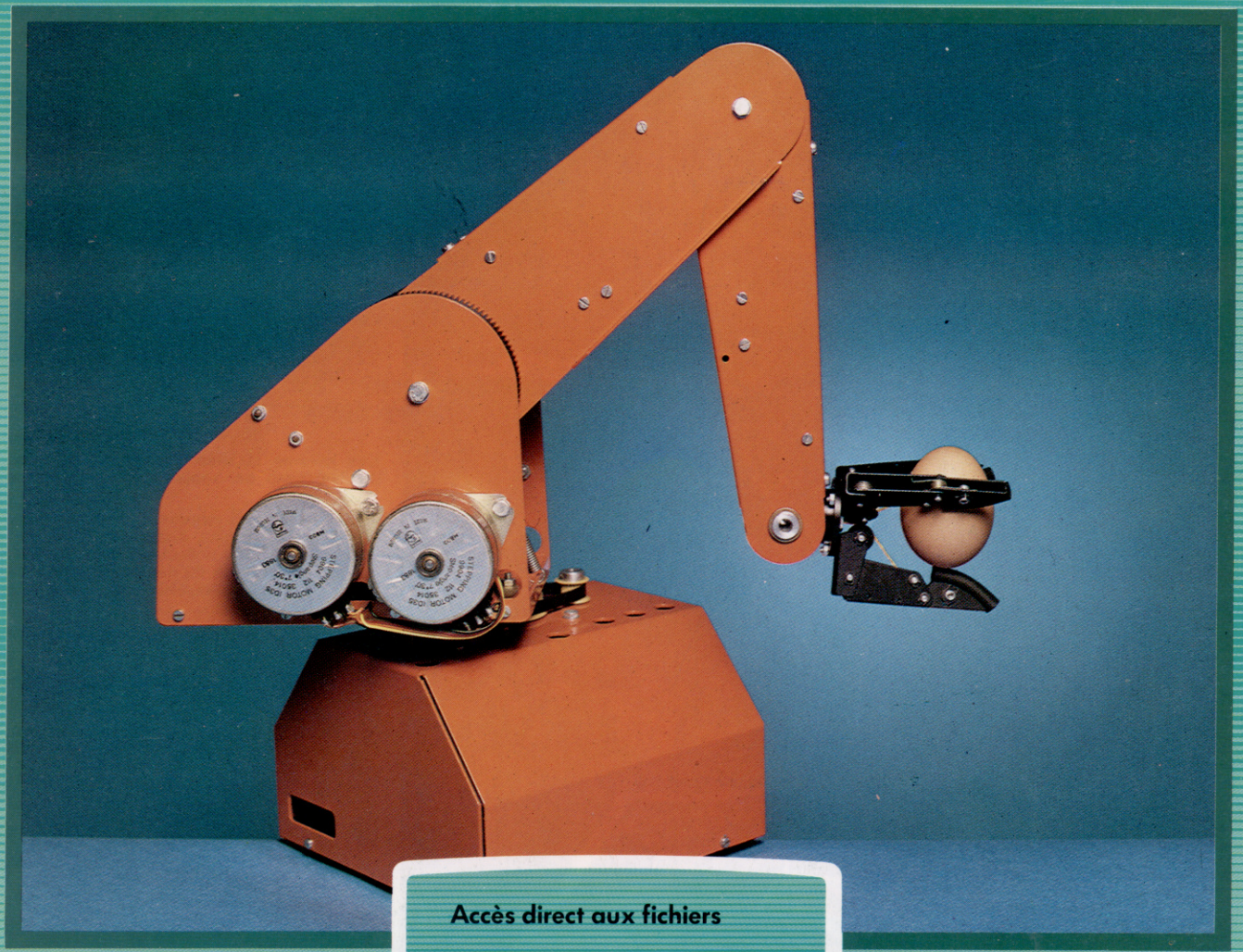
ABC INFORMATIQUE

N° 37

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE



Accès direct aux fichiers

L'Apricot en action

Le jeu des animaux

Instructions de piles

EDITIONS ATLAS

M6062-37-12F 85FB-3,80FS-$1.95

aBC INFORMATIQUE

cours d'informatique pratique et familiale

EDITIONS ATLAS

Édité par ÉDITIONS ATLAS s.a., tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*. Tél. : (37) 35-40-23. Services administratifs et commerciaux : 3, rue de la Taye, 28110 Lucé. Tél. : (37) 35-40-23.

Belgique : ÉDITIONS ATLEN s.a., Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, Montréal Nord.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, Mezzovico.

Réalisé par EDENA s.a., 29, boulevard Edgar-Quinet, 75014 Paris.

Direction éditoriale : J.-Fr. Gautier. *Service technique et artistique* : F. Givone et J.-Cl. Bernar. *Iconographie* : J. Pierre. *Correction* : B. Noël.
Publicité : Anne Cayla. Tél. : 202-09-80.

VENTE AU NUMÉRO

Les numéros parus peuvent être obtenus chez les marchands de journaux ou, à défaut, chez les éditeurs, au prix en vigueur au moment de la commande. Ils resteront en principe disponibles pendant six mois après la parution du dernier fascicule de la série. (Pour toute commande par lettre, joindre à votre courrier le règlement, majoré de 10 % de frais de port.)

Pour la France, s'adresser aux services commerciaux des ÉDITIONS ATLAS. Tél. : (37) 35-40-23.

Pour les autres pays, s'adresser aux éditeurs indiqués ci-dessous.

SOUSCRIPTION

Les lecteurs désirant souscrire à l'ensemble de cet ouvrage peuvent s'adresser à :

France : DIFFUSION ATLAS, 3, rue de la Taye, 28110 Lucé. Tél. : (37) 35-40-23.

Belgique : ÉDITIONS ATLEN s.a., 55, avenue Huart-Hamoir, 1030 Bruxelles. Tél. : (02) 242-39-00. Banque Bruxelles-Lambert, compte n° 310-0018465-24 Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, 11450 boulevard Albert-Hudon, Montréal Nord, H 1G 3J9.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, zona industriale 6849 Mezzovico-Lugano. Tél. : (091) 95-27-44.

RELIEZ VOS FASCICULES

Des reliures mobiles permettant de relier 12 fascicules sont en vente chez votre marchand de journaux.

ATTENTION : ces reliures, présentées sans numérotation, sont valables indifféremment pour tous les volumes de votre collection. Vous les numéroterez vous-même à l'aide du décalque qui est fourni (avec les instructions nécessaires) dans chaque reliure.

En vente tous les vendredis. Volume IV, n° 37.

ABC INFORMATIQUE est réalisé avec la collaboration de Trystan Mordrel *(secrétariat de rédaction)*, Jean-Pierre Bourcier *(coordination)*, Patrick Bazin, Jean-Paul Mourlon, Claire Rémy *(traduction)*, Ghislaine Goullier *(fabrication)*, Marie-Claire Jacquet *(iconographie)*, Patrick Boman *(correction)*.
Crédit photographique, couverture : Orbis Publishing.

Directeur de la publication : Paul Bernabeu. Imprimé en Italie par I.G.D.A., Officine Grafiche, Novara. Distribution en France : N.M.P.P. Tax. Dépôt légal : septembre 1984. 28849. Dépôt légal en Belgique : D/84/2783/27.

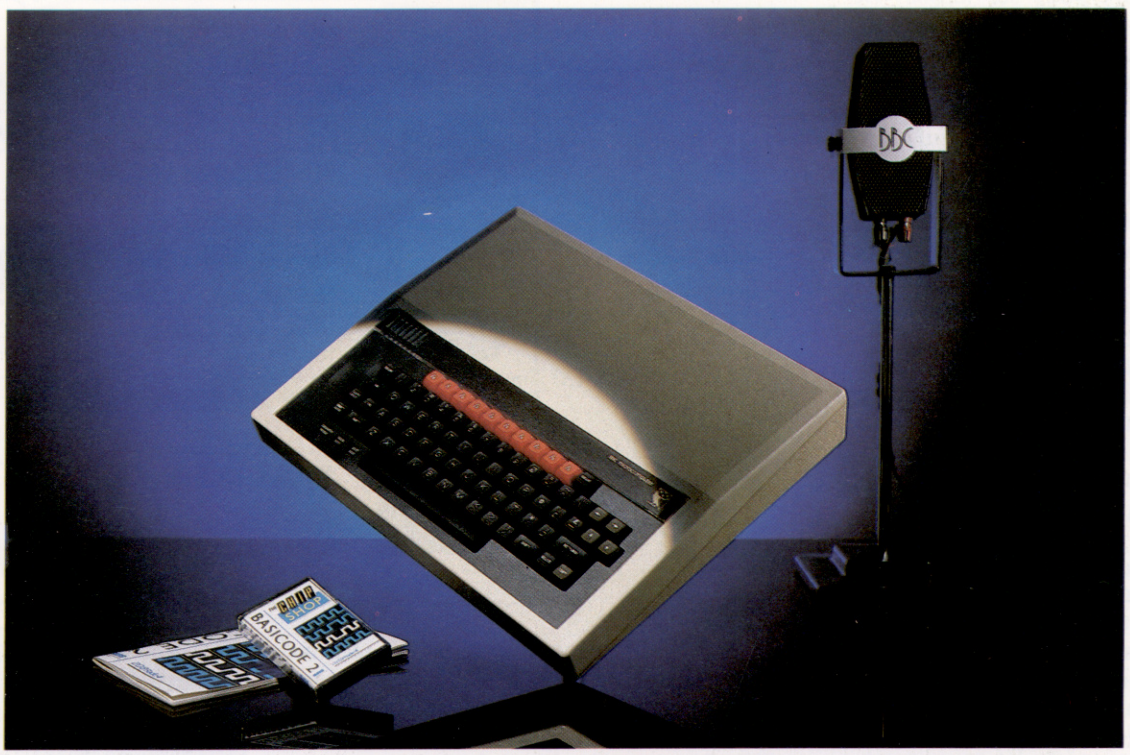
A NOS LECTEURS

En achetant chaque semaine votre fascicule chez le même marchand de journaux, vous serez certain d'être immédiatement servi, en nous facilitant la précision de la distribution. Nous vous en remercions d'avance.

Les Éditions Atlas

Dénominateur commun

Le principal obstacle que rencontrent les utilisateurs d'ordinateur qui désirent échanger des programmes est l'incompatibilité des matériels. Basicode est-il une réponse?



Normes communes
BASICODE permet aux micros de communiquer entre eux grâce à une standardisation du langage. Il utilise un jeu minimal de commandes BASIC pour permettre à environ une douzaine de micros d'échanger des programmes. Les programmes BASICODE sont transmis dans certains pays par des stations de radio, ce qui permet aux auditeurs possédant des micros différents d'utiliser les mêmes programmes. (Cl. Ian McKinnell.)

Le BASIC s'est aujourd'hui imposé comme le langage standard pour les ordinateurs domestiques. Cependant, comme le savent tous les utilisateurs d'ordinateur domestique, il n'existe pas un, mais plusieurs BASIC. Même lorsque les machines partagent un dialecte commun, comme le BASIC Microsoft, il n'est pas certain qu'un programme écrit sur un type d'ordinateur pourra être exécuté sur un modèle différent.

BASICODE représente un nouveau moyen pour résoudre le problème de la compatibilité. Il a d'abord été développé aux Pays-Bas pour « Hobbyscoop », une émission scientifique et technologique produite par Teleac, un organisme public de formation populaire. Lorsque « Hobbyscoop » commença à diffuser des émissions en 1978, le producteur s'appuyait sur les quatre machines les plus populaires du moment — l'Apple, l'Exidy Sorcerer, le PET et le Tandy TRS-80. Il ne pouvait y avoir une transmission que pour une seule machine par semaine. Comme deux de ces ordinateurs avaient des vitesses de transmission de données extrêmement basses, les auditeurs devaient attendre jusqu'à huit minutes pour que la transmission s'établisse. Il était évident que cette situation n'était pas satisfaisante. Lorsque de nouvelles machines arrivèrent sur le marché, chacune nécessitant sa propre émission, la transmission des programmes devint vraiment difficile.

Le problème intéressa d'abord un radio amateur passionné, nommé Klaas Robers, qui produisit la première version de BASICODE. Ce langage était composé d'un sous-ensemble commun de commandes BASIC qui pouvait être compris par tous les types d'ordinateur. Le nouveau système ne fonctionnait pas toujours parfaitement et cette tentative de standardisation échoua. Klaas Robers développa donc avec Jochem Herrmann une version améliorée du langage, nommée BASICODE-2.

Les premières diffusions de BASICODE-2 furent faites au début de 1983, et s'avérèrent bientôt un succès. Des auditeurs, habitant en Belgique, en France, en Grande-Bretagne, en Allemagne et au Danemark, chargèrent les programmes avec succès. Cet intérêt international augmenta lorsque le service de radiodiffusion néerlandais commença à transmettre BASICODE-2 sur son réseau externe.

BASICODE est fondé sur les 42 mots clés et les 11 symboles que la plupart des machines utilisent, pour exploiter le langage qu'elles ont en commun. Un mot clé n'est pas stocké sous la

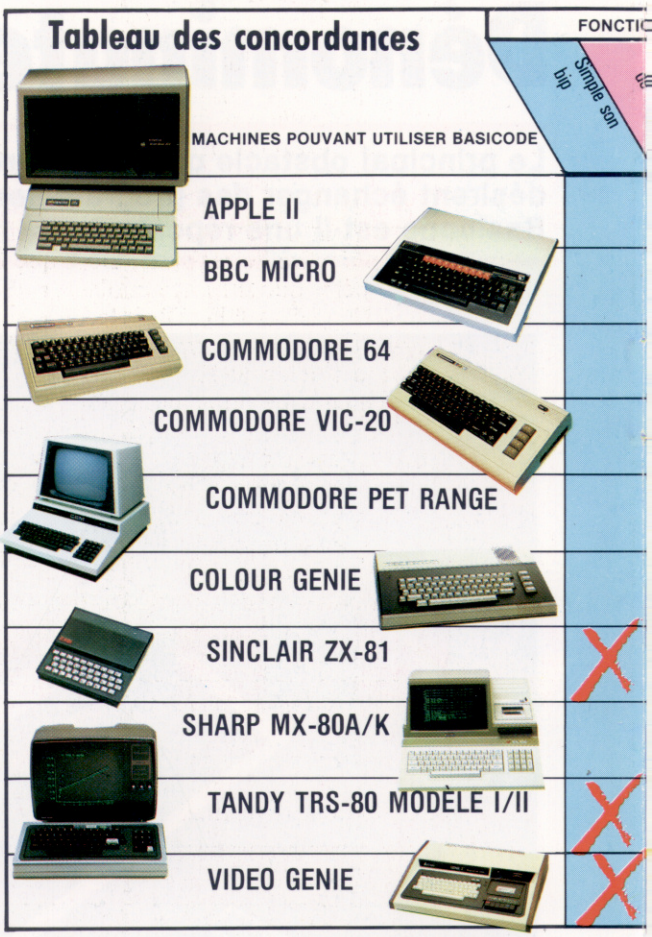
forme de caractères mais sous la forme d'un jeton d'un octet, symbole qui représente le mot clé. Par exemple, le mot clé LEFT$ est représenté sur le Commodore 64 par un seul octet renfermant la valeur 200, au lieu des cinq octets contenant les valeurs ASCII de L, E, F, T et $. Cela accélère le travail de l'interpréteur BASIC et permet d'utiliser beaucoup moins de mémoire RAM. Cependant, bien que chaque ordinateur utilise ce type de stockage par jetons pour interpréter un programme BASIC, chaque machine utilise différentes valeurs pour ses jetons. Le problème fut résolu par l'apport de deux programmes de traduction, BASICODE-Save et BASICODE-Load. Après avoir écrit un programme en BASIC, celui-ci est sauvegardé en utilisant le programme BASICODE-Save, qui remplace les jetons BASIC de l'ordinateur par les jetons BASICODE standard et produit sur bande un programme standard BASICODE. Ce programme peut alors être chargé sur une autre machine à l'aide du programme BASICODE-Load, qui remplace les jetons BASICODE par les jetons spécifiques de l'ordinateur.

