

MICRO

M A G

LISTING BIDUILLE PROGRAMMATION

CPC

- «Death's Ticket», shoot'em up d'enfer !

ST

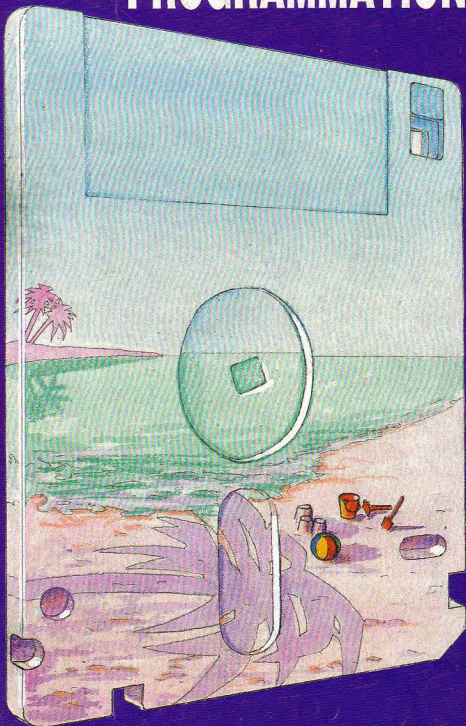
- Objets fractals & récursivité

AMIGA

- «Animatix», donnez vie à vos sprites

PC

- Manette des gaz pour simulateurs de vol



M 1729 - 14 - 25.00 F



SOMMAIRE

JUILLET
A O U T

1990 N°14

CPC

Programmation	<i>Symbol & Symbol after</i>	8
Listing	<i>Amsaisie V.2</i>	14
	<i>Death's Ticket</i>	15
	<i>La Molécule</i>	24
	<i>Mano Négra</i>	32

ST

Initiation	<i>Les modes d'adressage (suite)</i>	40
Programmation	<i>La récursivité</i>	44

AMIGA

Initiation	<i>Les modes d'adressage (suite)</i>	40
Listing	<i>Scopy</i>	50
	<i>Animatix</i>	52

PC

Bidouille	<i>Manette des gaz</i>	58
Listing	<i>Mad Dav</i>	60

Divers

Minitel & Téléchargement	6
--------------------------------	---

EDITO

Si vous avez le choix, ne partez pas en vacances.



N'apportez pas votre tribut au saccage du littoral. Evitez les joies illusives d'un brusque dépaysement. Réprimez vos envies, sources de dépenses somptuaires. La drague est risquée, le soleil malsain, les plages polluées et l'ambiance estivale sujette à caution.

Dédaignez à juste titre l'acquis du front populaire, ou plutôt, usez-en à rentabiliser votre matériel micro.

Œuvrez à nous fournir dès la rentrée, de quoi alimenter nos chères rubriques...

Jean-Claude Paulin, Tahiti.

Directeur de la publication: Jean Kaminsky.

REDACTION

Rédacteur en chef de ce numéro:
Jean-Claude Paulin.

Ont collaboré à ce numéro: David Farenzena, Claude Le Moullec, Daniel Provenier, Stéphane Rodriguez, Sébastien Röyer, Jean-François Six, Jean-Yves Trétout.

Couverture: Claude Marrel.

Maquettistes: Jean-Claude Paulin (PAO), Jean-jacques Galmiche.

ADMINISTRATION

Abonnements: Laser Presse OGP - 175, av. Jean-Jaurès, 75019 Paris. Tél. : (1) 42 41 30 10 de 8h 30 à 18h 00 du lundi au vendredi.
Comptabilité: Sylvie Kaminsky.

REGIE PUBLICITAIRE

NEO-MEDIA, 5-7, rue de l'Amiral Courbet, 94160 Saint-Mandé. Tél. : (1) 43 98 22 22.

Chef de publicité: Pascale Kittel.

MICRO-MAG

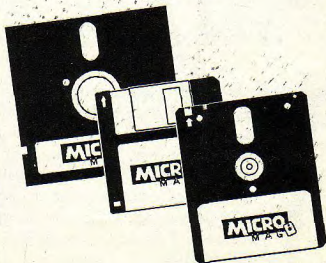
est édité par Laser Presse SA, 5-7, rue de l'Amiral Courbet, 94160 Saint-Mandé.

Commission paritaire: n°71178.

Dépôt légal: 3e trimestre 1990.

Impression: RBI.

Consultez notre Logithèque
Page n° 66



Télématique tac toc !

MINITEL ET TÉLÉCHARGEMENT

Le matériel de base est simple, vu qu'il ne nécessite qu'un ordinateur muni d'une prise RS 232, un minitel (1b) et le câble de liaison entre les deux. Pour la partie logicielle, il convient de se procurer un soft permettant la communication entre l'ordinateur et le serveur de votre choix. En effet, les différents serveurs n'utilisent pas tous le même protocole. Le matériel réuni, détaillons-en le fonctionnement (Les personnes réfractaires aux beautés de la technique moderne, peuvent se référer directement à l'encadré en fin d'article).

La RS232

Elle est une méthode de transmission qui permet d'échanger des informations de façon fiable, avec seulement 3 fils qui sont :

- Txd pour l'émission.
- Rxd pour la réception.
- Une masse (0V) pour servir de référence.

Le standard RS 232 complet comporte une quinzaine de fils avec des timings précis qu'il serait trop long de décrire en détail. Dans le cas d'une liaison ordinateur-minitel, les signaux autres que Txd, Rxd et Ov, sont fixés à certaines valeurs assurant le bon fonctionnement de l'ensemble. Une question se pose : comment faire transiter des octets

Reliée à tout ordinateur, une petite merveille nommée Minitel permet d'accéder à de nombreux serveurs et à leurs immenses banques de programmes. Bon voyage au pays du téléchargement...

comportant 8 bits par un seul fil ? La solution consiste à les envoyer les uns après les autres, en série. C'est pourquoi la RS 232 est parfois nommée «prise série». Mais ce n'est pas le tout de transmettre, encore faut-il pouvoir effectuer un minimum de contrôles. C'est pourquoi la méthode finalement adoptée est la suivante :

- 1 ou 2 bits de début.
- Le nombre de bits de données (7 pour de l'ASCII, 8 pour du binaire pur).
- 1 bit de parité paire ou impaire.
- 1, 1,5 ou 2 bit de stop.

Il ne reste plus qu'à fixer une vitesse de transmission, en bauds ou bits/seconde. Les échanges entre l'ordinateur et le minitel se font à 1200 bauds. Heureusement, la quasi-totalité des programmes de téléchargement liés à un serveur particulier prennent en charge la configuration de la RS232 et il n'y a pas à s'en préoccuper. En considérant qu'il faut transmettre une dizaine de bits par octet, on obtient, à 1200 bauds, une vitesse de 120 octets/seconde. Soit environ 7 Ko/minute.

Le minitel

L'intérêt essentiel du minitel est d'inclure une carte MODEM qui permet de faire transiter des signaux informatiques par les lignes téléphoniques. En effet, les signaux carrés, générés par la RS 232 de l'ordinateur, seraient complètement déformés à cause de la faible bande passante du réseau. Pendant un téléchargement, le minitel fait seulement office de transmetteur. Pour jouer ce rôle, il utilise le standard V23, c'est-à-dire qu'il émet vers le serveur à 75 bauds et reçoit à 1200. En revanche, la liaison avec l'ordinateur se fait à une vitesse fixe de 1200 bds dans les deux sens. Les minitels 1b ont un MODEM retournable. Dans ce mode, ils peuvent émettre à 1200 bds et recevoir à 75. C'est très pratique quand on veut envoyer des programmes (upload) au lieu de recevoir (download).

Le câble

Sa fonction est bien entendu de relier le minitel à l'ordina-

teur (ou le contraire). Mais il joue aussi un autre rôle. Le voltage des signaux émis par une RS232 classique n'est pas compatible avec ce qu'attend la prise péri-informatique du minitel. On trouve donc, dans le capot de la prise RS232, un petit montage destiné à adapter les tensions. Le branchement ne présente aucune difficulté si ce n'est que la prise minitel est parfois dissimulée par un petit volet. A noter qu'un tel câble permet également de s'envoyer des programmes entre copains. Il suffit que l'un des deux se transforme en serveur... Pour ce faire, il n'y a qu'à taper «fnct m r» sur un minitel 1b qui retourne alors son MODEM. Ensuite, un bon programme de comm. et c'est tout...

Le logiciel

Comme d'habitude, le soft est une composante fondamentale. C'est lui qui établit la synchronisation du téléchargement entre le serveur et votre ordinateur. Il veille également à son bon déroulement en faisant des vérifications régulières. A cet effet, les données sont découpées en paquets dont chacun fait l'objet d'un contrôle. Le soft peut ainsi demander au serveur la rémission d'un bloc déficient. Le mode d'emploi est très simple, il suffit de mettre le soft en attente de données. A l'aide du minitel, on se promè-

ne dans le serveur jusqu'à être séduit par un logiciel quelconque. On ordonne alors le téléchargement, le soft de communication le détecte, enregistre et sauve sur disque. C'est tout!

