
#PB_StatusBar_BorderLess:
 Sans bordure
```

Min, Max (optionnel) Les bornes minimum et maximum autorisées.
S'ils sont omis, ou si #PB_Ignore est spécifié, alors les valeurs précédentes seront utilisées.
Les valeurs par défaut de 'Min' et 'Max' pour les champs nouvellement créés sont respectivement 0 et 100.

Valeur de retour

Aucune.

Exemple

```
1  If OpenWindow(0, 0, 0, 360,
    50, "Barre d'état avec une
    barre de progression",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered
    | #PB_Window_SizeGadget)
2    If CreateStatusBar(0,
    WindowID(0))
3      AddStatusBarField(170)
4      AddStatusBarField(150)
5    EndIf
6
7    StatusBarText(0, 0,
    "Progression en cours...")
8    StatusBarProgress(0, 1, 0)
9    Delay(500)
10   StatusBarProgress(0, 1,
    25)
11   Delay(500)
12   StatusBarProgress(0, 1,
    50)
13   Delay(500)
14   StatusBarProgress(0, 1,
    75)
15   Delay(500)
16   StatusBarProgress(0, 1,
    100)
17   StatusBarText(0, 0,
    "Terminé !")
18
19   Repeat
20   Until WaitWindowEvent() =
    #PB_Event_CloseWindow
```

```
21  EndIf
```

Voir aussi

StatusBarText() , StatusBarImage() , CreateStatusBar()

OS Supportés

Tous

145.8 StatusBarText

Syntaxe

```
StatusBarText(#BarreEtat,
    Champ, Texte$ [,
    Apparence])
```

Description

Change le texte d'un champ.

Arguments

#BarreEtat La barre d'état à utiliser.

Champ Le champ à utiliser.
Le premier champ commence à zéro.

Texte$ Le texte à afficher.

Apparence (optionnel) Peut être une combinaison des valeurs suivantes (combinaison avec l'opérateur '||' (OR)) :

```
#PB_StatusBar_Raised     :
 Bordure en relief élevé
 (Sauf OS X et Windows
 avec thème graphique)
#PB_StatusBar_BorderLess:
 Sans bordure
#PB_StatusBar_Center     :
 Texte centré
#PB_StatusBar_Right      :
 Texte aligné à droite
```

Valeur de retour

Aucune.

Voir aussi

StatusBarImage() , StatusBarProgress() , CreateStatusBar()

OS Supportés

Tous

145.9 StatusBarHeight

Syntaxe

```
Resultat =
    StatusBarHeight(#BarreEtat)
```

Description

Renvoie la hauteur d'une barre d'état.

Arguments

#**BarreEtat** La barre d'état à utiliser.

Valeur de retour

Renvoie la hauteur en pixels de la barre d'état.

Voir aussi

CreateStatusBar()

OS Supportés

Tous

Chapitre 146

String

Généralités

La bibliothèque String vous permet d'effectuer les opérations courantes sur les chaînes de caractères. Par défaut, les Chaînes de caractères sont vues comme des chaînes unicodes.

OS Supportés

Tous

146.1 Asc

Syntaxe

```
Resultat = Asc(Chaine$)
```

Description

Renvoie la valeur du premier caractère d'une chaîne.

Arguments

Chaine$ La chaîne de caractères à utiliser.

Valeur de retour

Renvoie la valeur ASCII du premier caractère de la chaîne ou sa valeur Unicode si sa valeur est supérieur à 255.
Comme le compilateur est unicode , il renverra une valeur de caractère unicode, sur deux octets.

Remarques

Vous trouverez une table ASCII ici .
Il est également possible d'obtenir la valeur d'un caractère (et non d'une chaîne) en le plaçant entre des apostrophes directement, sans utiliser la fonction ASC.

Exemple

```
1  Debug "Veuillez vérifier
     que votre IDE est bien en
     UTF8: Fichier\Format du
     fichier\Encodage : Utf8
     doit être coché."
2  Debug '!'       ; Affiche 33
3  Debug Asc("!") ; Affiche 33
     (sur deux octets) car les
     valeur ascii sont
     intégrées dans unincode
4  Debug Asc(" ") ; Affiche la
     valeur 8364 = 20AC en
     hexadecimal
5
6  Unicode$="   "
7  Debug Asc(Unicode$) ;
     Affiche la valeur 8364
8  ShowMemoryViewer(@Unicode$,StringByteLength(Unicode$)
     + SizeOf(CHARACTER)) ;
     Affiche AC 20 AC 20 20 00
     00 00 (il est normal que
     AC et 20 soient "inversés"
     en mémoire)
```

Voir aussi

Chr()

OS Supportés

Tous

146.2 Bin

Syntaxe

```
Resultat\$ = Bin(Valeur.q [,
    Type])
```

Description

Convertit un nombre entier en un nombre binaire.

Arguments

Valeur.q Un nombre entier de type 'quad' (8 octets).

Type (optionnel) Permet de traiter la 'Valeur' comme un type différent :

```
#PB_Quad    : La valeur
  sera traitée comme un
  'quad' (0 à
  18446744073709551615)
  (Par défaut)
```

```
#PB_Byte    : La valeur
 sera traitée comme un
 'byte' (0 à 255)
#PB_Ascii   : La valeur
 sera traitée comme un
 'caractère ascii' (0 à
 255)
#PB_Word    : La valeur
 sera traitée comme un
 'word' (0 à 65535)
#PB_Unicode: La valeur
 sera traitée comme un
 'caractère unicode' (0 à
 65535)
#PB_Long    : La valeur
 sera traitée comme un
 'long' (0 à 4294967296)
```

Valeur de retour

Convertit une valeur numérique de type 'quad' en chaîne en utilisant la représentation binaire.

Exemple

```
1 Debug Bin(32) ; Affichera
   "100000"
2 Debug
   Bin(1024*1024*1024*1024) ;
   Affichera
   "10000000000000000000000000000000000000000000"
```

Note : Si des zéros supplémentaires sont nécessaires pour formater correctement le texte, il est possible d'utiliser RSet() .

Exemple

```
1 Debug RSet(Bin(32), 16,
   "0") ; Affichera
   "0000000000100000"
```

Voir aussi

Str() , Val() , Hex()

OS Supportés

Tous

146.3 Chr

Syntaxe

```
Resultat\$ =
    Chr(ValeurCaractere)
```

Description

Renvoie une chaîne obtenue avec la valeur du caractère donné.

Arguments

ValeurCaractere La valeur du caractère.

Valeur de retour

Renvoie une chaîne obtenue avec la valeur du caractère donné.

Remarques

Un tableau avec toutes les valeurs Ascii et leur code associé peut être consulté ici . Cette commande fonctionne en Unicode , qui renvoie les caractères associés à la valeur donnée.

Exemple

```
1  Debug Chr(33)    ; Affiche le caractère "!"
2  Debug Chr(8364)  ; Affiche le caractère " "
3  Debug Chr($BC)   ; Affiche le caractère "¼"
4  Debug Chr($BD)   ; Affiche le caractère "½"
```

Voir aussi

Asc() , Val()

OS Supportés

Tous

146.4 CountString

Syntaxe

```
Resultat =
    CountString(Chaine$,
    ChaineATrouver$)
```

Description

Renvoie le nombre d'occurences d'une sous-chaîne dans une chaîne de caractères.

Arguments

Chaine$ La chaîne de caractères à utiliser.

ChaineATrouver$ La chaîne de caractères à trouver.

Valeur de retour

Renvoie le nombre d'occurences de la 'ChaineATrouver$' dans la 'Chaine$'.

Remarques

Cette fonction ne raisonne pas sur des mots, donc si la 'ChaineATrouver$' se trouve à l'intérieur d'un mot, elle sera comptabilisée quand même, comme le prouve l'exemple suivant.

Exemple

```
1   Resultat =
      CountString("Combien de
      'ien' contient le mot
      chien ?", "ien") ; Le
      résultat sera 4
```

Voir aussi

FindString() , InsertString() , RemoveString() , ReplaceString() , ReverseString() , StringByteLength() , StringField()

OS Supportés

Tous

146.5 EscapeString

Syntaxe

```
Resultat\$ =
    EscapeString(Chaine$ [,
    Mode])
```

Description

Transforme une chaîne standard en chaîne avec des séquences d'échappement.

Arguments

String$ La chaîne à transformer.

Mode (optionnel)

```
      #PB_String_EscapeInternal:
    échappe une chaîne en
    utilisant le format de
    PureBasic (Par défaut).
                              (voir
    Règles générales
)
   #PB_String_EscapeXML      :
    échappe une chaîne en
    utilisant le format XML.
    Utile pour insérer une
    chaîne dans un arbre XML.
```

Valeur de retour

Renvoie la version échappée de la chaîne.

Remarques

La fonction UnescapeString() peut être utilisée pour faire l'opération inverse.
Rappel du format PureBasic :

```
\a: bip
 Chr(7)
\b: retour arrière
 Chr(8)
\f: saut de page
 Chr(12)
\n: retour à la ligne
 Chr(10)
\r: retour chariot
 Chr(13)
\t: tabulation horizontale
 Chr(9)
\v: tabulation verticale
 Chr(11)
\": double quote
 Chr(34)
\\: antislash
 Chr(92)
```

Attention : Sous Windows, \t ne fonctionne pas avec les fonctions graphiques des bibliothèques 2DDrawing et VectorDrawing.

Exemple

```
1  Debug
    EscapeString("Test="+Chr(34)+"Hello"+Chr(34)+".")
    ; Affiche "Test=\"Hello\"."
2  Debug
    EscapeString("<item>Hello</item>",
    #PB_String_EscapeXML) ;
    Affiche
    "&lt;item&gt;Hello&lt;/item&gt;"
```

Voir aussi

UnescapeString()

OS Supportés

Tous

146.6 FindString

Syntaxe

```
Resultat =
    FindString(Chaine$,
    ChaineCherchee$ [,
    PositionDepart] [, Mode]])
```

Description

Cherche une sous-chaîne dans une chaîne de caractères.

Arguments

Chaine$ La chaîne de caractères à utiliser.

ChaineCherchee$ La chaîne de caractères à trouver.

PositionDepart (optionnel) Position du caractère à partir de laquelle la recherche commence.
L'indice du premier caractère valide est 1.
Si ce paramètre n'est pas spécifié, la chaîne entière est recherchée.

Mode (optionnel) Peut être une des valeurs suivantes :

```
#PB_String_CaseSensitive:
 Recherche sensible à la
 casse (A <> a) (par
 défaut).
#PB_String_NoCase       :
 Recherche insensible à
 la casse (A = a).
```

Valeur de retour

Si 'ChaineCherchee$' est trouvée, sa position est renvoyée, en nombre de caractères, commençant à 1, sinon zéro est renvoyé.

Exemple

```
1  Position =
    FindString("PureBasic",
    "Bas") ; 'Position'
    recevra la valeur 5
```

Voir aussi

CountString() , InsertString() , RemoveString() , ReplaceString() , ReverseString() , StringByteLength() , StringField()

OS Supportés

Tous

146.7 Hex

Syntaxe

```
Resultat\$ = Hex(Valeur.q [,
    Type])
```

Description

Convertit un entier en une valeur hexadécimale.

Arguments

Valeur.q Un entier de type 'quad'.

Type (optionnel) #PB_Quad : La valeur sera traitée comme un 'quad' (0 à 18446744073709551615) (Par défaut)
#PB_Byte : La valeur est un octet (8-bit) allant de 0 à 255
#PB_Ascii : La valeur est un octet (8-bit) allant de 0 à 255
#PB_Word : La valeur est un word (16-bit) allant de 0 à 65535
#PB_Unicode: La valeur est un word (16-bit) allant de 0 à 65535
#PB_Long : La valeur est un long (32-bit) allant de 0 à 4294967295

Valeur de retour

Convertit une valeur numérique de type 'quad' en une chaîne de caractères au format hexadécimal.
La valeur renvoyée est toujours positive.

Exemple

```
1  Debug Hex(-1)
2  Debug Hex(-1, #PB_Byte)
3  Debug Hex(-1, #PB_Word)
4  Debug Hex(-1, #PB_Long)
5  Debug Hex(-1, #PB_Quad)
     ; Quad est la valeur par
     défaut.
6  Debug Hex(12) ; Affichera
     "C"
7  Debug Hex(1234567890) ;
     Affichera "499602D2"
```

Note : Si des zéros supplémentaires sont nécessaires pour formater correctement le texte, il est possible d'utiliser RSet() :

```
1  Debug RSet(Hex(12), 4, "0")
     ; Affichera "000C"
```

Voir aussi

Bin() , Str() , Val()

OS Supportés

Tous

146.8 InsertString

Syntaxe

```
Resultat\$ =
    InsertString(Chaine$,
    ChaineAInserer$, Position)
```

Description

Insère une sous-chaîne dans une chaîne de caractères.

Arguments

Chaine$ La chaîne de caractères à utiliser.

ChaineAInserer$ La chaîne de caractères à inserer.

Position La position d'insertion (commence à 1).

Valeur de retour

Insère la 'ChaineAInserer$' dans la 'Chaine$' à la 'Position' spécifiée.

Exemple

```
1  Debug InsertString("Hello
     !", "World", 7)  ;
     Affichera "Hello World!"
2  Debug InsertString("Hello
     !", "World", 1)  ;
     Affichera "WorldHello !"
```

Voir aussi

CountString() , FindString() , RemoveString() , ReplaceString() , ReverseString() , StringByteLength() , StringField()

OS Supportés

Tous

146.9 LCase

Syntaxe

```
Resultat\$ = LCase(Chaine$)
```

Description

Convertit une chaîne en minuscules

Arguments

Chaine$ La chaîne de caractères à convertir.

Valeur de retour

Renvoie la chaîne convertie en lettres minuscules (quand c'est possible).

Remarques

Cette commande accepte les caractères accentués. Si un 'É' majuscule est trouvé, il sera transformé en un 'é' minuscule.

Exemple

```
1  Debug LCase("PureBasic, LA
     PUISSANCE À L'ÉTAT PUR") ;
     Affichera "purebasic, la
     puissance à l'état pur"
```

Voir aussi

UCase()

OS Supportés

Tous

146.10 Left

Syntaxe

```
Resultat\$ = Left(Chaine$,
    Longueur)
```

Description

Renvoie la partie gauche d'une chaîne.

Arguments

Chaine$ La chaîne de caractères à utiliser.

Longueur Le nombre de caractères à renvoyer.

Valeur de retour

Renvoie les caractères de la partie gauche de la 'Chaine$'.

Remarques

Cette fonction renverra toute la chaîne si la longueur est supérieure à la taille de la chaîne de caractères.

Exemple

```
1   Debug Left("PureBasic", 4)
      ; Affichera "Pure"
```

Voir aussi

Len() , Mid() , Right()

OS Supportés

Tous

146.11 Len

Syntaxe

```
Resultat = Len(Chaine$)
```

Description

Renvoie la longueur d'une chaîne.

Arguments

Chaine$ La chaîne de caractères à utiliser.

Valeur de retour

Renvoie la longueur en caractères de la chaîne.

Exemple

```
a = Len("PureBasic") ; a recevra 9
```

Voir aussi

Left() , Mid() , Right()

OS Supportés

Tous

146.12 LSet

Syntaxe

```
Resultat\$ = LSet(Chaine$,
    Longueur [, Caractere$])
```

Description

Ajuste la longueur d'une chaîne de caractères en y ajoutant des caractères en fin de chaîne si nécessaire pour atteindre la longueur spécifiée.

Arguments

Chaine$ La chaîne de caractères à utiliser.

Longueur Le nombre de caractères à ajouter.

Caractere$ (optionnel) Peut être utilisé en lieu et place du caractère d'espacement qui est la valeur par défaut.

Valeur de retour

Renvoie une chaîne en y ajoutant des caractères en fin de chaîne si nécessaire pour atteindre la longueur spécifiée.

Remarques

Si la chaîne est plus longue que la longueur spécifiée, elle sera tronquée à partir de la fin de la chaîne.

Exemple

```
1  Resultat$ = LSet("L", 8)
                ; Resultat$ sera:
    "L       "
2  Resultat$ = LSet("L", 8,
    "-")        ; Resultat$ sera:
    "L-------"
3  Resultat$ =
    LSet("LongString", 4) ;
    Resultat$ sera: "Long"
```

Voir aussi

RSet()

OS Supportés

Tous

146.13 LTrim

Syntaxe

```
Resultat\$ = LTrim(Chaine$ [,
    Caractere$])
```

Description

Supprime tous les espaces situés au début d'une chaîne.

Arguments

Chaine$ La chaîne de caractères à utiliser.

Caractere$ (optionnel) Supprimera tous les 'Caractere$' situés au début de la chaîne.
'Caractere$' doit être une chaîne contenant seulement un caractère.

Valeur de retour

Renvoie une chaîne dont tous les espaces situés au début de la chaîne ont été supprimés.

Exemple

```
1  Debug LTrim("   PureBasic")
                ; Affichera
    "PureBasic".
2  Debug LTrim("!!Salut le
    Monde !!", "!") ;
    Affichera "Salut le Monde
    !!"
```

Voir aussi

RTrim() , Trim()

OS Supportés

Tous

146.14 Mid

Syntaxe

```
Resultat\$ = Mid(Chaine$,
    PositionDepart [,
    Longueur])
```

Description

Extrait une sous-chaîne d'une chaîne.

Arguments

Chaine$ La chaîne de caractères à utiliser.

PositionDepart La position de départ (commence à 1).

Longueur (optionnel) Précise combien de caractères seront extraits.
Si le paramètre 'Longueur' est omis, les caractères sont extraits jusqu'à la fin de la chaîne 'Chaine$'.

Valeur de retour

Renvoie une sous-chaîne de longueur 'Longueur', à partir du caractère situé à la position 'PositionDepart'.

Exemple

```
1  Debug Mid("Hello", 2) ;
     Affichera "ello"
2  Debug Mid("Hello", 2, 1) ;
     Affichera "e"
```

Voir aussi

Left() , Len() , Right()

OS Supportés

Tous

146.15 RemoveString

Syntaxe

```
Resultat\$ =
    RemoveString(Chaine$,
    ChaineASupprimer$ [, Mode
    [, Position [,
    NbOccurences]]])
```

Description

Supprime une sous-chaîne dans une chaîne.

Arguments

Chaine$ La chaîne de caractères à utiliser.

ChaineASupprimer$ La chaîne de caractères à supprimer.

Mode (optionnel) Peut être une combinaison des valeurs suivantes :

```
#PB_String_CaseSensitive :
 La suppression est
 sensible à la casse
 (A<>a). Par défaut.
#PB_String_NoCase        :
 La suppression est
 insensible à la casse
 (A=a).
```

Position (optionnel) Position du premier caractère à partir de laquelle la suppression s'effectue.
La position du premier caractère est 1.

NbOccurences (optionnel) La suppression s'arrêtera une fois 'NbOccurences' atteint.

Valeur de retour

Renvoie une chaîne dont toutes les occurrences ChaineASupprimer$ ont été supprimées.

Exemple

```
1  Debug RemoveString("deviser
     de l'Art", "de")
     ; Affichera "viser  l'Art"
2  Debug RemoveString("deviser
     de l'Art", "de", 0, 1, 1)
     ; Affichera "viser de
     l'Art"
```

Voir aussi

CountString() , FindString() , InsertString() , ReplaceString() , ReverseString() , StringByteLength() , StringField()

OS Supportés

Tous

146.16 ReplaceString

Syntaxe

```
Resultat\$ =
    ReplaceString(Chaine$,
    ChaineCherchee$,
    ChaineRemplacee$ [, Mode
    [, PositionDepart [,
    NbOccurences]]])
```

Description

Remplace une sous-chaîne par une autre.

Arguments

Chaine$ La chaîne de caractères à utiliser.

ChaineCherchee$ La chaîne de caractères à substituée.

ChaineRemplacee$ La chaîne de caractères de remplacement.

Mode (optionnel) Peut être une combinaison des valeurs suivantes :

```
#PB_String_CaseSensitive :
 La suppression est
 sensible à la casse
 (A<>a). Par défaut.
#PB_String_NoCase :
 Recherche les
 occurrences sans tenir
 compte de la casse (A=a).
#PB_String_InPlace:
 Remplacement direct dans
 la Chaîne de caractères
 initiale 'Chaîne$'. Dans
 ce cas,
                   'ChaineRemplacee$'
 doit absolument avoir la
 même taille que
 'ChaineCherchee$'.
                   C'est
 une fonctionnalité pour
 programmeurs
 expérimentés qui ont
 besoin d'une fonction
                   extrêmement
 rapide pour ce cas
 particulier. Quand cette
 option est utilisée, le
 résultat
                   renvoyé
 par ReplaceString() doit
 être ignoré (car c'est
```

```
la chaîne de caractères
passée en
                    paramètre
qui est modifiée).
```

PositionDepart (optionnel) Permet de spécifier à partir de quel caractère le remplacement doit être effectué. Sa valeur minimale est 1.

NbOccurences (optionnel) Le remplacement s'arrêtera une fois 'NbOccurences' atteint.

Valeur de retour

Renvoie un chaîne dont toutes les occurrences 'ChaineCherchee$' ont été substituées par 'ChaineRemplacee$'.

Exemple

```
1  Debug
     ReplaceString("deviser de
     l'Art", " de", " sur") ;
     Affichera "deviser sur
     l'Art"
2
3  test$ = "Salut les gens,
     Salut les gens"
4  Resultat$ =
     ReplaceString(test$,
     "SALUT", "ho non...",
     #PB_String_NoCase, 10) ;
     Affichera "Salut les gens,
     ho non... les gens"
5  Debug Resultat$
6
7  test$ = "Bundy, Barbie,
     Buddy"
8  ReplaceString(test$, "B",
     "Z", #PB_String_InPlace,
     1)  ; Tous les B seront
     remplacés par des Z  (Dans
     ce mode la chaîne de
     caractères passée en
     paramètre est modifiée)
9  Debug test$   ; Affichera
     "Zundy, Zarbie, Zuddy".
```

Voir aussi

CountString() , FindString() , InsertString() , RemoveString() , ReverseString() , StringByteLength() , StringField()

OS Supportés

Tous

146.17 ReverseString

Syntaxe

```
Resultat\$ =
    ReverseString(Chaine$)
```

Description

Inverse tous les caractères d'une chaîne.

Arguments

Chaine$ La chaîne de caractères à utiliser.

Valeur de retour

Inverse tous les caractères de la 'Chaine$'. Le dernier caractère devient le premier caractère et ainsi de suite.

Exemple

```
1   Debug
      ReverseString("Salut")   ;
      Affichera "tulaS"
```

Voir aussi

CountString() , FindString() , InsertString() , RemoveString() , ReplaceString() , StringByteLength() , StringField()

OS Supportés

Tous

146.18 Right

Syntaxe

```
Resultat\$ = Right(Chaine$,
    Longueur)
```

Description

Renvoie la partie droite d'une chaîne.

Arguments

Chaine$ La chaîne de caractères à utiliser.

Longueur Le nombre de caractères renvoyés est déterminé par Longueur.

Valeur de retour

Renvoie les caractères de la partie droite de la chaîne 'Chaine$'.

Remarques

Cette fonction renverra toute la chaîne si la longueur est supérieure à la taille de la chaîne de caractères.

Exemple

```
Debug Right("PureBasic", 5)
  ; Affichera "Basic".
```

Voir aussi

Left() , Len() , Mid()

OS Supportés

Tous

146.19 RSet

Syntaxe

```
Resultat\$ = RSet(Chaine$,
    Longueur [, Caractere$])
```

Description

Ajuste la longueur de la chaîne de caractères en y ajoutant des caractères en début de chaîne si nécessaire pour atteindre la longueur spécifiée.

Arguments

Chaine$ La chaîne de caractères à utiliser.

Longueur La longueur de chaîne à atteindre.

Caractere$ (optionnel) Peut être utilisé en lieu et place du caractère d'espacement qui est la valeur par défaut.

Valeur de retour

Renvoie une chaîne en y ajoutant des caractères en début de chaîne si nécessaire pour atteindre la longueur spécifiée.

Remarques

Si la chaîne est plus longue que la longueur spécifiée, elle sera tronquée à partir de la fin de la chaîne.

Exemple

```
1  Resultat$ = RSet("R", 8)
               ; Le résultat
    est: "       R"
2  Resultat$ = RSet("R", 8,
    "-")       ; Le résultat
    est: "-------R"
3  Resultat$ =
    RSet("LongString", 4) ; Le
    résultat est: "Long"
```

Voir aussi

LSet()

OS Supportés

Tous

146.20 RTrim

Syntaxe

```
Resultat\$ = RTrim(Chaine$ [,
    Caractere$])
```

Description

Supprime tous les espaces situés à la fin d'une chaîne.

Arguments

Chaine$ La chaîne de caractères à utiliser.

Caractere$ (optionnel) Supprimera tous les 'Caractere$' situés à la fin de la chaîne.
'Caractere$' doit être une chaîne contenant seulement un caractère.

Valeur de retour

Renvoie une chaîne dont tous les espaces situés à la fin de la chaîne ont été supprimés.

Exemple

```
1  Debug RTrim("PureBasic    ")
          ; Affichera
    "PureBasic".
2  Debug RTrim("Salut le Monde
    !!", "!") ; Affichera
    "Salut le Monde "
```

Voir aussi

LTrim() , Trim()

OS Supportés

Tous

146.21 StringField

Syntaxe

```
Resultat\$ =
    StringField(Chaine$,
    Index, Delimiteur$)
```

Description

Renvoie un champ d'une chaîne à l'index spécifié.

Arguments

Chaine$ La chaîne de caractères à utiliser.

Index La position du champ.
La première position est 1.

Delimiteur$ La chaîne choisie comme séparateur des champs (plusieurs caractères de longueur autorisé).

Valeur de retour

Renvoie un champ d'une chaîne à l'index spécifié.

Exemple

```
1 For k=1 To 7
2   Debug StringField("Je suis
      une chaîne contenant des
      champs", k, " ")
3 Next
```

Voir aussi

CountString() , FindString() , InsertString() , RemoveString() , ReplaceString() , ReverseString() , StringByteLength()

OS Supportés

Tous

146.22 StringByteLength

Syntaxe

```
Resultat =
    StringByteLength(Chaine$
    [, Format])
```

Description

Renvoie le nombre d'octets nécessaire pour stocker une chaîne de caractères en mémoire.

Arguments

Chaine$ La chaîne de caractères à utiliser.

Format (optionnel)

```
      #PB_Ascii   : La chaîne
    est considérée comme une
    chaine ASCII
  #PB_UTF8    : La chaîne est
    considérée comme une
    chaine UTF-8
  #PB_Unicode: La chaîne est
    considérée comme une
    chaine unicode (Par
    défaut)
```

Note : le nombre d'octets renvoyés n'inclut pas le caractère 'nul' de fin de chaîne. La taille du caractère nul est de 1 octet en mode ASCII et UTF-8 et de 2 octets en mode unicode.

Valeur de retour

Renvoie le nombre d'octets nécessaire pour stocker la chaîne de caractères en mémoire en fonction du 'Format' spécifié.

Exemple

```
1  Debug StringByteLength("é",
     #PB_UTF8) ; Le Resultat
     sera 2
```

Exemple

```
1  Texte.s = "Salut !!!"
2
3  *mem =
     AllocateMemory(StringByteLength(Texte)
     + SizeOf(CHARACTER))
```

```
4   CopyMemory(@Texte, *mem,
      StringByteLength(Texte) +
      SizeOf(CHARACTER))
5   Debug MemorySize(*mem)
6   Debug PeekS(*mem)
7   ShowMemoryViewer(*mem,
      MemorySize(*mem))
```

Voir aussi

CountString() , FindString() , InsertString() , RemoveString() , ReplaceString() , ReverseString() , StringField()

OS Supportés

Tous

146.23 StrF

Syntaxe

```
Resultat\$ = StrF(Valeur.f [,
    NombreDeDecimales])
```

Description

Convertit un nombre à virgule en une chaîne de caractères.

Arguments

Valeur.f Le nombre à virgule à convertir.

NombreDeDecimales (optionnel) Le nombre maximum de décimales, avec la suppression des zéros à droite, sinon le nombre sera arrondi à 10 décimales. Le nombre sera arrondi si 'NombreDeDecimales' est plus petit que le nombre de décimales existantes dans 'Valeur.f'.

Valeur de retour

Renvoie une chaîne de caractères représentant la 'Valeur'.

Remarques

Les nombres entiers signés doivent être convertis avec Str() et les nombres entiers non signés avec StrU() .
Il est possible d'omettre cette commande lors de la concaténation d'une chaîne et d'un 'float', cela utilisera alors le comportement par défaut de StrF() .

Exemple

```
1  valeur.f = 10.54
2  Debug "Résultat: " +
     StrF(valeur)     ; Sans le
     2ème paramètre, le nombre
     flottant est arrondi à 10
     décimales.
3  Debug "Résultat: " +
     StrF(valeur,2)   ; Résultat
     avec deux décimales, la
     valeur est arrondie.
4  Debug "Résultat: " +
     StrF(valeur,0)   ; Résultat
     sans décimales, la valeur
     est arrondie.
```

Voir aussi

Str() , StrD() , StrU() , Val() , ValD() , ValF() , FormatNumber()

OS Supportés

Tous

146.24 StrD

Syntaxe

```
Resultat\$ = StrD(Valeur.d [,
    NombreDeDecimales])
```

Description

Convertit un nombre à virgule en double précision en une chaîne de caractères.

Arguments

Valeur.d Le nombre à virgule à convertir.

NombreDeDecimales (optionnel) Le nombre maximum de décimales, avec la suppression des zéros à droite, sinon le nombre sera arrondi à 10 décimales. Le nombre sera arrondi si 'NombreDeDecimales' est plus petit que le nombre de décimales existantes dans 'Valeur.d'. L'arrondi se fait au plus proche comme avec la fonction Round() et l'option #PB_Round_Nearest.

Valeur de retour

Renvoie une chaîne de caractères représentant la 'Valeur'.

Remarques

Les nombres entiers signés doivent être convertis avec Str() et les nombres entiers non signés avec StrU() .
Il est possible d'omettre cette commande lors de la concaténation d'une chaîne et d'un 'double', cela utilisera alors le comportement par défaut de StrD() .

Exemple

```
1  valeur.d = 10.54
2  Debug "Résultat: " +
     StrD(valeur)    ; Sans le
     2ème paramètre, le nombre
     flottant est arrondi à 10
     décimales.
3  Debug "Résultat: " +
     StrD(valeur,2)  ; Résultat
     avec deux décimales, la
     valeur est arrondie.
4  Debug "Résultat: " +
     StrD(valeur,0)  ; Résultat
     sans décimales, la valeur
     est arrondie.
```

Voir aussi

Str() , StrF() , StrU() , Val() , ValD() , ValF() , FormatNumber()

OS Supportés

Tous

146.25 Str

Syntaxe

```
Resultat\$ = Str(Valeur.q)
```

Description

Convertit un nombre entier de type 'quad' en une chaîne de caractères.

Arguments

Valeur.q Le nombre entier à convertir.

Valeur de retour

Renvoie une chaîne de caractères représentant la 'Valeur'.

Remarques

Les nombres à virgule doivent être convertis avec StrF() ou StrD() et les nombres entiers non signés avec StrU() .

Exemple

```
Valeur.q =
 100000000000000001
Debug "Résultat: " +
 Str(Valeur)
```

Voir aussi

Val() , Hex() , Bin() , StrD() , StrF() , StrU() , Val() , ValD() , ValF() , FormatNumber()

OS Supportés

Tous

146.26 StrU

Syntaxe

```
Resultat\$ = StrU(Valeur.q [,
    Type])
```

Description

Convertit un nombre entier non-signé en une chaîne de caractères.

Arguments

Valeur Le nombre entier à convertir.

Type (optionnel) #PB_Quad : La valeur sera traitée comme un 'quad' (0 à 18446744073709551615) (Par défaut)
#PB_Byte : La valeur est un octet (8-bit) allant de 0 à 255
#PB_Ascii : La valeur est un octet (8-bit) allant de 0 à 255
#PB_Word : La valeur est un word (16-bit) allant de 0 à 65535
#PB_Unicode: La valeur est un word (16-bit) allant de 0 à 65535
#PB_Long : La valeur est un long (32-bit) allant de 0 à 4294967295

Valeur de retour

Renvoie une chaîne de caractères représentant la 'Valeur'.

Remarques

Les nombres entiers signés doivent être convertis avec Str() et les nombres à virgule (float) avec StrF() ou StrD() .

Exemple

```
1  byte.b = 255
2  Debug Str(byte) ; Affichera
     -1
3  Debug StrU(byte, #PB_Byte)
     ; Affichera 255
```

Voir aussi

Str() , StrD() , StrF() , Val() , ValD() , ValF() , Hex() , Bin() , FormatNumber()

OS Supportés

Tous

146.27 Space

Syntaxe

```
Resultat\$ = Space(Longueur)
```

Description

Créé une chaîne de caractères de longueur 'Longueur' et ne contenant que des espaces.

Arguments

Longueur Le nombre d'espaces à ajouter.

Valeur de retour

Renvoie une chaîne de caractères de longueur 'Longueur' et ne contenant que des espaces.

Exemple

```
1  Debug Space(10) ; Affichera
     "          ".
```

OS Supportés

Tous

146.28 Trim

Syntaxe

```
Resultat\$ = Trim(Chaine$ [,
    Caractere$])
```

Description

Supprime tous les espaces situés au début et à la fin de la chaîne de caractères.

Arguments

Chaine$ La chaîne de caractères à utiliser.

Caractere$ (optionnel) Supprime tous les 'Caractere$' situés au début et à la fin de la chaîne.
'Caractere$' doit être une chaîne contenant seulement un caractère.

Valeur de retour

Renvoie une chaîne de caractères dont tous les espaces situés au début et à la fin ont été supprimés.

Exemple

```
1  Debug Trim("   PureBasic
     ") ; Affichera "PureBasic".
2  Debug Trim("!!Salut!!",
     "!")  ; Affichera "Salut"
```

Voir aussi

LTrim() , RTrim()

OS Supportés

Tous

146.29 UCase

Syntaxe

```
Resultat\$ = UCase(Chaine$)
```

Description

Convertit une chaîne en majuscule.

Arguments

Chaine$ La chaîne de caractères à utiliser.

Valeur de retour

Renvoie dans Resultat$ la chaîne 'Chaine$' convertie en lettres majuscules (quand c'est possible).
Cette commande accepte les caractères accentués. Si un 'é' minuscule est trouvé, il sera transformé en un 'É' majuscule.

Exemple

```
1  Debug UCase("PureBasic, La puissance à l'état pur") ; Affichera "PUREBASIC, LA PUISSANCE À L'ÉTAT PUR"
```

Voir aussi

LCase()

OS Supportés

Tous

146.30 UnescapeString

Syntaxe

```
Resultat\$ =
    UnescapeString(Chaine$ [,
    Mode])
```

Description

Transforme une chaîne avec des séquences d'échappement en chaîne standard.

Arguments

Chaine$ La chaîne à transformer.

Mode (optionnel)
#PB_String_EscapeInternal : déséchappe la chaîne en utilisant le format de PureBasic (Par défaut). (voir Règles de syntaxe générales
.
#PB_String_EscapeXML : déséchappe la chaîne en utilisant le format XML.

```
                                        Utile
    pour lire une chaîne
    provenant d'un arbre XML.
```

Valeur de retour

Renvoie la version non échappée de la chaîne.

Remarques

La fonction EscapeString() peut être utilisée pour faire l'opération inverse. Attention : Sous Windows, \t ne fonctionne pas avec les fonctions graphiques des bibliothèques 2DDrawing et VectorDrawing.

Exemple

```
1  Debug
     UnescapeString(~"Test=\"Hello\".")
     ; Affichera "Test="Hello"."
2  Debug
     UnescapeString("&lt;item&gt;Hello&lt;/item&gt;",
     #PB_String_EscapeXML) ;
     Affichera
     "<item>Hello</item>"
```

Voir aussi

EscapeString()

OS Supportés

Tous

146.31 ValD

Syntaxe

```
Resultat.d = ValD(Chaine$)
```

Description

Convertit une chaîne en une valeur numérique de type 'Double'.

Arguments

Chaine$ La chaîne de caractères à utiliser.

Valeur de retour

Renvoie une valeur numérique de type 'Double'.
La chaîne doit représenter un double au format décimal ou au format scientifique (exponentiel).
La conversion s'arrêtera au premier caractère non numérique.

Remarques

Les chaînes représentant un entier 'long' peuvent également être converties avec Val() , des nombres flottants 32 bits avec ValF() (avec moins de précision que ValD()).
"NaN", "-Infinity" et "+Infinity" sont permis.

Exemple

```
1  Debug ValD("10.000024") ;
     Affichera 10.000024.
2  Debug ValD("1.2345e-2") ;
     Affichera  0.012345
```

Voir aussi

Str() , StrD() , StrF() , StrU() , Val() , ValF() , FormatNumber()

OS Supportés

Tous

146.32 ValF

Syntaxe

```
Resultat.f = ValF(Chaine$)
```

Description

Convertit une chaîne en une valeur numérique de type 'Float'.

Arguments

Chaine$ La chaîne de caractères à utiliser.

Valeur de retour

Renvoie une valeur numérique de type 'Float'
La chaîne doit représenter un float au format décimal ou au format scientifique (exponentiel).
La conversion s'arrêtera au premier caractère non numérique.

Remarques

Les chaînes représentant un entier long peuvent également être converties avec Val() , des nombres flottants 64 bits avec ValD() (avec plus de précision que ValF()). "NaN", "-Infinity" et "+Infinity" sont permis.

Exemple

```
1 Debug ValF("10.75")      ;
    Affichera   10.75
2 Debug ValF("1.2345e+3") ;
    Affichera 1234.5
```

Voir aussi

Str() , StrD() , StrF() , StrU() , Val() , ValD() , FormatNumber()

OS Supportés

Tous

146.33 Val

Syntaxe

```
Resultat.q = Val(Chaine$)
```

Description

Convertit une chaîne en une valeur numérique de type quad.

Arguments

Chaine$ La chaîne de caractères à utiliser.

Valeur de retour

Renvoie une valeur numérique entière. La chaîne doit représenter un entier au format décimal, hexadécimal (avec le préfixe '$') ou binaire (avec le préfixe '%'). La conversion s'arrêtera au premier caractère non numérique.

Remarques

Les chaînes représentant une valeur flottante 32 bits devraient être converties avec ValF() , et 64 bits avec ValD() .

Exemple

```
Debug Val("1024102410241024") ; affichera '1024102410241024'.
Debug Val("$10FFFFFFF") ; affichera '73014444031'.
Debug Val("%1000") ; affichera '8'.
Debug Val("1.2345e+3") ; affichera '1'. Utiliser ValF ou ValD.
```

Voir aussi

Str() , StrD() , StrF() , ValD() , ValF() , FormatNumber()

OS Supportés

Tous

146.34 Ascii

Syntaxe

```
*Resultat = Ascii(Chaine$)
```

Description

Crée un tampon en mémoire contenant la version au format ASCII d'une chaîne de caractères.

Arguments

Chaine$ La chaîne à convertir au format ASCII.

Valeur de retour

La représentation ASCII de la chaîne.

Remarques

Lorsque le tampon n'est plus nécessaire, il doit être libéré avec FreeMemory() .
Cette fonction est surtout utile lors de l'utilisation de bibliothèques tiers qui requièrent des chaînes de caractères au format ASCII en entrée.
Le pseudotype 'p-ascii' peuvent également être utilisé pour automatiser le processus de conversion lors de l'importation des fonctions externes.

Exemple

```
*Ascii = Ascii("Hélé")
ShowMemoryViewer(*Ascii,
 MemorySize(*Ascii)) ;
 Affiche l'adresse du
 tampon suivi de 48 E9 6C
 E9 00

Debug PeekS(*Ascii, -1,
 #PB_Ascii) ; Affiche "Hélé"
```

Exemple

```
Macro Unicode(Mem, Type =
 #PB_Ascii)
  PeekS(Mem, -1, Type)
EndMacro

*Mem1 = Ascii("Test -
 éàîöÊÜ")
*Mem2 = UTF8("Test -
 éàîöÊÜ")

Texte.s = Unicode(*Mem1)
Debug Texte ; Affiche "Test
 - éàîöÊÜ"

Texte.s = Unicode(*Mem2,
 #PB_UTF8)
Debug Texte ; Affiche "Test
 - éàîöÊÜ"
```

Voir aussi

UTF8()

OS Supportés

Tous

146.35 UTF8

Syntaxe

```
*Resultat = UTF8(Chaine$)
```

Description

Crée un tampon en mémoire contenant la version au format UTF8 d'une chaîne de caractères.

Arguments

Chaine$ La chaîne à convertir au format UTF8.

Valeur de retour

La représentation UTF8 de la chaîne.

Remarques

Lorsque le tampon n'est plus nécessaire, il doit être libéré avec FreeMemory() .
Cette fonction est surtout utile lors de l'utilisation de bibliothèques tiers qui requièrent des chaînes de caractères au format UTF8 en entrée.
Le pseudotype 'p-utf8' peuvent également être utilisé pour automatiser le processus de conversion lors de l'importation des fonctions externes.

Exemple

```
*UTF8 = UTF8("Hélé")
ShowMemoryViewer(*UTF8, MemorySize(*UTF8)) ; Affiche l'adresse du tampon suivi de 48 C3 A9 6C C3 A9 00
;ShowMemoryViewer(*UTF8, MemoryStringLength(*UTF8, #PB_UTF8|#PB_ByteLength));

Debug PeekS(*UTF8, -1, #PB_UTF8) ; Affiche "Hélé"

; Format UTF8: certains caractères sont codés sur 1 octet, d'autres sur 2, 3 ou 4 octets
; H = 48
; é = C3 A9
; l = 6C
; é =  C3 A9
```

Exemple

```
Macro Unicode(Mem, Type = #PB_Ascii)
  PeekS(Mem, -1, Type)
EndMacro

*Mem1 = Ascii("Test - éàîöÊÜ")
*Mem2 = UTF8("Test - éàîöÊÜ")

Texte.s = Unicode(*Mem1)
Debug Texte ; Affiche "Test - éàîöÊÜ"

```

```
11  Texte.s = Unicode(*Mem2,
      #PB_UTF8)
12  Debug Texte ; Affiche "Test
      - éàîöÊÜ"
```

Voir aussi

Ascii()

OS Supportés

Tous

146.36 FormatNumber

Syntaxe

```
Resultat\$ =
    FormatNumber(Nombre.d [,
    NbDecimales [, Virgule$ [,
    SeparateurMillier$]]])
```

Description

Formate un nombre au format monétaire.

Arguments

Nombre Le nombre à formater.

NbDecimales (optionnel) Nombre de décimales à afficher.

Virgule$ (optionnel) La chaîne de caractères à utiliser pour séparer la partie entière de la partie décimales.
Elle contenir plusieurs caractères.
La valeur par défaut est ".".

SeparateurMillier$ (optionnel) La chaîne de caractères à utiliser pour séparer les milliers.
Elle peut contenir plusieurs caractères.
La valeur par défaut est ",".

Valeur de retour

Le nombre formaté au format chaîne de caractères.

Exemple

```
1  Debug
     FormatNumber(125400.25) ;
     Affiche 125,400.25
2  Debug
     FormatNumber(1125400.25,
     2, ",", " ") ; Affiche 1
     125 400,25  Formatage Fr
```

Voir aussi

Str() , StrU() , StrF() , StrD()

OS Supportés

Tous

Chapitre 147

SysTray

Généralités

SysTray est une zone de notification située à droite de la barre des tâches qui contient des icônes (programmes résidents, antivirus...) et l'horloge. PureBasic gère l'accès à cette zone et vous permet d'ajouter un nombre quelconque d'icônes. Sur certaines distributions Linux (comme Ubuntu), les icônes de la zone de notification peuvent être masquées par défaut. Pour plus d'informations, consulter ce lien.

OS Supportés

Tous

147.1 AddSysTrayIcon

Syntaxe

```
Resultat =
    AddSysTrayIcon(#SysTrayIcone,
    FenetreID, ImageID)
```

Description

Ajoute une icône dans la zone de notification.

Arguments

#SysTrayIcone Le numéro de la nouvelle icône.
PB_Any # peut être utilisé pour générer automatiquement ce numéro.

FenetreID L'identifiant système de la fenêtre.
Peut être obtenu avec la fonction WindowID() .

ImageID Numéro de l'image préalablement chargée par la fonction LoadImage() .
Le format PNG a l'avantage d'utiliser la transparence.
Un ImageID valide peut être obtenu simplement avec la fonction ImageID() .

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Quand un évènement intervient sur une icône de la zone SysTray, l'évènement #PB_Event_SysTray est renvoyé.
EventGadget() peut alors être utilisé pour connaître l'icône cliquée.
La fonction EventType() est également mise à jour par cette fonction.
Toutes vos icônes SysTray sont automatiquement supprimées à la fermeture du programme.

Exemple

```
1   If OpenWindow(0, 100, 150,
      300, 100, "Zone de
      notification",
      #PB_Window_SystemMenu)
2
3   UsePNGImageDecoder()
4
5   Icone$ = #PB_Compiler_Home
      +
      "examples/sources/Data/world.png"
6
7   ; Ajout d'une icône dans la
      zone de notification
8   AddSysTrayIcon(0,
      WindowID(0), LoadImage(0,
      Icone$))
9
10  Repeat
11    Event = WaitWindowEvent()
12  Until Event =
      #PB_Event_CloseWindow
13
14  EndIf
```

Voir aussi

RemoveSysTrayIcon() ,
ChangeSysTrayIcon()

OS Supportés

Tous

147.2 ChangeSysTrayIcon

Syntaxe

```
ChangeSysTrayIcon(#SysTrayIcone,
    ImageID)
```

Description

Change une icône de la zone de notification.

Arguments

#SysTrayIcone Le numéro de l'icône à changer.

ImageID Numéro de la nouvelle image préalablement chargée par la fonction LoadImage() .
Le format PNG a l'avantage d'utiliser la transparence.
Un ImageID valide peut être obtenu simplement avec la fonction ImageID() .

Valeur de retour

Aucune.

Exemple

```
1  If OpenWindow(0, 100, 150,
     300, 100, "Zone de
     notification",
     #PB_Window_SystemMenu)
2
3  UsePNGImageDecoder()
4
5  Icone$ = #PB_Compiler_Home
     +
     "examples/sources/Data/world.png"
6
7  ; Ajout d'une icône dans la
     zone de notification
8  AddSysTrayIcon(0,
     WindowID(0), LoadImage(0,
     Icone$))
9
10 MessageRequester("Info",
     "Changement de l'icône de
     la zone de notification.")
11 Icone$ = #PB_Compiler_Home
     +
     "examples/sources/Data/Drive.bmp"
12 ; Changement de l'icône
```

```
13    ChangeSysTrayIcon (0,
       LoadImage(0, Icone$))
14    MessageRequester("Info",
       "Icône remplacée." +
       Chr(10) + "Observez la
       transparence de cette
       nouvelle icône...")
15
16    Repeat
17      Event = WaitWindowEvent()
18    Until Event =
       #PB_Event_CloseWindow
19
20  EndIf
```

Voir aussi

AddSysTrayIcon() , RemoveSysTrayIcon()

OS Supportés

Tous

147.3 IsSysTrayIcon

Syntaxe

```
Resultat =
    IsSysTrayIcon(#SysTrayIcone)
```

Description

Teste si une icône de la zone de notification est correctement initialisée.

Arguments

#**SysTrayIcone** Le numéro de l'icône.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.

Voir aussi

AddSysTrayIcon()

OS Supportés

Tous

147.4 SysTrayIconToolTip

Syntaxe

```
SysTrayIconToolTip(#SysTrayIcone,
    Texte$)
```

Description

Associe un texte flottant (bulle d'aide) à une icône de la zone de notification.

Arguments

#SysTrayIcone Le numéro de l'icône à utiliser.

Texte$ 'Tooltip' ou texte flottant qui apparaît au bout d'un certain temps lorsque le curseur de la souris est immobile au dessus de l'icône.

Valeur de retour

Aucune.

Exemple

```
1   If OpenWindow(0, 100, 150,
     300, 100, "Zone de
     notification",
     #PB_Window_SystemMenu)
2
3   UsePNGImageDecoder()
4
5   Icone$ = #PB_Compiler_Home
     +
     "examples/sources/Data/world.png"
6
7   ; Ajout d'une icône dans la
     zone de notification
8   AddSysTrayIcon(0,
     WindowID(0), LoadImage(0,
     Icone$))
9
10  ; Ajout d'un texte flottant
11  SysTrayIconToolTip(EventGadget(),
     "Hello le Monde")
12  MessageRequester("Info",
     "Survolez l'icône pour
     voir le texte apparaître.")
13
14  Repeat
15    Event = WaitWindowEvent()
```

```
  Until Event =
    #PB_Event_CloseWindow

EndIf
```

Voir aussi

AddSysTrayIcon()

OS Supportés

Tous

147.5 RemoveSysTrayIcon

Syntaxe

```
RemoveSysTrayIcon(#SysTrayIcone)
```

Description

Supprime une icône de la zone de notification.

Arguments

#SysTrayIcone Le numéro de l'icône à supprimer.
Si #PB_All est spécifié, toutes les icônes SysTray restantes sont libérées.

Valeur de retour

Aucune.

Remarques

Toutes les icônes restantes sont automatiquement supprimées à la fermeture du programme.

Exemple

```
  If OpenWindow(0, 100, 150,
    300, 100, "Zone de
    notification",
    #PB_Window_SystemMenu)

  UsePNGImageDecoder()

  Icone$ = #PB_Compiler_Home
    +
    "examples/sources/Data/world.png"

  ; Ajout d'une icône dans la
    zone de notification
```

```
  AddSysTrayIcon(0, WindowID(0), LoadImage(0, Icone$))

  MessageRequester("Info", "Suppression de l'icône.") ; Suppression de l'icône
  RemoveSysTrayIcon (0)
  MessageRequester("Info", "Icône supprimée.")

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow

EndIf
```

Voir aussi

AddSysTrayIcon()

OS Supportés

Tous

Chapitre 148

System

Généralités

La bibliothèque système offre l'accès à certaines informations spécifiques au système, comme le nombre de processeurs, la quantité de mémoire disponible, etc.

OS Supportés

Tous

148.1 CocoaMessage

Syntaxe

```
Resultat =
    CocoaMessage(AddresseValeurDeRetour,
    Objet, Methode$ [,
    ValeurParametre [,
    Parametre$, [,
    ValeurParametre, ...]]])
```

Description

Pour les utilisateurs avancés. Disponible sur Mac OS X uniquement, elle permet d'envoyer facilement un message Objective-C au framework OS X et d'accéder à toute l'API. Habituellement Objective-C utilise des parenthèses pour avoir une syntaxe claire pour les messages. Comme PureBasic ne supporte pas Objective-C en natif, il doit l'émuler, alors la syntaxe est un peu différente. Une fois apprise, il est facile d'appeler une API. Pour plus d'exemples, n'hésitez pas à lire le thread suivant sur le forum.

Arguments

AddresseValeurDeRetour Si l'appel de l'API renvoie une structure ou un type différent d'entier 'integer', ce champ est

utilisé pour définir le résultat du retour. Une adresse vers la structure ou la variable doit être spécifiée. Si zéro est spécifié, le résultat sera ignoré.

Objet L'objet sur lequel les méthodes Objective-C seront appelées. Il peut être égal à zéro si la méthode est une méthode statique (c'est le cas la plupart du temps lors de la création d'un objet).

**Methode$** La méthode de l'objet à appeler, suivi d'un ' ;' comme d'habitude. Si la méthode a besoin d'une structure comme paramètre, '@' doit être ajouté après le point-virgule. Si la méthode s'attend à une chaîne comme paramètre, '$' peut être ajouté après le point-virgule, de sorte que la chaîne sera automatiquement convertie dans une NSString temporaire. Ce n'est pas requis, mais il peut être utile et ça facilite le codage. Si la méthode n'est pas prise en charge par l'objet, un message du débogueur sera levé lors de l'exécution.

ValeurParametre (optionnel) La valeur du paramètre associé à la méthode précédente.

Parameter$ (optionnel) Le paramètre de la méthode suivante. PureBasic supporte jusqu'à 7 paramètres par méthode.

Valeur de retour

Valeur de retour de type 'Integer'. Utile pour la création d'identifiant d'objets.

Remarques

PureBasic a déjà configuré un pool de mémoire temporaire qui est vidé à chaque fois que WindowEvent() ou WaitWindowEvent() est appelé. Si vous avez besoin de libérer immédiatement les gros objets, vous devrez créer un pool de mémoire locale pour chacun de vos appels.

Exemple : avec une chaîne

```
1  ; Objective-C:
2  ;   ColorList =
     [NSColorList
     colorListNamed:@"Crayons"];
3  ;
4  ColorList = CocoaMessage(0,
     0, "NSColorList
     colorListNamed:$",
     @"Crayons") ; Créera un
     NSString pour "Crayons"
```

Exemple : avec un type complexe

```
; Objective-C:
;   Transform = [NSAffineTransform transform];
;
Transform = CocoaMessage(0, 0, "NSAffineTransform transform") ; Obtenir une 'identity transform'

; Objective-C:
;   [TransForm scaleXBy:sx yBy:sy];
;
sx.CGFloat = 5.5
sy.CGFloat = 20
CocoaMessage(0, TransForm, "scaleXBy:@", @sx, "yBy:@", @sy) ; Agrandit x par 5.5, y par 20. comme sx n'est pas un 'integer', '@' doit être utilisé.

; Objective-C:
;   NSAffineTransform TransformStruct = [TransForm transformStruct];
;
CocoaMessage(@TransformStruct.NSAffineTransform, TransForm, "transformStruct") ; Obtenir une structure transform

Debug TransformStruct\m11 ; debug affiche 5.5
```

OS Supportés

MacOS X

148.2 CPUName

Syntaxe

```
Resultat\$ = CPUName()
```

Description

Renvoie le nom du processeur.

Arguments

Aucun.

Valeur de retour

Renvoie le nom du CPU (la totalité des informations du fournisseur). Il n'y a pas de norme à ce propos, alors cette fonction ne doit pas être utilisée pour identifier le CPU lors de l'exécution, mais peut être utile pour du deboguage par exemple.

Exemple

```
1  Debug CPUName() ; Ex:
     "Intel(R) Core(TM) i7 CPU
     860  @ 2.80GHz"
```

Voir aussi

ComputerName() , UserName()

OS Supportés

Tous

148.3 Delay

Syntaxe

```
Delay(Temps)
```

Description

Suspend totalement l'exécution du programme pendant une durée spécifiée.

Arguments

Temps Le délai en millisecondes.
Attention, le délai réel peut être plus long que le temps spécifié.

Valeur de retour

Aucune.

Remarques

Cette fonction est particulièrement utile dans les boucles d'évènements utilisant WindowEvent() ou ExamineKeyboard() , pour éviter qu'elles ne monopolisent tout le temps processeur.
En fait, cette fonction suspend le thread en cours d'exécution.

Voir aussi

ElapsedMilliseconds()

OS Supportés

Tous

148.4 ElapsedMilliseconds

Syntaxe

```
Resultat.q =
    ElapsedMilliseconds()
```

Description

Renvoie le nombre de millisecondes qui se sont écoulées depuis le démarrage de l'ordinateur.

Arguments

Aucun.

Valeur de retour

Renvoie le temps écoulé en millisecondes.

Remarques

La valeur absolue qui est retournée n'est d'aucune utilité, car elle varie en fonction du système d'exploitation. Au lieu de cela, cette fonction doit être utilisée pour calculer les différences de temps entre plusieurs appels à ElapsedMilliseconds().
Cette fonction est relativement précise : il peut y avoir de légères différences en fonction de l'OS sur lequel le programme s'exécute. cela est dû au fait que certains systèmes ont une minuterie avec une résolution inférieure à d'autres.

Exemple

```
1  ;
2  ; exemple d'utilisation
     comme chronomètre
3  ;
4  TempsDepart.q =
     ElapsedMilliseconds()  ;
     Récupère la valeur actuelle
5  Delay(1000)  ; Attend 1000
     millisecondes
6  TempsEcoule.q =
     ElapsedMilliseconds()-TempsDepart
      ; La valeur 'TempsEcoule'
     devrait être d'environ
     1000 millisecondes
7
```

```
8 | Debug "Temps écoulé :
    "+Str(TempsEcoule)+"
    millisecondes"
```

Voir aussi

Delay()

OS Supportés

Tous

148.5 DoubleClickTime

Syntaxe

```
Resultat = DoubleClickTime()
```

Description

Renvoie le réglage système de la durée du double-clic.
Si deux clics de souris se produisent dans cette durée, ils sont considérés comme un double-clic.

Arguments

Aucun.

Valeur de retour

Renvoie la durée du double-clic en millisecondes.

Remarques

Cette fonction peut être utilisée avec Openscreen ou CanvasGadget() .

Voir aussi

ElapsedMilliseconds() , OpenScreen() , OpenWindowedScreen() , CanvasGadget()

OS Supportés

Tous

148.6 OSVersion

Syntaxe

```
Resultat = OSVersion()
```

Description

Renvoie la version du système d'exploitation sur lequel le programme s'exécute.

Arguments

Aucun.

Valeur de retour

Renvoie l'une des valeurs suivantes, selon le système d'exploitation :
Windows

```
#PB_OS_Windows_NT3_51
#PB_OS_Windows_95
#PB_OS_Windows_NT_4
#PB_OS_Windows_98
#PB_OS_Windows_ME
#PB_OS_Windows_2000
#PB_OS_Windows_XP
#PB_OS_Windows_Server_2003
#PB_OS_Windows_Vista
#PB_OS_Windows_Server_2008
#PB_OS_Windows_7
#PB_OS_Windows_Server_2008_R2
#PB_OS_Windows_8
#PB_OS_Windows_Server_2012
#PB_OS_Windows_8_1
#PB_OS_Windows_Server_2012_R2
#PB_OS_Windows_10
#PB_OS_Windows_11
#PB_OS_Windows_Future  ;
 Nouvelle version de
 Windows (n'existant pas
 lorsque le programme a
 été écrit)
```

Linux

```
#PB_OS_Linux_2_2
#PB_OS_Linux_2_4
#PB_OS_Linux_2_6
#PB_OS_Linux_Future ;
 Nouvelle version de
 Linux (n'existant pas
 lorsque le programme a
 été écrit)
```

Mac OSX

```
#PB_OS_MacOSX_10_0
#PB_OS_MacOSX_10_1
#PB_OS_MacOSX_10_2
#PB_OS_MacOSX_10_3
#PB_OS_MacOSX_10_4
#PB_OS_MacOSX_10_5
#PB_OS_MacOSX_10_6
#PB_OS_MacOSX_10_7
#PB_OS_MacOSX_10_8
#PB_OS_MacOSX_10_9
#PB_OS_MacOSX_10_10
#PB_OS_MacOSX_10_11
#PB_OS_MacOSX_10_12
#PB_OS_MacOSX_10_13
```

```
#PB_OS_MacOSX_10_14
#PB_OS_MacOSX_10_15
#PB_OS_MacOSX_11
#PB_OS_MacOSX_12
#PB_OS_MacOSX_Future  ;
  Nouvelle version de
  MacOS X (n'existant pas
  lorsque le programme a
  été écrit)
```

Exemple

```
1  Select OSVersion()
2    Case #PB_OS_Windows_98
3      MessageRequester("Info",
     "Windows 98")
4
5    Case #PB_OS_Windows_2000
6      MessageRequester("Info",
     "Windows 2000")
7
8    Case #PB_OS_Windows_XP
9      MessageRequester("Info",
     "Windows XP")
10
11   Default
12     MessageRequester("Info",
     "version de Windows
     inconnue")
13 EndSelect
```

Note : La valeur des constantes respecte l'ordre chronologique de sortie des différentes versions, ce qui permet de déterminer rapidement si une version est plus ancienne ou plus récente qu'une version donnée.