Une question importante est soulevée : comment garantir que les divers types d'ordinateur lisent et écrivent de la même manière sur la bande? De nouveau, bien que toutes les machines utilisent le même principe pour charger et pour sauvegarder des programmes sur bande, un programme produit sur bande par un ordinateur peut, en pratique, être très différent de celui produit par une autre machine. Non seulement les données peuvent être décrites et lues sur la bande à des vitesses de transmission différentes, mais les bits de départ et d'arrêt (les marqueurs qui indiquent à l'ordinateur où commencent et où se terminent les données) et les totaux de contrôle (le système qui permet à la machine de vérifier l'exactitude de la transmission des données) peuvent également être totalement différents. La solution adoptée fut de supprimer les méthodes de gestion de bande de l'ordinateur et d'imposer un format *audiocode* commun pour la transmission.

Dans ce format, les données sont transmises à 1 200 bits par seconde. Chaque octet de données, précédé d'un bit de départ (valeur 0), est transmis en envoyant d'abord le bit le moins significatif suivi de deux bits d'arrêt (tous deux avec une valeur de 1). Par exemple, la valeur ASCII de « A » est 65 — 01000001 en binaire — et cela serait transmis en audiocode comme 010000001011. Un marqueur d'amorce, composé d'une séquence de bits d'arrêt transmise pendant cinq secondes, indique le début d'un programme BASICODE. Un code « début de texte » (82 en hexadécimal) suit. Le programme BASIC est suivi d'un octet de total de contrôle qui permet à l'ordinateur de vérifier l'exactitude des données transmises. Une autre séquence de cinq minutes de bits d'arrêt indique la fin de la transmission de données.

Bien que presque toutes les machines puissent être adaptées à BASICODE uniquement par logiciel, le TRS-80 modèles I et III et le Video Genie requièrent la présence d'une petite interface pour leur permettre de lire les bandes correctement. Le manuel fourni avec la cassette BASICODE-2 donne des détails complets sur la manière de construire l'interface.

Ce diagramme illustre à quel niveau de conformité se situent les machines pouvant utiliser BASICODE. Les machines dont les spécifications ne répondent pas aux normes sont marquées au moyen d'une croix. Les machines marquées d'un crochet ont des caractéristiques qui vont au-delà de ce qui est permis par BASICODE.



Pour écrire un programme BASICODE, vous devez d'abord charger le programme BASICODE-Save. Ce programme ne permet pas seulement de sauvegarder sur cassette, dans un format standard, le code venant d'être écrit, mais communique aussi une liste de sous-programmes qui sont propres à chaque machine. Ces routines sont stockées entre les lignes 1 à 999, qui ne peuvent donc pas être utilisées par le programmeur.

Ces routines font partie du programme de traduction BASICODE-2, parce qu'une commande commune à plusieurs machines — comme l'instruction demandant d'effacer l'écran (CLS) — peut être exécutée de différentes façons. Au lieu d'utiliser la commande CLS, le programmeur utilise GOSUB 100, qui correspond au sous-programme BASICODE qui exécute cette fonction.

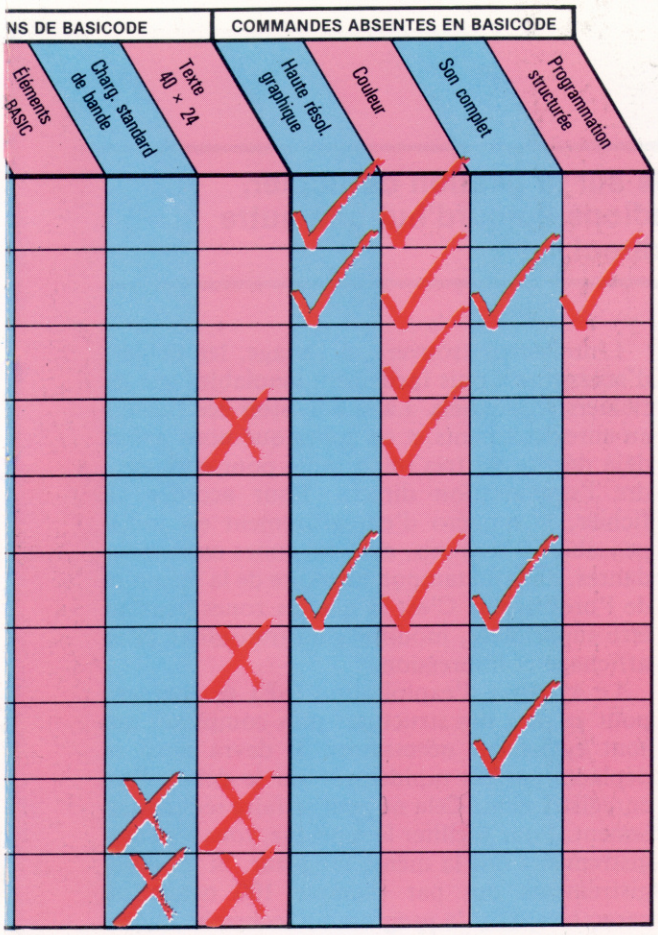
La première ligne écrite par le programmeur doit se présenter comme ceci :

```
1000 A=(valeur):GOTO 20:REM nom du programme
```

où (valeur) est le nombre maximal de caractères utilisé par toutes les chaînes. Le programmeur peut dès lors programmer comme il le désire. Il y a cependant plusieurs contraintes au niveau du format du code. Par exemple, les variables

Commandes d'opération
Voici une liste des commandes et des opérations permises par BASICODE.
Des mots clés ne seront pas reconnus par BASICODE.

ABS	NEXT
AND	NOT
ASC	ON
ATN	OR
CHR$	PRINT
COS	READ
DATA	REM
DIM	RESTORE
END	RETURN
EXP	RIGHT$
FOR	RUN
GOSUB	SGN
GOTO	SIN
IF	SQR
INPUT	STEP
INT	STOP
LEFT$	TAB
LEN	TAN
LET	THEN
LOG	TO
MID$	VAL
+	<
←	≤
-	≤>
*	<>
/	>=
↑	>=



doivent être initialisées avant qu'elles puissent être utilisées dans une opération. Pour que la commande LET T=T+1 puisse être exécutée, T doit d'abord être mis à zéro.

D'autres contraintes concernant l'utilisation de plusieurs mots clés BASIC apparaissent. Par exemple :

```
5000 INPUT "MOT DE PASSE?";A$
```

est interdit. Le format correct est :

```
5000 PRINT "MOT DE PASSE?":INPUT A$
```

De plus, une ligne de programme ne peut compter plus de 60 caractères, et l'interpréteur suppose une dimension d'écran de 24 lignes de 40 caractères.

Mais pourquoi ces contraintes sont-elles nécessaires? Afin de pouvoir utiliser BASICODE sur le plus grand nombre de machines possible. Les concepteurs ont en effet adopté l'approche du « plus petit commun dénominateur ». La sophistication de BASICODE ayant pour conséquence inévitable une restriction dans le choix des ordinateurs en mesure de l'utiliser, cela conduit à un nivellement par le bas qui permet à BASICODE de s'adapter au niveau des possibilités des machines les plus faibles, créant ainsi des contraintes sur les modèles plus évolués.

Par exemple, de nombreuses fonctions recherchées par les utilisateurs de micros ne sont pas offertes dans le format BASICODE. Le système ne contient aucune fonction permettant de faire varier le registre et la durée des sons. On ne dispose que d'une commande BEEP plutôt limitée. De même, BASICODE ne permet que de programmer des graphiques basse résolution. Et ils ne peuvent être qu'en noir et blanc.

Un autre problème réside dans le fait que depuis l'invention de BASICODE les techniques de programmation structurée ont grandement évolué. BASICODE ne comporte aucune instruction conditionnelle comme WHILE... WEND ou même aucune commande DEF FN. La programmation structurée ne peut être réalisée qu'avec la commande GOSUB, de laquelle le protocole du langage dépend largement.

En outre, il est important de souligner qu'en dépit de ces contraintes certaines des machines pouvant utiliser BASICODE ne peuvent répondre aux normes modestes définies par le protocole. Par exemple, le VIC-20, le ZX-81, le TRS-80 et le Video Genie se situent tous au-dessous de la norme de dimension d'affichage de 40 × 24 caractères.

Le programmeur expérimenté devrait trouver de l'intérêt dans la programmation en BASICODE. Puisque les règles sont contraignantes, le programmeur doit être très méticuleux pour que le programme soit transférable. Il se souviendra qu'il ne doit utiliser que les 50 mots clés et opérateurs employés par le langage, et qu'il doit utiliser des commandes GOSUB pour remplacer des instructions non standard comme CLS. Nous devons également souligner que certaines machines fonctionnant avec BASICODE ont des capacités mémoire très limitées. De très longs programmes, parfaitement valables en BASICODE, ne fonctionneront pas dans la RAM disponible sur certaines machines. Il est parfaitement possible d'écrire un programme fonctionnant sur votre machine, mais dont l'essai sera négatif sur une autre.

A l'intérieur du programme principal, le programmeur peut ajouter certaines fonctions qui ne seraient pas permises normalement. C'est possible en ajoutant des instructions REM, expliquant précisément l'intention du programmeur. Les auteurs de BASICODE recommandent d'insérer ces instructions entre les lignes 20000 et 24999, bien que cela ne soit pas obligatoire. Une fois le programme chargé, il est possible d'ajouter des fonctions prises en charge par la machine.

En Grande-Bretagne, les instructions complètes concernant l'utilisation de BASICODE-2 avec le logiciel destiné aux séries d'émissions « Chip Shop » sont fournies. L'utilisateur reçoit une cassette renfermant sur la face 1 les programmes de traduction pour les diverses machines. Tous les programmes sont séparés par des messages audibles pour aider l'utilisateur à trouver celui qu'il recherche. Sur la face 2 de la cassette, on retrouve 18 programmes de démonstration qui illustrent les possibilités de BASICODE.

Étant donné les variations étonnantes qu'on trouve dans ce qui est censé être un langage standard, les auteurs de BASICODE ont réussi un coup de maître.

Accès direct

Nous avons vu, dans le dernier article sur la gestion de fichier, l'organisation séquentielle. Nous étudions aujourd'hui une autre technique complémentaire : l'accès direct.

Les limites de l'organisation séquentielle tiennent au fait que l'information doit être passée en revue depuis le début du fichier jusqu'à l'enregistrement recherché. Les fichiers à accès direct répondent à ce problème puisqu'ils autorisent l'accès dans n'importe quel ordre et très rapidement à tout enregistrement. Les fichiers à accès aléatoire (de l'anglais « random ») ne signifient pas qu'ils sont désordonnés. Au contraire, chacun de leurs segments peut être lu ou mis à jour directement, sans passer par les enregistrements précédents.

Le problème qui se pose est que les fichiers sur cassette sont organisés de façon séquentielle. Il n'est pas possible d'accéder directement à un enregistrement se trouvant quelque part sur une bande. La totalité de la bande doit être lue avant de rencontrer l'enregistrement. Aussi la seule manière d'utiliser les fichiers à accès direct sur des micros qui fonctionnent avec une mémoire de masse sur cassettes consiste à changer toutes les données du fichier concerné en mémoire centrale. Cela présente l'inconvénient majeur de poser des limites à la taille des fichiers. Les lecteurs de disquettes sont pratiquement indispensables pour un accès intéressant. Mais cela n'est pas toujours possible sur tous les modèles.

Directs contre séquentiels

	ACCÈS DIRECT	ACCÈS SÉQUENTIEL
POUR	• Accès rapide à un enregistrement.	• Prend peu de place. • Disponible sur bande.
CONTRE	• Perte de place. • Nécessite des disques.	• Lent et lourd.
APPLICATIONS	• Données de taille prévisible et de formats déterminés. • Pour un petit nombre d'enregistrements différents ; par exemple, bibliothèque, titres de livres. Le taux de succès de recherche est faible.	• Grande quantité de données. • Enregistrements traités ensemble. Par exemple, pour la paie de toute une entreprise. Le taux de succès est élevé.

L'utilisateur trouvera, à l'usage, les fichiers d'accès direct plus facilement maniables que les fichiers séquentiels. La subdivision des fichiers en enregistrements et en zones que nous avons déjà décrite est très importante pour comprendre l'organisation directe. Pour accéder au fichier, le numéro d'enregistrement est indispensable. Il sera mis, avec les zones correspondantes, dans un tampon mémoire de la mémoire de l'ordinateur. C'est là que les zones peuvent être supprimées, modifiées ou transmises pour affichage et impression.

Le système d'exploitation (SE) se chargera pour sa part des structures plus complexes qui sont également nécessaires. Il devra ainsi se déplacer parfois rapidement sur le début d'un enregistrement. Cela ne peut se faire séquentiellement. Pour faciliter le repérage, tous les enregistrements ont la même longueur. Si les enregistrements ont par exemple 100 octets ou caractères, et si le programme recherche l'enregistrement n° 83, le système d'exploitation positionnera la tête du disque sur le début du 8300e octet du fichier. Le SE tient le compte du nombre d'octets par secteur, et peut donc calculer la position exacte de l'enregistrement demandé.

Cette méthode de lecture d'un fichier peut sembler compliquée et lente, mais elle est beaucoup plus rapide que l'organisation séquentielle.

Lorsque vous choisissez la taille standard pour les fichiers, vous devez bien sûr tenir compte des plus longs enregistrements possibles. Les plus courts doivent être complétés de blancs ou caractères espaces (32 en code ASCII). Cela constitue l'un des inconvénients des fichiers directs, puisque le remplissage des positions manquantes fait perdre beaucoup de place mémoire. On utilisera donc plus volontiers les fichiers à accès direct pour de petits enregistrements dont l'accès doit être très rapide. L'organisation séquentielle concernera plutôt des fichiers recouvrant des masses énormes d'informations pour lesquelles le temps d'accès a peu d'importance.

Les zones d'enregistrement doivent également être d'une longueur standard. Cela concerne plus particulièrement les systèmes qui permettent d'accéder directement à une zone déterminée, et non plus seulement à un enregistrement. Pour les autres systèmes, cela n'en constitue pas moins un moyen de définition plus rigoureux de l'accès. La première étape, lors de la création

d'un fichier à accès direct, est de lister les différentes zones et de leur choisir une longueur standard. Ainsi, une zone destinée à recevoir le nom d'une personne doit comporter au moins 20 caractères, alors que la zone pour l'âge de la personne prendra seulement 2 caractères.

Lors de la conception du fichier, ayez constamment à l'esprit la nécessité d'organiser parcimonieusement les zones. Il vous faudra continuellement chercher un moyen terme entre le nombre d'informations stockées et le nombre des zones. Souvent vous aurez recours à des systèmes de codage servant à réduire la place prise par les données. Ainsi, on pourra attribuer respectivement les codes de couleur 1, 2 et 3 au noir, rouge et vert; ou encore des codes de date comme 841011 pour le 11 octobre 1984. Les systèmes de codage doivent rester internes au système, et les programmes doivent pouvoir traduire les codes sous une forme compréhensible, pour l'affichage ou la saisie des zones.

Deux autres facteurs doivent rester à l'esprit lors du choix de la longueur des enregistrements. La plupart des systèmes imposent en effet des limites à cette dernière. Cela peut varier de 128 octets à 2 048. En outre, il est souvent plus efficace de choisir une longueur multiple ou facteur de la taille d'un secteur : les chiffres 64, 128, 256 ou 512 sont souvent utilisés. Cela évite aux enregistrements de dépasser la taille d'un secteur et permet de faire un seul appel par enregistrement, d'où une économie d'appels de disque.