Compactage et décompactage...

...sont les deux mamelles du téléchargement. En effet, il est important de limiter au maximum le temps de connexion et c'est ainsi que la plupart des programmes présents sur un serveur sont compactés. Par exemple, sur Micromag (le serveur), ils portent alors le suffixe «.ARC», du nom du célèbre programme d'archiva-

ge utilisé. Une fois que vous avez téléchargé un «.ARC», il faut lancer le programme de décompactage présent sur la disquette du kit de téléchargement. Après cette dernière étape, votre logithèque s'est enrichie d'un ou plusieurs nouveaux titres car un seul «.ARC» peut contenir différents fichiers.

Conclusion

Il y a encore des foules de choses à dire sur les activités liées au mariage ordinateur-minitel et ce sujet fera bientôt l'objet d'un dossier complet. Disons seulement que le téléchargement et la possibilité pour tous de communiquer à distance via le minitel,

Comment télécharger?

- Relier le câble de liaison entre la prise RS232 (prise série) de votre ordinateur et la prise péri-informatique du minitel.
- Connectez-vous au serveur et mettez le programme de téléchargement en attente.
- Demandez au serveur d'envoyer le logiciel ou les données que vous désirez. Le téléchargement se fait automatiquement.
- Si votre fichier porte le suffixe «.ARC», il faut le décompacter. Lancez alors le programme de décompactage fourni avec votre kit de téléchargement.
- Voilà, c'est tout et ce logiciel n'a coûté que la communication.

ouvrent de nouveaux espaces d'applications à la micro-informatique, l'empêchant ainsi de se refermer sur elle-même (aïe!). Citons les messageries nationales et internationales,

le travail à domicile, la possibilité de diffuser vos logiciels...etc.

SYMBOL & SYMBOL AFTER

LES COURS DU PROFESSEUR ALI GATOR

Depuis des lustres, vous savez redéfinir un caractère avec Symbol. Le principe est simple: un numéro de caractère, 32 à 255, suivi de 8 nombres de 0 à 255, donnent une nouvelle valeur aux 8 octets réservés en ROM pour chaque caractère. Cet espace se situe, pour ceux que cela intéresse, aux adresses &3800 à &3FFF. La ROM n'étant pas modifiable, l'opération a lieu en RAM. C'est là qu'intervient la commande *Symbol After*.

Symbol After

En fait, il s'agit d'une barrière qui permet de réserver, en haut de la RAM, un espace suffisant pour la redéfinition des caractères. Celle-ci n'est pas nécessaire lorsque les caractères redéfinis ont une valeur supérieure à 240. Effectivement, entre &AB80 et &ABFF, figure une zone permettant le stockage et la redéfinition des caractères 240 à 255. Une commande similaire, *Memory*, dresse elle aussi une barrière entre l'espace réservé au basic et celui dédié aux routines binaires. Il y a parfois conflit entre ces deux commandes. Nous verrons plus loin comment y remédier. Maintenant, faites en direct:

```
PRINT HIMEM
```

Suivant les configurations de CPC et des extensions utilisées, vous allez obtenir une valeur que nous appellerons «A». Si vous faites :

Tout jeu Basic, pas trop indigent, fait appel aux commandes Symbol et Symbol after. Etes-vous certain d'en tirer le meilleur parti?

```
SYMBOL AFTER 240 : PRINT  
HIMEM
```

La valeur obtenue sera identique à la précédente. Tout simplement parce que l'espace nécessaire pour ces 16 caractères est déjà réservé dès l'initialisation de l'ordinateur. Espace qui peut être récupéré de la façon suivante.

```
SYMBOL AFTER 255 : PRINT  
HIMEM
```

La valeur obtenue «B» est supérieure à «A» de $8 \times 16 = 128$ octets. Ce n'est pas grand chose, mais cela peut toujours servir. Maintenant, faisons :

```
SYMBOL AFTER 40 : PRINT  
HIMEM
```

La valeur obtenue «C»; est inférieure à «A» de $8 \times 200 = 1600$. 1600 octets nécessaires pour modifier 200 caractères. *Symbol After* sert de barrière mais aussi de copie de la ROM vers la RAM du jeu de caractères. Pour vous en convaincre; tapez les lignes suivantes:

```
10 SYMBOL AFTER 32 :  
X=HIMEM+1 : MODE 2  
20 CLS : FOR H=1 TO 8 :  
PRINT BIN$(PEEK(X), 8)  
30 X=X+1 : NEXT : CALL
```

```
&BB18 : GOTO 20
```

Apparaissent alors un à un tous les caractères dans leur représentation binaire. A partir de là, toutes les manipulations sont possibles: lettres sens dessus dessous, tronquées, inversées, en italiques, etc. A vous de faire preuve d'imagination. Si vous en manquez, essayez les trois petits programmes de notre exemple.

Improper Argument

Ceux d'entre vous qui tapent régulièrement des listings n'ont sûrement pas échappé à l'*Improper Argument* qui apparaît parfois sur la ligne contenant l'instruction *Symbol After*. Cette ligne, qui en elle-même ne comporte aucune erreur, vient d'entrer en conflit avec une commande *Memory* rencontrée dans le listing. Pour comprendre le phénomène, tapez les lignes suivantes.

```
10 MEMORY &8000  
20 SYMBOL AFTER 200  
30 PRINT "BONJOUR" : END
```

Quoique l'on fasse, le «Bonjour» de la ligne 30 ne s'affichera pas, le programme renvoyant un tétu *Improper Argument* en ligne 20. *Memory* et *Symbol After* sont deux bar-

rières, mais le système n'admet pas que le *Memory* soit défini avant le *Symbol after*. Il aurait fallu écrire :

```
10 SYMBOL AFTER 200  
20 MEMORY &8000  
30 PRINT "BONJOUR" : END
```

Dans ce cas, RUN affichera correctement le message, mais faites un second RUN. Coucou, revoilou l'*Improper Argument!* En fait, lors du premier lancement, la barrière *Memory* placée en &8000 reste active jusqu'au *Reset* complet de l'ordinateur. De ce fait, le deuxième RUN recrée la situation du premier programme. Alors que faire? Il existe deux solutions.

- Premier cas: vous travaillez à la mise au point d'un listing dont vous n'êtes pas l'auteur. Le *Memory* reste actif tant qu'un *Reset* n'a pas lieu. Il en est de même pour *Symbol After* et les caractères redéfinis. Donc, lors des nombreux RUN d'essais, inutile de les faire relire par le programme. Rajouter, juste avant ce passage, une petite ligne avec un GOTO pour enjamber l'obstacle. Dans notre exemple cela donnera: 5 GOTO 20. Ainsi, la mise au point pourra s'effectuer sans problème. Celle-ci terminée et avant d'effectuer la sauvegarde définitive, supprimez ce GOTO superflu.

- La deuxième solution est encore plus simple. Il existe une routine système en &BB4E qui réinitialise tout ce qui a trait à la gestion des textes à

l'écran. *Symbol* et *Symbol after* rentrent dans cette catégorie. Ladite routine agit donc comme un *Reset sélectif*, exactement ce que nous recherchons. Ainsi la ligne : 5 CALL &BB4E appliquée à nos deux exemples, fera dispa-

raître tous nos problèmes d'*Improper Argument*.

Petite restriction: CALL &BB4E positionne le curseur en LOCATE 1,1. En début de listing, ceci ne devrait pas porter à conséquences.

de dégradés. Essayez donc des combinaisons du genre (n° de couleur) 26, 24, 15, 6 ou encore 26, 16, 1, 2 sur fond noir. Résultats garantis.

Reste maintenant à définir l'adresse de départ du fichier binaire qui va se créer. En fait, cela n'a que peut d'importance car il peut être relugé n'importe où. Il suffira de forcer l'adresse lors du chargement par un : LOAD "FICHIER", adresse désirée.

«>» idem.

«<<» idem.

«<<» remplacé par le point.

«» remplacé par la virgule.

«>>» remplacé par l'apostrophe.

«?» idem.

«à» remplacé par le point d'exclamation.

RECREATION : ALPHA-COLOR

Sur le point de créer un jeu, le mode 1 avec ses 4 couleurs vous semble insuffisant. Il sera donc en mode 0. Or, les 20 caractères possibles par ligne vous posent problème. Comment faire?

Désormais, la solution existe et s'appelle Alpha-color. Ce petit utilitaire permet en effet de créer à volonté un alphabet multicolore autorisant 40 caractères sur une ligne en MODE 0. J'avoue, que l'idée est due à l'équipe *Black System* qui, dans chacun de ses jeux, utilise un tel procédé. J'ai donc conçu un programme basic générant un fichier binaire relogable et utilisable par tout-un-chacun, même sans connaissance de l'assembleur.