```
1  If OsVersion() <
     #PB_OS_Windows_2000
2    ;
3    ; Toutes les versions
     plus anciennes que Windows
     2000
4    ;
5  EndIf
```

OS Supportés

Tous

148.7 ComputerName

Syntaxe

```
Resultat\$ = ComputerName()
```

Description

Renvoie le nom de l'ordinateur.

Arguments

Aucun.

Valeur de retour

Renvoie le nom de l'ordinateur.

Exemple

```
Debug "Nom de l'ordinateur : " + ComputerName()
```

Voir aussi

UserName() , CPUName()

OS Supportés

Tous

148.8 UserName

Syntaxe

```
Resultat\$ = UserName()
```

Description

Renvoie le nom de l'utilisateur en cours.

Arguments

Aucun.

Valeur de retour

Renvoie le nom de l'utilisateur actuellement loggé.

Exemple

```
Debug "Utilisateur actuellement connecté: " + UserName()
```

Voir aussi

ComputerName() , CPUName()

OS Supportés

Tous

148.9 MemoryStatus

Syntaxe

```
Resultat.q =
    MemoryStatus(Type)
```

Description

Renvoie les informations sur un type de mémoire demandé.

Arguments

Type Le type de mémoire à tester.
Peut être une des valeurs suivantes :

```
#PB_System_TotalPhysical:
 La quantité totale de
 mémoire installée, en
 octets.
#PB_System_FreePhysical :
 La mémoire disponible,
 en octets.
#PB_System_TotalVirtual :
 La taille de la mémoire
 virtuelle totale, en
 octets (Windows
 uniquement).
#PB_System_FreeVirtual  :
 La taille de la mémoire
 virtuelle disponible, en
 octets (Windows
 uniquement).
#PB_System_TotalSwap    :
 La taille de la mémoire
 de swap total, en octets
 (Windows et Linux
 uniquement).
#PB_System_FreeSwap     :
 La taille de la mémoire
 de swap disponible, en
 octets (Windows et Linux
 uniquement).
#PB_System_PageSize     :
 La taille de la page
 mémoire, en octets
 (généralement 4 Ko).
```

Valeur de retour

Renvoie les informations sur le type de mémoire spécifiée.

Exemple

```
Debug "La quantité totale de mémoire installée, en octets: " + MemoryStatus(#PB_System_TotalPhysical)
Debug "La mémoire disponible, en octets: " + MemoryStatus(#PB_System_FreePhysical)
Debug "La taille de page mémoire, en octets (généralement 4 Ko): " + MemoryStatus(#PB_System_PageSize)
;(Windows et Linux uniquement)
;Debug "La taille de la mémoire de swap total, en octets (Windows et Linux uniquement): " + MemoryStatus(#PB_System_TotalSwap)
;Debug "La taille de la mémoire de swap disponible, en octets (Windows et Linux uniquement): " + MemoryStatus(#PB_System_FreeSwap)
;(Windows uniquement)
;Debug "La taille de la mémoire virtuelle totale, en octets (Windows uniquement): " + MemoryStatus(#PB_System_TotalVirtual)
;Debug "La taille de la mémoire virtuelle disponible, en octets (Windows uniquement): " + MemoryStatus(#PB_System_FreeVirtual)
```

Voir aussi

CountCPUs()

OS Supportés

Tous

148.10 CountCPUs

Syntaxe

```
Resultat = CountCPUs([Type])
```

Description

Renvoie le nombre de core de microprocesseurs.

Arguments

Type (optionnel) Peut être une des valeurs suivantes :

```
#PB_System_CPUs         :
 Le nombre total de
 coeurs de processeurs.
 (Par défaut)
#PB_System_ProcessCPUs :
 Le nombre de coeurs de
 processeurs disponibles
 pour le processus en
 cours.
                          Ceci
 est utile car le système
 d'exploitation peut
 obliger un processus à
                          n'utiliser
 qu'un petit nombre de
 CPU.
```

Valeur de retour

Renvoie le nombre de coeurs.

Exemple

```
1  Debug "Le nombre total de
     coeurs installés sur
     l'ordinateur: " +
     CountCPUs(#PB_System_CPUs)
2  Debug "Le nombre de coeurs
     disponibles pour le
     processus en cours: " +
     CountCPUs(#PB_System_ProcessCPUs)
```

Voir aussi

MemoryStatus()

OS Supportés

Tous

Chapitre 149

Terrain

Généralités

Les terrains sont des scènes 3D extérieures qui simulent un environnement naturel réaliste basé sur des plans en 2D pré-calculés.
Ils sont utiles dans de nombreux cas, comme par exemple les jeux de simulation en 3D.
InitEngine3D() doit être appelé avec succès avant l'utilisation des commandes 'Terrain'.
Il est possible de créer un terrain gigantesque (La taille est fonction de la puissance de l'ordinateur) en réunissant plusieurs 'parcelles' plus petites appelées 'tiles'.

OS Supportés

Tous

149.1 FreeTerrain

Syntaxe

```
FreeTerrain(#Terrain)
```

Description

Libère un terrain et toute la mémoire associée.

Arguments

#Terrain Le terrain à libérer.
Si #PB_All est spécifié, tous les terrains restants sont libérés.

Valeur de retour

Aucune.

Remarques

Ce terrain ne doit pas être utilisé en utilisant son numéro avec les autres fonctions de cette bibliothèque, après l'appel de cette fonction, à moins de le créer à nouveau.
Tous les terrains restants sont automatiquement libérés lorsque le programme se termine.

Voir aussi

CreateTerrain()

OS Supportés

Tous

149.2 FreeTerrainBody

Syntaxe

```
FreeTerrainBody(#Terrain)
```

Description

Libère le corps associé au terrain.

Arguments

#**Terrain** Le terrain à utiliser.

Valeur de retour

Aucune.

Voir aussi

CreateTerrainBody()

OS Supportés

Tous

149.3 SetupTerrains

Syntaxe

```
SetupTerrains(LumiereID,
    DistanceCarteComposite.f,
    Options)
```

Description

Configuration des paramètres par défaut pour tous les futurs terrains.

Arguments

LumiereID La lumière à utiliser pour le rendu du terrain.
Pour obtenir un identifiant valide, utilisez LightID() .

DistanceCarteComposite.f La distance à laquelle on commence à utiliser une carte composite si elle est présente, dans l'unité du monde.

Options Peut être une combinaison des valeurs suivantes :

```
#PB_Terrain_Lightmap      :
 Active la carte allégée
 pour les terrains (une
 texture avec des ombres
 statiques).
#PB_Terrain_NormalMapping:
 Active le mapping normal
 pour les terrains. Il
 s'agit d'une texture
 spéciale
                           qui
 simule le relief, comme
 les petits rochers, etc.
```

Valeur de retour

Aucune.

Voir aussi

CreateTerrain()

OS Supportés

Tous

149.4 CreateTerrain

Syntaxe

```
Resultat =
    CreateTerrain(#Terrain,
    Taille, TailleMonde,
    Echelle, NbCouches,
    NomFichier$, Extension$)
```

Description

Crée un nouveau terrain.

Arguments

#**Terrain** Le numéro d'identification du nouveau terrain.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Taille La taille du nouveau terrain (moins une arête).

TailleMonde La taille du monde du nouveau terrain.

Echelle Le facteur d'échelle à appliquer au nouveau terrain.

NbCouches Le nombre de couches de texture du nouveau terrain.
Pour ajouter une couche de texture, utilisez AddTerrainTexture() .

NomFichier$ Le nom du fichier (sans l'extension) pour stocker les données de terrain précalculées.
Comme le terrain peut être complexe et prendre beaucoup de temps pour sa création, un cache sera écrit sur le disque et rechargé automatiquement s'il est présent (et non périmé).

Extension$ L'extension utilisée par les fichiers de données.

Valeur de retour

Renvoie une valeur non nulle en cas de succès et zéro en cas d'échec.

Remarques

SetupTerrains() doit être appelé auparavant pour régler les paramètres par défaut du nouveau terrain. Après la création du terrain, de nouvelles parcelles (tiles) peuvent être définies par DefineTerrainTile() et les textures peuvent être appliquées avec AddTerrainTexture() . Une fois la définition du terrain terminée, BuildTerrain() doit être appelé pour le construire.

Voir aussi

FreeTerrain() , SetupTerrains() , BuildTerrain() , DefineTerrainTile() , AddTerrainTexture()

OS Supportés

Tous

149.5 CreateTerrainBody

Syntaxe

```
CreateTerrainBody(#Terrain,
    Restitution, Friction)
```

Description

Ajoute un corps physique statique au terrain.

Arguments

#**Terrain** Le terrain à utiliser.

Restitution Le facteur de restitution.
Si défini à 0.0, le terrain ne restituera aucune force, ce qui signifie que l'objet qui entre en collision avec lui ne rebondira pas sur le terrain. Avec une valeur supérieure à 0.0, il restituera une certaine force à l'objet qui va rebondir en cas de collision avec le terrain. Plus la valeur du coefficient de restitution est élevée et plus le rebond sera important.

Friction La force de frottement.
Si défini à 0.0, l'objet glissera sur le terrain sans perdre de force. Avec une valeur supérieure à 0.0, l'objet perdra de la vitesse (freinage) en cas de collision avec le terrain. Plus la valeur de la force de frottement est élevée et plus la vitesse décroît rapidement (décélération).

Valeur de retour

Aucune.

Remarques

Cela permet aux objets physiques d'entrer en collision avec le terrain.

Voir aussi

CreateTerrain() , FreeTerrainBody()

OS Supportés

Tous

149.6 DefineTerrainTile

Syntaxe

```
Resultat =
    DefineTerrainTile(#Terrain,
    ParcelleX, ParcelleY,
    NomCarteHauteur$,
    RetournementX,
    RetournementY)
```

Description

Définit le contenu d'une parcelle du terrain final.

Arguments

#**Terrain** Le terrain à utiliser.

ParcelleX, ParcelleY L'indice en x et en y de la parcelle, par rapport à la parcelle centrale du terrain.
La parcelle centrale commence à 0.0.
La valeur peut être négative.

RetournementX, RetournementY
#True : L'image sera renversée.
#False: Aucun retournement ne sera effectué.

Valeur de retour

Renvoie une valeur non nulle en cas de succès et zéro en cas d'échec.

Voir aussi

CreateTerrain()

OS Supportés

Tous

149.7 AddTerrainTexture

Syntaxe

```
AddTerrainTexture(#Terrain,
    Couche, TailleMonde,
    NomCarteDiffusionSpeculaire$,
    NomCarteHauteurNormale$)
```

Description

Ajoute une texture au terrain.

Arguments

#**Terrain** Le terrain à utiliser.

Couche L'indice de la couche (calque) sur laquelle la texture est appliquée.
L'indice de la première couche commence à 0.
Le nombre maximum de couches est défini lors de la création du terrain, voir CreateTerrain() .

TailleMonde La taille du monde de la texture.

NomCarteDiffusionSpeculaire$ Le nom de la carte (image) de diffusion spéculaire à appliquer sur cette couche.
Le mot spéculaire désigne la direction dans laquelle la lumière se réfléchit d'après les lois de Descartes.

NomCarteHauteurNormale$ Le nom de la carte (image) de hauteur normale (perpendiculaire) à appliquer sur cette couche.

Valeur de retour

Aucune.

Voir aussi

CreateTerrain()

OS Supportés

Tous

149.8 BuildTerrain

Syntaxe

```
BuildTerrain(#Terrain)
```

Description

Construit un terrain.

Arguments

#**Terrain** Le terrain à construire.

Valeur de retour

Aucune.

Remarques

Avant de construire un terrain, les parcelles doivent être définies par DefineTerrainTile() et les textures ajoutées avec AddTerrainTexture() .

Voir aussi

CreateTerrain() , DefineTerrainTile() , AddTerrainTexture()

OS Supportés

Tous

149.9 TerrainLocate

Syntaxe

```
TerrainLocate(#Terrain, X, Y,
    Z)
```

Description

Modifie l'emplacement absolu d'un terrain dans le monde.

Arguments

#Terrain Le terrain à utiliser.

X, Y, Z Nouvel emplacement absolu dans le monde (dans l'unité du monde).

Valeur de retour

Aucune.

Voir aussi

CreateTerrain()

OS Supportés

Tous

149.10 TerrainHeight

Syntaxe

```
Resultat =
    TerrainHeight(#Terrain, X,
    Z)
```

Description

Renvoie la hauteur d'un terrain à la position spécifiée dans le monde.

Arguments

#Terrain Le terrain à utiliser.

X, Z La position en 'X' et en 'Z' dans le monde (en unité du monde) de la hauteur du terrain.

Valeur de retour

Renvoie la hauteur du terrain (en unité du monde) à la position spécifiée.
Si le terrain ne se trouve pas à la position spécifiée, le résultat sera nul.

Voir aussi

CreateTerrain()

OS Supportés

Tous

149.11 TerrainTileHeightAtPosition

Syntaxe

```
Resultat =
    TerrainTileHeightAtPosition(#Terrain,
    ParcelleX, ParcelleY,
    Couche, X, Y)
```

Description

Renvoie la hauteur de la parcelle d'un terrain aux coordonnées spécifiées.

Arguments

#Terrain Le terrain à utiliser.

ParcelleX, ParcelleY L'indice 'X' et 'Y' de la parcelle.

Couche L'indice de la couche.
L'indice de la première couche commence à 0.
Le nombre maximum de couches est défini lors de la création du terrain, voir CreateTerrain() .

X, Y La position en 'X' et en 'Y' dans la parcelle, en pixels, de la hauteur du terrain.

Valeur de retour

Renvoie la hauteur de la parcelle du terrain (dans l'unité du monde) aux coordonnées spécifiées.

Voir aussi

CreateTerrain()

OS Supportés

Tous

149.12 TerrainTilePointX

Syntaxe

```
Resultat =
    TerrainTilePointX(#Terrain,
    ParcelleX, ParcelleY, X,
    Y, Z)
```

Description

Renvoie la position 'X' du point (X, Y, Z) dans la parcelle d'un terrain.

Arguments

#Terrain Le terrain à utiliser.

ParcelleX, ParcelleY L'indice 'X' et 'Y' de la parcelle.

X, Y, Z La position en X, Y, Z du point dans le monde, en unité du monde.

Valeur de retour

Renvoie la position en 'X', en pixels, du point dans la parcelle du terrain.

Voir aussi

CreateTerrain() , DefineTerrainTile()

OS Supportés

Tous

149.13 TerrainTilePointY

Syntaxe

```
Resultat =
    TerrainTilePointY(#Terrain,
    ParcelleX, ParcelleY, X,
    Y, Z)
```

Description

Renvoie la position 'Y' du point (X, Y, Z) dans la parcelle d'un terrain.

Arguments

#Terrain Le terrain à utiliser.

ParcelleX, ParcelleY L'indice 'X' et 'Y' de la parcelle.

X, Y, Z La position X, Y, Z du point dans le monde, en unité du monde.

Valeur de retour

Renvoie la position en 'Y', en pixels, du point dans la parcelle du terrain.

Voir aussi

CreateTerrain() , DefineTerrainTile()

OS Supportés

Tous

149.14 TerrainTileSize

Syntaxe

```
Resultat =
    TerrainTileSize(#Terrain,
    ParcelleX, ParcelleY)
```

Description

Renvoie la taille de la parcelle du terrain.

Arguments

#**Terrain** Le terrain à utiliser.

ParcelleX, ParcelleY L'indice 'X' et 'Y' de la parcelle.

Valeur de retour

Renvoie la taille, en pixels, de la parcelle du terrain.
Comme une parcelle est toujours carrée, la taille correspond à la largeur et à la hauteur de la parcelle.

Voir aussi

CreateTerrain() , DefineTerrainTile()

OS Supportés

Tous

149.15 GetTerrainTileHeightAtPoint

Syntaxe

```
Resultat =
    GetTerrainTileHeightAtPoint(#Terrain,
    ParcelleX, ParcelleY, X, Y)
```

Description

Renvoie la hauteur de la parcelle au point spécifié.

Arguments

#**Terrain** Le terrain à utiliser.

ParcelleX, ParcelleY L'indice 'X' et 'Y' de la parcelle.

X, Y La position en 'X' et en 'Y' dans la parcelle, en pixels.

Valeur de retour

Renvoie la hauteur de la parcelle de terrain, dans l'unité du monde, aux coordonnées spécifiées.

Voir aussi

CreateTerrain() , DefineTerrainTile()

OS Supportés

Tous

149.16 SetTerrainTileHeightAtPoint

Syntaxe

```
SetTerrainTileHeightAtPoint(#Terrain,
    ParcelleX, ParcelleY, X,
    Y, Hauteur)
```

Description

Définit la hauteur de la parcelle d'un terrain aux coordonnées spécifiées.

Arguments

#**Terrain** Le terrain à utiliser.

ParcelleX, ParcelleY L'indice 'X' et 'Y' de la parcelle.

X, Y La position en 'X' et en 'Y' dans la parcelle, en pixels.

Hauteur La nouvelle hauteur du terrain, dans l'unité du monde, à la position spécifiée dans la parcelle.

Valeur de retour

Aucune.

Remarques

Le changement ne sera pas pris en compte immédiatement, UpdateTerrain() doit être appelé une fois toutes les modifications faites.

Voir aussi

CreateTerrain() ,
GetTerrainTileHeightAtPoint() ,
UpdateTerrain()

OS Supportés

Tous

149.17 UpdateTerrain

Syntaxe

```
UpdateTerrain(#Terrain)
```

Description

Mises à jour d'un terrain.

Arguments

#**Terrain** Le terrain à utiliser.

Valeur de retour

Aucune.

Remarques

Nécessaire après la modification du terrain
avec des commandes comme
SetTerrainTileHeightAtPoint() .

Voir aussi

CreateTerrain() ,
SetTerrainTileHeightAtPoint()

OS Supportés

Tous

149.18 TerrainTileLayerMapSize

Syntaxe

```
Resultat =
    TerrainTileLayerMapSize(#Terrain,
    ParcelleX, ParcelleY)
```

Description

Renvoie la taille de la carte de mélange des
couches de la parcelle d'un terrain.

Arguments

#**Terrain** Le terrain à utiliser.

ParcelleX, ParcelleY L'indice 'X' et 'Y' de la parcelle.

Valeur de retour

Renvoie la taille (en pixels) de la carte de mélange du terrain.
Comme la carte de couche de mélange est toujours carrée, la taille correspond à la largeur et à la hauteur.

Voir aussi

CreateTerrain() , DefineTerrainTile()

OS Supportés

Tous

149.19 GetTerrainTileLayerBlend

Syntaxe

```
Resultat.f =
    GetTerrainTileLayerBlend(#Terrain,
    ParcelleX, ParcelleY,
    Couche, X, Y)
```

Description

Renvoie la valeur de mélange de la couche de la parcelle d'un terrain à la position spécifiée.

Arguments

#**Terrain** Le terrain à utiliser.

ParcelleX, ParcelleY L'indice 'X' et 'Y' de la parcelle.

Couche L'indice de la couche.
L'indice de la première couche commence à 0.
Le nombre maximum de couches est définie lors de la création du terrain, voir CreateTerrain() .

X, Y La position dans la parcelle, en pixels, de la valeur de mélange.

Valeur de retour

Renvoie la valeur de mélange de la couche de la parcelle à la position spécifiée.
La valeur de mélange varie de 0.0 (entièrement transparent) à 1.0 (complètement opaque).

Voir aussi

CreateTerrain()

OS Supportés

Tous

149.20 SetTerrainTileLayerBlend

Syntaxe

```
SetTerrainTileLayerBlend(#Terrain,
    ParcelleX, ParcelleY,
    Couche, X, Y, Valeur.f)
```

Description

Définit la valeur de mélange de la couche de la parcelle du terrain à la position spécifiée.

Arguments

#**Terrain** Le terrain à utiliser.

ParcelleX, ParcelleY L'indice 'X' et 'Y' de la parcelle.

Couche L'indice de la couche.
L'indice de la première couche commence à 0.
Le nombre maximum de couches est défini lors de la création du terrain, voir CreateTerrain() .

X, Y La position dans la parcelle, en pixels, de la valeur de mélange.

Valeur.f La valeur de mélange de couche de parcelle du terrain à la position spécifiée.
La valeur de mélange varie de 0.0 (entièrement transparent) à 1.0 (complètement opaque).

Valeur de retour

Aucune.

Remarques

Le changement ne sera pas pris en compte immédiatement, UpdateTerrainTileLayerBlend() doit être appelé une fois toutes les modifications éffectuées.

Voir aussi

CreateTerrain()

OS Supportés

Tous

149.21 UpdateTerrainTileLayerBlend

Syntaxe

```
UpdateTerrainTileLayerBlend(#Terrain,
    ParcelleX, ParcelleY,
    Couche)
```

Description

Met à jour la couche de mélange d'une parcelle d'un terrain.

Arguments

#Terrain Le terrain à utiliser.

ParcelleX, ParcelleY L'indice 'X' et 'Y' de la parcelle.

Couche L'indice de la couche à mettre à jour.
L'indice de la première couche commence à 0.
Le nombre maximum de couches est défini lors de la création du terrain, voir CreateTerrain() .

Valeur de retour

Aucune.

Remarques

Ceci est nécessaire après la modification de la couche de mélange avec la valeur SetTerrainTileLayerBlend() .

Voir aussi

CreateTerrain() ,
SetTerrainTileLayerBlend()

OS Supportés

Tous

149.22 TerrainMousePick

Syntaxe

```
Resultat =
    TerrainMousePick(#Terrain,
    CameraID, X, Y)
```

Description

Simule un clic de souris sur un terrain au point 2D spécifié sur la caméra spécifiée.

Arguments

#Terrain Le terrain à utiliser.

CameraID La caméra à utiliser.
Pour obtenir un identifiant valide, voir CameraID() .

X, Y La position du point ,en pixels, dans le champ de la caméra.

Valeur de retour

Si le terrain a reçu un clic simulé de souris, 'Resultat' vaudra #True, sinon il vaudra #False.
Pour obtenir des informations sur la position du clic, utilisez PickX() , PickY() et PickZ() .

Voir aussi

CreateTerrain() , CreateCamera()

OS Supportés

Tous

149.23 SaveTerrain

Syntaxe

```
SaveTerrain(#Terrain,
    Modification)
```

Description

Enregistre le terrain sur disque en utilisant le nom de fichier et l'extension définis par CreateTerrain() .

Arguments

#Terrain Le terrain à sauvegarder.

Modification #True : Le terrain ne sera sauvé que s'il a été modifié. #False: Il sera toujours sauvé.

Valeur de retour

Aucune.

Voir aussi

CreateTerrain()

OS Supportés

Tous

149.24 TerrainRenderMode

Syntaxe

```
TerrainRenderMode(#Terrain,
    Options)
```

Description

Modifie la façon dont le terrain est rendu.

Arguments

#**Terrain** Le terrain à utiliser.

Options Peut être une combinaison des valeurs suivantes :

```
#PB_Terrain_CastShadows
 : Active la projection
 des ombres dynamiques
 sur le terrain (peut
 être lent).
#PB_Terrain_LowLODShadows
 : Active la projection
 des ombres de faible
 qualité (pour avoir un
 rendu plus rapide).
```

Valeur de retour

Aucune.

Voir aussi

CreateTerrain()

OS Supportés

Tous

Chapitre 150

Text3D

Généralités

La bibliothèque Text3D est destinée à faciliter l'affichage du texte dans le monde de la 3D. Il est possible d'utiliser n'importe quelle police, le texte peut être déplacé ou sa taille peut changer en fonction de l'objet à suivre.
InitEngine3D() doit être appelé avec succès avant d'utiliser ces fonctions.

OS Supportés

Tous

150.1 CreateText3D

Syntaxe

```
Resultat =
    CreateText3D(#Texte3D,
    Texte$ [, Police$,
    Hauteur, Couleur])
```

Description

Crée un nouveau texte 3D.

Arguments

#**Texte3D** Le numéro du nouveau texte 3D.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Texte$ Le texte à afficher.
Il peut être modifié avec Text3DCaption() .

Police$ (optionnel) Le nom de la police à utiliser.
La police doit être présente dans le chemin Add3DArchive() et doit être définie dans le fichier 'proper-definitions.fontdef'.

Hauteur (optionnel) La hauteur de la police à utiliser.

Couleur (optionnel) La couleur RGBA à utiliser.
Pour obtenir une couleur valide, utiliser RGBA() .
La couleur peut être changée avec Text3DColor() .

Valeur de retour

Renvoie une valeur non nulle si le texte a été créé avec succès ou zéro en cas d'erreur. Si #PB_Any a été utilisé pour le paramètre #Texte3D alors le nouveau numéro généré est renvoyé en cas de succès.

Remarques

Pour être affiché, le texte doit être relié à un LibraryLink "node" "noeud" or an entité .

Exemple

```
  ; Initialisation du monde 3D
InitEngine3D()
InitSprite()
InitKeyboard()
InitMouse()
Add3DArchive(#PB_Compiler_Home + "examples/3d/Data/Textures", #PB_3DArchive_FileSystem)
Add3DArchive(#PB_Compiler_Home + "examples/3d/Data/fonts", #PB_3DArchive_FileSystem)
Parse3DScripts()

  ; Ouverture de la fenêtre
OpenWindow(0,0,0,1000,1000,"Texte 3D",#PB_Window_ScreenCentered|#PB_Window_SystemMenu)
OpenWindowedScreen(WindowID(0),0,0,1000,1000,1,0,0)

  ; Création du monde 3D
CreateSphere(0, 2)
CreateMaterial(0, LoadTexture(0, "clouds.jpg"))
CreateEntity(0, MeshID(0), MaterialID(0))
CreateCamera(0, 0, 0, 100, 100)
MoveCamera(0, 0, 0, 10, #PB_Absolute)

  ; Création d'un texte 3D
```

```
CreateText3D(0,
   FormatDate("%hh:%ii:%ss",
   Date()))  ; Création du
   texte 3D
Text3DColor(0, RGBA(255, 0,
   0, 255))                ;
   Couleur et transparence
Text3DAlignment(0,
   #PB_Text3D_HorizontallyCentered)
   ; Alignement du texte
AttachEntityObject(0, "",
   Text3DID(0))
   ; Il est obligatoire de
   lier un texte 3D à une
   entité (ou un noeud)
MoveText3D(0, 0.5, 1, 2)

   ; Position du texte 3D
ScaleText3D(0,0.8,0.2,0.5,#PB_Absolute)
               ; Dimension du
   texte 3D


  ; Gestion de la fenêtre, du
   clavier et de la souris
 Repeat
  Repeat
    Event  = WindowEvent()
    Select Event
      Case
    #PB_Event_CloseWindow
    End
    EndSelect
  Until Event = 0

  ExamineKeyboard()
  ExamineMouse()

  If MouseDeltaX()>0
    MoveEntity(noeud,0.5,0,0)
  ElseIf MouseDeltaX()<0
    MoveEntity(noeud,-0.5,0,0)
  EndIf
  If MouseDeltaY()>0
    MoveEntity(noeud,0,-0.5,0)
  ElseIf MouseDeltaY()<0
    MoveEntity(noeud,0,0.5,0)
  EndIf
  If
   MouseButton(#PB_MouseButton_Left)
   <>0
    End
  EndIf
  If MouseButton(
   #PB_MouseButton_Right) <>0
    MoveEntity(noeud,0,0,0,#PB_Absolute)
  EndIf
  tiks = MouseWheel()
  If tiks > 0
```

```
61      MoveEntity(noeud,0,0,1)
62
63    EndIf
64    If tiks < 0
65      MoveEntity(noeud,0,0,-1)
66    EndIf
67    If
       KeyboardPushed(#PB_Key_Escape)
68      quitter + 1
69    EndIf
70
71    If
       KeyboardPushed(#PB_Key_Space)
72      If rotation=1
73        rotation=0
74      Else
75        rotation=1
76      EndIf
77    EndIf
78
79    ; Barre espace = ON/OFF
       rotation de l'entité avec
       le texte 3d attaché à elle
80    If rotation =1
81      RotateEntity(0, 1, 1, 1,
       #PB_Relative)
82    EndIf
83
84    ; Mise à jour de l'heure
85    Text3DCaption(0,
       FormatDate("%hh:%ii:%ss",
       Date()))
86
87    ; Affichage de la scène
88    StartDrawing(WindowOutput(0))
89    DrawText(0,5,"Souris et
       Clavier: Espace et Molette
       souris")
90    DrawText(0,30,".: Echap ou
       clic gauche pour quitter
       :.")
91    StopDrawing()
92
93    RenderWorld()
94
95    FlipBuffers()
96
97  Until
      KeyboardPushed(#PB_Key_Escape)
      Or Quitter = 1
98
99  End
```

Voir aussi

FreeText3D()

OS Supportés

Tous

150.2 FreeText3D

Syntaxe

```
FreeText3D(#Texte3D)
```

Description

Libère un texte 3D.

Arguments

#Texte3D Le texte 3D à libérer.
Si #PB_All est spécifié, tous les textes 3D restants sont libérés.

Valeur de retour

Aucune.

Remarques

Une fois que le texte est libéré, il ne peut plus être utilisé.
Tout les textes restants sont automatiquement libérés lorsque le programme se termine.

Voir aussi

CreateText3D()

OS Supportés

Tous

150.3 Text3DID

Syntaxe

```
Resultat = Text3DID(#Texte3D)
```

Description

Renvoie l'identifiant système unique d'un texte 3D.

Arguments

#Texte3D Le texte 3D à utiliser.

Valeur de retour

Renvoie l'identifiant système unique du texte 3D.

OS Supportés

Tous

150.4 IsText3D

Syntaxe

```
Resultat = IsText3D(#Texte3D)
```

Description

Teste si le nombre donné est un texte 3D valide et correctement initialisé.

Arguments

#Texte3D Le texte 3D à tester.

Valeur de retour

Renvoie une valeur non nulle si #Texte3D est un texte valide ou zéro sinon.

Remarques

Cette fonction est robuste et peut être utilisée avec n'importe quelle valeur. C'est une bonne façon de s'assurer que le texte est prêt à être utilisé.

Voir aussi

CreateText3D()

OS Supportés

Tous

150.5 MoveText3D

Syntaxe

```
MoveText3D(#Texte3D, X.f,
    Y.f, Z.f [, Mode])
```

Description

Déplace un texte 3D.

Arguments

#Texte3D Le texte 3D à utiliser.

X.f, Y.f, Z.f La nouvelle position du texte.

Mode (optionnel) Le mode de déplacement. Peut être une des valeurs suivantes :

```
#PB_Relative: Déplacement
 relatif, à partir de la
 position courante du
 texte (par défaut).
#PB_Absolute: Déplacement
 absolu vers la position
 spécifiée.
```

Combiné à l'une des valeurs suivantes :

```
#PB_Local : Déplacement
 local.
#PB_Parent: Déplacement
 par rapport à la
 position du parent.
#PB_World : Déplacement
 par rapport au monde.
```

Valeur de retour

Aucune.

Exemple

Voir CreateText3D()

Voir aussi

ScaleText3D()

OS Supportés

Tous

150.6 ScaleText3D

Syntaxe

```
ScaleText3D(#Texte3D, X.f,
    Y.f, Z.f [,Mode])
```

Description

Change la taille d'un texte 3D suivant les 3 directions de l'espace.

Arguments

#Texte3D Le texte 3D à utiliser.

X.f, Y.f, Z.f Les facteurs d'échelle (agrandissement ou rapetissement) en X, Y et Z.

Mode (optionnel) Le mode du changement de taille. Peut être une des valeurs suivantes :

```
#PB_Relative: Changement
 relatif, sur la base de
 la taille initiale (par
 défaut). Une valeur de
 1.0 ne change pas la
 taille.
#PB_Absolute: Changement
 absolu, dans l'unité du
 monde.
```

Valeur de retour

Aucune.

Remarques

Avec le mode #PB_Relative, la taille sera multipliée par la valeur donnée.

Exemple

Voir CreateText3D()

Voir aussi

MoveText3D()

OS Supportés

Tous

150.7 Text3DCaption

Syntaxe

```
Text3DCaption(#Texte3D,
    Texte$)
```

Description

Modifie le texte affiché d'un texte 3D.

Arguments

#Texte3D Le texte 3D à utiliser.
Texte$ Le nouveau texte à afficher.

Valeur de retour

Aucune.

Exemple

Voir CreateText3D()

Voir aussi

CreateText3D()

OS Supportés

Tous

150.8 Text3DColor

Syntaxe

```
Text3DColor(#Texte3D, Couleur)
```

Description

Change la couleur d'un texte 3D.

Arguments

#Texte3D Le texte 3D à utiliser.

Couleur La nouvelle couleur RGBA() et transparence du texte.

Valeur de retour

Aucune.

Exemple

Voir CreateText3D()

Voir aussi

CreateText3D()

OS Supportés

Tous

150.9 Text3DAlignment

Syntaxe

```
Text3DAlignment(#Texte3D,
    Alignement)
```

Description

Modifie l'alignement d'un texte 3D.

Arguments

#Texte3D Le texte 3D à utiliser.

Alignement L'alignement du texte. Peut être une combinaison des valeurs suivantes :

```
#PB_Text3D_Left   : Le
 texte sera aligné à
 gauche
#PB_Text3D_Top    : Le
 texte sera aligné en haut
#PB_Text3D_Bottom: Le
 texte sera aligné en bas
#PB_Text3D_HorizontallyCentered:
 Le texte sera centré
 horizontalement
#PB_Text3D_VerticallyCentered
  : Le texte sera centré
 verticalement
```

Valeur de retour

Aucune.

Exemple

Voir CreateText3D()

Voir aussi

CreateText3D()

OS Supportés

Tous

150.10 Text3DX

Syntaxe

```
Resultat = Text3DX(#Texte3D)
```

Description

Renvoie la position absolue 'x' du texte dans le monde.

Arguments

#Text3D Le texte 3D à utiliser.

Valeur de retour

Renvoie la position absolue 'x' du texte.

Voir aussi

Text3DY() , Text3DZ() , MoveText3D()

OS Supportés

Tous

150.11 Text3DY

Syntaxe

```
Resultat = Text3DY(#Text3D)
```

Description

Renvoie la position absolue 'y' du texte dans le monde.

Arguments

#**Text3D** Le texte 3D à utiliser.

Valeur de retour

Renvoie la position absolue 'y' du texte.

Voir aussi

Text3DX() , Text3DZ() , MoveText3D()

OS Supportés

Tous

150.12 Text3DZ

Syntaxe

```
Resultat = Text3DZ(#Text3D)
```

Description

Renvoie la position absolue 'z' du texte dans le monde.

Arguments

#**Text3D** Le texte 3D à utiliser.

Valeur de retour

Renvoie la position absolue 'z' du texte.

Voir aussi

Text3DX() , Text3DY() , MoveText3D()

OS Supportés

Tous

Chapitre 151

Texture

Généralités

Les textures permettent aux objets 3D (Meshs) d'avoir un aspect réaliste.
En effet, sans textures les objets 3D seraient affichés avec une seule couleur.
PureBasic offre la possibilité de créer des textures directement à l'aide des outils 2D de base (bibliothèque 2DDrawing) ou de les charger à partir de fichiers.
InitEngine3D() doit être appelé avec succès avant de pouvoir utiliser les commandes relatives aux textures.

OS Supportés

Tous

151.1 CopyTexture

Syntaxe

```
Resultat =
    CopyTexture(#Texture,
    #NouvelleTexture)
```

Description

Copie une texture.

Arguments

#Texture La texture à copier.

#NouvelleTexture Le numéro de la nouvelle texture.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Si #NouvelleTexture a déjà été créée, elle sera libérée automatiquement et remplacée par la nouvelle.

Voir aussi

CreateTexture()

OS Supportés

Tous

151.2 CreateTexture

Syntaxe

```
Resultat =
    CreateTexture(#Texture,
    Largeur, Hauteur [,
    NomTexture$])
```

Description

Crée une nouvelle texture.

Arguments

#Texture Le numéro de la nouvelle texture.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Largeur, Hauteur Les dimensions de la texture, en pixels.

NomTexture$ (optionnel) Le nom de la nouvelle texture dans le système OGRE. Cela permet d'utiliser ce nom dans le script pour assigner un shader à cette texture.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si la texture a été déjà créée, elle est automatiquement libérée et remplacée par la nouvelle.

Remarques

Il est préférable que les dimensions des textures soient une puissance de 2 et carrée si possible, pour que le rendu soit optimal. Par exemple : 64*64, 128*128, 256*256, mais aussi 128x64, 16x32... etc. En effet, les anciennes cartes graphiques ont des limitations strictes quant à la taille des textures, et il est conseillé de se limiter à des

textures de 256*256 pour une compatibilité maximale. Le fait d'utiliser une texture en haute définition permet un rendu magnifique sur une carte graphique actuelle mais causera un ralentissement, parfois très important sur une carte ancienne.
L'utilisation de textures transparentes (mode Alpha) est possible.

Exemple

```
1   CreateTexture(0, 256, 256)
      ; Crée une nouvelle
      texture de dimension
      256x256.
```

Voir aussi

CopyTexture() , CreateRenderTexture()

OS Supportés

Tous

151.3 CreateCubicTexture

Syntaxe

```
Resultat =
    CreateCubicTexture(#Texture,
    #Texture1, #Texture2,
    #Texture3, #Texture4,
    #Texture5, #Texture6)
```

Description

Crée une nouvelle texture cubique en utilisant les textures spécifiées.
Les textures cubiques sont utiles pour créer des reflets du monde.

Arguments

#**Texture** Un numéro pour identifier la nouvelle texture.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

#**Texture1** La texture à utiliser pour la première face de la texture cubique.

#**Texture2** La texture à utiliser pour la seconde face de la texture cubique.

#**Texture3** La texture à utiliser pour la troisième face de la texture cubique.

#**Texture4** La texture à utiliser pour la quatrième face de la texture cubique.

#**Texture5** La texture à utiliser pour la cinquième face de la texture cubique.

#**Texture6** La texture à utiliser pour la sixième face de la texture cubique.

Valeur de retour

Renvoie une valeur non nulle si la texture cubique a été créée avec succès, zéro sinon. Si #PB_Any a été utilisé pour le paramètre #Texture, le nombre généré est renvoyé en cas de succès.

Remarques

#PB_Material_EnvironmentMap doit être spécifié avec SetMaterialAttribute() pour activer la réflexion cubique.

Exemple

```
1   InitEngine3D() :
     InitSprite() :
     InitKeyboard()
2
3   OpenWindow(0, 0,0, 800,
     600, "Texture cubique et
     reflets - [Esc] pour
     quitter",
     #PB_Window_ScreenCentered)
4   OpenWindowedScreen(WindowID(0),
     0, 0, WindowWidth(0),
     WindowHeight(0), 0, 0, 0)
5
6   Add3DArchive(#PB_Compiler_Home
     +
     "Examples/3D/Data/Textures",
     #PB_3DArchive_FileSystem)
7   Add3DArchive(#PB_Compiler_Home
     +
     "Examples/3D/Data/Packs/desert.zip",
     #PB_3DArchive_Zip)
8   Parse3DScripts()
9
10  CreateCamera(0, 0, 0, 100,
     100)
11  MoveCamera(0,0,0,-8)
12  CameraLookAt(0,0,0,0)
13
14  CreateLight(0,$ffffff,
     -100, 100, 50)
15  AmbientColor($111111*2)
16  CameraBackColor(0,$880044)
17
18  SkyBox("desert07.jpg")
19
20  LoadTexture(0,"desert07_RT.jpg")
21  LoadTexture(1,"desert07_LF.jpg")
```

```
22   LoadTexture(2,"desert07_UP.jpg")
23   LoadTexture(3,"desert07_DN.jpg")
24   LoadTexture(4,"desert07_FR.jpg")
25   LoadTexture(5,"desert07_BK.jpg")
26   CreateCubicTexture(10, 0,
      1, 2, 3, 4, 5)
27
28   LoadTexture(11, "dirt.jpg")
29   CreateMaterial(0,
      TextureID(11))
30   AddMaterialLayer(0,
      TextureID(10),
      #PB_Material_Add)
31   SetMaterialAttribute(0,
      #PB_Material_EnvironmentMap,
      #PB_Material_ReflectionMap,
      1)
32
33   CreateTorus(0,2,1,32,32)
34   CreateEntity(0, MeshID(0),
      MaterialID(0))
35
36   Repeat
37     While WindowEvent():Wend
38     ExamineKeyboard()
39     RotateEntity(0,1,1,1,
      #PB_Relative)
40     RenderWorld()
41     FlipBuffers()
42   Until
      KeyboardReleased(#PB_Key_Escape)
```

OS Supportés

Tous

151.4 CreateRenderTexture

Syntaxe

```
Resultat =
    CreateRenderTexture(#Texture,
    CameraID, Largeur, Hauteur
    [, Options [,
    NomTextureRendu$]])
```

Description

Crée une texture de rendu.

Arguments

#**Texture** Le numéro de la nouvelle texture.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

CameraID Le numéro de la caméra à associer à la texture.
Cet ID peut être obtenu par CameraID() .

Largeur, Hauteur Les dimensions de la nouvelle texture, en pixels.

Options (optionnel)

```
      #PB_Texture_AutomaticUpdate:
    La texture est mise à
    jour automatiquement à
    chaque RenderWorld()
  (par défaut).
   #PB_Texture_ManualUpdate
    : La texture n'est pas
    mise à jour
    automatiquement,
    UpdateRenderTexture()
  doit
                                être
    appelé manuellement.
   #PB_Texture_CameraViewPort
    : La fenêtre caméra ne
    sera pas supprimée,
    utile pour toujours être
    en mesure
                                de
    faire une capture de la
    caméra.
```

NomTextureRendu$ (optionnel) Le nom de la nouvelle texture dans le système OGRE.
Cela permet d'utiliser ce nom dans le script pour assigner un shader à cette texture.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Remarques

La caméra associée à la texture rendra son point de vue directement sur la texture, sans être affichée à l'écran. Cela peut être très utile pour des objets qui affichent une partie de la scène comme un écran de télévision, un miroir, etc.
Note : TextureOutput() n'est pas supporté sur les textures rendues.

Voir aussi

UpdateRenderTexture() , SaveRenderTexture()

OS Supportés

Tous

151.5 UpdateRenderTexture

Syntaxe

```
UpdateRenderTexture(#Texture)
```

Description

Met à jour une texture avec la vue actuelle de la caméra.

Arguments

#Texture La texture à mettre à jour.

Valeur de retour

Aucune.

Remarques

Si la texture de rendu a été créée avec l'option #PB_Texture_AutomaticUpdate, cette fonction n'est pas nécessaire.

Voir aussi

CreateRenderTexture() , SaveRenderTexture()

OS Supportés

Tous

151.6 SaveRenderTexture

Syntaxe

```
Resultat =
    SaveRenderTexture(#Texture,
    NomFichier$)
```

Description

Sauvegarde le contenu d'une texture de rendu.

Arguments

#Texture La texture à utiliser.

NomFichier$ Le nom du fichier où la texture sera sauvée.
Cela peut être un chemin absolu ou relatif (dans le répertoire courant).

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Il peut être utile de faire des captures d'écran d'une scène particulière.
Le format de sauvegarde peut être "BMP", "PNG", "JPG", "TGA", etc.

Voir aussi

CreateRenderTexture()

OS Supportés

Tous

151.7 CreateCubeMapTexture

Syntaxe

```
Resultat =
    CreateCubeMapTexture(#Texture,
    Largeur, Hauteur,
    NomTexture$ [,
    CouleurFond])
```

Description

Crée une nouvelle texture cube map.
Une texture cube map refléchit le milieu qui l'entoure.

Arguments

#Texture Le numéro d'identification de la nouvelle texture.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Largeur, Hauteur Les dimensions de la nouvelle texture, en pixels.

NomTexture$ Le nom de la texture dans les scripts OGRE.
Cette texture doit être définie dans le script avec la bonne valeur pour avoir un cube mapping qui fonctionne.
Si 'NomTexture$' est réglé sur 'CubeMapTexture', un script matériau doit contenir la définition suivante (qui peut être adaptée en fonction de vos besoins) :

```
material CubeMapMaterial
{
  technique
```

```
    {
      pass
      {
        texture_unit
        {
          cubic_texture
    CubeMapTexture
    combinedUVW
          tex_address_mode
    clamp
          env_map
    cubic_reflection
        }
      }
    }
  }
```

CouleurFond (optionnel) La couleur de fond de la scène de cube mapping, si aucune SkyBox() n'est utilisée.
Pour obtenir une couleur valide, utiliser RGB() .

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Remarques

Cette texture doit se trouver dans un script OGRE.

Voir aussi

EntityCubeMapTexture()

OS Supportés

Tous

151.8 EntityCubeMapTexture

Syntaxe

```
Resultat =
    EntityCubeMapTexture(#Texture,
    #Entité)
```

Description

Applique la texture cube map à l' entité .

Arguments

#**Texture** La texture à utiliser.
La texture doit avoir été créée par CreateCubeMapTexture() .

#**Entité** L'entité sur laquelle s'applique la texture.
Une même texture peut être appliquée sur de nombreuses entités.

Valeur de retour

Aucune.

Remarques

L'entité sera le reflet du monde qui l'entoure.

Voir aussi

CreateCubeMapTexture()

OS Supportés

Tous

151.9 FreeTexture

Syntaxe

```
FreeTexture(#Texture)
```

Description

Supprime une texture.

Arguments

#**Texture** La texture à utiliser.
Si #PB_All est spécifié, toutes les textures restantes sont libérées.

Valeur de retour

Aucune.

Remarques

Toutes les textures restantes sont automatiquement supprimées quand le programme se termine.

Voir aussi

CreateTexture() , LoadTexture() .

OS Supportés

Tous

151.10 IsTexture

Syntaxe

```
Resultat = IsTexture(#Texture)
```

Description

Teste si une texture est correctement initialisée.

Arguments

#**Texture** La texture à tester.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.

Voir aussi

CreateTexture()

OS Supportés

Tous

151.11 GetScriptTexture

Syntaxe

```
Resultat =
    GetScriptTexture(#Texture,
    Nom$)
```

Description

Obtenir une texture définie dans un fichier de script OGRE.

Arguments

#**Texture** La texture à utiliser.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Nom$ Le nom de la texture dans le fichier script.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Les scripts sont chargés et parsés avec la fonction Parse3DScripts() .

Voir aussi

LoadTexture()

OS Supportés

Tous

151.12 LoadTexture

Syntaxe

```
Resultat =
    LoadTexture(#Texture,
    Fichier$)
```

Description

Charge une texture à partir d'un fichier.

Arguments

#Texture Le numéro de la texture à charger.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Fichier$ Le nom du fichier qui peut être au format PNG, TGA ou JPG.
Le fichier doit être accessible dans le chemin géré par Add3DArchive() .

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.
Si la texture était déjà créée, elle est automatiquement supprimée et remplacée par la nouvelle.

Remarques

Il est préférable que les dimensions des textures soient une puissance de 2 et carrée si possible, pour que le rendu soit optimal. Par exemple : 64*64, 128*128, 256*256, mais aussi 128x64, 16x32... etc. En effet, les anciennes cartes graphiques ont des limitations strictes quant à la taille des

textures, et il est conseillé de se limiter à des textures de 256*256 pour une compatibilité maximale. Le fait d'utiliser une texture en haute définition permet un rendu magnifique sur une carte graphique actuelle mais causera un ralentissement, parfois très important sur une carte ancienne.

Voir aussi

GetScriptTexture() , FreeTexture()

OS Supportés

Tous

151.13 TextureID

Syntaxe

```
Resultat = TextureID(#Texture)
```

Description

Renvoie l'identifiant unique d'une texture.

Arguments

#**Texture** Le numéro de la texture à utiliser.

Valeur de retour

Renvoie le numéro de la texture.

Remarques

Cet identifiant est notamment nécessaire pour les commandes suivantes : AddMaterialLayer() (), CreateMaterial() .

Voir aussi

TextureOutput()

OS Supportés

Tous

151.14 TextureHeight

Syntaxe

```
Resultat =
    TextureHeight(#Texture)
```

Description

Renvoie la hauteur d'une texture.

Arguments

#**Texture** La texture à utiliser.

Valeur de retour

Renvoie la hauteur en pixels de la texture.

Voir aussi

TextureWidth()

OS Supportés

Tous

151.15 TextureOutput

Syntaxe

```
Resultat =
    TextureOutput(#Texture)
```

Description

Renvoie le numéro d'une texture.

Arguments

#**Texture** La texture à utiliser.

Valeur de retour

Renvoie l'identifiant d'une texture nécessaire à StartDrawing() pour dessiner directement sur la texture.

Remarques

Les textures créées avec CreateRenderTexture() ne sont pas prises en charge.

Exemple

```
1  ...
2  StartDrawing(TextureOutput(#Texture))
3    ; dessiner ici ...
4  StopDrawing()
```

Voir aussi

TextureID() , StartDrawing() , CreateRenderTexture()

OS Supportés

Tous

151.16 TextureWidth

Syntaxe

```
Resultat =
    TextureWidth(#Texture)
```

Description

Renvoie la largeur d'une texture.

Arguments

#**Texture** La texture à utiliser.

Valeur de retour

Renvoie la largeur en pixels de la texture.

Voir aussi

TextureHeight()

OS Supportés

Tous

Chapitre 152

Thread

Généralités

Un thread est une partie d'un programme qui peut être lancée de manière asynchrone en arrière plan du programme. Cela signifie qu'il est possible d'effectuer certaines opérations longues (compression, manipulation d'images ...) sans interrompre l'ensemble du programme, et de laisser ainsi l'utilisateur effectuer d'autres tâches. Un thread s'exécute à l'intérieur du programme, il ne s'agit pas d'un autre processus. Lorsque le programme se termine, tous les threads sont détruits. Sous PureBasic, les threads sont des procédures appelées de manière asynchrone. Un thread tourne jusqu'à ce que la procédure se termine. PureBasic a une option spéciale de compilation pour créer des exécutables multi-threadés (Le switch /THREAD pour le compilateur en ligne de commande ou l'option "Créer un exécutable multi threadé" dans les options de compilation de l'IDE). Sans cette option, certaines fonctions (ainsi que la gestion des strings) sont plus rapides, mais inutilisables dans les threads. Il est toujours possible de créer des threads sans être dans ce mode, mais ce n'est pas recommandé, car même une simple manipulation de chaîne de caractères dans un thread peut être dangereux et doit être protégé. Activer cette option rend tout cela possible, mais au prix d'une diminution de la rapidité d'exécution. La décision de compiler en mode threadé doit donc être prise uniquement si le besoin s'en fait sentir. Note : Les threads doivent être utilisés avec précaution, car il est possible que des ressources partagées (mémoire, fichiers, variables etc..) soient accédées au même moment ce qui causera des résultats aléatoires. Les commandes relatives aux 'Mutex' peut être utilisées pour synchroniser ces ressources partagées.

Note : Ne pas utiliser DirectX dans un Thread ! C'est une limitation de MS Windows. Si vous avez besoin d'afficher des graphiques dans des Thread alors utiliser les bibliothèques Images et 2DDrawing .

OS Supportés

Tous

152.1 IsThread

Syntaxe

```
Resultat = IsThread(Thread)
```

Description

Détermine si le numéro du thread est un thread valide, et s'il est toujours en cours d'exécution.

Arguments

Thread Le thread à utiliser.

Valeur de retour

Renvoie une valeur non nulle si le thread est valide et en cours d'utilisation, zéro sinon.

Voir aussi

CreateThread()

OS Supportés

Tous

152.2 ThreadID

Syntaxe

```
Resultat = ThreadID(Thread)
```

Description

Renvoie l'identifiant système d'un thread.

Arguments

Thread Le thread à utiliser.

Valeur de retour

Renvoie l'identifiant système du thread spécifié.

Remarques

L'identifiant est parfois appelé 'Handle'. Pour plus d'informations consultez le chapitre Numéros et Identifiants (Handles) .

Voir aussi

CreateThread()

OS Supportés

Tous

152.3 CreateMutex

Syntaxe

```
Resultat = CreateMutex()
```

Description

Crée un nouveau mutex.

Arguments

Aucun.

Valeur de retour

Renvoie le numéro du mutex s'il a été créé avec succès, zéro sinon.

Remarques

L'objectif principal des mutex est de faire de la synchronisation entre les threads. Ils ne consomment pas beaucoup de temps processeur, mais ils ne fonctionnent qu'au sein du programme (il n'est pas possible de synchroniser plusieurs programmes différents). Un mutex est un objet qui ne peut être verrouillé que par un seul thread à la fois, donc il est utilisé pour protéger l'accès aux ressources partagées (pour éviter que plusieurs threads n'accèdent en même temps aux mêmes données).

Exemple

```
1 ; Exécutez ce programme une fois comme il se présente.
2 ; Vous verrez que les lignes imprimées seront
3 ; mélangées entre les threads. Puis supprimez
4 ; les commentaires devant les commandes de Mutex
```

```
5   ; et les chaînes seront imprimées dans l'ordre,
6   ; car un seul thread à la fois sera autorisé
7   ; à exécuter les commandes d'impression.
8   ;
9   Procedure SansMutex(*Numero)
10    Shared Mutex
11
12    For a = 1 To 5
13      ;LockMutex(Mutex)    ; supprimez le commentaire pour voir la différence
14
15      PrintN("Thread "+Str(*Numero)+" essaie d'imprimer 5 fois a la suite :")
16      For b = 1 To 5
17        Delay(50)
18        PrintN("Thread "+Str(*Numero)+" Ligne "+Str(b))
19      Next b
20
21      ;UnlockMutex(Mutex)  ; supprimez le commentaire pour voir la différence
22    Next a
23  EndProcedure
24
25  OpenConsole()
26  Mutex = CreateMutex()
27
28  thread1 = CreateThread(@SansMutex(), 1)
29  Delay(25)
30  thread2 = CreateThread(@SansMutex(), 2)
31  Delay(25)
32  thread3 = CreateThread(@SansMutex(), 3)
33
34  WaitThread(thread1)
35  WaitThread(thread2)
36  WaitThread(thread3)
37
38  Input()
```

Voir aussi

FreeMutex()

OS Supportés

Tous

152.4 CreateThread

Syntaxe

```
Resultat =
    CreateThread(@NomdeProcedure(),
    *Valeur)
```

Description

Crée un nouveau Thread.

Arguments

@NomdeProcedure() L'adresse de la procédure que vous souhaitez utiliser comme code du nouveau thread. N'oubliez pas de mettre le '@' devant pour obtenir le nom, et les '()'.

***Valeur** La valeur transmise à la procédure de thread en tant que paramètre. C'est à vous de décider de son utilisation.

Valeur de retour

Renvoie le numéro du Thread s'il a été créé avec succès, zéro sinon.

Remarques

Si le thread est créé correctement, la fonction renvoie le numéro du thread qui pourra être utilisé avec les autres commandes de la bibliothèque, telles que KillThread() , PauseThread() , etc.

Exemple

```
1  Procedure
     MaProcedureThread(*Valeur)
2      ; La variable '*Valeur'
     contiendra 23
3  EndProcedure
4
5  CreateThread(@MaProcedureThread(),
     23)
```

Exemple : Passer plusieurs paramètres à un thread

```
1   Structure Personne
2     Nom$
3     Age.b
4     Telephone.l
5   EndStructure
6
7   Procedure
     Thread(*Parametres.Personne)
8
9     ; Affiche les parametres
10    ;
11    Debug *Parametres\Nom$
12    Debug *Parametres\Age
13    Debug
     *Parametres\Telephone
14
15    ; Une fois qu'on n'en a
     plus besoin,
     ClearStructure() est
     nécessaire pour s'assurer
16    ; que les objets
     dynamiques (s'il y en a)
     sont correctement libérés.
17    ClearStructure(*Parametres,
     Personne)
18    FreeMemory(*Parametres)
19
20  EndProcedure
21
22  ; On utilise de la mémoire
     dynamique, donc cela
     continuera de fonctionner
     même si l'appel
23  ; est fait à l'interieur
     d'une procédure.
24  ;
25  *Parametres.Personne =
     AllocateMemory(SizeOf(Personne))
26  *Parametres\Nom$ = "Jean"
27  *Parametres\Age   = 30
28  *Parametres\Telephone =
     10203040
29
30  CreateThread(@Thread(),
     *Parametres) ; Crée le
     thread
31
32  Delay(2000)
```

Voir aussi

KillThread()

OS Supportés

Tous

152.5 FreeMutex

Syntaxe

```
FreeMutex(Mutex)
```

Description

Supprime un Mutex et sa mémoire associée.

Arguments

Mutex Le mutex à libérer.

Valeur de retour

Aucune.

Remarques

Un mutex doit être déverrouillé avant d'être supprimé. Pour être sûr de cela, il est préférable que tous les threads utilisant ce mutex soient terminés.

Voir aussi

CreateMutex()

OS Supportés

Tous

152.6 KillThread

Syntaxe

```
KillThread(Thread)
```

Description

Termine un thread.

Arguments

Thread Le thread à terminer.
(Cette valeur provient de CreateThread()
.)

Valeur de retour

Aucune.

Remarques

Cette fonction est très dangereuse, ne l'utilisez que si nécessaire, car quand un thread est tué, il n'a aucune chance de libérer ses propres ressources mémoires, sa pile, etc. Il est possible d'utilser un 'flag' comme une variable globale pour dire au thread de se terminer lui-même tout en prenant en charge la libération de ses propres ressources.

Exemple

```
1  ; Une procedure/thread qui ne se terminera
2  ; jamais. A ne pas faire, mais cela
3  ; montre comment KillThread fonctionne
4  Procedure ImprimeUnTruc(*Intervalle)
5    Repeat
6      Print(".")
7      Delay(*Intervalle)
8    ForEver
9  EndProcedure
10
11 If OpenConsole()
12   thread = CreateThread(@ImprimeUnTruc(), 500)
13   If thread
14     For i=0 To 10
15       Print("A")
16       Delay(999)
17
18       If i=5
19         KillThread(thread)
20       EndIf
21     Next
22   EndIf
23 EndIf
```

Voir aussi

CreateThread()

OS Supportés

Tous

152.7 LockMutex

Syntaxe

```
LockMutex(Mutex)
```

Description

Verrouillage d'un mutex.

Arguments

Mutex Le mutex à verrouiller.

Valeur de retour

Aucune.

Remarques

Cette fonction attend la disponibilité du mutex avant de le verrouiller et continue son exécution. Le prochain thread qui demandera le verrouillage de ce mutex ne pourra continuer son exécution que lorsque ce mutex sera déverrouillé.
Il est garanti que seulement un thread continue d'exécuter le code protégé par ce mutex et peut ainsi accéder aux ressources partagées en lecture et en écriture sans risque de corruption.
Si un autre thread appel LockMutex() alors que ce mutex est verrouillé, il sera stoppé et mis en attente jusqu'à ce que ce thread appelle UnlockMutex() pour déverrouiller le mutex.
Note : étant donné que LockMutex() attend indéfiniment que le mutex soit déverrouillé, il peut y avoir des situations de blocage si un appel à UnlockMutex() est oublié.

Exemple

```
1  ; Exécutez ce programme une fois comme il se présente.
2  ; Vous verrez que les lignes imprimées seront
3  ; mélangées entre les threads. Puis supprimez
4  ; les commentaires devant les commandes de Mutex
5  ; et les chaînes seront imprimées dans l'ordre,
6  ; car un seul thread à la fois sera autorisé
7  ; à exécuter les commandes d'impression.
8  ;
9  Procedure SansMutex(*Numero)
10   Shared Mutex
11
12   For a = 1 To 5
```

```
13        ;LockMutex(Mutex)     ;
      supprimez le commentaire
      pour voir la différence
14
15        PrintN("Thread
      "+Str(*Numero)+" essaie
      d'imprimer 5 fois a la
      suite :")
16        For b = 1 To 5
17          Delay(50)
18          PrintN("Thread
      "+Str(*Numero)+" Ligne
      "+Str(b))
19        Next b
20
21        ;UnlockMutex(Mutex)  ;
      supprimez le commentaire
      pour voir la différence
22      Next a
23    EndProcedure
24
25    OpenConsole()
26    Mutex = CreateMutex()
27
28    thread1 =
       CreateThread(@SansMutex(),
       1)
29    Delay(25)
30    thread2 =
       CreateThread(@SansMutex(),
       2)
31    Delay(25)
32    thread3 =
       CreateThread(@SansMutex(),
       3)
33
34    WaitThread(thread1)
35    WaitThread(thread2)
36    WaitThread(thread3)
37
38    Input()
```

Voir aussi

UnlockMutex() , CreateMutex()

OS Supportés

Tous

152.8 PauseThread

Syntaxe