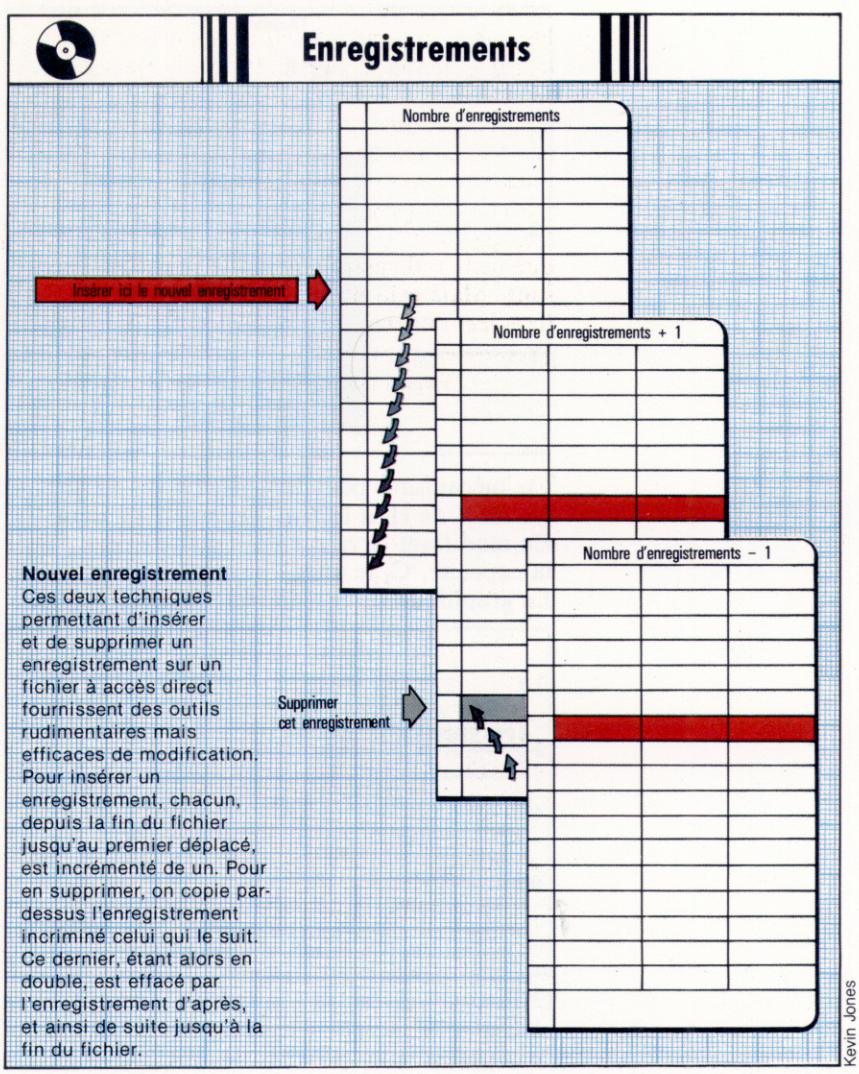
Les fichiers directs sont généralement plus maniables que les fichiers séquentiels. Pour les deux systèmes, vous devez tenir le compte du nombre d'enregistrements. Dans un fichier à accès direct, le premier enregistrement, de numéro 0, contient cette information avec d'autres informations importantes telles que la date de la création du fichier. Le mode séquentiel et sa structure rigide pour un enregistrement et ses zones ne conviendraient pas ici.

Un enregistrement peut être modifié en le lisant, et ensuite remis à sa place dans le fichier. C'est son numéro qui permet d'y avoir accès. Un utilisateur ne peut pas, de toute évidence, se souvenir d'un enregistrement par son numéro. Aussi existe-t-il toute une variété de techniques pour chercher et repérer des enregistrements, semblables à celles qui permettent de passer en revue des tableaux BASIC. Souvent une zone déterminée, ou un nom de zone, est utilisée comme clé d'accès. L'ordinateur va lire dans la zone de la clé et établit un index qui sert à identifier les noms de zone.

Les fichiers à accès direct qui ne comportent pas d'index sont souvent lus enregistrement par enregistrement, comme des fichiers séquentiels. Si les enregistrements sont triés dans la zone clé, les résultats seront beaucoup plus performants.

La méthode la plus élémentaire pour supprimer un enregistrement consiste à copier, à la place de l'enregistrement incriminé, l'enregistrement qui le suit. L'enregistrement supprimé est dit « recouvert en écriture ». Comme ce dernier se trouve être ainsi en double, on répète la même opération pour lui et ensuite pour tous les autres enregistrements jusqu'à la fin du fichier. Chaque enregistrement venant effacer le précédent qui n'est autre que lui-même recopié. On supprime ainsi les doubles jusqu'à la fin du fichier. En dernier lieu, le compte du nombre d'enregistrements est diminué d'une unité. De manière semblable, il est possible d'insérer un enregistrement en n'importe quel point en incrémentant le dernier enregistrement du fichier d'une unité et en reportant cette incrémentation sur tous les enregistrements compris entre lui et l'enregistrement nouvellement créé.

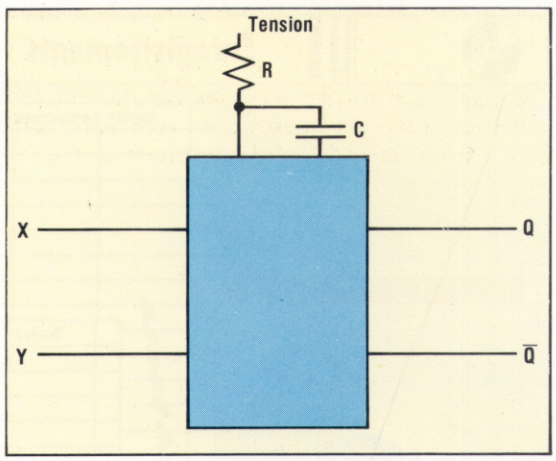
Les fichiers à accès direct présentent encore deux avantages. Premièrement, s'il est plus rapide d'écrire et de lire des blocs entiers d'enregistrements à la suite, les fichiers en deviennent désordonnés. C'est pourquoi la plupart des programmes permettent de trier les enregistrements selon un ordre logique, et d'écarter les enregistrements supprimés. Deuxièmement, le système qui consiste à marquer seulement comme devant être supprimés les enregistrements à effacer fournit une sécurité appréciable dans la mesure où ils restent récupérables. Cette garantie jouera jusqu'à ce que le programme de remise en ordre réorganise le fichier.



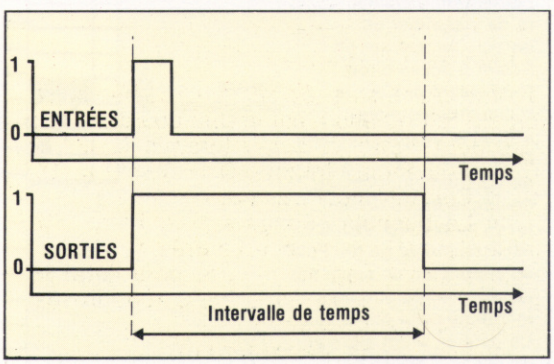
Dans les temps

Un ordinateur doit être doté d'un chronométrage précis. Nous voyons ici trois types de circuits qui produisent des signaux de synchronisation — circuits monostables (à un seul état), de type D, et bascules de type J-K.

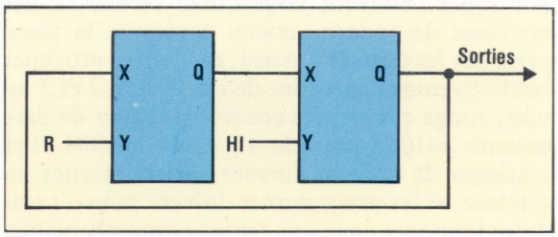
Un *circuit monostable* permet d'introduire des intervalles de temps fixes dans des opérations de circuits logiques. Lorsqu'un circuit monostable reçoit en entrée une impulsion, la sortie prend la valeur 1 (HI/HIgh-activé) pour un intervalle de temps (durée) déterminé, avant de reprendre son état initial de sortie, 0 (LO/LOw-inhibé). La durée de cet intervalle est déterminée par certains composants du circuit. Exemple :



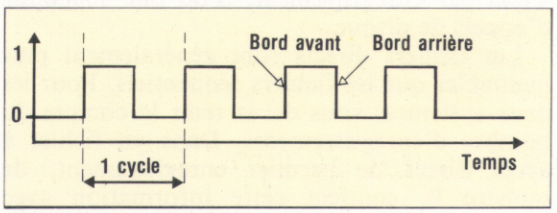
Ce mécanisme peut être déclenché en faisant passer X de HI à LO ou, encore, Y de LO à HI. En modifiant la valeur de la résistance, R, et de la capacité, C, le temps de sortie peut changer. Le graphique suivant montre la relation E/S :



L'intervalle de temps à la sortie HI peut servir à contrôler le moteur d'avancement d'un lecteur de cassettes. Il est possible de relier deux circuits monostables pour générer des impulsions qui oscilleront selon un intervalle fixe entre HI et LO :



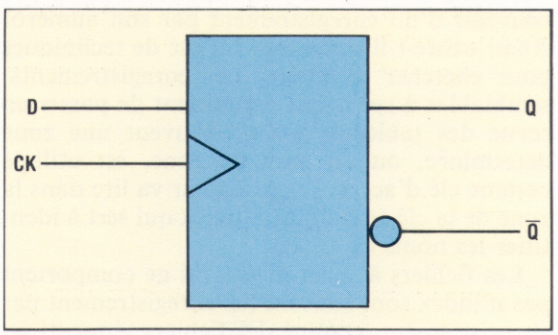
La sortie produit une courbe en dents de scie ressemblant davantage à des créneaux. L'intervalle de temps entre deux passes HI de l'horloge s'appelle un cycle. Il est souvent de l'ordre du millionième de seconde. C'est ce signal régulier qui constitue le battement de cœur d'un ordinateur ; c'est lui qui dirige les diverses fonctions de l'UC. La courbe suivante indique les noms pour les phases montante et descendante de l'activation du signal (passage de HI à LO, et vice versa, pour une impulsion) :



Voyons maintenant deux nouvelles sortes de bascules dirigées par les impulsions de l'horloge.

La bascule de type D

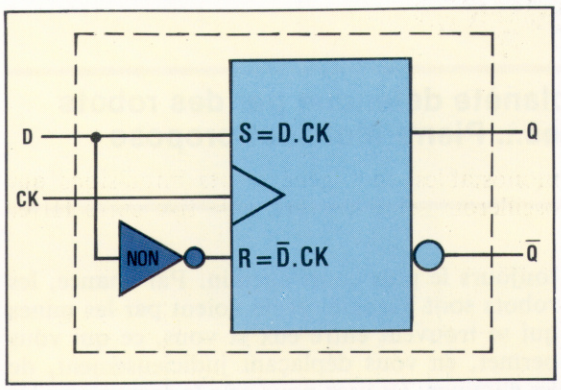
La bascule de type D comporte une entrée logique (D) et une entrée signal d'horloge (CK).

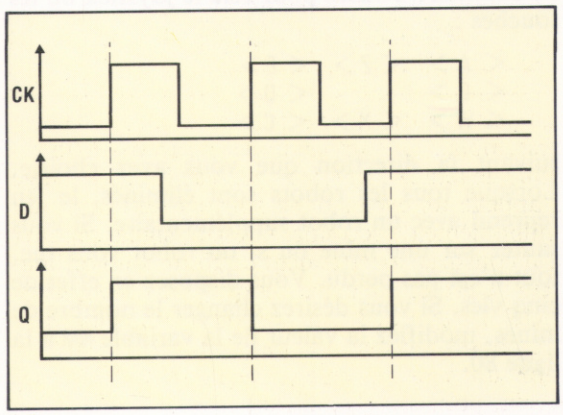


La bascule D est conçue sur la base d'une bascule R-S. C'est le signal d'horloge en entrée qui en fait l'originalité, avec ce fonctionnement particulier appelé *verrouillage*.

La sortie du circuit Q dépend du début du cycle d'horloge. Si l'entrée en D est HI, la sortie Q est alors activée (HI). Dans le cas contraire, si

l'entrée en D est inhibée (LO), la sortie Q est également inhibée (LO).

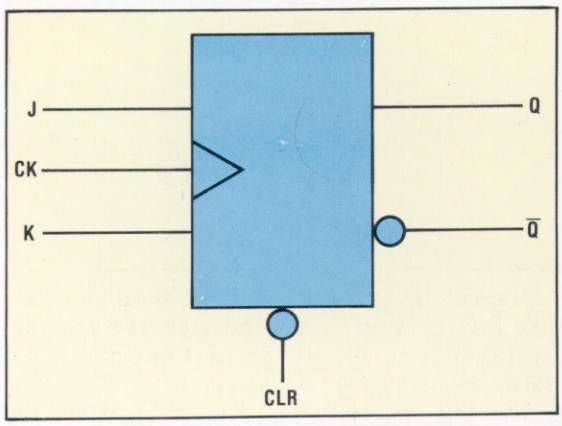




Il ressort de ces courbes que la sortie Q ne peut changer que pendant le passage de LO à HI du signal d'horloge. La bascule de type D est « à déclenchement au bord avant » du signal.

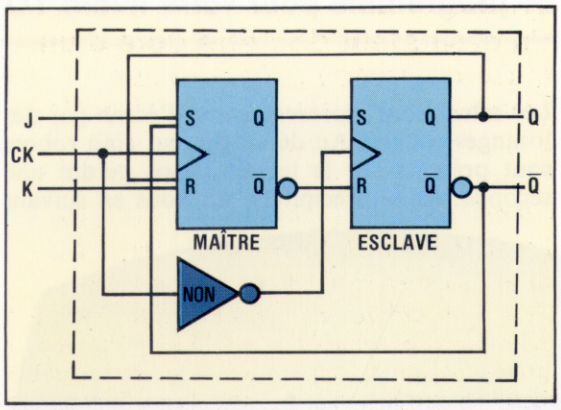
La bascule de type J-K

La bascule J-K est appelée dispositif maître/esclave du fait qu'elle comporte deux bascules R-S qui entretiennent un rapport serveur-servant. Cela permet de stocker une impulsion d'entrée dans l'une des bascules tout en fournissant une sortie depuis l'autre en relation avec l'entrée de la première. Le tout pendant un seul signal d'horloge. On peut donner comme exemple l'opération de décalage, commune à la plupart des processeurs :



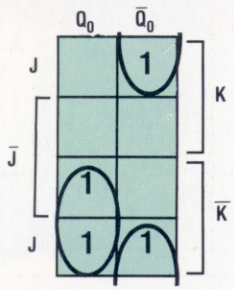
Liz Dixon

Le diagramme suivant montre comment les deux bascules de type R-S sont en rapport mutuel. L'une est le maître, l'autre l'esclave. Supposons qu'une entrée soit appliquée à J ou à K : si l'impulsion de l'horloge est HI, alors l'entrée alimente le maître; si l'entrée de l'horloge est LO, alors c'est l'esclave qui est alimenté, jusqu'à ce que les bascules R-S fassent un bond en avant :



Nous donnons dans la marge la *table d'états* pour la bascule de type J-K. Cette table est semblable à une table de vérité avec en plus la variable Q_0, la dernière entrée. Vous remarquez que les entrées HI, simultanément en J et en K, suscitent un changement d'état de la bascule, à chaque impulsion d'horloge. Cela s'appelle *effet de levier* (toggling), et est dû à l'*effet de retour* (feedback) des sorties asservies vers les entrées serveuses. Pour une bascule de type R-S, cela revient à un refus de combinaison en entrée :

Q_0	J	K	Q
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0



Nous pouvons déduire l'expression logique de la table de Karnaugh :

$$Q = \overline{Q}_0.J + Q_0.\overline{K}$$

Cette équation est appelée la *caractéristique de la bascule de type J-K.*

Réponses à l'exercice 7

1. Une bascule est également appelée bistable du fait qu'elle n'est stable que dans l'un des deux états (à savoir lorsque Q = 1 et $\overline{Q}$ = 0 ou encore lorsque Q = 0 et $\overline{Q}$ = 1).

2. *a*) Cet état n'est pas stable.

b) La bascule changera pour l'état RESET (re-initialisation) lorsque l'entrée supérieure est considérée en premier, ou pour l'état SET (activation) lorsque la porte inférieure est d'abord prise en compte.

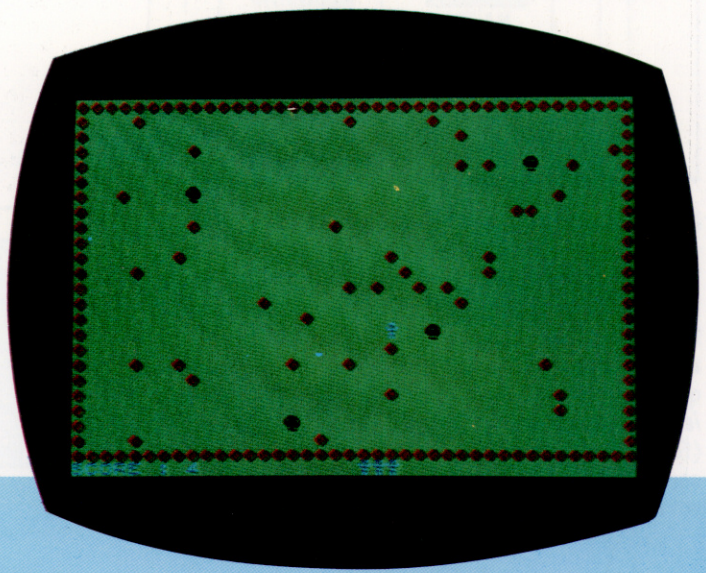
c) Oui (voir la réponse du *b*).

d) Pour que les registres passent à l'état stable lors de la mise sous tension, il faut faire en sorte qu'ils soient mis à R ou à S.