Caractères redéfinis

Tous les chiffres, quelques signes de ponctuation et l'alphabet sont redéfinis deux par deux dans un caractère. Rien ne vous oblige à être d'accord avec mon style de redéfinition. Lorsque vous maîtriserez correctement le logiciel, modifiez à votre guise le dessin des lettres. Car il faut bien l'avouer, certaines comme le N, le M ou le W peuvent prêter à confusion.

Variables de base

Ligne 460 : Memory &3FAF. Une petite routine de capture sera placée entre l'adresse &3FB0 et &3FFF. La zone comprise entre &4000 et &A1E0 est réservée au stockage du fichier où bon vous semble.

Ligne 470 : initialisation d'un DIM pour conserver le numéro des encres.

Lignes 480 - 490 : mode transparent et mode XOR pour sous-programme de recherche des couleurs.

La recherche des couleurs

Il est bien évident que les couleurs de votre logiciel seront préalablement choisies. Ce sous-programme permet de reproduire exactement la palette. A noter que je me suis contenté de reprendre un sous-programme déjà utilisé dans «Aux voleurs». Faites de même.

Le questionnaire

La sélection de votre palette de couleurs est la première question à laquelle répondre. Apparaît alors la liste des styles et les encres qui leur sont affectées. A vous d'en choisir 4 pour rendre votre alphabet multicolore. L'usage m'a prouvé que les mélanges qui en «jetten» sont toujours à base

Création du fichier

Ligne 750 : affectation des couleurs.

Lignes 760 - 770 : affichage des lettres redéfinies.

Lignes 780 - 840 : coloration des lettres suivant les styles choisis.

Lignes 850 - 910 : implantation de la routine d'affichage et capture une à une de toutes les lettres. A ce stade du programme, tout est déjà terminé.

Le fichier

Le fichier ainsi créé mérite quelques commentaires. Il commence à l'adresse que vous avez choisie par une petite routine d'affichage de 32 octets. Celle-ci est calibrée pour des sprites de 2 octets sur 8 lignes; la taille de nos lettres. Après la routine figure la représentation des lettres, le fichier total faisant 720 octets.

J'attire votre attention sur le jeu de caractères disponibles. Vous disposez de tous les chiffres et de tout l'alphabet en majuscule. Ils suivent tous la progression logique telle qu'elle est représentée dans la liste des codes ASCII. Mais entre chiffres et lettres, 7 caractères pour un embryon de ponctuation qui ne correspond pas tout à fait à la réalité. Ainsi, si le «>» est bien à sa place, le signe «<<» a été remplacé par le point. Voici les équivalences.

Utilisation du fichier

Elle est en fait donnée en exemple dans le logiciel. Etudiez avec attention les lignes 920 à 1000.

Ligne 920 : fonction permettant de transformer l'équivalent d'un LOCATE X, Y en MODE 1, en une adresse écran exploitable par la routine d'affichage.

Ligne 930 : data d'une phrase à afficher. Les deux premiers chiffres sont les données X, Y du LOCATE.

Ligne 980 : la phrase est dans la variable A\$. En majuscule. Chaque lettre est prise une à une et, dans SP, on calcule son numéro de code ASCII. 48 lui est retiré car le 0 est notre premier caractère contenu dans le fichier.

Ligne 990 : si le caractère n'existe pas dans le fichier, il est remplacé par un espace. Ce dernier, situé à la position 43, est le dernier caractère contenu dans le fichier.

Ligne 1000 : affichage une à une des lettres. La variable ADD est l'adresse de départ du fichier fournie dans le questionnaire. La fonction FN po(ZX, ZY) définie en 920 entre en action. DEP1 = ADD+32. On pointe sur le début de la première lettre. Il suffit de lui additionner la valeur SP multipliée par 16 pour pointer sur la lettre désirée. La première lettre s'affiche, un bouclage affiche toute la phrase. Avec l'espoir que ce petit utilitaire permette de donner un air «pro» à vos prochaines réalisations...

Claude Le Moulec

PROGRAMMATION
INITIATION

10	REM	::: a l'envers	::: [1059]	H	230	SYMBOL	205,0,0,0,0,34,0,34,4	[1259]
20	MODE	1:SYMBOL AFTER 32:x=HIMEM	[2804]	I	240	SYMBOL	206,0,0,0,0,0,34,34,4	[1823]
30	FOR	h=32 TO 255:FOR g=1 TO 8	[1349]	J	250	SYMBOL	207,34,37,69,1,2,0,2,0	[1384]
40	va(g)	=PEEK(x+g):NEXT g:FOR g=8	[3814]	K	260	SYMBOL	208,39,39,37,37,39,5,3	[2196]
		TO 1 STEP -1		L	270	SYMBOL	209,119,119,84,84,100,	[2202]
50	z=z+1:POKE x+z,va(g):NEXT PRIN	[2496]				84,119,0		
T	CHR\$(h);			H	280	SYMBOL	210,103,119,84,84,86,8	[1798]
60	x=x+8:z=0:NEXT h:CALL &BB18	[2179]				4,119		
70	REM	::: demi caractere	::: [513]	N	290	SYMBOL	211,119,119,69,68,100,	[1809]
80	MODE 0:SYMBOL AFTER 48:x=HIMEM	[2774]				69,71,0		
90	FOR h=48 TO 122:FOR g=1 TO 8	[2330]		O	300	SYMBOL	212,87,82,82,82,114,82	[2210]
100	b\$=BIN\$(PEEK(X+G),8)	[969]		P	310	SYMBOL	213,117,37,37,38,38,37	[1852]
110	a=VAL("&X"+MID\$(B\$,2,1)+MID\$([3189]				.101,0		
	B\$,5,1)+MID\$(B\$,7,1)+"00000")			Q	320	SYMBOL	214,69,71,71,69,69,69,	[2153]
						117,0		
115	POKE x+g,a	[304]		R	330	SYMBOL	215,87,85,117,117,117,	[2471]
120	NEXT:PRINT CHR\$(h)::x=x+8:NEX	[2817]				85,87,0		
T	h:CALL &BB18			S	340	SYMBOL	216,119,119,85,85,117,	[1992]
130	REM	::: caractere nain	::: [583]			69,71,2		
140	MODE 0:SYMBOL AFTER 48:x=HIME	[2774]		T	350	SYMBOL	217,119,119,84,84,103,	[2466]
M						81,87,0		
150	FOR h=48 TO 122:FOR g=1 TO 8	[2330]		U	360	SYMBOL	218,117,117,37,37,37,3	[2248]
160	va(g)=PEEK(x+g):NEXT:FOR g=1	[2649]				7,39,0		
	TO 8			V	370	SYMBOL	219,85,85,85,85,87,87,	[2314]
170	POKE (x+g),0:NEXT	[1186]		W	380	SYMBOL	220,85,85,85,34,34,82,	[2042]
180	POKE x+3,va(1):POKE x+4,va(3)	[1552]				82,0		
	:POKE x+5,va(4)			X	390	SYMBOL	221,112,16,16,32,32,64	[1935]
185	POKE x+6,va(5):POKE x+7,va(7)	[1825]				.112,0		
190	PRINT CHR\$(h)::x=x+8:NEXT h	[1613]		Y	400	SYMBOL	255,0,60,60,60,60,60,6	[4779]
				Z	0:SYMBOL	254,126,66,66,66,66,66		
						.66,126		
					410	REM	::: [1736]	
					420	REM	::: [419]	
					430	REM	: VARIABLES DE BASE	[2081]
					440	REM	::: [419]	
					450	REM	::: [1736]	
					460	MEMORY &3FAF		[744]
					470	DIM color(17):FOR H=0 TO 15:C		[3816]
						OLOR(H)=H:NEXT:INK 0,0:INK 1,24:M		
						ODE 2		
					480	xr\$=CHR\$(23)+CHR\$(1):nr\$=CHR\$	[3091]	
						(23)+CHR\$(0)		
					490	tr\$=CHR\$(22)+CHR\$(1):no\$=CHR\$	[1638]	
						(22)+CHR\$(0)		
					500	REM	::: [1736]	
					510	REM	::: [419]	
					520	REM	: QUESTIONNAIRE	[1361]
					530	REM	::: [419]	
					540	REM	::: [1736]	
					550	LOCATE 1,1:PRINT"PREMIERE CHO		[5912]
						SE : EDITER LA PALETTE DE COULEUR		
						UTILISEE POUR VOTRE LOGICIEL."		
					560	LOCATE 1,4:PRINT CHR\$(242)+"	[4133]	
						et "+CHR\$(243)+" POUR LE NUMERO D		
						E L'ENCRE."		
					570	LOCATE 1,5:PRINT CHR\$(240)+"	[5223]	
						et "+CHR\$(241)+" POUR LE NUMERO D		
						E LA COULEUR.		
					580	LOCATE 1,6:PRINT "PUIS <ENTER	[3751]	
						> QUAND C'EST FINI..."		
					590	LOCATE 28,8:PRINT "APPUYEZ SU	[3283]	
						R UNE TOUCHE SVP":CALL &BB18		
					600	GOSUB 1220		[871]