```
PauseThread(Thread)
```

Description

Met en pause un thread.

Arguments

Thread Le thread à utiliser.
(Cette valeur provient de CreateThread()
.)

Valeur de retour

Aucune.

Remarques

Le thread peut être relancé avec
ResumeThread() .

Exemple

```
1   Procedure ImprimeUnTruc(*ok)
2     For i = 0 To 10
3       PrintN(".")
4       Delay(200)
5     Next
6   EndProcedure
7
8   If OpenConsole()
9     thread =
      CreateThread(@ImprimeUnTruc(),
      0)
10    If thread
11
12      Delay(100)
13      PauseThread(thread)
14      For i = 0 To 10
15        PrintN("A")
16        Delay(50)
17      Next
18
19      ; Relance le thread en
      lui donnant assez de temps
      pour terminer
20      ResumeThread(thread)
21      Delay(3000)
22    EndIf
23  EndIf
```

Voir aussi

ResumeThread() , CreateThread()

OS Supportés

Tous

152.9 ResumeThread

Syntaxe

```
ResumeThread(Thread)
```

Description

Relance un thread préalablement interrompu avec PauseThread() .

Arguments

Thread Le thread à utiliser. (Cette valeur provient de CreateThread() .)

Valeur de retour

Aucune.

Exemple

```
1   Procedure ImprimeUnTruc(*ok)
2     For i = 0 To 10
3       PrintN(".")
4       Delay(200)
5     Next
6   EndProcedure
7
8   If OpenConsole()
9     thread =
      CreateThread(@ImprimeUnTruc(),
      0)
10    If thread
11
12      Delay(100)
13      PauseThread(thread)
14      For i = 0 To 10
15        PrintN("A")
16        Delay(50)
17      Next
18
19      ; Relance le thread en
      lui donnant assez de temps
      pour terminer
20      ResumeThread(thread)
21      Delay(3000)
22    EndIf
23  EndIf
```

Voir aussi

PauseThread() , CreateThread()

OS Supportés

Tous

152.10 ThreadPriority

Syntaxe

```
Resultat =
    ThreadPriority(Thread,
    Priorite)
```

Description

Change la priorité du thread spécifié et renvoie l'ancienne priorité.

Arguments

Thread Le thread à utiliser.
(Cette valeur provient de CreateThread().)

Priorite Peut prendre une valeur entre 1 et 32.
0, aucun changement n'est effectué (utile pour seulement trouver le niveau de priorité du thread sans le modifier).
1 est la priorité la plus basse, 16 est une priorité normale et 32 correspond à un niveau de priorité en temps réel (la plus haute, à n'utiliser que si vous êtes sûr de ce que vous faites).
Windows ne supporte pas les 32 niveaux de priorité, voici le tableau correspondant :

```
- 1 : Le plus bas
- Entre 2 et 15 : En
 dessous de la normale
- 16: Normal
- Entre 17 et 30: Au
 dessus de la normale
- 31: Le plus haut
- 32: Temps critique
```

Valeur de retour

Renvoie la priorité du thread avant l'appel de cette commande. Ce qui peut être utile si vous voulez seulement augmenter la priorité du thread pour un court moment et ensuite revenir à l'ancienne valeur. La valeur renvoyée n'est pas nécessairement la même que celle donnée en paramètre de ThreadPriority(), car elle dépend de la granularité du réglage de la priorité offerte par le système.

Remarques

Un thread avec une haute priorité et qui utilise beaucoup de temps, comme le traitement d'image par exemple,

monopolisera le processeur et les autres threads auront peu de chance de s'exécuter. Réservez donc les hautes priorités aux threads qui demande peu de temps d'exécution.
Windows planifie les threads, il choisit celui qui tournera pendant un court laps de temps, en utilisant une stratégie préemptive de même niveau de priorité. Cela signifie que s'il est temps d'exécuter un autre thread (comme un changement de contexte), alors le thread avec la plus haute priorité disponible sera le prochain à être exécuté. S'il y a plus d'un thread avec la plus haute priorité alors ils seront alternés en fonction du contexte.

Exemple

```
1  ; Une procedure toujours en cours d'exécution
2  ; (aucune commande Delay n'est employée
3  ; car cela obligerait le thread à s'arrêter
4  ; pendant qu'il serait en attente)
5  Procedure ImprimeUnTruc(*Intervalle)
6    For i = 0 To 1000000000
7      ; occupé par une méchante attente
8    Next
9  EndProcedure
10
11 If OpenConsole()
12   thread = CreateThread(@ImprimeUnTruc(), 500)
13   If thread
14     ; Augmente la priorité au-dessus du thread principal
15     ; Vous devriez remarquer un délai avant que la commande d'impression
16     ; ne soit exécutée. Maintenant changez le 17 en 15 (plus bas que la priorité normale)
17     ; et observez que la commande d'impression s'exécute instantanément
18     ThreadPriority(thread, 17)
19     PrintN("Attend que le thread de plus haute priorite se termine")
```

```
20      EndIf
21
22      PrintN("Appuyer sur
        Entree pour quitter")
23      Input()
24    EndIf
```

OS Supportés

Windows

152.11 TryLockMutex

Syntaxe

```
Resultat = TryLockMutex(Mutex)
```

Description

Essaie de verrouiller un Mutex. Contrairement à LockMutex() , cette commande ne stoppe pas l'exécution du thread jusqu'à ce que le mutex soit libre. Ceci peut être pratique dans une situation ou le thread peut continuer à faire autre chose en attendant que le mutex soit libéré.

Arguments

Mutex Le mutex à verrouiller.

Valeur de retour

Renvoie une valeur non nulle si le mutex a été verrouillé, zéro sinon.

Remarques

Si le mutex a été verrouillé, la commande UnlockMutex() doit être appelée pour libérer le mutex une fois que les accès aux ressources partagées sont terminés. Ne pas le faire peut engendrer facilement des situations de blocage.

Exemple

```
1  Procedure
     ThreadProcedure(*Valeur)
2     Shared Mutex
3
4     Repeat
5       If TryLockMutex(Mutex)
6         PrintN("Mutex
        verrouille avec succes.")
7
8         UnlockMutex(Mutex)
```

```
9         Break ; quitte la
    boucle et le thread
10      Else
11        PrintN("Toujours en
    train d'attendre d'avoir
    accès au mutex...")
12        Delay(200)
13      EndIf
14     ForEver
15  EndProcedure
16
17  OpenConsole()
18
19  Mutex = CreateMutex()
20  LockMutex(Mutex) ; le
     programme principal
     verrouille en premier le
     mutex
21  Thread =
     CreateThread(@ThreadProcedure(),
     0)
22
23  Delay(4000)
24  UnlockMutex(Mutex) ;
     maintenant déverrouille le
     mutex, pour que le thread
     puisse y accéder
25
26  Input()
```

Voir aussi

UnlockMutex() , LockMutex() , CreateMutex()

OS Supportés

Tous

152.12 UnlockMutex

Syntaxe

```
UnlockMutex(Mutex)
```

Description

Déverrouille un mutex préalablement verrouillé avec LockMutex() .

Arguments

Mutex Le mutex à déverrouiller.

Valeur de retour

Aucune.

Remarques

Un mutex ne peut être déverrouillé que par le thread qui l'a bloqué.

Exemple

```
; Exécutez ce programme une fois comme il se présente.
; Vous verrez que les lignes imprimées seront
; mélangées entre les threads. Puis supprimez
; les commentaires devant les commandes de Mutex
; et les chaînes seront imprimées dans l'ordre,
; car un seul thread à la fois sera autorisé
; à exécuter les commandes d'impression.
;
Procedure SansMutex(*Numero)
  Shared Mutex

  For a = 1 To 5
    ;LockMutex(Mutex)    ; supprimez le commentaire pour voir la différence

    PrintN("Thread "+Str(*Numero)+" essaie d'imprimer 5 fois a la suite :")
    For b = 1 To 5
      Delay(50)
      PrintN("Thread "+Str(*Numero)+" Ligne "+Str(b))
    Next b

    ;UnlockMutex(Mutex)  ; supprimez le commentaire pour voir la différence
  Next a
EndProcedure

OpenConsole()
Mutex = CreateMutex()

thread1 = CreateThread(@SansMutex(), 1)
Delay(25)
thread2 = CreateThread(@SansMutex(), 2)
Delay(25)
```

```
32  thread3 =
     CreateThread(@SansMutex(),
     3)
33
34  WaitThread(thread1)
35  WaitThread(thread2)
36  WaitThread(thread3)
37
38  Input()
```

Voir aussi

LockMutex() , TryLockMutex() , CreateMutex()

OS Supportés

Tous

152.13 WaitThread

Syntaxe

```
Resultat = WaitThread(Thread
    [, Délai])
```

Description

Arrête l'exécution du programme jusqu'à ce que le 'Thread' se termine ou que le délai en option soit atteint. Si ce thread est déjà terminé le retour est immédiat.

Arguments

Thread Le thread à utiliser. (Cette valeur provient de CreateThread() .)

Délai (optionnel) Temps d'attente, en millisecondes.

Valeur de retour

Renvoie une valeur non nulle si le thread est terminé, zéro si le délai est atteint.

Exemple

```
1  Procedure
    ImprimeUnTruc(*Intervalle)
2    For i = 0 To 10
3      PrintN(".")
4      Delay(*Intervalle)
5    Next
6  EndProcedure
7
```

```
8   If OpenConsole()
9     thread =
      CreateThread(@ImprimeUnTruc(),
      500)
10    If thread
11      ; Attend que le thread
      se termine avant de
      continuer
12      ; Essayer de mettre en
      commentaire la commande
      WaitThread et regardez ce
      qu'il se produit
13      WaitThread(thread)
14
15      For i = 0 To 10
16        PrintN("A")
17        Delay(1000)
18      Next
19    EndIf
20  EndIf
```

Voir aussi

CreateThread() , PauseThread()

OS Supportés

Tous

152.14 CreateSemaphore

Syntaxe

```
Resultat =
    CreateSemaphore([CompteurInitial])
```

Description

Crée un nouveau sémaphore.

Arguments

CompteurInitial (optionnel) Ce doit être une valeur positive qui indique le compteur initial du sémaphore. S'il n'est pas spécifié, le compteur initial vaudra 0.

Valeur de retour

Renvoie le nouveau sémaphore, ou zéro en cas d'échec.

Remarques

Un sémaphore est un objet de synchronisation de threads, qui contient un compteur interne. Il gère deux types

d'opérations : attente et signalement . L'opération d'attente diminue la valeur du compteur du sémaphore de un. Si le compteur devait passer à une valeur inférieure à 0, l'opération d'attente sera bloquante, jusqu'à ce qu'une opération de signalement soit faite. Cette dernière augmente la valeur du compteur de un, libérant le thread bloqué (si il y en a un). Un sémaphore permet d'appliquer un comptage minimum et maximum entre les threads par exemple pour éviter à la file d'attente de manquer d'éléments ou au contraire d'en avoir trop.
Contrairement à un mutex , un sémaphore n'appartient pas à un thread particulier, ce qui permet aux opérations d'attente/signalement d'être effectuées depuis n'importe quel thread, ce qui n'est pas le cas de LockMutex() et UnlockMutex() . Une utilisation courante des sémaphores est la création d'un thread qui attend sur le sémaphore avec WaitSemaphore() et d'un autre qui le débloque avec SignalSemaphore() : c'est le pattern producteur/consommateur.

Exemple

Cet exemple montre un thread "producteur" qui ajoute des éléments à une liste et un thread principal qui affiche la liste au fur et à mesure que des éléments sont ajoutés. Le sémaphore permet au thread principal de se mettre en attente si la liste ne contient aucun élément. A noter que cela aurait pu aussi être effectué avec un mutex, en testant à intervalle régulier si la liste est vide ou non (en utilisant Delay()). Le sémaphore est bien plus efficace, car il débloque immédiatemment le thread lorsque le signalement est effectué, et élimine l'intervalle d'attente.

```
1  ; Assurez vous que l'option
     "Activer la gestion des
     threads" est sélectionnée
2  ;
3  Global Semaphore =
     CreateSemaphore()
4  Global Mutex     =
     CreateMutex()
5  Global NewList Queue()
6
7  Procedure Producer(Total)
8
9    For i = 1 To Total
10     Delay(Random(750) + 250)
11
```

```
12      ; L'accès à la liste
    nécessite tout de même un
    mutex pour être threadsafe
13      LockMutex(Mutex)
14        LastElement(Queue())
15        AddElement(Queue())
16        Queue() = i
17      UnlockMutex(Mutex)
18
19      ; Envoie un signal pour
    indiquer qu'un nouvel
    élément est disponible
20      SignalSemaphore(Semaphore)
21    Next i
22
23  EndProcedure
24
25  If
    CreateThread(@Producer(),
    30)
26
27    For i = 1 To 30
28      ; Attente d'un nouvel
    élément
29      WaitSemaphore(Semaphore)
30
31      ; Affiche l'état de la
    liste
32      LockMutex(Mutex)
33        Queue$ = "Queue:"
34        ForEach Queue()
35          Queue$ + " " +
    Str(Queue())
36        Next Queue()
37        Debug Queue$
38
39        ; Efface le premier
    élément de la liste
40        FirstElement(Queue())
41        DeleteElement(Queue())
42      UnlockMutex(Mutex)
43
44    Next i
45
46  EndIf
```

Voir aussi

FreeSemaphore()

OS Supportés

Tous

152.15 FreeSemaphore

Syntaxe

```
FreeSemaphore(Semaphore)
```

Description

Supprime le sémaphore et toutes les ressources associées.

Arguments

Semaphore Le sémaphore à supprimer.

Valeur de retour

Aucune.

Voir aussi

CreateSemaphore()

OS Supportés

Tous

152.16 SignalSemaphore

Syntaxe

```
SignalSemaphore(Semaphore)
```

Description

Augmente la valeur du compteur interne du sémaphore de un, ce qui a pour effet de débloquer le thread en attente sur ce sémaphore (si il y en a un).

Arguments

Semaphore Le sémaphore à signaler.

Valeur de retour

Aucune.

Remarques

Le compteur maximal d'un sémaphore est limité à une valeur 32 bits signée, donc un maximum de 2147483647 appels à SignalSemaphore() peuvent être effectués sans être contre-balancés par des appels à WaitSemaphore() .

```
1  ; Assurez vous que l'option
     "Activer la gestion des
     threads" est sélectionnée
2  ;
3  Global Semaphore =
     CreateSemaphore()
```

```
Global Mutex      =
 CreateMutex()
Global NewList Queue()

Procedure Producer(Total)

  For i = 1 To Total
    Delay(Random(750) + 250)

    ; L'accès à la liste
 nécessite tout de même un
 mutex pour être threadsafe
    LockMutex(Mutex)
      LastElement(Queue())
      AddElement(Queue())
      Queue() = i
    UnlockMutex(Mutex)

    ; Envoie un signal pour
 indiquer qu'un nouvel
 élément est disponible
    SignalSemaphore(Semaphore)
  Next i

EndProcedure

If
 CreateThread(@Producer(),
 30)

  For i = 1 To 30
    ; Attente d'un nouvel
 élément
    WaitSemaphore(Semaphore)

    ; Affiche l'état de la
 liste
    LockMutex(Mutex)
      Queue$ = "Queue:"
      ForEach Queue()
        Queue$ + " " +
 Str(Queue())
      Next Queue()
      Debug Queue$

      ; Efface le premier
 élément de la liste
      FirstElement(Queue())
      DeleteElement(Queue())
    UnlockMutex(Mutex)

  Next i

EndIf
```

Voir aussi

WaitSemaphore() , TrySemaphore()

OS Supportés

Tous

152.17 WaitSemaphore

Syntaxe

```
WaitSemaphore(Semaphore)
```

Description

Diminue la valeur du compteur interne du sémaphore de un, ce qui a pour effet de bloquer le thread, si la valeur venait à tomber en dessous de zéro. Le thread sera débloqué dès qu'un autre thread appellera SignalSemaphore() .

Arguments

Semaphore Le sémaphore à attendre.

Valeur de retour

Aucune.

Remarques

Le compteur maximal d'un sémaphore est limité à une valeur 32-bit signée, donc un maximum de 2147483647 appels à WaitSemaphore() peuvent être effectués sans être contre-balancés par des appels à SignalSemaphore() .

Exemple

```
1  ; Assurez vous que l'option
    "Activer la gestion des
    threads" est sélectionnée
2  ;
3  Global Semaphore =
    CreateSemaphore()
4  Global Mutex     =
    CreateMutex()
5  Global NewList Queue()
6
7  Procedure Producer(Total)
8
9    For i = 1 To Total
10     Delay(Random(750) + 250)
11
12     ; L'accès à la liste
    nécessite tout de même un
    mutex pour être threadsafe
13     LockMutex(Mutex)
14       LastElement(Queue())
```

```
15          AddElement(Queue())
16          Queue() = i
17        UnlockMutex(Mutex)
18
19        ; Envoie un signal pour
    indiquer qu'un nouvel
    élément est disponible
20        SignalSemaphore(Semaphore)
21      Next i
22
23    EndProcedure
24
25    If
      CreateThread(@Producer(),
      30)
26
27      For i = 1 To 30
28        ; Attente d'un nouvel
    élément
29        WaitSemaphore(Semaphore)
30
31        ; Affiche l'état de la
    liste
32        LockMutex(Mutex)
33          Queue$ = "Queue:"
34          ForEach Queue()
35            Queue$ + " " +
    Str(Queue())
36          Next Queue()
37          Debug Queue$
38
39          ; Efface le premier
    élément de la liste
40          FirstElement(Queue())
41          DeleteElement(Queue())
42        UnlockMutex(Mutex)
43
44      Next i
45
46    EndIf
```

Voir aussi

SignalSemaphore() , TrySemaphore()

OS Supportés

Tous

152.18 TrySemaphore

Syntaxe

```
Resultat =
    TrySemaphore(Semaphore)
```

Description

Diminue la valeur du compteur interne du sémaphore de un seulement si le compteur est supérieur à zéro. Cette commande a le même comportement que WaitSemaphore() , sauf qu'elle ne bloquera pas si le compteur devait tomber en dessous de zéro.

Arguments

Semaphore Le sémaphore à utiliser.

Valeur de retour

Renvoie une valeur non nulle si le compteur du sémaphore a été diminué, ou zéro si le compteur n'a pas pu être diminué (il était déjà à zéro).

Exemple

```
1  Global ThreadSemaphore =
       CreateSemaphore()
2
3  Procedure MyThread(val)
4
5    Debug "Thread ok"
6    Repeat
7      Debug "Thread En cours"
8      Debug "Pour quitter,
       revenir sur la console et
       appuyer sur 'Echap'"
9      ShowDebugOutput()
10     ClearDebugOutput()
11     Delay(2000)
12     For i = 1 To 5
13       Debug x
14       x + 1
15     Next i
16     Delay(1000)
17   Until
       TrySemaphore(ThreadSemaphore)
18   Debug "Thread Fermé"
19
20 EndProcedure
21
22 Thread =
       CreateThread(@MyThread(),
       0)
23
24
25 If OpenConsole()
26   PrintN("Appuyez sur une
       touche SVP.")
27   PrintN("Appuyez sur [Echap]
       pour quitter")
28
29   Repeat
```

```
30      KeyPressed$ = Inkey()
31
32      If KeyPressed$ <> ""
33
34        PrintN("Vous avez
        appuye sur : " +
        KeyPressed$)
35        PrintN("Son identifiant
        numerique est :
        "+Str(RawKey()))
36
37      ElseIf RawKey()
38
39        PrintN("Vous avez
        appuye sur une touche qui
        n'est pas
        alpha-numerique.")
40        PrintN("Son identifiant
        numerique est :
        "+Str(RawKey()))
41
42      Else
43        Delay(20) ; Evite de
        monopoliser tout le temps
        processeur. Utile pour un
        OS multi-tâches.
44      EndIf
45
46    Until KeyPressed$ = Chr(27)
        ; Attends jusqu'à ce que
        la touche [Echap] soit
        appuyée
47  EndIf
48
49  If IsThread(Thread)
50    SignalSemaphore(ThreadSemaphore)
51  EndIf
52  End
```

Voir aussi

WaitSemaphore() , SignalSemaphore()

OS Supportés

Tous

Chapitre 153

ToolBar

Généralités

Les barres d'outils (Toolbar) sont très utiles pour accéder rapidement aux fonctions d'une application, comme le copier-coller par exemple, en cliquant simplement sur une icône. Les barres d'outils représentent souvent des raccourcis de menus. PureBasic vous permet de créer un nombre quelconque de barres d'outils et de les gérer comme si elles étaient des menus.

OS Supportés

Tous

153.1 CreateToolBar

Syntaxe

```
Resultat =
    CreateToolBar(#BarreOutils,
    FenetreID [, Options])
```

Description

Crée une barre d'outils vide.

Arguments

#BarreOutils L'identifiant de la barre d'outils
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

FenetreID L'identifiant de la fenêtre. 'FenetreID' peut être récupéré facilement grâce à la commande WindowID() .

Options (optionnel) Peut être une combinaison des valeurs suivantes :

```
- #PB_ToolBar_Small      :
  Petites icônes (16x16)
  (par défaut)
```

```
- #PB_ToolBar_Large      :
  Grandes icônes (24x24)
- #PB_ToolBar_Text       :
  Le texte sera affiché
  sous le bouton
- #PB_ToolBar_InlineText:
  Le texte sera affiché à
  droite du bouton
  (Windows uniquement)
```

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

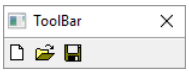
Remarques

Cette barre d'outils devient celle par défaut pour les créations des éléments qu'elle doit contenir. Vous pouvez maintenant utiliser ToolBarStandardButton() , ToolBarImageButton() et ToolBarSeparator() pour ajouter des éléments.
La détection des évènements sur les barres d'outils est similaire à celle des menus, et nécessite donc la commande EventMenu() .
Les barres d'outils sont souvent utilisées comme raccourci des éléments de menu, ainsi en attribuant le même numéro ID à un menu et à un bouton de la barre d'outils, les deux événements sont traités en utilisant le même code.

Exemple

```
1  If OpenWindow(0, 0, 0, 150, 25, "Barre d'outils", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2   If CreateToolBar(0, WindowID(0))
3     ToolBarStandardButton(0, #PB_ToolBarIcon_New)
4     ToolBarStandardButton(1, #PB_ToolBarIcon_Open)
5     ToolBarStandardButton(2, #PB_ToolBarIcon_Save)
6   EndIf
7   Repeat
8     Event = WaitWindowEvent()
9     If Event = #PB_Event_Menu
10      Debug "Identifiant Barre d'outils: "+Str(EventMenu())
11    EndIf
```

```
12      Until Event =
          #PB_Event_CloseWindow
13    EndIf
```



Voir aussi

ToolBarStandardButton() ,
ToolBarImageButton() ,
ToolBarSeparator() , FreeToolBar()

OS Supportés

Tous

153.2 FreeToolBar

Syntaxe

```
FreeToolBar(#BarreOutils)
```

Description

Supprime une barre d'outils.

Arguments

#BarreOutils La barre d'outils à supprimer.
Si #PB_All est spécifiée, toutes les barres d'outils restantes sont libérées.

Valeur de retour

Aucune.

Remarques

Toutes les barres d'outils restantes sont automatiquement supprimées quand le programme se termine.

Voir aussi

CreateToolBar()

OS Supportés

Tous

153.3 DisableToolBarButton

Syntaxe

```
DisableToolBarButton(#BarreOutils,
    Bouton, Etat)
```

Description

Active ou désactive un bouton d'une barre d'outils.

Arguments

#BarreOutils La barre d'outils à supprimer.

Bouton Le bouton à activer ou à désactiver.

Etat `#False : Le bouton est activé.`
`#True : Le bouton est désactivé.`

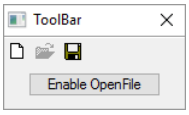
Valeur de retour

Aucune.

Exemple

```
1   If OpenWindow(0, 0, 0, 150,
     60, "Barre d'outils",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2     If CreateToolBar(0,
    WindowID(0))
3       ToolBarStandardButton(0,
    #PB_ToolBarIcon_New)
4       ToolBarStandardButton(1,
    #PB_ToolBarIcon_Open)
5       ToolBarStandardButton(2,
    #PB_ToolBarIcon_Save)
6       DisableToolBarButton(0,
    1, 1) : Disabled = #True
7     EndIf
8
9     ButtonGadget(0, 10, 30,
    130, 20, "Active l'icône
    'Ouvrir'")
10
11    Repeat
12      Event =
    WaitWindowEvent()
13      If Event =
    #PB_Event_Gadget
14        If EventGadget() = 0
15          If Disabled = #True
16            DisableToolBarButton(0,
    1, 0)
```

```
17              SetGadgetText(0,"Désactive
    l'icône 'Ouvrir'")
18              Disabled = #False
19            Else
20              DisableToolBarButton(0,
    1, 1)
21              SetGadgetText(0,"Active
    l'icône 'Ouvrir'")
22              Disabled = #True
23            EndIf
24          EndIf
25        EndIf
26      Until Event =
    #PB_Event_CloseWindow
27    EndIf
```



Voir aussi

ToolBarStandardButton() ,
ToolBarImageButton()

OS Supportés

Tous

153.4 GetToolBarButtonState

Syntaxe

```
Resultat =
    GetToolBarButtonState(#BarreOutils,
    Bouton)
```

Description

Renvoie l'état d'un Bouton.

Arguments

#BarreOutils La barre d'outils à utiliser.
Bouton Le bouton à tester.

Valeur de retour

Renvoie une valeur non nulle si le bouton de la barre d'outils est activé (enfoncé), zéro sinon.

Remarques

Attention, le bouton doit avoir été créé avec l'option #PB_ToolBar_Toggle.
SetToolBarButtonState() permet de changer l'état d'un bouton.

Voir aussi

SetToolBarButtonState()

OS Supportés

Tous

153.5 IsToolBar

Syntaxe

```
Resultat =
    IsToolBar(#BarreOutils)
```

Description

Teste si une barre d'outils est correctement initialisée.

Arguments

#**BarreOutils** La barre d'outils à utiliser.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage.

Voir aussi

CreateToolBar()

OS Supportés

Tous

153.6 SetToolBarButtonState

Syntaxe

```
SetToolBarButtonState(#BarreOutils,
    Bouton, Etat)
```

Description

Change l'état d'un Bouton.

Arguments

#BarreOutils La barre d'outils à utiliser.

Bouton Le bouton à utiliser.

```
Etat   #False : Etat relâché
  #True  : Etat enfoncé
```

Valeur de retour

Aucune.

Remarques

Attention, le bouton doit avoir été créé avec l'option #PB_ToolBar_Toggle.
GetToolBarButtonState() permet de connaître l'état d'un bouton.

OS Supportés

Tous

153.7 ToolBarHeight

Syntaxe

```
Resultat =
    ToolBarHeight(#BarreOutils)
```

Description

Renvoie la hauteur en pixel d'une barre d'outils.

Arguments

#BarreOutils La barre d'outils à utiliser.

Valeur de retour

Renvoie la hauteur en pixel de la barre d'outils.

Remarques

C'est utile pour calculer correctement la hauteur d'une fenêtre lorsqu'une barre d'outils est utilisée.
Sous OS X cette commande renvoie 0, car la barre d'outils ne fait pas partie de l'intérieur de la fenêtre, aucun calcul n'est nécessaire.

Voir aussi

CreateToolBar()

OS Supportés

Tous

153.8 ToolBarImageButton

Syntaxe

```
ToolBarImageButton(#Bouton,
    ImageID [, Mode [,
    Texte$]])
```

Description

Ajoute un bouton image à une barre d'outils en cours de création.

Arguments

#Button Le numéro du nouveau bouton.

ImageID L'identifiant de l'image à utiliser. 'ImageID' peut être obtenu simplement en utilisant ImageID() de la bibliothèque Image. Cela peut être une image chargée avec LoadImage() ou créée en mémoire avec CreateImage() .
Pour avoir un fond transparent, utiliser le format de fichier Windows .ICO ou .PNG, ou le format de fichier PNG sous Linux ou MacOS X.

Mode (optionnel) Peut prendre l'une des valeurs suivantes :

```
#PB_ToolBar_Normal: Bouton
  standard (par défaut)
#PB_ToolBar_Toggle: Bouton
  bascule (soit enfoncé,
  soit relâché)
```

Les commandes GetToolBarButtonState() et SetToolBarButtonState() permettent de changer l'état d'un bouton de type #PB_ToolBar_Toggle.

Texte$ (optionnel) Le texte à afficher avec le bouton.
La barre d'outils doit être créée avec l'option #PB_ToolBar_Text sinon le texte ne sera pas affiché.

Valeur de retour

Aucune.

Remarques

La fonction CreateToolBar() doit avoir été appelée avant d'utiliser cette fonction.
La détection des évènements sur les barres d'outils est similaire à celle des menus, et nécessite donc la commande EventMenu() .

Exemple

```
If OpenWindow(0, 0, 0, 150, 25, "Barre d'outils", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)

  CreateImage(0, 16, 16)
  StartDrawing(ImageOutput(0))
    Box(0, 0, 16, 16, RGB(255, 255, 255))
    Box(4, 4, 8, 8, RGB(255, 0, 0))
  StopDrawing()

  CreateImage(1, 16, 16)
  StartDrawing(ImageOutput(1))
    Box(0, 0, 16, 16, RGB(255, 0, 0))
    Box(4, 4, 8, 8, RGB(255, 255, 255))
  StopDrawing()

  If CreateToolBar(0, WindowID(0))
    ToolBarImageButton(0, ImageID(0))
    ToolBarImageButton(1, ImageID(1))
  EndIf

  Repeat
  Until WaitWindowEvent() = #PB_Event_CloseWindow
EndIf
```

Voir aussi

CreateToolBar() ,
ToolBarStandardButton() ,
ToolBarSeparator()

OS Supportés

Tous

153.9 ToolBarSeparator

Syntaxe

```
ToolBarSeparator()
```

Description

Ajoute un séparateur vertical à la barre d'outils en cours de création.

Arguments

Aucun.

Valeur de retour

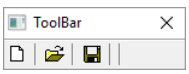
Aucune.

Remarques

La fonction CreateToolBar() doit avoir été appelée avant d'utiliser cette fonction.

Exemple

```
1   If OpenWindow(0, 0, 0, 150,
     25, "Barre d'outils",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2     If CreateToolBar(0,
     WindowID(0))
3       ToolBarStandardButton(0,
     #PB_ToolBarIcon_New)
4       ToolBarSeparator()
5       ToolBarStandardButton(1,
     #PB_ToolBarIcon_Open)
6       ToolBarSeparator()
7       ToolBarStandardButton(2,
     #PB_ToolBarIcon_Save)
8       ToolBarSeparator()
9       ToolBarSeparator()
10    EndIf
11    Repeat
12      Event =
     WaitWindowEvent()
13      If Event =
     #PB_Event_Menu
14        Debug "Identifiant de
     la barre d'outils :
     "+Str(EventMenu())
15      EndIf
16    Until Event =
     #PB_Event_CloseWindow
17  EndIf
```



Voir aussi

CreateToolBar() ,
ToolBarStandardButton() ,
ToolBarImageButton()

OS Supportés

Tous

153.10 ToolBarStandardButton

Syntaxe

```
ToolBarStandardButton(#Bouton,
    #IconeBouton [, Mode [,
    Texte$]])
```

Description

Ajoute un bouton standard à la barre d'outils en cours de construction.
Un bouton standard est un bouton dont l'icône est fournie par le système d'exploitation.

Arguments

#**Button** Le numéro du nouveau bouton.

#**IconeBouton** Peut être l'une des constantes suivantes :

```
#PB_ToolBarIcon_New
          (Nouveau
 fichier)
#PB_ToolBarIcon_Open
         (Ouvrir Fichier)
#PB_ToolBarIcon_Save
         (Enregistrer
 Fichier)
#PB_ToolBarIcon_Print
        (Imprimer)
#PB_ToolBarIcon_PrintPreview
 (Aperçu avant impression)
#PB_ToolBarIcon_Find
         (Chercher une
 occurence)
#PB_ToolBarIcon_Replace
      (Remplacer une
 occurence)

#PB_ToolBarIcon_Cut
          (Couper)
#PB_ToolBarIcon_Copy
         (Copier)
#PB_ToolBarIcon_Paste
        (Coller)
#PB_ToolBarIcon_Undo
         (Annuler
 l'action)
#PB_ToolBarIcon_Redo
         (Repéter
 l'action)

#PB_ToolBarIcon_Delete
       (Effacer)
```

```
#PB_ToolBarIcon_Properties
   (Propriétés)
#PB_ToolBarIcon_Help
          (Aide)
```

Mode (optionnel) Peut prendre l'une des valeurs suivantes :

```
#PB_ToolBar_Normal: Bouton
 standard (par défaut)
#PB_ToolBar_Toggle: Bouton
 bascule (soit enfoncé,
 soit relâché)
```

Les commandes GetToolBarButtonState() et SetToolBarButtonState() permettent de changer l'état d'un bouton de type #PB_ToolBar_Toggle.

Texte$ (optionnel) Le texte à afficher avec le bouton.
La barre d'outils doit être créée avec l'option #PB_ToolBar_Text sinon le texte ne sera pas affiché.

Valeur de retour

Aucune.

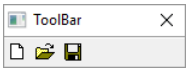
Remarques

La fonction CreateToolBar() doit avoir été appelée avant d'utiliser cette fonction.
La détection des évènements sur les barres d'outils est similaire à celle des menus, et nécessite donc la commande EventMenu() .

Exemple

```
1  If OpenWindow(0, 0, 0, 150,
    25, "Barre d'outils",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2    If CreateToolBar(0,
    WindowID(0))
3      ToolBarStandardButton(0,
    #PB_ToolBarIcon_New)
4      ToolBarStandardButton(1,
    #PB_ToolBarIcon_Open)
5      ToolBarStandardButton(2,
    #PB_ToolBarIcon_Save)
6    EndIf
7    Repeat
8      Event =
    WaitWindowEvent()
9      If Event =
    #PB_Event_Menu
10       Debug "Identifiant de
    la barre d'outils :
    "+Str(EventMenu())
```

```
    EndIf
  Until Event =
  #PB_Event_CloseWindow
EndIf
```



Voir aussi

CreateToolBar() , ToolBarImageButton() , ToolBarSeparator()

OS Supportés

Windows, Linux

153.11 ToolBarButtonText

Syntaxe

```
ToolBarButtonText(#BarreOutils,
    Bouton, Texte$)
```

Description

Modifie le texte d'un bouton.

Arguments

#BarreOutils La barre d'outils à utiliser.

Button Le numéro du bouton à utiliser.

Texte$ Le nouveau texte.

Valeur de retour

Aucune.

Remarques

La barre d'outils a dû être créée avec l'option #PB_ToolBar_Text.

Exemple

```
If OpenWindow(0, 0, 0, 150,
 20, "Barre d'outils",
 #PB_Window_SystemMenu |
 #PB_Window_ScreenCentered)
  If CreateToolBar(0,
 WindowID(0),
 #PB_ToolBar_Large |
 #PB_ToolBar_Text)
    ToolBarStandardButton(0,
 #PB_ToolBarIcon_New,  0,
 "Nouveau")
```

```
4       ToolBarStandardButton(1,
    #PB_ToolBarIcon_Open, 0,
    "Ouvrir")
5       ToolBarStandardButton(2,
    #PB_ToolBarIcon_Save, 0,
    "Enregistrer")
6       ToolBarButtonText(0, 0,
    "Vieux !")
7     EndIf
8     Repeat
9     Until WaitWindowEvent() =
    #PB_Event_CloseWindow
10  EndIf
```

Voir aussi

ToolBarStandardButton() ,
ToolBarImageButton() ,
ToolBarSeparator() , CreateToolBar()

OS Supportés

Tous

153.12 ToolBarToolTip

Syntaxe

```
ToolBarToolTip(#BarreOutils,
    Bouton, Texte$)
```

Description

Crée ou remplace une info-bulle.

Arguments

#BarreOutils La barre d'outils à utiliser.

Button Le numéro du bouton à utiliser.

Texte$ Le nouveau texte.
Si le texte est vide, l'info-bulle est supprimée.

Valeur de retour

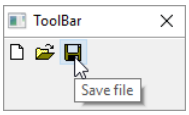
Aucune.

Remarques

Un 'Tooltip' ou info-bulle est un texte flottant qui apparait au bout d'un certain temps lorsque le curseur de la souris est immobile au dessus d'un bouton d'une barre d'outils.

Exemple

```
1   If OpenWindow(0, 0, 0, 150,
     60, "Barre d'outils",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2     If CreateToolBar(0,
     WindowID(0))
3       ToolBarStandardButton(0,
     #PB_ToolBarIcon_New)
4       ToolBarStandardButton(1,
     #PB_ToolBarIcon_Open)
5       ToolBarStandardButton(2,
     #PB_ToolBarIcon_Save)
6       ToolBarToolTip(0, 0,
     "Nouveau document")
7       ToolBarToolTip(0, 1,
     "Ouvrir un document")
8       ToolBarToolTip(0, 2,
     "Enregistrer le document")
9     EndIf
10    Repeat
11    Until WaitWindowEvent() =
     #PB_Event_CloseWindow
12  EndIf
```



Voir aussi

ToolBarStandardButton() ,
ToolBarImageButton() ,
ToolBarSeparator()

OS Supportés

Tous

153.13 ToolBarID

Syntaxe

```
Resultat =
    ToolBarID(#BarreOutils)
```

Description

Renvoie l'identifiant d'une barre d'outils.

Arguments

#**BarreOutils** La barre d'outils à utiliser.

Valeur de retour

Renvoie l'identifiant système unique de la barre d'outils.

Voir aussi

CreateToolBar()

OS Supportés

Tous

Chapitre 154

VectorDrawing

Généralités

La bibliothèque de dessin vectoriel fournit des opérations de dessin de haute qualité et indépendantes de la résolution de l'affichage, aussi bien sur des images qu'à l'impression. Contrairement à la bibliothèque 2DDrawing , elle peut fonctionner avec une grande variété d'unités de mesure et permet des changements de coordonnées à l'envie. Ainsi, l'écriture de routine de dessin en est facilitée car elle s'adaptera à la sortie de l'utilisateur, quelle que soit sa résolution ou sa taille. De plus, cette bibliothèque prend en charge la transparence alpha dans toutes ses opérations. Seules les polices vectorielles sont autorisées, comme TrueType, les polices bitmap ne sont donc pas autorisées. Vous ne pouvez pas non plus utiliser de police enregistrée avec RegisterFontFile()

Mode d'emploi

Les opérations de dessin de cette bibliothèque comportent trois étapes de base :
1) Construction d'un chemin avec des fonctions telles que AddPathLine() , AddPathCurve() , etc.
2) Sélection d'une source de dessin comme avec VectorSourceColor() , etc.
3) Le chemin est rempli avec un contour , un motif plein , des points , des traits ou un mélange de points et de traits .
Après avoir rempli le contour ou l'intérieur d'une figure, le chemin est remis à zéro et un nouveau chemin peut être construit pour l'opération de dessin suivante. La sélection de la source de dessin (étape 2) n'a pas besoin d'être répétée à chaque fois, car elle n'est pas remise à zéro.
Ce type de dessin à base de chemin permet de dessiner des formes complexes avec des attributs intéressants comme dessiner des

lignes épaisses, avec un motif à base de points et de traits, aux coins arrondis ou pointus, le tout sans introduire d'artefacts d'affichage à la jonction des segments et des figures. Cela est du au fait que l'ensemble du chemin est tracé en une seule fois.
Voir la fonction AddPathLine() pour un exemple simple.

Les unités de mesure

Chaque sortie de dessin a une unité de mesure par défaut. L'unité par défaut est le pixel pour les écrans ou les images matricielles (bmp, jpg, gif, png, etc) et le point pour les imprimantes ou les images vectorielles. Il est cependant possible de sélectionner une unité de mesure différente lors de la création avec les fonctions ImageVectorOutput() , PrinterVectorOutput() ,... Toutes les opérations de dessin utiliseront l'unité de mesure sélectionnée et convertiront en interne les valeurs en coordonnées réelles pour le dispositif de sortie. Utiliser l'unité de mesure par défaut permet d'écrire du code qui est indépendant de la sortie utilisée. L'unité de mesure peut être vérifiée avec VectorUnit() .

Transformation de coordonnées

Il est possible de translater , de changer d'échelle (homothétie) , d'appliquer une rotation , d'inverser ou d' incliner le système de coordonnées utilisé pour le dessin. Les transformations peuvent être combinées librement. Ces transformations affectent toutes les opérations de dessin. Les utilisations possibles de ces transformations de coordonnées est de faire pivoter une figure ou de l'étirer sans avoir à modifier le code. Par exemple, pour imprimer en paysage, il suffit de faire tourner simplement les coordonnées (et donc toutes les sorties).
Il existe quatre systèmes de coordonnées différents et certaines fonctions prennent un paramètre facultatif permettant de sélectionner le système qui devrait être utilisé. Ce sont les options suivantes :
#PB_Coordinate_Device

Ce système de coordonnées représente les coordonnées physiques du périphérique de sortie. Il ne peut pas être transformé. Ce système de coordonnées est utile lors de la conversion de coordonnées entre le périphérique et le dessin avec

ConvertCoordinateX() et ConvertCoordinateY() .

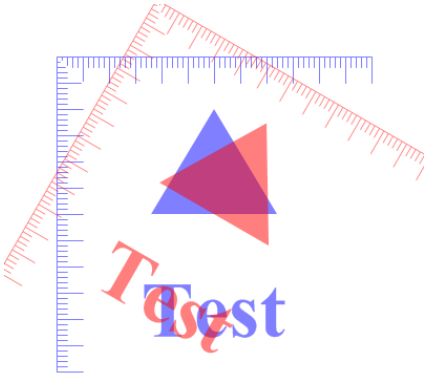
#PB_Coordinate_Output
Coordonnées initiales de sortie en unité de mesure sélectionnée. Ce système de coordonnées est égal à #PB_Coordinate_Device sauf pour une éventuelle mise à l'échelle avec une unité de mesure différente. Ce système de coordonnées ne peut pas être transformée.

#PB_Coordinate_User
Ceci est le système de coordonnées utilisé pour toutes les opérations de dessin. Ce système de coordonnées est utilisé chaque fois qu'un autre système n'est pas explicitement spécifié. Il peut être transformé librement. Initialement, ce système de coordonnées est égal au système #PB_Coordinate_Output et peut être réinitialisé avec ResetCoordinates() .

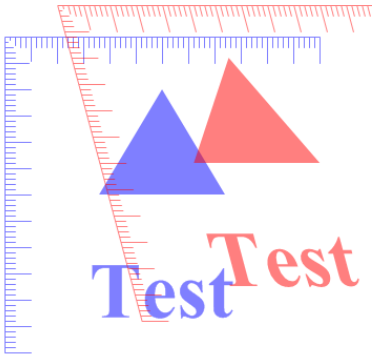
#PB_Coordinate_Source
Ce système de coordonnées est utilisé par les fonctions qui sélectionnent la source de dessin vectoriel. Il est utile avec VectorSourceImage() afin de transformer l'image de la source. Ce système de coordonnées est relié au système #PB_Coordinate_User, de sorte que toute transformation du système de #PB_Coordinate_User aura une incidence sur ce système.

Dans la plupart des cas, le système #PB_Coordinate_User est le plus intéressant et est donc la valeur par défaut. Les autres systèmes sont utiles principalement pour la conversion de coordonnées ou dans des buts spéciaux, tels que la transformation de l'image source.

Exemple : Rotation du système de coordonnées



Exemple : Combinaison de translation et d'inclinaison



Etat et couches

Un certain nombre de propriétés de la sortie de dessin tels que les transformations de coordonnées, le zonage (clipping) ou la source de dessin peuvent être sauvegardées et restaurées ultérieurement avec les fonctions SaveVectorState() et RestoreVectorState() . Cela permet de faire des modifications temporaires sur la sortie de dessin puis de restaurer l'état précédent, plus tard. Les commandes fonctionnent dans une pile, il est donc possible de sauvegarder/restaurer plusieurs états de dessin.
La fonction BeginVectorLayer() permet de sauvegarder l'état de dessin actuel en créant une nouvelle couche de dessin virtuelle. Les futures opérations de dessin seront dirigées sur cette couche. Un appel à EndVectorLayer() combinera la couche avec la sortie du dessous et restaurera l'état du dessin précédent. Ceci permet de combiner un certain nombre d'opérations de dessin, puis de les appliquer en tant que couche de sortie. De multiples couches temporaires peuvent être créées de cette façon.

Remarque

Afin d'atteindre un rendu parfait, la bibliothèque utilise l'anti crénelage (antialiasing). Ainsi, il n'est pas possible d'exécuter des figures d'un seul pixel d'épaisseur, stricto sensu. Utiliser pour cela la bibliothèque dessin 2D .

OS Supportés

Tous

154.1 StartVectorDrawing

Syntaxe

```
Resultat =
    StartVectorDrawing(Sortie)
```

Description

Prépare la bibliothèque de dessin vectoriel pour dessiner sur la sortie spécifiée.

Arguments

Sortie Les dessins seront rendus directement sur :

```
    WindowVectorOutput()
  : Une fenêtre
    ImageVectorOutput()
    : Une image (voir
     CreateImage()
)
    PrinterVectorOutput()
: Une imprimante
    CanvasVectorOutput()
  : Un CanvasGadget()

    PdfVectorOutput()
      : Un fichier PDF (Linux
     et MacOS seulement)
    SvgVectorOutput()
      : Un fichier SVG (Linux
     seulement)
```

Valeur de retour

Renvoie une valeur non nulle si le dessin est possible, zéro sinon.

Remarques

Lorsque tous les dessins sont terminés, la fonction StopVectorDrawing() doit être appelée.
Si "Activer la gestion des Threads" est coché dans les options du compilateur alors chaque thread a sa propre surface de dessin, ce qui signifie que deux threads peuvent dessiner sur des surfaces de dessin différentes en même temps.

Voir aussi

StopVectorDrawing()

OS Supportés

Tous

154.2 StopVectorDrawing

Syntaxe

```
StopVectorDrawing()
```

Description

Termine une séquence d'opérations de dessin vectoriel et libère toutes les ressources allouées.

Arguments

Aucun.

Valeur de retour

Aucune.

Voir aussi

StartVectorDrawing()

OS Supportés

Tous

154.3 VectorOutputWidth

Syntaxe

```
Resultat.d =
    VectorOutputWidth()
```

Description

Renvoie la largeur de la zone en sortie du dessin vectoriel.

Arguments

Aucun.

Valeur de retour

Renvoie la largeur de sortie.

Voir aussi

VectorOutputHeight() , VectorUnit() , VectorResolutionX() , VectorResolutionY()

OS Supportés

Tous

154.4 VectorOutputHeight

Syntaxe

```
Resultat.d =
    VectorOutputHeight()
```

Description

Renvoie la hauteur de la zone en sortie du dessin vectoriel.

Arguments

Aucun.

Valeur de retour

Renvoie la hauteur de sortie.

Exemple

Voir aussi

VectorOutputWidth() , VectorUnit() , VectorResolutionX() , VectorResolutionY()

OS Supportés

Tous

154.5 VectorResolutionX

Syntaxe

```
Resultat.d =
    VectorResolutionX()
```

Description

Renvoie la résolution horizontale de la zone de sortie de dessin vectoriel.

Arguments

Aucun.

Valeur de retour

Renvoie la résolution horizontale en dpi (points par pouce)

Voir aussi

VectorResolutionY()

OS Supportés

Tous

154.6 VectorResolutionY

Syntaxe

```
Resultat.d =
    VectorResolutionY()
```

Description

Renvoie la résolution verticale de la zone de sortie de dessin vectoriel.

Arguments

Aucun.

Valeur de retour

Renvoie la résolution verticale en dpi (ppp : points par pouce).

Remarques

La résolution verticale peut différer de la résolution horizontale dans le cas d'une sortie sur imprimante.

Voir aussi

VectorResolutionX()

OS Supportés

Tous

154.7 VectorUnit

Syntaxe

```
Resultat = VectorUnit()
```

Description

Renvoie l'unité dans laquelle toutes les coordonnées et les dimensions sont mesurées sur la sortie de dessin vectoriel en cours. Cette unité a été spécifiée lors de la création de la sortie.

Arguments

Aucun.

Valeur de retour

Renvoie l'une des valeurs suivantes :

```
#PB_Unit_Pixel      : Les
 valeurs sont mesurées en
 pixels (ou point (dots)
 pour les imprimantes)
#PB_Unit_Point      : Les
 valeurs sont mesurées en
 points (1/72 pouce =
 25.4/72 mm = 0,352 778 mm)
#PB_Unit_Inch       : Les
 valeurs sont mesurées en
 pouces (25,4 millimètres)
#PB_Unit_Millimeter: Les
 valeurs sont mesurées en
 millimètres (0,039 370
 pouce)
```

OS Supportés

Tous

154.8 SaveVectorState

Syntaxe

```
SaveVectorState()
```

Description

Enregistre l'état du dessin vectoriel en cours en vue d'être restauré plus tard.

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

De multiples états peuvent être sauvegardés sur une pile et restaurés dans l'ordre inverse.
Les informations suivantes sont enregistrées :
- Les transformations de coordonnées
- La source de dessin
- La police de dessin
- Le zonage (clipping)

Notez que le chemin courant n'est pas sauvegardé par cette commande.

Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0, 400, 200)
  LoadFont(0, "Times New Roman", 20, #PB_Font_Bold)

  If StartVectorDrawing(CanvasVectorOutput(0))

    VectorSourceColor(RGBA(255, 0, 0, 255))
    VectorFont(FontID(0))

    MovePathCursor(20, 20)
    DrawVectorText("Texte normal")

    ; Les modifications apportées à l'état de dessin au sein de ce bloc ne touchent pas les autres commandes
    SaveVectorState()
      MovePathCursor(120, 160)
      RotateCoordinates(120, 160, -50)
      VectorSourceColor(RGBA(0, 0, 255, 255))
      DrawVectorText("Rotation texte")
    RestoreVectorState()

    MovePathCursor(220, 140)
    DrawVectorText("Texte normal")

    StopVectorDrawing()
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

RestoreVectorState() , BeginVectorLayer()

OS Supportés

Tous

154.9 RestoreVectorState

Syntaxe

```
RestoreVectorState()
```

Description

Restaure l'état de dessin vectoriel qui a été stocké avec l'appel à SaveVectorState() .

Arguments

Aucun.

Valeur de retour

Aucune.

Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
 CanvasGadget(0, 0, 0, 400, 200)
 LoadFont(0, "Times New Roman", 20, #PB_Font_Bold)

 If StartVectorDrawing(CanvasVectorOutput(0))

    VectorSourceColor(RGBA(255, 0, 0, 255))
    VectorFont(FontID(0))

    MovePathCursor(20, 20)
    DrawVectorText("Texte normal")

    ; Les modifications apportées à l'état de dessin au sein de ce bloc ne touchent pas les autres commandes
    SaveVectorState()
      MovePathCursor(120, 160)
      RotateCoordinates(120, 160, -50)
      VectorSourceColor(RGBA(0, 0, 255, 255))
      DrawVectorText("Rotation de texte")
    RestoreVectorState()

    MovePathCursor(220, 140)
```

```
22        DrawVectorText("Texte normal")
23
24        StopVectorDrawing()
25      EndIf
26
27      Repeat
28        Event = WaitWindowEvent()
29      Until Event = #PB_Event_CloseWindow
30    EndIf
```

Voir aussi

SaveVectorState()

OS Supportés

Tous

154.10 BeginVectorLayer

Syntaxe

```
BeginVectorLayer([Transparence])
```

Description

Commence une nouvelle couche vide au-dessus du dessin vectoriel en cours. Désormais, toutes les opérations de dessin seront effectuées sur cette couche jusqu'à ce que EndVectorLayer() soit appelée.

Arguments

Transparence (optionnel) Le niveau de transparence (alpha) de la nouvelle couche.
Entre 0 (transparent) et 255 (opaque par défaut).

Valeur de retour

Aucune.

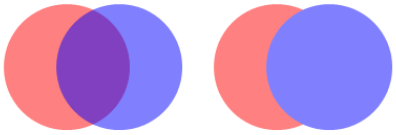
Remarques

Cette commande enregistre également l'état de dessin courant de la même manière que SaveVectorState() .
Plusieurs couches peuvent être créées.
Les ressources nécessaires pour créer la couche temporaire dépendent de la taille du chemin clippé . Il est donc recommandé de définir un chemin de détourage qui ne

couvre que la zone nécessaire afin d'économiser les ressources et d'améliorer les performances de dessin.

Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0, 400, 200)

  If StartVectorDrawing(CanvasVectorOutput(0))

    ; Dessin semi-transparent sur la couche de base
    AddPathCircle(75, 100, 60)
    VectorSourceColor(RGBA(255, 0, 0, 127))
    FillPath()
    AddPathCircle(125, 100, 60)
    VectorSourceColor(RGBA(0, 0, 255, 127))
    FillPath()

    ; Dessin opaque sur une couche semi-transparente
    BeginVectorLayer(127)
      AddPathCircle(275, 100, 60)
      VectorSourceColor(RGBA(255, 0, 0, 255))
      FillPath()
      AddPathCircle(325, 100, 60)
      VectorSourceColor(RGBA(0, 0, 255, 255))
      FillPath()
    EndVectorLayer()

    StopVectorDrawing()
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```



Voir aussi

EndVectorLayer() , SaveVectorState()

OS Supportés

Tous

154.11 EndVectorLayer

Syntaxe

```
EndVectorLayer()
```

Description

Termine le dessin sur une couche temporaire créée par BeginVectorLayer() .

Arguments

Aucun.

Valeur de retour

Aucune.

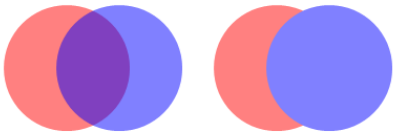
Remarques

Le contenu de la couche est dessiné vers la couche inférieure suivante en utilisant la transparence (alpha) de la couche temporaire.
Cette commande rétablit également l'état de dessin qui était en vigueur lorsque BeginVectorLayer() a été appelé.

Exemple

```
1  If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2   CanvasGadget(0, 0, 0, 400, 200)
3
4   If StartVectorDrawing(CanvasVectorOutput(0))
5
```

```
6       ; Dessin semi-transparent sur la couche de base
7       AddPathCircle(75, 100, 60)
8       VectorSourceColor(RGBA(255, 0, 0, 127))
9       FillPath()
10      AddPathCircle(125, 100, 60)
11      VectorSourceColor(RGBA(0, 0, 255, 127))
12      FillPath()
13
14      ; Dessin opaque sur une couche semi-transparente
15      BeginVectorLayer(127)
16        AddPathCircle(275, 100, 60)
17        VectorSourceColor(RGBA(255, 0, 0, 255))
18        FillPath()
19        AddPathCircle(325, 100, 60)
20        VectorSourceColor(RGBA(0, 0, 255, 255))
21        FillPath()
22      EndVectorLayer()
23
24      StopVectorDrawing()
25    EndIf
26
27    Repeat
28      Event = WaitWindowEvent()
29    Until Event = #PB_Event_CloseWindow
30  EndIf
```



Voir aussi

BeginVectorLayer() , SaveVectorState()

OS Supportés

Tous

154.12 NewVectorPage

Syntaxe

```
NewVectorPage()
```

Description

Termine la page de dessin vectoriel en cours et commence une nouvelle page.

Remarques

Les sorties suivantes sont prises en charge :
PrinterVectorOutput()
PdfVectorOutput() (Linux et MacOS seulement)
SvgVectorOutput() (Linux seulement)

Arguments

Aucun.

Valeur de retour

Aucune.

Voir aussi

NewPrinterPage()

OS Supportés

Tous

154.13 FillVectorOutput

Syntaxe

```
FillVectorOutput()
```

Description

Remplit toute la zone de dessin (à l'exception des zones non clippées) avec la source de dessin en cours.

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Cette opération est équivalente à la construction d'un chemin qui couvre la totalité de la zone de dessin après un appel à FillPath() .

Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0, 400, 200)

  If StartVectorDrawing(CanvasVectorOutput(0))

    ; Mettre la sortie entièrement en rouge
    VectorSourceColor(RGBA(255, 0, 0, 255))
    FillVectorOutput()

    StopVectorDrawing()
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

FillPath() , ClipPath()

OS Supportés

Tous

154.14 ResetCoordinates

Syntaxe

```
ResetCoordinates([Systeme])
```

Description

Rétablit les transformations qui ont été appliquées sur le dessin vectoriel ainsi que le système de coordonnées qui était en vigueur lorsque StartVectorDrawing() a été appelé.

Arguments

Systeme (optionnel) Indique le système de coordonnées à changer.
Peut être l'une des valeurs suivantes :

```
#PB_Coordinate_User   :
 Change le système de
 coordonnées des points
 du chemin de dessin (par
 défaut)
#PB_Coordinate_Source:
 Change le système de
 coordonnées de la source
 du dessin vectoriel
```

Valeur de retour

Aucune.

Remarques

Voir l'aperçu de la bibliothèque VectorDrawing pour une introduction aux différents systèmes de coordonnées.

Voir aussi

TranslateCoordinates() , ScaleCoordinates() , RotateCoordinates() , SkewCoordinates() , FlipCoordinatesX() , FlipCoordinatesY() , ConvertCoordinateX() , ConvertCoordinateY()

OS Supportés

Tous

154.15 TranslateCoordinates

Syntaxe

```
TranslateCoordinates(X.d, Y.d
    [, Systeme])
```

Description

Déplace l'origine du système de coordonnées du dessin vectoriel.

Arguments

X.d, Y.d Le déplacement en X et en Y de l'origine.

Systeme (optionnel) Indique le système de coordonnées à changer.
Peut être l'une des valeurs suivantes :

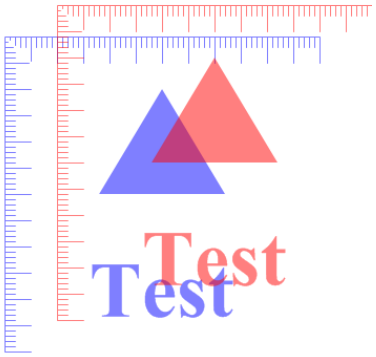
```
#PB_Coordinate_User   :
 Change le système de
 coordonnées des points
 du chemin de dessin (par
 défaut)
#PB_Coordinate_Source:
 Change le système de
 coordonnées de la source
 du dessin vectoriel
```

Valeur de retour

Aucune.

Remarques

Toutes les futures opérations de dessin se feront par rapport à la nouvelle origine. Voir l'aperçu de la bibliothèque VectorDrawing pour une introduction aux différents systèmes de coordonnées. L'image suivante montre l'effet de la translation de coordonnées. Une même figure est dessinée à deux reprises, l'originale est en bleu, et la version après translation de coordonnées est en rouge.



Exemple

```
1   If OpenWindow(0, 0, 0, 400,
     200, "VectorDrawing",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2     CanvasGadget(0, 0, 0,
     400, 200)
3
4     If
     StartVectorDrawing(CanvasVectorOutput(0))
5       VectorFont(LoadFont(0,
     "Times New Roman", 60,
     #PB_Font_Bold))
6
7       VectorSourceColor(RGBA(0,
     0, 255, 128))
```

```
8         MovePathCursor(50, 50)
9         DrawVectorText("Test")
10
11        TranslateCoordinates(30,
      30) ; toutes les
      coordonnées sont déplacés
      de 30 pixels dans chaque
      direction
12
13        VectorSourceColor(RGBA(255,
      0, 0, 128))
14        MovePathCursor(50, 50)
15        DrawVectorText("Test")
16
17        StopVectorDrawing()
18      EndIf
19
20      Repeat
21        Event =
      WaitWindowEvent()
22      Until Event =
      #PB_Event_CloseWindow
23    EndIf
```

Voir aussi

ResetCoordinates() , ScaleCoordinates() , RotateCoordinates() , SkewCoordinates() , FlipCoordinatesX() , FlipCoordinatesY() , ConvertCoordinateX() , ConvertCoordinateY()

OS Supportés

Tous

154.16 ScaleCoordinates

Syntaxe

```
ScaleCoordinates(EchelleX.d,
    EchelleY.d [, Systeme])
```

Description

Changement d'échelle du système de coordonnées du dessin vectoriel en l'étirant dans les directions X et/ou Y.