Robots

Vous voici seul, abandonné sur une planète défendue par des robots meurtriers; et le sol est truffé de mines... Pierre Monsaut propose ce programme pour votre micro TO 7.

Les mines sont représentées sur l'écran par des losanges rouges. Au début du jeu, cinq robots sont présents sur le terrain. Sans perdre une seconde, ils se précipitent sur vous en suivant toujours le plus court chemin. Par chance, les robots sont aveugles et ne voient par les mines qui se trouvent entre eux et vous, ce qui vous permet, en vous déplaçant judicieusement, de les éliminer. Utilisez pour cela le joystick ou les touches :



```
< A >  < Z >  < E >
< Q >         < D >
< W >  < X >  < C >
```

suivant la direction que vous avez choisie. Lorsque tous les robots sont éliminés, le jeu reprend avec un robot supplémentaire. Si vous sautez sur une mine ou si un robot vous tue, tout n'est pas perdu. Vous disposez en effet de cinq vies. Si vous désirez changer le nombre de mines, modifiez la valeur de la variable NM à la ligne 80.

```
10 REM **********
20 REM * ROBOTS *
30 REM **********
40 DEFINT A-Z
50 CLEAR ,,3
60 NH=5
70 N1=5
80 NM=40
90 NR=N1
100 DIM R(30,1)
110 GOSUB 1580
120 GOSUB 1470
130 GOSUB 910
140 ON JS GOSUB 710,810
150 C=POINT(HX*8+4,HY*8+4)
160 IF C<>-3 AND C<>4 THEN 470
170 COLOR 4
180 LOCATE X,Y
190 PRINT N$;
200 LOCATE HX,HY
210 PRINT H$;
220 X=HX
230 Y=HY
240 T=0
250 FOR I=1 TO NR
260 IF R(I,0)=0 THEN 400
270 T=1
280 RX=R(I,0)+SGN(HX-R(I,0))
290 RY=R(I,1)+SGN(HY-R(I,1))
300 C=POINT(RX*8+4,RY*8+4)
310 IF C=1 OR C=0 THEN S=S+1:LOCATE R(I,
0),R(I,1):PRINT N$;:R(I,0)=0:GOTO 400
320 IF C=4 THEN 470
330 COLOR 0
340 LOCATE R(I,0),R(I,1)
350 PRINT N$;
360 LOCATE RX,RY
370 PRINT R$;
380 R(I,0)=RX
390 R(I,1)=RY
400 NEXT I
410 IF T=0 THEN 430
420 GOTO 140
430 S=S+10
440 IF INKEY$<>"" THEN 440
450 IF NR<30 THEN NR=NR+1
460 GOTO 130
470 NH=NH-1
480 COLOR 7
490 LOCATE X,Y
500 PRINT N$;
510 LOCATE HX,HY
520 PRINT H$;
530 PLAY "L96REL72REL24REL96REL72FAL24MI
L72MIL24REL72REL24DO#L96RE"
540 IF INKEY$<>"" THEN 540
550 IF NH>0 THEN NR=N1:GOTO 130
560 CLS
570 SCREEN 1,6,6
580 ATTRB 1,1
590 LOCATE 9,10
600 PRINT "SCORE :";S;
610 LOCATE 9,20
620 PRINT "UNE AUTRE ?";
630 COLOR 4
640 ATTRB 0,0
650 IF INKEY$<>"" THEN 650
660 D$=INKEY$
670 IF D$="" THEN 660
680 IF D$<>"N" THEN RUN
690 CLS
700 END
710 D$=INKEY$
720 IF D$="A" THEN HX=HX-1:HY=HY-1
730 IF D$="Z" THEN HY=HY-1
740 IF D$="E" THEN HY=HY-1:HX=HX+1
750 IF D$="Q" THEN HX=HX-1
760 IF D$="D" THEN HX=HX+1
770 IF D$="W" THEN HX=HX-1:HY=HY+1
780 IF D$="X" THEN HY=HY+1
790 IF D$="C" THEN HY=HY+1:HX=HX+1
800 RETURN
810 J=STICK(0)
820 IF J=1 THEN HY=HY-1
830 IF J=2 THEN HY=HY-1:HX=HX+1
840 IF J=3 THEN HX=HX+1
850 IF J=4 THEN HX=HX+1:HY=HY+1
860 IF J=5 THEN HY=HY+1
870 IF J=6 THEN HY=HY+1:HX=HX-1
880 IF J=7 THEN HX=HX-1
890 IF J=8 THEN HX=HX-1:HY=HY-1
900 RETURN
910 CLS
920 COLOR 4
930 LOCATE 0,24
940 PRINT "SCORE :";S;
950 IF NH=1 THEN 1000
960 FOR HX=1 TO NH-1
970 LOCATE 19+HX,24
980 PRINT H$;
990 NEXT HX
1000 COLOR 1
1010 FOR HX=0 TO 39
1020 LOCATE HX,0
1030 PRINT M$;
1040 LOCATE HX,23
1050 PRINT M$;
1060 NEXT HX
1070 FOR HY=1 TO 22
1080 LOCATE 0,HY
1090 PRINT M$;
1100 LOCATE 39,HY
1110 PRINT M$;
1120 NEXT HY
1130 FOR I=1 TO NM
1140 HX=INT(RND*38)+1
1150 HY=INT(RND*22)+1
1160 IF SCREEN(HX,HY)<>32 THEN 1140
1170 LOCATE HX,HY
1180 PRINT M$;
1190 NEXT I
1200 COLOR 0
1210 FOR I=1 TO NR
1220 R(I,0)=INT(RND*38)+1
1230 R(I,1)=INT(RND*22)+1
1240 IF SCREEN(R(I,0),R(I,1))<>32 THEN 1
220
1250 LOCATE R(I,0),R(I,1)
1260 PRINT R$;
1270 NEXT I
1280 HX=INT(RND*38)+1
1290 HY=INT(RND*22)+1
1300 IF SCREEN(HX,HY)<>32 THEN 1280
1310 X=HX
1320 Y=HY
1330 FOR I=1 TO 5
1340 LOCATE HX,HY
1350 COLOR 5
1360 PRINT CHR$(127);
1370 BEEP
1380 FOR J=1 TO 50
1390 NEXT J
1400 LOCATE HX,HY
1410 COLOR 4
1420 PRINT H$;
1430 FOR J=1 TO 50
1440 NEXT J
1450 NEXT I
1460 RETURN
1470 CLS
1480 SCREEN 4,2,0
1490 ATTRB 1,1
1500 LOCATE 10,10,0
1510 PRINT "JOYSTICK ?";
1520 ATTRB 0,0
1530 D$=INKEY$
1540 C=RND
1550 IF D$="" THEN 1530
1560 IF D$="0" THEN JS=2 ELSE JS=1
1570 RETURN
1580 DEFGR$(0)=28,28,73,62,8,28,20,20
1590 DEFGR$(1)=60,126,219,255,255,126,36
,60
1600 DEFGR$(2)=0,0,24,60,126,126,60,24
1610 H$=GR$(0)
1620 R$=GR$(1)
1630 M$=GR$(2)
1640 N$=CHR$(32)
1650 RETURN
```

L'Apricot en action

L'Apricot est un ordinateur très compact conçu par la société britannique ACT. Ce n'est pas un « véritable » portable, mais il est assez léger pour être déplacé facilement.

Conçu en vue d'une utilisation professionnelle, l'Apricot est livré en deux versions : l'une comprend deux lecteurs de disquettes 3 1/2 pouces et l'autre un seul lecteur de disquetes 3 1/2 pouces et un disque dur de 10 Moctets. Bien qu'offrant de nombreuses fonctions destinées à attirer l'utilisateur sérieux, l'Apricot a fait peu d'efforts en direction du marché domestique — il n'a pas de couleur, de port cassette, de manette de jeux ou de sortie de téléviseur. Cependant, la version de base offre un moniteur monochrome à haute résolution, un port d'imprimante parallèle, un port série RS232, un connecteur destiné à une souris optionnelle, certains programmes et un clavier de qualité.

Le clavier de l'Apricot mérite que l'on s'y attarde. Le Microscreen est un nouveau dispositif qui propose un affichage à cristaux liquides de deux lignes sur 40 colonnes dans le coin supérieur droit du clavier. Lors de la mise sous tension, la ligne supérieure de ce mini-écran affiche la date et l'heure, qui peuvent être modifiées à l'aide des programmes utilitaires. Ces données sont maintenues à jour grâce à une horloge alimentée par piles quand l'ordinateur n'est pas sous tension.

Lors de la mise sous tension de l'ordinateur, un programme test est automatiquement exécuté. Ce test affiche la quantité de mémoire disponible (la version standard offre 256 K mais ce nombre peut être porté à 768) et demande à l'utilisateur d'insérer la disquette MS-DOS. Pour les utilisateurs non familiers avec les systèmes d'exploitation comme CP/M ou MS-DOS, un menu nommé le « Manager » permet une sélection facile des programmes d'application (comme Supercalc, Multiplan, BASIC Microsoft, etc.) ou des utilitaires (comme le configurateur de clavier ou l'éditeur des polices de caractères).

Le micro-écran étant commandé par logiciel, il peut donc afficher autre chose que l'heure et la date. Il y a six touches de fonction programmables, et les fonctions affectées à ces touches peuvent être indiquées sur l'écran à cristaux liquides. Par conséquent, lorsqu'un programme affiche un menu sur l'écran, le même menu peut être reproduit sur le micro-écran. Il suffit d'appuyer sur la touche de fonction appropriée pour sélectionner une option qui peut également être choisie à l'aide des touches de déplacement du curseur et de la touche RETURN. Remarque négative : les touches à membrane sont difficiles à utiliser et moins efficaces que celles de type

Une bonne affaire
L'Apricot est offert à un prix modique pour un micro de gestion, et cela avec une qualité de construction remarquable. Il utilise un microprocesseur 16 bits et la version de base comporte une mémoire de 256 K et un moniteur d'excellente qualité. (Cl. Chris Stevens.)

machine à écrire. On retrouve également huit touches de fonction ordinaires. Leurs fonctions normales sont indiquées — HELP, PRINT, MENU, FINISH, etc. Cependant, elles peuvent être reconfigurées avec le programme fourni « Keyedit ». La conception du clavier est digne d'un ordinateur professionnel, mais la position des touches Control et Espace est un peu étrange. Pour faciliter le déplacement de la machine, le clavier peut être attaché sous l'unité principale. Le poids du moniteur est cependant trop élevé pour qualifier cette machine de véritable portable.

Le logiciel fourni avec l'Apricot est constitué de plusieurs systèmes utilitaires et de Supercalc. Supercalc et Superplanner assurent les tâches de comptabilité et de prévision. Il est évident ici (comme pour d'autres programmes) que le constructeur a eu peu de temps pour mettre au point ces programmes. Deux systèmes d'exploitation sont livrés avec la machine, MS-DOS et CP/M-86. ACT a promis de distribuer des copies gratuites de CP/M-86. Seule la première version de Supercalc est livrée avec l'Apricot, mais les versions deux et trois sont disponibles.

L'une des premières critiques qui fut adressée à l'Apricot concernait certains problèmes de mise en application du système d'exploitation MS-DOS dont le fonctionnement était plutôt lent. Ce problème semble maintenant résolu. Les programmes d'application fournis avec la machine fonctionnent assez rapidement, et des programmes d'évaluation des performances ont révélé que le BASIC Microsoft est lui aussi assez

Premier lecteur
Apricot se dit le premier micro de gestion à utiliser la nouvelle génération de petites disquettes. ACT a choisi le microlecteur Sony, qui utilise une disquette de 3 1/2 pouces, protégée dans un boîtier rigide. Cela rend cette disquette plus robuste que les disquettes 5 1/2 pouces traditionnelles. Une porte actionnée par un ressort met le lecteur à l'abri de la poussière.

Apricot XI
La capacité des microdisquettes est limitée, ACT offre donc Apricot XI. Cet appareil est muni d'un disque dur intégré de 10 Moctets qui remplace l'un des deux microlecteurs.

Affichage
Un moniteur au phosphore vert à hau résolution offre un affichage net et préci

rapide. Malgré cela, on a souvent l'impression que l'Apricot n'est pas aussi rapide qu'on le souhaiterait d'une machine munie d'un processeur 8086.

La documentation de l'Apricot comprend une introduction destinée aux débutants, un guide complet pour le système d'exploitation MS-DOS, deux guides pratiques pour Supercalc et Superplanner, et des manuels détaillés pour Wordstar et Multiplan. ACT fournit peu d'informations sur le matériel, bien que les utilitaires répondent à presque tous les besoins de configuration du système. Il n'existe aucun détail concernant la topographie mémoire ou les appels système, comme pourrait en avoir besoin une société de logiciel désireuse de produire des programmes pour cette machine.

C'est avant tout un ordinateur de gestion — ce n'est pas un système destiné à l'ingénieur logiciel ou à l'amateur d'informatique. Si l'Apricot remporte le succès du Sirius, nous pouvons prévoir que des constructeurs indépendants proposeront des cartes enfichables et que ACT introduira lui-même des modules de mémoire additionnels et un modem. Mis à part sa grande souplesse d'utilisation et son faible coût, la disponibilité des logiciels MS-DOS pour cet ordinateur fait de l'Apricot une machine plutôt intéressante.

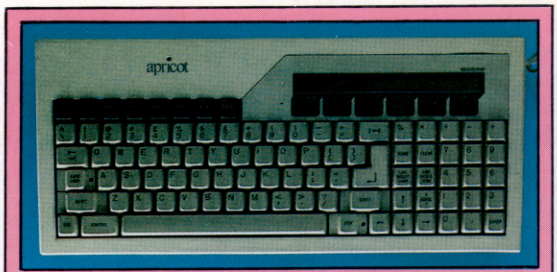
Affichage à cristaux liquides
Une unité d'affichage à cristaux liquides sur deux lignes peut être utilisée pour des messages, comme horloge ou calculatrice.

Microlecteurs Sony
Les microlecteurs de 315 K offrent un stockage compact et pratique.

Touches de fonction dédiées
Ces touches appellent des fonctions standard comme HELP et REPEAT pour différents programmes.

Touches de fonction à effleurement
L'écran à cristaux liquides peut afficher le nom de la fonction correspondant à chacune de ces six touches dans divers programmes.