PROGRAMMATION
INITIATION

```

610 PLOT 320,350:DRAW 320,16:FOR [3173]      &3FB0, ECR, DEP: ECR=ECR+2: DEP=DEP+1
H=0 TO AFF: LOCATE 46, H+6                6: NEXT
620 PRINT "STYLO NR ": PRINT USIN [5040]      910 ecr=&C050: FOR h=1 TO 22: CALL [4181]
G "##": H: PRINT " " = ": PRINT USIN      &3FB0, ECR, DEP: ECR=ECR+2: DEP=DEP+1
G "##": COLOR(H): NEXT                    6: NEXT
630 LOCATE 1,1: PRINT "IL Y A 4 STY [9260]      920 DEF FN PO(X,Y)=&C000+(Y-1)*80 [585]
LOS POSSIBLES POUR CHAQUE LETTRE.      +(X-1)*2
LE CHOIX DOIT SE FAIRE EN ALLANT      930 DATA 2,10, "C>EST TERMINE IL N [3357]
DUHAUT VERS LE BAS DE LA LETTRE.      E RESTE PLUS QU>A "
640 FOR H=6 TO 9 [804]                    940 DATA 5,12, "SAUVEGARDER LE FIC [2333]
650 LOCATE 1,H: PRINT SPACES(40): L      HIER <<<<<"
OCATE 3,H: PRINT "TRAIT NR "+CHR$(      950 DATA 7,14, "ENTER POUR LA SAU [2062]
24): H-5: PRINT CHR$(24)+ " AVEC LE      VEGARDE"
STYLO NR : " [9870]
660 X1=35: Y1=H: LG=2: GOSUB 1480: EC [4308]      960 DATA 7,16, "OU SPACE POUR RECO [1569]
(H-5)=VAL(LIG$): IF EC(H-5)>AFF TH      NMENCER"
EN PRINT CHR$(7): GOTO 650
670 NEXT H [372]
680 LOCATE 1,15: PRINT "ADRESSE DE [5892]      970 CLS: RESTORE 920: FOR H=1 TO 4: [1492]
STOCKAGE DU FICHIER": LOCATE 1,16:      READ ZX, ZY, AS
PRINT "MINIMUM &4000 : MAXIMUM &A1      980 AS=UPPER$(AS): FOR T=1 TO LEN( [2706]
E0 : &": X1=34: Y1=16: LG=4: GOSUB 14      AS): SP=(ASC(MID$(AS,T,1))-48
80 [372]
690 AS=LIG$: ADD=VAL("&"+RIGHT$(AS, [3691]      990 IF sp<0 OR sp>42 THEN sp=43 [1232]
2))+VAL("&"+LEFT$(AS,2))*256
700 REM : : : : : : : : : : : : : : [1736]      1000 CALL ADD.FN PO(ZX,ZY)+(T*2), [2380]
710 REM : : : : : : : : : : : : : : [419]      DEP1+(SP*16): NEXT T, H
720 REM : CREATION FICHIER : [2209]      1010 AS=INKEY$: IF AS="" THEN 1010 [1470]
730 REM : : : : : : : : : : : : : : [419]      1020 IF AS=CHR$(13) THEN 1090 [730]
740 REM : : : : : : : : : : : : : : [1736]      1030 IF AS="" THEN RUN 470 ELSE [1254]
750 MODE MO: FOR H=0 TO AFF: INK H,      1010
COLOR(H): NEXT
760 LOCATE 1,1: PEN 1: FOR H=200 TO [3378]      1040 REM : : : : : : : : : : : : : [1736]
210: PRINT CHR$(H): NEXT
770 LOCATE 1,2: PEN 1: FOR H=211 TO [3234]      1050 REM : : : : : : : : : : : : : [419]
221: PRINT CHR$(H): NEXT
780 IF MO=0 THEN LG=400 ELSE LG=2 [1290]      1060 REM : SAUVEGARDE FICHIER : [1448]
00 [419]
790 FOR H=0 TO LG STEP 2: FOR G=1 [2194]      1070 REM : : : : : : : : : : : : : [419]
TO 4
800 IF TEST(H,401-(G*4))=1 THEN P [1967]      1080 REM : : : : : : : : : : : : : [1736]
LOT H,401-(G*4), EC(G)
810 IF TEST(H,403-(G*4))=1 THEN P [4322]      1090 MODE 2: INK 1,24: PEN 1:
LOT H,403-(G*4), EC(G)
820 IF TEST(H,385-(G*4))=1 THEN P [2998]      1100 LOCATE 1,1: PRINT "DEBUT DE F
LOT H,385-(G*4), EC(G)
830 IF TEST(H,387-(G*4))=1 THEN P [3218]      ICHIER EN : &": PRINT HEX$(ADD)
LOT H,387-(G*4), EC(G)
840 NEXT G, H [389]
850 DATA dd, 5e, 00, dd, 56, 01, dd, 6e, [3690]
02, dd, 66, 03, 06, 08, C5, E5, 06, 02, 7E,
12, 13, 23, 10, FA, E1, CD, 26, BC, C1, 10,
EF, C9
860 DATA dd, 5e, 00, dd, 56, 01, dd, 6e, [3583]
02, dd, 66, 03, 06, 08, C5, E5, 06, 02, 1A,
77, 23, 13, 10, FA, E1, CD, 26, BC, C1, 10,
EF, C9
870 RESTORE 850: FOR H=&3FB0 TO &3 [2101]
FB0+31: READ AS: POKE H, VAL("&"+AS)
: NEXT
880 RESTORE 860: FOR H=ADD TO ADD+ [4068]
31: READ AS: POKE H, VAL("&"+AS): NEXT
T
890 DEP=ADD+32: dep1=dep [1544]
900 ecr=&C000: FOR h=1 TO 22: CALL [3113]

```

PROGRAMMATION
INITIATION

EN pex=pex-1:GOTO 1330		1410 PEN 0:FOR h=1 TO eff:LOCATE	[2908]
1300 IF a\$=CHR\$(240) AND color(pe	[2008]	h,1:PRINT CHR\$(143):NEXT	
x-5)<26 THEN 1380		1420 INK 0,0:INK 1,24:MODE 2:PEN	[904]
1310 IF a\$=CHR\$(241) AND color(pe	[879]	1:RETURN	
x-5)>0 THEN 1400		1430 REM :::::::::::::::::::::	[1736]
1320 PRINT CHR\$(7):GOTO 1260	[1604]	1440 REM :	[419]
1330 PEN 0:LOCATE pez,1:PRINT CHR	[3500]	1450 REM : COMMANDE INPUT	[497]
\$(254):PEN 1:LOCATE pex,1:PRINT C		1460 REM :	[419]
HR\$(254)		1470 REM :::::::::::::::::::::	[1736]
1340 PEN 1:pez=pex:LOCATE 1,1:PRI	[3955]	1480 lig\$="":LOCATE x1,y1:zr=134	[1664]
NT no\$:col=color(pex-5)		1490 a\$=INKEY\$:zr=zr XOR 15:PRINT	[3998]
1350 IF pex-5<10 THEN LOCATE 1,1:	[5182]	CHR\$(zr);CHR\$(8):: IF a\$="" THEN	
PRINT "0":PRINT USING "#":pex-5		1490	
ELSE LOCATE 1,1:PRINT USING "##";		1500 a\$=UPPER\$(a\$):IF ASC(a\$)=13	[9882]
pex-5		THEN 1520 ELSE IF (a\$>"Z" AND ASC	
1360 IF col<10 THEN LOCATE 3,1:PR	[4970]	(a\$)<>127) THEN 1490 ELSE IF ASC(
INT "0":PRINT USING "#":col ELSE		a\$)=127 AND LEN(lig\$)>0 THEN lig	
LOCATE 3,1:PRINT USING "##":col		\$=LEFT\$(lig\$,LEN(lig\$)-1):PRINT C	
1370 LOCATE 1,1:PRINT tr\$:GOTO 12	[1407]	HR\$(8);" ":CHR\$(8);CHR\$(8)::GOTO	
60		1490	
1380 col=color(pex-5):col=col+1:c	[3084]	1510 IF LEN(lig\$)=lg THEN PRINT C	[5963]
olor(pex-5)=col:INK pex-5,col		HR\$(7)::GOTO 1490 ELSE lig\$=lig\$+	
1390 PEN pex-5:LOCATE pex,1:PRINT	[2383]	a\$:PRINT RIGHT\$(lig\$,1)::GOTO 149	
CHR\$(255):GOTO 1340		0	
1400 col=color(pex-5):col=col-1:c	[3000]	1520 LOCATE x1,y1:PRINT CHR\$(24)+	[3785]
olor(pex-5)=col:INK pex-5,col:GOT		" "+lig\$+" "+CHR\$(24)	
O 1390		1530 RETURN	[555]

COMMENT TAPER UN LISTING SANS ERREURS?
 (C'EST UNE BONNE MANIÈRE DE GÂNER DU BÉCOT!)