Arguments

EchelleX.d, EchelleY.d Le facteur d'échelle dans chaque direction.

```
< 0.0: Un facteur négatif
 donne un effet miroir.
< 1.0: Rapetisse le
 système de coordonnées
  1.0: Un facteur de 1
 laisse les coordonnées
 inchangées
> 1.0: Agrandit le système
 de coordonnées
```

Systeme (optionnel) Indique le système de coordonnées à changer.
Peut être l'une des valeurs suivantes :

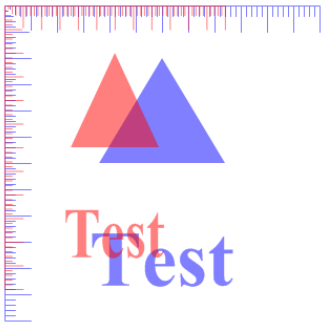
```
#PB_Coordinate_User   :
 Change le système de
 coordonnées des points
 du chemin de dessin (par
 défaut)
#PB_Coordinate_Source:
 Change le système de
 coordonnées de la source
 du dessin vectoriel
```

Valeur de retour

Aucune.

Remarques

Voir l'aperçu de la bibliothèque VectorDrawing pour une introduction aux différents systèmes de coordonnées.
L'image suivante montre l'effet d'échelle de coordonnées. Une même figure est dessinée à deux reprises, l'originale est en bleu, et la version après l'effet est en rouge.



Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0, 400, 200)

  If StartVectorDrawing(CanvasVectorOutput(0))
    VectorFont(LoadFont(0, "Times New Roman", 60, #PB_Font_Bold))

    VectorSourceColor(RGBA(0, 0, 255, 128))
    MovePathCursor(50, 50)
    DrawVectorText("Test")

    ScaleCoordinates(0.7, 0.9)

    VectorSourceColor(RGBA(255, 0, 0, 128))
    MovePathCursor(50, 50)
    DrawVectorText("Test")

    StopVectorDrawing()
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

ResetCoordinates() ,
TranslateCoordinates() ,
RotateCoordinates() , SkewCoordinates() ,
FlipCoordinatesX() , FlipCoordinatesY() ,
ConvertCoordinateX() ,
ConvertCoordinateY()

OS Supportés

Tous

154.17 RotateCoordinates

Syntaxe

```
RotateCoordinates(X.d, Y.d,
    Angle.d [, Systeme])
```

Description

Rotation du système de coordonnées du dessin vectoriel autour du point donné.

Arguments

X.d, Y.d Indique le centre de rotation.

Angle.d Angle de rotation en degrés. Un angle positif tourne dans le sens horaire.

Systeme (optionnel) Indique le système de coordonnées à changer.
Peut être l'une des valeurs suivantes :

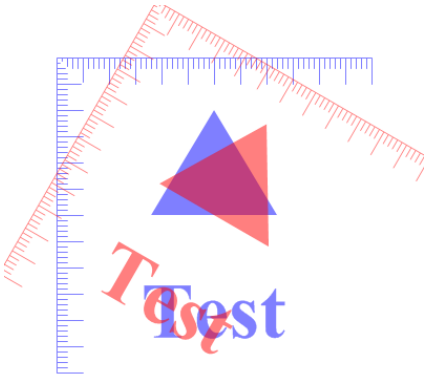
```
#PB_Coordinate_User   :
 Change le système de
 coordonnées des points
 du chemin de dessin (par
 défaut)
#PB_Coordinate_Source:
 Change le système de
 coordonnées de la source
 du dessin vectoriel
```

Valeur de retour

Aucune.

Remarques

Le centre est exprimé en termes de système de coordonnées courant.
Voir l'aperçu de la bibliothèque VectorDrawing pour une introduction aux différents systèmes de coordonnées.
L'image suivante montre l'effet de la rotation de coordonnées. Une même figure est dessinée à deux reprises, l'originale est en bleu, et la version après rotation de coordonnées est en rouge.



Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
 CanvasGadget(0, 0, 0, 400, 200)

 If StartVectorDrawing(CanvasVectorOutput(0))
    VectorFont(LoadFont(0, "Times New Roman", 60, #PB_Font_Bold))

    VectorSourceColor(RGBA(0, 0, 255, 128))
    MovePathCursor(50, 50)
    DrawVectorText("Test")

    RotateCoordinates(50, 50, -20) ; rotation de -20 degrés autour du point(50, 50)

    VectorSourceColor(RGBA(255, 0, 0, 128))
    MovePathCursor(50, 50)
    DrawVectorText("Test")

    StopVectorDrawing()
 EndIf

 Repeat
    Event = WaitWindowEvent()
 Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

ResetCoordinates() , TranslateCoordinates() , ScaleCoordinates() , SkewCoordinates() , FlipCoordinatesX() , FlipCoordinatesY() , ConvertCoordinateX() , ConvertCoordinateY()

OS Supportés

Tous

154.18 SkewCoordinates

Syntaxe

```
SkewCoordinates(AngleX.d,
    AngleY.d [, Systeme])
```

Description

Incline le système de coordonnées du dessin vectoriel dans la direction X et/ou Y.

Arguments

AngleX.d, AngleY.d Angle de cisaillement dans chaque direction en degrés.

Systeme (optionnel) Indique le système de coordonnées à changer.
Peut être l'une des valeurs suivantes :

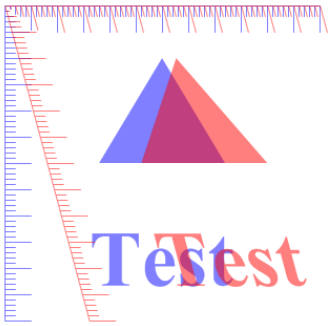
```
#PB_Coordinate_User   :
 Change le système de
 coordonnées des points
 du chemin de dessin (par
 défaut)
#PB_Coordinate_Source:
 Change le système de
 coordonnées de la source
 du dessin vectoriel
```

Valeur de retour

Aucune.

Remarques

Voir l'aperçu de la bibliothèque VectorDrawing pour une introduction aux différents systèmes de coordonnées. L'image suivante montre l'effet de cisaillement de coordonnées. Une même figure est dessinée à deux reprises, l'originale est en bleu, et la version après inclinaison de coordonnées est en rouge.



Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0, 400, 200)

  If StartVectorDrawing(CanvasVectorOutput(0))
    VectorFont(LoadFont(0, "Times New Roman", 60, #PB_Font_Bold))

    VectorSourceColor(RGBA(0, 0, 255, 128))
    MovePathCursor(50, 50)
    DrawVectorText("Test")

    SkewCoordinates(45, 0)

    VectorSourceColor(RGBA(255, 0, 0, 128))
    MovePathCursor(50, 50)
    DrawVectorText("Test")

    StopVectorDrawing()
  EndIf

  Repeat
    Event = WaitWindowEvent()
```

```
22      Until Event =
      #PB_Event_CloseWindow
23    EndIf
```

Voir aussi

ResetCoordinates() , TranslateCoordinates() , ScaleCoordinates() , RotateCoordinates() , FlipCoordinatesX() , FlipCoordinatesY() , ConvertCoordinateX() , ConvertCoordinateY()

OS Supportés

Tous

154.19 FlipCoordinatesX

Syntaxe

```
FlipCoordinatesX(AxeX.d [,
    Systeme])
```

Description

Effet miroir du système de coordonnées du dessin vectoriel par rapport à l'axe X spécifié.

Arguments

AxeX.d La coordonnée X à laquelle le système de coordonnées doit être inversé.

Systeme (optionnel) Indique le système de coordonnées à changer.
Peut être l'une des valeurs suivantes :

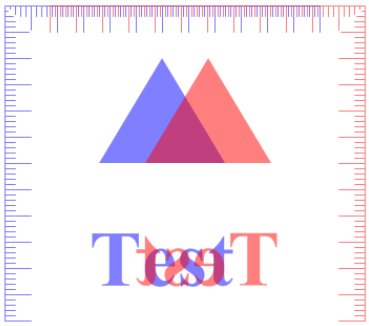
```
#PB_Coordinate_User   :
  Change le système de
  coordonnées des points
  du chemin de dessin (par
  défaut)
#PB_Coordinate_Source:
  Change le système de
  coordonnées de la source
  du dessin vectoriel
```

Valeur de retour

Aucune.

Remarques

Voir l'aperçu de la bibliothèque VectorDrawing pour une introduction aux différents systèmes de coordonnées. L'image suivante montre l'effet miroir de coordonnées. Une même figure est dessinée à deux reprises, l'originale est en bleu, et la version après l'effet miroir de coordonnées est en rouge.



Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0, 400, 200)

  If StartVectorDrawing(CanvasVectorOutput(0))
    VectorFont(LoadFont(0, "Times New Roman", 60, #PB_Font_Bold))

    VectorSourceColor(RGBA(0, 0, 255, 128))
    MovePathCursor(50, 50)
    DrawVectorText("Test")

    FlipCoordinatesX(200)

    VectorSourceColor(RGBA(255, 0, 0, 128))
    MovePathCursor(50, 50)
    DrawVectorText("Test")

    StopVectorDrawing()
  EndIf

```

```
20      Repeat
21        Event =
        WaitWindowEvent()
22      Until Event =
        #PB_Event_CloseWindow
23    EndIf
```

Voir aussi

ResetCoordinates() , TranslateCoordinates() , ScaleCoordinates() , RotateCoordinates() , SkewCoordinates() , FlipCoordinatesY() , ConvertCoordinateX() , ConvertCoordinateY()

OS Supportés

Tous

154.20 FlipCoordinatesY

Syntaxe

```
FlipCoordinatesY(AxeY.d [,
    Systeme])
```

Description

Effet miroir du système de coordonnées du dessin vectoriel par rapport à l'axe Y spécifié.

Arguments

AxeY.d La coordonnée Y à laquelle le système de coordonnées doit être inversé.

Systeme (optionnel) Indique le système de coordonnées à changer.
Peut être l'une des valeurs suivantes :

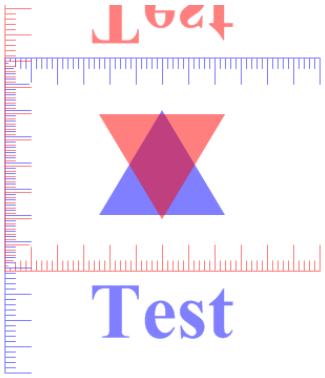
```
#PB_Coordinate_User   :
 Change le système de
 coordonnées des points
 du chemin de dessin (par
 défaut)
#PB_Coordinate_Source:
 Change le système de
 coordonnées de la source
 du dessin vectoriel
```

Valeur de retour

Aucune.

Remarques

Voir l'aperçu de la bibliothèque VectorDrawing pour une introduction aux différents systèmes de coordonnées. L'image suivante montre l'effet miroir de coordonnées. Une même figure est dessinée à deux reprises, l'originale est en bleu, et la version après l'effet miroir de coordonnées est en rouge.



Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0, 400, 200)

  If StartVectorDrawing(CanvasVectorOutput(0))
    VectorFont(LoadFont(0, "Times New Roman", 60, #PB_Font_Bold))

    VectorSourceColor(RGBA(0, 0, 255, 128))
    MovePathCursor(50, 50)
    DrawVectorText("Test")

    FlipCoordinatesY(120)

    VectorSourceColor(RGBA(255, 0, 0, 128))
    MovePathCursor(50, 50)
    DrawVectorText("Test")

    StopVectorDrawing()
  EndIf

```

```
20      Repeat
21        Event =
        WaitWindowEvent()
22      Until Event =
        #PB_Event_CloseWindow
23    EndIf
```

Voir aussi

ResetCoordinates() , TranslateCoordinates() , ScaleCoordinates() , RotateCoordinates() , SkewCoordinates() , FlipCoordinatesX() , ConvertCoordinateX() , ConvertCoordinateY()

OS Supportés

Tous

154.21 ConvertCoordinateX

Syntaxe

```
Resultat.d =
    ConvertCoordinateX(X.d,
    Y.d [, Source, Cible])
```

Description

Convertit un point d'un système de coordonnées dans un autre.

Arguments

X.d, Y.d Les coordonnées du point à convertir.

Source, Cible (optionnel) Indique les systèmes de coordonnées source et cible à utiliser.
Chacun d'eux peut être l'une des valeurs suivantes :

```
#PB_Coordinate_Device: Le
 système de coordonnées
 du dispositif de sortie
#PB_Coordinate_Output: Le
 système de coordonnées
 comme il a été créé avec
 la fonction de sortie de
 dessin
```

```
#PB_Coordinate_User   :
  Change le système de
  coordonnées des points
  du chemin de dessin
#PB_Coordinate_Source:
  Change le système de
  coordonnées de la source
  du dessin vectoriel
```

La conversion par défaut va de #PB_Coordinate_User vers #PB_Coordinate_Output.

Valeur de retour

Renvoie la coordonnée X du point dans le système de coordonnées cible.

Remarques

La coordonnée Y peut être récupérée avec la fonction ConvertCoordinateY() .
Voir l'aperçu de VectorDrawing pour une introduction aux différents systèmes de coordonnées.

Exemple

```
1  ; Cet exemple dessine un
     point à l'emplacement de
     la souris, même dans un
     système de coordonnées
     modifiées
2  ; en utilisant les
     coordonnées du dispositif
     (pixels) en coordonnées de
     l'utilisateur (point)
3  ;
4  If OpenWindow(0, 0, 0, 400,
     200, "VectorDrawing",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
5     CanvasGadget(0, 0, 0,
     400, 200)
6
7     Repeat
8       Event =
     WaitWindowEvent()
9
10      If Event =
     #PB_Event_Gadget And
     EventGadget() = 0 And
     EventType() =
     #PB_EventType_LeftButtonDown
11
12        If
     StartVectorDrawing(CanvasVectorOutput(0,
     #PB_Unit_Point))
```

```
13            RotateCoordinates(0,
    0, 30)
14
15            CanvasX =
    GetGadgetAttribute(0,
    #PB_Canvas_MouseX)
16            CanvasY =
    GetGadgetAttribute(0,
    #PB_Canvas_MouseY)
17
18            DrawingX =
    ConvertCoordinateX(CanvasX,
    CanvasY,
    #PB_Coordinate_Device,
    #PB_Coordinate_User)
19            DrawingY =
    ConvertCoordinateY(CanvasX,
    CanvasY,
    #PB_Coordinate_Device,
    #PB_Coordinate_User)
20
21            AddPathCircle(DrawingX,
    DrawingY, 10)
22            VectorSourceColor(RGBA(Random(255),
    Random(255), Random(255),
    255))
23            FillPath()
24
25            StopVectorDrawing()
26          EndIf
27
28        EndIf
29
30      Until Event =
    #PB_Event_CloseWindow
31    EndIf
```

Voir aussi

ResetCoordinates() , TranslateCoordinates() , ScaleCoordinates() , RotateCoordinates() , SkewCoordinates() , FlipCoordinatesX() , FlipCoordinatesY() , ConvertCoordinateY()

OS Supportés

Tous

154.22 ConvertCoordinateY

Syntaxe

```
Resultat.d =
    ConvertCoordinateY(X.d,
    Y.d [, Source, Cible])
```

Description

Convertit un point d'un système de coordonnées dans un autre.

Arguments

X.d, Y.d Les coordonnées du point à convertir.

Source, Cible (optionnel) Indique les systèmes de coordonnées source et cible à utiliser.
Chacun d'eux peut être l'une des valeurs suivantes :

```
#PB_Coordinate_Device: Le
 système de coordonnées
 du dispositif de sortie
#PB_Coordinate_Output: Le
 système de coordonnées
 comme il a été créé avec
 la fonction de sortie de
 dessin
#PB_Coordinate_User  :
 Change le système de
 coordonnées des points
 du chemin de dessin
#PB_Coordinate_Source:
 Change le système de
 coordonnées de la source
 du dessin vectoriel
```

La conversion par défaut va de #PB_Coordinate_User vers #PB_Coordinate_Output.

Valeur de retour

Renvoie la coordonnée Y du point dans le système de coordonnées cible.

Remarques

La coordonnée X peut être récupérée avec la fonction ConvertCoordinateX() .
Voir l'aperçu de VectorDrawing pour une introduction aux différents systèmes de coordonnées.

Exemple

```
1  ; Cet exemple dessine un
    point à l'emplacement de
    la souris, même dans un
    système de coordonnées
    modifiées
2  ; en utilisant les
    coordonnées du dispositif
    (pixels) en coordonnées de
    l'utilisateur (point)
```

```
;
If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0, 400, 200)

  Repeat
    Event = WaitWindowEvent()

    If Event = #PB_Event_Gadget And EventGadget() = 0 And EventType() = #PB_EventType_LeftButtonDown

      If StartVectorDrawing(CanvasVectorOutput(0, #PB_Unit_Point))
        RotateCoordinates(0, 0, 30)

        CanvasX = GetGadgetAttribute(0, #PB_Canvas_MouseX)
        CanvasY = GetGadgetAttribute(0, #PB_Canvas_MouseY)

        DrawingX = ConvertCoordinateX(CanvasX, CanvasY, #PB_Coordinate_Device, #PB_Coordinate_User)
        DrawingY = ConvertCoordinateY(CanvasX, CanvasY, #PB_Coordinate_Device, #PB_Coordinate_User)

        AddPathCircle(DrawingX, DrawingY, 10)
        VectorSourceColor(RGBA(Random(255), Random(255), Random(255), 255))
        FillPath()

        StopVectorDrawing()
      EndIf

    EndIf

  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

ResetCoordinates() , TranslateCoordinates() , ScaleCoordinates() , RotateCoordinates() , SkewCoordinates() , FlipCoordinatesX() , FlipCoordinatesY() , ConvertCoordinateX()

OS Supportés

Tous

154.23 ResetPath

Syntaxe

```
ResetPath()
```

Description

Remet à zéro le chemin de dessin vectoriel

Arguments

Aucun.

Valeur de retour

Aucune.

Remarques

Le chemin est vide et le curseur se déplace à la position (0,0).

Voir aussi

IsPathEmpty()

OS Supportés

Tous

154.24 ClosePath

Syntaxe

```
ClosePath()
```

Description

Ferme la figure courante dans le chemin de dessin vectoriel en ajoutant une ligne droite vers le point de départ de la figure.

Arguments

Aucun.

Valeur de retour

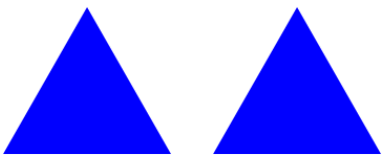
Aucune.

Remarques

Le point de départ est l'emplacement du dernier appel à MovePathCursor() . Lorsqu'un chemin est rempli , seules les figures fermées sont prise en compte.

Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0, 400, 200)

  If StartVectorDrawing(CanvasVectorOutput(0))

    ; Crée un chemin avec deux triangles fermés
    MovePathCursor(20, 160)
    AddPathLine(100, 20)
    AddPathLine(180, 160)
    ClosePath()

    MovePathCursor(220, 160)
    AddPathLine(300, 20)
    AddPathLine(380, 160)
    ClosePath()

    ; remplir le chemin
    VectorSourceColor(RGBA(0, 0, 255, 255))
    FillPath()

    StopVectorDrawing()
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```



Voir aussi

FillPath() , IsInsidePath() , MovePathCursor() , AddPathLine()

OS Supportés

Tous

154.25 MovePathCursor

Syntaxe

```
MovePathCursor(X.d, Y.d [,
    Options])
```

Description

Déplace le curseur du chemin de dessin vectoriel vers un nouvel emplacement.

Arguments

X.d, Y.d La nouvelle position du curseur dans le chemin.

Options (optionnel) Peut prendre l'une des valeurs suivantes :

```
#PB_Path_Default :
 Position absolue (par
 défaut)
#PB_Path_Relative:
 Position relative à la
 dernière position.
```

Valeur de retour

Aucune.

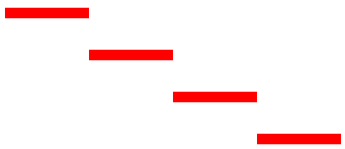
Remarques

En plus du déplacement du curseur, cette fonction commence aussi une nouvelle figure dans le chemin, ce qui signifie qu'un appel à ClosePath() va dessiner une ligne en arrière à cet endroit.

Exemple

```
1  If OpenWindow(0, 0, 0, 400,
    200, "VectorDrawing",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2   CanvasGadget(0, 0, 0,
   400, 200)
3
4   If
   StartVectorDrawing(CanvasVectorOutput(0))
```

```
5
6         MovePathCursor(40, 40)
7         For i = 1 To 4
8           AddPathLine(80, 0,
      #PB_Path_Relative)
9           MovePathCursor(0, 40,
      #PB_Path_Relative)
10        Next i
11
12        VectorSourceColor(RGBA(255,
      0, 0, 255))
13        StrokePath(10)
14
15        StopVectorDrawing()
16      EndIf
17
18      Repeat
19        Event =
      WaitWindowEvent()
20      Until Event =
      #PB_Event_CloseWindow
21    EndIf
```



Voir aussi

ClosePath() , AddPathLine() , FillPath() , StrokePath()

OS Supportés

Tous

154.26 AddPathLine

Syntaxe

```
AddPathLine(X.d, Y.d [,
    Options])
```

Description

Ajoute une ligne droite sur le chemin de dessin vectoriel, de la position actuelle du curseur aux coordonnées données.

Arguments

X.d, Y.d La position de l'extrémité de la ligne.
Devient la nouvelle position du curseur.

Options (optionnel) Peut prendre l'une des valeurs suivantes :

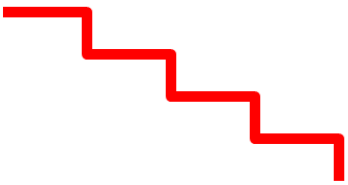
```
#PB_Path_Default :
 Position absolue (par
 défaut)
#PB_Path_Relative:
 Position relative à la
 dernière position.
```

Valeur de retour

Aucune.

Exemple

```
If OpenWindow(0, 0, 0, 400,
 200, "VectorDrawing",
 #PB_Window_SystemMenu |
 #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0,
 400, 200)

  If
 StartVectorDrawing(CanvasVectorOutput(0))

    MovePathCursor(40, 20)
    For i = 1 To 4
      AddPathLine(80, 0,
 #PB_Path_Relative)
      AddPathLine(0, 40,
 #PB_Path_Relative)
    Next i

    VectorSourceColor(RGBA(255,
 0, 0, 255))
    StrokePath(10,
 #PB_Path_RoundCorner)

    StopVectorDrawing()
  EndIf

  Repeat
    Event =
 WaitWindowEvent()
  Until Event =
 #PB_Event_CloseWindow
EndIf
```



Voir aussi

MovePathCursor() , ClosePath() , AddPathArc() , AddPathCurve() , AddPathCircle() , AddPathEllipse() , AddPathBox()

OS Supportés

Tous

154.27 AddPathArc

Syntaxe

```
AddPathArc(X1.d, Y1.d, X2.d,
    Y2.d, Rayon.d, [, Options])
```

Description

Ajoute une ligne droite en direction de (X1, Y1) suivi d'un arc dans la direction de (X2, Y2) sur le chemin de dessin vectoriel.

Arguments

X1.d, Y1.d La position de la cible pour la ligne droite.

X2.d, Y2.d La position cible pour la direction de l'arc.

Rayon.d Le rayon du coin arrondi.

Options (optionnel) Peut prendre l'une des valeurs suivantes :

```
#PB_Path_Default :
 Position absolue (par
 défaut)
#PB_Path_Relative:
 Position relative à la
 dernière position.
```

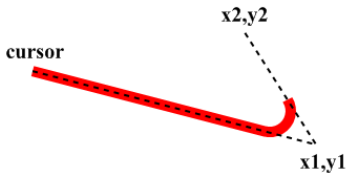
Valeur de retour

Aucune.

Remarques

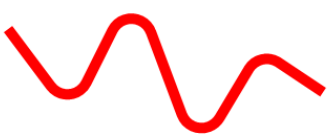
Cette fonction peut être utilisée pour créer des chemins avec des coins arrondis.
La nouvelle position du curseur sera le point final de l'arc.
L'image suivante illustre la signification des droites et des deux points de référence. Notez qu'il n'y a pas de deuxième ligne droite ajoutée en direction du point (X2, Y2).

Cela permet d'utiliser AddPathArc() de nouveau pour ajouter un coin arrondi à la position (X2, Y2).



Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0, 400, 200)

  If StartVectorDrawing(CanvasVectorOutput(0))

    MovePathCursor(40, 60)
    AddPathArc(100, 140, 160, 20, 20)
    AddPathArc(160, 20, 220, 180, 20)
    AddPathArc(220, 180, 280, 80, 20)
    AddPathArc(280, 80, 340, 120, 20)
    AddPathLine(340, 120)

    VectorSourceColor(RGBA(255, 0, 0, 255))
    StrokePath(10)

    StopVectorDrawing()
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```



Voir aussi

MovePathCursor() , AddPathLine() , AddPathCurve() , AddPathCircle() , AddPathEllipse() , AddPathBox()

OS Supportés

Tous

154.28 AddPathCurve

Syntaxe

```
AddPathCurve(X1.d, Y1.d,
   X2.d, Y2.d, X3.d, Y3.d [,
   Options])
```

Description

Ajoute une courbe de Bézier cubique au chemin de dessin vectoriel.

Arguments

X1.d, Y1.d Le premier point de control de la courbe.

X2.d, Y2.d Le second point de control de la courbe.

X3.d, Y3.d Le dernier point de la courbe. Ce point devient la nouvelle position de dessin.

Options (optionnel) Peut prendre l'une des valeurs suivantes :

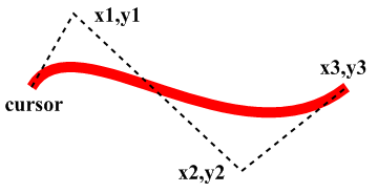
```
#PB_Path_Default :
 Position absolue (par
 défaut)
#PB_Path_Relative:
 Position relative à la
 dernière position.
```

Valeur de retour

Aucune.

Remarques

La courbe commence à la position courante du chemin et se termine à (X3, Y3). Les deux autres points déterminent la forme de la courbe.
L'image ci-dessous montre la position des points de référence.
Pour plus d'informations sur les courbes de Bézier, voir ici en français, là en anglais



Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0, 400, 200)

  If StartVectorDrawing(CanvasVectorOutput(0))

    MovePathCursor(50, 100)
    AddPathCurve(90, 30, 250, 180, 350, 100)
    VectorSourceColor(RGBA(255, 0, 0, 255))
    StrokePath(10)

    StopVectorDrawing()
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

MovePathCursor() , AddPathLine() , AddPathArc() , AddPathCircle() , AddPathEllipse() , AddPathBox()

OS Supportés

Tous

154.29 AddPathBox

Syntaxe

```
AddPathBox(X.d, Y.d, Largeur.d, Hauteur.d [, Options])
```

Description

Ajoute un rectangle (box ou boîte) dans le chemin du dessin vectoriel.

Arguments

X.d, Y.d Origine de la boîte.

Largeur.d, Hauteur.d Largeur et hauteur de la boîte.

Options (optionnel) Peut être une combinaison des valeurs suivantes :

```
#PB_Path_Default  : Pas de
 comportement spécial
 (Par défaut)
#PB_Path_Relative : Les
 positions sont relatives
 à la dernière position
 du curseur.
#PB_Path_Connected: La
 boîte est reliée au
 chemin existant avec une
 ligne et non pas
 automatiquement à une
 figure fermée.
```

Valeur de retour

Aucune.

Remarques

C'est une fonction pratique qui combine les appels nécessaires à AddPathLine() pour créer une forme de type simple boîte (box). Par défaut, cette fonction achève la figure en cours dans le chemin et ajoute la boîte en tant que figure fermée et non connectée au chemin (à savoir qu'une boîte peut être remplie).
Ce comportement peut être modifié avec les options appropriées.

Exemple

```
1  If OpenWindow(0, 0, 0, 400,
    200, "VectorDrawing",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2   CanvasGadget(0, 0, 0,
   400, 200)
3
4   If
   StartVectorDrawing(CanvasVectorOutput(0))
5
6     AddPathBox(50, 50, 200,
   50)
```

```
7         AddPathBox(150, 75,
      200, 50)
8         VectorSourceColor(RGBA(255,
      0, 0, 255))
9         StrokePath(10)
10
11        StopVectorDrawing()
12      EndIf
13
14      Repeat
15        Event =
      WaitWindowEvent()
16      Until Event =
      #PB_Event_CloseWindow
17    EndIf
```

Voir aussi

MovePathCursor() , AddPathLine() , AddPathArc() , AddPathCircle() , AddPathEllipse() , AddPathCurve()

OS Supportés

Tous

154.30 AddPathCircle

Syntaxe

```
AddPathCircle(X.d, Y.d,
    Rayon.d [, AngleDebut.d,
    AngleFin.d [, Options]])
```

Description

Ajoute un cercle ou un cercle partiel sur le trajet de dessin vectoriel.

Arguments

X.d, Y.d Centre du cercle.

Rayon.d Rayon du cercle.

AngleDebut.d, AngleFin.d (optionnel) Angle de début et de fin du cercle (du secteur), en degrés.
L'angle 0 indique l'axe X positif.
Les valeurs par défaut vont de 0 à 360 degrés.

Options (optionnel) Peut être une combinaison des valeurs suivantes :

```
#PB_Path_Default          :
 Pas de comportement
 spécial (Par défaut)
#PB_Path_Relative         :
 Les positions sont
 relatives à la dernière
 position du curseur.
#PB_Path_Connected        :
 Le cercle est relié au
 chemin existant avec une
 ligne et non pas
 automatiquement à une
 figure fermée.
#PB_Path_CounterClockwise:
 La direction de dessin
 entre les angles de
 début et de fin est dans
 le sens antihoraire.
```

Valeur de retour

Aucune.

Remarques

Cette fonction est faite pour dessiner des cercles autonomes ou des arcs de cercle. Pour dessiner des figures avec des coins arrondis, utiliser la fonction AddPathArc() , qui calcule automatiquement les angles appropriés et le point central afin de dessiner les coins arrondis.
Par défaut, cette fonction achève la figure en cours dans le chemin et ajoute le cercle non connecté au chemin (les cercles complets sont marqués comme fermé).
Ce comportement peut être modifié avec les options appropriées.

Exemple

```
1  If OpenWindow(0, 0, 0, 400,
    200, "VectorDrawing",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2    CanvasGadget(0, 0, 0,
   400, 200)
3
4    If
   StartVectorDrawing(CanvasVectorOutput(0))
5
6      ; cercle partiel
7      AddPathCircle(100, 100,
   75, 0, 235)
8
```

```
9         ; cercle partiel avec
      des lignes vers le centre
10        MovePathCursor(300, 100)
11        AddPathCircle(300, 100,
      75, 0, 235,
      #PB_Path_Connected)
12        ClosePath()
13
14        VectorSourceColor(RGBA(255,
      0, 0, 255))
15        StrokePath(10)
16
17        StopVectorDrawing()
18      EndIf
19
20      Repeat
21        Event =
      WaitWindowEvent()
22      Until Event =
      #PB_Event_CloseWindow
23    EndIf
```

Voir aussi

MovePathCursor() , AddPathLine() , AddPathArc() , AddPathBox() , AddPathEllipse() , AddPathCurve()

OS Supportés

Tous

154.31 AddPathEllipse

Syntaxe

```
AddPathEllipse(X.d, Y.d,
    RayonX.d, RayonY.d [,
    AngleDebut.d, AngleFin.d
    [, Options]])
```

Description

Ajoute une ellipse ou une ellipse partielle sur le trajet du dessin vectoriel.

Arguments

X.d, Y.d Centre de l'ellipse.

RayonX.d, RayonY.d Rayon de l'ellipse dans la direction X et Y.

AngleDebut.d, AngleFin.d (optionnel) Angle de début et de fin de l'ellipse (secteur d'ellipse), en degrés. L'angle 0 indique l'axe X positif. Les valeurs par défaut vont de 0 à 360 degrés.

Options (optionnel) Peut être une combinaison des valeurs suivantes :

```
#PB_Path_Default          :
 Pas de comportement
 spécial (Par défaut)
#PB_Path_Relative         :
 Les positions sont
 relatives à la dernière
 position du curseur.
#PB_Path_Connected        :
 L'ellipse est reliée au
 chemin existant avec une
 ligne et non pas
 automatiquement à une
 figure fermée.
#PB_Path_CounterClockwise:
 La direction de dessin
 entre les angles de
 début et de fin est dans
 le sens antihoraire.
```

Valeur de retour

Aucune.

Remarques

Pour dessiner une ellipse suivant un angle, faire pivoter le système de coordonnées autour du centre de l'ellipse avant d'ajouter l'ellipse comme le montre l'exemple ci-dessous. Le système de coordonnées courant peut être préservé en utilisant SaveVectorState() et RestoreVectorState() . Par défaut, cette fonction achève la figure en cours dans le chemin et ajoute l'ellipse non connectée au chemin (les ellipses pleines sont marquées comme fermées). Ce comportement peut être modifié avec les options appropriées.

Exemple

```
1  If OpenWindow(0, 0, 0, 400,
    200, "VectorDrawing",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2   CanvasGadget(0, 0, 0,
   400, 200)
```

```
3
4       If StartVectorDrawing(CanvasVectorOutput(0))
5
6         ; ellipse
7         AddPathEllipse(100, 100, 80, 30)
8
9         ; ellipse avec rotation
10        SaveVectorState()
11          RotateCoordinates(300, 100, 45)
12          AddPathEllipse(300, 100, 80, 30)
13        RestoreVectorState()
14
15        VectorSourceColor(RGBA(255, 0, 0, 255))
16        StrokePath(10)
17
18        StopVectorDrawing()
19      EndIf
20
21      Repeat
22        Event = WaitWindowEvent()
23      Until Event = #PB_Event_CloseWindow
24    EndIf
```

Voir aussi

MovePathCursor() , AddPathLine() , AddPathArc() , AddPathBox() , AddPathCircle() , AddPathCurve()

OS Supportés

Tous

154.32 AddPathText

Syntaxe

```
AddPathText(Texte$)
```

Description

Ajoute le contour des caractères du texte donné à la position actuelle du curseur dans le chemin du dessin vectoriel.

Arguments

Texte$ Le texte (une seule ligne) à ajouter au chemin du dessin.

Valeur de retour

Aucune.

Remarques

Seules les polices vectorielles sont autorisées, comme TrueType, les polices bitmap ne sont donc pas autorisées.
La position courante peut être réglée avec MovePathCursor() .
Après l'appel à cette fonction, le curseur est déplacé à la fin du texte.
La fonction DrawVectorText() devrait être préférée. En effet, la conversion de texte en un chemin est une opération coûteuse et peut entraîner une perte de qualité du texte et le contour du texte peut même être légèrement différent (selon la police) par rapport à un dessin de texte directement dessiné sur la sortie avec DrawVectorText() . La fonction DrawVectorText() est plus efficace et peut faire usage de méthode de calcul, tel que le rendu sous-pixel. Par conséquent, la fonction AddPathText() ne doit être utilisée que si le texte est explicitement nécessaire comme un chemin et non pas pour dessiner un simple texte.

Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0, 400, 200)
  LoadFont(0, "Times New Roman", 20, #PB_Font_Bold)

  If StartVectorDrawing(CanvasVectorOutput(0))
    VectorFont(FontID(0), 150)

    MovePathCursor(50, 25)
    AddPathText("Text")

    VectorSourceColor(RGBA(255, 0, 0, 255))
    DashPath(3, 6)

    StopVectorDrawing()
  EndIf
```

```
16
17      Repeat
18        Event = WaitWindowEvent()
19      Until Event = #PB_Event_CloseWindow
20    EndIf
```

Voir aussi

DrawVectorText() , DrawVectorParagraph() , VectorTextWidth() , VectorTextHeight()

OS Supportés

Tous

154.33 AddPathSegments

Syntaxe

```
AddPathSegments(Segments$ [, Options])
```

Description

Ajoute des segments décrits sous forme de chaîne dans le chemin du dessin vectoriel. Cette commande peut être utilisé pour reproduire les commandes de chemin enregistrées avec la commande PathSegments() .

Arguments

Segments$ Indique les commandes à exécuter.
La description du segment se compose de commandes à une lettre suivie par un nombre approprié de coordonnées. Les valeurs peuvent être séparés par des espaces ou des virgules. Les commandes en majuscules interprètent leurs arguments comme des coordonnées absolues, les commandes équivalentes en minuscules interprètent ses arguments comme relatif au dernier segment ajouté.

```
Chemin absolu:
M x y
  MovePathCursor()

L x y
  AddPathLine()

C x1 y1 x2 y2 x3 y3
  AddPathCurve()

Z
  ClosePath()


Chemin relatif:
m x y
  MovePathCursor()

l x y
  AddPathLine()

c x1 y1 x2 y2 x3 y3
  AddPathCurve()

z
  ClosePath()
```

En plus de cette syntaxe simplifiée, la commande accepte également les descriptions de chemin dans le format défini par le Standard SVG Tiny qui contient quelques lettres de commandes supplémentaires.
Une traduction en français et de nombreuses informations cachées » ici «

Options (optionnel) Peut être une combinaison des valeurs suivantes :

```
#PB_Path_Default
  : Aucun comportement
  spécial (par défaut)
#PB_Path_Relative
  : Interprète toutes les
  coordonnées par rapport
  à la position courante
  du curseur
```

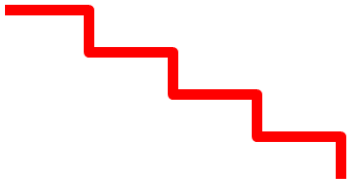
Valeur de retour

Aucune.

Exemple

```
1  If OpenWindow(0, 0, 0, 400,
     200, "AddPathSegments",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2    CanvasGadget(0, 0, 0,
     400, 200)
```

```
3
4       If StartVectorDrawing(CanvasVectorOutput(0))
5
6         AddPathSegments("M 40 20 L 120 20 L 120 60 L 200 60 L 200 100 L 280 100 L 280 140 L 360 140 L 360 180")
7         VectorSourceColor(RGBA(255, 0, 0, 255))
8         StrokePath(10, #PB_Path_RoundCorner)
9
10        StopVectorDrawing()
11      EndIf
12
13      Repeat
14        Event = WaitWindowEvent()
15      Until Event = #PB_Event_CloseWindow
16    EndIf
```



Voir aussi

PathSegments()

OS Supportés

Tous

154.34 IsInsidePath

Syntaxe

```
Resultat = IsInsidePath(X.d, Y.d [, CoordonneeSysteme])
```

Description

Teste si les coordonnées données sont dans une figure fermée dans le chemin du dessin vectoriel en cours.

Arguments

X.d, Y.d Les coordonnées du point à tester.

CoordonneeSysteme (optionnel)
Indique le système de coordonnées à utiliser.
Peut être l'une des valeurs suivantes :

```
#PB_Coordinate_Device: Le
 système de coordonnées
 du dispositif de sortie
#PB_Coordinate_Output: Le
 système de coordonnées
 comme il a été créé avec
 la fonction de sortie de
 dessin
#PB_Coordinate_User  : Le
 système de coordonnées
 des points du chemin de
 dessin (Par défaut)
#PB_Coordinate_Source: Le
 système de coordonnées
 de la source du dessin
 vectoriel
```

Valeur de retour

Renvoie une valeur non nulle si le point est dans le chemin, zéro sinon.

Remarques

Cette fonction renvoie une valeur non nulle si le point donné serait comblé par un appel à FillPath() .
Voir l'aperçu de VectorDrawing pour une introduction aux différents systèmes de coordonnées.

Exemple

```
; Cet exemple utilise la fonction IsInsidePath() pour colorer la figure en vert
; tant que la souris est à l'intérieur du chemin, en bleu sinon
;
Procedure Draw()
  x = GetGadgetAttribute(0, #PB_Canvas_MouseX)
  y = GetGadgetAttribute(0, #PB_Canvas_MouseY)

  If StartVectorDrawing(CanvasVectorOutput(0))
    VectorSourceColor(RGBA(255, 255, 255, 255))    ; efface le contenu précédent
    FillVectorOutput()
```

```
11
12          AddPathEllipse(200, 100, 150, 75)
            ; prépare le chemin
13
14          If IsInsidePath(x, y, #PB_Coordinate_Device)  ; vérifie si la souris est à l'intérieur
15            VectorSourceColor(RGBA(0, 255, 0, 255))
16          Else
17            VectorSourceColor(RGBA(0, 0, 255, 255))
18          EndIf
19
20          FillPath()

            ; remplit le chemin
21          StopVectorDrawing()
22        EndIf
23    EndProcedure
24
25    If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
26      CanvasGadget(0, 0, 0, 400, 200)
27      LoadFont(0, "Times New Roman", 20, #PB_Font_Bold)
28      Draw()
29
30      Repeat
31        Event = WaitWindowEvent()
32
33        If Event = #PB_Event_Gadget And EventGadget() = 0 And EventType() = #PB_EventType_MouseMove
34          Draw()
35        EndIf
36
37      Until Event = #PB_Event_CloseWindow
38    EndIf
```

Voir aussi

IsInsideStroke() , FillPath() , ClosePath() , ResetPath()

OS Supportés

Tous

154.35 IsInsideStroke

Syntaxe

```
Resultat =
    IsInsideStroke(X.d, Y.d,
    Largeur.d [, Options [,
    CoordonneeSysteme]])
```

Description

Teste si les coordonnées indiquées sont dans un contour qui sera dessiné par un appel à StrokePath() .

Arguments

X.d, Y.d Les coordonnées du point à tester.

Largeur.d La largeur de ligne à utiliser pour le test.

Options (optionnel) Peut être une combinaison des valeurs suivantes :

```
#PB_Path_Default       :
Pas de comportement
spécial (Par défaut)
#PB_Path_Preserve      :
Ne pas réinitialiser le
chemin après cette
fonction
#PB_Path_RoundEnd      :
Trace la ligne avec des
extrémités arrondies
#PB_Path_SquareEnd     :
Trace la ligne avec un
carré aux extrémités
#PB_Path_RoundCorner   :
Trace la ligne avec des
coins arrondis
#PB_Path_DiagonalCorner:
Trace la ligne avec les
coins coupés en diagonal
```

CoordonneeSysteme (optionnel) Indique le système de coordonnées à utiliser.
Peut être l'une des valeurs suivantes :

```
#PB_Coordinate_Device: Le
 système de coordonnées
 du dispositif de sortie
#PB_Coordinate_Output: Le
 système de coordonnées
 comme il a été créé avec
 la fonction de sortie de
 dessin
#PB_Coordinate_User  : Le
 système de coordonnées
 des points du chemin de
 dessin (Par défaut)
```

```
#PB_Coordinate_Source: Le
 système de coordonnées
 de la source du dessin
 vectoriel
```

Valeur de retour

Renvoie une valeur non nulle si le point est dans le trait, zéro sinon.

Remarques

Voir l'aperçu de VectorDrawing pour une introduction aux différents systèmes de coordonnées.

Exemple

```
; Cet exemple utilise la fonction IsInsideStroke () pour colorer la contour en vert
; tant que la souris est dessus, en bleu sinon
;
Procedure Draw()
  x = GetGadgetAttribute(0, #PB_Canvas_MouseX)
  y = GetGadgetAttribute(0, #PB_Canvas_MouseY)

  If StartVectorDrawing(CanvasVectorOutput(0))
    VectorSourceColor(RGBA(255, 255, 255, 255))          ; efface le contenu précédent
    FillVectorOutput()

    AddPathEllipse(200, 100, 150, 75)                    ; prépare le chemin

    If IsInsideStroke(x, y, 20, #PB_Path_Default, #PB_Coordinate_Device)  ; vérifie si la souris est à l'intérieur
      VectorSourceColor(RGBA(0, 255, 0, 255))
    Else
      VectorSourceColor(RGBA(0, 0, 255, 255))
    EndIf

    StrokePath(20)

  ; chemin avec motif
```

```
21        StopVectorDrawing()
22      EndIf
23    EndProcedure
24
25    If OpenWindow(0, 0, 0, 400,
       200, "VectorDrawing",
       #PB_Window_SystemMenu |
       #PB_Window_ScreenCentered)
26      CanvasGadget(0, 0, 0,
       400, 200)
27      LoadFont(0, "Times New
       Roman", 20, #PB_Font_Bold)
28      Draw()
29
30      Repeat
31        Event =
       WaitWindowEvent()
32
33        If Event =
       #PB_Event_Gadget And
       EventGadget() = 0 And
       EventType() =
       #PB_EventType_MouseMove
34          Draw()
35        EndIf
36
37      Until Event =
       #PB_Event_CloseWindow
38    EndIf
```

Voir aussi

FillPath() , DotPath() , DashPath() , CustomDashPath() , IsInsideStroke() , ResetPath()

OS Supportés

Tous

154.36 IsPathEmpty

Syntaxe

```
Resultat = IsPathEmpty()
```

Description

Teste si le chemin de dessin vectoriel en cours est vide.

Arguments

Aucun.

Valeur de retour

Renvoie une valeur non nulle si le chemin est vide, zéro sinon (le chemin contient des segments de ligne.)

Voir aussi

ResetPath() , IsInsidePath() , IsInsideStroke()

OS Supportés

Tous

154.37 StrokePath

Syntaxe

```
StrokePath(Largeur.d [,
    Options])
```

Description

Détour le chemin de dessin courant avec la source de dessin en cours. Ceci dessine le chemin sous la forme d'une ligne pleine (non pointillée).

Arguments

Largeur.d Largeur de la ligne.

Options (optionnel) Peut être une combinaison des valeurs suivantes :

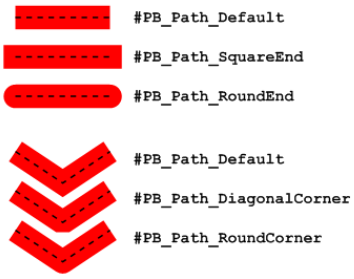
```
#PB_Path_Default       :
Pas de comportement
spécial (Par défaut)
#PB_Path_Preserve      :
Ne pas réinitialiser le
chemin après cette
fonction
#PB_Path_RoundEnd      :
Trace la ligne avec des
extrémités arrondies
#PB_Path_SquareEnd     :
Trace la ligne avec un
carré aux extrémités
#PB_Path_RoundCorner   :
Trace la ligne avec des
coins arrondis
#PB_Path_DiagonalCorner:
Trace la ligne avec les
coins coupés en diagonal
```

Valeur de retour

Aucune.

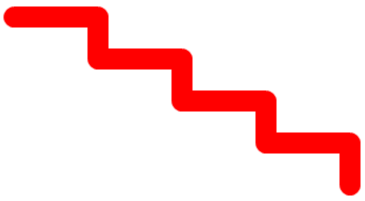
Remarques

Par défaut, le chemin est réinitialisé après l'appel à cette fonction. Ceci peut être évité avec les options appropriées.
L'image suivante montre l'effet des différentes options. Les options de coin et de fin peuvent être combinées avec l'opérateur binaire OR ('||')pour combiner les effets.



Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0, 400, 200)

  If StartVectorDrawing(CanvasVectorOutput(0))

    MovePathCursor(40, 20)
    For i = 1 To 4
      AddPathLine(80, 0, #PB_Path_Relative)
      AddPathLine(0, 40, #PB_Path_Relative)
    Next i

    VectorSourceColor(RGBA(255, 0, 0, 255))
    StrokePath(20, #PB_Path_RoundCorner|#PB_Path_RoundEnd)

    StopVectorDrawing()
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```



Voir aussi

FillPath() , DotPath() , DashPath() , CustomDashPath() , IsInsideStroke() , ResetPath()

OS Supportés

Tous

154.38 DotPath

Syntaxe

```
DotPath(Largeur.d, Distance.d
    [, Options [, Decalage.d]])
```

Description

Dessine le chemin de dessin courant avec une ligne de points.

Arguments

Largeur.d Largeur de la ligne.

Distance.d Distance entre le centre de chaque point.

Options (optionnel) Peut être une combinaison des valeurs suivantes :

```
#PB_Path_Default      :
Pas de comportement
spécial (Par défaut)
#PB_Path_Preserve     : Ne
pas réinitialiser le
chemin après cette
fonction
#PB_Path_RoundEnd     :
Trace des points ronds
#PB_Path_SquareEnd    :
Trace des points carrés
```

Decalage.d (optionnel) La distance après laquelle le motif de points commence à être dessiné.
La valeur par défaut est 0.

Valeur de retour

Aucune.

Remarques

Par défaut, le chemin est réinitialisé après l'appel à cette fonction. Ceci peut être évité avec les options appropriées.

Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0, 400, 200)

  If StartVectorDrawing(CanvasVectorOutput(0))

    MovePathCursor(40, 20)
    For i = 1 To 4
      AddPathLine(80, 0, #PB_Path_Relative)
      AddPathLine(0, 40, #PB_Path_Relative)
    Next i

    VectorSourceColor(RGBA(255, 0, 0, 255))
    DotPath(5, 10, #PB_Path_RoundEnd)

    StopVectorDrawing()
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

FillPath() , StrokePath() , DashPath() , CustomDashPath() , IsInsideStroke() , ResetPath()

OS Supportés

Tous

154.39 DashPath

Syntaxe

```
DashPath(Largeur.d,
    Longueur.d [, Options [,
    Decalage.d]])
```

Description

Dessine le chemin de dessin courant avec une série de tirets d'égale longueur et d'égal intervalle.

Arguments

Largeur.d Largeur de la ligne.
Cette valeur ne comprend pas la largeur de l'extrêmité de la ligne dessinée par un rond ou un carré.

Longueur.d Longueur de chaque trait (et l'espace entre les traits).

Options (optionnel) Peut être une combinaison des valeurs suivantes :

```
#PB_Path_Default       :
Pas de comportement
spécial (Par défaut)
#PB_Path_Preserve      :
Ne pas réinitialiser le
chemin après cette
fonction
#PB_Path_RoundEnd      :
Trace la ligne avec des
extrémités arrondies
#PB_Path_SquareEnd     :
Trace la ligne avec un
carré aux extrémités
#PB_Path_RoundCorner   :
Trace la ligne avec des
coins arrondis
#PB_Path_DiagonalCorner:
Trace la ligne avec les
coins coupés en diagonal
```

Decalage.d (optionnel) La distance après laquelle le motif commence à être dessiné.
La valeur par défaut est 0.

Valeur de retour

Aucune.

Remarques

Par défaut, le chemin est réinitialisé après l'appel à cette fonction. Ceci peut être évité avec les options appropriées.

Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0, 400, 200)

  If StartVectorDrawing(CanvasVectorOutput(0))

    MovePathCursor(40, 20)
    For i = 1 To 4
      AddPathLine(80, 0, #PB_Path_Relative)
      AddPathLine(0, 40, #PB_Path_Relative)
    Next i

    VectorSourceColor(RGBA(255, 0, 0, 255))
    DashPath(5, 15)

    StopVectorDrawing()
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

FillPath() , StrokePath() , DotPath() , CustomDashPath() , IsInsideStroke() , ResetPath()

OS Supportés

Tous

154.40 CustomDashPath

Syntaxe

```
CustomDashPath(Largeur.d,
    Tableau.d() [, Options [,
    Decalage.d]])
```

Description

Dessine le chemin de dessin courant avec un motif personnalisé à base de trait.

Arguments

Largeur.d Largeur de la ligne.

Tableau.d() Indique la longueur de chaque trait et chaque espace qui le suit. Le tableau doit avoir un nombre d'entrées pair.
Lorsque l'opération de dessin atteint la fin de la matrice, le motif se répète.
Une longueur de 0 dessinera un seul point.

Options (optionnel) Peut être une combinaison des valeurs suivantes :

```
#PB_Path_Default       :
Pas de comportement
spécial (Par défaut)
#PB_Path_Preserve      :
Ne pas réinitialiser le
chemin après cette
fonction
#PB_Path_RoundEnd      :
Trace la ligne avec des
extrémités arrondies
#PB_Path_SquareEnd     :
Trace la ligne avec un
carré aux extrémités
#PB_Path_RoundCorner   :
Trace la ligne avec des
coins arrondis
#PB_Path_DiagonalCorner:
Trace la ligne avec les
coins coupés en diagonal
```

Decalage.d (optionnel) La distance après laquelle le motif commence à être dessiné.
La valeur par défaut est 0.

Valeur de retour

Aucune.

Remarques

Par défaut, le chemin est réinitialisé après l'appel à cette fonction. Ceci peut être évité avec les options appropriées.

Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
 CanvasGadget(0, 0, 0, 400, 200)

 If StartVectorDrawing(CanvasVectorOutput(0))

    MovePathCursor(40, 20)
    For i = 1 To 4
      AddPathLine(80, 0, #PB_Path_Relative)
      AddPathLine(0, 40, #PB_Path_Relative)
    Next i

    VectorSourceColor(RGBA(255, 0, 0, 255))

    Dim dashes.d(7)
    dashes(0) = 20
    dashes(1) = 10
    dashes(2) = 0    ; dessine un point
    dashes(3) = 10
    dashes(4) = 0
    dashes(5) = 10
    dashes(6) = 20
    dashes(7) = 10
    CustomDashPath(5, dashes())

    StopVectorDrawing()
 EndIf

 Repeat
    Event = WaitWindowEvent()
 Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

FillPath() , StrokePath() , DotPath() , DashPath() , IsInsideStroke() , ResetPath()

OS Supportés

Tous

154.41 FillPath

Syntaxe

```
FillPath([Options])
```

Description

Remplit toutes les figures fermées dans le chemin du dessin vectoriel courant avec la couleur de la source de dessin.

Arguments

Options (optionnel) Peut être une combinaison des valeurs suivantes :

```
#PB_Path_Default : Pas de
 comportement spécial
 (Par défaut)
#PB_Path_Preserve: Ne pas
 réinitialiser le chemin
 après cette fonction
#PB_Path_Winding : Remplit
 tout le chemin, y
 compris les figures qui
 se chevauchent (pas de
 mode pair/impair).
```

Valeur de retour

Aucune.

Remarques

Par défaut, le chemin est réinitialisé après l'appel à cette fonction. Ceci peut être évité avec les options appropriées.
Si le chemin comporte des figures qui se chevauchent, le remplissage se fait suivant une séquence pair/impair, sauf si l'option #PB_Path_Winding est spécifiée. Les surfaces fermées de rang impair sont remplies, alors que les surfaces fermées de rang pair ne le sont pas.

Exemple

```
1  If OpenWindow(0, 0, 0, 400,
    200, "VectorDrawing",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2   CanvasGadget(0, 0, 0,
    400, 200)
```

```
3
4       If StartVectorDrawing(CanvasVectorOutput(0))
5
6         AddPathBox(50, 50, 200, 50)
7         AddPathBox(150, 75, 200, 50)
8         VectorSourceColor(RGBA(0, 0, 255, 255))
9         FillPath()
10
11        StopVectorDrawing()
12      EndIf
13
14      Repeat
15        Event = WaitWindowEvent()
16      Until Event = #PB_Event_CloseWindow
17    EndIf
```

Voir aussi

StrokePath() , DotPath() , DashPath() , CustomDashPath() , ResetPath() , ClipPath()

OS Supportés

Tous

154.42 ClipPath

Syntaxe

```
ClipPath([Options])
```

Description

Clip du chemin de dessin vectoriel en cours.

Arguments

Options (optionnel) Peut être l'une des valeurs suivantes :

```
#PB_Path_Default : Pas de comportement spécial (Par défaut)
```

```
#PB_Path_Preserve: Ne pas
 réinitialiser le chemin
 après cette fonction
```

Valeur de retour

Aucune.

Remarques

Les futures opérations de dessin ne toucheront que les zones clippées dans le chemin courant. La zone sera combinée avec tout clipping qui existait auparavant.
Par défaut, le chemin est réinitialisé après l'appel à cette fonction. Ceci peut être évité avec les options appropriées.
Il n'y a pas de fonction "UnclipPath()". La zone de découpage de la sortie de dessin peut être réduite en ajoutant plus de clipping, mais ne peut pas être agrandie. Cependant, la zone découpée peut être sauvegardée et restaurée en utilisant les fonctions SaveVectorState() et RestoreVectorState() respectivement. Ainsi, pour appliquer une découpe temporaire, il suffit d'enregistrer l'état du dessin puis de le restaurer plus tard pour revenir à la découpe originale.

Exemple

```
1  If OpenWindow(0, 0, 0, 400,
    200, "VectorDrawing",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2     CanvasGadget(0, 0, 0,
    400, 200)
3     LoadFont(0, "Times New
    Roman", 20, #PB_Font_Bold)
4
5     If
    StartVectorDrawing(CanvasVectorOutput(0))
6
7        ; Mise en place d'un
    chemin de découpe complexe
    avec des ellipses
    imbriquées
8        ;(chaque seconde
    ellipse sera tronquée)
9
10       For i = 10 To 150 Step 5
11         AddPathEllipse(200,
    100, 2*i, i)
12       Next i
13       ClipPath()
14
```

```
15        ; Dessin d'un texte avec cet écrêtage
16        VectorFont(FontID(0), 150)
17        VectorSourceColor(RGBA(255, 0, 0, 255))
18
19        MovePathCursor(50, 25)
20        DrawVectorText("Text")
21
22        StopVectorDrawing()
23      EndIf
24
25      Repeat
26        Event = WaitWindowEvent()
27      Until Event = #PB_Event_CloseWindow
28    EndIf
```

Voir aussi

OS Supportés

Tous

154.43 PathCursorX

Syntaxe

```
Resultat.d = PathCursorX()
```

Description

Renvoie la coordonnée X du curseur.

Arguments

Aucun.

Valeur de retour

La coordonnée X du curseur dans le chemin.

Remarques

Ceci est l'endroit où de nouveaux segments de chemin seront ajoutés ou du texte sera dessiné.

Voir aussi

PathCursorY() , MovePathCursor() , ResetPath()

OS Supportés

Tous

154.44 PathCursorY

Syntaxe

```
Resultat.d = PathCursorY()
```

Description

Renvoie la coordonnée Y du curseur.

Arguments

Aucun.

Valeur de retour

La coordonnée Y du curseur dans le chemin.

Remarques

Ceci est l'endroit où de nouveaux segments de chemin seront ajoutés ou du texte sera dessiné.

Voir aussi

PathCursorX() , MovePathCursor() , ResetPath()

OS Supportés

Tous

154.45 PathPointX

Syntaxe

```
Resultat.d =
    PathPointX(Distance.d)
```

Description

Renvoie la coordonnée X du point à la distance donnée depuis le début du chemin du dessin vectoriel en cours.

Arguments

Distance.d La distance depuis le début du chemin.
Si ce paramètre est négatif ou plus grand que la longueur totale du chemin, le point de départ/d'arrivé du chemin est renvoyé.
La longueur totale du chemin peut être déterminée avec PathLength() .

Valeur de retour

La coordonnée X du point de la trajectoire.

Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "PathPointX", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0, 400, 200)

  If StartVectorDrawing(CanvasVectorOutput(0))

    ; construction du chemin
    MovePathCursor(150, 125)
    AddPathCurve(0, 270, 0, -150, 350, 180)

    ; localisation & angle du point sur le chemin
    x = PathPointX(200)
    y = PathPointY(200)
    a = PathPointAngle(200)

    ; dessin du chemin
    VectorSourceColor($FF0000FF)
    StrokePath(5)

    ; dessiner un marqueur au point dans le chemin
    AddPathCircle(x, y, 10)
    VectorSourceColor($FFFF0000)
    FillPath()

    MovePathCursor(x, y)
    AddPathLine(30*Cos(Radian(a)), 30*Sin(Radian(a)), #PB_Path_Relative)
    StrokePath(5)

    StopVectorDrawing()
  EndIf

  Repeat
    Event = WaitWindowEvent()
```

```
33      Until Event =
        #PB_Event_CloseWindow
34    EndIf
```

Voir aussi

PathPointY() , PathPointAngle() , PathLength()

OS Supportés

Tous

154.46 PathPointY

Syntaxe

```
Resultat.d =
    PathPointY(Distance.d)
```

Description

Renvoie la coordonnée Y du point à la distance donnée depuis le début du chemin du dessin vectoriel en cours.