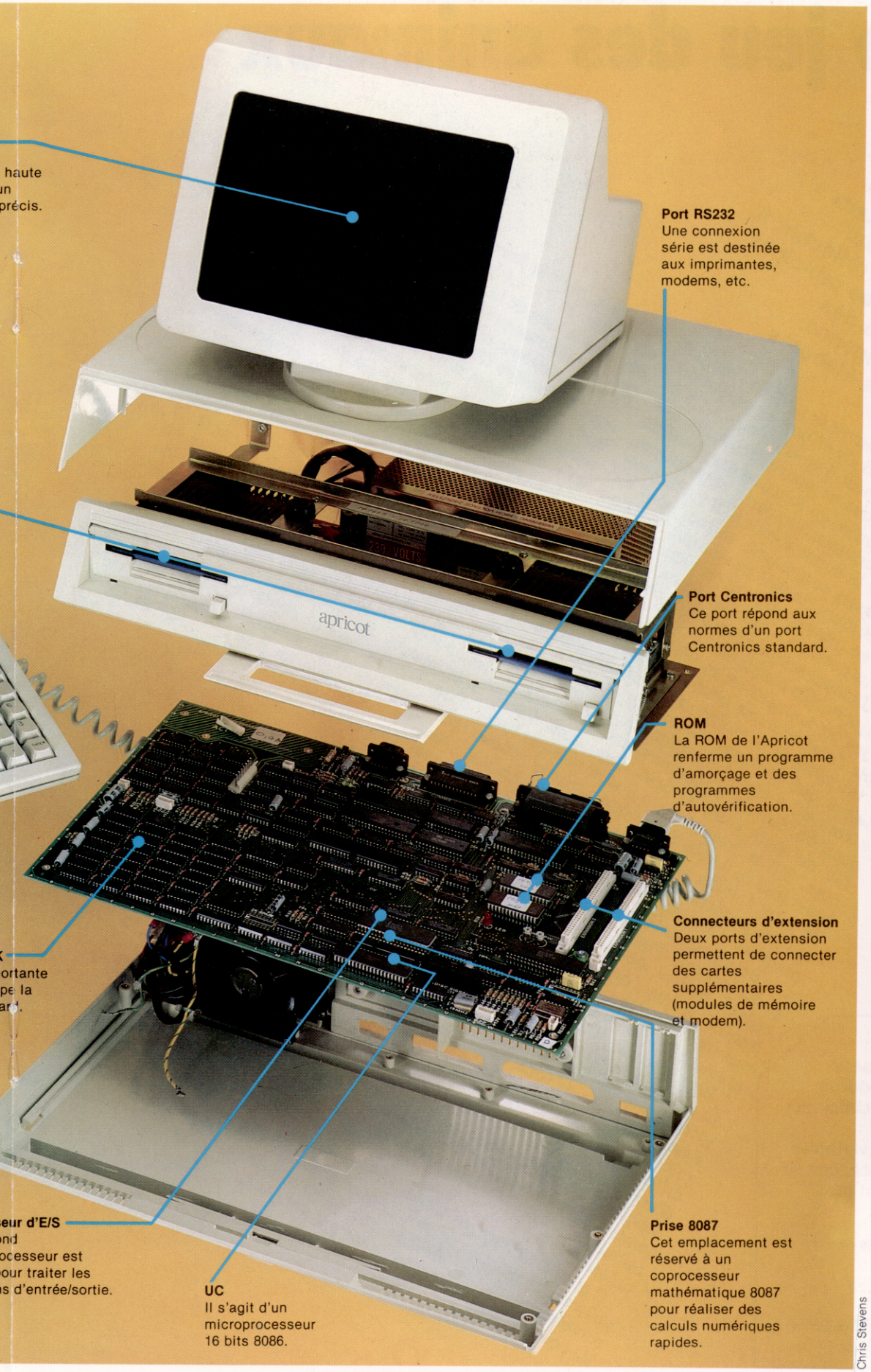
RAM de 256 K
Une RAM importa de 256 K équipe l version standar



Clavier de l'Apricot
En plus de touches de haute qualité, l'Apricot possède six touches à effleurement. Elles sont réservées à des fonctions spéciales dans divers programmes. Puisque ces fonctions changent d'un programme à l'autre, il est possible d'afficher au-dessus des touches le nom de la fonction sur l'écran à cristaux liquides. L'écran peut aussi afficher la date et l'heure.

Moniteur de l'Apricot
Bien que l'écran ne fasse que 9 pouces, il peut afficher 50 lignes de 132 caractères, mais il est normalement utilisé dans le mode 25 lignes de 80 caractères. La qualité d'affichage est bonne, et l'écran est muni d'un filtre antireflet intégré. En raison du poids de son moniteur, l'Apricot ne peut être qualifié de « portable », bien que certains utilisateurs aient choisi de garder un moniteur à la maison et un autre au travail, et n'ont plus alors qu'à transporter le micro d'un endroit à l'autre.

Processeur
Un second microproce utilisé pour fonctions d



Chris Stevens

Apricot ACT

PRIX

****

DIMENSIONS

488 × 413 × 313 mm.

UC

8086 avec le processeur mathématique optionnel 8087.

MÉMOIRE

256 K de RAM, pouvant être portée à 768 K.

ÉCRAN

Affichage de 50 lignes de 132 caractères ou 25 lignes de 80 caractères.
La résolution graphique est de 800 × 400 points en affichage monochrome uniquement.

INTERFACES

Centronics, RS232, prise « souris » et connecteurs d'extension internes.

LECTEURS DE DISQUETTES

Un ou deux microlecteurs d'une capacité de 315 ou 720 K chacun. Le modèle Apricot XI possède un disque dur de 10 Moctets et un microlecteur.

SYSTÈMES D'EXPLOITATION

MS-DOS, CP/M-86 et CP/M-86 Concurrent.

CLAVIER

90 touches de type machine à écrire plus 6 touches à effleurement avec un écran à cristaux liquides pour les identifier.

DOCUMENTATION

Complète et bien présentée.

FORCES

Une machine agréable à utiliser qui est de meilleure qualité que des machines de gestion bien cotées et beaucoup plus coûteuses. Et l'esthétique est réussie !

FAIBLESSES

D'un bon rapport qualité/prix, l'Apricot demeure un peu coûteux pour l'utilisateur domestique. Il est également dommage qu'il ne puisse utiliser les programmes CP/M-80 standard.

Le jeu des animaux

L'attrait du « jeu des animaux » vient de ce que l'ordinateur semble être capable de réfléchir; il repose d'ailleurs sur des principes que mettent en œuvre des programmes d'intelligence artificielle bien plus élaborés.

L'ordinateur doit découvrir le nom de l'animal auquel vous pensez. Il s'y efforce en posant des questions de type « peut-il voler? » ou « a-t-il de la fourrure? ». Vous ne pouvez répondre que par oui ou par non et, en fonction des éléments que vous lui fournissez, l'appareil apprend peu à peu à trouver la bonne solution. Il y a là quelque chose d'assez surprenant, surtout pour les profanes, toujours étonnés de voir que le dialogue se fait en français courant (et bien que vous deviez vous borner à « oui » ou « non »), tandis que l'ordinateur semble disposer de connaissances toujours plus étendues.

Le jeu des animaux est en fait un exemple très simple de *programme heuristique* (qui apprend de lui-même à accroître ses performances à mesure que le temps passe). Au départ il ne « connaît » que deux animaux, et une seule question. Suivant la réponse donnée à celle-ci, il fait un choix entre les deux noms dont il dispose. S'il se trompe (ce qui sera presque toujours le cas), il vous demandera d'entrer le nom de votre animal, ainsi qu'une nouvelle question qui lui permette de faire la différence avec la première. Ces informations sont intégrées dans une base de données, qui après plusieurs parties prend la forme d'une structure « arborescente »; celle-ci devient de plus en plus importante, jusqu'à ce que l'ordinateur soit finalement en mesure de deviner correctement presque tous les animaux qu'on lui propose.

Bien entendu, il ne « connaît », en fait, rien du tout aux animaux, et se borne à suivre aveuglément une méthode intangible, tirée de l'expérience combinée de tous les joueurs. On pourrait très bien remplacer les animaux par les différentes marques de bière, les composants d'une motocyclette, les symptômes des maladies, les membres de votre famille... En d'autres termes, ce ne sont pas les données qui font fonctionner le programme, mais la façon dont il est organisé.

Il n'est d'ailleurs pas très difficile à mettre au point en BASIC. Les structures nécessaires sont contenues dans des tableaux : on se servira de T$() pour les questions et les noms des animaux, et de O() et N() pour faire le lien entre les réponses apportées en T$. Le programme peut ainsi se déplacer le long de la structure en arbre. O() indique une réponse positive à la question posée en T$, N() une réponse négative. Les recherches prennent fin lorsque T$() ne comporte plus une question, mais le nom d'un animal : O() et N() sont remis à zéro, et le programme considère alors qu'il a trouvé la bonne réponse.

La version du jeu que nous vous proposons est très simple et très courte : nous nous sommes efforcés avant tout de vous indiquer clairement les principes mis en œuvre. Il vous sera facile de l'améliorer en soignant la présentation (couleurs, son, etc.) et surtout en permettant le stockage de la base de données sur cassette ou sur disquette. Les meilleures variantes du jeu sont celles que les joueurs ont peaufinées avec les années, mêlant animaux réels et mythiques, personnages célèbres, amis et objets au sein d'une énorme base de données. Il serait encore plus intéressant de modifier les questions en cherchant à donner au programme une valeur pratique, qui permettrait de l'utiliser de façon sérieuse.

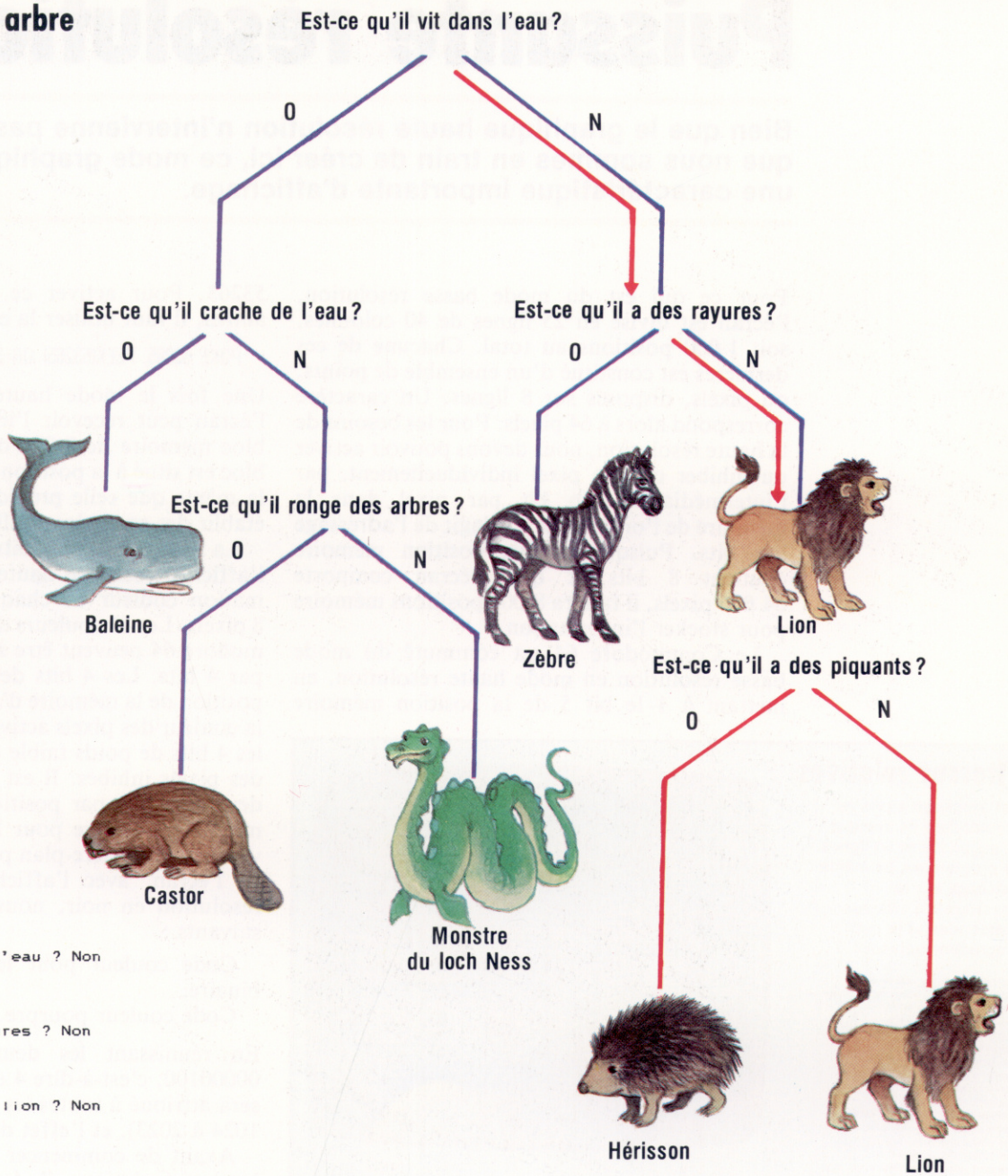
Variantes de basic

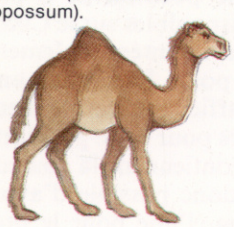
Ce programme est écrit en BASIC Microsoft, et devrait donc pouvoir tourner sans aucune modification sur la plupart des machines.
Sur le Spectrum, toutes les variables doivent être par ailleurs précédées de LET. Procédez par conséquent aux modifications suivantes :

```
45 LET L=40:REM Nombre de caractères dans une question
50 DIM O(N):DIM N(N):DIM T$(N,L)
150 LET I$=A$(1):LET P$="A "
200 IF A=30 THEN PRINT:PRINT "AU REVOIR":STOP
230 IF Y(P)=0 AND N(P)=0 THEN GOTO 290
```

Une structure en arbre

Le diagramme suivant montre l'état du jeu après plusieurs parties. Le programme connaît désormais cinq animaux, et quatre questions permettant de les distinguer. D'après l'exemple donné (indiqué en rouge), voyez comment l'ordinateur parcourt l'arbre ainsi obtenu pour répondre aux indications du prochain joueur. Celui-ci pense à un hérisson, et le programme doit pouvoir intégrer ce nouvel élément, une fois qu'il s'est rendu compte qu'il ne s'agissait pas d'un lion. Il demande alors un moyen lui permettant de distinguer l'un et l'autre. Essayez de voir si vous pouvez intégrer un chameau. N'oubliez pas de faire précéder le nom de votre animal par un article indéfini (*une* otarie, *un* opossum).





```
Un exemple

Une petite partie ? Oui

Est-ce qu'il vit dans l'eau ? Non

Est-ce qu'il a des rayures ? Non

Votre animal est-il un lion ? Non

J'abandonne !

Quel est votre animal ? un hérisson

Veuillez entrer une question permettant de faire

la différence entre un hérisson et un lion

A-t-il des piquants ?

Pour un lion la réponse serait ? Non

Je connais maintenant 6 animaux différents !

Une petite partie ?
```

```
10 REM Jeu des animaux
20 REM
30 REM** Initialisation
40 N=100 : REM Nb. maximum d'animaux
50 DIM Y(N), N(N), T$(N)
60 C=3: FOR I=1 to 3 : READ Y(I), N(I), T$(I): next I
70 PRINT : PRINT "JEU DES ANIMAUX " : PRINT
80 GOTO 190
90 REM** Réponse par Oui ou Non
100 PRINT : PRINT Q$; " ";:INPUT A$
110 IF A$:"o" OR A$ ="O" OR A$="OUI" OR A$="oui
THEN A=1 : RETURN
120 IF A$="n" OR A$ ="N" OR A$="NON" OR A$="non"
THEN A=0 : RETURN
130 PRINT:PRINT "VEUILLEZ REPONDRE PAR OUI OU PAR NON":
GOTO 100
180 REM** Début d'une partie
190 Q$="Une petite partie ?": GOSUB 100
200 IF A=0 THEN PRINT : PRINT "AU REVOIR": END
210 P=1
220 REM** Déroulement de la partie
230 IF O(P)=0 AND N(P)=0 THEN 290
240 Q$=T$(P): GOSUB 100
250 IF A=1 THEN P=Y(P)
260 IF A=0 THEN P=N(P)
270 GOTO 230
280 REM** Découverte d'un animal
290 A$=T$(P): T$=A$
300 Q$="Votre animal est-il "+A$: GOSUB 100
310 IF A=1 THEN PRINT : PRINT "Je le savais!!!":
GOTO 430
320 REM** Un nouvel animal en mémoire
330 PRINT : PRINT "J'abandonne!!!": PRINT "Quel
est votre animal ":: INPUT N$
340 A$=N$
350 PRINT / PRINT "Veuillez entrer une question
permettant de faire la différence "; PRINT "entre";
A$; " et"; T$ : INPUT D$
360 Q$="Pour "+T$+" la réponse serait ": GOSUB
100
370 A$=T$(P):T$(P)=D$:T$(C+1)=A$:T$(C+2)=N$
380 IF A=1 THEN O(P)=C+1:N(P)=C+2
390 IF A=0 THEN O(P)=C+2:N(P)=C+1
400 O(C+1)=0:N(C+1)=0:O(C+2)=0:N(C+2)=0
410 C=C+2
420 REM**Fin de partie et début d'un nouveau jeu
430 A=INT(C/2)+1
440 PRINT:PRINT"Je connais maintenant";A;"animaux
différents !"
450 GOTO 190
460 REM** Données initiales
470 DATA2, 3, "Est-ce qu'il vit dans l'eau "
480 DATA 0, 0, "une baleine"
490 DATA0, 0, "un lion"
```

David Higham

Puissante résolution

Bien que le graphique haute résolution n'intervienne pas dans le jeu que nous sommes en train de créer ici, ce mode graphique constitue une caractéristique importante d'affichage.