VERIFICATEURS V.2

Le savez-vous? La présence dans nos listings de numéros entre crochets, assuré la détection d'éventuelles erreurs lors de la saisie d'un programme Basic. Tapez minutieusement et sauvegardez le listing vérificateur relatif à votre machine

(464, 664 ou 6128), conservez-le précieusement et lancez-le avant toute frappe ou vérification de listing; les indications nécessaires sont incluses. Chaque somme affichée devra rigoureusement correspondre à celle figurant dans nos colonnes.

Nota : les deux traits verticaux présents en ligne 80 et 90, correspondent à une barre verticale sur un clavier QWERTY, ou à un "û" sur clavier AZERTY. Ces caractères - CHR\$(124) - s'obtiennent par appui simultané sur SHIFT/trait vertical.

```

10 REM Verificateur 464
20 MEMORY &A4FF
30 FOR aZ=&A500 TO &A607
40 READ byte#
50 POKE aZ,VAL("&"&byte#)
60 NEXT
70 CLS:LOCATE 5,1:PRINT CHR$(24)
  " Verificateur V.2 464 installe
  "CHR$(24):PRINT:PRINT"Chaque val
  idation affiche désormais une so
  mme de controle." :PRINT
80 PRINT":CHECK,2 : liste le pro
  gramme avec les      somm
  es de controle." :PRINT:PRINT":CH
  ECK,8 : imprime le programme ave
  c les      sommes de contro
  le." :PRINT:PRINT      "CHR$(24) "
  Taper ou charger un programme. "
  CHR$(24)
90 CALL &A500:ON
100 NEW
110 DATA 21,09,a5,01,0d,a5,c3,d1
120 DATA bc,00,00,00,00,18,a5,c3
130 DATA 2a,a5,c3,2f,a5,c3,43,a5
140 DATA 4f,ce,4f,46,ce,43,48,a5
150 DATA 43,cb,00,00,cf,98,aa,c3
160 DATA a8,a5,21,27,a5,18,03,21
170 DATA 24,a5,28,06,cd,00,b9,c3
180 DATA ed,b0,c9,4f,cd,00,b9,0d
190 DATA 28,08,0d,20,ea,dd,7e,02
210 DATA 18,04,7b,11,01,0d,cd,a2
220 DATA c1,cd,69,e8,e5,4e,23,46
230 DATA 23,5e,23,56,e1,78,b1,c8
240 DATA cd,3c,4a,e5,09,e3,cd,63
250 DATA e1,21,a4,ac,cd,7a,a5,e1
260 DATA 18,e2,e5,cd,ba,a5,e3,cd
270 DATA 98,a5,cd,98,f3,e3,cd,f6
280 DATA a5,cd,4e,c3,e1,7e,a7,c8
290 DATA cd,98,a5,cd,4e,c3,18,f5
300 DATA 3a,24,ac,d6,08,47,7e,a7
310 DATA c8,cd,4e,e1,23,10,f7,c9
320 DATA cd,24,a5,f5,c5,d5,e5,cd
330 DATA ba,a5,cd,f6,a5,e1,d1,c1
340 DATA f1,c9,eb,1b,af,47,67,6f
350 DATA 2f,32,23,a5,13,1a,d6,30
360 DATA 38,04,fe,0a,38,fe,1a,13
370 DATA a7,c8,4f,3a,23,a5,a1,fe
380 DATA 20,28,f3,79,fe,22,20,07
390 DATA 3a,23,a5,2f,32,23,a5,3a
400 DATA 23,a5,a7,79,c4,ba,ff,4f
410 DATA ad,07,6f,09,18,d8,3e,20
420 DATA cd,5c,c3,3e,5b,cd,a0,c3
430 DATA cd,79,ee,3e,5d,c3,5c,c3
  
```

```

10 REM Verificateur 664
20 MEMORY &A4FF
30 FOR aZ=&A500 TO &A607
40 READ byte#
50 POKE aZ,VAL("&"&byte#)
60 NEXT
70 CLS:LOCATE 5,1:PRINT CHR$(24)
  " Verificateur V.2 664 installe
  "CHR$(24):PRINT:PRINT"Chaque val
  idation affiche désormais une so
  mme de controle." :PRINT
80 PRINT":CHECK,2 : liste le pro
  gramme avec les      somm
  es de controle." :PRINT:PRINT":CH
  ECK,8 : imprime le programme ave
  c les      sommes de contro
  le." :PRINT:PRINT      "CHR$(24) "
  Taper ou charger un programme. "
  CHR$(24)
90 CALL &A500:ON
100 NEW
110 DATA 21,09,a5,01,0d,a5,c3,d1
120 DATA bc,00,00,00,00,18,a5,c3
130 DATA 2a,a5,c3,2f,a5,c3,43,a5
140 DATA 4f,ce,4f,46,ce,43,48,a5
150 DATA 43,cb,00,00,cf,02,ac,c3
160 DATA a8,a5,21,27,a5,18,03,21
170 DATA 24,a5,28,06,cd,00,b9,c3
180 DATA 4c,cb,11,5b,bd,01,03,00
190 DATA ed,b0,c9,4f,cd,00,b9,0d
200 DATA 28,08,0d,20,ea,dd,7e,02
210 DATA 18,04,7b,11,01,0d,cd,a2
220 DATA c1,cd,69,e8,e5,4e,23,46
230 DATA 23,5e,23,56,e1,78,b1,c8
240 DATA cd,75,c4,e5,09,e3,cd,59
250 DATA e2,21,8a,ac,cd,7a,a5,e1
260 DATA 18,e2,e5,cd,ba,a5,e3,cd
270 DATA 98,a5,cd,98,f3,e3,cd,f6
280 DATA a5,cd,9b,c3,e1,7e,a7,c8
290 DATA cd,98,a5,cd,9b,c3,18,f5
300 DATA 3a,09,ac,d6,08,47,7e,a7
310 DATA c8,cd,22,e2,23,10,f7,c9
320 DATA cd,24,a5,f5,c5,d5,e5,cd
330 DATA ba,a5,cd,f6,a5,e1,d1,c1
340 DATA f1,c9,eb,1b,af,47,67,6f
350 DATA 2f,32,23,a5,13,1a,d6,30
360 DATA 38,04,fe,0a,38,fe,1a,13
370 DATA a7,c8,4f,3a,23,a5,a1,fe
380 DATA 20,28,f3,79,fe,22,20,07
390 DATA 3a,23,a5,2f,32,23,a5,3a
400 DATA 23,a5,a7,79,c4,ba,ff,4f
410 DATA ad,07,6f,09,18,d8,3e,20
420 DATA cd,a3,c3,3e,5b,cd,a0,c3
430 DATA cd,49,ef,3e,58,c3,a3,c3
  
```

```

10 REM Verificateur 6128
20 MEMORY &A4FF
30 FOR aZ=&A500 TO &A607
40 READ byte#
50 POKE aZ,VAL("&"&byte#)
60 NEXT
70 CLS:LOCATE 4,1:PRINT CHR$(24)
  " Verificateur V.2 6128 installe
  "CHR$(24):PRINT:PRINT"Chaque va
  lidation affiche désormais une s
  omme de controle." :PRINT
80 PRINT":CHECK,2 : liste le pro
  gramme avec les      somm
  es de controle." :PRINT:PRINT":CH
  ECK,8 : imprime le programme ave
  c les      sommes de contro
  le." :PRINT:PRINT      "CHR$(24) "
  T
  aper ou charger un programme. "C
  HR$(24)
90 CALL &A500:ON
100 NEW
110 DATA 21,09,a5,01,0d,a5,c3,d1
120 DATA bc,00,00,00,00,18,a5,c3
130 DATA 2a,a5,c3,2f,a5,c3,43,a5
140 DATA 4f,ce,4f,46,ce,43,48,a5
150 DATA 43,cb,00,00,cf,02,ac,c3
160 DATA a8,a5,21,27,a5,18,03,21
170 DATA 24,a5,28,06,cd,00,b9,c3
180 DATA 49,cb,11,5b,bd,01,03,00
190 DATA ed,b0,c9,4f,cd,00,b9,0d
200 DATA 28,08,0d,20,ea,dd,7e,02
210 DATA 18,04,7b,11,01,0d,cd,a6
220 DATA c1,cd,6a,e8,e5,4e,23,46
230 DATA 23,5e,23,56,e1,78,b1,c8
240 DATA cd,72,c4,e5,09,e3,cd,54
250 DATA e2,21,8a,ac,cd,7a,a5,e1
260 DATA 18,e2,e5,cd,ba,a5,e3,cd
270 DATA 98,a5,cd,98,f3,e3,cd,f6
280 DATA a5,cd,9b,c3,e1,7e,a7,c8
290 DATA cd,98,a5,cd,98,c3,18,f5
300 DATA 3a,09,ac,d6,08,47,7e,a7
310 DATA c8,cd,1e,d2,23,10,f7,c9
320 DATA cd,24,a5,f5,c5,d5,e5,cd
330 DATA ba,a5,cd,f6,a5,e1,d1,c1
340 DATA f1,c9,eb,1b,af,47,67,6f
350 DATA 2f,32,23,a5,13,1a,d6,30
360 DATA 38,04,fe,0a,38,fe,1a,13
370 DATA a7,c8,4f,3a,23,a5,a1,fe
380 DATA 20,28,f3,79,fe,22,20,07
390 DATA 3a,23,a5,2f,32,23,a5,3a
400 DATA 23,a5,a7,79,c4,ba,ff,4f
410 DATA ad,07,6f,09,18,d8,3e,20
420 DATA cd,a0,c3,3e,5b,cd,a0,c3
430 DATA cd,44,ef,3e,5d,c3,a0,c3
  
```


Vive la chasse !