Arguments

Distance.d La distance depuis le début du chemin.
Si ce paramètre est négatif ou plus grand que la longueur totale du chemin, le point de départ/d'arrivé du chemin est renvoyé. La longueur totale du chemin peut être déterminée avec PathLength() .

Valeur de retour

La coordonnée Y du point de la trajectoire.

Exemple

```
1   If OpenWindow(0, 0, 0, 400,
      200, "PathPointY",
      #PB_Window_SystemMenu |
      #PB_Window_ScreenCentered)
2     CanvasGadget(0, 0, 0,
      400, 200)
3
```

```
 If StartVectorDrawing(CanvasVectorOutput(0))

    ; construction du chemin
    MovePathCursor(150, 125)
    AddPathCurve(0, 270, 0, -150, 350, 180)

    ; localisation & angle du point sur le chemin
    x = PathPointX(200)
    y = PathPointY(200)
    a = PathPointAngle(200)

    ; dessin du chemin
    VectorSourceColor($FF0000FF)
    StrokePath(5)

    ; dessiner un marqueur au point dans le chemin
    AddPathCircle(x, y, 10)
    VectorSourceColor($FFFF0000)
    FillPath()

    MovePathCursor(x, y)
    AddPathLine(30*Cos(Radian(a)), 30*Sin(Radian(a)), #PB_Path_Relative)
    StrokePath(5)

    StopVectorDrawing()
 EndIf

 Repeat
    Event = WaitWindowEvent()
 Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

PathPointX() , PathPointAngle() , PathLength()

OS Supportés

Tous

154.47 PathPointAngle

Syntaxe

```
Resultat.d =
    PathPointAngle(Distance.d)
```

Description

Renvoie l'angle du chemin au point à la distance donnée depuis le début du chemin du dessin vectoriel en cours.

Arguments

Distance.d La distance depuis le début du chemin.
Si ce paramètre est négatif ou plus grand que la longueur totale du chemin, le point de départ/d'arrivé du chemin est renvoyé.
La longueur totale du chemin peut être déterminée avec PathLength() .

Valeur de retour

L'angle de la trajectoire au point donné en degrés.
L'angle 0 marque l'axe des X positifs.

Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "PathPointAngle", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0, 400, 200)

  If StartVectorDrawing(CanvasVectorOutput(0))

    ; construction du chemin
    MovePathCursor(150, 125)
    AddPathCurve(0, 270, 0, -150, 350, 180)

    ; localisation & angle du point sur le chemin
    x = PathPointX(200)
    y = PathPointY(200)
    a = PathPointAngle(200)

    ; dessin du chemin
    VectorSourceColor($FF0000FF)
    StrokePath(5)

    ; dessiner un marqueur au point dans le chemin
    AddPathCircle(x, y, 10)
```

```
21        VectorSourceColor($FFFF0000)
22        FillPath()
23
24        MovePathCursor(x, y)
25        AddPathLine(30*Cos(Radian(a)),
      30*Sin(Radian(a)),
      #PB_Path_Relative)
26        StrokePath(5)
27
28        StopVectorDrawing()
29      EndIf
30
31      Repeat
32        Event =
      WaitWindowEvent()
33      Until Event =
      #PB_Event_CloseWindow
34    EndIf
```

Voir aussi

PathPointX() , PathPointY() , PathLength()

OS Supportés

Tous

154.48 PathLength

Syntaxe

```
Resultat.d = PathLength()
```

Description

Renvoie la longueur totale du chemin du dessin vectoriel en cours.

Arguments

Aucun.

Valeur de retour

Renvoie la longueur du chemin courant.

Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "PathLength", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0, 400, 200)

  If StartVectorDrawing(CanvasVectorOutput(0))

    ; construction du chemin
    MovePathCursor(150, 125)
    AddPathCurve(0, 270, 0, -150, 350, 180)

    ; longueur du chemin
    Debug "Path length: " + PathLength()

    ; dessin du chemin
    VectorSourceColor($FF0000FF)
    StrokePath(5)

    StopVectorDrawing()
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

PathPointX() , PathPointY() , PathPointAngle()

OS Supportés

Tous

154.49 PathBoundsX

Syntaxe

```
Resultat.d = PathBoundsX()
```

Description

Renvoie la coordonnée X (du coin en haut et gauche) de la zone de délimitation du chemin de dessin vectoriel en cours.

Arguments

Aucun.

Valeur de retour

La coordonnée X de la zone de délimitation.

Remarques

Le résultat est la coordonnée X la plus petite du chemin en cours.

Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "PathBoundsX", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0, 400, 200)

  If StartVectorDrawing(CanvasVectorOutput(0))

    ; construction du chemin
    MovePathCursor(150, 125)
    AddPathCurve(0, 270, 0, -150, 350, 180)

    ; coordonnées et dimensions de la zone de délimitation du chemin
    x = PathBoundsX()
    y = PathBoundsY()
    w = PathBoundsWidth()
    h = PathBoundsHeight()

    ; dessin du chemin
    VectorSourceColor($FF0000FF)
    StrokePath(5)

    ; dessin de la zone de délimitation du chemin
    AddPathBox(x, y, w, h)
    VectorSourceColor($FF000000)
    DashPath(2, 5)

    StopVectorDrawing()
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

PathBoundsY() , PathBoundsWidth() , PathBoundsHeight()

OS Supportés

Tous

154.50 PathBoundsY

Syntaxe

```
Resultat.d = PathBoundsY()
```

Description

Renvoie la coordonnée Y (du coin en haut et gauche) de la zone de délimitation du chemin de dessin vectoriel en cours.

Arguments

Aucun.

Valeur de retour

La coordonnée Y de la zone de délimitation.

Remarques

Le résultat est la coordonnée Y la plus petite du chemin en cours.

Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "PathBoundsY", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0, 400, 200)

  If StartVectorDrawing(CanvasVectorOutput(0))

    ; construction du chemin
    MovePathCursor(150, 125)
    AddPathCurve(0, 270, 0, -150, 350, 180)
```

```
9
10      ; coordonnées et
        dimensions de la zone de
        délimitation du chemin
11      x = PathBoundsX()
12      y = PathBoundsY()
13      w = PathBoundsWidth()
14      h = PathBoundsHeight()
15
16      ; dessin du chemin
17      VectorSourceColor($FF0000FF)
18      StrokePath(5)
19
20      ; dessin de la zone de
        délimitation du chemin
21      AddPathBox(x, y, w, h)
22      VectorSourceColor($FF000000)
23      DashPath(2, 5)
24
25      StopVectorDrawing()
26    EndIf
27
28    Repeat
29      Event =
      WaitWindowEvent()
30    Until Event =
      #PB_Event_CloseWindow
31  EndIf
```

Voir aussi

PathBoundsX() , PathBoundsWidth() , PathBoundsHeight()

OS Supportés

Tous

154.51 PathBoundsWidth

Syntaxe

```
Resultat.d = PathBoundsWidth()
```

Description

Renvoie la longueur de la zone de délimitation du chemin de dessin vectoriel en cours.

Arguments

Aucun.

Valeur de retour

La longueur de la zone de délimitation.

Remarques

Le résultat est la différence entre la coordonnée X la plus élevée et la coordonnée X la plus petite du chemin en cours.

Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "PathBoundsWidth", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0, 400, 200)

  If StartVectorDrawing(CanvasVectorOutput(0))

    ; construction du chemin
    MovePathCursor(150, 125)
    AddPathCurve(0, 270, 0, -150, 350, 180)

    ; coordonnées et dimensions de la zone de délimitation du chemin
    x = PathBoundsX()
    y = PathBoundsY()
    w = PathBoundsWidth()
    h = PathBoundsHeight()

    ; dessin du chemin
    VectorSourceColor($FF0000FF)
    StrokePath(5)

    ; dessin de la zone de délimitation du chemin
    AddPathBox(x, y, w, h)
    VectorSourceColor($FF000000)
    DashPath(2, 5)

    StopVectorDrawing()
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

PathBoundsX() , PathBoundsY() , PathBoundsHeight()

OS Supportés

Tous

154.52 PathBoundsHeight

Syntaxe

```
Resultat.d =
    PathBoundsHeight()
```

Description

Renvoie la hauteur de la zone de délimitation du chemin de dessin vectoriel en cours.

Arguments

Aucun.

Valeur de retour

La hauteur de la zone de délimitation.

Remarques

Le résultat est la différence entre la coordonnée Y la plus élevée et la coordonnée Y la plus petite du chemin en cours.

Exemple

```
1   If OpenWindow(0, 0, 0, 400,
     200, "PathBoundsHeight",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2     CanvasGadget(0, 0, 0,
    400, 200)
3
4     If
    StartVectorDrawing(CanvasVectorOutput(0))
5
6       ; construction du chemin
```

```
7         MovePathCursor(150, 125)
8         AddPathCurve(0, 270, 0,
      -150, 350, 180)
9
10        ; coordonnées et
      dimensions de la zone de
      délimitation du chemin
11        x = PathBoundsX()
12        y = PathBoundsY()
13        w = PathBoundsWidth()
14        h = PathBoundsHeight()
15
16        ; dessin du chemin
17        VectorSourceColor($FF0000FF)
18        StrokePath(5)
19
20        ; dessin de la zone de
      délimitation du chemin
21        AddPathBox(x, y, w, h)
22        VectorSourceColor($FF000000)
23        DashPath(2, 5)
24
25        StopVectorDrawing()
26      EndIf
27
28      Repeat
29        Event =
      WaitWindowEvent()
30      Until Event =
      #PB_Event_CloseWindow
31    EndIf
```

Voir aussi

PathBoundsX() , PathBoundsY() , PathBoundsWidth()

OS Supportés

Tous

154.53 PathSegments

Syntaxe

```
Resultat\$ = PathSegments()
```

Description

Renvoie une chaîne décrivant le chemin de dessin vectoriel en cours.

Arguments

Aucun.

Valeur de retour

La chaîne renvoyée contient une série de lettre suivie par le nombre approprié de coordonnées. Chaque valeur est séparée par un seul espace. Toutes les coordonnées sont absolues.

```
M x y
  MovePathCursor()

L x y
  AddPathLine()

C x1 y1 x2 y2 x3 y3
  AddPathCurve()

Z
  ClosePath()
```

Remarques

Le résultat peut être utilisé pour examiner le chemin courant ou bien pour reproduire le même chemin plus tard, en utilisant les commandes AddPathSegments() .
Il n'y a pas de chaîne pour les commandes AddPathCircle() et AddPathEllipse() , et leurs résultats sont convertis en interne en courbes par la bibliothèque de dessin vectoriel.

Exemple

```
1   If OpenWindow(0, 0, 0, 400,
    200, "PathSegments",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2    CanvasGadget(0, 0, 0,
    400, 200)
3
4    If
    StartVectorDrawing(CanvasVectorOutput(0))
5
6       MovePathCursor(40, 20)
7       For i = 1 To 4
8         AddPathLine(80, 0,
    #PB_Path_Relative)
9         AddPathLine(0, 40,
    #PB_Path_Relative)
10      Next i
11
12      ; afficher les path
    segments
13      Debug PathSegments()
```

```
14
15        VectorSourceColor(RGBA(255, 0, 0, 255))
16        StrokePath(10, #PB_Path_RoundCorner)
17
18        StopVectorDrawing()
19      EndIf
20
21      Repeat
22        Event = WaitWindowEvent()
23      Until Event = #PB_Event_CloseWindow
24    EndIf
```

Voir aussi

AddPathSegments()

OS Supportés

Tous

154.54 VectorSourceColor

Syntaxe

```
VectorSourceColor(Couleur)
```

Description

Sélectionne une seule couleur comme source pour les opérations de dessin vectoriel telles que FillPath() , StrokePath() et autres.

Arguments

Couleur La couleur RGBA 32 bits comprenant la transparence alpha.

Valeur de retour

Aucune.

Voir aussi

VectorSourceLinearGradient() , VectorSourceCircularGradient() , VectorSourceImage() , FillPath() , FillVectorOutput() , StrokePath() , DotPath() , DashPath() , CustomDashPath()

OS Supportés

Tous

154.55 VectorSourceLinearGradient

Syntaxe

```
VectorSourceLinearGradient(X1.d,
    Y1.d, X2.d, Y2.d)
```

Description

Sélectionne un dégradé de couleur linéaire comme source pour les opérations de dessin vectoriel telles que FillPath() ou StrokePath() .

Arguments

X1.d, Y1.d Indique le point qui représente le début (position 0.0) du gradient. Les coordonnées sont indiquées en termes de système de coordonnées #PB_Coordinate_Source.

X2.d, Y2.d Indique le point qui représente la fin (position 1.0) du gradient. Les coordonnées sont indiquées en termes de système de coordonnées #PB_Coordinate_Source.

Valeur de retour

Aucune.

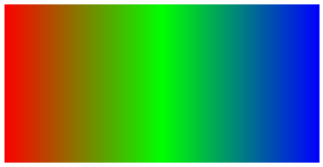
Remarques

Initialement, le gradient est solide et noir. La couleur d'arrêt doit être ajoutée à la fonction VectorSourceGradientColor() . Voir l'aperçu de VectorDrawing pour une introduction aux différents systèmes de coordonnées.
Le gradient de couleur est défini uniquement dans la zone entre les points (X1, Y1) et (X2, Y2). En dehors de ces points, la couleur de la source est fonction du système d'exploitation, de sorte que les opérations de dessin à l'extérieur de cette région de gradient doivent être évitées.

Exemple

```
1  If OpenWindow(0, 0, 0, 400,
     200, "VectorDrawing",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2     CanvasGadget(0, 0, 0,
     400, 200)
3
4     If
     StartVectorDrawing(CanvasVectorOutput(0))
```

```
5
6       VectorSourceLinearGradient(50,
    0, 350, 0)
7       VectorSourceGradientColor(RGBA(255,
    0, 0, 255), 0.0)
8       VectorSourceGradientColor(RGBA(0,
    255, 0, 255), 0.5)
9       VectorSourceGradientColor(RGBA(0,
    0, 255, 255), 1.0)
10
11      AddPathBox(50, 25, 300,
    150)
12      FillPath()
13
14      StopVectorDrawing()
15    EndIf
16
17    Repeat
18      Event =
    WaitWindowEvent()
19    Until Event =
    #PB_Event_CloseWindow
20  EndIf
```



Voir aussi

VectorSourceGradientColor() ,
VectorSourceCircularGradient() ,
VectorSourceColor() , VectorSourceImage()

OS Supportés

Tous

154.56 VectorSourceCircularGradient

Syntaxe

```
VectorSourceCircularGradient(X.d,
    Y.d, Rayon.d, [CentreX.d,
    CentreY.d])
```

Description

Sélectionne un dégradé de couleur linéaire comme source pour les opérations de dessin vectoriel telles que FillPath() ou StrokePath() .

Arguments

X.d, Y.d Les coordonnées du centre du cercle qui définit le gradient.
Les coordonnées sont indiquées en termes de système de coordonnées #PB_Coordinate_Source.
Le centre du cercle représente le début du gradient (position 0.0) et le périmètre du cercle représente la fin du gradient (position 1.0).

Rayon.d Le rayon du cercle définissant le gradient.

CentreX.d, CentreY.d (optionnel) Indique un décalage pour le point de départ du gradient à partir du centre du cercle. Avec ces paramètres, le gradient peut commencer n'importe où, entre le centre et la circonférence.

Valeur de retour

Aucune.

Remarques

Initialement, le gradient est solide et noir.
La couleur d'arrêt doit être ajoutée à la fonction VectorSourceGradientColor() .
Voir l'aperçu de VectorDrawing pour une introduction aux différents systèmes de coordonnées.
Le gradient de couleur est défini uniquement dans la zone entre le centre et la circonférence. En dehors de ces points, la couleur de la source utilisée est fonction du système d'exploitation, de sorte que les opérations de dessin à l'extérieur de cette région de gradient doivent être évitées.

Exemple

```
1   If OpenWindow(0, 0, 0, 400,
    200, "VectorDrawing",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2    CanvasGadget(0, 0, 0,
    400, 200)
3
4    If
    StartVectorDrawing(CanvasVectorOutput(0))
5
6       VectorSourceCircularGradient(200,
    100, 150, -50, -50)
7       VectorSourceGradientColor(RGBA(255,
    255, 255, 255), 0.0)
8       VectorSourceGradientColor(RGBA(0,
    0, 0, 255), 1.0)
```

```
9
10        FillVectorOutput()
11
12        StopVectorDrawing()
13      EndIf
14
15      Repeat
16        Event =
        WaitWindowEvent()
17      Until Event =
        #PB_Event_CloseWindow
18    EndIf
```

Voir aussi

VectorSourceGradientColor() ,
VectorSourceLinearGradient() ,
VectorSourceColor() , VectorSourceImage()

OS Supportés

Tous

154.57 VectorSourceGradientColor

Syntaxe

```
VectorSourceGradientColor(Couleur,
    Position.d)
```

Description

Ajoute un nouvelle position de couleur d'arrêt au gradient défini par VectorSourceLinearGradient() ou VectorSourceCircularGradient() .

Arguments

Couleur La couleur RGBA 32 bits comprenant la transparence alpha.

Position.d La position à laquelle ajouter la couleur.
La valeur doit être comprise entre 0,0 (le début du gradient) et 1.0 (la fin du gradient).

Valeur de retour

Aucune.

Remarques

Un gradient doit au moins avoir une couleur à la position 0.0 et une à la position 1.0. Si ces deux positions n'ont pas de couleur, elles prennent par défaut la couleur noire. Un nombre quelconque de positions de couleur peut être ajouté à un gradient.

Exemple

```
If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0, 400, 200)

  If StartVectorDrawing(CanvasVectorOutput(0))

    VectorSourceLinearGradient(50, 0, 350, 0)
    VectorSourceGradientColor(RGBA(255, 0, 0, 255), 0.0)
    VectorSourceGradientColor(RGBA(0, 255, 0, 255), 0.5)
    VectorSourceGradientColor(RGBA(0, 0, 255, 255), 1.0)

    AddPathBox(50, 25, 300, 150)
    FillPath()

    StopVectorDrawing()
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

VectorSourceLinearGradient() , VectorSourceCircularGradient()

OS Supportés

Tous

154.58 VectorSourceImage

Syntaxe

```
VectorSourceImage(ImageID [,
    Transparence [, Largeur.d,
    Hauteur.d [, Options]]])
```

Description

Sélectionne une image en tant que source pour les opérations de dessin vectoriel telles que FillPath() ou StrokePath() .

Arguments

ImageID L'image à utiliser comme source. Utiliser la fonction ImageID() pour obtenir l'identifiant de l'image.

Transparence (optionnel) La transparence (alpha) à appliquer à l'image source.
Cette transparence est appliquée en plus de tous les pixels transparents déjà présents dans l'image source. La valeur par défaut est 255 (pas de transparence supplémentaire).

Largeur.d, Hauteur.d (optionnel) Spécifie une largeur et une hauteur de l'image.
Les valeurs sont interprétées en termes de coordonnées système #PB_Coordinate_Source. Si non spécifiées alors les dimensions de l'image source (en pixels) sont converties dans l'unité de la sortie de dessin vectoriel (taille d'origine).

Options (optionnel) Peut-être l'une des valeurs suivantes :

```
#PB_VectorImage_Default:
 Les zones en dehors de
 l'image source sont
 transparentes (Par
 défaut)
#PB_VectorImage_Repeat :
 L'image source est
 répétée pour couvrir
 toute la zone de dessin
```

Valeur de retour

Aucune.

Remarques

Ces fonctions utiliseront chaque pixel de l'image sur la sortie de dessin où il y a quelque chose à dessiner.
Voir l'aperçu de VectorDrawing pour une introduction aux différents systèmes de coordonnées.

En transformant le système de coordonnées #PB_Coordinate_Source, l'image de la source peut être transformée (déplacée, tournée, étirées, inclinée).
Voir le deuxième exemple ci-dessous pour une démonstration.

Exemple : Image source répétée

```
If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  CanvasGadget(0, 0, 0, 400, 200)

  LoadImage(0, #PB_Compiler_Home + "examples/Sources/Data/PureBasicLogo.bmp")

  If StartVectorDrawing(CanvasVectorOutput(0))

    AddPathBox(50, 50, 200, 50)
    AddPathBox(150, 75, 200, 50)

    VectorSourceImage(ImageID(0), 255, ImageWidth(0), ImageHeight(0), #PB_VectorImage_Repeat)
    StrokePath(20)

    StopVectorDrawing()
  EndIf

  Repeat
    Event = WaitWindowEvent()
  Until Event = #PB_Event_CloseWindow
EndIf
```

Exemple : Image source tournée et renversée

```
If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing",
```

```
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
 2   CanvasGadget(0, 0, 0,
    400, 200)
 3
 4   LoadImage(0,
    #PB_Compiler_Home +
    "examples/Sources/Data/PureBasicLogo.bmp")
 5   LoadFont(0, "Impact", 20,
    #PB_Font_Bold)
 6
 7   If
    StartVectorDrawing(CanvasVectorOutput(0))
 8
 9     FlipCoordinatesY(50,
    #PB_Coordinate_Source)
10     RotateCoordinates(50,
    50, -45,
    #PB_Coordinate_Source)
11     VectorSourceImage(ImageID(0),
    255, ImageWidth(0),
    ImageHeight(0),
    #PB_VectorImage_Repeat)
12
13     VectorFont(FontID(0),
    150)
14     MovePathCursor(20, 20)
15     DrawVectorText("TEXT")
16
17     StopVectorDrawing()
18   EndIf
19
20   Repeat
21     Event =
    WaitWindowEvent()
22   Until Event =
    #PB_Event_CloseWindow
23 EndIf
```

Voir aussi

VectorSourceColor() ,
VectorSourceLinearGradient() ,
VectorSourceCircularGradient()

OS Supportés

Tous

154.59 DrawVectorImage

Syntaxe

```
DrawVectorImage(ImageID [,
    Transparence [, Largeur.d,
    Hauteur.d]])
```

Description

Dessine l'image spécifiée directement sur la sortie de dessin vectoriel.

Arguments

ImageID L'image à utiliser comme source. Utiliser la fonction ImageID() pour obtenir l'identifiant de l'image.

Transparence (optionnel) La transparence (alpha) à appliquer à l'image source.
Cette transparence est appliquée en plus de tous les pixels transparents déjà présents dans l'image source. La valeur par défaut est 255 (pas de transparence supplémentaire).

Largeur.d, Hauteur.d (optionnel) Spécifie une largeur et une hauteur de l'image.
Les valeurs sont interprétées en termes de coordonnées système #PB_Coordinate_Source. Si non spécifiées alors les dimensions de l'image source (en pixels) sont converties dans l'unité de la sortie de dessin vectoriel (taille d'origine).

Valeur de retour

Aucune.

Remarques

L'image sera dessinée à l'emplacement du curseur dans le chemin .
A la fin, le curseur sera placé dans le coin en bas à droite de l'image.

Exemple

```
1   If OpenWindow(0, 0, 0, 400,
      200, "VectorDrawing",
      #PB_Window_SystemMenu |
      #PB_Window_ScreenCentered)
2     CanvasGadget(0, 0, 0,
      400, 200)
3
```

```
    LoadImage(0, #PB_Compiler_Home + "examples/Sources/Data/PureBasicLogo.bmp")

    If StartVectorDrawing(CanvasVectorOutput(0))

      MovePathCursor(50, 50)
      DrawVectorImage(ImageID(0), 127)

      MovePathCursor(75, 75)
      DrawVectorImage(ImageID(0), 127, ImageWidth(0) / 2, ImageHeight(0))

      MovePathCursor(120, 0)
      RotateCoordinates(120, 0, 35)
      DrawVectorImage(ImageID(0), 127)

      StopVectorDrawing()
    EndIf

    Repeat
      Event = WaitWindowEvent()
    Until Event = #PB_Event_CloseWindow
  EndIf
```

Voir aussi

MovePathCursor() , PathCursorX() , PathCursorY() , VectorSourceImage()

OS Supportés

Tous

154.60 DrawVectorText

Syntaxe

```
DrawVectorText(Texte$)
```

Description

Dessine une seule ligne de texte à l'emplacement actuel du curseur dans le chemin .

Arguments

Texte$ Le texte à dessiner (une seule ligne).

Valeur de retour

Aucune.

Remarques

A la fin, le curseur sera déplacé à la fin du texte.
La police à utiliser peut être réglée avec VectorFont() .
Seules les polices vectorielles sont autorisées, comme TrueType, les polices bitmap ne sont donc pas autorisées.
La fonction DrawVectorParagraph() est plus adaptée pour dessiner du texte multiligne, elle offre en plus une mise en page automatique, prend en compte les sauts de ligne, etc. Sinon, il vous faudra appel plusieurs fois à DrawVectorText() pour afficher plusieurs lignes, sans oublier de prendre en compte les dimensions du texte avec VectorTextWidth() et VectorTextHeight() .

Exemple

```
1   If OpenWindow(0, 0, 0, 400, 200, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2     CanvasGadget(0, 0, 0, 400, 200)
3     LoadFont(0, "Impact", 20, #PB_Font_Bold)
4
5     If StartVectorDrawing(CanvasVectorOutput(0))
6
7       VectorFont(FontID(0), 25)
8       VectorSourceColor(RGBA(0, 0, 0, 80))
9       Text$ = "Le rapide renard brun sauta par dessus le chien paresseux"
10
11      For i = 1 To 6
```

```
12          MovePathCursor(200 -
    VectorTextWidth(Text$)/2,
    100 -
    VectorTextHeight(Text$)/2)
13          DrawVectorText(Text$)
14          RotateCoordinates(200,
    100, 30)
15        Next i
16
17        StopVectorDrawing()
18      EndIf
19
20      Repeat
21        Event =
    WaitWindowEvent()
22      Until Event =
    #PB_Event_CloseWindow
23    EndIf
```

Voir aussi

VectorTextWidth() , VectorTextHeight() , DrawVectorParagraph() , AddPathText() , VectorFont()

OS Supportés

Tous

154.61 DrawVectorParagraph

Syntaxe

```
DrawVectorParagraph(Texte$,
    Largeur.d, Hauteur.d [,
    Options])
```

Description

Dessine un paragraphe de texte (plusieurs lignes) dans une boîte de sélection avec des sauts de lignes automatiques.

Arguments

Texte$ Le texte à dessiner.

Largeur.d La largeur du paragraphe. Les sauts de ligne seront ajoutés si le texte est plus long que la largeur spécifiée.

Hauteur.d La hauteur maximale du paragraphe.
Si le texte ne correspond pas à cette hauteur il sera tronqué. La hauteur requise pour un paragraphe peut être calculée avec VectorParagraphHeight() .

Options (optionnel) Peut-être l'une des valeurs suivantes :

```
#PB_VectorParagraph_Left
: Le paragraphe est
aligné à gauche (Par
défaut)
#PB_VectorParagraph_Right
: Le paragraphe est
aligné à droite
#PB_VectorParagraph_Center:
Le paragraphe est centré
#PB_VectorParagraph_Block
: Le paragraphe est
justifié (sauf Windows)
```

Valeur de retour

Aucune.

Remarques

Seules les polices vectorielles sont autorisées, comme TrueType, les polices bitmap ne sont donc pas autorisées.
La police à utiliser peut être réglée avec VectorFont() .
Si le texte ne rentre pas en entier dans la boîte, il sera tronqué.

Exemple

```
1  If OpenWindow(0, 0, 0, 400,
   250, "VectorDrawing",
   #PB_Window_SystemMenu |
   #PB_Window_ScreenCentered)
2   CanvasGadget(0, 0, 0,
   400, 250)
3   LoadFont(0, "Times New
   Roman", 20)
4
5   If
   StartVectorDrawing(CanvasVectorOutput(0))
6
7     Texte$ = "Chaque sortie
   de dessin a une unité de
   mesure par défaut. L'unité
   par défaut est le pixel " +
8            "pour les
   écrans ou les images et
   les points pour les
   imprimantes et les images
   vectorielles. " +
```

```
9               "Il est
    cependant possible de
    sélectionner une unité de
    mesure différente avec
    ImageVectorOutput(), " +
10              "PrinterVectorOutput()
    ou une fonction similaire."
11
12      VectorFont(FontID(0),
    18)
13
14      MovePathCursor(25, 25)
15      DrawVectorParagraph(Texte$,
    350, 200)
16
17
18      StopVectorDrawing()
19    EndIf
20
21    Repeat
22      Event =
    WaitWindowEvent()
23    Until Event =
    #PB_Event_CloseWindow
24  EndIf
```

Every drawing output has a default unit of measurement. The default unit is pixels for screen or raster image outputs and points for printer or vector image outputs. It is however possible to select a different unit of measurement for the output when creating it with the ImageVectorOutput(), PrinterVectorOutput() or similar function.

Voir aussi

VectorParagraphHeight() , DrawVectorText() , AddPathText() , VectorFont()

OS Supportés

Tous

154.62 VectorFont

Syntaxe

```
VectorFont(PoliceID [,
    Taille.d])
```

Description

La police à utiliser pour le dessin vectoriel. Seules les polices vectorielles sont autorisées, comme TrueType, les polices bitmap ne sont donc pas autorisées.

Arguments

PoliceID Le numéro d'identification de la police à utiliser pour le dessin.

Taille.d (optionnel) La taille de la police. La taille est mesurée en unités de mesure utilisée par la sortie de dessin vectoriel. Si aucune taille n'est spécifiée, la taille utilisée dans la commande LoadFont() sera convertie en unité de dessin vectoriel en cours.

Valeur de retour

Aucune.

Remarques

Vous ne pouvez pas utiliser de police enregistrée avec RegisterFontFile()

Voir aussi

DrawVectorText() , DrawVectorParagraph() , VectorTextWidth() , VectorTextHeight() , VectorParagraphHeight()

OS Supportés

Tous

154.63 VectorTextWidth

Syntaxe

```
Resultat.d =
    VectorTextWidth(Texte$ [,
    Options])
```

Description

Mesure la largeur du texte donné dans la police de dessin vectoriel en cours.

Arguments

Texte$ Le texte (une seule ligne) à mesurer.

Options (optionnel) Peut être une combinaison des valeurs suivantes :

```
    #PB_VectorText_Default:
    Renvoie la zone de
    délimitation logique du
    texte
    #PB_VectorText_Visible:
    Renvoie la zone de
    délimitation visible du
    texte
```

```
#PB_VectorText_Offset :
Renvoie le décalage de
la zone de délimitation
de la position actuelle
à la place de la largeur
```

Valeur de retour

Renvoie la largeur du texte en unités de mesure utilisée par la sortie du dessin vectoriel.

Remarques

Les dimensions du texte peuvent être définies en terme de deux boîtes englobantes :
La "boîte englobante logique" d'un caractère ou d'un texte définit l'espace que le curseur doit parcourir pour dessiner correctement le texte. Cependant, certains caractères peuvent s'étendre au-delà de cette boîte (par exemple en cas de cursive ou d'Empattement (sérif). Intéressant pour déterminer l'emplacement d'un texte.
La "boîte englobante visible" d'un caractère ou d'un texte définit la zone dans laquelle le texte est effectivement dessiné.
Cette zone est généralement plus grande que la zone de délimitation logique. Les dimensions visibles du texte peuvent être récupérées en spécifiant l'option #PB_VectorText_Visible. Les dimensions visibles du texte peuvent être en décalage par rapport aux dimensions logiques. Ce décalage peut être calculé en spécifiant l'option #PB_VectorText_Offset.
L'exemple suivant montre un texte avec la zone de délimitation logique en bleu, la zone de délimitation visible en rouge et l'emplacement de la ligne de base en vert. Le début du texte est dessiné dans le coin supérieur gauche de la zone de délimitation logique (bleu).

Exemple

```
1  If OpenWindow(0, 0, 0, 500,
     250, "VectorDrawing",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2     CanvasGadget(0, 0, 0,
     500, 250)
3     LoadFont(0, "Monotype
     Corsiva", 20,
     #PB_Font_Italic)
4
```

```
5       If StartVectorDrawing(CanvasVectorOutput(0))
6
7          VectorFont(FontID(0), 125)
8          Texte$ = "Exemple"
9
10         ; dessin du texte
11         MovePathCursor(25, 25)
12         DrawVectorText(Texte$)
13
14         ; dessin  de la boîte englobante logique (en bleu)
15         AddPathBox(25, 25, VectorTextWidth(Texte$), VectorTextHeight(Texte$))
16         VectorSourceColor(RGBA(0, 0, 255, 255))
17         DashPath(2, 10)
18
19         ; dessin de la boîte englobante visible (en rouge)
20         AddPathBox(25 + VectorTextWidth(Texte$, #PB_VectorText_Visible|#PB_VectorText_Offset),
21                    25 + VectorTextHeight(Texte$, #PB_VectorText_Visible|#PB_VectorText_Offset),
22                    VectorTextWidth(Texte$, #PB_VectorText_Visible),
23                    VectorTextHeight(Texte$, #PB_VectorText_Visible))
24         VectorSourceColor(RGBA(255, 0, 0, 255))
25         DashPath(2, 10)
26
27         ; dessin de la ligne de base (en vert)
28         MovePathCursor(25, 25 + VectorTextHeight(Texte$, #PB_VectorText_Baseline))
29         AddPathLine(VectorTextWidth(Texte$), 0, #PB_Path_Relative)
30         VectorSourceColor(RGBA(0, 255, 0, 255))
31         DashPath(2, 10)
32
33         StopVectorDrawing()
34      EndIf
35
36      Repeat
37         Event = WaitWindowEvent()
38      Until Event = #PB_Event_CloseWindow
39   EndIf
```

Voir aussi

VectorTextHeight() , DrawVectorText() , DrawVectorParagraph() , VectorParagraphHeight() , VectorFont()

OS Supportés

Tous

154.64 VectorTextHeight

Syntaxe

```
Resultat.d =
    VectorTextHeight(Texte$ [,
    Options])
```

Description

Mesure la hauteur du texte donné dans la police de dessin vectoriel en cours.

Arguments

Texte$ Le texte (une seule ligne) à mesurer.

Options (optionnel) Peut être une combinaison des valeurs suivantes :

```
#PB_VectorText_Default :
 Renvoie la zone de
 délimitation logique du
 texte
#PB_VectorText_Visible :
 Renvoie la zone de
 délimitation visible du
 texte
#PB_VectorText_Offset  :
 Renvoie le décalage de
 la zone de délimitation
 de la position actuelle
 à la place de la largeur
#PB_VectorText_Baseline:
 Renvoie le décalage de
 la ligne de base du
 texte à partir de la
 position courante
```

Valeur de retour

Renvoie la hauteur du texte en unité de mesure utilisée par la sortie du dessin vectoriel.

Valeur de retour

Remarques

Les dimensions du texte peuvent être définies en terme de deux boîtes englobantes :
La "boîte englobante logique" d'un caractère ou d'un texte définit l'espace que le curseur doit parcourir pour dessiner correctement le texte. Cependant, certains caractères peuvent s'étendre au-delà de cette boîte (par exemple en cas de cursive ou d'empattement (sérif). Intéressant pour déterminer l'emplacement d'un texte.
La "boîte englobante visible" d'un caractère ou d'un texte définit la zone dans laquelle le texte est effectivement dessiné.
Cette zone est généralement plus grande que la zone de délimitation logique. Les dimensions visibles du texte peuvent être récupérées en spécifiant l'option #PB_VectorText_Visible. Les dimensions visibles du texte peuvent être en décalage par rapport aux dimensions logiques. Ce décalage peut être calculé en spécifiant l'option #PB_VectorText_Offset.
L'exemple suivant montre un texte avec la zone de délimitation logique en bleu, la zone de délimitation visible en rouge et l'emplacement de la ligne de base en vert. Le début du texte est dessiné dans le coin supérieur gauche de la zone de délimitation logique (bleu).

Exemple

```
1  If OpenWindow(0, 0, 0, 500, 250, "VectorDrawing", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2   CanvasGadget(0, 0, 0, 500, 250)
3   LoadFont(0, "Monotype Corsiva", 20, #PB_Font_Italic)
4
5   If StartVectorDrawing(CanvasVectorOutput(0))
6
7     VectorFont(FontID(0), 125)
8     Texte$ = "Exemple"
```

```
9
10          ; dessin du texte
11          MovePathCursor(25, 25)
12          DrawVectorText(Texte$)
13
14          ; dessin de la boîte englobante logique (en bleu)
15          AddPathBox(25, 25, VectorTextWidth(Texte$), VectorTextHeight(Texte$))
16          VectorSourceColor(RGBA(0, 0, 255, 255))
17          DashPath(2, 10)
18
19          ; dessin de la boîte englobante visible (en rouge)
20          AddPathBox(25 + VectorTextWidth(Texte$, #PB_VectorText_Visible|#PB_VectorText_Offset),
21                     25 + VectorTextHeight(Texte$, #PB_VectorText_Visible|#PB_VectorText_Offset),
22                     VectorTextWidth(Texte$, #PB_VectorText_Visible),
23                     VectorTextHeight(Texte$, #PB_VectorText_Visible))
24          VectorSourceColor(RGBA(255, 0, 0, 255))
25          DashPath(2, 10)
26
27          ; dessin de la ligne de base (en vert)
28          MovePathCursor(25, 25 + VectorTextHeight(Texte$, #PB_VectorText_Baseline))
29          AddPathLine(VectorTextWidth(Texte$), 0, #PB_Path_Relative)
30          VectorSourceColor(RGBA(0, 255, 0, 255))
31          DashPath(2, 10)
32
33          StopVectorDrawing()
34       EndIf
35
36       Repeat
37          Event = WaitWindowEvent()
38       Until Event = #PB_Event_CloseWindow
39    EndIf
```

Voir aussi

VectorTextWidth() , DrawVectorText() , DrawVectorParagraph() , VectorParagraphHeight() , VectorFont()

OS Supportés

Tous

154.65 VectorParagraphHeight

Syntaxe

```
Resultat.d =
    VectorParagraphHeight(Texte$,
    Largeur.d, Hauteur.d)
```

Description

Renvoie la hauteur nécessaire pour dessiner le paragraphe de texte donné en utilisant la fonction DrawVectorParagraph() .

Arguments

Texte$ Le paragraphe de texte à mesurer (peut contenir plusieurs lignes).

Largeur.d La largeur à utiliser pour le paragraphe.

Hauteur.d La hauteur maximale disponible pour le paragraphe.

Valeur de retour

La hauteur réelle nécessaire pour dessiner le paragraphe.

Remarques

Si le texte ne tient pas dans la zone de délimitation définie, le résultat sera égal à la valeur de "Hauteur.d". Cela indique que l'appel à DrawVectorParagraph() couperait le texte.

Voir aussi

DrawVectorParagraph() , DrawVectorText() , VectorFont() , VectorTextWidth() , VectorTextHeight()

OS Supportés

Tous

154.66 PdfVectorOutput

Syntaxe

```
Resultat =
    PdfVectorOutput(Fichier$,
    Largeur.d, Hauteur.d [,
    Unite])
```

Description

Crée un fichier PDF (Linux et Mac OSX seulement).

Arguments

Fichier$ Le nom du fichier PDF à créer. Si le fichier existe, il sera écrasé.

Largeur.d, Hauteur.d Les dimensions de la page dans le fichier PDF en unité de mesure utilisée par la sortie de dessin vectoriel.

Unite (optionnel) Peut prendre l'une des valeurs suivantes :
L'unité par défaut pour les fichiers PDF est #PB_Unit_Point.

```
#PB_Unit_Pixel      : Les
 valeurs sont mesurées en
 pixels (ou point (dots)
 pour les imprimantes)
#PB_Unit_Point      : Les
 valeurs sont mesurées en
 points (1/72 pouce =
 25.4/72 mm = 0,352 778
 mm)
#PB_Unit_Inch       : Les
 valeurs sont mesurées en
 pouces (25,4 millimètres)
#PB_Unit_Millimeter: Les
 valeurs sont mesurées en
 millimètres (0,039 370
 pouce)
```

Valeur de retour

Renvoie le OutputID du fichier donné afin d'effectuer le rendu 2D à l'aide de StartVectorDrawing() .

Remarques

Les opérations de dessin doivent se faire à l'intérieur d'un bloc StartVectorDrawing() /StopVectorDrawing() .

Le fichier PDF peut contenir plusieurs pages en utilisant fonction NewVectorPage() .

Exemple

```
1  LoadFont(0, "Times New Roman", 20)
2
3  If StartVectorDrawing(PdfVectorOutput("test.pdf", 595, 842))
4     VectorFont(FontID(0), 25)
5
6     MovePathCursor(20, 20)
7     DrawVectorText("Voici la page 1...")
8
9     NewVectorPage()
10
11    MovePathCursor(20, 20)
12    DrawVectorText("Voici la page 2...")
13
14    StopVectorDrawing()
15 EndIf
```

Voir aussi

SvgVectorOutput() , ImageVectorOutput() , PrinterVectorOutput() , WindowVectorOutput() , CanvasVectorOutput()

OS Supportés

Linux, MacOS X

154.67 SvgVectorOutput

Syntaxe

```
Resultat = SvgVectorOutput(Fichier$, Largeur.d, Hauteur.d [, Unite])
```

Description

Crée un fichier SVG (scalable vector graphics) (Linux seulement).

Arguments

Fichier$ Le nom du fichier SVG à créer. Si le fichier existe, il sera écrasé.

Largeur.d, Hauteur.d Les dimensions de la page dans le fichier SVG en unité de mesure utilisée par la sortie de dessin vectoriel.

Unite (optionnel) Peut prendre l'une des valeurs suivantes :
L'unité par défaut pour les fichiers SVG est #PB_Unit_Point.

```
#PB_Unit_Pixel      : Les
 valeurs sont mesurées en
 pixels (ou point (dots)
 pour les imprimantes)
#PB_Unit_Point      : Les
 valeurs sont mesurées en
 points (1/72 pouce =
 25.4/72 mm = 0,352 778
 mm)
#PB_Unit_Inch       : Les
 valeurs sont mesurées en
 pouces (25,4 millimètres)
#PB_Unit_Millimeter: Les
 valeurs sont mesurées en
 millimètres (0,039 370
 pouce)
```

Valeur de retour

Renvoie le OutputID du fichier donné afin d'effectuer le rendu à l'aide de StartVectorDrawing() .

Remarques

Les opérations de dessin doivent se faire à l'intérieur d'un bloc StartVectorDrawing() /StopVectorDrawing() .
Le fichier SVG peut contenir plusieurs pages en utilisant fonction NewVectorPage() .
Plus d'informations sur les fichiers SVG ici.

Exemple

```
1  If StartVectorDrawing(SvgVectorOutput("test.svg", 400, 200))
2
3    AddPathBox(50, 50, 200, 50)
4    AddPathBox(150, 75, 200, 50)
5    VectorSourceColor(RGBA(255, 0, 0, 255))
6    StrokePath(10)
7
8    StopVectorDrawing()
9  EndIf
```

Voir aussi

PdfVectorOutput() , ImageVectorOutput() ,
PrinterVectorOutput() ,
WindowVectorOutput() ,
CanvasVectorOutput()

OS Supportés

Linux

Chapitre 155

Vehicle

Généralités

Les véhicules (vehicle) sont habituellement composés d'un seul corps et d'une ou de plusieurs roues (voitures, camions, vélos, etc.). Le nouveau véhicule créé est une simple entité , de sorte que toutes les fonctions concernant les entités peuvent être utilisées pour le manipuler.
InitEngine3D() doit être appelé avec succès avant d'utiliser les fonctions de cette bibliothèque.

OS Supportés

Tous

155.1 AddVehicleWheel

Syntaxe

```
AddVehicleWheel(#Entity,
    #EntityRoue,
    TeteSuspensionX.f,
    TeteSuspensionY.f,
    TeteSuspensionZ.f,
    MoyeuRoueX.f,
    MoyeuRoueY.f,
    MoyeuRoueZ.f,
    DebattementMaxSuspension.f,
    RayonRoue.f, TractionRoue,
    RoulisRoue.f)
```

Description

Ajoute une nouvelle roue à un véhicule précédemment créé avec CreateVehicle() .

Arguments

#**Entity** L'entité véhicule à utiliser.

#**EntityRoue** L'entité à utiliser pour la roue.

TeteSuspensionX.f, TeteSuspensionY.f, TeteSuspensionZ.f Coordonnées du point d'attache de la suspension sur le chassis.

MoyeuRoueX.f, MoyeuRoueY.f, MoyeuRoueZ.f Coordonnée de l'axe de roue (moyeu).

DebattementMaxSuspension.f Le débattement maximum de la suspension, en mètres.

RayonRoue.f Le rayon de la roue.

TractionRoue `#True : La roue est motrice, ApplyVehicleForce() et ApplyVehicleSteering() s'appliquent à cette roue. #False: La roue est libre, non motrice.`

RoulisRoue Réduit le roulis de la roue (0.0 : pas de roulis, 1.0 : comportement physique).
Si le frottement est trop élevé, il pourrait être nécessaire de réduire cette valeur pour arrêter le balancement du véhicule (Equivalent au réglage de la barre de torsion dans le monde réel).

Valeur de retour

Aucune.

Voir aussi

ApplyVehicleBrake() , ApplyVehicleForce() , CreateVehicle()

OS Supportés

Tous

155.2 ApplyVehicleForce

Syntaxe

```
ApplyVehicleForce(#Entity, Roue, Force.f)
```

Description

Applique une force de traction sur une roue du véhicule.

Arguments

#Entity L'entité véhicule à utiliser.

Roue L'indice de la roue, à partir de 0.

Force La force de traction à appliquer sur la roue.

Valeur de retour

Aucune.

Remarques

La nouvelle valeur de la force de traction remplace la force précédemment appliquée à la roue du véhicule.

Voir aussi

ApplyVehicleSteering() , ApplyVehicleBrake() , AddVehicleWheel()

OS Supportés

Tous

155.3 ApplyVehicleBrake

Syntaxe

```
ApplyVehicleBrake(#Entity,
    Roue, Frein.f)
```

Description

Applique une force de freinage sur une roue du véhicule.

Arguments

#Entity L'entité véhicule à utiliser.

Roue L'indice de la roue, à partir de 0.

Frein La force de freinage à appliquer sur la roue.

Valeur de retour

Aucune.

Remarques

La nouvelle valeur de la force de freinage remplace la force précédemment appliquée à la roue du véhicule.

Voir aussi

ApplyVehicleSteering() , ApplyVehicleForce() , AddVehicleWheel()

OS Supportés

Tous

155.4 ApplyVehicleSteering

Syntaxe

```
ApplyVehicleSteering(#Entity,
    Roue, Braquage.f)
```

Description

Applique une force directive sur une roue du véhicule.

Arguments

#**Entity** L'entité véhicule à utiliser.

Roue L'indice de la roue, à partir de 0.

Braquage La force directive à appliquer sur la roue.

Valeur de retour

Aucune.

Remarques

La nouvelle valeur de la force directive remplace la force précédemment appliquée à la roue du véhicule.

Voir aussi

ApplyVehicleBrake() , ApplyVehicleForce() , AddVehicleWheel()

OS Supportés

Tous

155.5 CreateVehicle

Syntaxe

```
Resultat =
    CreateVehicle(#Entity)
```

Description

Crée une nouvelle entité véhicule.

Arguments

#**Entity** Le numéro d'identification la nouvelle entité.
#PB_Any peut être utilisé pour générer automatiquement ce numéro.

Valeur de retour

Renvoie zéro si l'entité véhicule ne peut pas être créée.
Si #PB_Any est utilisé comme paramètre '#Entity', le nouveau numéro d'entité de véhicule est renvoyé.

Voir aussi

FreeEntity()

OS Supportés

Tous

155.6 CreateVehicleBody

Syntaxe

```
CreateVehicleBody(#Entity,
    Masse.f, Restitution.f,
    Friction.f [,
    SuspensionRigidité.f,
    SuspensionCompression.f,
    SuspensionAmortissement.f,
    SuspensionCompressionMax.f,
    AdherencePneu.f)
```

Description

Crée un corps physique associé à une entité véhicule.

Arguments

Masse Masse du véhicule.
Ne pas utiliser une trop grande valeur car il pourrait se produire des incohérences physiques (1 est la valeur préférée).

Restitution Restitution du véhicule.
Cette valeur peut également être obtenue ou configurer via GetEntityAttribute() et SetEntityAttribute()

Friction Friction du véhicule.
Cette valeur peut également être obtenue ou configurer via GetEntityAttribute() et SetEntityAttribute()

SuspensionRigidité (optionnel) La rigidité de la suspension (10 : Buggy tout terrain, 50 : Voiture de sports, 200 : Voiture F1)

SuspensionCompression (optionnel) Le coefficient d'amortissement à utiliser lorsque la suspension est comprimée. Mettre à valeur * 2 * RacineCarrée(SuspensionRigidité), elle

est donc proportionnelle à l'amortissement critique.
Exemples de valeur :

```
valeur = 0: non amortie et
 bondissant
valeur = 1: amortissement
 critique
```

Les valeurs recommandées vont de 0.1 à 0.3.

SuspensionAmortissement (optionnel)
L'amortissement lorsque la suspension est totallement étirée.
Voir SuspensionCompression pour définir cette valeur.
La valeur de SuspensionAmortissement doit être légèrement plus grande que SuspensionCompression.
Les valeurs recommandées vont de 0.2 à 0.5.

SuspensionCompressionMax (optionnel)
La longueur à laquelle la suspension peut être comprimée (en centimètres).

AdherencePneu (optionnel) Le frottement entre le pneumatique et le sol.
Devrait être d'environ 0,8 pour les voitures réalistes mais peut augmenter pour une meilleure manipulation.
Une grande valeur (10000.0) peut être utilisé pour les karts.

Valeur de retour

Aucune.

Remarques

Pour que les collisions soient gérées par le moteur physique, l'entité doit avoir un corps (body).
En fait, seul le corps est connu par le moteur physique qui fera tout le calcul sur l'entité, vérifier la masse, la friction et s'il entre en collision.
Pour avoir un effet, le moteur physique doit être activé avec EnableWorldPhysics() .

Voir aussi

FreeEntityBody()

OS Supportés

Tous

155.7 GetVehicleAttribute

Syntaxe

```
Resultat.f =
    GetVehicleAttribute(#Entity,
    Attribut, Roue)
```

Description

Obtenir l'attribut spécifié d'une entité véhicule.

Arguments

#**Entity** L'entité véhicule à utiliser.

Attribut L'attribut à obtenir. Les attributs suivants sont disponibles :

```
    #PB_Vehicle_Friction
                    : La
      valeur de frottement de
      la roue (voir
      CreateVehicleBody()
  pour plus d'info).
    #PB_Vehicle_MaxSuspensionForce
          : La valeur de la
      force maximale de la
      suspension de roue (voir
      CreateVehicleBody()
  pour plus d'info).
    #PB_Vehicle_SuspensionStiffness
         : La valeur de
      rigidité de la
      suspension (voir
      CreateVehicleBody()
  pour plus d'info).
    #PB_Vehicle_MaxSuspensionCompression:
      La valeur de compression
      maximale de la
      suspension de roue (voir
      CreateVehicleBody()
  pour plus d'info).
    #PB_Vehicle_MaxSuspensionLength
         : La longueur
      maximale de la
      suspension (mètres).
    #PB_Vehicle_WheelDampingCompression
      : La valeur de
      compression
      d'amortissement de la
      roue.
    #PB_Vehicle_WheelDampingRelaxation
       : La valeur de détente
      d'amortissement de la
      roue.
    #PB_Vehicle_RollInfluence
                : La valeur de
      l'influence du roulis de
```

```
    la roue (voir
    AddVehicleWheel()
  pour plus d'info).
   #PB_Vehicle_IsInContact
                : Renvoie
    #True si le véhicule est
    en contact avec un autre
    objet, #False sinon.
   #PB_Vehicle_CurrentSpeed
               : La vitesse
    actuelle du véhicule en
    km/heure.
   #PB_Vehicle_ContactPointX
              : La
    coordonnée X du point de
    contact.
   #PB_Vehicle_ContactPointY
              : La
    coordonnée Y du point de
    contact.
   #PB_Vehicle_ContactPointZ
              : La
    coordonnée Z du point de
    contact.
   #PB_Vehicle_ContactPointNormalX
         : La valeur normale
    X du point de contact.
   #PB_Vehicle_ContactPointNormalY
         : La valeur normale
    Y du point de contact.
   #PB_Vehicle_ContactPointNormalZ
         : La valeur normale
    Z du point de contact.
   #PB_Vehicle_ForwardVectorX
             : La valeur X
    du vecteur du point de
    contact.
   #PB_Vehicle_ForwardVectorY
             : La valeur Y
    du vecteur du point de
    contact.
   #PB_Vehicle_ForwardVectorZ
             : La valeur Z
    du vecteur du point de
    contact.
```

Roue L'indice de la roue, à partir de 0. Si non spécifié, ou mis à #PB_All, la nouvelle valeur de l'attribut est appliquée à toutes les roues.

Valeur de retour

Renvoie la valeur de l'attribut spécifié ou 0 si le véhicule ne supporte pas l'attribut.

Voir aussi

SetVehicleAttribute()

OS Supportés

Tous

155.8 SetVehicleAttribute

Syntaxe

```
SetVehicleAttribute(#Entity,
    Attribut, Valeur.f [,
    Roue])
```

Description

Définir la valeur de l'attribut spécifié d'une entité véhicule.

Arguments

#**Entity** L'entité véhicule à utiliser.

Attribut L'attribut à définir. Les attributs suivants sont disponibles :

```
    #PB_Vehicle_Friction
                    : La
      valeur de frottement de
      la roue (voir
      CreateVehicleBody()
  pour plus d'info).
    #PB_Vehicle_MaxSuspensionForce
          : La valeur de la
      force maximale de la
      suspension de roue (voir
      CreateVehicleBody()
  pour plus d'info).
    #PB_Vehicle_SuspensionStiffness
         : La valeur de
      rigidité de la
      suspension (voir
      CreateVehicleBody()
  pour plus d'info).
    #PB_Vehicle_MaxSuspensionCompression:
      La valeur de compression
      maximale de la
      suspension de roue (voir
      CreateVehicleBody()
  pour plus d'info).
    #PB_Vehicle_MaxSuspensionLength
         : La longueur
      maximale de la
      suspension (mètres).
    #PB_Vehicle_WheelDampingCompression
      : La valeur de
      compression
      d'amortissement de la
      roue.
    #PB_Vehicle_WheelDampingRelaxation
       : La valeur de détente
      d'amortissement de la
      roue .
```

```
    #PB_Vehicle_RollInfluence
                : La valeur de
      l'influence du roulis de
      la roue (voir
      AddVehicleWheel()
    pour plus d'info).
```

Valeur.f Valeur de l'attribut.

Roue (optionnel) L'indice de la roue, à partir de 0.
Si non spécifié, ou mis à #PB_All, la nouvelle valeur de l'attribut est appliquée à toutes les roues.

Valeur de retour

Aucune.

Voir aussi

GetVehicleAttribute()

OS Supportés

Tous

Chapitre 156

VertexAnimation

Généralités

Les animations vertex permettent de gérer les ”animations-poses” d'un mesh .
Le mesh doit contenir une liste prédéfinie de poses.

OS Supportés

Tous

156.1 CreateVertexAnimation

Syntaxe

```
Resultat =
    CreateVertexAnimation(#Mesh,
    Animation$, Durée)
```

Description

Crée une nouvelle animation.

Arguments

#Mesh Le mesh à utiliser.

Animation$ Le nom de la nouvelle animation .
Attention ce nom est sensible à la casse.

Durée La durée de la nouvelle animation, en millisecondes.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Voir aussi

CreateVertexTrack()

OS Supportés

Tous

156.2 CreateVertexTrack

Syntaxe

```
Resultat =
    CreateVertexTrack(#Mesh,
    Animation$, Index)
```

Description

Crée une nouvelle piste d'animation.

Arguments

#Mesh Le mesh à utiliser.

Animation$ Le nom de l'animation.
Attention ce nom est sensible à la casse.

Index Le nouvel index de la piste.
Index doit commencer à partir de zéro.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro sinon.

Remarques

L'animation doit déjà être créée avec CreateVertexAnimation() ou être prédéfinie dans le mesh. Toutes les pistes ont la même longueur, tel que défini dans CreateVertexAnimation() .

Voir aussi

CreateVertexAnimation()

OS Supportés

Tous

156.3 CreateVertexPoseKeyFrame

Syntaxe

```
Resultat =
    CreateVertexPoseKeyFrame(#Mesh,
    Animation$, Piste, Temps)
```

Description

Crée une nouvelle image-clé dans une animation.

Arguments

#**Mesh** Le mesh à utiliser.

Animation$ Le nom de l'animation.
Attention ce nom est sensible à la casse.

Piste L'index de piste.
La piste doit être préalablement créée avec CreateVertexTrack() .

Temps Le temps dans l'animation de l'image clé, en millisecondes.
Cette valeur doit être comprise entre zéro et la 'longueur' définie dans CreateVertexAnimation() .

Valeur de retour

Renvoie l'index de la nouvelle image-clé.

Remarques

L'animation doit déjà être créée avec CreateVertexAnimation() ou être prédéfinie dans le mesh.

Voir aussi

CreateVertexAnimation() ,
CreateVertexTrack()

OS Supportés

Tous

156.4 AddVertexPoseReference

Syntaxe

```
AddVertexPoseReference(#Mesh,
    Animation$, Piste,
    ImageClé, IndexPose,
    Influence.f)
```

Description

Ajoute une nouvelle pose de référence pour une animation.

Arguments

#**Mesh** Le mesh à utiliser.

Animation$ Le nom de l'animation.
Attention ce nom est sensible à la casse.

Piste L'index de la piste.
La piste doit être préalablement créée avec CreateVertexTrack() .

ImageClé L'index de l'image-clé.
L'image-clé doit être préalablement créée avec CreateVertexPoseKeyFrame() .

IndexPose L'index de pose.
La première pose a un index qui commence à zéro.

Influence.f L'influence de la pose.
Cette valeur varie de 0.0 (pas d'influence) à 1.0 (influence totale).

Valeur de retour

Aucune.

Remarques

L'animation doit déjà être créée avec CreateVertexAnimation() ou être prédéfinie dans le mesh.

Voir aussi

CreateVertexAnimation() , CreateVertexTrack()

OS Supportés

Tous

156.5 UpdateVertexPoseReference

Syntaxe

```
UpdateVertexPoseReference(#Mesh,
    Animation$, Piste,
    ImageClé, IndexPose,
    Influence.f)
```

Description

Mise à jour d'une nouvelle pose de référence pour une animation.

Arguments

#Mesh Le mesh à utiliser.

Animation$ Le nom de l'animation.
Attention ce nom est sensible à la casse.

Piste L'index de piste.
La piste doit être préalablement créée avec CreateVertexTrack() .

ImageClé L'index de l'image-clé.
L'image-clé doit être préalablement créée avec CreateVertexPoseKeyFrame() .

IndexPose L'index de pose.
La première pose a un index qui commence à zéro.

Influence.f L'influence de la pose.
Cette valeur varie de 0.0 (pas d'influence) à 1.0 (influence totale).

Valeur de retour

Aucune.

Remarques

L'animation doit déjà être créée avec CreateVertexAnimation() ou être prédéfinie dans le mesh.

Voir aussi

CreateVertexAnimation() , CreateVertexTrack()

OS Supportés

Tous

156.6 VertexPoseReferenceCount

Syntaxe

```
Resultat =
    VertexPoseReferenceCount(#Mesh,
    Animation$, Piste,
    ImageClé)
```

Description

Renvoie le nombre de poses de référence dans l'image-clé spécifiée.