Pour ce qui est du mode basse résolution, l'écran est divisé en 25 lignes de 40 colonnes, soit 1 000 positions au total. Chacune de ces dernières est constitué d'un ensemble de points, ou pixels, disposés sur 8 lignes. Un caractère correspond alors à 64 pixels. Pour les besoins de la haute résolution, nous devons pouvoir activer ou inhiber chaque pixel individuellement, par l'intermédiaire d'un bit par pixel dans la mémoire de l'ordinateur. Il s'agit de l'adressage des bits. Puisque chaque position mémoire contient 8 bits et que l'écran comporte 64 000 pixels, il faudra 8 000 positions mémoire pour stocker l'information.

Le Commodore 64 est commuté du mode basse résolution en mode haute résolution, en mettant à 1 le bit 5 de la position mémoire 53265. Pour activer ce bit sans affecter les autres, il faut utiliser la commande suivante :

```
POKE 53265, PEEK(53265) OR 32
```

Une fois le mode haute résolution attribué, l'écran peut recevoir l'information depuis un bloc mémoire de 8 000 octets. Le début de ce bloc est situé à la position mémoire 53272. C'est la même que celle précédemment utilisée pour établir des caractères-utilisateurs.

La zone mémoire habituellement utilisée pour l'affichage sert, en haute résolution, à l'information couleur de chaque position de 8 par 8 pixels. Les 16 couleurs disponibles sur le Commodore 64 peuvent être représentées seulement par 4 bits. Les 4 bits de poids fort, pour une position de la mémoire d'affichage, contiennent la couleur des pixels activés pour cette position; les 4 bits de poids faible contiennent la couleur des pixels inhibés. Il est donc possible d'avoir deux couleurs par position, l'une pour le premier plan, l'autre pour l'arrière-plan. Si nous voulons un arrière-plan pourpre pour la totalité de l'écran, avec l'affichage graphique haute résolution en noir, nous utiliserons les codes suivants :

Code couleur pour le noir, 0 = 0000 en binaire.

Code couleur pourpre, 4 = 0100 en binaire.

En réunissant les deux codes, on obtient 00000100, c'est-à-dire 4 en décimal. Par POKE, 4 sera attribué à toutes les positions mémoire (de 1024 à 2023), et l'effet désiré sera obtenu.

Avant de commencer à dessiner sur l'écran haute résolution, il faut vider la zone de 8 000 octets qui commande l'affichage : chaque position sera ré-initialisée par POKE. Cela prendra plusieurs secondes en BASIC. Si cela n'était pas fait, l'affichage serait illisible car cette zone mémoire prend, lors de la mise sous tension de la machine, des valeurs aléatoires.

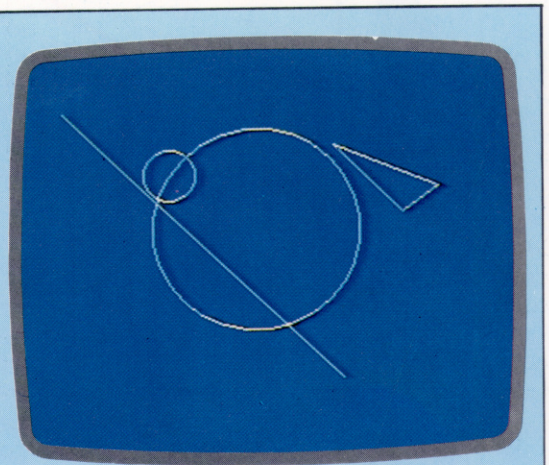
Tracé de points

Un programme graphique haute résolution nécessite que l'on puisse activer ou inhiber librement les pixels. Si chaque point reçoit les coordonnées X et Y (X et Y étant respectivement compris entre 0 et 319 et 0 et 199), le programme est alors à même d'identifier le bit correspondant dans la carte mémoire de 8 000 octets, bit devant être mis à 1 ou à 0. La position horizontale de l'octet peut être obtenue

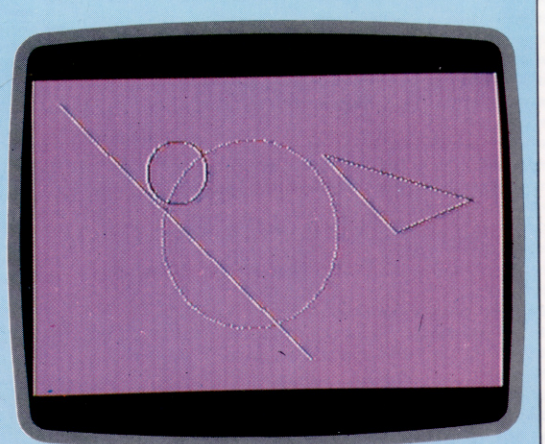
Vitesses relatives

Cet affichage a pris au Commodore 64 quelque 90 secondes et 50 lignes de programme. Le même affichage sur le Spectrum n'a pris que 2 secondes et le petit programme suivant :

```
4000 REM********Hi Res Demo********
4050 LET N=18: DIM U(N): INK 6: PAPER 1:
BORDER 1: RESTORE 4100
4100 DATA 20,160,160,-160
4120 DATA 80,123,15
4140 DATA 130,90,60
4160 DATA 175,140
4180 DATA 40,-40
4200 DATA 20,15
4220 DATA -60,25
4250 FOR K=1 TO N:READ U(K):NEXT K
4300 PLOT U(1),U(2):DRAW U(3),U(4)
4350 CIRCLE U(5),U(6),U(7)
4400 CIRCLE U(8),U(9),U(10)
4450 PLOT U(11),U(12):DRAW U(13),U(14)
4500 DRAW U(15),U(16):DRAW U(17),U(18)
4600 PAUSE 0
```

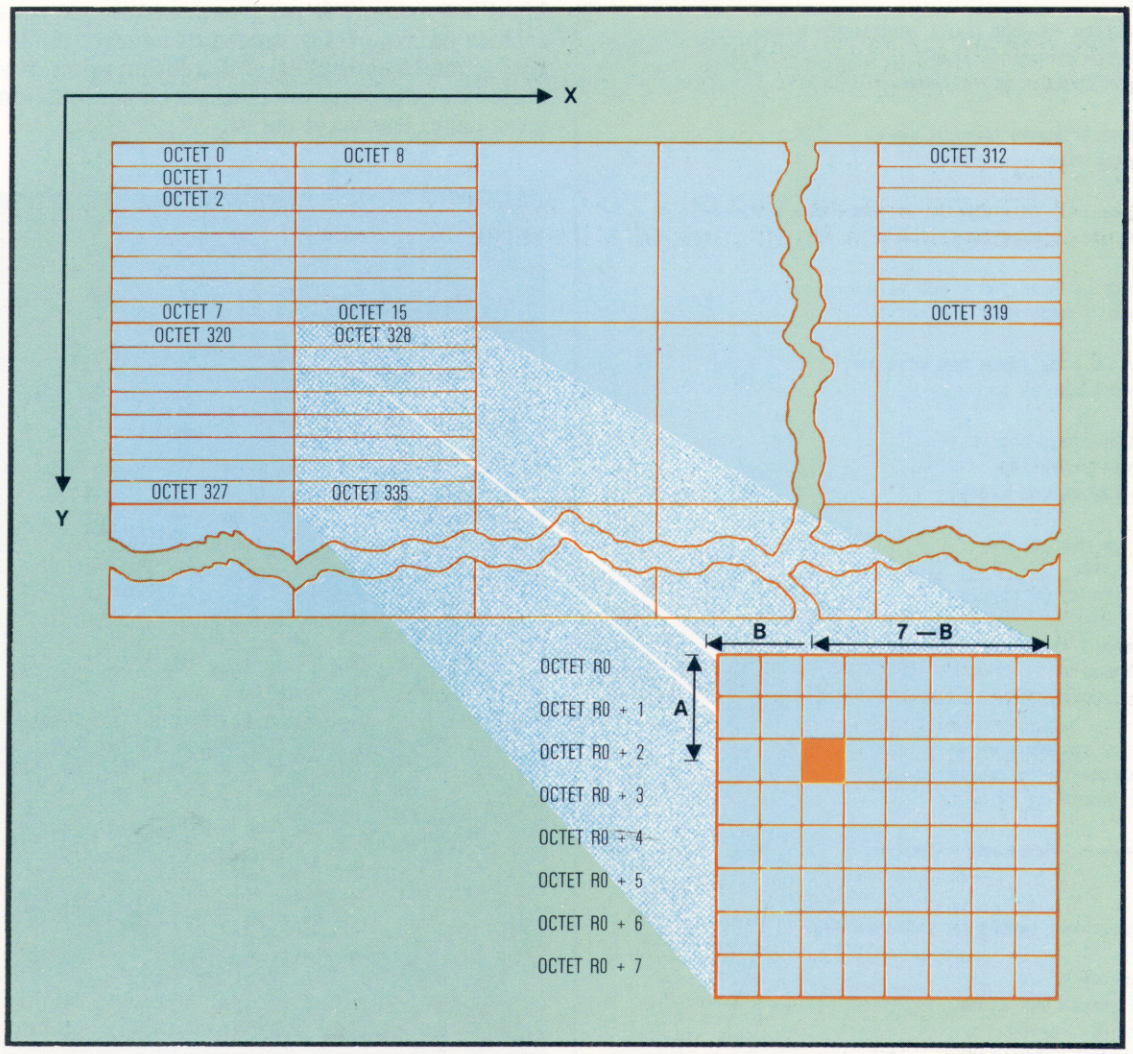


TEMPS D'EXÉCUTION = 1,85 s.



TEMPS D'EXÉCUTION : 89,6 s.

Liz Heaney



Point par point
Les pixels-points qui sont à la base du mode haute résolution du Commodore 64 ne sont pas directement accessibles. L'écran texte 25 × 40 est divisé en 8 000 octets de mémoire vive, chaque position étant décrite par 8 octets. Un pixel est défini par X, sa distance (en pixels) depuis la gauche de l'écran, et par Y, sa distance depuis le haut de l'écran. Ces nombres doivent être traduits à l'adresse de l'octet qui contient le pixel, et au numéro du bit approprié dans cet octet. (Cl. Liz Dixon.)

à partir de la coordonnée X, par la commande suivante (OH = octet horizontal) :

```
OH = INT(X/8)
```

De manière similaire, la coordonnée verticale (OV = octet vertical) peut être déduite de Y :

```
OV = INT(Y/8)
```

Le premier octet de la position, qui comporte le bit demandé, R0, peut être obtenu par OH et OV :

```
R0 = OV*320 + OH*8
```

L'octet qui contient le bit voulu est R0, auquel s'ajoute le reste de l'opération lorsque Y a été divisé par 8. Ce reste peut être facilement déterminé à partir des trois bits les plus à droite de la valeur de Y. Si A=Y AND 7 et si BASE est l'adresse du premier octet du bloc de 8 000 octets, alors l'adresse de l'octet, BY, qui contient le bit voulu peut être obtenue :

```
BY = BASE + R0 + A
```

Le bit de l'octet BY peut être obtenu en calculant le reste de l'opération lorsque la coordonnée X est divisée par 8. Si B=X AND 7, l'ordre POKE suivant sera attribué à l'un des bits correspondant au pixel de coordonnées X et Y :

```
POKE BY, PEEK(BY) OR (2↑(7-B))
```

Maintenant que chaque pixel peut être individuellement activé, il est possible de créer des routines pour dessiner à l'écran. Le programme suivant indique comment tracer des lignes à partir d'un point (X1,Y1), vers un autre (X2,Y2). Un cercle peut être obtenu en spécifiant les coordonnées de son centre (CX,CY), et le rayon (RA).

Il est intéressant de noter que ce programme consiste en des sous-programmes dépendants les uns des autres. La routine de plus bas niveau trace un seul point à l'écran. Elle est exploitée par une routine d'un degré supérieur qui trace une ligne droite. A un niveau encore au-dessus, la routine PLOT TRIANGLE utilise la routine PLOT LINE trois fois pour tracer les trois côtés du triangle. Cette approche de la programmation présente plusieurs avantages. Elle est souple puisqu'il serait facile de concevoir une routine pour tracer un hexagone régulier par exemple. Cette dernière appellerait la routine PLOT LINE, qui appellerait à son tour la routine PLOT POINT (tracer un point). On pourrait aussi utiliser la routine DRAW TRIANGLE afin de tracer un hexagone à partir de triangles équilatéraux. Dans ce cas, la routine DRAW HEXAGON serait un quatrième niveau de structure de programme. Ayez soin de sauvegarder ce programme par SAVE avant de l'exécuter, une instruction POKE erronée étant suscepti-

```
65 REM **** DEMONSTRATION DU MODE HAUTE RESOLUTION ****
70 PRINT CHR$(147) : REM EFFACER L'ECRAN
80 POKE 53280 ,0 : REM COULEUR NOIRE CADRE
90 :
100 REM **** ZONE MEMOIRE ECRAN COULEUR ****
110 FOR I=1024 TO 2023: POKE I,4: NEXT I
120 :
130 REM **** POSITIONNER LE POINTEUR DE LA TABLE DES BITS****
140 BASE =8192: POKE 53272. PEEK(53272) OR 8
150 :
160 REM **** SUPPRIMER LE MODE TABLE DES BITS ****
170 FOR I=BASE TO BASE + 7999:POKE I,0:NET I
180 :
190 REM **** ACTIVER LE MODE TABLE DES BITS **
200 POKE 53265, PEEK(53265) OR 32
210 :
220 REM **** TIRER UNE LIGNE DROITE ****
230 X1=20: X2=190: Y1=15: Y2:180
240 GOSUB 800 : REM TRACER UNE LIGNE
250 :
300 REM **** TRACER UN CERCLE ****
310 CX=150: CY=100: RA=60
320 GOSUB 600 : REM TRACER UN CERCLE
330 :
370 **** UN AUTRE CERCLE ****
380 CX=100: CY=60: RA=20
390 GOSUB 900: PLOT CIRCLE
400 :
410 REM **** TRACER UN TRIANGLE ****
420 XA=200:XB=250:XC=300:YA=50:YB=100:YC=80
430 GOSUB 600 : REM TRACER UN TRIANGLE
440 :
450 GOTO 450 : REM FIN DU PROGRAMME PRINCIPAL
460 :
470 :
600 REM **** SOUS_PROGRAMME TRACER UN TRIANGLE ****
610 :
620 X1=XA: X2=XB: Y1=YA: Y2=YB
630 GOSUB 800 : REM TRACER UNE LIGNE
640 X1=XB: X2=XC: Y1=YB: Y2=YC
650 GOSUB 800 : REM TRACER UNE LIGNE
660 X1=YC: X2=XA: Y1=YC: Y2=YA
670 GOSUB 800 : REM TRACER UNE LIGNE
680 RETURN
690 :
800 REM **** SOUS-PROGRAMME TRACER UNE LIGNE ****
810 S=1
820 IF X2<X1 THEN S=-1
830 FOR X=X1 TO X2 STEP S
840 Y=(Y2-Y1)*(X-X1)/(X2-X1)+Y1
850 GOSUB 1000 : REM TRACER UN POINT
860 NEXT X
870 RETURN
880 :
900 REM **** SOUS-PROGRAMME TRACER UN CERCLE ****
910 :
920 FOR ANGLE = 0 TO 2*PI STEP .04
930 X=INT (RA*COS(ANGLE)+CX)
940 Y=INT (CY-RA*SIN(ANGLE))
950 GOSUB 1000 : REM TRACER UN POINT
960 ANGLE SUIVANT
970 RETURN
980 :
1000 REM **** SOUS-PROGRAMME TRACER UN POINT ****
1010 :
1020 IF X>319 OR X<0 OR Y>199 OR Y<0 THEN
GOTO 1070
1030 OH=INT(X/8): OV=INT(Y/8)
1040 RO=OV*320+OH*8: A= Y AND 7: B=X AND 7
1050 BY=BASE+RO+A
1060 POKE BY, PEEK(BY)OR(2"(7-B))
1070 RETURN
```

ble d'interrompre le programme sans que vous sachiez pourquoi. Lorsque vous voudrez réintégrer le mode normal basse résolution après exécution de ce programme, utilisez en conjonction les touches Run/Stop et Restore.