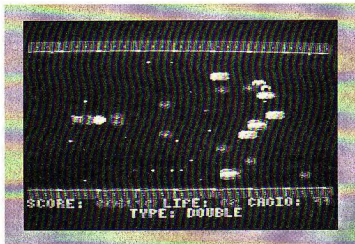
DEATH'S TICKET

Les munitions, en nombre limité, devront être utilisées à bon escient. Le valeureux pilote pourra néanmoins se réapprovisionner en cours de jeu auprès de quelques rares containers. De plus, il rencontrera aussi peu fréquemment des types d'afûts capables de renforcer l'efficacité de son tir (double, triple, etc.). Les commandes sont classiques (Joystick ou touches directionnelles + espace) et «ESC» permet l'abandon. Malgré la simplicité du scénario, ce «shoot'em up» est remarquablement animé.

Sauvegarde

Sauvez sous un nom de votre choix, le court listing Basic de chargement. Entrez ensuite par

Aux commandes du redoutable «Monster vice», Kevin Juck se couvrit de gloire en affrontant les hordes impérialistes de la planète Zégon.



Amsiaise V.2 en vous reportant à son mode d'emploi, les quatre listings de codes hexadécimaux.

Nom	Adr. déb.	Long
DEATH1	&4000	&401
DEATH2	&4800	&2800
DEATH3	&8000	&ED4
DEATH4	&94E0	&238

La longueur est ici précisée à l'attention de ceux qui envisagent raisonnablement de morceler leur travail en plusieurs fichiers qui devront ultérieurement être réunis en quatre fichiers définitifs.

Sébastien Royer

```

1 ***** [665]
2 ** DEATH'S TICKET ** [700]
3 ** (C) R.O.S.SOFT ** [2034]
4 * by * [227]
5 * Sebastien Royer * [1874]
6 ***** [665]
7 Special thanks to: [838]
8 Robert.Vincent.Steph [1509]
ane
9 * Nelly,Aude,Stephanie [1848]
  Severine
10 and MARIE [384]
11 OPENOUT"MARIE":MEMORY [3263]
  AFFF:CLOSEOUT
12 LOAD"DEATH1",&1000 [1394]
13 LOAD"DEATH2",&1800 [1356]
14 LOAD"DEATH3",&8000 [1092]
15 LOAD"DEATH4",&94E0 [1495]
16 CALL &000 [453]

```

```

4000:00 00 00 00 00 00 00:40
4008:00 00 00 00 00 00 00:48
4010:04 00 04 00 44 00 44 00:90
4018:10 00 00 00 10 00 00:88
4020:08 08 88 20 00 00 00:90
4028:00 00 00 00 00 00 00:68
4030:04 04 04 04 4C 88 44 00:98
4038:CC 20 10 00 10 00 00:94
4040:0C 88 4C 00 4C 00 CC 88:00

```

```

4048:44 88 44 20 30 20 00:08
4050:08 08 08 88 00 88 44 00:BC
4058:10 00 20 00 20 88 20:88
4060:0C 00 08 00 4C 88 88 88:68
4068:CC 20 00 20 10 20 00:44
4070:00 08 00 88 10 00 00:30
4078:00 00 00 00 00 00 00:88
4080:00 08 04 00 04 00 44 00:CC
4088:44 00 10 00 20 00 00:FC
4090:04 00 00 08 00 08 88:64
4098:00 88 00 20 10 00 00:88
40A0:00 00 00 00 00 00 88 44:38
40A8:04 88 44 88 20 10 00 00:88
40B0:00 00 00 00 04 10 00 00:20
40B8:4C 88 10 00 10 00 00:D4
40C0:00 00 00 00 00 00 00:00
40C8:00 00 08 00 88 10 00:C8
40D0:00 00 00 00 00 00 00:10
40D8:4C 20 00 00 00 00 00:A4
40E0:00 00 00 00 00 00 00:20
40E8:00 00 04 10 10 00 00:84
40F0:00 08 00 08 00 88 44 00:6C
40F8:44 00 88 20 20 00 20:94
4100:41 00 82 02 82 02 88:F4
4108:02 A8 A8 A8 54 00 00:D7
4110:41 00 C3 00 01 00 01 00:E7
4118:54 00 54 00 54 00 00:4D
4120:41 00 82 82 02 01 00:43
4128:02 00 A8 00 FC A8 00:3F
4130:41 00 82 02 00 02 01 00:53

```

```

4138:00 02 A8 A8 54 00 00 00:B9
4140:82 00 82 02 82 02 03:4C
4148:00 A8 00 A8 00 A8 00:69
4150:C3 82 82 00 02 00 01 00:EF
4158:00 02 A8 A8 54 00 00 00:D9
4160:41 00 82 02 82 00 03 00:0F
4168:02 A8 A8 A8 54 00 00:37
4170:C3 82 00 82 00 02 01 02:A3
4178:00 A8 00 A8 00 A8 00:99
4180:41 00 82 82 82 02 01 00:2D
4188:02 A8 02 A8 54 00 00 00:65
4190:41 00 82 82 82 02 01 02:4D
4198:00 A8 A8 A8 54 00 00 00:65
41A0:00 00 00 00 04 00 44 00:D1
41A8:00 00 04 00 10 00 00 00:45
41B0:00 00 00 00 04 00 44 00:E1
41B8:00 00 04 10 00 20 00:35
41C0:00 00 00 00 04 00 08 00:6D
41C8:88 00 44 00 00 20 00 00:1D
41D0:00 00 00 00 00 00 00:11
41D8:0C 88 00 98 20 00 00:ED
41E0:00 00 08 00 04 00 00 08:8D
41E8:00 88 44 00 20 00 00 00:A5
41F0:04 00 08 00 88 44 00 00:79
41F8:10 00 00 00 10 00 00 00:99
4200:04 00 4C 08 4C 88 88:EE
4208:20 20 20 20 10 20 00 00:1A
4210:04 00 08 08 88 88 88:8A
4218:88 20 88 20 20 20 00:9A
4220:0C 00 08 08 08 88 CC 00:92

```

4228:88 20 88 20 30 00 00 00:3A
4230:04 00 08 00 00 88 00:8E
4235:88 00 20 20 00 00 00:32
4240:0C 00 00 88 08 88 20:D4
4245:88 20 88 20 30 00 00:5A
4250:0C 08 00 00 08 00 CC:00:82
4255:88 00 20 00 30 00 00:32
4260:04 00 00 08 00 00 CC:00:9A
4265:88 00 20 00 20 00 00:32
4270:04 00 00 08 00 00 88:0E
4275:88 20 20 20 10 00 00:00:B2
4280:08 00 08 88 08 88 CC:88:3E
4285:88 20 88 20 20 20 00:00:2A
4290:0C 08 04 20 00 44 00 00:2A
4295:44 00 10 00 20 30 00:00:FE
42A0:04 00 00 08 00 08 00 88:86
42A5:00 88 20 20 10 00 00 00:2A
42B0:08 08 88 08 88 00 CC:00:2E
42B5:88 88 88 20 20 20 00:00:0A
42C0:08 08 00 88 00 88 00:82
42C5:88 00 20 00 30 00 00:32
42D0:08 08 4C 88 4C 88 88:8D
42D5:88 20 20 20 20 20 00:22
42E0:08 08 88 88 4C 88 CC:88:FE
42E5:CC 20 88 20 20 20 00:00:AE
42F0:04 00 00 08 08 88 88:8E
42F5:88 20 20 20 10 00 00:00:32
4300:0C 00 00 08 08 88 CC:00:73
4305:88 00 20 00 20 00 00:00:D3
4310:04 00 00 08 08 88 88:8D
4315:88 88 20 20 10 88 00:00:53
4320:0C 00 00 08 88 88 CC:00:93
4325:88 88 88 20 20 20 00:7B
4330:04 00 08 08 00 44 00:00:33
4335:08 88 20 20 10 00 00:00:BB
4340:0C 00 04 00 44 00 44:00:DB
4345:44 00 10 00 10 00 00:4F
4350:08 08 88 88 88 88 88:8B
4355:88 20 20 20 10 00 00:00:93
4360:08 08 88 88 88 88 88:8C
4365:20 20 10 00 10 00 00:00:8B
4370:08 08 08 08 08 88 88:85
4375:CC 20 98 20 20 20 00:00:6F
4380:08 08 08 20 08 88 44:00:47
4385:88 88 88 20 20 20 00:00:DB
4390:08 08 88 88 88 88 44:00:7B
4395:44 00 10 00 10 00 00:00:2F
43A0:0C 00 00 08 08 88 44:00:9F
43A5:88 00 20 30 30 20 00:00:83
43B0:04 00 04 00 04 00 44:00:03
43B5:88 00 10 00 10 00 00:00:7F
43C0:08 00 88 00 88 20 00:00:77
43C5:44 00 00 88 00 20 00:20:2F
43D0:04 00 08 08 08 88 88:8B
43D5:00 88 00 20 10 20 00:00:BB
43E0:00 00 00 08 04 00 CC:00:0B
43E5:CC 98 10 00 00 20 00:00:17
43F0:64 0C 44 04 00 64 8C:47
43F5:08 64 0C 44 0C 44 88 30:8C:CB
4400:84 00 00 00 00 00 00:00:08