Arguments

#**Mesh** Le mesh à utiliser.

Animation$ Le nom de l'animation.
Attention ce nom est sensible à la casse.

Piste L'index de piste.
La piste doit être préalablement créée avec CreateVertexTrack() .

ImageClé L'index de l'image-clé.
L'image-clé doit être préalablement créée avec CreateVertexPoseKeyFrame() .

Valeur de retour

Renvoie le nombre de poses de référence dans l'image-clé spécifiée.

Remarques

L'animation doit déjà être créée avec CreateVertexAnimation() ou être prédéfinie dans le mesh.

Voir aussi

CreateVertexAnimation() ,
CreateVertexTrack() ,
AddVertexPoseReference()

OS Supportés

Tous

156.7 MeshPoseCount

Syntaxe

```
Resultat =
    MeshPoseCount(#Mesh)
```

Description

Renvoie le nombre de poses dans un mesh.

Arguments

#**Mesh** Le mesh à utiliser.

Valeur de retour

Renvoie le nombre de poses dans le mesh spécifié.

Remarques

Une pose est une animation vertex prédéfinie dans le mesh.

Voir aussi

MeshPoseName() ,
AddVertexPoseReference()

OS Supportés

Tous

156.8 MeshPoseName

Syntaxe

```
Resultat\$ =
    MeshPoseName(#Mesh,
    IndexPose)
```

Description

Renvoie le nom d'une pose dans un mesh.

Arguments

#**Mesh** Le mesh à utiliser.

IndexPose L'index de la pose. La première pose commence à zéro. L'index doit être inférieur au résultat de MeshPoseCount() .

Valeur de retour

Renvoie le nom de la pose à l'index spécifié.

Remarques

Une pose est une animation vertex prédéfinie dans le mesh.

Voir aussi

MeshPoseName() ,
AddVertexPoseReference()

OS Supportés

Tous

Chapitre 157

Window

Généralités

Cette bibliothèque vous permet de gérer et manipuler les composants essentiels et indispensables à une interface graphique moderne : Les fenêtres.

OS Supportés

Tous

157.1 AddKeyboardShortcut

Syntaxe

```
AddKeyboardShortcut(#Fenetre,
    Raccourci, Evenement)
```

Description

Ajoute ou remplace un raccourci clavier.

Arguments

#**Fenetre** La fenêtre à utiliser.

Raccourci Le raccourci clavier peut prendre l'une des valeurs suivantes :

```
#PB_Shortcut_Back
 (Retour Arrière)
#PB_Shortcut_Tab
 (Tabulation)
#PB_Shortcut_Clear
#PB_Shortcut_Return
 (Entrée)
#PB_Shortcut_Menu
 (Alt)
#PB_Shortcut_Pause
#PB_Shortcut_Print
#PB_Shortcut_Capital
 (Verrouillage Majuscule)
#PB_Shortcut_Escape
 (Echap)
```

#PB_Shortcut_Space
 (Espace)
#PB_Shortcut_PageUp
 (Page précédente)
#PB_Shortcut_PageDown
 (Page Suivante)
#PB_Shortcut_End
 (Fin)
#PB_Shortcut_Home
 (Début)
#PB_Shortcut_Left
 (Flèche à gauche)
#PB_Shortcut_Up
 (Flèche en haut)
#PB_Shortcut_Right
 (Flèche à droite)
#PB_Shortcut_Down
 (Flèche en bas)
#PB_Shortcut_Select
#PB_Shortcut_Execute
#PB_Shortcut_Snapshot
 (Impr écran)
#PB_Shortcut_Insert
#PB_Shortcut_Delete
 (Suppr)
#PB_Shortcut_Help
#PB_Shortcut_0
#PB_Shortcut_1
#PB_Shortcut_2
#PB_Shortcut_3
#PB_Shortcut_4
#PB_Shortcut_5
#PB_Shortcut_6
#PB_Shortcut_7
#PB_Shortcut_8
#PB_Shortcut_9
#PB_Shortcut_A
#PB_Shortcut_B
#PB_Shortcut_C
#PB_Shortcut_D
#PB_Shortcut_E
#PB_Shortcut_F
#PB_Shortcut_G
#PB_Shortcut_H
#PB_Shortcut_I
#PB_Shortcut_J
#PB_Shortcut_K
#PB_Shortcut_L
#PB_Shortcut_M
#PB_Shortcut_N
#PB_Shortcut_O
#PB_Shortcut_P
#PB_Shortcut_Q
#PB_Shortcut_R
#PB_Shortcut_S
#PB_Shortcut_T
#PB_Shortcut_U
#PB_Shortcut_V
#PB_Shortcut_W

```
#PB_Shortcut_X
#PB_Shortcut_Y
#PB_Shortcut_Z
#PB_Shortcut_LeftWindows
 (Touche fenêtre main
 gauche)
#PB_Shortcut_RightWindows
 (Touche fenêtre main
 droite)
#PB_Shortcut_Apps
#PB_Shortcut_Pad0
#PB_Shortcut_Pad1
#PB_Shortcut_Pad2
#PB_Shortcut_Pad3
#PB_Shortcut_Pad4
#PB_Shortcut_Pad5
#PB_Shortcut_Pad6
#PB_Shortcut_Pad7
#PB_Shortcut_Pad8
#PB_Shortcut_Pad9
#PB_Shortcut_Multiply
 (*)
#PB_Shortcut_Add
 (+)
#PB_Shortcut_Separator
#PB_Shortcut_Subtract
 (-)
#PB_Shortcut_Decimal
 (.)
#PB_Shortcut_Divide
 (/)
#PB_Shortcut_F1
#PB_Shortcut_F2
#PB_Shortcut_F3
#PB_Shortcut_F4
#PB_Shortcut_F5
#PB_Shortcut_F6
#PB_Shortcut_F7
#PB_Shortcut_F8
#PB_Shortcut_F9
#PB_Shortcut_F10
#PB_Shortcut_F11
#PB_Shortcut_F12
#PB_Shortcut_F13
#PB_Shortcut_F14
#PB_Shortcut_F15
#PB_Shortcut_F16
#PB_Shortcut_F17
#PB_Shortcut_F18
#PB_Shortcut_F19
#PB_Shortcut_F20
#PB_Shortcut_F21
#PB_Shortcut_F22
#PB_Shortcut_F23
#PB_Shortcut_F24
#PB_Shortcut_Numlock
#PB_Shortcut_Scroll
 (Arrêt défil)
```

Les touches de raccourci ci-dessus

peuvent être combinées avec les valeurs ci-dessous si nécessaire :

```
#PB_Shortcut_Shift   :
 'Shift' ou 'Majuscule'
#PB_Shortcut_Control:
 'Control' ou 'CTRL'
#PB_Shortcut_Alt     : 'Alt'
#PB_Shortcut_Command:
 'Apple' ou 'Pomme'
 (MacOS X)
```

Evenement Un nombre unique défini par l'utilisateur qui permet d'identifier le raccourci clavier.
Ce nombre doit être compris entre 0 et 64000 et il correspond à l'évènement de type menu qui lui est associé.
Par défaut, une fenêtre a déjà 2 raccourcis clavier pour permettre les déplacements entre les gadgets avec la touche 'Tab' (Tabulation) et 'Shift+Tab' (Majuscule + Tabulation) : #PB_Shorcut_Tab et #PB_Shortcut_Tab‖#PB_Shortcut_Shift

Remarques

Un raccourci génère un évènement de type menu (comme le ferait un élément de menu) car la plupart du temps les raccourcis sont utilisés en conjonction avec les menus.
La constante #PB_Shortcut_Command est seulement utile sous MacOS X et permet d'utiliser la touche 'Apple' (droite ou gauche). Cette constante est aussi supportée sur les autres OS (pour faciliter la portabilité) et elle aura alors un rôle identique à #PB_Shortcut_Control.
Un raccourci clavier peut être enlevé avec RemoveKeyboardShortcut() .

Exemple

```
1  AddKeyboardShortcut(0,
    #PB_Shortcut_Control |
    #PB_ShortCut_F, 15) ; Crée
    un raccourci clavier
    CTRL+F sur la fenêtre 0
2
    qui générera un évènement
    de valeur 15.
```

Voir aussi

RemoveKeyboardShortcut()

OS Supportés

Tous

157.2 AddWindowTimer

Syntaxe

```
AddWindowTimer(#Fenetre,
    Minuteur, Temps)
```

Description

Ajoute un nouveau minuteur à une fenêtre. Un minuteur a pour fonction de générer un évènement de type #PB_Event_Timer tous les 'Temps' millisecondes.

Arguments

#Fenetre La fenêtre à utiliser.

Minuteur Le numéro d'identification du minuteur.
Ce numéro sera renvoyé par EventTimer() lors d'un évènement de type #PB_Event_Timer.

Temps Temps écoulé, en millisecondes, entre deux évènements #PB_Event_Timer events.

Valeur de retour

Aucune.

Remarques

Un minuteur est toujours lié à une fenêtre et sera supprimé avec la fermeture de la fenêtre.
Plusieurs minuteurs peuvent fonctionner en même temps dans une fenêtre.
De plus, des minuteurs actifs sur différentes fenêtres peuvent avoir le même numéro.
Les évènements 'minuteur' ne seront générés que si aucun autre évènement ne doit être traité (les minuteurs ont une priorité basse). Cela implique qu'ils sont relativement peu précis, et que la durée entre deux évènements du même minuteur peut varier. Ils n'ont pas pour vocation à être utilisés pour de la précision, mais plutôt pour effectuer des tâches périodiques comme par exemple la mise à jour d'un gadget.
Pour retirer un minuteur, il faut utiliser RemoveWindowTimer() .
Pour modifier la durée de la minuterie, il faut d'abord supprimer le minuteur, puis ajouter ce même minuteur avec une autre valeur de temps :

```
1    RemoveWindowTimer(#Fenetre,
     Timer0)
2    AddWindowTimer(#Fenetre,
     Timer0, NouvelleValeur)
```

Exemple

```
If OpenWindow(0, 0, 0, 400, 100, "Exemple Minuteur", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
  ProgressBarGadget(0, 10, 10, 380, 20, 0, 100)

  ; Ajout du minuteur n°123 dans la fenêtre 0 qui se déclenchera toutes les 250 ms
  AddWindowTimer(0, 123, 250)

  Value = 0
  Repeat
    Event = WaitWindowEvent()

    If Event = #PB_Event_Timer And EventTimer() = 123
      Value = (Value + 5) % 100
      SetGadgetState(0, Value)
    EndIf

  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

RemoveWindowTimer() , EventTimer()

OS Supportés

Tous

157.3 RemoveWindowTimer

Syntaxe

```
RemoveWindowTimer(#Fenetre, Minuteur)
```

Description

Retire un minuteur.

Arguments

#Fenetre La fenêtre à utiliser.

Minuteur Le numéro d'identification du minuteur à retirer.

Valeur de retour

Aucune.

Remarques

Le paramêtre 'Minuteur' doit avoir la même valeur que celle spécifiée lors de sa création avec AddWindowTimer() . Il n'y aura plus d'évènements générés par ce minuteur.

Voir aussi

AddWindowTimer()

OS Supportés

Tous

157.4 EventTimer

Syntaxe

```
Resultat = EventTimer()
```

Description

Renvoie le numéro du minuteur qui a envoyé un évènement #PB_Event_Timer.

Arguments

Aucun.

Valeur de retour

La valeur renvoyée est la même que celle spécifiée lors de la création du minuteur avec AddWindowTimer() .

Remarques

L'évènement de type #PB_Event_Timer est renvoyé par WindowEvent() ou WaitWindowEvent() .

Voir aussi

AddWindowTimer()

OS Supportés

Tous

157.5 CloseWindow

Syntaxe

```
CloseWindow(#Fenetre)
```

Description

Ferme une fenêtre.

Arguments

#Fenetre La fenêtre à fermer.
Si #PB_All est spécifié, toutes les autres fenêtres sont fermées.

Valeur de retour

Aucune.

Remarques

Toutes les fenêtres restant ouvertes sont automatiquement fermées quand le programme se termine.
Note : Les éléments d'une fenêtre sont détruits et leur mémoires automatiquement libérées lorsque la fenêtre est fermée. Pour information, cela concerne les gadgets , les raccourcis clavier , les menus , les statusBars , les timers , les tolbars et les évènements liés (avec BindEvent()).

Exemple

```
1   If OpenWindow(0, 0, 0, 220, 100, "Exemple...", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2
3     ButtonGadget  (1, 10, 60, 200, 30, "Fermer")
4
5     Repeat
6       Event = WaitWindowEvent()
7
8       Select Event
9
10        Case #PB_Event_Gadget
11          Select EventGadget()
12            Case 1
13               CloseWindow(0)
14               End
15          EndSelect
16
17      EndSelect
18    Until Event = #PB_Event_CloseWindow
19  EndIf
```

Voir aussi

OpenWindow()

OS Supportés

Tous

157.6 DisableWindow

Syntaxe

```
DisableWindow(#Fenetre, Etat)
```

Description

Active ou désactive une fenêtre.

Arguments

#**Fenetre** La fenêtre à activer/désactiver.

Etat Peut prendre une des valeurs suivantes :

```
#True : La fenêtre est
 désactivée.
#False: La fenêtre est
 activée.
```

Valeur de retour

Aucune.

Exemple

```
If OpenWindow(0, 0, 0, 220,
 100, "Exemple...",
 #PB_Window_SystemMenu |
 #PB_Window_ScreenCentered)

 ButtonGadget  (1, 10, 60,
 200, 30, "Désactiver la
 fenêtre pendant 5s")

 Repeat
   Event = WaitWindowEvent()

   Select Event

     Case #PB_Event_Gadget
       Select EventGadget()
         Case 1

           DisableWindow(0,1)
 ; Désactive la fenêtre
           Delay(5000)
        ; Attendre 5
 secondes
```

```
16              DisableWindow(0,0)
    ; Active la fenêtre
17
18          EndSelect
19
20      EndSelect
21    Until Event =
    #PB_Event_CloseWindow
22  EndIf
```

OS Supportés

Tous

157.7 Event

Syntaxe

```
Resultat = Event()
```

Description

Renvoie l'évènement en cours.

Arguments

Aucun.

Valeur de retour

Renvoie l'évènement en cours.

Remarques

Il s'agit de la même valeur renvoyée par WindowEvent() et WaitWindowEvent() . Est utile principalement lors de l'utilisation d'un callback pour déterminer l'évènement qui l'a déclenché.

Exemple

```
1   Procedure EventHandler()
2     Select Event()
3       Case
    #PB_Event_CloseWindow
4         End
5
6       Case #PB_Event_Gadget
7         Debug "Clic sur
    Gadget #" + EventGadget()
8     EndSelect
9   EndProcedure
10
```

```
11  OpenWindow(0, 100, 100,
     240, 100, "",
     #PB_Window_SizeGadget |
     #PB_Window_SystemMenu |
     #PB_Window_MaximizeGadget)
12  ButtonGadget(0,  10, 10,
     100, 30, "Cliquez moi !")
13  ButtonGadget(1, 130, 10,
     100, 30, "Cliquez moi !")
14
15  ; Utiliser un seul callback
     pour tous les évènements
16  BindGadgetEvent(0,
     @EventHandler())
17  BindGadgetEvent(1,
     @EventHandler())
18  BindEvent(#PB_Event_CloseWindow,
     @EventHandler())
19
20  ; Nous ne traitons pas les
     évènements ici, donc nous
     pouvons exécuter une
     boucle infinie et l'oublier
21  Repeat
22    WaitWindowEvent()
23  ForEver
```

Voir aussi

WindowEvent() , WaitWindowEvent()

OS Supportés

Tous

157.8 EventGadget

Syntaxe

```
Resultat = EventGadget()
```

Description

Renvoie le numéro du gadget qui a envoyé un évènement.
Permet de connaître quel gadget a été utilisé ou cliqué par l'utilisateur.

Arguments

Aucun.

Valeur de retour

Renvoie le numéro du gadget associé à l'évènement.

Remarques

Utilisez cette fonction après un évènement de type #PB_Event_Gadget renvoyé par WindowEvent() ou WaitWindowEvent() .

Exemple

```
If OpenWindow(0, 0, 0, 230, 90, "Exemple de gestion des évènements...", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)

  ButtonGadget  (1, 10, 10, 200, 20, "Cliquez moi")
  CheckBoxGadget(2, 10, 40, 200, 20, "Cochez  moi")

  Repeat
    Event = WaitWindowEvent()
    Select Event

      Case #PB_Event_Gadget
        Select EventGadget()
          Case 1 : Debug "Bouton cliqué !"
          Case 2 : Debug "Case à cocher cliquée !"
        EndSelect

    EndSelect
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

WindowEvent() , WaitWindowEvent()

OS Supportés

Tous

157.9 EventMenu

Syntaxe

```
Resultat = EventMenu()
```

Description

Renvoie le numéro du menu qui a envoyé un évènement.
C'est le même évènement qui permet de connaître aussi l'icône d'une barre d'outils ou le raccourci clavier qui a été utilisé.

Arguments

Aucun.

Valeur de retour

Renvoie le numéro du menu associé à l'évènement.

Remarques

Utilisez cette fonction après un évènement de type #PB_Event_Menu renvoyé par WindowEvent() ou WaitWindowEvent() . Cette commande permet aussi de détecter quel bouton d'une barre d'outil ou quel raccourci clavier a été utilisé.
Un évènement sur une barre d'outils est similaire à un évènement sur un menu (car les barres d'outils sont la plupart du temps des racourcis pour des opérations disponibles dans les menus). Il est donc préférable de prendre les même identifiants pour l'élément du menu et le bouton de la barre d'outils qui représentent la même action, comme ça aucun code supplémentaire n'est nécessaire pour gérer la barre d'outils.

Exemple

```
1   If OpenWindow(0, 0, 0, 230, 90, "Exemple de gestion des évènements...", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2
3   ;Raccourcis clavier
4   AddKeyboardShortcut(0, #PB_Shortcut_Control | #PB_Shortcut_N, 1)
5   AddKeyboardShortcut(0, #PB_Shortcut_Control | #PB_Shortcut_O, 2)
6   AddKeyboardShortcut(0, #PB_Shortcut_Control | #PB_Shortcut_S, 3)
7
8   ;Menu
9   If CreateMenu(0, WindowID(0))
10    MenuTitle("Menu")
11    MenuItem(1, "Nouveau"+Chr(9)+"Ctrl+N")
12    MenuItem(2, "Ouvrir"+Chr(9)+"Ctrl+O")
13    MenuItem(3, "Enregistrer"+Chr(9)+"Ctrl+S")
14  EndIf
```

```
15
16    ;Barre d'outils
17    If CreateToolBar(0,
       WindowID(0))
18      ToolBarStandardButton(1,
       #PB_ToolBarIcon_New)
19      ToolBarStandardButton(2,
       #PB_ToolBarIcon_Open)
20      ToolBarStandardButton(3,
       #PB_ToolBarIcon_Save)
21    EndIf
22
23    Repeat
24      Event = WaitWindowEvent()
25
26      Select Event
27
28        Case #PB_Event_Menu
29          Select EventMenu()
30            Case 1 : Debug
       "Menu 'Nouveau' cliqué ou
       Bouton 'Nouveau' cliqué ou
       'CTRL+N' au clavier !"
31            Case 2 : Debug
       "Menu 'Ouvrir' cliqué ou
       Bouton 'Ouvrir' cliqué ou
       'CTRL+O' au clavier !"
32            Case 3 : Debug
       "Menu 'Enregistrer' cliqué
       ou Bouton 'Enregistrer'
       cliqué ou 'CTRL+S' au
       clavier !"
33          EndSelect
34
35      EndSelect
36    Until Event =
       #PB_Event_CloseWindow
37  EndIf
```

Voir aussi

WindowEvent() , WaitWindowEvent()

OS Supportés

Tous

157.10 EventData

Syntaxe

```
Resultat = EventData()
```

Description

Renvoie la valeur associée à l'évènement en cours.

Arguments

Aucun.

Valeur de retour

Renvoie la donnée associée à l'évènement en cours.
Si l'évènement courant n'est pas un évènement personnalisé envoyé par PostEvent() , alors cette valeur est indéfinie.

Remarques

L'évènement doit être un évènement personnalisé envoyé avec PostEvent() .

Exemple

```
Enumeration
  #PB_Event_FirstCustomValue
  #EvenementDebutAction
  #EvenementFinAction
EndEnumeration


Procedure Thread(Valeur)
  PostEvent(#EvenementDebutAction, 0, 1, #PB_EventType_FirstCustomValue, 10)

  Delay(3000)
  PostEvent(#EvenementFinAction, 0, 1, #PB_EventType_FirstCustomValue, 100)
EndProcedure

OpenWindow(0, 200, 200, 100, 100, "PostEvent")

CreateThread(@Thread(), 0)

Repeat
  Event = WaitWindowEvent()

  Select Event
    Case #EvenementDebutAction
      Debug "Le Thread commence une action... "
      Debug EventData()

    Case #EvenementFinAction
      Debug "Le Thread a terminé une action"
      Debug EventData()

```

```
30      EndSelect
31
32    Until Event =
       #PB_Event_CloseWindow
```

Voir aussi

PostEvent() , WindowEvent()

OS Supportés

Tous

157.11 EventType

Syntaxe

```
Resultat = EventType()
```

Description

Renvoie le type d'évènement renvoyé par WindowEvent() ou WaitWindowEvent() .

Arguments

Aucun.

Valeur de retour

Les valeurs suivantes sont possibles lorsqu'un évènement du type #PB_Event_Gadget (bibliothèque Gadget) ou #PB_Event_SysTray (bibliothèque SysTray) survient :

```
#PB_EventType_LeftClick
        : Clic avec le
 bouton gauche de la souris
#PB_EventType_RightClick
       : Clic avec le bouton
 droit de la souris
#PB_EventType_LeftDoubleClick
 : Double-clic avec le
 bouton gauche de la souris
#PB_EventType_RightDoubleClick:
 Double-clic avec le bouton
 droit de la souris
#PB_EventType_Focus
            : Obtention du
 focus.
#PB_EventType_LostFocus
        : Perte du focus.
#PB_EventType_Change
           : Le contenu a
 changé.
```

```
  #PB_EventType_DragStart
         : L'utilisateur a
    essayé de lancer une
    opération de Glisser &
    Déposer
.
```

Remarques

Ne peut être utilisé qu'avec les gadgets suivants :

```
  - CanvasGadget()
 (Le CanvasGadget a un
    ensemble spécial de types
    d'évènement.)
  - ComboBoxGadget()

  - DateGadget()

  - EditorGadget()

  - ExplorerListGadget()

  - ExplorerTreeGadget()

  - ImageGadget()

  - ListViewGadget()

  - ListIconGadget()

  - MDIGadget()

  - OpenGLGadget()

  - SpinGadget()

  - StringGadget()

  - WebGadget()
 (Le WebGadget a un ensemble
    spécial de types
    d'évènement.)
\\

  (Voir la définition du
    gadget pour connaître les
    évènements valides)
```

Exemple

```
1  If OpenWindow(0, 0, 0, 230,
     120, "Exemple EventType()
     ...",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2
```

```
3     ListIconGadget(1, 10, 10,
    150, 100,
    "ListIconGadget", 140,
    #PB_ListIcon_GridLines)
4       For a= 1 To 4
5         AddGadgetItem(1, -1,
    "Ligne "+Str(a))
6       Next a
7
8       Repeat
9         Event =
    WaitWindowEvent()
10
11        Select Event
12
13          Case #PB_Event_Gadget
14            Select
    EventGadget()
15              Case 1
16                Select
    EventType()
17                  Case
    #PB_EventType_LeftClick
              : Debug "Clic avec
    le bouton gauche de la
    souris"
18                  Case
    #PB_EventType_RightClick
            : Debug "Clic avec
    le bouton droit de la
    souris"
19                  Case
    #PB_EventType_LeftDoubleClick
     : Debug "Double-clic avec
    le bouton gauche de la
    souris"
20                  Case
    #PB_EventType_RightDoubleClick
    : Debug "Double-clic avec
    le bouton droit de la
    souris"
21                EndSelect
22            EndSelect
23
24        EndSelect
25      Until Event =
    #PB_Event_CloseWindow
26    EndIf
```

Voir aussi

WaitWindowEvent() , WindowEvent()

OS Supportés

Tous

157.12 EventWindow

Syntaxe

```
Resultat = EventWindow()
```

Description

Renvoie le numéro de la fenêtre dans laquelle s'est produit le dernier évènement renvoyé par WindowEvent() ou WaitWindowEvent() .

Arguments

Aucun.

Valeur de retour

Le numéro de la fenêtre dans laquelle s'est produit l'évènement.

Exemple

```
If OpenWindow(0, 0, 0, 220, 100, "Exemple...", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)

  ButtonGadget  (1, 10, 60, 200, 30, "Cliquer")

  Repeat
    Event = WaitWindowEvent()
    NumeroFenetre = EventWindow()

    Select Event

      Case #PB_Event_Gadget
        Select EventGadget()
          Case 1
            Debug "Ce bouton appartient à la fenêtre numéro " + Str(NumeroFenetre)
        EndSelect

    EndSelect
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

WaitWindowEvent() , WindowEvent()

OS Supportés

Tous

157.13 GetActiveWindow

Syntaxe

```
Resultat = GetActiveWindow()
```

Description

Renvoie le numéro de la fenêtre qui est active.

Arguments

Aucun.

Valeur de retour

Renvoie le numéro de la fenêtre active, qui a donc le focus clavier ou -1 si aucune fenêtre n'est active.

Remarques

La fonction renvoie l'ID de la fenêtre dans le programme en cours seulement.
Une fenêtre dans le programme en cours peut être activée (détient le focus) avec la fonction SetActiveWindow() .

Exemple

```
1   If OpenWindow(0, 0, 0, 220, 100, "Exemple...", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
2
3   ButtonGadget  (1, 10, 60, 200, 30, "Cliquer")
4
5   Repeat
6     Event = WaitWindowEvent()
7
8     Select Event
9
10      Case #PB_Event_Gadget
11        Select EventGadget()
12          Case 1
13            Fenetre = GetActiveWindow()
14            Debug "La fenêtre active est la fenêtre numéro " + Str(Fenetre)
15        EndSelect
```

```
16
17      EndSelect
18    Until Event =
      #PB_Event_CloseWindow
19  EndIf
```

Voir aussi

SetActiveWindow()

OS Supportés

Tous

157.14 GetWindowColor

Syntaxe

```
Resultat =
    GetWindowColor(#Fenetre)
```

Description

Renvoie la couleur de fond d'une fenêtre.

Arguments

#**Fenetre** La fenêtre à tester.

Valeur de retour

Renvoie la couleur de fond d'une fenêtre qui a été préalablement spécifiée par la commande SetWindowColor() .
Si aucune couleur de fond n'a été spécifiée, la valeur -1 est renvoyée.

Exemple

```
1   If OpenWindow(0, 0, 0, 220,
      100, "Exemple...",
      #PB_Window_SystemMenu |
      #PB_Window_ScreenCentered)
2     SetWindowColor(0, RGB(255,
      255, 0)) ; Fenêtre avec un
      fond de couleur jaune
3
4     ButtonGadget  (1, 10, 60,
      200, 30, "Cliquer")
5
6     Repeat
7       Event = WaitWindowEvent()
8
9       Select Event
10
11        Case #PB_Event_Gadget
12          Select EventGadget()
```

```
13            Case 1
14              Couleur =
    GetWindowColor(0)
15
16              Debug "La
    couleur de la fenêtre est
    " + Str(Couleur)
17              Debug
    "Composante Rouge : " +
    Str(Red(Couleur))
18              Debug
    "Composante Verte : " +
    Str(Green(Couleur))
19              Debug
    "Composante Bleue : " +
    Str(Blue(Couleur))
20          EndSelect
21
22      EndSelect
23    Until Event =
    #PB_Event_CloseWindow
24  EndIf
```

Voir aussi

SetWindowColor()

OS Supportés

Tous

157.15 GetWindowData

Syntaxe

```
Resultat =
    GetWindowData(#Fenetre)
```

Description

Renvoie la donnée qui a été stockée dans une fenêtre avec la fonction SetWindowData() .
Ceci permet d'associer une valeur personnalisée à n'importe quelle fenêtre.

Arguments

#**Fenetre** La fenêtre à utiliser.

Valeur de retour

Renvoie la donnée associée à la fenêtre. S'il n'y a pas de donnée associée à cette fenêtre, la fonction renvoie 0.

Voir aussi

SetWindowData() , GetGadgetData() , SetGadgetData()

OS Supportés

Tous

157.16 GetWindowState

Syntaxe

```
Resultat =
    GetWindowState(#Fenetre)
```

Description

Renvoie l'état minimisé ou maximisé d'une fenêtre.

Arguments

#**Fenetre** La fenêtre à tester.

Valeur de retour

Peut prendre l'une des valeurs suivantes :

```
#PB_Window_Normal   : La
 fenêtre est affichée
 normalement.
#PB_Window_Maximize: La
 fenêtre est agrandie.
#PB_Window_Minimize: La
 fenêtre est réduite.
```

Remarques

L'état d'affichage d'une fenêtre est modifiable avec la commande SetWindowState() .

Exemple

```
1   If OpenWindow(0, 0, 0, 220,
     100, "Exemple...",
     #PB_Window_SystemMenu |
     #PB_Window_MaximizeGadget
     |
     #PB_Window_ScreenCentered)
2
3    ButtonGadget  (1, 10, 60,
     200, 30, "Cliquer")
4
5    Repeat
6      Event = WaitWindowEvent()
7
```

```
8       Select Event
9
10        Case #PB_Event_Gadget
11          Select EventGadget()
12            Case 1
13              Etat =
    GetWindowState(0)
14
15              Select Etat
16                  Case
    #PB_Window_Normal
17                    Debug "La
    fenêtre est affichée
    normalement."
18                  Case
    #PB_Window_Maximize
19                    Debug "La
    fenêtre est agrandie."
20                  Case
    #PB_Window_Minimize
21                    Debug "La
    fenêtre est réduite."
22                  Default
23                    Debug
    "Etat inconnue"
24              EndSelect
25
26            EndSelect
27
28      EndSelect
29    Until Event =
    #PB_Event_CloseWindow
30  EndIf
```

Voir aussi

SetWindowState()

OS Supportés

Tous

157.17 GetWindowTitle

Syntaxe

```
Resultat\$ =
    GetWindowTitle(#Fenetre)
```

Description

Renvoie le titre d'une fenêtre.

Arguments

#**Fenetre** La fenêtre à utiliser.

Valeur de retour

Renvoie le texte affiché dans la barre de titre de la fenêtre specifiée.

Remarques

Le titre de la fenêtre est modifiable avec la commande SetWindowTitle() .

Exemple

```
If OpenWindow(2, 100, 100, 200, 100, "Un joli titre")
Titre$ = GetWindowTitle(2) ; Titre$ contiendra la chaîne "Un joli titre"
MessageRequester("Réponse","Le titre de la fenêtre est : "+Titre$)
EndIf
```

Exemple

```
If OpenWindow(0, 0, 0, 220, 100, "Exemple...", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)

  ButtonGadget  (1, 10, 60, 200, 30, "Cliquer")

  Repeat
    Event = WaitWindowEvent()

    Select Event

      Case #PB_Event_Gadget
        Select EventGadget()
          Case 1

            Titre$ = GetWindowTitle(0)
            Debug "Le titre de la fenêtre est : " + Titre$

          EndSelect

    EndSelect
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

SetWindowTitle()

OS Supportés

Tous

157.18 HideWindow

Syntaxe

```
HideWindow(#Fenetre, Etat, [,
    Options])
```

Description

Cache ou rend visible une fenêtre.

Arguments

#**Fenetre** La fenêtre à utiliser.

Etat Peut prendre une des valeurs suivantes :

```
#True  : La fenêtre est
 masquée
#False : La fenêtre est
 affichée. Elle est aussi
 activée (focus)
         sauf avec
 l'option
 #PB_Window_NoActivate.
```

Options Peut être une combinaison des valeurs suivantes :

```
#PB_Window_NoActivate    :
 La fenêtre s'affiche
 mais n'est pas activée.
 (valable uniquement si
 la fenêtre est visible).
#PB_Window_ScreenCentered:
 La fenêtre sera centrée
 dans l'écran (valable
 uniquement si la fenêtre
 est visible).
#PB_Window_WindowCentered:
 La fenêtre sera centrée
 dans la fenêtre (valable
 uniquement si la fenêtre
 est visible).
```

Valeur de retour

Aucune.

Exemple

```
1  If OpenWindow(0, 200, 200,
    220, 100, "Exemple...",
    #PB_Window_SystemMenu)
2
```

```
3     ButtonGadget  (1, 10, 60,
      200, 30, "Cacher la
      fenêtre")
4
5     Repeat
6       Event = WaitWindowEvent()
7
8       Select Event
9
10        Case #PB_Event_Gadget
11          Select EventGadget()
12            Case 1
13
14              HideWindow(0,
      #True)
15              Debug "La
      fenêtre est cachée."
16              MessageRequester("Info",
      "La fenêtre est cachée.
      Cliquez sur OK pour la
      faire réapparaitre")
17              HideWindow(0,
      #False)
18              Debug "La
      fenêtre est visible."
19
20            EndSelect
21
22      EndSelect
23    Until Event =
      #PB_Event_CloseWindow
24  EndIf
```

Voir aussi

OpenWindow()

OS Supportés

Tous

157.19 IsWindow

Syntaxe

```
Resultat = IsWindow(#Fenetre)
```

Description

Teste si le numéro de fenêtre est valide et correctement initialisé.

Arguments

#**Fenetre** La fenêtre à utiliser.

Valeur de retour

Renvoie une valeur non nulle si le numéro est valide, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'une fenêtre est prête à être utilisée.

Voir aussi

OpenWindow()

OS Supportés

Tous

157.20 OpenWindow

Syntaxe

```
Resultat =
    OpenWindow(#Fenetre, X, Y,
    LargeurInterne,
    HauteurInterne, Titre$ [,
    Options [, FenetreMereID]])
```

Description

Ouvre une nouvelle fenêtre avec les paramètres spécifiés.

Arguments

#Fenetre Le numéro identifiant la nouvelle fenêtre.
Si #PB_Any est utilisé pour le paramètre '#Fenetre', le numéro de la fenêtre sera renvoyé dans 'Resultat'.

X, Y Les coordonnées en pixels, de la position de la fenêtre (sauf si l'une des options de centrage est utilisée).
Si l'une d'elle est définie avec #PB_Ignore alors le système choisira une position pour la fenêtre.

LargeurInterne, HauteurInterne Les dimensions de la zone interne de la fenêtre (dite zone cliente), sans les bordures, la barre de titre et autres décorations.

Titre$ Le titre de la nouvelle fenêtre.

Options (optionnel) Peut être une combinaison de :

#PB_Window_SystemMenu
: Autorise le menu système sur la barre de titre (par défaut).
#PB_Window_MinimizeGadget
: Ajoute l'icône 'Réduire' sur la barre de titre. #PB_Window_SystemMenu est ajouté automatiquement.
#PB_Window_MaximizeGadget
: Ajoute l'icône 'Agrandir' sur la barre de titre. #PB_Window_SystemMenu est ajouté automatiquement.
(MacOS : #PB_Window_SizeGadget sera également ajouté automatiquement).
#PB_Window_SizeGadget
: Ajoute les possibilités de redimensionnement à la fenêtre.
#PB_Window_Invisible
: Crée la fenêtre mais ne l'affiche pas.
#PB_Window_TitleBar
: Crée la fenêtre avec une barre de titre.
#PB_Window_Tool
: Crée une fenêtre avec une barre de titre plus fine, et non visible dans la barre des tâches.
#PB_Window_BorderLess
: Crée une fenêtre sans bordure.
#PB_Window_ScreenCentered
: Centre la fenêtre au milieu de l'écran. Les paramètres X,Y sont ignorés.
#PB_Window_WindowCentered
: Centre la fenêtre au milieu de la fenêtre mère ('FenetreMereID' doit être spécifié). Les paramètres x,y sont ignorés.
#PB_Window_Maximize
: Ouvre la fenêtre en mode maximisé. (Note: sous Linux, certains gestionnaires de fenêtres ne supportent

```
    pas cela)
  #PB_Window_Minimize
    : Ouvre la fenêtre en
    mode minimisé.
  #PB_Window_NoGadgets
    : Empêche la création
    d'une GadgetList.
    UseGadgetList()
  peut être utilisé pour le
    faire plus tard.
  #PB_Window_NoActivate
    : Ne pas activer la
    fenêtre après son
    ouverture.
```

FenetreMereID' (optionnel) Le numéro de la fenêtre mère. 'FenetreMereID' peut être obtenu facilement par la commande WindowID() .

Valeur de retour

Renvoie une valeur non nulle si la création de la nouvelle fenêtre s'est bien déroulée, zéro sinon.
Si #PB_Any a été utilisé comme paramètre alors le numéro de la nouvelle fenêtre est renvoyé dans 'Resultat'.

Remarques

Tous les évènements possibles dans une fenêtre sont gérés par les commandes WindowEvent() et WaitWindowEvent() . Pour les situations spéciales les callbacks sont utilisées, voir la description de SetWindowCallback() . Sous Windows, la largeur et la hauteur d'une fenêtre avec une barre de titre ne peuvent pas être inférieures à environ 100 pixels. Pour ouvrir une fenêtre plus petite, utiliser l'option #PB_Window_BorderLess.
Note : Une fenêtre ne devrait pas être ouverte dans un thread , car il y a une limitation sur OS X et Linux. Une erreur du débogueur sera levée.

Exemple

```
1  If OpenWindow(0, 0, 0, 220,
     100, "Exemple...",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2
3    ButtonGadget  (1, 10, 60,
     200, 30, "Fermer")
4
5    Repeat
```

```
6       Event = WaitWindowEvent()
7
8       Select Event
9
10        Case #PB_Event_Gadget
11          Select EventGadget()
12            Case 1
13               CloseWindow(0)
14               End
15          EndSelect
16
17      EndSelect
18    Until Event =
      #PB_Event_CloseWindow
19  EndIf
```

Voir aussi

CloseWindow()

OS Supportés

Tous

157.21 PostEvent

Syntaxe

```
Resultat =
    PostEvent(Evenement [,
    Fenetre, Objet [, Type [,
    Valeur]]])
```

Description

Envoie un évènement à la fin de la file d'attente interne.

Arguments

Evenement L'évènement à envoyer.
Pour une liste des évènements PureBasic, voir WindowEvent() .
Lors de l'utilisation d'évènements personnalisés, la première valeur doit être au moins #PB_Event_FirstCustomValue, pour ne pas entrer en conflit avec les évènements internes.

Fenetre (optionnel) Le numéro de fenêtre associé à l'évènement.
Lors de l'utilisation d'un évènement personnalisé, cela peut être n'importe quel nombre entier.
Cette valeur peut être récupérée avec EventWindow() .

Objet (optionnel) Le numéro de l'objet associé à l'évènement.
Cela peut être par exemple un numéro de gadget ou de menu .
Lors de l'utilisation d'un évènement personnalisé, cela peut être n'importe quel nombre entier positif.
Cette valeur peut être récupérée avec EventGadget() .

Type (optionnel) Le type associé à l'évènement.
Lors de l'utilisation d'un évènement personnalisé la première valeur doit être au moins égale à #PB_EventType_FirstCustomValue, pour ne pas entrer en conflit avec les valeurs internes.
Cette valeur peut être récupérée avec EventType() .

Valeur (optionnel) La données associées à l'évènement.
Seulement avec un évènement personnalisé, cela peut être n'importe quel nombre entier.
Cette valeur peut être récupérée avec EventData() .

Valeur de retour

Renvoie une valeur non nulle si l'envoi de l'évènement à la fin de la file d'attente interne s'est déroulé sans erreur, zéro sinon.

Remarques

Cette commande peut être très utile pour communiquer entre les threads et la boucle principale des évènements. Par exemple, un thread peut envoyer un évènement personnalisé quand il a fini son traitement (avec une donnée associée), de sorte que la boucle principale peut utiliser cette donnée dans un traitement ultérieur.

```
1  ; Tous nos évènements personnalisés
2  Enumeration #PB_Event_FirstCustomValue
3    #EvenementDebutAction
4    #EvenementFinAction
5  EndEnumeration
6
7
8  Procedure Thread(Valeur)
9    PostEvent(#EvenementDebutAction)
10
11   Delay(3000)
12   PostEvent(#EvenementFinAction)
13 EndProcedure
```

```
14
15  OpenWindow(0, 200, 200,
     100, 100, "PostEvent")
16
17  CreateThread(@Thread(), 0)
18
19  Repeat
20    Event = WaitWindowEvent()
21
22    Select Event
23      Case
     #EvenementDebutAction
24        Debug "Le Thread
     commence une action... "
25
26      Case #EvenementFinAction
27        Debug "Le Thread a
     terminé une action"
28    EndSelect
29
30  Until Event =
     #PB_Event_CloseWindow
```

Voir aussi

WindowEvent() , EventWindow() , EventGadget() , EventType() , EventData()

OS Supportés

Tous

157.22 RemoveKeyboardShortcut

Syntaxe

```
RemoveKeyboardShortcut(#Fenetre,
    Raccourci)
```

Description

Enlève le raccourci clavier de la #Fenêtre précédemment ajouté avec AddKeyboardShortcut() .

Arguments

#Fenetre La fenêtre à utiliser.

Raccourci Le raccourci à enlever. Si le paramètre est défini à #PB_Shortcut_All alors tous les raccourcis claviers de la #Fenêtre sont enlevés d'un coup.

Valeur de retour

Aucune.

Remarques

Pour obtenir la liste complète des raccourcis disponibles, voir AddKeyboardShortcut() .

Exemple

```
RemoveKeyboardShortcut(0, #PB_Shortcut_All) ; Enlève tous les raccourcis clavier de la fenêtre 0
```

Voir aussi

AddKeyboardShortcut()

OS Supportés

Tous

157.23 ResizeWindow

Syntaxe

```
ResizeWindow(#Fenetre, X, Y, Largeur, Hauteur)
```

Description

Déplace et redimensionne une fenêtre.

Arguments

#Fenetre La fenêtre à utiliser.

X, Y Les nouvelles coordonnées de la fenêtre, en pixels.
Si 'X' ou 'Y' sont définis avec #PB_Ignore alors les valeurs actuelles de 'X' ou 'Y' ne seront pas modifiées.

Largeur, Hauteur Les nouvelles dimensions de la fenêtre.
Si 'Largeur' ou 'Hauteur' sont définies avec #PB_Ignore alors les valeurs actuelles de 'Largeur' ou 'Hauteur' ne seront pas modifiées.

Valeur de retour

Aucune.

Exemple

```
If OpenWindow(0, 0, 0, 220, 100, "Exemple...", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
```

```
2
3     ButtonGadget  (1, 10, 60,
      200, 30, "Déplacer et
      Agrandir la fenêtre")
4
5     Repeat
6       Event = WaitWindowEvent()
7
8       Select Event
9
10        Case #PB_Event_Gadget
11          Select EventGadget()
12            Case 1
13
14              ResizeWindow(0,
      100, 100, 350, #PB_Ignore)
15
16            EndSelect
17
18      EndSelect
19    Until Event =
      #PB_Event_CloseWindow
20  EndIf
```

OS Supportés

Tous

157.24 SetActiveWindow

Syntaxe

```
SetActiveWindow(#Fenetre)
```

Description

Active une fenêtre et lui donne le focus.

Arguments

#**Fenetre** La fenêtre à activer.

Valeur de retour

Aucune.

Remarques

Activer une fenêtre c'est aussi lui donner le focus à l'intérieur du programme mais cela ne veut pas dire que le programme s'affichera en avant-plan.

Exemple

```
1  If OpenWindow(0, 400, 200,
     220, 200, "Exemple1",
     #PB_Window_SystemMenu)
2    OpenWindow(1, 500, 200,
     220, 200, "Exemple2",
     #PB_Window_SystemMenu)
3
4    SetActiveWindow(0)
5
6    Repeat:Until
     WaitWindowEvent() =
     #PB_Event_CloseWindow
7  EndIf
```

Voir aussi

GetActiveWindow()

OS Supportés

Tous

157.25 SetWindowCallback

Syntaxe

```
SetWindowCallback(@NomProcedure()
    [, #Fenetre])
```

Description

Mise en place d'une procédure de gestion des évènements, dite de 'Callback'.
Pour programmeurs expérimentés.
Cette fonction n'est disponible que sous Microsoft Windows.

Arguments

@NomProcedure() L'adresse de la procedure de Callback à utiliser.
La procédure de Callback doit avoir 4 paramètres et se présente sous la forme suivante :

```
1  Procedure
     MaProcedureCallback(WindowID,
     Message, WParam, LParam)
2      Resultat =
     #PB_ProcessPureBasicEvents
3      ;
4      ; Votre code ici
5      ;
6      ProcedureReturn Resultat
7  EndProcedure
```

#**Fenetre (optionnel)** Permet d'affecter la callback seulement à une fenêtre spécifiée.
Si ce paramêtre est omis, la procedure de Callback sera appelée par toutes les fenêtres.

Valeur de retour

Aucune.

Remarques

Les évènements classiques devraient toujours être gérés avec les commandes WaitWindowEvent() ou WindowEvent() . Attention, c'est relativement bas niveau et peut interférer avec les évènements PureBasic si elle est incorrectement utilisée. Pour annuler une Callback (fonction de rappel), il suffit d'appeler SetWindowCallback(0 [, #Fenetre]). L'exemple qui suit montre comment tester quelques paramètres d'une fenêtre. (En utilisant les constantes de l' API Windows) :

```
1   Procedure WinCallback(hWnd, uMsg, WParam, LParam)
2     ; Windows remplit automatiquement les paramètres.
3     ; Ces paramètre sont utilisable dans le code de la callback.
4
5     If uMsg = #WM_SIZE
6       Select WParam
7         Case #SIZE_MINIMIZED
8           Debug "La fenêtre est minimisée"
9         Case #SIZE_RESTORED
10          Debug "La fenêtre est rétablie"
11        Case #SIZE_MAXIMIZED
12          Debug "La fenêtre est agrandie"
13      EndSelect
14    EndIf
15
16    ProcedureReturn #PB_ProcessPureBasicEvents
17  EndProcedure
18
19
20  If OpenWindow(0, 0, 0, 200, 100, "Messages", #PB_Window_MinimizeGadget | #PB_Window_MaximizeGadget)
```

```
21
22      SetWindowCallback(@WinCallback())
          ; active la callback
23
24      Repeat
25        Select WaitWindowEvent()
26          Case
      #PB_Event_CloseWindow
27            End
28        EndSelect
29      ForEver
30
31    EndIf
```

OS Supportés

Windows

157.26 SetWindowColor

Syntaxe

```
SetWindowColor(#Fenetre,
    Couleur)
```

Description

Change la couleur de fond d'une fenêtre.

Arguments

#Fenetre La fenêtre à utiliser.

Couleur La nouvelle couleur de fond. La commande RGB() peut être utilisée pour obtenir une couleur valide. Un tableau avec les couleurs les plus courantes est disponible ici . Utiliser la valeur #PB_Default comme paramètre 'Couleur' pour réinitialiser la couleur de fond par la couleur par défaut.

Valeur de retour

Aucune.

Remarques

La couleur de fond d'une fenêtre peut être récupérée à l'aide de GetWindowColor() .

Exemple

```
1   If OpenWindow(0, 0, 0, 220,
      100, "Exemple...",
      #PB_Window_SystemMenu |
      #PB_Window_ScreenCentered)
```

```
    SetWindowColor(0, RGB(255, 0 ,0)) ; Fenêtre Rouge

    ButtonGadget  (1, 10, 60, 200, 30, "Changer la couleur de la fenêtre")

    Repeat
      Event = WaitWindowEvent()

      Select Event

        Case #PB_Event_Gadget
          Select EventGadget()
            Case 1

              SetWindowColor(0, RGB(255, 255, 0))

            EndSelect

      EndSelect
    Until Event = #PB_Event_CloseWindow
  EndIf
```

Voir aussi

GetWindowColor()

OS Supportés

Tous

157.27 SetWindowData

Syntaxe

```
SetWindowData(#Fenetre, Valeur)
```

Description

Stocke une donnée dans une fenêtre.
Cette valeur peut ensuite être lue avec la fonction GetWindowData() .
Ceci permet d'associer une valeur personnalisée à n'importe quelle fenêtre.

Arguments

#**Fenetre** La fenêtre à utiliser.

Valeur La donnée à associer à la fenêtre.

Valeur de retour

Aucune.

Voir aussi

GetWindowData() , SetGadgetData() , GetGadgetData()

OS Supportés

Tous

157.28 SetWindowState

Syntaxe

```
SetWindowState(#Fenetre, Etat)
```

Description

Change l'état minimisé ou maximisé d'une fenêtre.

Arguments

#**Fenetre** La fenêtre à utiliser.

Etat Peut prendre l'une des valeurs suivantes :

```
#PB_Window_Normal   : La
 fenêtre sera affichée
 normalement.
#PB_Window_Maximize: La
 fenêtre sera
 maximisée.(Linux,
 certains gestionnaires
 de fenêtres ne le
 supportent pas)
#PB_Window_Minimize: La
 fenêtre sera minimisée.
```

Remarques

L'état d'affichage d'une fenêtre est récupérable avec la commande GetWindowState() .

Exemple

```
1  If OpenWindow(0, 0, 0, 220,
    100, "Exemple...",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2
3   ButtonGadget  (1, 10, 60,
    200, 30, "Agrandir la
    fenêtre en plein écran")
4
5   Repeat
6     Event = WaitWindowEvent()
7
```

```
8       Select Event
9
10        Case #PB_Event_Gadget
11          Select EventGadget()
12            Case 1
13
14              SetWindowState(0,
    #PB_Window_Maximize)
15
16            EndSelect
17
18      EndSelect
19    Until Event =
    #PB_Event_CloseWindow
20  EndIf
```

Voir aussi

GetWindowState()

OS Supportés

Tous

157.29 SetWindowTitle

Syntaxe

```
SetWindowTitle(#Fenetre,
    NouveauTitre$)
```

Description

Change le texte actuellement affiché dans la barre de titre d'une fenêtre.

Arguments

#Fenetre La fenêtre à utiliser.

NouveauTitre$ Le nouveau titre à utiliser.

Valeur de retour

Aucune.

Remarques

Le titre de la fenêtre est récupérable avec la commande GetWindowTitle() .

Exemple

```
If OpenWindow(0, 0, 0, 220, 100, "Exemple...", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)

  ButtonGadget  (1, 10, 60, 200, 30, "Changer le titre de la fenêtre")

  Repeat
    Event = WaitWindowEvent()

    Select Event

      Case #PB_Event_Gadget
        Select EventGadget()
          Case 1

           SetWindowTitle(0, "Nouveau Titre")

          EndSelect

    EndSelect
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

GetWindowTitle()

OS Supportés

Tous

157.30 SmartWindowRefresh

Syntaxe

```
SmartWindowRefresh(#Fenetre, Etat)
```

Description

Active une méthode de rafraîchissement de l'affichage des fenêtres visant à réduire les scintillements lors d'un redimensionnement.

Arguments

#**Fenetre** La fenêtre à utiliser.

Etat Peut prendre l'une des valeurs suivantes :

```
#True : Rafraîchissement intelligent activé
```

```
#False: Rafraîchissement
 intelligent desactivé
```

Valeur de retour

Aucune.

Remarques

Si la fenêtre n'est pas redimensionnable, cette commande n'est pas utile.
Il n'est pas garanti que cette méthode fonctionne dans tous les cas de figure, et le mieux est de l'activer, faire des tests pour voir si elle a vraiment un effet.

OS Supportés

Windows

157.31 StickyWindow

Syntaxe

```
StickyWindow(#Fenetre, Etat)
```

Description

Affiche une fenêtre toujours au premier plan, devant tous les autres programmes, même si elle n'est pas active.

Arguments

#**Fenetre** La fenêtre à utiliser.

Etat Peut prendre l'une des valeurs suivantes :

```
#True : La fenêtre sera
 toujours au premier plan.
#False: Le fenêtre ne
 restera pas au premier
 plan si elle n'a pas le
 focus.
```

Valeur de retour

Aucune.

Exemple

```
1  If OpenWindow(0, 0, 0, 220,
    100, "Exemple...",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2
```

```
3     ButtonGadget  (1, 10, 60,
      200, 30, "Fenêtre toujours
      au premier plan")
4
5     Repeat
6       Event = WaitWindowEvent()
7
8       Select Event
9
10        Case #PB_Event_Gadget
11          Select EventGadget()
12            Case 1
13
14              StickyWindow(0,
      1)
15              RunProgram("notepad.exe")
16
17            EndSelect
18
19      EndSelect
20    Until Event =
      #PB_Event_CloseWindow
21  EndIf
```

OS Supportés

Tous

157.32 WindowEvent

Syntaxe

```
Resultat = WindowEvent()
```

Description

Teste si un évènement s'est produit sur chacune des fenêtres ouvertes.

Arguments

Aucun.

Valeur de retour

Renvoie le prochain évènement de la file d'attente ou zéro s'il n'y a plus d'évènements.
Contrairement à la fonction WaitWindowEvent() , elle n'attend pas le prochain évènement et le renvoi est donc toujours immédiat. Cela peut être utile quand un traitement doit être fait sans attendre (par exemple une transaction réseau).
Event() peut être utilisé pour récupérer cette valeur.
Les évènements renvoyés sont :

```
  #PB_Event_Menu            :
   Un menu
 a été sélectionné
  #PB_Event_Gadget          :
   Un gadget
 a été cliqué
  #PB_Event_SysTray         :
   Une icône dans la zone
   SysTray
 a été cliquée
  #PB_Event_Timer           :
   Le temps d'un minuteur
 s'est écoulé
  #PB_Event_CloseWindow     :
   L'icône de fermeture de
   fenêtre a été cliquée
  #PB_Event_Repaint         :
   Tout ou partie du contenu
   de la fenêtre a été
   détruit et doit être
   reconstitué
                                (Utile
   pour les applications
   graphiques 2D
)
  #PB_Event_SizeWindow      :
   La fenêtre a été
   redimensionnée
  #PB_Event_MoveWindow      :
   La fenêtre a été déplacée
  #PB_Event_MinimizeWindow  :
   La fenêtre a été minimisée
  #PB_Event_MaximizeWindow  :
   La fenêtre a été maximisée
  #PB_Event_RestoreWindow   :
   La fenêtre a été restaurée
   à sa taille normale
  #PB_Event_ActivateWindow  :
   La fenêtre a été activée
   (gain du focus)
  #PB_Event_DeactivateWindow:
   La fenêtre a été
   désactivée (perte du focus)
  #PB_Event_LeftDoubleClick :
   Un double clic gauche de
   la souris s'est produit
   sur la fenêtre
  #PB_Event_LeftClick       :
   Un clic gauche de la
   souris s'est produit sur
   la fenêtre
  #PB_Event_RightClick      :
   Un clic droit de la souris
   s'est produit sur la
   fenêtre. Cela peut être
   utile pour afficher un
   menu contextuel
  #PB_Event_WindowDrop      :
   Une opération Glisser &
```

```
    Déposer
  s'est terminée sur une
    fenêtre (Voir remarque
    ci-dessous)
   #PB_Event_GadgetDrop       :
    Une opération Glisser &
    Déposer
  s'est terminée sur un gadget
  (Voir remarque ci-dessous)
```

Vous trouverez un exemple dans la description de WaitWindowEvent() .

Remarques

Remarque importante n° 1 : Comme cette commande n'est pas bloquante, elle peut consommer beaucoup de temps processeur quand elle est dans une boucle d'attente. Dans ce cas, utilisez la commande Delay() , sauf en présence de la fonction FlipBuffers() , ou mieux, il est préférable d'utiliser la commande WaitWindowEvent() avec une petite valeur de minuteur (timeout), comme par exemple 'WaitWindowEvent(1)'.
Remarque importante n° 2 : La boucle d'évènements de la fenêtre ne doit pas être traitée dans un thread , car il y a une limitation sur OS X et Linux. Une erreur du débogueur sera levée.
Pour obtenir le numéro de la fenêtre dans laquelle s'est produit l'évènement, utilisez la fonction EventWindow() .
Après un évènement #PB_Event_WindowDrop ou #PB_Event_GadgetDrop, les fonctions de la bibliothèque Drag & Drop peuvent être utilisées pour examiner et lire les données déposées.
Si votre code n'utilise pas la commande FlipBuffers() et si la fonction 'WaitWindowEvent(1) ne convient pas alors la boucle WindowEvent() doit respecter la structure suivante :

```
1  Repeat
2    Event = WindowEvent()
3
4    If Event     ; Un
     évènement est dans la file
     d'attente, il faut le
     gérer.
5      ....
6    Else
7      Delay(1)   ; Plus
     d'évènements dans la file
     d'attente, laissons du
     temps processeur aux
     autres applications !
```

```
    EndIf
  Until Event =
   #PB_Event_CloseWindow
```

Important : Le délai ne doit pas être spécifié après chaque évènement, car quand il y a de nombreux évènements dans la file d'attente (lors d'un rafraîchissement de fenêtre, de gadgets etc..) l'application attendra 1 ms entre chaque évènement ce qui rendra l'affichage très lent. Le délai est utile uniquement quand il n'y a plus d'évènement dans la file d'attente (donc quand WindowEvent() renvoie 0). Une autre façon de procéder est d'utiliser les 'timers' avec WaitWindowEvent(1) ou d'en créer un avec AddWindowTimer() .

Exemple : Avec Gadget

```
    If OpenWindow(0, 0, 0,
   600, 100, "Position de la
   souris sur la fenêtre: ",
   #PB_Window_SystemMenu |
   #PB_Window_ScreenCentered)
    TextGadget(0, 10, 6, 200,
   20, "")

    Repeat
      Event = WindowEvent()

      If Event <> 0 ; Tous
   les évènements sont traités
        SetWindowTitle(0,
   "Position de la souris sur
   la fenêtre: " +
   Str(WindowMouseX(0)) + ","
   + Str(WindowMouseX(0)))
      Else

      Delay(1) ; En absence
   de FlipBuffers(), Delay()
   libére le processeur
      EndIf
    Until Event =
   #PB_Event_CloseWindow
  EndIf
```

Exemple : Avec Gadget (Variante)

```
  If OpenWindow(0, 0, 0, 300,
   30, "Position de la souris
   sur le bureau",
   #PB_Window_SystemMenu |
   #PB_Window_ScreenCentered)
    TextGadget(0, 10, 6, 200,
   20, "")
```

```
3
4       Repeat
5         Event = WindowEvent()
6
7         If Event = 0 ; Il n'y a plus d'évènement dans la file d'attente
8           SetGadgetText(0, "Coordonnées : "+Str(DesktopMouseX())+","+Str(DesktopMouseY()))
9         EndIf
10
11        Delay(20) ; En absence de FlipBuffers(), Delay() libére le processeur pour le multi-tâche
12
13      Until Event = #PB_Event_CloseWindow
14    EndIf
```

Exemple : Sans Gadgets

```
1   ;Quelques variables
2   BalleX = 400
3   BalleY = 200
4   BalleVitesseY.f = 5
5   Gravitation.f = 2
6
7   ;Initialisation
8   If InitSprite()
9     InitKeyboard()
10    InitMouse()
11  EndIf
12
13  ;Création de la fenêtre
14  OpenWindow(0, 0, 0, 800, 600, "WindowEvent", #PB_Window_SystemMenu|#PB_Window_ScreenCentered)
15  OpenWindowedScreen(WindowID(0), 0, 0, 800, 600)
16
17  ;Création du sol
18  Sol = CreateSprite(#PB_Any, 800, 30)
19  StartDrawing(SpriteOutput(Sol))
20  Box(0,0,800,30,RGB(128, 0, 0))
21  StopDrawing()
22
23  ;Création de la Balle
24  Balle = CreateSprite(#PB_Any, 16, 16)
25  StartDrawing(SpriteOutput(Balle))
26  Box(0,0,16,16,RGB(135, 206, 235))
```