Programme du jeu

Une partie importante du programme de jeu auquel nous travaillons est de mettre à jour le score par une routine spéciale. Il y a plusieurs manières d'attribuer les scores pour un jeu de cette sorte. Notre système sera fondé sur les règles suivantes :

1. La profondeur et la vitesse auxquelles se déplace le sous-marin sont des facteurs très importants. Le score attribué à un sous-marin devra en tenir compte.
2. Lorsqu'un sous-marin est touché, sa valeur vient s'ajouter au score du joueur qui a ainsi fait mouche. Mais si le sous-marin atteint le bord de l'écran sans dommages, sa valeur est retranchée du score du joueur. Les scores négatifs ne sont pas admis.

Nous traiterons plus loin de la routine qui choisit de manière aléatoire la vitesse et la profondeur. Pour l'instant, sachons simplement que la profondeur est déterminée par la variable Y3, et que la vitesse est donnée par la variable DX. C'est sur ces bases que se calcule la valeur du sous-marin. Afin de s'assurer que seules les valeurs entières sont retenues, la fonction INT sera utilisée de la sorte :

```
VALEUR DU SOUS-MARIN = INT (Y3+DX*30)
```

Le score courant du joueur est attribué à la variable SC. Il reste simplement à additionner ou à soustraire la valeur du sous-marin au score du joueur, selon que celui-ci a été respectivement détruit ou manqué. Le sous-programme UPDATE SCORE (mise à jour du score) est utilisé par deux parties du programme :

1. Lorsque la position du sous-marin est testée afin de savoir s'il a atteint le bord de l'écran.
2. Lors de la routine HIT.

Le drapeau DS peut être arboré pendant l'une de ces deux parties du programme pour indiquer quelle partie utilise le sous-programme UPDATE SCORE. En attribuant à DS la valeur 1 dans la routine HIT, et —1 dans la routine EDGE OF SCREEN (bord de l'écran), le score peut être additionné ou retranché selon la formule suivante :

```
SC = SC + INT (Y3+DX*30)*DS
```

Après avoir vérifié que le score n'est pas devenu négatif, la nouvelle valeur de SC (SCore), peut être affichée en haut de l'écran. Ajoutez les lignes suivantes à votre programme et sauvegardez à nouveau votre programme.

```
5500 REM ****MISE A JOUR DU SCORE****
5510 SC=SC+INT(Y3+DX*30)*DS
5520 IF SC < 0THEN SC=0
5530 PRINT CHR$(19);CHR$(144);"SCORE";
SC;CHR$(157);" "
5540 RETURN
```

Dernier entré, premier sorti

La pile est une zone définie de mémoire d'ordinateur liée à l'UC, qui sert d'espace de travail. On y accède par des instructions de piles, qui permettent de copier et de restaurer le contenu du registre.

La gestion de mémoire est l'essentiel de la programmation en langage d'assemblage. La plupart des instructions que nous avons étudiées jusqu'à présent concernent le chargement de données vers ou en provenance des emplacements mémoire. Ces emplacements sont accessibles de diverses manières — modes d'adressage — mais les instructions en question prenaient toujours une adresse mémoire pour une partie de l'opérande. Il existe un ensemble d'instructions, cependant, qui ont accès à une zone spécifique de mémoire, mais ne prennent pas une adresse comme opérande. Ces instructions opèrent sur la zone de mémoire appelée *pile,* et on les désigne par opérations de pile.

La pile est prévue à la fois pour l'UC et pour le programmeur comme mémoire de travail temporaire. C'est un peu comme un bloc-notes sur lequel on écrit, lit et efface facilement. Les opérations de pile copient des données des registres d'UC dans des zones vacantes de la pile, ou bien recopient des données de la pile en retour dans les registres d'UC. Ces instructions n'ont pas besoin d'opérande adresse, parce qu'un registre d'UC spécifié, le *pointeur de pile,* contient toujours l'adresse du prochain emplacement de pile libre. Ainsi, tout ce qui est écrit sur la pile est automatiquement transcrit dans l'octet indiqué par le pointeur de pile, et les données chargées à partir de la pile sont toujours copiées en commençant par le dernier emplacement de pile sur lequel on a écrit.

Dans les systèmes 6502, la pile comporte les 256 octets de RAM compris entre $0100 et $01FF; sur Z80, l'emplacement et la taille de la pile sont déterminés par le système d'exploitation, mais peuvent être modifiés par le programmeur. Cette variation reflète les différences d'organisation interne des deux microprocesseurs : le 6502 a un pointeur de pile à un seul octet, tandis que le pointeur de pile du Z80 en a deux.

Le contenu du pointeur de pile du 6502 est traité par l'UC comme l'octet lo de l'adresse de pile, et un octet hi de $01 y est automatiquement ajouté grâce à un « neuvième bit » câblé dans le pointeur de pile. Ce bit supplémentaire est toujours égal à un, de sorte que les adresses de pile 6502 sont toutes en page une.

Le pointeur de pile de Z80 est un registre à deux octets capable d'adresser tout emplacement compris entre $0000 et $FFFF — la totalité de l'espace d'adressage du Z80. La pile peut donc être placée en n'importe quel endroit de la RAM, et son emplacement peut être modifié par le programmeur. Ce n'est cependant pas recommandé, puisque le système d'exploitation donne initialement un emplacement à la pile et y stocke les données.

Comme le système d'exploitation peut à tout instant interrompre l'exécution d'un programme en langage machine, et qu'il s'attend à trouver les données correspondant à cette opération sur la pile, toute altération de l'emplacement de la pile signifiera que les données ne seront plus disponibles et le système pourra « se planter » dit-on dans notre jargon.

Considérons la routine suivante, qui échange les contenus de deux emplacements mémoire :

6502		Z80	
LDA	LOC1	LD	A,(LOC1)
PHA		PUSH	AF
LDA	LOC2	LD	A,(LOC2)
STA	LOC1	LD	(LOC1),A
PLA		POP	AF
STA	LOC2	LD	(LOC2),A

Le contenu de LOC1 est chargé dans l'accumulateur et, de là, copié ou « refoulé » (PUSH) sur la pile. Le contenu de LOC2 est ensuite chargé dans l'accumulateur et stocké en LOC1. Le contenu de l'octet supérieur de la pile est alors copié ou « ressorti » (POP) vers l'accumulateur, qui restaure le contenu originel de LOC1 dans l'accumulateur. Celui-ci est copié en LOC2, et l'échange est complet. Notez que les opérations de pile ont sauvegardé le contenu de LOC1 en mémoire aussi longtemps que nécessaire, mais sans que le programme ne spécifie aucun emplacement de mémoire — sauf, par implication, le dernier emplacement libre sur la pile.

Ce fragment de programme montre beaucoup de choses sur les opérations de pile. Premièrement, elles sont réciproques et séquentielles. Le dernier article entré dans la pile est récupéré le premier. Si l'on entre plusieurs articles successivement sans en sortir, ceux-ci seront écrits l'un « au-dessus » de l'autre, et en sortie on aura accès aux articles du « haut » vers le « bas » de la pile.

Pour visualiser la pile, imaginez que vous écriviez des cartes postales et que vous les empiliez à côté de vous sur votre bureau. Puis vous les lisez et les écartez jusqu'à ce que la pile soit épuisée. La dernière carte écrite restant sur la pile est toujours celle du dessus. C'est pourquoi la pile est désignée comme structure de données LIFO (Last In First Out = dernier entré premier

sorti). Au contraire, une structure FIFO (First In First Out = premier entré premier sorti) est une queue. Par convention, on parle du premier octet libre de la pile comme du « haut » de la pile et l'on imagine que la pile se développe vers le haut. Cependant, dans le Z80 et le 6502, le pointeur de pile est décrémenté à chaque entrée, de sorte que le haut de la pile a, en fait, une adresse inférieure au bas. C'est plus facile à comprendre si l'on décrit la pile comme « s'élevant vers zéro ».

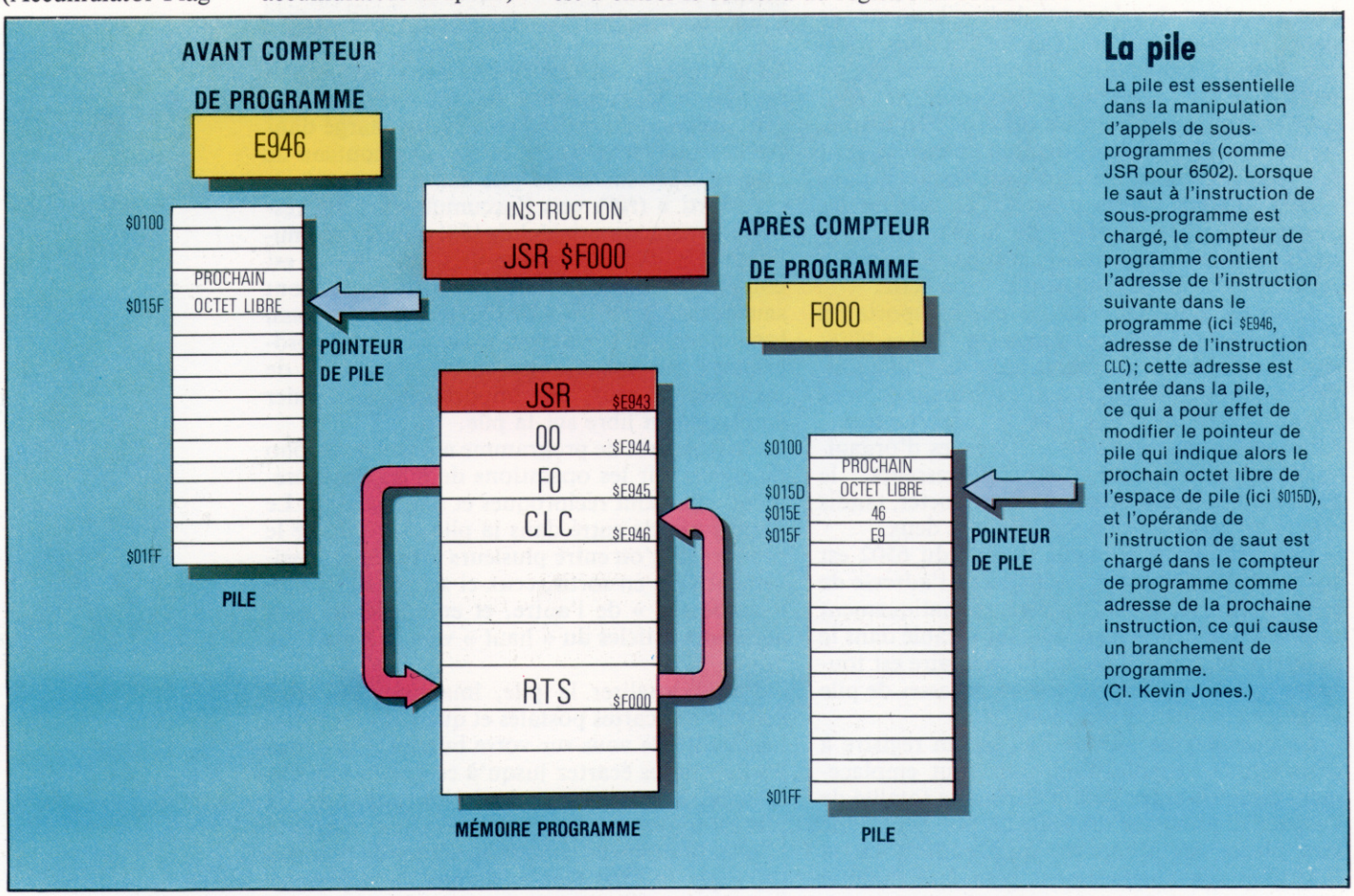
Le premier fragment de programme illustre l'utilisation de la pile, du fait que le nombre d'instructions qui y sont entrées est exactement contrebalancé par le nombre de sorties. Cela n'est pas essentiel, mais si l'on n'y prête pas attention, en écrivant des sous-programmes, il peut y avoir des erreurs de retour, d'où des défaillances du programme. C'est l'une des erreurs les plus communes en programmation en langage d'assemblage, mais elle peut être aisément décelée en comparant le nombre d'instructions d'entrée et de sortie.

La version Z80 de la routine diffère sensiblement de celle de 6502, du fait que le 6502 n'entre que des registres à un octet dans la pile, alors que le Z80 entre toujours un registre à deux octets. Quand on entre ou sort l'accumulateur Z80, on fait de même pour le contenu du registre d'état du processeur (PSR), car l'UC traite ces deux registres à un octet comme un seul registre à deux octets appelé AF (Accumulator-Flag = accumulateur-drapeau). La puissance du Z80 provient en grande partie de sa capacité à manipuler des registres à deux octets.

C'est une bonne habitude de programmation que de commencer les sous-programmes en entrant le contenu de tous les registres d'UC dans la pile, puis de les ressortir juste avant le retour du sous-programme.

	6502		Z80	
	LDA	LOC1	LD	A,LOC1
SUM	ADC	#$6C	ADC	A,$6C
GSUB	JSR	SUBR0	CALL	SUBR0
TEST	BNE	SUM	JR	NZ,SUM
EXIT	RTS		RET	
SUBR0	PHP		PUSH	AF
	PHA		PUSH	HL
	TXA		PUSH	DE
	PHA		PUSH	BC
	TYA		PUSH	IX
SUBR1	PHA		PUSH	IY
SUBR2	STA	LOC2	LD	(LOC2),A
	LDA	#$00	LD	A,$00
SUBR3	PLA		POP	IY
	TAY		POP	IX
	PLA		POP	DE
	TAX		POP	BC
	PLA		POP	HL
SUBR4	PLP		POP	AF
	RTS		RET	

Ici, l'effet des instructions entre SUBR0 et SUBR1 est d'entrer le contenu du registre en cours dans



La pile

La pile est essentielle dans la manipulation d'appels de sous-programmes (comme JSR pour 6502). Lorsque le saut à l'instruction de sous-programme est chargé, le compteur de programme contient l'adresse de l'instruction suivante dans le programme (ici $E946, adresse de l'instruction CLC); cette adresse est entrée dans la pile, ce qui a pour effet de modifier le pointeur de pile qui indique alors le prochain octet libre de l'espace de pile (ici $015D), et l'opérande de l'instruction de saut est chargé dans le compteur de programme comme adresse de la prochaine instruction, ce qui cause un branchement de programme.
(Cl. Kevin Jones.)

la pile, et l'effet des instructions entre SUBR3 et SUBR4 est de restituer ce contenu aux registres. Les deux instructions importantes du sous-programme sont celles qui commencent en SUBR2, mais la seconde n'a pas d'effet puisque les instructions suivantes changent complètement l'état de l'accumulateur.

Notez que les instructions Z80 PUSH (entrer) et POP (sortir) peuvent prendre pour opérande n'importe quelle paire de registres, alors que le 6502 ne peut opérer que sur l'accumulateur (PHA et PLA) et le PSR (PHP et PLP). D'où la nécessité des transferts de registre-accumulateur (TXA, TAX, TYA, TAY) dans la version 6502.

Notez également que nous avons fait exprès une erreur dans la version Z80, qui consistait à ne pas sortir tous les registres dans l'ordre inverse à celui des entrées. Cela illustre le soin indispensable à porter aux opérations de pile, mais démontre aussi qu'on peut entrer dans la pile un registre, puis sortir de la pile cette même valeur pour la placer dans un registre différent — procédé laborieux mais parfois commode pour faire des transferts de données entre registres. Les fonctions et usages des registres d'UC seront vus plus tard, et nous conclurons notre étude générale des instructions de langage d'assemblage. Nous commencerons alors l'étude de l'arithmétique du langage machine.

Exercices

1. Réécrire la seconde routine donnée dans la solution des précédents exercices, de façon que le message en LABL1 soit à nouveau stocké en LABL1, mais dans l'ordre inverse, comme ceci :

LABL1 EGASSEM NU TSE ICEC

Utiliser la pile pour cette inversion.

2. Développer cette routine de sorte que les mots du message restent dans leur ordre originel, mais que les caractères de chaque mot soient inversés, comme ceci :

LABL1 ICEC TSE NU EGASSEM

Solutions des exercices précédents

1. Ce sous-programme stocke les nombres de $0F à $00 par ordre décroissant dans le bloc de $10 octets réservés pour le pseudo-op en LABL1.