4888:B1 9D A1 93 78 DA 05 91:09
4890:YD 55 51 D7 0B BB 16 26:06
4895:11 70 3F 4C 00 00 00:00:27
48A0:A3 58 E6 74 FC E9 F8 71:AD
48A5:44 62 5C 19 11 F3 D8 40:5F
48B0:EA 90 56 40 5B 68 FD 55:CE
48B5:A1 88 BC R1 44 48 7F:43
48C0:6D E8 8A A5 70 F2 B8 27:58
48C5:F9 D8 4F 06 55 A6 BD 4B:CE
48D0:45 AE A9 D1 A1 28 B7 99:DE
48D5:AC 62 02 2F 51 AD 79 23:5C
48E0:2F 66 B6 43 D1 EE 1D 35:6D
48E5:F1 08 CB 52 3E 9C 47 A8:93
48F0:1E 88 8B 25 D6 D7 08 44:2B
48F5:77 87 97 4E B5 81 C7 06:FE
4900:4A AD A6 24 F3 C7 EF 1B:39
4905:93 69 6C 2D C8 62 88 48:FF:2
4910:D7 0D 5F 05 6F 73 25 43:11
4915:7D EC D1 EB C6 E8 79 D1:AC
4920:49 AB 1C 29 6C 72 FC AD:84
4925:64 B4 D5 22 4A 18 AE 7D:50
4930:2C 4D 14 3A A1 66 76 F8:CC
4935:5A DF 0D 4F 38 CD BF 40:75
4940:40 4F 09 95 00 CC DA 12:6F
4945:CE 2E 6B 21 77 AE C3 C3:64
4950:12 A9 20 5E 3A 0E 12 3D:B1
4955:0D 8B A5 B5 94 86 3F BD:D8
4960:EA 69 FE DF 4F A0 55 0C:D9
4965:A3 C4 B2 DB 05 80 3A 84:2D
4970:B5 87 5F 85 EE 89 79 67:10
4975:38 8C C3 11 19 1B 24 45:E4
4980:82 3D AB BE 09 15 CE:5B
4985:47 14 36 4E 31 34 C8 A3:37
4990:A6 31 39 1C 2D 0B 51 5B:28
4995:0C 8B DC AB CB F9 58 33:7E
49A0:16 CF FC C3 E8 69 58 91:91
49A5:47 9D B7 F7 04 85 9C 21:93
49B0:9D C7 04 18 15 64 CA 58:97
49B5:51 03 D6 47 C2 D2 E9 69:98
49C0:5F C5 36 F1 41 87 E7 70:98
49C5:58 C1 53 39 19 11 AD A3:7E
49D0:8B 6E 2F 67 8B 6E 5C 5:7E
49D5:A1 EB F4 F0 4E 89 69 B7:A7
49E0:02 71 F2 6B 79 64 80 80:44
49E5:99 A1 D9 06 7E 18 61 F6:07
49F0:04 AD A5 1F BE 89 F7 0B:07
49F5:D6 AC F7 EA 44 43 30 18:52
4A00:D3 D4 3B 3A 46 B2 8A FA:50
4A05:A5 AD F5 67 2A 63 DF 91:91
4A10:0C EB D1 0A D1 5A FD 98:0C
4A15:D4 AF 7C 6F D0 99 E3 B7:77
4A20:FC E9 27 01 EA D1 7D 64:14
4A25:C7 4E EA 07 EA 41 C6 4F:59
4A30:1F 77 2E 58 62 46 F5:E1
4A35:6C E8 0B 3D 1E AF A5:ED
4A40:2B 25 0F B7 CE BD 94 8E:FA
4A45:0C 74 97 F6 A4 64 3C DF:F4
4A50:58 69 A4 F3 24 55 8D AF:7A
4A55:08 B2 2E 2B 15 08 28 AF:5B
4A60:27 2C 1D 9C 94 AD 6D 0B:55
4A65:45 E5 70 AC EB 6B 8E 02:CC
4A70:8D 77 05 B3 6F 28 2D 1E:CA
4A75:DA B1 B7 6B 34 F2 82 6B:7D
4A80:27 8E 9E AE 09 48 A0 1:3D
4A85:D2 83 31 36 9E FE 76 0E:D3
4A90:96 C0 CD 72 75 74 24:1A
4A95:BB B3 77 78 0C 9A ED 31:03
4AA0:66 82 CF 4A 80 19 91 93:9A
4AA5:FB FE 8D FB 14 B5 98 BA
4AB0:6F 7A 12 91 12 02 A1 F1:2C
4AB5:8F 79 BD D7 EB 47 1D:BC
4AC0:51 38 DB C7 93 2D 7A 11:63
4AC5:9A B9 F2 11 C6 7D EE C8:CD
4AD0:09 02 D7 B8 FD 3B A2 A4:6D
4AD5:A9 0F 77 EE 3C 07 13 9D:09
4AE0:15 AF FD 43 87 D3 02 67:CB
4AE5:FD CE 47 E3 71 3D 73 8A:64
4AF0:12 BE 08 52 44 30 53:F4
4AF5:8D C3 A8 35 B9 FA 2B F9:8F

4B00:27 1A 33 96 D6 B2 DE 70:83
4B05:74 BC F2 D6 8C CC FC:39
4B10:2B 60 66 64 7E 00 93:07
4B15:EC 91 92 B8 06 FE 0C AF:F1
4B20:39 41 47 E7 27 12 DA 42:53
4B25:45 08 5D CA EC CF DE 32:C9
4B30:37 31 D9 69 A7 B2 30 B3:9A
4B35:A9 3A 6C 11 5A 20 D8 C2:F2
4B40:AE 1E 64 65 E5 B1 B9 D8:43
4B45:00 CE 2D FD 6C 09 DE CF:87:60
4B50:48 E0 EB 96 1E 16 EC 88:8A
4B55:0A 37 1A 0A CC 93 35 64:92
4B60:CC 15 B3 36 29 75 7A D1:FB
4B65:69 14 3F 65 26 04 83 9F:38
4B70:35 68 64 B4 F4 6A 81 DA:23
4B75:7A 6B D1 FD B9 DC 30 2F:07
4B80:E1 FD 44 84 CC 05 AD 85:BF
4B85:17 15 10 9F 40 FE 1A 58:5A
4B90:1D 68 05 E8 76 6E 92 10:07
4B95:78 62 59 CA B3 59 F9 80:70
4BA0:C5 64 55 1C 61 F2 5F 5A:E9
4BA5:20 98 A1 4C 0E 44 5F 26:FD
4BB0:55 0B 71 92 97 B6 B6 AF:7D
4BB5:CB C9 1F 18 06 09 B6 8E:AB
4BC0:B7 A3 10 08 C4 03 00 26:AE
4BC5:C9 B2 2D 8B 28 D9 A5:11
4BD0:49 BA BF 48 59 65 00 17:14
4BD5:7E 1D F5 17 52 9F 13 5E:DF
4BE0:C1 E7 17 6B 5A 1C 6A 26:2B
4BE5:0C 36 6C 5E 51 EF 49 01:09:D3
4BF0:81 F9 CA 28 5F 10 77 8A:68
4BF5:7E 70 FC 55 05 6F 83 2D:F9
4C00:11 4C A5 EC F0 68 CD 30:22
4C05:14 49 27 49 50 84 BC 55:19
4C10:B4 D9 2A F2 D0 0A 20 2F:AE
4C15:08 83 D5 93 58 97 0C 83:07
4C20:9F 72 84 33 CE CC 6B 89:CA
4C25:A8 36 88 8A 02 41 37 0C:DA
4C30:E2 CA 5C D2 78 03 6A 3D:39
4C35:49 5F CC AD E8 09 D4 42:47
4C40:17 1A 45 64 9A E3 66 14:F4
4C45:2D C8 1E 9F 68 1C 58 4A:8F
4C50:5C BC D3 DA EB AE 7E 58:2E
4C55:30 CC BA EA 8E 69 32 D1 50:88
4C60:53 7C 76 64 64 82 92 3D:01
4C65:BD 81 7D 64 D4 19 25 DB 3A:B2
4C70:E0 02 7B DB 26 B9 1E:0A
4C75:31 EE C8 E2 F1 D6 8E 0F:CA
4C80:F5 9E 9F 11 9D 7A 9D 87:8E
4C85:05 86 BB 88 AE 2F AD 18:DE
4C90:51 20 13 46 D0 91 EF DE:4D
4C95:08 0E 43 80 80 9C EF 81:03
4CA0:72 9C 6E 44 C3 3C 6A 1A:61
4CA5:20 00 D2 93 4D 79 31 BA:62
4CB0:63 0E D3 C6 21 71 58 B2:4F
4CB5:E7 85 86 52 BF 38 96 A7:AF
4CC0:A3 79 1E BD E1 13 75 9B:D1
4CC5:98 D1 29 FA 0A 8F 7F 11:BE
4CC8:0C EB 94 8B 49 8E 3E A4:39
4CD0:00 4B 1C 24 87 95 3B 55:04
4CD5:FC 3E F7 0C FB F4 37 ED:41
4CE0:30 CF 31 36 81 05 58 F8:D8
4CF0:16 04 36 39 5A 18 5F 5D:29
4CF5:1E 2A 17 EA 5B 53 01 B4:84
4D00:C2 2B 90 5D 9A 27 77 27:12
4D05:8E 52 F1 67 65 54 D4 C9:90
4D10:E6 15 2F 62 66 62 EA 19:19
4D15:A0 65 58 92 49 EA B5 E9:1F
4D20:B3 80 8E 0C 4D 3C 05 05:DF
4D25:C6 C9 36 12 1D 04 9A 3B:8A
4D30:91 15 B2 6E 42 C6 60 6D:1C
4D35:F8 D8 6E 43 6E CA 81 66:10
4D40:CB 8D EA CD 13 79 62 BA:14
4D45:B8 CA 74 55 AD 96 F4 F5:CA
4D50:4C C8 68 C3 EF 82 16 98:EE
4D55:20 74 2B EF 8D 0C 2C 52:0E
4D60:0F 96 F6 98 BF 11 14 F3:25
4D65:22 1C B5 DF EB 8B A8 A1:34
4D70:F2 B4 89 52 BB C3 25 72:C6