```
27  Circle(8,8,8,RGB(255, 255, 0))
28  StopDrawing()
29
30  ;Création de la jauge
31  Image = CreateImage(#PB_Any, 8, 8, 24, RGB(255, 255, 255))
32  *Memoire=EncodeImage(Image,#PB_ImagePlugin_BMP)
33  Jauge = CatchSprite(#PB_Any, *Memoire)
34
35
36  ;Boucle principale
37  Repeat
38
39    Repeat
40    ;Gestion des évènements de la fenêtre
41    ;=====================================
42    ;Essayez les 3 possibilités mais une seule à la fois
43    Evenement = WindowEvent()  ; Animation
44    ;Evenement = WaitWindowEvent()  ; Blocage de l'animation
45    ;Evenement = WaitWindowEvent(1) ; Animation mais délai d'1 ms inutile et de plus c'est une mauvaise
46                                                          ; façon de programmer car la pile des évènements n'est pas vidée
47
48    Select Evenement
49      Case #PB_Event_CloseWindow
50      End
51    EndSelect
52    Until Evenement=0
53
54    FlipBuffers() ; ==> Avec WindowEvent(), FlipBuffers() libère le processeur pour le multitâche et dispense d'un Delay(1)
55    ClearScreen(RGB(135, 206, 235))
56
57    ExamineKeyboard()
      ;Evènement clavier
58
```

```
59      DisplaySprite(Jauge, 50,
        570-BalleY) ;Affichage de
        la jauge
60      ZoomSprite(Jauge,  20,
        570)
61
62      DisplaySprite(Sol, 0,
        570) ;Affichage du sol
63
64      DisplaySprite(Balle,
        BalleX, BalleY) ;Affichage
        de la Balle
65
66      ;Mouvement de la Balle
67      BalleVitesseY =
        BalleVitesseY + Gravitation
68      BalleY = BalleY +
        BalleVitesseY
69
70      ;Gestion de la collision
        de la balle avec le sol
71      If SpriteCollision(Balle,
        BalleX, BalleY+16, Sol, 0,
        570)
72      BalleY= 554
73      BalleVitesseY =
        -BalleVitesseY
74      EndIf
75
76    Until
      KeyboardPushed(#PB_Key_Escape)
```

Voir aussi

WaitWindowEvent() , EventWindow() , Event() , EventGadget() , EventMenu() , EventTimer() , EventData() , EventType() , PostEvent() , BindEvent() , UnbindEvent()

OS Supportés

Tous

157.33 WaitWindowEvent

Syntaxe

```
Resultat =
    WaitWindowEvent([Minuteur])
```

Description

Attend qu'un nouvel évènement se produise. Cette fonction est identique à WindowEvent() , mais en plus, elle bloque l'exécution du programme, ce qui est très important dans un environnement multi-tâches.

Arguments

Minuteur (optionnel) Permet de spécifier le temps maximal (en millisecondes) durant lequel la fonction sera bloquante si aucun évènement ne survient.
Si aucun délai n'est spécifié, il attend indéfiniment jusqu'à ce qu'un évènement se produise.

Valeur de retour

Renvoie l'évènement qui s'est produit. Event() peut être utilisé pour récupérer cette valeur.
Comme avec la fonction WindowEvent() , les évènements possibles sont :

```
  #PB_Event_Menu             :
   Un menu
 a été sélectionné
  #PB_Event_Gadget           :
   Un gadget
 a été cliqué
  #PB_Event_SysTray          :
   Une icône dans la zone
   SysTray
 a été cliquée
  #PB_Event_Timer            :
   Le temps d'un minuteur
 s'est écoulé
  #PB_Event_CloseWindow      :
   L'icône de fermeture de
   fenêtre a été cliquée
  #PB_Event_Repaint          :
   Tout ou partie du contenu
   de la fenêtre a été
   détruit et doit être
   reconstitué
                                (Utile
   pour les applications
   graphiques 2D
)
  #PB_Event_SizeWindow       :
   La fenêtre a été
   redimensionnée
  #PB_Event_MoveWindow       :
   La fenêtre a été déplacée
  #PB_Event_MinimizeWindow   :
   La fenêtre a été minimisée
  #PB_Event_MaximizeWindow   :
   La fenêtre a été maximisée
  #PB_Event_RestoreWindow    :
   La fenêtre a été restaurée
   à sa taille normale
  #PB_Event_ActivateWindow   :
   La fenêtre a été activée
   (gain du focus)
```

```
#PB_Event_DeactivateWindow:
  La fenêtre a été
  désactivée (perte du focus)
#PB_Event_WindowDrop      :
  Une opération Glisser &
  Déposer
s'est terminée sur une
  fenêtre
#PB_Event_GadgetDrop      :
  Une opération Glisser &
  Déposer
s'est terminée sur un gadget

#PB_Event_RightClick      :
  Un clic droit de la souris
  s'est produit sur la
  fenêtre. Cela peut être
  utile pour afficher un
  menu contextuel
#PB_Event_LeftClick       :
  Un clic gauche de la
  souris s'est produit sur
  la fenêtre
#PB_Event_LeftDoubleClick :
  Un double clic gauche de
  la souris s'est produit
  sur la fenêtre
```

Remarques

Une application devrait, si possible, toujours utiliser cette fonction en préférence à WindowEvent() car elle ne prend pas de temps CPU en attente d'un évènement.
La boucle d'évènements de la fenêtre ne doit pas être traitée dans un thread , car il y a une limitation sur OS X et Linux. Une erreur du débogueur sera levée.
WaitWindowEvent() ne peut être appelé qu'une seule fois par boucle d'évènements, sinon les évènements seront "perdus" (chaque évènement ne peut être traité qu'une seule fois et n'est plus disponible pour un deuxième traitement).
Pour obtenir le numéro de la fenêtre où s'est produit l'évènement, utiliser la fonction EventWindow() .

Exemple : Cas général

```
1  If OpenWindow(0, 0, 0, 230,
     90, "Exemple de gestion
     des évènements...",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2
3    ButtonGadget  (1, 10, 10,
     200, 20, "Cliquez moi")
```

```
    CheckBoxGadget(2, 10, 40, 200, 20, "Cochez  moi")

    If CreateMenu(0, WindowID(0))
      MenuTitle("Menu")
      MenuItem(1, "Item 1")
      MenuItem(2, "Item 2")
      MenuItem(3, "Item 3")
    EndIf

    Repeat
      Event = WaitWindowEvent()

      Select Event

        Case #PB_Event_Gadget
          Select EventGadget()
            Case 1 : Debug "Bouton cliqué !"
            Case 2 : Debug "Case à cocher cliquée !"
          EndSelect

        Case #PB_Event_Menu
          Select EventMenu()
            Case 1 : Debug "Menu item 1 cliqué !"
            Case 2 : Debug "Menu item 2 cliqué !"
            Case 3 : Debug "Menu item 3 cliqué !"
          EndSelect

      EndSelect
    Until Event = #PB_Event_CloseWindow
  EndIf
```

Exemple : Avec minuteur dans WaitWindowEvent

```
  If OpenWindow(0, 0, 0, 300, 30, "Position de la souris sur le bureau", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)
    TextGadget(0, 10, 6, 200, 20, "")

    Repeat
      Event = WaitWindowEvent(20)

      If Event = 0 ; Il n'y a plus d'évènement dans la file d'attente, libère le processeur quelques
```

```
    millisecondes pour le
    multi-tâches
8         SetGadgetText(0,
    "Coordonnées : " +
    Str(DesktopMouseX()) + ","
    + Str(DesktopMouseY()))
9       EndIf
10
11     Until Event =
    #PB_Event_CloseWindow
12  EndIf
```

Exemple : Avec une minuterie de fenêtre

```
1   If OpenWindow(0, 0, 0, 300,
     30, "Position de la souris
    sur le bureau",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2    TextGadget(0, 10, 6, 200,
    20, "")
3    AddWindowTimer(0, 0, 10)
    ; Minuterie de 10 ms
4
5    Repeat
6      Event =
    WaitWindowEvent()
7      If Event =
    #PB_Event_Timer ; Toutes
    les 10 ms => Affichage des
    coordonnées
8         SetGadgetText(0,
    "Coordonnées : " +
    Str(DesktopMouseX()) + ","
    + Str(DesktopMouseY()))
9       EndIf
10
11     Until Event =
    #PB_Event_CloseWindow
12  EndIf
```

Voir aussi

WindowEvent() , EventWindow() , Event() , EventGadget() , EventMenu() , EventTimer() , EventData() , EventType() , PostEvent() , BindEvent() , UnbindEvent()

OS Supportés

Tous

157.34 BindEvent

Syntaxe

```
BindEvent(Evenement,
    @Callback() [, Fenetre [,
    Objet [, TypeEvenement]]])
```

Description

Ajoute un évènement à la file d'attente des évènements grace à une procédure dite de 'callback'.
C'est un moyen supplémentaire pour gérer les évènements dans PureBasic, qui fonctionne sans problème avec les habituelles commandes WindowEvent() et WaitWindowEvent() . Cela permet également d'avoir des notifications d'évènements en temps réel car le callback peut être invoqué dès que l'évènement se produit (utile pour ScrollBarGadget() , redimensionnement d'une fenêtre, etc.) Un évènement peut être supprimé avec UnbindEvent() .

Arguments

Evenement L'évènement à ajouter.
Pour une liste complète des évènements, consultez WindowEvent() .
Les évènements personnalisés sont également pris en charge, en utilisant PostEvent() .

@Callback() La procédure à appeler lorsque l'évènement se produit. Elle doit être déclarée comme ceci :

```
1  Procedure EventHandler()
2    ; Du code ici...
3  EndProcedure
```

Les fonctions de PureBasic comme EventGadget() , EventWindow() , EventMenu() , EventType() et EventData() sont disponibles pour obtenir plus d'informations sur l'évènement.
Note : WindowEvent() et WaitWindowEvent() ne devraient jamais être appelées depuis l'intérieur du Callback() sinon le programme peut se verrouiller ou avoir un comportement erroné.

Fenetre (optionnel) Le numéro de la fenêtre à utiliser.
L'évènement se produit uniquement s'il provient de cette fenêtre.
Avec #PB_All, toutes les fenêtres sont susceptibles d'être à l'origine de l'évènement (si spécifié, les paramètres "Objet" et "TypeEvenement" doivent être mis à #PB_All).

Objet (optionnel) L'évènement sera lié à un objet en particulier. Pour cela, utiliser son numéro d'identification.
Ce peut être un gadget , un élément d'un menu ou un systray .
Avec #PB_All tous les gadgets sont susceptibles d'être à l'origine de l'évènement (si spécifié, le paramètre "TypeEvenement" doit être mis à #PB_All).

TypeEvenement (optionnel) Le type d'évènement en particulier.
Pour une liste complète des types pris en charge, voir EventType() .
Avec #PB_All tous les types d'évènements sont susceptibles d'être à l'origine de l'évènement.

Valeur de retour

Aucune.

Exemple

```
1   Procedure SizeWindowHandler()
2     Debug "Evènement - Redimensionnement- de la fenêtre #" + EventWindow()
3
4     ; Redimensionne le gadget pour l'adapter aux nouvelles dimensions de la fenêtre
5     ;
6     ResizeGadget(0, #PB_Ignore, #PB_Ignore, WindowWidth(EventWindow())-20, WindowHeight(EventWindow())-20)
7   EndProcedure
8
9   OpenWindow(0, 100, 100, 200, 200, "Redimensionnement en temps réel", #PB_Window_SizeGadget | #PB_Window_SystemMenu)
10  EditorGadget(0, 10, 10, 180, 180)
11
12  BindEvent(#PB_Event_SizeWindow, @SizeWindowHandler())
13
14  Repeat
15    Event = WaitWindowEvent()
16  Until Event = #PB_Event_CloseWindow
```

Voir aussi

BindGadgetEvent() , BindMenuEvent() , UnbindEvent() , WindowEvent() , WaitWindowEvent()

OS Supportés

Tous

157.35 UnbindEvent

Syntaxe

```
UnbindEvent(Evenement,
    @Callback() [, Fenetre [,
    Objet [, TypeEvenement]]])
```

Description

Supprime un évènement d'un gadget ajouté avec BindEvent() .
Si l'évènement correspondant n'est pas trouvé, cette commande n'a aucun effet.

Arguments

Evenement L'évènement à supprimer.
Pour une liste complète des évènements, consultez WindowEvent() .
Les évènements personnalisés sont également pris en charge, en utilisant PostEvent() .

@Callback() La procédure callback à utiliser.

Fenetre (optionnel) Le numéro de la fenêtre à utiliser.

Objet (optionnel) Le numéro de l'objet à utiliser.
Ce peut être un gadget , un élément d'un menu ou un systray .

TypeEvenement (optionnel) Le type d'évènement à supprimer.
Pour une liste complète des types pris en charge, voir EventType() .

Valeur de retour

Aucune.

Exemple

```
1  Procedure
     SizeWindowHandler()
2     Debug "Evènement -
     Redimensionnement - de la
     fenêtre #" + EventWindow()
```

```
3   EndProcedure
4
5
6   OpenWindow(0, 100, 100,
     300, 200, "Test
     Redimensionnement",
     #PB_Window_SizeGadget |
     #PB_Window_SystemMenu)
7
8   BindEvent(#PB_Event_SizeWindow,
     @SizeWindowHandler())
9   UnbindEvent(#PB_Event_SizeWindow,
     @SizeWindowHandler()) ;
     Unbind it immediately
10
11  Repeat
12    Event = WaitWindowEvent()
13  Until Event =
     #PB_Event_CloseWindow
```

Voir aussi

BindEvent() , BindGadgetEvent() , BindMenuEvent() , WindowEvent() , WaitWindowEvent()

OS Supportés

Tous

157.36 WindowBounds

Syntaxe

```
WindowBounds(#Fenetre,
    LargeurMinimale,
    HauteurMinimale,
    LargeurMaximale,
    HauteurMaximale)
```

Description

Change les dimensions minimale et maximale par défaut d'une fenêtre (en pixels).

Arguments

#Fenetre La fenêtre à utiliser.

LargeurMinimale La nouvelle largeur minimale de la fenêtre.
Si #PB_Ignore est utilisé en guise de paramètre alors la largeur minimale restera inchangée.
Si #PB_Default est utilisé en guise de paramètre, la valeur de largeur minimale est remis à la valeur par défaut du système.

HauteurMinimale La nouvelle hauteur minimale de la fenêtre.
Si #PB_Ignore est utilisé en guise de paramètre alors la hauteur minimale restera inchangée.
Si #PB_Default est utilisé en guise de paramètre, la valeur de hauteur minimale est remis à la valeur par défaut du système.

LargeurMaximale La nouvelle largeur maximale de la fenêtre.
Si #PB_Ignore est utilisé en guise de paramètre alors la largeur maximale restera inchangée.
Si #PB_Default est utilisé en guise de paramètre, la valeur de largeur maximale est remis à la valeur par défaut du système.

HauteurMaximale La nouvelle hauteur maximale de la fenêtre.
Si #PB_Ignore est utilisé en guise de paramètre alors la hauteur maximale restera inchangée.
Si #PB_Default est utilisé en guise de paramètre, la valeur de hauteur maximale est remis à la valeur par défaut du système.

Valeur de retour

Aucune.

Remarques

C'est utile pour éviter qu'une fenêtre ne devienne trop petite ou trop grande quand un utilisateur la redimensionne. Il est possible d'utiliser #PB_Ignore en guise de paramètre pour conserver une des valeurs. Ne fonctionne pas avec les fenêtres sans bordures.

Exemple

```
1  If OpenWindow(0, 0, 0, 300, 300, "Redimensionne moi !", #PB_Window_SystemMenu | #PB_Window_ScreenCentered | #PB_Window_SizeGadget)
2    WindowBounds(0, 200, 200, 400, 400)
3
4    Repeat
5      Event = WaitWindowEvent()
6    Until Event = #PB_Event_CloseWindow
```

```
  EndIf
```

OS Supportés

Tous

157.37 WindowHeight

Syntaxe

```
Resultat =
    WindowHeight(#Fenetre [,
    Mode])
```

Description

Renvoie la hauteur d'une fenêtre.

Arguments

#**Fenetre** La fenêtre à utiliser.

Mode (optionnel) Le mode de calcul de la hauteur de la fenêtre.
Peut être une des valeurs suivantes :

```
#PB_Window_InnerCoordinate:
 Hauteur de la zone
 intérieure de la
 fenêtre. (où un gadget
 peut être ajouté), à
 l'exclusion
                          des
 bordures (par défaut).
#PB_Window_FrameCoordinate:
 Hauteur de la fenêtre,
 incluant les bordures.
```

Valeur de retour

Renvoie la hauteur en pixels de la fenêtre spécifiée.

Exemple

```
1  If OpenWindow(0, 0, 0, 220,
    100, "Exemple...",
    #PB_Window_SystemMenu |
    #PB_Window_ScreenCentered)
2
3   ButtonGadget  (1, 10, 60,
    200, 30, "Hauteur de la
    fenêtre")
4
5   Repeat
6     Event = WaitWindowEvent()
7
8     Select Event
```

```
9
10          Case #PB_Event_Gadget
11            Select EventGadget()
12              Case 1
13
14                Hauteur =
    WindowHeight(0,
    #PB_Window_InnerCoordinate)
15                Debug "Hauteur
    interne : " + Str(Hauteur)
16
17                Hauteur =
    WindowHeight(0,
    #PB_Window_FrameCoordinate)
18                Debug "Hauteur
    totale : " + Str(Hauteur)
19
20              EndSelect
21
22        EndSelect
23      Until Event =
        #PB_Event_CloseWindow
24  EndIf
```

Voir aussi

OpenWindow() , WindowWidth()

OS Supportés

Tous

157.38 WindowID

Syntaxe

```
Resultat = WindowID(#Fenetre)
```

Description

Renvoie l'identifiant unique de la fenêtre dans le système d'exploitation.

Arguments

#**Fenetre** La fenêtre à utiliser.

Remarques

Cette fonction est très utile car FenetreID est souvent demandé par d'autres fonctions. Le résultat FenetreID est aussi appelé 'Handle'. Voir le chapitre de l'aide Handles et Nombres pour plus d'informations.

Exemple

```
If OpenWindow(0, 0, 0, 220, 100, "Exemple...", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)

  ButtonGadget  (1, 10, 60, 200, 30, "Identifiant de la fenêtre")

  Repeat
    Event = WaitWindowEvent()

    Select Event

      Case #PB_Event_Gadget
        Select EventGadget()
          Case 1

            ID=WindowID(0)
            Debug "Identifiant de la fenêtre : " + Str(ID)

          EndSelect

    EndSelect
  Until Event = #PB_Event_CloseWindow
EndIf
```

OS Supportés

Tous

157.39 WindowWidth

Syntaxe

```
Resultat = WindowWidth(#Fenetre [, Mode])
```

Description

Renvoie la largeur d'une fenêtre.

Arguments

#**Fenetre** La fenêtre à utiliser.

Mode (optionnel) Le mode de calcul de la largeur de la fenêtre.
Peut être une des valeurs suivantes :

```
#PB_Window_InnerCoordinate: Largeur de la zone intérieure de la
```

```
    fenêtre. (où un gadget
    peut être ajouté), à
    l'exclusion
                                des
    bordures (par défaut).
  #PB_Window_FrameCoordinate:
    Largeur de la fenêtre,
    incluant les bordures.
```

Valeur de retour

Renvoie la largeur en pixels de la fenêtre spécifiée.

Exemple

```
1   If OpenWindow(0, 0, 0, 220,
     100, "Exemple...",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2
3    ButtonGadget  (1, 10, 60,
     200, 30, "Largeur de la
     fenêtre")
4
5    Repeat
6      Event = WaitWindowEvent()
7
8      Select Event
9
10       Case #PB_Event_Gadget
11         Select EventGadget()
12           Case 1
13
14             Largeur =
     WindowWidth(0,
     #PB_Window_InnerCoordinate)
15             Debug "Largeur
     interne : " + Str(Largeur)
16
17             Largeur =
     WindowWidth(0,
     #PB_Window_FrameCoordinate)
18             Debug "Largeur
     totale : " + Str(Largeur)
19
20           EndSelect
21
22     EndSelect
23   Until Event =
     #PB_Event_CloseWindow
24 EndIf
```

Voir aussi

OpenWindow() , WindowHeight()

OS Supportés

Tous

157.40 WindowX

Syntaxe

```
Resultat = WindowX(#Fenetre
    [, Mode])
```

Description

Renvoie la position X d'une fenêtre dans l'écran.

Arguments

#Fenetre La fenêtre à utiliser.

Mode (optionnel) Le mode de calcul de la position X de la fenêtre.
Peut être une des valeurs suivantes :

```
#PB_Window_FrameCoordinate:
 Position X de la
 fenêtre, incluant les
 bordures (par défaut).
#PB_Window_InnerCoordinate:
 Position X de la zone
 intérieure de la fenêtre
                          (où
 gadget peut être
 ajouté), à l'exclusion
 des bordures.
```

Valeur de retour

Renvoie la position X depuis le bord gauche de l'écran (en pixels) de la fenêtre.

Exemple

```
1   If OpenWindow(0, 0, 0, 220,
     100, "Exemple...",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2
3    ButtonGadget  (1, 10, 60,
     200, 30, "Position X de la
     fenêtre")
4
5    Repeat
6      Event = WaitWindowEvent()
7
8      Select Event
9
10       Case #PB_Event_Gadget
11         Select EventGadget()
```

```
12          Case 1
13
14            X = WindowX(0,
    #PB_Window_InnerCoordinate)
15            Debug "Position
    X interne : " + Str(X)
16
17            X= WindowX(0,
    #PB_Window_FrameCoordinate)
18            Debug "Position
    X totale : " + Str(X)
19
20          EndSelect
21
22      EndSelect
23    Until Event =
    #PB_Event_CloseWindow
24  EndIf
```

Voir aussi

OpenWindow() , WindowY()

OS Supportés

Tous

157.41 WindowY

Syntaxe

```
Resultat = WindowY(#Fenetre
    [, Mode])
```

Description

Renvoie la position Y d'une fenêtre dans l'écran.

Arguments

#**Fenetre** La fenêtre à utiliser.

Mode (optionnel) Le mode de calcul de la position Y de la fenêtre.
Peut être une des valeurs suivantes :

```
#PB_Window_FrameCoordinate:
 Position Y de la
 fenêtre, incluant les
 bordures (par défaut).
#PB_Window_InnerCoordinate:
 Position Y de la zone
 intérieure de la fenêtre
                          (où
 gadget peut être
 ajouté), à l'exclusion
 des bordures.
```

Valeur de retour

Renvoie la position Y depuis le bord haut de l'écran (en pixels) de la fenêtre.

Exemple

```
If OpenWindow(0, 0, 0, 220, 100, "Exemple...", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)

  ButtonGadget  (1, 10, 60, 200, 30, "Position Y de la fenêtre")

  Repeat
    Event = WaitWindowEvent()

    Select Event

      Case #PB_Event_Gadget
        Select EventGadget()
          Case 1

            Y = WindowY(0, #PB_Window_InnerCoordinate)
            Debug "Position X interne : " + Str(Y)

            Y= WindowY(0, #PB_Window_FrameCoordinate)
            Debug "Position X totale : " + Str(Y)

          EndSelect

    EndSelect
  Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

OpenWindow() , WindowX()

OS Supportés

Tous

157.42 WindowMouseX

Syntaxe

```
Resultat = WindowMouseX(#Fenetre)
```

Description

Renvoie la position horizontale de la souris dans la zone interne d'une fenêtre.

Arguments

#**Fenetre** La fenêtre à utiliser.

Valeur de retour

Renvoie la position en X de la souris dans la zone interne de la fenêtre.
Renvoie -1 si la souris se trouve à l'extérieur de la zone.

Remarques

Pour avoir la position absolue en 'X' de la souris sur le bureau, utiliser la commande DesktopMouseX() .

Exemple

```
If OpenWindow(0, 0, 0, 300, 30, "Position de la souris dans la zone interne de la fenêtre.", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)

 TextGadget(0, 10, 6, 500, 20, "")

 Repeat
   Event = WaitWindowEvent(20) ; Boucle au moins chaque 20ms pour une mise à jour

   SetGadgetText(0, "Coordonnées: " + Str(WindowMouseX(0)) + "," + Str(WindowMouseY(0)))
 Until Event = #PB_Event_CloseWindow
EndIf
```

Voir aussi

OpenWindow() , WindowMouseY() , DesktopMouseX() , DesktopMouseY()

OS Supportés

Tous

157.43 WindowMouseY

Syntaxe

```
Resultat =
    WindowMouseY(#Fenetre)
```

Description

Renvoie la position verticale de la souris dans la zone interne d'une fenêtre.

Arguments

#**Fenetre** La fenêtre à utiliser.

Valeur de retour

Renvoie la position en Y de la souris dans la zone interne de la fenêtre.
Renvoie -1 si la souris se trouve à l'extérieur de la zone.

Remarques

Pour avoir la position absolue en 'Y' de la souris sur le bureau, utiliser la commande DesktopMouseY() .

Exemple

```
1   If OpenWindow(0, 0, 0, 300,
     30, "Position de la souris
     dans la zone interne de la
     fenêtre.",
     #PB_Window_SystemMenu |
     #PB_Window_ScreenCentered)
2
3     TextGadget(0, 10, 6, 500,
    20, "")
4
5     Repeat
6       Event =
    WaitWindowEvent(20) ;
    Boucle au moins chaque
    20ms pour une mise à jour
7
8       SetGadgetText(0,
    "Coordonnées: " +
    Str(WindowMouseX(0)) + ","
    + Str(WindowMouseY(0)))
9     Until Event =
    #PB_Event_CloseWindow
10  EndIf
```

Voir aussi

OpenWindow() , WindowMouseX() , DesktopMouseX() , DesktopMouseY()

OS Supportés

Tous

157.44 WindowOutput

Syntaxe

```
Resultat =
    WindowOutput(#Fenetre)
```

Description

Renvoie la valeur OutputID d'une fenêtre nécessaire à la fonction StartDrawing() de la bibliothèque Dessin 2D pour effectuer les dessins 2D directement dessus.

Arguments

#**Fenetre** La fenêtre à utiliser.

Valeur de retour

Renvoie l'OutputID de la fenêtre afin d'effectuer un rendu 2D directement dessus.

Remarques

La mémoire allouée par WindowOutput() est libérée avec StopDrawing() .
Le contenu dessiné sur une fenêtre sera effacé dès que la fenêtre ou une partie de cette dernière sera recouverte par une autre fenêtre, quand elle sera déplacée en dehors de l'écran ou quand elle sera cachée ou minimisée. Pour garder le contenu visible, il est obligatoire de le redessiner après chaque évènement #PB_Event_Repaint. Une manière plus confortable pour afficher un dessin sur une fenêtre est d'utiliser une image et ImageGadget() qu'il est possible de mettre à jour avec SetGadgetState() . Tout l'affichage sera géré par le gadget.

Exemple : Dessin sans #PB_Event_Repaint

```
1   ; Faites disparaître une
      partie de la fenêtre en
      dehors de l'écran
2   If OpenWindow(0, 0, 0, 220,
      100, "Exemple...",
      #PB_Window_SystemMenu |
      #PB_Window_ScreenCentered)
3
```

```
    ButtonGadget  (1, 10, 60, 200, 30, "Dessiner sur la fenêtre")

    Repeat
      Event = WaitWindowEvent()

      Select Event

        Case #PB_Event_Gadget
          Select EventGadget()
            Case 1
              ; Dessin d'un rectangle rouge sur la fenêtre
              If StartDrawing(WindowOutput(0))
                Box(10,10, 200, 30, RGB(255, 0, 0))
                StopDrawing()
              EndIf

        EndSelect

      EndSelect
    Until Event = #PB_Event_CloseWindow
  EndIf
```

Exemple : Dessin avec #PB_Event_Repaint

```
  ; Faites disparaître une partie de la fenêtre en dehors de l'écran
  If OpenWindow(0, 0, 0, 220, 100, "Exemple...", #PB_Window_SystemMenu | #PB_Window_ScreenCentered)

    Repeat
      Event = WaitWindowEvent()

      If Event = #PB_Event_Repaint
        StartDrawing(WindowOutput(0))
          Box(10, 10, 50, 50, RGB(255, 0, 0))
        StopDrawing()
      EndIf
    Until Event = #PB_Event_CloseWindow
  EndIf
```

Voir aussi

StartDrawing() , WindowVectorOutput()

OS Supportés

Tous

157.45 WindowVectorOutput

Syntaxe

```
Resultat =
    WindowVectorOutput(#Fenetre
    [, UniteDeMesure])
```

Description

Renvoie le numéro d'identificationt OutputID d'une fenêtre pour effectuer des opérations de dessin vectoriel.

Arguments

#**Fenetre** La fenêtre à utiliser.

UniteDeMesure (optionnel) Spécifie l'unité utilisée pour mesurer les distances sur le dessin.

```
#PB_Unit_Pixel      : Les
 valeurs sont mesurées en
 pixels (Par défaut)(ou
 point (dots) pour les
 imprimantes)
#PB_Unit_Point      : Les
 valeurs sont mesurées en
 points (1/72 pouce =
 25.4/72 mm = 0,352 778
 mm)
#PB_Unit_Inch       : Les
 valeurs sont mesurées en
 pouces (25,4 millimètres)
#PB_Unit_Millimeter: Les
 valeurs sont mesurées en
 millimètres (0,039 370
 pouce)
```

Valeur de retour

Le OutputID de la fenêtre spécifiée afin d'être utilisé pour des opérations de rendu 2D, avec la fonction StartVectorDrawing() .

Remarques

Sera utilisée avec la bibliothèque VectorDrawing , et ne peut être utilisé que dans un bloc StartVectorDrawing() / StopVectorDrawing() .

La mémoire allouée avec WindowVectorOutput() sera libérée avec StopVectorDrawing() .
Le contenu dessiné sur une fenêtre sera effacé dès que la fenêtre ou une partie de cette dernière sera recouverte par une autre fenêtre, quand elle sera déplacée en dehors de l'écran ou quand elle sera cachée ou minimisée. Pour garder le contenu visible, il est obligatoire de le redessiner après chaque évènement #PB_Event_Repaint. Une manière plus confortable pour afficher un dessin sur une fenêtre est de dessiner dans une image via ImageVectorOutput() et de l'afficher avec un ImageGadget() qu'il est possible de mettre à jour avec SetGadgetState() , si nécessaire. Le rafraîchissement de l'image sera géré par le gadget.

Voir aussi

StartVectorDrawing() , WindowOutput()

OS Supportés

Tous

157.46 EventwParam

Syntaxe

```
Resultat = EventwParam()
```

Description

Renvoie le paramètre WPARAM du dernier évènement.

Arguments

Aucun.

Valeur de retour

Renvoie le paramètre WPARAM du dernier évènement.

Remarques

Cette fonction n'est plus supportée et ne devrait plus être utilisée. Utilisez plutôt la fonction SetWindowCallback() afin d'avoir un plein accès aux messages Windows.

Exemple : Sans callback

```
OpenWindow(0, 0, 0, 300, 200, "Messages", #PB_Window_SystemMenu|#PB_Window_ScreenCentered)

Repeat
  Event = WaitWindowEvent()
  Debug "-> Evènement n°"+Str(Event)+" : WParam="+Str(EventwParam())+", LParam="+Str(EventlParam())
  Select Event
    Case #PB_Event_CloseWindow
      End
  EndSelect
ForEver
```

Exemple : Avec callback

```
Procedure WinCallback(hWnd, uMsg, wParam, lParam)
  Debug "Evènement n°"+Str(uMsg)+" : WParam="+Str(wParam)+" , LParam="+Str(lParam)
  ProcedureReturn #PB_ProcessPureBasicEvents
EndProcedure

OpenWindow(0, 0, 0, 300, 200, "Messages", #PB_Window_SystemMenu|#PB_Window_ScreenCentered)

SetWindowCallback(@WinCallback())

Repeat
  Event = WaitWindowEvent()
  ;Debug "-> Evènement n°"+Str(Event)+" : WParam="+Str(EventwParam())+", LParam="+Str(EventlParam())
  Select Event
    Case #PB_Event_CloseWindow
      End
  EndSelect
ForEver
```

Exemple

```
; Appuyer sur une combinaison de plusieurs
```

```
    touches : CTRL, MAJ,
    Bouton souris, puis
    cliquez gauche
2   OpenWindow(0, 0, 0, 300,
    200, "Messages",
    #PB_Window_SystemMenu|#PB_Window_ScreenCentered)
3
4   TextGadget(0,4,4,392,92,"Clic")
5
6   Repeat
7   event = WaitWindowEvent()
8
9   Select event
10    Case #WM_LBUTTONDOWN
11      x = EventlParam() &
    $FFFF    ; Mot de poids
    faible(16 Bits)
12      y = EventlParam()>>16
          ; Mot de poids fort
    (16 Bits)
13      cles = EventwParam()
14
15      SetGadgetText(0,"X= " +
    Str(x) + "  Y= " + Str(y)
    + Chr(13) + Chr(10) +
    "Clés: " +
    RSet(Bin(cles),32,"0"))
16
17      combinaison$ = ""
18      If cles & #MK_CONTROL
19        combinaison$ =
    combinaison$ + " CTRL "
20      EndIf
21      If cles & #MK_SHIFT
22        combinaison$ =
    combinaison$ + " MAJ "
23      EndIf
24      If cles & #MK_MBUTTON
25        combinaison$ =
    combinaison$ + " Clic
    milieu "
26      EndIf
27      If cles & #MK_RBUTTON
28        combinaison$ =
    combinaison$ + " Clic
    droit "
29      EndIf
30      If cles & #MK_LBUTTON
31        combinaison$ =
    combinaison$ + " Clic
    gauche "
32      EndIf
33
34      Debug  combinaison$
35
36  EndSelect
37  Until event =
    #PB_Event_CloseWindow
```

Voir aussi

EventlParam()

OS Supportés

Windows

157.47 EventlParam

Syntaxe

```
Resultat = EventlParam()
```

Description

Renvoie le paramètre LPARAM du dernier évènement.

Arguments

Aucun.

Valeur de retour

Renvoie le paramètre LPARAM du dernier évènement.

Remarques

Cette fonction n'est plus supportée et ne devrait plus être utilisée. Utilisez plutôt la fonction SetWindowCallback() afin d'avoir un plein accès aux messages Windows.

Exemple : Sans callback

```
1   OpenWindow(0, 0, 0, 300,
     200, "Messages",
     #PB_Window_SystemMenu|#PB_Window_ScreenCentered)
2
3   Repeat
4     Event = WaitWindowEvent()
5     Debug "-> Evènement
     n°"+Str(Event)+" :
     WParam="+Str(EventwParam())+"
     ,
     LParam="+Str(EventlParam())
6     Select Event
7       Case
     #PB_Event_CloseWindow
8         End
9     EndSelect
10  ForEver
```

Exemple : Avec callback

```
Procedure WinCallback(hWnd, uMsg, wParam, lParam)
  Debug "Evènement n°"+Str(uMsg)+" : WParam="+Str(wParam)+" , LParam="+Str(lParam)
  ProcedureReturn #PB_ProcessPureBasicEvents
EndProcedure

OpenWindow(0, 0, 0, 300, 200, "Messages", #PB_Window_SystemMenu|#PB_Window_ScreenCentered)

SetWindowCallback(@WinCallback())

Repeat
  Event = WaitWindowEvent()
  ;Debug "-> Evènement n°"+Str(Event)+" : WParam="+Str(EventwParam())+" , LParam="+Str(EventlParam())
  Select Event
    Case #PB_Event_CloseWindow
      End
  EndSelect
ForEver
```

Exemple

```
; Appuyer sur une combinaison de plusieurs touches : CTRL, MAJ, Bouton souris, puis cliquez gauche
OpenWindow(0, 0, 0, 300, 200, "Messages", #PB_Window_SystemMenu|#PB_Window_ScreenCentered)

TextGadget(0,4,4,392,92,"Clic")

Repeat
event = WaitWindowEvent()

Select event
  Case #WM_LBUTTONDOWN
    x = EventlParam() & $FFFF    ; Mot de poids faible(16 Bits)
    y = EventlParam()>>16
        ; Mot de poids fort (16 Bits)
    cles = EventwParam()

```

```
15        SetGadgetText(0,"X= " +
      Str(x) + "  Y= " + Str(y)
      + Chr(13) + Chr(10) +
      "Clés: " +
      RSet(Bin(cles),32,"0"))
16
17        combinaison$ = ""
18        If cles & #MK_CONTROL
19          combinaison$ =
      combinaison$ + " CTRL "
20        EndIf
21        If cles & #MK_SHIFT
22          combinaison$ =
      combinaison$ + " MAJ "
23        EndIf
24        If cles & #MK_MBUTTON
25          combinaison$ =
      combinaison$ + " Clic
      milieu "
26        EndIf
27        If cles & #MK_RBUTTON
28          combinaison$ =
      combinaison$ + " Clic
      droit "
29        EndIf
30        If cles & #MK_LBUTTON
31          combinaison$ =
      combinaison$ + " Clic
      gauche "
32        EndIf
33
34        Debug  combinaison$
35
36    EndSelect
37    Until event =
       #PB_Event_CloseWindow
```

Voir aussi

EventwParam()

OS Supportés

Windows

Chapitre 158

Window3D

Généralités

La bibliothèque Window3D permet de créer des interfaces graphiques complexes (GUI) dans un environnement 3D. Elle est principalement destinée à la création de jeux ou d'applications qui doivent se lancer en mode plein écran et qui nécessitent une interface utilisateur.
La syntaxe de cette bibliothèque est similaire à la bibliothèque Window .
Le moteur utilisé pour l'interface graphique se nomme CEGUI, il permet l'utilisation de skins, il est rapide et contient de nombreux gadgets prédéfinis. Vous trouverez plus d'informations au sujet de CEGUI ici : http ://www.cegui.org.uk.
InitEngine3D() doit être appelé avec succès avant de pouvoir utiliser les commandes relatives aux fenêtres 3D.
Pour utiliser les fenêtres 3D, une caméra doit être créée au préalable.

OS Supportés

Tous

158.1 CloseWindow3D

Syntaxe

```
CloseWindow3D(#Fenetre3D)
```

Description

Ferme une fenêtre 3D.

Arguments

#Fenetre3D La fenêtre 3D à fermer. Si #PB_All est spécifié, toutes les fenêtres 3D restantes sont fermées.

Valeur de retour

Aucune.

Remarques

Toutes les fenêtres restant ouvertes sont automatiquement fermées quand le programme se termine.

OS Supportés

Tous

158.2 DisableWindow3D

Syntaxe

```
DisableWindow3D(#Fenetre3D,
    Etat)
```

Description

Active ou désactive les interactions de l'utilisateur avec une fenêtre 3D.

Arguments

#**Fenetre3D** La fenêtre 3D à utiliser.

Etat Peut prendre une des valeurs suivantes :

```
#True : La fenêtre 3D est
  désactivée.
#False: La fenêtre 3D est
  activée.
```

Valeur de retour

Aucune.

OS Supportés

Tous

158.3 EventGadget3D

Syntaxe

```
Resultat = EventGadget3D()
```

Description

Utiliser cette fonction après un évènement de type #PB_Event3D_Gadget (renvoyé par WindowEvent3D()) pour connaitre le numéro du gadget qui a été utilisé.

Arguments

Aucun.

Valeur de retour

Le numéro du gadget 3D qui a été utilisé.

Voir aussi

WindowEvent3D()

OS Supportés

Tous

158.4 EventType3D

Syntaxe

```
Resultat = EventType3D()
```

Description

Utiliser cette fonction après WindowEvent3D() pour savoir quel est le type du dernier évènement survenu.

Arguments

Aucun.

Valeur de retour

Les valeurs suivantes sont possibles lorsqu'un évènement du type #PB_Event3D_Gadget (bibliothèque Gadget3D) survient :

```
#PB_EventType3D_Focus
            : Obtention du
  focus.
#PB_EventType3D_LostFocus
          : Perte du focus.
#PB_EventType3D_Change
           : Le contenu a
  changé.
```

Peut être utilisé avec les gadgets suivants :
- SpinGadget3D()
- StringGadget3D()
(Voir la définition du gadget pour connaître les évènements valides)

Voir aussi

WindowEvent3D()

OS Supportés

Tous

158.5 EventWindow3D

Syntaxe

```
Resultat = EventWindow3D()
```

Description

Utiliser cette fonction après WindowEvent3D() pour savoir sur quelle fenêtre s'est produit le dernier évènement.

Arguments

Aucun.

Valeur de retour

Renvoie le numéro de la fenêtre 3D où s'est produit le dernier évènement.

OS Supportés

Tous

158.6 InputEvent3D

Syntaxe

```
InputEvent3D(SourisX,
    SourisY,
    BoutonGaucheSouris, [,
    Texte$, ToucheSpeciale])
```

Description

Injecte des événements dans l'interface graphique 3D (GUI).

Arguments

SourisX, SourisY La position de la souris sur l'interface 3D, en pixels.

BoutonGaucheSouris L'état du bouton gauche de la souris

```
0 : Bouton relâché.
1 : Bouton pressé.
```

Texte$ (optionnel) Le texte injecté dans l'interface graphique, par exemple pour alimenter un StringGadget3D() .

ToucheSpeciale (optionnel) Touche muette du clavier injectée dans le système de l'interface graphique, par exemple pour gérer le Retour Arrière (Backspace), Entrée ou Retour Chariot (Enter) et autres.
Les touches spéciales disponibles sont :

```
#PB_Key_Back   : Retour
 arrière
#PB_Key_Delete: Supprimer
#PB_Key_Return: Entrée
#PB_Key_Up     : Flèche haut
#PB_Key_Down   : Flèche bas
#PB_Key_Left   : Flèche
 gauche
#PB_Key_Right : Flèche
 droite
```

Valeur de retour

Aucune.

Remarques

Cette fonction est nécessaire pour que WindowEvent3D() fonctionne.
Pour plus de souplesse, les événements ne sont pas obtenus automatiquement à partir de la souris et du clavier, mais injectés à la demande.

Exemple

```
1  ; Voici une utilisation
    classique pour renseigner
    les événements de la
    souris et du clavier
2  ;
3  InputEvent3D(MouseX(),
    MouseY(),
    MouseButton(#PB_MouseButton_Left,
    Input$, 0)
```

Voir aussi

WindowEvent3D() , Add3DArchive() , MousePick()

OS Supportés

Tous

158.7 GetActiveWindow3D

Syntaxe

```
Resultat = GetActiveWindow3D()
```

Description

Renvoie le numéro de la fenêtre 3D qui est actuellement active (qui a le 'focus').

Arguments

Aucun.

Valeur de retour

Renvoie le numéro de la fenêtre 3D qui a le 'focus' clavier ou -1 sinon.

Remarques

Une fenêtre peut être activée avec la commande SetActiveWindow3D() .

Voir aussi

SetActiveWindow3D()

OS Supportés

Tous

158.8 GetWindowTitle3D

Syntaxe

```
Resultat\$ =
    GetWindowTitle3D(#Fenetre3D)
```

Description

Renvoie le texte de la barre de titre d'une fenêtre 3D.

Arguments

#**Fenetre3D** La fenêtre 3D à utiliser.

Valeur de retour

Renvoie le texte affiché dans la barre de titre de la fenêtre 3D spécifiée.

Remarques

Le titre de la fenêtre est modifiable avec la commande SetWindowTitle3D() .

Voir aussi

SetWindowTitle3D()

OS Supportés

Tous

158.9 HideWindow3D

Syntaxe

```
HideWindow3D(#Fenetre3D, Etat)
```

Description

Affiche ou cache une fenêtre 3D.

Arguments

#Fenetre3D La fenêtre 3D à utiliser.

Etat Peut prendre une des valeurs suivantes :

```
#True : La fenêtre 3D est
  cachée.
#False: La fenêtre 3D est
  rendue visible.
```

Valeur de retour

Aucune.

OS Supportés

Tous

158.10 IsWindow3D

Syntaxe

```
Resultat =
    IsWindow3D(#Fenetre3D)
```

Description

Teste si une fenêtre 3D est correctement initialisée.

Arguments

#Fenetre3D La fenêtre 3D à utiliser.

Valeur de retour

Renvoie une valeur non nulle si la fenêtre 3D est valide, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'un objet est correctement initialisé.

Voir aussi

OpenWindow3D()

OS Supportés

Tous

158.11 OpenWindow3D

Syntaxe

```
Resultat =
    OpenWindow3D(#Fenetre3D,
    X, Y, LargeurInterne,
    HauteurInterne, Titre$ [,
    Options])
```

Description

Ouvre une nouvelle fenêtre 3D.

Arguments

#Fenetre3D Le numéro d'identification de la fenêtre 3D.
#PB_Any peut être utilisé pour autogénérer ce numéro.

X, Y La position initiale de la fenêtre 3D dans l'écran (sauf si l'une des options de centrage est utilisée).

LargeurInterne, HauteurInterne Taille initiale de la zone client de la fenêtre 3D (sans bordure ni barre de titre...).

Titre$ Le titre de la fenêtre 3D.

Options (optionnel) Peut être une combination de :

```
#PB_Window3D_SizeGadget
      : Redimensionnement
  de la fenêtre 3D.
#PB_Window3D_Invisible
       : Crée la fenêtre
  3D mais ne l'affiche pas.
#PB_Window3D_BorderLess
      : Crée une fenêtre
  3D sans bordures.
```

Valeur de retour

Renvoie une valeur non nulle si la fenêtre 3D est ouverte dans l'écran en cours, zéro sinon.

Remarques

La nouvelle fenêtre 3D devient la fenêtre active sans avoir besoin d'utiliser SetActiveWindow3D() (sauf si elle a été créée en mode invisible).
Tous les évènements possibles des fenêtres 3D sont gérés par la commande WindowEvent3D() .

Voir aussi

CloseWindow3D() , WindowEvent3D()

OS Supportés

Tous

158.12 ResizeWindow3D

Syntaxe

```
ResizeWindow3D(#Fenetre3D, X,
    Y, Largeur, Hauteur)
```

Description

Déplace et redimensionne une fenêtre 3D.

Arguments

#Fenetre3D La fenêtre 3D à utiliser.

X, Y La nouvelle position de la fenêtre 3D. #PB_Ignore peut être utilisée pour laisser un paramètre inchangé.

Largeur, Hauteur Les nouvelles dimensions de la fenêtre 3D. #PB_Ignore peut être utilisée pour laisser un paramètre inchangé.

Valeur de retour

Aucune.

OS Supportés

Tous

158.13 SetActiveWindow3D

Syntaxe

```
SetActiveWindow3D(#Fenetre3D)
```

Description

Active une fenêtre 3D, elle obtient le 'focus'.

Arguments

#**Fenetre3D** La fenêtre 3D à utiliser.

Valeur de retour

Aucune.

Voir aussi

GetActiveWindow3D()

OS Supportés

Tous

158.14 SetWindowTitle3D

Syntaxe

```
SetWindowTitle3D(#Fenetre3D,
    Titre$)
```

Description

Change le texte de la barre de titre d'une fenêtre 3D.

Arguments

#**Fenetre3D** La fenêtre 3D à utiliser.

Titre$ Le nouveau titre.

Valeur de retour

Aucune.

Voir aussi

GetWindowTitle3D()

OS Supportés

Tous

158.15 WindowEvent3D

Syntaxe

```
Resultat = WindowEvent3D()
```

Description

Teste si un évènement s'est produit sur chacune des fenêtres 3D.

Arguments

Aucun.

Valeur de retour

Les évènements possibles sont :

```
  #PB_Event3D_Gadget
    : Un Gadget3D
 a été utilisé
  #PB_Event3D_CloseWindow
    : L'icône de fermeture de
    fenêtre a été cliquée
  #PB_Event3D_SizeWindow
    : La fenêtre a été
    redimensionnée
  #PB_Event3D_MoveWindow
    : La fenêtre a été déplacée
  #PB_Event3D_ActivateWindow
    : La fenêtre a été activée
    (gain du focus)
```

Remarques

InputEvent3D() doit être utilisé pour envoyer des événements au système d'interface graphique 3D et pour pouvoir avoir accès aux évènements provenant de la fenêtre.
Cette fonction est à retour immédiat, elle renvoie le prochain événement de la file d'attente ou zéro s'il n'y en a plus.
Elle n'attend pas pour le prochain évènement.
Pour obtenir le numéro de fenêtre où s'est produit l'évènement, utiliser la fonction EventWindow3D() .

Voir aussi

EventWindow3D() , InputEvent3D()

OS Supportés

Tous

158.16 WindowHeight3D

Syntaxe

```
Resultat =
    WindowHeight3D(#Fenetre3D)
```

Description

Renvoie la hauteur d'une fenêtre 3D.

Arguments

#**Fenetre3D** La fenêtre 3D à utiliser.

Valeur de retour

Renvoie la hauteur en pixels de la fenêtre 3D spécifiée.

Voir aussi

WindowWidth3D()

OS Supportés

Tous

158.17 WindowID3D

Syntaxe

```
WindowID3D =
    WindowID3D(#Fenetre3D)
```

Description

Renvoie l'identifiant unique d'une fenêtre 3D.

Arguments

#**Fenetre3D** La fenêtre 3D à utiliser.

Valeur de retour

Renvoie l'identifiant unique de la fenêtre 3D spécifiée, dans le système d'exploitation.

OS Supportés

Tous

158.18 WindowWidth3D

Syntaxe

```
Resultat =
    WindowWidth3D(#Fenetre3D)
```

Description

Renvoie la largeur d'une fenêtre 3D.

Arguments

#**Fenetre3D** La fenêtre 3D à utiliser.

Valeur de retour

Renvoie la largeur en pixels de la fenêtre 3D spécifiée.

Voir aussi

WindowHeight3D()

OS Supportés

Tous

158.19 WindowX3D

Syntaxe

```
Resultat =
    WindowX3D(#Fenetre3D)
```

Description

Renvoie la position en 'X' (par rapport au bord gauche d'un écran) d'une fenêtre 3D.

Arguments

#**Fenetre3D** La fenêtre 3D à utiliser.

Valeur de retour

Renvoie la position gauche de la fenêtre 3D spécifiée, en pixels.

Voir aussi

WindowY3D()

OS Supportés

Tous

158.20 WindowY3D

Syntaxe

```
Resultat =
    WindowY3D(#Fenetre3D)
```

Description

Renvoie la position en 'Y' (par rapport au bord supérieur d'un écran) d'une fenêtre 3D.

Arguments

#**Fenetre3D** La fenêtre 3D à utiliser.

Valeur de retour

Renvoie la position supérieure de la fenêtre 3D spécifiée, en pixels.

Voir aussi

WindowX3D()

OS Supportés

Tous

Chapitre 159

XML

Généralités

Cette bibliothèque permet de manipuler très facilement les fichiers XML, que ce soit en en lecture ou en écriture. Elle est basée sur expat qui est sous licence MIT consultable ici . Expat est déjà utilisée dans de nombreux projets d'envergures (comme Mozilla ou Perl), et est à la fois robuste et rapide.
Important : La licence d'expat est très permissive, et permet son utilisation gratuite dans n'importe quel type de projet (commercial ou non) à partir du moment ou le texte de la licence est fourni avec l'application lors de sa distribution. Donc si vous utilisez cette bibliothèque, vous devez inclure un fichier contenant la licence mentionnée ci-dessus.
Les fonctions d'Expat peuvent être utilisées directement dans PureBasic comme les autres fonctions API. Cependant vous devrez ajouter "pb_" au début des noms de fonctions en plus du caractère soulignement (_) à la fin. Cela permet de profiter des fonctionnalités d'expat qui ne sont pas directement fournies par cette bibliothèque. Les constantes et les structures définies dans expat.h sont aussi directement disponibles. Pour plus de détails sur les fonctions d'expat, consultez la documentation sur http ://expat.sourceforge.net/
La bibliothèque supporte partiellement les 'Document Type Definitions' DTD et les espaces de noms. Le but est de garder l'ensemble des commandes très simple tout en permettant à cette bibliothèque de manipuler n'importe quel document XML. Le parser de la bibliothèque expat est un analyseur syntaxique ("parser") non validant, ce qui signifie qu'il vérifie si les documents parsés comportent des erreurs dans les balises (un document doit être "bien formé" selon la norme XML), mais il

ne valide pas le document par rapport à un DTD. En parsant le document cette bibliothèque place les DTDs dans un noeud spécifique de l'arbre XML en lui attribuant le type #PB_XML_DTD. Ce noeud contient la balise DOCTYPE complète. De cette manière on peut facilement y accèder et en manipuler le contenu mais on peut aussi l'ignorer si ces informations ne sont pas nécessaires. La balise est simplement recopiée quand le document est exporté ou sauvegardé.
Les espaces de noms ne sont pas résolus lors de l'analyse syntaxique du document ("parsing"). Cela signifie que dans un document utilisant des espaces de noms, ces espaces de noms sont accessibles comme des attributs de noeud normaux, et les noms des noeuds/attributs utilisant les espaces de noms seront visibles en tant que : "namespace :tagname" (i.e "espace_de_noms :balise"). Cela permet à un document utilisant les espaces de noms d'être lu et sauvegardé comme n'importe quel autre document sans détruire sa structure. Pour fonctionner avec des espaces de noms plus simples, les fonctions ResolveXMLNodeName() et ResolveXMLAttributeName() sont fournies pour résoudre les conflits éventuels de noms à l'intérieur des documents qui utilisent les espaces de noms.
Le débogueur PureBasic permet d'examiner les objets XML pendant l'exécution du programme à l'aide du visualisateur de bibliothèques .
La spécification officielle du XML et des Namespaces XML par le W3C est disponible ici :
Spécification XML
Namespaces XML
Traductions de divers documents relatifs au XML
XML sur Wikipedia fournit aussi une bonne introduction au XML.

OS Supportés

Tous

159.1 IsXML

Syntaxe

```
Resultat = IsXML(#XML)
```

Description

Teste si un arbre XML est valide et correctement initialisé.

Arguments

#**XML** L'arbre XML à utiliser.

Valeur de retour

Renvoie une valeur non nulle si l'arbre est valide, zéro sinon.

Remarques

Cette fonction a été créée pour pouvoir passer n'importe quelle valeur en paramètre sans qu'il ne puisse y avoir de plantage. C'est la fonction idéale pour vérifier qu'un arbre est correctement initialisé.

Voir aussi

LoadXML() , CreateXML()

OS Supportés

Tous

159.2 FreeXML

Syntaxe

```
FreeXML(#XML)
```

Description

Supprime l'arbre XML et toutes les données qu'il contenait.

Arguments

#**XML** L'arbre XML à supprimer.
Si #PB_All est spécifié, tous les objets XML restants sont libérés.

Valeur de retour

Aucune.

Remarques

Tous les objets XML restants sont automatiquement supprimés quand le programme se termine.

OS Supportés

Tous

159.3 CreateXML

Syntaxe

```
Resultat = CreateXML(#XML [,
    Encodage])
```

Description

Crée un nouvel arbre XML vide.

Arguments

#XML L'arbre XML à utiliser. #PB_Any peut être utilisé pour générer automatiquement ce numéro.

Encodage (optionnel) L'encodage de l'arbre XML.

```
#PB_UTF8 (valeur par
 défaut)
#PB_Ascii
#PB_Unicode
```

Valeur de retour

Renvoie une valeur non nulle si l'arbre a été créé correctement, zéro sinon.

Remarques

Le nouvel arbre ne contient qu'un noeud racine, accessible avec RootXMLNode() . Pour ajouter de nouveaux noeuds, utiliser CreateXMLNode() .

Exemple

```
1  ; Création d'un arbre xml
2  xml = CreateXML(#PB_Any)
3  mainNode =
    CreateXMLNode(RootXMLNode(xml),
    "Zoo")
4  SetXMLNodeName(mainNode,
    "Zoo")
5
6  ; Création du premier noeud
7  item =
    CreateXMLNode(mainNode,
    "Animal")
8  SetXMLNodeName(item,
    "Animal")
9  SetXMLAttribute(item, "id",
    "1")
```

```
10  SetXMLNodeText(item,
      "Elephant")
11
12  ; Création du second noeud
13  item =
      CreateXMLNode(mainNode,
      "Animal")
14  SetXMLNodeName(item,
      "Animal")
15  SetXMLAttribute(item, "id",
      "2")
16  SetXMLNodeText(item,
      "Tigre")
17
18  ; Enregistre l'arbre XML
      dans le fichier spécifié.
19  SaveXML(xml, "demo.xml")
```

Voir aussi

FreeXML() , CreateXMLNode() , RootXMLNode()

OS Supportés

Tous

159.4 LoadXML

Syntaxe

```
Resultat = LoadXML(#XML,
    Fichier$ [, Encodage]))
```

Description

Charge un arbre XML depuis un fichier.

Arguments

#XML L'arbre XML à utiliser. #PB_Any peut être utilisé pour générer automatiquement ce numéro.

Fichier$ Le nom du fichier contenant le code XML.

Encodage (optionnel) Force l'encodage de l'arbre XML quelque soit celui trouvé dans le fichier.

```
#PB_UTF8 (valeur par
  défaut)
#PB_Ascii
#PB_Unicode
```

Ce paramètre devra être utilisé lorsque le document n'a pas de déclaration XML, ou si l'information d'encodage est fournie en dehors du document XML, par

exemple à travers un en-tête de type ”mime” d'un protocole de communication.

Valeur de retour

Renvoie une valeur non nulle si le fichier a pu être ouvert et lu correctement.
Cela ne veut pas dire que le XML contenu dans ce fichier était valide. Pour vérifier les erreurs d'interprétation, il convient d'utiliser XMLStatus() .
Dans le cas d'une erreur d'interprétation, toutes les données correctement lues avant l'erreur sont quand même disponibles dans l'arbre XML.
Si #PB_Any a été utilisé comme paramètre alors le numéro du nouvel arbre est renvoyé dans 'Resultat'.

Voir aussi

CreateXML() , FreeXML() , ParseXML() , CatchXML()

OS Supportés

Tous

159.5 CatchXML

Syntaxe

```
Resultat = CatchXML(#XML,
    *Adresse, Taille [,
    Options [, Encodage]])
```

Description

Crée un arbre XML à partir d'un code XML déjà chargé en mémoire.
La création de l'arbre XML peut se faire en une seule fois, ou de manière progressive par exemple lors de la réception d'un flux XML par le réseau .

Arguments

#XML L'arbre XML à utiliser. #PB_Any peut être utilisé pour générer automatiquement ce numéro.

***Adresse** L'adresse d'une zone mémoire valide.

Taille La taille en octets de la zone mémoire.

Options (optionnel) Si ce paramètre est omis, la zone mémoire doit être suffisamment grande pour contenir toutes les données XML.

Pour interpréter un flux XML en plusieurs fois, les options suivantes sont disponibles :

```
#PB_XML_StreamStart:
 Démarre l'interprétation
 du premier bloc
#PB_XML_StreamNext :
 Continue
 l'interprétation avec un
 nouveau bloc
#PB_XML_StreamEnd  :
 Termine l'interprétation
 après ce bloc
```

Lors de l'appel de cette fonction avec l'option #PB_XML_StreamStart ou #PB_XML_StreamEnd, le paramètre 'Taille' peut être égal à 0.
A noter que lors de l'interprétation par blocs, toute donnée fournie est immédiatement accessible dans l'arbre XML avant même que l'appel avec #PB_XML_StreamEnd n'ait eu lieu.

Encodage (optionnel) Force l'encodage de l'arbre XML quelque soit celui trouvé dans la déclaration XML. (la déclaration XML est modifiée selon l'encodage choisi).

```
#PB_UTF8 (Par défaut)
#PB_Ascii
#PB_Unicode
```

Ce paramètre devra être utilisé lorsque le document n'a pas de déclaration XML, ou si l'information d'encodage est fournie en dehors du document XML, par exemple à travers un en-tête de type "mime" d'un protocole de communication.