6502		Z80	
ORIGIN	ORG $7000	ORIGIN	ORG $C000
LABL1	DS $10	LABL1	DS $10
LABL2	DW $700	LABL2	DW $C100
		OFFST	EQU $0F
BEGIN	LDY #$FF	BEGIN	LD IX,LABL1
	LDX #$10		LD B,OFFST
LOOP0	INY	LOOP0	LD (IX+0),B
	DEX		INC IX
	TXA	ENDLP0	DJNZ LOOP0
	STA LABL1,Y		LD (IX+0),B
ENDLP0	BNE LOOP0		RET
	RTS		

Les différences d'approche et d'instructions entre Z80 et 6502 sont révélatrices. Le 6502 utilise le registre Y comme index pour l'adresse LABL1, et le registre X comme compteur de boucle et source des données à stocker. Notez que le registre X est décrémenté de deux instructions avant le test BNE en ENDLP0, mais, du fait que TXA (transfert du contenu de X vers l'accumulateur) et STA n'affectent pas le PSR, le test a pour effet de décrémenter X.

La version Z80 utilise l'adressage indirect de IX pour l'adresse de stockage, et le registre B comme compteur et source de données. En ENDLP0, dans la version Z80, nous voyons DJNZ LOOP0, qui signifie « décrémenter le registre B, et faire un saut relatif jusqu'à LOOP0 si le résultat n'est pas nul ».

2. Cette routine copie le message stocké en LABL1 vers le bloc commençant à l'adresse stockée en LABL2. La valeur S0D (code ASCII pour RETURN ou ENTER) est stockée à la fin du message.

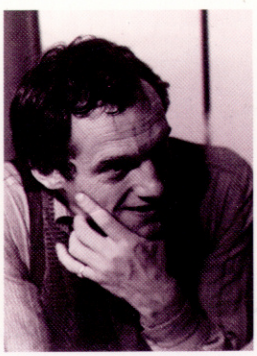
6502		Z80	
ORIGIN	ORG $7000	ORIGIN	ORG $C000
LABL1	DB « CECI EST UN MESSAGE »	LABL1	DB « CECI EST UN MESSAGE »
TERMN8	DB S0D	TERMN8	DB S0D
LABL2	DW $7100	LABL2	DW $C100
CR	EQU $0D	CR	EQU $0D
ZPL0	EQU $FB		
BEGIN	LDA LBL2	BEGIN	LD IX,LABL1
	STA ZPL0		LD IY,(LABL2)
	LDA LABL2+1	LOOP0	LD A,(IX+0)
	STA ZPL0+1		LD (IY+0),A
	LDY $FF		INC IX
LOOP0	INY		INC IY
	LDA LABL1,Y		CP CR
	STA (ZPL0),Y	ENDLP0	JR NZ,LOOP0
	CMP CR		RET
ENDLP0	BNE LOOP0		
	RTS		

La version 6502 utilise le registre Y comme index pour l'adresse indirecte ZPL0, en mode indirect postindexé. Ce mode n'est possible qu'avec le registre Y, et requiert une adresse d'opérande en page zéro — d'où l'initialisation de ZPL0 et ZPL0+1 avec l'adresse stockée en LABL2. Le système d'exploitation des machines 6502 utilise la plupart des emplacements en page zéro, mais les adresses $FB à $FF sur le Commodore 64, et $70 à $8F sur le BBC Micro sont inusitées, c'est pourquoi on met ZPL0 dans l'un d'eux.

Les deux routines utilisent une instruction « comparer l'accumulateur » — CMP CR (6502) et CP CR (Z80) — dans laquelle l'opérande est soustrait du contenu de l'accumulateur, ce qui affecte les drapeaux du PSR. Le contenu de l'accumulateur est alors rétabli, tandis que le PSR donne les résultats de la comparaison. Lorsque l'accumulateur contient S0D (fin du message), le résultat de la comparaison sera le drapeau de zéro. Ainsi le test ENDLP0 ne sera pas vérifié.

Cambridge connection

La ville de Cambridge deviendrait-elle l'équivalent britannique de la Silicon Valley? Des firmes connues, Sinclair Research ou Acorn, et d'autres, qui le sont moins, comme Camputers, y sont déjà installées.



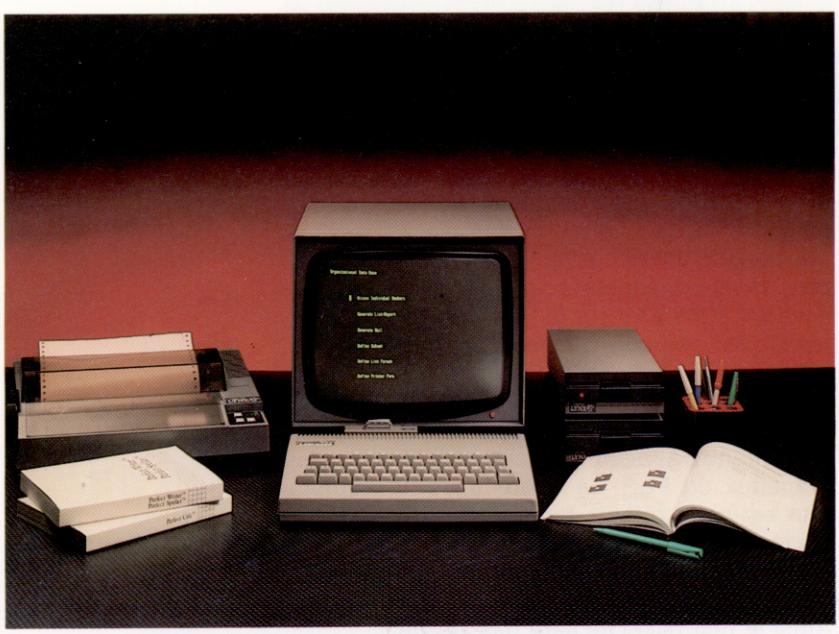
John Shirreff

Camputers fut créé par un seul homme. David Greenwood commença en 1976 une carrière de concepteur en électronique, travaillant en indépendant pour des firmes comme Pye Telecoms. Quelques années plus tard, il formait sa propre compagnie, spécialisée dans des travaux de développement plus élaborés. Mais il gardait toujours une base contractuelle : Greenwood aborda ainsi le domaine du logiciel, et rédigea un programme de gestion permettant aux brasseries de contrôler leurs ventes aux débits de boissons.

La firme entreprit ensuite de créer un micro-ordinateur. Les premiers projets eurent recours au microprocesseur Z80 de Zilog. En février 1981, Greenwood mit sur pied Camtronic Circuits (qui devint plus tard Camputers), et, grâce à un des prêts que le gouvernement britannique consentait aux petites entreprises, il commença à travailler sur le Lynx dès l'été suivant. Son but avoué était alors « d'apprendre au Z80 A à jongler avec les problèmes, et non à les bousculer ».

John Shirreff fut chargé de toutes les questions relatives au matériel; c'était un diplômé de l'université de Cambridge, mais il avait travaillé dans les milieux de la musique rock avant de rejoindre la compagnie. En Grande-Bretagne, cela ne surprend personne. David Jansons se vit confier la création de logiciels. Il écrivit la version du BASIC utilisée par l'appareil.

Ordinateur de gestion
On voit ici le Lynx Laureate, un système modulaire destiné à la petite gestion. Il peut recourir au système d'exploitation CP/M grâce à 128 K de mémoire interne. (Cl. Camputers.)



Le Lynx fut commercialisé en 1982. Il avait une allure très professionnelle : un boîtier gris très attrayant, un véritable clavier type machine à écrire, une mémoire de 48 K (extensible à 192 K), huit couleurs, une haute résolution de 248 × 256, ainsi qu'un haut-parleur intégré afin de tirer le meilleur parti de ses possibilités sonores.

Malheureusement le Lynx ne connut jamais un très grand succès en Grande-Bretagne, bien que les ventes à l'étranger aient été assez encourageantes pour que Camputers décide de mettre au point une version plus élaborée de ce modèle de base. Ce fut, peu après, le Lynx 96, qui avait plus de 37 K de RAM directement accessibles à l'utilisateur; il pouvait recourir aux disquettes 5 pouces, et se voyait pourvu d'effets sonores prédéfinis. Des interfaces (série et parallèle) pour imprimante étaient disponibles en option.

Problèmes financiers

Plus récemment, la compagnie a mis en vente le Lynx Laureate en visant le marché de la petite gestion. Construit autour du Z80, l'appareil peut accueillir le système d'exploitation CP/M; l'utilisateur a donc accès aux innombrables logiciels qui ont été écrits pour lui au cours de ces dix dernières années. C'est là un argument de vente important, la micro-informatique de gestion étant inconcevable sans programmes appropriés. Le Laureate est par ailleurs compatible avec les deux modèles précédents, et un bus d'extension à quarante voies autorise l'usage de très nombreux périphériques : imprimante en parallèle, manche à balai, logiciels implantés sur cartouche ROM.

Camputers — actuellement présidée par Stanley Charles — a également l'intention de lancer sur le marché une nouvelle version du Laureate, cette fois sous forme intégrée (et non plus modulaire, comme c'est le cas actuellement). Le Lynx 48 devrait ressortir en Grande-Bretagne, sous le nouveau nom de Leisure : la compagnie est en effet désireuse de s'implanter véritablement sur le marché de la micro-informatique domestique. Les chercheurs travaillent aussi à une machine qui devrait pouvoir concurrencer le QL de Sinclair. Mais les graves difficultés que rencontre Camputers (on parle d'un déficit d'un million de livres sterling) risquent de faire s'évanouir tous ces projets! Ce n'est d'ailleurs pas la seule société britannique d'informatique qui rencontre des difficultés.

PROGRAMME N° 21

GRAPHIQUE HAUTE RÉSOLUTION (suite)

Nous avons introduit dans notre programme n° 20 l'instruction HGR, qui permet de travailler en haute résolution graphique. Nous poursuivons aujourd'hui avec un programme permettant, à la manière du précédent, de dessiner une figure différente avec des tailles différentes.

```
 5   REM DESSIN AVEC ECHELLE
10    DATA  0,10,10,0,0,20,10,0
20    DATA  0,-10,20,0,0,10
30    DATA  10,0,0,-20,3,0,0,-3
40    DATA  _3,0,0,3,-30,0,0,-10
50    DATA  -20,0
60    DATA  99,99
70    HGR
80    HCOLOR=3
90   A1 = 50:B1 = 50:K = 1
100   GOSUB 200
110   RESTORE
112  A1 = 120:B1 = 70:K = 0.5
114   GOSUB 200
120   INPUT "TAPEZ RETURN POUR CON
      TINUER ";Z$
130   TEXT
140   END
200   FOR I = 1 TO 1000
210   READ DX: READ DY: IF DX = 99
       AND DY = 99 THEN 270
220  A2 = A1 + DX * K
230  B2 = B1 + DY * K
240   HPLOT A1,B1 TO A2,B2
250  A1 = A2:B1 = B2
260   NEXT I
270   RETURN
```

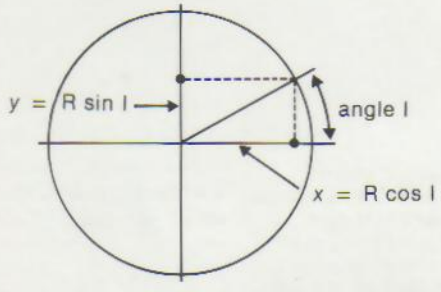
Les figures résultant de ce petit programme ne doivent présenter aucune imperfection ou discontinuité.
Voyons maintenant le tracé de quelques figures géométriques.

Cercle ou ellipse

Il n'existe pas d'instruction en BASIC Apple Soft de tracé de cercle. Il faut tracer celui-ci, point par point ou par segments droits.
On calcul les coordonnées X et Y d'un cercle de rayon R par l'équation d'un cercle en coordonnées polaires (I étant l'angle) :

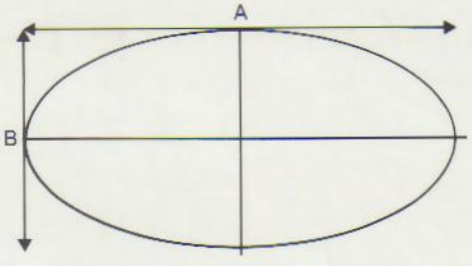
$x = r \cos I$
$y = r \sin I$

en faisant varier l'angle de 0 à 2 π, c'est-à-dire de 0 à 6,28 rad



Pour une ellipse, l'équation en coordonnées polaires donne :

$x = (A/2) \cos I$
$y = (B/2) \sin I$



Si on remplace A/2 et B/2 par R, un rayon, on retrouve bien entendu l'équation d'un cercle.

```
10    HGR
20    HCOLOR= 3
30   L = 100:H = 50:A1 = 100:B1 = 1
       00
40    GOSUB 80
```

Programmes : D. Le Goff S.I.I.

```
50   GET Z$
60   TEXT
70   END
80   FOR A = 0 TO 6.29 STEP .1
90  C = A1 + (L / 2) *  COS (A)
100  B = B1 + (H / 2) *  SIN (A)
110   HPLOT C,B
120   NEXT A
130   RETURN
```

On peut accélérer le tracé du cercle ou de l'ellipse en dessinant ceux-ci, non plus point par point mais par segments de droites. Naturellement, la définition est moins bonne mais le tracé est beaucoup plus rapide.

```
10   REM  PASSAGE EN HGR ET CHOIX
      DE LA COULEUR
20   HGR
30   HCOLOR= 3
40   REM SAISIE DE LA LONGUEUR ET
       DE LA HAUTEUR
50  L = 100
60  H = 50
70   REM  SAISIE DES COORDONNEES D
      U CENTRE
80  XC = 90
90  YC = 120
91   GOSUB 100
92   GET Z$
93   TEXT
94   END
95   REM  TRACE
100  XA = XC + L / 2
110  YA = YC + L / 2
120   FOR A = 0 TO 6.29 STEP .1
130  X = XC + (L / 2) *  COS (A)
140  Y = YC + (H / 2) *  SIN (A)
150   HPLOT XA,YA TO X,Y
160  XA = X
170  YA = Y
180   NEXT A
190   RETURN
```

Voyons maintenant quelques dessins simples.
Tracé d'un œuf :

```
10   REM  DESSIN D UN OEUF
20   REM  PASSAGE EN HGR
30   HGR
40   REM  CHOIX DE LA COULEUR
50   HCOLOR= 6
60   REM  COORDONNEES DU CENTRE
70  XC = 100
80  YC = 60
90   REM  SAISIE DE LA LONGUEUR ET
       DE LA LARGEUR
100  L = 60
110  H = 180
120   REM  TRACE
130   FOR A = 0 TO 3.14 * 2 STEP 3
      .14 / 50
140  X = XC + L / 2 *  COS (A)
150  Y = YC + L / 2 *  SIN (A)
160   HPLOT XC,YC + H / 2 TO X,Y
170   NEXT A
180   INPUT "TAPEZ RETURN POUR CON
      TINUER";Z$
190   TEXT
```

Tracé de rectangles emboîtés :

```
20   HGR
30   HCOLOR= 3
40  I = 0.2
50  XA = 100
60  YA = 100
70  XB = 60
80  YB = 60
90   FOR B = 1 TO 20
100   HPLOT XA + XB,YA - YB
105   HPLOT  TO XA + XB,YA + YB
110   HPLOT  TO XA - XB,YA + YB
120   HPLOT  TO XA - XB,YA - YB
130   HPLOT  TO XA + XB,YA - YB
140  XB = XB - (XB - YB) * I
150  YB = YB - (XB + YB) * I
160   NEXT B
```

Tracé d'un soleil :

```
10   HGR
20   HCOLOR= 3
30  XA = 90
40  YA = 90
50   FOR I = 0 TO 6.29 STEP 3.14 / 50
60  X = 30 *  COS (I) + XA
70  Y = 30 *  SIN (I) + YA
80   HPLOT XA,YA TO X,Y
90   NEXT I
200   FOR I = 0 TO 6.29 STEP 3.14 / 20
210  Z = 30 + 40 *  RND (1)
220  X = Z *  COS (I)
230  Y = Z *  SIN (I)
240   HPLOT XA,YA TO XA + X,YA + Y
250   NEXT I
```