Valeur de retour

Cette commande renvoie 0 seulement lors d'options invalides ou si une zone de mémoire est corrompue.
Pour vérifier les erreurs d'interprétation, utiliser XMLStatus() . Dans le cas d'une erreur d'interprétation, toutes les données correctement lues avant l'erreur sont quand même disponibles dans l'arbre XML.

Voir aussi

FreeXML() , CreateXML() , LoadXML() , ParseXML()

OS Supportés

Tous

159.6 ParseXML

Syntaxe

```
Resultat = ParseXML(#XML,
    Chaîne$)
```

Description

Crée un nouvel arbre XML à partir de données XML d'une chaîne de caractères.

Arguments

#XML L'arbre XML à utiliser. #PB_Any peut être utilisé pour générer automatiquement ce numéro.

Chaîne$ La chaîne de caractères contenant le code XML.

Valeur de retour

Renvoie une valeur non nulle en cas de succès, zéro pour les erreurs de mémoire. Pour vérifier les erreurs d'analyse XMLStatus() doit être utilisé.
En cas d'une erreur d'analyse, toutes les données déjà analysées sont quand même accessibles dans l'arborescence XML.

Remarques

Le XML est encodé dans le format des chaînes de l'exécutable (ASCII ou Unicode). Si un autre encodage doit être utilisé, la fonction CatchXML() peut être utilisée à la place.

Voir aussi

FreeXML() , CreateXML() , LoadXML()

OS Supportés

Tous

159.7 XMLStatus

Syntaxe

```
Resultat = XMLStatus(#XML)
```

Description

Renvoie le statut de la dernière interprétation faite sur un arbre XML avec LoadXML() ou CatchXML() .

Arguments

#**XML** L'arbre XML à utiliser.

Valeur de retour

Une valeur nulle (#PB_XML_Success) indique une interprétation réussie, toutes les autres valeurs indiquent un type d'erreurs.

```
#PB_XML_Success
                    : Aucune
 erreur
#PB_XML_NoMemory
                   : Plus
 assez de mémoire
#PB_XML_Syntax
                     :
 Erreur de syntaxe
#PB_XML_NoElements
                  : Aucun
 élément trouvé
#PB_XML_InvalidToken
                : Document
 malformé (balise invalide)
#PB_XML_UnclosedToken
              : Balise
 fermante manquante
#PB_XML_PartialCharacter
           : Caractère
 partiel
#PB_XML_TagMismatch
                : Balise qui
 ne correspond pas
#PB_XML_DublicateAttribute
          : Attribut en double
#PB_XML_JunkAfterDocElement
         : Erreur après un
 élément du document
#PB_XML_ParamEntityRef
             : Paramètre
 illégal dans la référence
 de l'entité
#PB_XML_UndefinedEntity
            : Entité
 indéfinie
#PB_XML_RecursiveEntityRef
          : Référence
 récursive d'entités
#PB_XML_AsyncEntity
                : Entité
 asynchrone
#PB_XML_BadCharacterRef
            : Référence à un
 numéro de caractère
 invalide
#PB_XML_BinaryEntityRef
            : Référence à
 une entité binaire
```

```
#PB_XML_AttributeExternalEntityRef:
 Référence à une entité
 externe dans un attribut
#PB_XML_MisplacedXML
              : XML ou
 texte de déclaration mal
 placé
#PB_XML_UnknownEncoding
           : Encodage
 inconnu
#PB_XML_IncorrectEncoding
          : L'encodage de la
 déclaration XML est
 incorrect
#PB_XML_UnclosedCDataSection
       : Section CDATA non
 fermée
#PB_XML_ExternalEntityHandling
     : Erreur dans
 l'interprétation d'une
 entité externe
#PB_XML_NotStandalone
              : Le document
 n'est pas autonome
#PB_XML_UnexpectedState
           : Erreur
 d'interprétation inattendue
#PB_XML_EntityDeclaredInPE
         : Entité déclarée
 dans le paramètre entité
#PB_XML_FeatureRequiresDTD
         : La fonctionnalité
 requise nécessite le
 support XML_DTD d'Expat
#PB_XML_CantChangeFeatures
         : Impossible de
 changer de paramètre
 pendant l'interprétation
#PB_XML_UnboundPrefix
              : Préfixe non
 lié
#PB_XML_UndeclaringPrefix
          : Ne pas enlever
 la déclaration du préfix
#PB_XML_IncompletePE
              : Balise
 incomplète dans le
 paramètre entité
#PB_XML_XMLDeclaration
            : XML
 déclaration malformée
#PB_XML_TextDeclaration
           : Texte de
 déclaration malformé
#PB_XML_PublicID
                  :
 Caractère illégal dans un
 identifiant public
```

```
#PB_XML_Suspended
                  :
 Interpréteur suspendu
#PB_XML_NotSuspended
               :
 Interpréteur non suspendu
#PB_XML_Aborted
                    :
 Interprétation annulée
#PB_XML_Finished
                   :
 Interprétation finie
#PB_XML_SuspendedPE
                : Suspension
 impossible dans un
 paramètre entité externe
#PB_XML_ReservedPrefixXML
          : Le préfixe
 réservé (xml) ne doit pas
 être délié ou lié à un
 autre espace de noms
#PB_XML_ReservedPrefixXMLNS
        : Le préfixe réservé
 (xmlns) ne doit pas être
 déclaré ou enlevé
#PB_XML_ReservedNamespaceURI
       : Un prefixe ne doit
 pas empiéter sur un espace
 de noms réservé
```

Remarques

Cette commande devrait être utilisée après chaque appel de LoadXML() ou CatchXML() pour s'assurer que tout s'est correctement déroulé. Le message de l'erreur sous forme textuelle (en anglais) est disponible grâce à la commande XMLError() .

OS Supportés

Tous

159.8 XMLError

Syntaxe

```
Resultat\$ = XMLError(#XML)
```

Description

Quand une erreur d'interprétation XML survient, cette commande renvoie un message (en anglais) décrivant succintement l'erreur. XMLStatus() peut être utilisée pour détecter si une erreur est survenue.

Arguments

#**XML** L'arbre XML à utiliser.

Valeur de retour

L'erreur sous forme de texte en anglais.

Remarques

Pour avoir plus d'informations sur l'erreur, XMLErrorLine() et XMLErrorPosition() sont disponibles.

Voir aussi

XMLErrorLine() , XMLErrorPosition() , XMLStatus()

OS Supportés

Tous

159.9 XMLErrorLine

Syntaxe

```
Resultat = XMLErrorLine(#XML)
```

Description

Quand une erreur d'interprétation XML survient, cette commande renvoie la ligne incriminée dans le document. XMLStatus() peut être utilisée pour détecter si une erreur est survenue.

Arguments

#**XML** L'arbre XML à utiliser.

Valeur de retour

Renvoie la ligne qui contient une erreur d'interprétation.
La numérotation des lignes commence à 1

Remarques

Pour savoir à quelle position dans la ligne l'erreur est apparue , utiliser XMLErrorPosition() .

Voir aussi

XMLError() , XMLErrorPosition() , XMLStatus()

OS Supportés

Tous

159.10 XMLErrorPosition

Syntaxe

```
Resultat =
    XMLErrorPosition(#XML)
```

Description

Quand une erreur d'interprétation XML survient, cette commande renvoie la position du caractère dans la ligne incriminée (voir XMLErrorLine()). XMLStatus() peut être utilisée pour détecter si une erreur est survenue.

Arguments

#XML L'arbre XML à utiliser.

Valeur de retour

La position du caractère.
La numérotation des caractères commence à 1

Voir aussi

XMLError() , XMLErrorLine() , XMLStatus()

OS Supportés

Tous

159.11 SaveXML

Syntaxe

```
Resultat = SaveXML(#XML,
    Fichier$ [, Options])
```

Description

Enregistre un arbre XML dans un fichier.

Arguments

#XML L'arbre XML à sauvegarder.

Fichier$ Le nom du fichier.

Options (optionnel) Peut être une combinaison des valeurs suivantes (avec l'opérateur '||') :

```
#PB_XML_StringFormat :
  Ecrit un BOM (Byte Order
  Mark). Voir
  WriteStringFormat()
pour plus d'informations.
#PB_XML_NoDeclaration: La
  déclaration XML n'est
  pas incluse.
```

Note : Selon les spécifications XML, la déclaration XML peut être omise seulement si le document est encodé en UTF-8 ou UTF-16 ou si l'information d'encodage est fournie en dehors du document (par l'intermédiaire d'un protocole de communication par exemple). Malgré tout, il est conseillé de conserver la déclaration dans le document.

Valeur de retour

Renvoie une valeur non nulle si l'enregistrement s'est déroulé avec succès, zéro sinon.

Remarques

Le code XML ainsi créé n'est pas reformaté. Il est écrit comme il avait été interprété lors de son chargement ou de sa création. Le nombre d'espaces ou de retour à la ligne est conservé dans le champ 'text' de chaque noeud XML (voir GetXMLNodeText() pour plus d'informations). Pour reformater le XML avant de l'enregistrer, le champ 'text' de chaque noeud XML peut être modifié manuellement ou avec FormatXML() .

Voir aussi

LoadXML() , CreateXML() , FormatXML() , ExportXML() , ComposeXML()

OS Supportés

Tous

159.12 ExportXMLSize

Syntaxe

```
Resultat = ExportXMLSize(#XML
    [, Options])
```

Description

Renvoie la taille en octets qui sera nécessaire pour exporter complètement un

arbre XML dans une zone de mémoire (Buffer). Cette commande est nécessaire à la fonction ExportXML() .

Arguments

#XML L'arbre XML à utiliser.

Options (optionnel) Peut être utilisé pour indiquer les mêmes options que dans la commande ExportXML() . Ce qui permet de tenir compte des options pour le calcul de la taille.

Valeur de retour

La taille en octets pour exporter l'arbre XML dans un buffer mémoire .

Voir aussi

ExportXML()

OS Supportés

Tous

159.13 ExportXML

Syntaxe

```
Resultat = ExportXML(#XML,
    *Adresse, Taille [,
    Options])
```

Description

Ecrit un arbre XML dans une zone mémoire (buffer).

Arguments

#XML L'arbre XML à exporter.

***Adresse** L'adresse de la zone mémoire.

Taille La taille de la zone mémoire, en octets.

Options (optionnel) Peut être une combinaison des valeurs suivantes (avec l'opérateur '|') :

```
    #PB_XML_StringFormat :
     Ecrit un BOM (Byte Order
     Mark). Voir
     WriteStringFormat()
  pour plus d'informations.
    #PB_XML_NoDeclaration: La
     déclaration XML n'est
     pas incluse.
```

Note : Selon les spécifications XML, la déclaration XML peut être omise seulement si le document est encodé en UTF-8 ou UTF-16 ou si l'information d'encodage est fournie en dehors du document (par l'intermédiaire d'un protocole de communication par exemple). Malgré tout, il est conseillé de conserver la déclaration dans le document.

Valeur de retour

Renvoie une valeur non nulle si la 'Taille' était suffisante pour contenir l'arbre XML au complet, zéro sinon.
ExportXMLSize() permet de déterminer la taille nécessaire à l'export.

Remarques

Le code XML ainsi créé n'est pas reformaté. Il est écrit comme il avait été interprété lors de son chargement ou de sa création. Le nombre d'espaces ou de retour à la ligne est conservé dans le champ 'text' de chaque noeud XML (voir GetXMLNodeText() pour plus d'informations). Pour reformater le XML avant de l'enregistrer, le champ 'text' de chaque noeud XML peut être modifié manuellement ou avec FormatXML() .

Voir aussi

ExportXMLSize() , FormatXML() , ComposeXML() , SaveXML()

OS Supportés

Tous

159.14 ComposeXML

Syntaxe

```
Resultat\$ = ComposeXML(#XML
    [, Options])
```

Description

Renvoie le code XML dans une chaîne de caractères.

Arguments

#**XML** L'arbre XML à exporter.

Options (optionnel) Peut être une combinaison des valeurs suivantes (avec l'opérateur '||') :

```
#PB_XML_StringFormat : Inclut un BOM (byte order mark). Voir WriteStringFormat()
.
#PB_XML_NoDeclaration: Ne pas inclure la déclaration XML.
```

Note : Selon la spécification XML, la déclaration XML ne peut être omise si le document est encodé en UTF-8 ou UTF-16 ou si les informations de codage sont prévues à l'extérieur grâce à un protocole de transfert par exemple. Même alors, il est conseillé de conserver la déclaration dans le document.

Valeur de retour

Renvoie le balisage XML dans une chaîne de caractères.

Remarques

Le XML sera renvoyé dans le format des chaînes de l'exécutable (ASCII ou Unicode) indépendamment du réglage renvoyé par GetXMLEncoding() . La fonction ExportXML() peut être utilisée pour créer des balises dans un codage différent.
Le balisage XML créé n'est pas reformaté. Il est renvoyé comme il a été initialement analysé/créé. Les sauts de ligne/espaces entre les balises sont stockés dans le 'texte' de chaque noeud XML. Voir GetXMLNodeText() pour plus d'informations. Pour reformater le balisage XML avant d'enregistrer, le 'texte' de chaque noeud XML peut être modifié ou FormatXML() peut être utilisé pour appliquer certaines options de reformatage de l'arbre.

Voir aussi

FormatXML() , ExportXML() , SaveXML()

OS Supportés

Tous

159.15 FormatXML

Syntaxe

```
FormatXML(#XML, Options [,
    Indentation])
```

Description

Nettoie ou reformate un arbre XML pour le rendre plus lisible lors d'un export ou d'un enregistrement , ou plus compact si la taille est primordiale (avant un transfert réseau par exemple).

Arguments

#XML L'arbre XML à formater.

Options Peut être une combinaison des valeurs suivantes :

```
#PB_XML_WindowsNewline:
 Change tous les retours
 chariot en CRLF
#PB_XML_LinuxNewline  :
 Change tous les retours
 chariot en LF
#PB_XML_MacNewline    :
 Change tous les retours
 chariot en CR

#PB_XML_CutNewline    :
 Enlève tous les retours
 chariot
#PB_XML_ReduceNewline :
 Enlève toutes les lignes
 vides

#PB_XML_CutSpace      :
 Enlève tous les espaces
#PB_XML_ReduceSpace   :
 Enlève tous les espaces
 en double

#PB_XML_ReFormat      :
 Reformate complètement
 l'arbre XML
#PB_XML_ReIndent      :
 Change l'indentation des
 lignes
```

Pour #PB_XML_ReFormat et #PB_XML_ReIndent, le paramètre 'Indentation' définit le nombre d'espaces à rajouter par niveau d'indentation.
Note : Il n'y a pas de reformatage dans les sections CData et 'Processing Instructions' mis à part les retours chariots, car les espaces contenus dans ces sections peuvent être importants.
Note : Depuis MacOSX, le retour chariot 'CR' est devenu de moins en moins commun au profit du retour de ligne 'LF' utilisé sur tous les systèmes unix.

L'option #PB_XML_MacNewline est fournie pour des raisons de compatibilité, mais il est conseillé d'utiliser #PB_XML_LinuxNewline sur MacOS X.

Indentation (optionnel) L'Indentation à appliquer (en caractères) avec #PB_XML_ReFormat ou #PB_XML_ReIndent

Valeur de retour

Aucune.

Remarques

Le formatage de l'arbre XML est stocké dans les champs 'text' et 'offset' de chaque noeud (voir GetXMLNodeText() et GetXMLNodeOffset() pour plus d'informations).

Voir aussi

ExportXML() , SaveXML()

OS Supportés

Tous

159.16 GetXMLEncoding

Syntaxe

```
Resultat =
    GetXMLEncoding(#XML)
```

Description

Renvoie l'encodage du texte utilisé pour exporter ou enregistrer un arbre XML.

Arguments

#XML L'arbre XML à utiliser.

Valeur de retour

Renvoie une des valeurs suivantes :

```
#PB_Ascii
#PB_Unicode (UTF16)
#PB_UTF8
```

Voir aussi

ExportXML() , SaveXML() , SetXMLEncoding()

OS Supportés

Tous

159.17 SetXMLEncoding

Syntaxe

```
SetXMLEncoding(#XML, Encodage)
```

Description

Change l'encodage du texte utilisé pour exporter ou enregistrer un arbre XML.

Arguments

#**XML** L'arbre XML à utiliser.

Encodage Peut être une des valeurs suivantes :

```
#PB_Ascii
#PB_Unicode (UTF16)
#PB_UTF8
```

Ce paramètre affecte seulement l' export et l' enregistrement de l'arbre. Les données internes de l'arbre XML sont toujours stockées dans le format de 'string' de PureBasic (unicode). Donc l'encodage peut sans problème être changé en #PB_Ascii lors de l'enregistrement et effectuer d'autres changements sans craindre une altération de ses données.

Valeur de retour

Aucune.

Voir aussi

ExportXML() , SaveXML() , GetXMLEncoding()

OS Supportés

Tous

159.18 GetXMLStandalone

Syntaxe

```
Resultat =
    GetXMLStandalone(#XML)
```

Description

Renvoie la valeur de l'attribut "autonome" (standalone) de la déclaration XML du document.

Arguments

#**XML** L'arbre XML à utiliser.

Valeur de retour

Renvoie une des valeurs suivantes :

```
#PB_XML_StandaloneYes   : Le
 mode du document est
 autonome.
#PB_XML_StandaloneNo    : Le
 mode du document n'est pas
 autonome.
#PB_XML_StandaloneUnset: Le
 mode autonome n'est pas
 spécifié dans la
 déclaration.
```

Voir aussi

SetXMLStandalone()

OS Supportés

Tous

159.19 SetXMLStandalone

Syntaxe

```
SetXMLStandalone(#XML,
    Autonome)
```

Description

Change l'attribut "autonome" (standalone) de la déclaration XML lors de l'export ou de l'enregistrement du document.

Arguments

#**XML** L'arbre XML à utiliser.

Autonome Peut être une des valeurs suivantes :

```
#PB_XML_StandaloneYes   :
 Le mode du document est
 autonome.
#PB_XML_StandaloneNo    :
 Le mode du document
 n'est pas autonome.
#PB_XML_StandaloneUnset:
 Le mode autonome n'est
 pas spécifié dans la
 déclaration.
```

Note : Tant que la bibliothèque ne valide pas les définitions de type de document (DTDs), la valeur de cet attribut n'a aucun effet sur l'analyse syntaxique / l'enregistrement des documents avec cette bibliothèque excepté qu'ils sont lus depuis la déclaration XML ou écrits dans cette déclaration. Cette valeur est cependant importante quand on travaille avec des documents XML qui sont destinés à un analyseur syntaxique validant, c'est la raison pour laquelle cette commande existe.

Voir aussi

GetXMLStandalone()

OS Supportés

Tous

159.20 RootXMLNode

Syntaxe

```
Resultat = RootXMLNode(#XML)
```

Description

Renvoie le noeud racine.

Arguments

#XML L'arbre XML à utiliser.

Valeur de retour

Renvoie toujours un noeud XML valide si l'arbre #XML est correctement initialisé.

Remarques

Ce noeud est toujours présent. Il représente le document XML au complet. Le texte de ce noeud contient les espaces en dehors de tout noeud XML (il ne peut pas y avoir de texte en dehors des noeuds). Les fils de ce noeud sont le noeud principal et tous les commentaires situés en dehors du noeud principal. Le type de ce noeud est #PB_Xml_Root.

Voir aussi

XMLNodeType() , MainXMLNode()

OS Supportés

Tous

159.21 MainXMLNode

Syntaxe

```
Resultat = MainXMLNode(#XML)
```

Description

Renvoie le noeud principal.

Arguments

#XML L'arbre XML à utiliser.

Valeur de retour

Renvoie le noeud principal, ou zéro si l'arbre XML n'a pas de noeud principal (ce qui peut arriver si l'arbre est vide ou que le noeud principal a été effacé).

Remarques

Un arbre XML valide doit toujours avoir un noeud principal qui contient tous les autres noeuds. En dehors de ce noeud, il peut seulement y avoir des commentaires. Le type de ce noeud est #PB_Xml_Normal.

Voir aussi

XMLNodeType() , RootXMLNode()

OS Supportés

Tous

159.22 ChildXMLNode

Syntaxe

```
Resultat = ChildXMLNode(Noeud
    [, Index])
```

Description

Renvoie un noeud fils.

Arguments

Noeud Le noeud XML qui contient le noeud fils.

Index (optionnel) L'index du noeud fils. Le premier index commence à 1. S'il est omis alors le premier noeud est renvoyé.

Valeur de retour

Renvoie le noeud fils indiqué, ou zéro s'il n'existe pas.

Voir aussi

ParentXMLNode()

OS Supportés

Tous

159.23 ParentXMLNode

Syntaxe

```
Resultat =
    ParentXMLNode(Noeud)
```

Description

Renvoie le noeud parent.

Arguments

Noeud Le noeud XML qui contient le noeud parent.

Valeur de retour

Renvoie le noeud parent, ou zéro si 'Noeud' est le noeud racine.

Remarques

Chaque noeud XML a un parent, sauf le noeud racine .

Voir aussi

ChildXMLNode() , RootXMLNode()

OS Supportés

Tous

159.24 XMLChildCount

Syntaxe

```
Resultat =
    XMLChildCount(Noeud)
```

Description

Renvoie le nombre de noeuds fils.

Arguments

Noeud Le noeud XML à utiliser.

Valeur de retour

Le nombre de noeuds fils du noeud considéré.

OS Supportés

Tous

159.25 NextXMLNode

Syntaxe

```
Resultat = NextXMLNode(Noeud)
```

Description

Renvoie le noeud suivant de même niveau.

Arguments

Noeud Le noeud XML à utiliser.

Valeur de retour

Renvoie le noeud suivant, ou zéro s'il n'y a plus de noeud après celui-ci.

Voir aussi

PreviousXMLNode()

OS Supportés

Tous

159.26 PreviousXMLNode

Syntaxe

```
Resultat =
    PreviousXMLNode(Noeud)
```

Description

Renvoie le noeud précédent de même niveau.

Arguments

Noeud Le noeud XML à utiliser.

Valeur de retour

Renvoie le noeud précédent, ou zéro s'il n'y a plus de noeud avant celui-ci.

Voir aussi

NextXMLNode()

OS Supportés

Tous

159.27 XMLNodeFromPath

Syntaxe

```
Resultat =
    XMLNodeFromPath(NoeudParent,
    Chemin$)
```

Description

Renvoie le noeud contenu dans un noeud parent, dans un 'Chemin$'.

Arguments

NoeudParent Le noeud parent à utiliser.

Chemin$ Contient la liste des noeuds séparée par '/' indiquant le chemin à suivre pour aller du noeud parent au noeud cible.
Par exemple "voitures/cabriolet" représente le premier noeud avec le nom "cabriolet" dans le noeud nommé "voitures" du 'NoeudParent'.
Le nom d'un noeud peut avoir un index (commençant à 1) pour spécifier lequel des noeuds fils portant le même nom doit être sélectionné. "voitures/cabriolet[3]" représente le 3ème "cabriolet" dans "voitures".
Autres règles :
- Si un chemin commence avec '/', il est relatif à la racine de l'arbre XML. Le paramètre 'NoeudParent' est ignoré.
- Un motif "*" est utilisable à la place d'un nom de noeud pour indiquer que n'importe quel noeud est valide.
- Un noeud de commentaire porte le nom "#comment"
- Un noeud CData porte le nom "#cdata"
- Un noeud de DTD porte le nom "#dtd"
- Un noeud traitement d'instruction porte le nom "#instruction"
Quelques exemples de chemins :

```
"/mainnode/#comment[4]" -
 le 4ème commentaire dans
 le noeud "mainnode" du
 noeud racine.
"*[10]"                 -
 le 10ème noeud (de
 n'importe quel type) du
 'NoeudParent'
"*/*/*"                 -
 le premier noeud 3
 niveau en dessous du
 'NoeudParent' en ne
 tenant pas compte de
 leurs types
"node[3]/*[3]/#cdata"   -
 la première section
 CData dans le 3ème noeud
 de n'importe quel type
 dans le 3ème noeud
 "node" du 'NoeudParent'
```

Note : Cette commande n'est pas mise en uvre de la spécification XPath. La syntaxe utilisée et comprise par cette commande est seulement un petit sous ensemble de XPath. Cela signifie un chemin de retour de XMLNodePath() est une requête XPath valide, mais cette commande ne comprend que la syntaxe décrite ici, pas n'importe quelle requête XPath.

Valeur de retour

Renvoie le noeud cible, ou zéro si le chemin ne représente pas un noeud valide.

Remarques

XMLNodePath() peut être utilisé pour obtenir un tel chemin pour un noeud donné.

Voir aussi

XMLNodePath()

OS Supportés

Tous

159.28 XMLNodeFromID

Syntaxe

```
Resultat =
    XMLNodeFromID(#XML, ID$)
```

Description

Renvoie le noeud correspondant à son attribut ID.

Arguments

#XML L'arbre XML à utiliser.

ID$ L'identifiant (ID) à rechercher.

Valeur de retour

Renvoie le noeud correspondant à l'ID spécifié, ou zéro si aucun noeud n'a été trouvé.

Remarques

Dans un arbre XML valide, si un noeud a un attribut nommé "ID", la valeur de cet attribut doit être unique dans tout le document.

OS Supportés

Tous

159.29 XMLNodeType

Syntaxe

```
Resultat = XMLNodeType(Noeud)
```

Description

Renvoie le type d'un noeud.

Arguments

Noeud Le noeud à utiliser.

Valeur de retour

Les différentes valeurs possibles sont :

#PB_XML_Root
: C'est le noeud racine de l'arbre. Il représente le document lui-même. Ce noeud ne peut pas être ajouté ou supprimé. A l'intérieur de ce noeud racine, il ne peut seulement y avoir qu'un seul noeud du type #PB_XML_Normal et, de plus, aucun texte. (Ceci est indispensable pour obtenir un document XML bien formé).

#PB_XML_Normal

C'est un noeud normal dans l'arborescence. Il peut avoir des attributs et contenir du texte ou / et des fils.
Exemple : <attribut noeud="bonjour"> texte contenu </node>

#PB_XML_Comment

Ce noeud contient un commentaire. Il ne peut avoir ni fils ni attribut, seulement du texte.
Exemple : < !– texte du commentaire –>

#PB_XML_CData

C'est une section CData. Une section CData contient seulement du texte. Son contenu n'est pas interprété par l'analyseur lexical elle peut donc contenir des caractères spéciaux non décorés comme "<" ou ">" par exemple. Les sections CData peuvent être utilisées pour inclure d'autres balises ou codes à l'intérieur d'un document sans avoir à décorer tous les caractères spéciaux qui pourraient être interprétés comme étant du XML.
Exemple : < ![CDATA[contenu de cdata]]>

#PB_XML_DTD

C'est une déclaration de type de document (DTD). Cette bibliothèque n'utilise pas un analyseur lexical validant, ainsi ces déclarations sont en réalité ignorées quand le document est analysé. Afin de pouvoir les sauvegarder correctement elles sont insérées à l'intérieur de ce type de noeud DTD. Elles peuvent être lues et modifiées grâce à des commandes telles que SetXMLNodeText() et seront sauvegardées en l'état dans le document lors de l'export / enregistrement. La commande SetXMLStandalone() peut également être très utile pour travailler avec les DTDs.
Exemple : < !DOCTYPE name SYSTEM "external dtd uri">

#PB_XML_Instruction

Ce noeud représente une instruction (un ordre). Les instructions contiennent des informations destinées à être interprétées / exécutées par l'application cible. Elles ont un nom pour préciser le contenu de l'instruction et les données de l'instruction auxquelles GetXMLNodeText() doit pouvoir accéder.

Exemple : < ?php if (...) ... ?>
(Ici "php" est le nom du noeud, et ce qui suit le " ?>" est le texte du noeud.)

OS Supportés

Tous

159.30 GetXMLNodeText

Syntaxe

```
Resultat\$ =
    GetXMLNodeText(Noeud)
```

Description

Renvoie le texte contenu dans un noeud.

Arguments

Noeud Le noeud XML à utiliser.
Pour un noeud de type #PB_XML_Normal, c'est tout le texte et les espaces contenus dans le noeud qui ne sont pas situés dans des noeuds fils.
Pour le noeud racine , c'est tout les espaces situés en dehors du noeud principal (il ne peut pas y avoir de texte en dehors du noeud principal).
Pour les noeuds #PB_XML_Comment ou #PB_XML_CData, c'est tout le texte du noeud.

Valeur de retour

Le texte à l'intérieur du noeud XML.

Voir aussi

SetXMLNodeText()

OS Supportés

Tous

159.31 SetXMLNodeText

Syntaxe

```
SetXMLNodeText(Noeud, Texte$)
```

Description

Change le texte contenu dans un noeud.

Arguments

Noeud Le noeud à utiliser.

Texte$ Le nouveau texte à attribuer au noeud.

Valeur de retour

Aucune.

Remarques

Voir GetXMLNodeText() pour plus d'informations.
Si le noeud contient des fils, changer son contenu peut aussi demander un ajustement du décalage des noeuds fils.

Voir aussi

GetXMLNodeText()

OS Supportés

Tous

159.32 GetXMLNodeOffset

Syntaxe

```
Resultat =
    GetXMLNodeOffset(Noeud)
```

Description

Renvoie le décalage en caractères d'un noeud par rapport à son parent.

Arguments

Noeud Le noeud à utiliser.

Valeur de retour

La valeur renvoyée représente le nombre de caractères dans le texte du noeud parent qui lie ce noeud et le fils précédent. Donc si ce noeud suit directement le noeud précédent, cette valeur sera zéro.

Voir aussi

SetXMLNodeOffset()

OS Supportés

Tous

159.33 SetXMLNodeOffset

Syntaxe

```
SetXMLNodeOffset(Noeud,
    Decalage)
```

Description

Change le décalage en caractères d'un noeud par rapport au texte de son parent.

Arguments

Noeud Le noeud à utiliser.

Decalage Le nouveau décalage, en caractères.

Valeur de retour

Aucune.

Remarques

Voir GetXMLNodeOffset() pour plus d'informations.

Voir aussi

GetXMLNodeOffset()

OS Supportés

Tous

159.34 GetXMLNodeName

Syntaxe

```
Resultat\$ =
    GetXMLNodeName(Noeud)
```

Description

Renvoie le nom d'un noeud.

Arguments

Noeud Le noeud à utiliser. ReturnValue
Le nom 'tagname' du noeud.
Si le noeud n'est pas du type #PB_XML_Normal ou #PB_XML_Instruction alors une chaîne vide sera renvoyée.

Voir aussi

SetXMLNodeName()

OS Supportés

Tous

159.35 SetXMLNodeName

Syntaxe

```
SetXMLNodeName(Noeud, Nom$)
```

Description

Change le nom d'un noeud.

Arguments

Noeud Le noeud à utiliser.

Nom$ Le nouveau nom (tag-name).

Valeur de retour

Aucune.

Remarques

Si le noeud n'est pas du type #PB_XML_Normal ou #PB_XML_Instruction alors cette fonction n'aura aucun effet.
Pour rappel, d'après le standard XML, le nom d'un noeud doit suivre les règles suivantes :
- Sensible à la casse
- Doit commencer par une lettre ou un soulignement '_'
- Ne doit pas commencer par XML et ses déclinaison (Xml, xml, etc)
- Sont autorisés les lettres, chiffres, tiret '-' ou point '.' mais pas '! " # $ % & ' ( ) * + , / ; < = > ? @ [\ ] ^ ` { | } * * ' :
- Les espaces sont interdits

Voir aussi

GetXMLNodeName()

OS Supportés

Tous

159.36 XMLNodePath

Syntaxe

```
Resultat\$ =
    XMLNodePath(Noeud [,
    NoeudParent])
```

Description

Renvoie la réprésentation textuelle de la relation entre un noeud et son noeud parent.

Arguments

Noeud Le noeud à utiliser.

NoeudParent (optionnel) Doit être un parent ou un grandparent du noeud. si omis, le noeud racine est utilisé.

Valeur de retour

Une chaîne de caractères réprésentant la relation entre le 'Noeud' et son 'NoeudParent'.
Voir XMLNodeFromPath() pour une description sur le format du résultat renvoyé.

Voir aussi

XMLNodeFromPath()

OS Supportés

Tous

159.37 GetXMLAttribute

Syntaxe

```
Resultat\$ =
    GetXMLAttribute(Noeud,
    Attribut$)
```

Description

Renvoie la valeur d'un attribut d'un noeud.

Arguments

Noeud Le noeud à utiliser.

Attribut$ Le nom de l'attribut.

Valeur de retour

La valeur d'un attribut du Noeud.
Si l'attribut n'existe pas, une chaîne vide sera renvoyée.

Remarques

Seuls les noeuds du type #PB_XML_Normal peuvent avoir des attributs. Pour tous les autres types de noeuds, cette fonction renvoie une chaîne vide.

Voir aussi

SetXMLAttribute() ,
RemoveXMLAttribute()

OS Supportés

Tous

159.38 SetXMLAttribute

Syntaxe

```
SetXMLAttribute(Noeud,
    Attribut$, Valeur$)
```

Description

Change la valeur d'un attribut d'un noeud.

Arguments

Noeud Le noeud à utiliser.
Attribut$ Le nom de l'attribut à changer ou à créer.
Valeur$ La valeur à attribuer.

Valeur de retour

Aucune.

Remarques

Si l'attribut n'existe pas encore, il sera ajouté.
Seuls les noeuds du type #PB_XML_Normal peuvent avoir des attributs. Pour tous les autres types de noeuds, cette fonction renvoie une chaîne vide.

Voir aussi

GetXMLAttribute() ,
RemoveXMLAttribute()

OS Supportés

Tous

159.39 RemoveXMLAttribute

Syntaxe

```
RemoveXMLAttribute(Noeud,
    Attribut$)
```

Description

Supprime l'attribut d'un noeud.

Arguments

Noeud Le noeud à utiliser.

Attribut$ L'attribut à supprimer. Si l'attribut n'existe pas, il ne se passera rien.

Valeur de retour

Aucune.

Remarques

Seuls les noeuds du type #PB_XML_Normal peuvent avoir des attributs. Pour tous les autres types de noeuds, cette fonction sera sans effet.

Voir aussi

GetXMLAttribute() , SetXMLAttribute()

OS Supportés

Tous

159.40 ExamineXMLAttributes

Syntaxe

```
Resultat =
    ExamineXMLAttributes(Noeud)
```

Description

Démarre l'énumération des attributs d'un noeud.

Arguments

Noeud Le noeud à utiliser.

Valeur de retour

Renvoie une valeur non nulle si le noeud est de type #PB_XML_Normal, zéro sinon (car ces noeuds ne peuvent pas avoir d'attributs).

Voir aussi

NextXMLAttribute() , XMLAttributeName() , XMLAttributeValue()

OS Supportés

Tous

159.41 NextXMLAttribute

Syntaxe

```
Resultat =
    NextXMLAttribute(Noeud)
```

Description

Cette fonction doit être appelée après ExamineXMLAttributes() et permet de passer à l'attribut suivant.

Arguments

Noeud Le noeud à utiliser.

Valeur de retour

Renvoie une valeur non nulle ou zéro s'il n'y a plus d'attributs à énumérer.

Voir aussi

ExamineXMLAttributes() , XMLAttributeName() , XMLAttributeValue()

OS Supportés

Tous

159.42 XMLAttributeName

Syntaxe

```
Resultat\$ =
    XMLAttributeName(Noeud)
```

Description

Après avoir appelé ExamineXMLAttributes() et NextXMLAttribute() cette fonction renvoie le nom de l'attribut du noeud en cours d'examen.

Arguments

Noeud Le noeud à utiliser.

Valeur de retour

Le nom de l'attribut en cours d'examen.

Voir aussi

ExamineXMLAttributes() , NextXMLAttribute() , XMLAttributeValue()

OS Supportés

Tous

159.43 XMLAttributeValue

Syntaxe

```
Resultat\$ =
    XMLAttributeValue(Noeud)
```

Description

Après avoir appelé ExamineXMLAttributes() et NextXMLAttribute() cette fonction renvoie la valeur de l'attribut du noeud en cours d'examen.

Arguments

Noeud Le noeud à utiliser.

Valeur de retour

La valeur de l'attribut en cours d'examen.

Voir aussi

ExamineXMLAttributes() , NextXMLAttribute() , XMLAttributeName()

OS Supportés

Tous

159.44 CreateXMLNode

Syntaxe

```
Resultat =
    CreateXMLNode(NoeudParent,
    Nom$ [, NoeudPrecedent [,
    Type]])
```

Description

Crée un nouveau noeud et l'insert sous un noeud parent.

Arguments

NoeudParent Le noeud sous lequel le nouveau noeud doit être inséré. Pour insérer le noeud à la racine de l'arbre XML, utiliser RootXMLNode() .

Nom$ Le nom du noeud, c'est une chaîne de caractères.
Peut être une chaine vide si le nom du noeud n'est pas requis par le parse.

NoeudPrecedent (optionnel) Représente le noeud fils de 'NoeudParent' après lequel le nouveau noeud devra être inséré. Si sa valeur est omise ou égale à 0, le nouveau noeud est inséré en tant que premier fils du noeud parent. Si sa valeur est -1, alors le nouveau noeud est inséré comme dernier fils du noeud parent.

Type (optionnel) Spécifie le type pour ce nouveau noeud. Le type par défaut est #PB_XML_Normal. Note : le type du noeud ne peut plus être modifié une fois qu'il a été créé.

Valeur de retour

Renvoie le noeud nouvellement créé ou zéro s'il n'a pas pu être inséré à cet endroit.

Remarques

Les règles suivantes doivent être respectées pour que l'insertion se passe correctement :
- 'NoeudParent' ne peut pas être de type #PB_XML_Comment ou #PB_XML_CData
- 'NoeudPrecedent' doit être un fils direct du 'NoeudParent' (s'il est spécifié)
- Un noeud de type #PB_XML_Root ne peut pas être créé manuellement
- Si l'arbre XML a déjà un noeud principal , seuls les noeuds de type autres que #PB_XML_Normal et #PB_XML_CData peuvent être insérés à la racine

Voir aussi

DeleteXMLNode()

OS Supportés

Tous

159.45 CopyXMLNode

Syntaxe

```
Resultat = CopyXMLNode(Noeud,
    NoeudParent [,
    NoeudPrecedent])
```

Description

Duplique un noeud et tout son contenu (texte et noeuds fils) à un nouvel emplacement.
Cette fonction peut copier des noeuds de documents XML différents.
Pour déplacer un noeud à un nouvel emplacement, utiliser MoveXMLNode() .

Arguments

Noeud Le noeud à dupliquer.

NoeudParent Le noeud sous lequel le noeud copié sera inséré.
Pour insérer le noeud à la racine de l'arbre XML, utiliser RootXMLNode() .

NoeudPrecedent (optionnel) Le noeud fils de 'NoeudParent' après lequel le nouveau noeud devra être inséré. Si sa valeur est omise ou égale à 0, le nouveau noeud est inséré en tant que premier fils du noeud parent. Si sa valeur est -1 alors le nouveau noeud est inséré comme dernier fils du noeud parent.

Valeur de retour

Le nouveau noeud si la copie est un succès, zéro sinon.

Remarques

Les règles suivantes doivent être respectées pour que la copie se déroule correctement :
- Le noeud racine ne peut pas être copié
- 'NoeudParent' ne peut pas être de type #PB_XML_Comment ou #PB_XML_CData
- 'NoeudPrecedent' doit être un fils direct du 'NoeudParent' (s'il est spécifié)
- Si l'arbre XML a déjà un noeud principal , seuls les noeuds de type autres que #PB_XML_Normal et #PB_XML_CData peuvent être insérés à la racine

Voir aussi

DeleteXMLNode() , MoveXMLNode()

OS Supportés

Tous

159.46 MoveXMLNode

Syntaxe

```
Resultat = MoveXMLNode(Noeud,
    NoeudParent [,
    NoeudPrecedent])
```

Description

Déplace un noeud et tout son contenu (texte et noeuds fils) à un nouvel emplacement. Cette fonction peut déplacer des noeuds de documents XML différents.
Pour copier un noeud à un nouvel emplacement, utiliser CopyXMLNode() .

Arguments

Noeud Le noeud à déplacer.

NoeudParent Le noeud sous lequel le noeud déplacé sera inséré. Pour insérer le noeud à la racine de l'arbre XML, utiliser RootXMLNode() .

NoeudPrecedent (optionnel) Le noeud fils de 'NoeudParent' après lequel le nouveau noeud devra être inséré. Si sa valeur est omise ou égale à 0, le nouveau noeud est inséré en tant que premier fils du noeud parent. Si sa valeur est -1 alors le nouveau noeud est inséré comme dernier fils du noeud parent.

Valeur de retour

Renvoie une valeur non nulle si le déplacement est un succès, zéro sinon.

Remarques

Les règles suivantes doivent être respectées pour que le déplacement se déroule correctement :
- Le noeud racine ne peut pas être déplacé
- 'NoeudParent' ne peut pas être de type #PB_XML_Comment ou #PB_XML_CData
- 'NoeudPrecedent' doit être un fils direct du 'NoeudParent' (s'il est spécifié)
- 'Noeud' et 'NoeudPrecedent' ne peuvent pas être identiques
- 'NoeudParent' ne peut pas être égal à 'Noeud' ou à un de ses fils (un noeud ne peut pas être déplacé dans lui-même)
- Si l'arbre XML a déjà un noeud principal , seuls les noeuds de type autres que #PB_XML_Normal et #PB_XML_CData peuvent être insérés à la racine.

Voir aussi

DeleteXMLNode() , CopyXMLNode()

OS Supportés

Tous

159.47 DeleteXMLNode

Syntaxe

```
DeleteXMLNode(Noeud)
```

Description

Efface un noeud et tout son contenu (texte et fils).

Arguments

Noeud Le noeud à utiliser.

Remarques

Le noeud racine d'un document XML ne peut pas être effacé.

Voir aussi

CreateXMLNode()

OS Supportés

Tous

159.48 ResolveXMLNodeName

Syntaxe

```
Resultat\$ =
    ResolveXMLNodeName(Noeud
    [, Separateur$])
```

Description

Renvoie le nom du noeud fourni dans le document qui utilise les espaces de noms XML.
Ce nom comporte l'URI (Uniform Resource Identifier) de l'espace de noms (s'il existe) et le nom du noeud local, séparés par le caractère de séparation donné dans 'Separateur$'. Le caractère de séparation par défaut est "/".

Arguments

Noeud Le noeud à utiliser.

Separateur$ (optionnel) Le séparateur (chaîne de caractères) à utiliser.
Par défaut : "/".

Valeur de retour

Dans un document utilisant les espaces de noms, cette fonction renvoie le nom qualifié du noeud s'il a été correctement résolu ou une chaîne vide si un préfixe d'espaces de noms est utilisé mais n'a jamais été déclaré (ce qui n'est pas valide).
Dans un document sans espaces de noms, la fonction renvoie le nom du noeud.

Voir aussi

ResolveXMLAttributeName()

OS Supportés

Tous

159.49 ResolveXMLAttributeName

Syntaxe

```
Resultat\$ =
    ResolveXMLAttributeName(Noeud,
    Attribut$ [, Separateur$])
```

Description

Renvoie le nom complet du noeud donné en attribut dans un document qui utilise les espaces de noms XML.
Le nom complet se compose de l'URI (Uniform Resource Identifier) de l'espace de noms (s'il existe) et du nom de l'attribut local), séparé par le séparateur donné par 'Separateur$'. Le caractère de séparation (séparateur) par défaut est "/".

Arguments

Noeud Le noeud à utiliser.

Attribut$ L'attribut à trouver.

Separateur$ (optionnel) Le séparateur (chaîne de caractères) à utiliser.
Par défaut : "/".

Valeur de retour

Dans un document utilisant les espaces de noms, renvoie le nom complet de l'attribut s'il peut-être correctement résolu ou une chaîne vide si le préfixe de l'espace de noms est utilisé sans être jamais déclaré (ce qui n'est pas valide).
Dans un document sans espaces de noms, renvoie le nom de l'attribut.

Remarques

Note : à la différence des noms de noeuds , l'espace de noms par défaut n'est pas utilisé pour qualifier un nom qui n'a pas de préfixe lié à un espace de noms. Ainsi, les noms qualifiés sans préfixe d'espace de noms ont simplement leur nom local renvoyé.

Voir aussi

ResolveXMLNodeName()

OS Supportés

Tous

159.50 InsertXMLArray

Syntaxe

```
Resultat =
    InsertXMLArray(NoeudParent,
    Tableau() [,
    NoeudPrecedent])
```

Description

Insère un tableau dans un noeud.

Arguments

NoeudParent Le noeud dans lequel insérer le nouveau noeud.
Pour insérer le nouveau noeud à la racine de l'arbre, utiliser RootXMLNode() .

Tableau() Le tableau à insérer.

NoeudPrecedent (optionnel) Un noeud enfant du noeud parent après qui le nouveau noeud doit être inséré.
Si cette valeur est 0 ou non spécifiée, le nouveau noeud est inséré comme premier enfant de son parent.
Si cette valeur est -1, le noeud est inséré comme dernier enfant de son parent.

Valeur de retour

Le nouveau noeud XML s'il a été créé avec succès, zéro sinon.

Remarques

Les règles spécifiées dans CreateXMLNode() pour le cas où un nouveau noeud est inséré s'appliquent également à cette fonction. Le noeud inséré est nommé "array" et les éléments du tableau sont insérés sous le nom "element". Si le tableau a plusieurs dimensions, chaque élément aura des attributs indiquant la coordonnée de l'élément à l'intérieur du tableau, nommée "a", "b" et ainsi de suite. Voir ci-dessous un exemple.

Exemple

```
; Cet exemple produit l'arbre XML suivant:
;
; <array>
;   <element>rouge</element>
;   <element>vert</element>
;   <element>bleu</element>
; </array>
;
Dim Couleurs$(2)
Couleurs$(0) = "rouge"
Couleurs$(1) = "vert"
Couleurs$(2) = "bleu"

If CreateXML(0)
  InsertXMLArray(RootXMLNode(0), Couleurs$())
  FormatXML(0, #PB_XML_ReFormat)
  Debug ComposeXML(0)
EndIf
```

Exemple

```
; Cet exemple produit l'arbre XML suivant:
;
; <array>
;   <element a="0" b="0">0</element>
;   <element a="0" b="1">1</element>
;   <element a="1" b="0">10</element>
;   <element a="1" b="1">11</element>
```

```
8   ;   <element a="2"
     b="0">20</element>
9   ;   <element a="2"
     b="1">21</element>
10  ; </array>
11  ;
12  Dim MultiArray(2, 1)
13  For a = 0 To 2
14    For b = 0 To 1
15      MultiArray(a, b) = a *
     10 + b
16    Next b
17  Next a
18
19  If CreateXML(0)
20    InsertXMLArray(RootXMLNode(0),
     MultiArray())
21    FormatXML(0,
     #PB_XML_ReFormat)
22    Debug ComposeXML(0)
23  EndIf
```

Voir aussi

ExtractXMLArray() , InsertXMLList() , InsertXMLMap() , InsertXMLStructure() ,

OS Supportés

Tous

159.51 InsertXMLList

Syntaxe

```
Resultat =
    InsertXMLList(NoeudParent,
    Liste() [, NoeudPrecedent])
```

Description

Insère une liste dans un noeud.

Arguments

NoeudParent Le noeud dans lequel insérer le nouveau noeud.
Pour insérer le nouveau noeud à la racine de l'arbre, utiliser RootXMLNode() .

Liste() La liste à insérer.

NoeudPrecedent (optionnel) Un noeud enfant du noeud parent après qui le nouveau noeud doit être inséré.
Si cette valeur est 0 ou non spécifiée, le nouveau noeud est inséré comme premier enfant de son parent.
Si cette valeur est -1, le noeud est inséré comme dernier enfant de son parent.

Valeur de retour

Le nouveau noeud XML s'il a été créé avec succès, zéro sinon.

Remarques

Les règles spécifiées dans CreateXMLNode() pour le cas où un nouveau noeud est inséré s'appliquent également à cette fonction.
Le noeud inséré est nommé "list" et les éléments de la liste sont insérés sous le nom "element". Voir ci-dessous un exemple.

Exemple

```
; Cet exemple produit l'arbre XML suivant:
;
; <list>
;   <element>carré</element>
;   <element>cercle</element>
;   <element>triangle</element>
; </list>
;
NewList Formes$()
AddElement(Formes$()): Formes$() = "carré"
AddElement(Formes$()): Formes$() = "cercle"
AddElement(Formes$()): Formes$() = "triangle"

If CreateXML(0)
  InsertXMLList(RootXMLNode(0), Formes$())
  FormatXML(0, #PB_XML_ReFormat)
  Debug ComposeXML(0)
EndIf
```

Exemple

```
; Cet exemple produit l'arbre XML suivant:
;
; <list>
;   <element>
;     <x>100</x>
;     <y>200</y>
;   </element>
;   <element>
;     <x>200</x>
;     <y>400</y>
;   </element>
; </list>
```

```
13    ;
14    Structure Position
15      x.l
16      y.l
17    EndStructure
18
19    NewList Positions.Position()
20
21    For i = 1 To 2
22      AddElement(Positions())
23      Positions()\x = 100 * i
24      Positions()\y = 200 * i
25    Next i
26
27    If CreateXML(0)
28      InsertXMLList(RootXMLNode(0),
        Positions())
29      FormatXML(0,
        #PB_XML_ReFormat)
30      Debug ComposeXML(0)
31    EndIf
```

Voir aussi

ExtractXMLList() , InsertXMLArray() , InsertXMLMap() , InsertXMLStructure()

OS Supportés

Tous

159.52 InsertXMLMap

Syntaxe

```
Resultat =
    InsertXMLMap(NoeudParent,
    Map() [, NoeudPrecedent])
```

Description

Insère une map dans un noeud.

Arguments

NoeudParent Le noeud dans lequel insérer le nouveau noeud.
Pour insérer le nouveau noeud à la racine de l'arbre, utiliser RootXMLNode() .

Map() La map à insérer.

NoeudPrecedent (optionnel) Un noeud enfant du noeud parent après qui le nouveau noeud doit être inséré.
Si cette valeur est 0 ou non spécifiée, le nouveau noeud est inséré comme premier enfant de son parent.
Si cette valeur est -1, le noeud est inséré comme dernier enfant de son parent.

Valeur de retour

Le nouveau noeud XML s'il a été créé avec succès, zéro sinon.

Remarques

Les règles spécifiées dans CreateXMLNode() pour le cas où un nouveau noeud est inséré s'appliquent également à cette fonction.
Le noeud inséré est nommé "map" et les éléments de la liste sont insérés sous le nom "element". Chaque noeud d'élément aura un attribut nommé "key" contenant la clé de l'élément.

Exemple

```
; Cet exemple produit l'arbre XML suivant:
;
; <map>
;   <element key="DE">Allemagne</element>
;   <element key="US">United States</element>
;   <element key="FR">France</element>
; </map>
;
NewMap Pays.s()
Pays("DE") = "Allemagne"
Pays("FR") = "France"
Pays("US") = "United States"

If CreateXML(0)
  InsertXMLMap(RootXMLNode(0), Pays())
  FormatXML(0, #PB_XML_ReFormat)
  Debug ComposeXML(0)
EndIf
```

Voir aussi

ExtractXMLMap() , InsertXMLArray() , InsertXMLList() , InsertXMLStructure()

OS Supportés

Tous

159.53 InsertXMLStructure

Syntaxe

```
Resultat =
    InsertXMLStructure(NoeudParent,
    *Memoire, Structure [,
    NoeudPrecedent])
```

Description

Insère une structure dans un noeud.

Arguments

NoeudParent Le noeud dans lequel insérer le nouveau noeud.
Pour insérer le nouveau noeud à la racine de l'arbre, utiliser RootXMLNode() .

***Memoire** L'adresse de la structure à insérer.

Structure Le type de la structure à insérer.

NoeudPrecedent (optionnel) Un noeud enfant du noeud parent après qui le nouveau noeud doit être inséré.
Si cette valeur est 0 ou non spécifiée, le nouveau noeud est inséré comme premier enfant de son parent.
Si cette valeur est -1, le noeud est inséré comme dernier enfant de son parent.

Valeur de retour

Le nouveau noeud XML s'il a été créé avec succès, zéro sinon.

Remarques

Les règles spécifiées dans CreateXMLNode() pour le cas où un nouveau noeud est inséré s'appliquent également à cette fonction.
Le noeud inséré est nommé comme la structure. Chaque élément de la structure est ajouté sous forme d'un sous-noeud à l'intérieur du noeud de la structure. Chaque caractère '*' ou '$' est retiré du nom de l'élément de la structure. Si l'élément de la structure contient un tableau, une liste, une map ou une structure, plusieurs noeuds sont ajoutés de manière récursive.

Exemple

```
1  ; Cet exemple produit
     l'arbre XML suivant:
2  ;
3  ; <Personne>
4  ;   <Nom>John Smith</Nom>
5  ;   <Age>42</Age>
6  ;   <Livres>
```

```
;   <element>Investir pour les Nuls</element>
;   <element>Programmer pour les Nuls</element>
;  </Livres>
; </Personne>
;
Structure Personne
  Nom$
  Age.l
  List Livres.s()
EndStructure

Define P.Personne

P\Nom$ = "John Smith"
P\Age   = 42
AddElement(P\Livres()):
 P\Livres() = "Investir pour les Nuls"
AddElement(P\Livres()):
 P\Livres() = "Programmer pour les Nuls"

If CreateXML(0)
  InsertXMLStructure(RootXMLNode(0), @P, Personne)
  FormatXML(0, #PB_XML_ReFormat)
  Debug ComposeXML(0)
EndIf
```

Voir aussi

ExtractXMLStructure() , InsertXMLArray() , InsertXMLList() , InsertXMLMap()

OS Supportés

Tous

159.54 ExtractXMLArray

Syntaxe

```
ExtractXMLArray(Noeud,
    Tableau() [, Options])
```

Description

Extrait les éléments d'un noeud dans un tableau .

Arguments

Noeud Le noeud XML contenant les données à extraire.

Tableau() Le tableau à remplir avec les éléments XML. Le tableau sera redimensionné pour avoir la même taille que le nombre d'éléments. Tout contenu précédent du tableau sera perdu.

Options (optionnel) Si ce paramètre est réglé sur #PB_XML_NoCase alors la comparaison des noms de noeuds et d'attribut XML est insensible à la casse. La valeur par défaut est d'être sensible à la casse.

Valeur de retour

Aucune.

Remarques

L'extraction est réalisée de manière récursive si le tableau est structuré. Les noeuds XML doivent avoir la forme décrite dans la fonction InsertXMLArray() . Les noeuds avec des noms différents sont ignorés par l'extraction. Si le tableau comporte plus d'une dimension, il est prévu pour chaque élément d'avoir des attributs indiquant les coordonnées de l'élément nommé "a", "b" et ainsi de suite.

Exemple

```
1   Xml$ =
     "<array><element>1</element><element>10</element><element>10
2
3   If ParseXML(0, Xml$) And
     XMLStatus(0) =
     #PB_XML_Success
4     Dim MyArray(0) ; il sera
     redimensionné lors du
     prochain appel
5     ExtractXMLArray(MainXMLNode(0),
     MyArray())
6
7     For i = 0 To
     ArraySize(MyArray())
8       Debug MyArray(i)
9     Next i
10  Else
11    Debug XMLError(0)
12  EndIf
```

Voir aussi

InsertXMLArray() , ExtractXMLList() , ExtractXMLMap() , ExtractXMLStructure()

OS Supportés

Tous

159.55 ExtractXMLList

Syntaxe

```
ExtractXMLList(Noeud, Liste()
    [, Options])
```

Description

Extrait les éléments d'un noeud dans une liste .

Arguments

Noeud Le noeud XML contenant les données à extraire.

List() La liste à remplir avec les éléments XML.
La liste sera effacée avant d'extraire les éléments.

Options (optionnel) Si ce paramètre est réglé sur #PB_XML_NoCase alors la comparaison des noms de noeuds et d'attribut XML est insensible à la casse. La valeur par défaut est d'être sensible à la casse.

Valeur de retour

Aucune.

Remarques

L'extraction est réalisée de manière récursive si la liste est structurée. Les noeuds XML doivent avoir la forme décrite dans la fonction InsertXMLList() . Les noeuds avec des noms différents sont ignorés par l'extraction.

Exemple

```
1   Xml$ =
      "<list><element>1</element><element>10</element><ELEMENT>100
2
3   If ParseXML(0, Xml$) And
      XMLStatus(0) =
      #PB_XML_Success
4       NewList Values()
5       ExtractXMLList(MainXMLNode(0),
      Values(), #PB_XML_NoCase)
6
7       ForEach Values()
```

```
8         Debug Values()
9       Next
10    Else
11      Debug XMLError(0)
12    EndIf
```

Voir aussi

InsertXMLList() , ExtractXMLArray() , ExtractXMLMap() , ExtractXMLStructure()

OS Supportés

Tous

159.56 ExtractXMLMap

Syntaxe

```
ExtractXMLMap(Noeud, Map() [,
    Options])
```

Description

Extrait les éléments d'un noeud dans une map .

Arguments

Noeud Le noeud XML contenant les données à extraire.

Map() La map à remplir avec les éléments XML.
La map sera effacée avant d'extraire les éléments.

Options (optionnel) Si ce paramètre est réglé sur #PB_XML_NoCase alors la comparaison des noms de noeuds et d'attribut XML est insensible à la casse. La valeur par défaut est d'être sensible à la casse.

Valeur de retour

Aucune.

Remarques

L'extraction est réalisée de manière récursive si la map est structurée. Les noeuds XML doivent avoir la forme décrite dans la fonction InsertXMLMap() . Les noeuds avec des noms différents sont ignorés par l'extraction.

Exemple

```
Xml$ = "<map><element key="
 + Chr(34) + "La clé" +
 Chr(34) + ">La
 valeur</element></map>"

If ParseXML(0, Xml$) And
 XMLStatus(0) =
 #PB_XML_Success
  NewMap Test.s()
  ExtractXMLMap(MainXMLNode(0),
 Test())

  ForEach Test()
    Debug MapKey(Test()) +
 " -> " + Test()
  Next
Else
  Debug XMLError(0)
EndIf
```

Voir aussi

InsertXMLMap() , ExtractXMLArray() , ExtractXMLList() , ExtractXMLStructure()

OS Supportés

Tous

159.57 ExtractXMLStructure

Syntaxe

```
ExtractXMLStructure(Noeud,
   *Memoire, Structure [,
   Options])
```

Description

Extrait les éléments d'un noeud dans une structure .

Arguments

Noeud Le noeud XML contenant les données à extraire.

***Memoire** L'adresse de la mémoire de la structure.

Structure La structure à remplir avec les éléments XML.
La structure sera effacée avant d'extraire les éléments.

Options (optionnel) Si ce paramètre est réglé sur #PB_XML_NoCase alors la comparaison des noms de noeuds et d'attribut XML est insensible à la casse. La valeur par défaut est d'être sensible à la casse.

Valeur de retour

Aucune.

Remarques

Si un élément de structure ne dispose pas d'un noeud correspondant dans le fichier XML, il est laissé vide.
Les noeuds XML doivent avoir la forme décrite avec la fonction InsertXMLStructure() . A savoir, chaque noeud doit être nommé d'après un membre de la structure sans les caractères '*' ou '$'.
Si un noeud pour le même élément de structure existe en plusieurs exemplaires, le premier noeud sera utilisé.

Exemple

```
1   Structure Personne
2     Nom$
3     Age.l
4   EndStructure
5
6   Xml$ = "<Personne><Nom>John
     Smith</Nom><Age>42</Age></Personne>"
7
8   If ParseXML(0, Xml$) And
     XMLStatus(0) =
     #PB_XML_Success
9     Define P.Personne
10    ExtractXMLStructure(MainXMLNode(0),
     @P, Personne)
11
12    Debug P\Nom$
13    Debug P\Age
14  Else
15    Debug XMLError(0)
16  EndIf
```

Voir aussi

InsertXMLStructure() , ExtractXMLArray() , ExtractXMLList() , ExtractXMLMap()

OS Supportés

Tous