4800:EC 8C F3 3B 20 CD C3 35:7A
4805:73 AC 1D 15 9B FA 24:EB
4810:EF 9F C2 3 7E 52 23:DT
4815:E1 DF CD 3D BC 63 A4 42:9D
4820:D0 2E FC 63 73 75 42:9D
4825:10 46 EC 88 D7 11 EB 9E E6:29
4830:24 46 D4 03 1D B4 CC 5F:05
4835:F6 0B 6C B2 16 CA C6:02
4840:0F 73 55 49 2F CC 9E 9C:85
4845:93 D8 67 14 7E 43 C2 9E:9E
4850:94 1E E2 BE 15 51 B0 AE:8D
4855:B1 76 89 8A 84 F6 6E C6:36
4860:DF 48 AE 94 A7 E5 CA 94:FE
4865:30 2B VD F6 73 57 2D 8D:E0
4870:84 52 B5 20 C9 65 3B BC:8A
4875:17 FB D1 33 2F AB D2 ES:DF
4880:4A C2 64 51 02 23 D5 7C:95

4885:B1 9D A1 93 78 DA 05 91:09
4890:YD 55 51 D7 0B BB 16 26:06
4895:11 70 3F 4C 00 00 00:00:27
48A0:A3 58 E6 74 FC E9 F8 71:AD
48A5:44 62 5C 19 11 F3 D8 40:5F
48B0:EA 90 56 40 5B 68 FD 55:CE
48B5:A1 88 BC R1 44 48 7F:43
48C0:6D E8 8A A5 70 F2 B8 27:58
48C5:F9 D8 4F 06 55 A6 BD 4B:CE
48D0:45 AE A9 D1 A1 28 B7 99:DE
48D5:AC 62 02 2F 51 AD 79 23:5C
48E0:2F 66 B6 43 D1 EE 1D 35:6D
48E5:F1 08 CB 52 3E 9C 47 A8:93
48F0:1E 88 8B 25 D6 D7 08 44:2B
48F5:77 87 97 4E B5 81 C7 06:FE
4900:4A AD A6 24 F3 C7 EF 1B:39
4905:93 69 6C 2D C8 62 88 48:FF:2
4910:D7 0D 5F 05 6F 73 25 43:11
4915:7D EC D1 EB C6 E8 79 D1:AC
4920:49 AB 1C 29 6C 72 FC AD:84
4925:64 B4 D5 22 4A 18 AE 7D:50
4930:2C 4D 14 3A A1 66 76 F8:CC
4935:5A DF 0D 4F 38 CD BF 40:75
4940:40 4F 09 95 00 CC DA 12:6F
4945:CE 2E 6B 21 77 AE C3 C3:64
4950:12 A9 20 5E 3A 0E 12 3D:B1
4955:0D 8B A5 B5 94 86 3F BD:D8
4960:EA 69 FE DF 4F A0 55 0C:D9
4965:A3 C4 B2 DB 05 80 3A 84:2D
4970:B5 87 5F 85 EE 89 79 67:10
4975:38 8C C3 11 19 1B 24 45:E4
4980:82 3D AB BE 09 15 CE:5B
4985:47 14 36 4E 31 34 C8 A3:37
4990:A6 31 39 1C 2D 0B 51 5B:28
4995:0C 8B DC AB CB F9 58 33:7E
49A0:16 CF FC C3 E8 69 58 91:91
49A5:47 9D B7 F7 04 85 9C 21:93
49B0:9D C7 04 18 15 64 CA 58:97
49B5:51 03 D6 47 C2 D2 E9 69:98
49C0:5F C5 36 F1 41 87 E7 70:98
49C5:58 C1 53 39 19 11 AD A3:7E
49D0:8B 6E 2F 67 8B 6E 5C 5:7E
49D5:A1 EB F4 F0 4E 89 69 B7:A7
49E0:02 71 F2 6B 79 64 80 80:44
49E5:99 A1 D9 06 7E 18 61 F6:07
49F0:04 AD A5 1F BE 89 F7 0B:07
49F5:D6 AC F7 EA 44 43 30 18:52
4A00:D3 D4 3B 3A 46 B2 8A FA:50
4A05:A5 AD F5 67 2A 63 DF 91:91
4A10:0C EB D1 0A D1 5A FD 98:0C
4A15:D4 AF 7C 6F D0 99 E3 B7:77
4A20:FC E9 27 01 EA D1 7D 64:14
4A25:C7 4E EA 07 EA 41 C6 4F:59
4A30:1F 77 2E 58 62 46 F5:E1
4A35:6C E8 0B 3D 1E AF A5:ED
4A40:2B 25 0F B7 CE BD 94 8E:FA
4A45:0C 74 97 F6 A4 64 3C DF:F4
4A50:58 69 A4 F3 24 55 8D AF:7A
4A55:08 B2 2E 2B 15 08 28 AF:5B
4A60:27 2C 1D 9C 94 AD 6D 0B:55
4A65:45 E5 70 AC EB 6B 8E 02:CC
4A70:8D 77 05 B3 6F 28 2D 1E:CA
4A75:DA B1 B7 6B 34 F2 82 6B:7D
4A80:27 8E 9E AE 09 48 A0 1:3D
4A85:D2 83 31 36 9E FE 76 0E:D3
4A90:96 C0 CD 72 75 74 24:1A
4A95:BB B3 77 78 0C 9A ED 31:03
4AA0:66 82 CF 4A 80 19 91 93:9A
4AA5:FB FE 8D FB 14 B5 98 BA
4AB0:6F 7A 12 91 12 02 A1 F1:2C
4AB5:8F 79 BD D7 EB 47 1D:BC
4AC0:51 38 DB C7 93 2D 7A 11:63
4AC5:9A B9 F2 11 C6 7D EE C8:CD
4AD0:09 02 D7 B8 FD 3B A2 A4:6D
4AD5:A9 0F 77 EE 3C 07 13 9D:09
4AE0:15 AF FD 43 87 D3 02 67:CB
4AE5:FD CE 47 E3 71 3D 73 8A:64
4AF0:12 BE 08 52 44 30 53:F4
4AF5:8D C3 A8 35 B9 FA 2B F9:8F

4B00:27 1A 33 96 D6 B2 DE 70:83
4B05:74 BC F2 D6 8C CC FC:39
4B10:2B 60 66 64 7E 00 93:07
4B15:EC 91 92 B8 06 FE 0C AF:F1
4B20:39 41 47 E7 27 12 DA 42:53
4B25:45 08 5D CA EC CF DE 32:C9
4B30:37 31 D9 69 A7 B2 30 B3:9A
4B35:A9 3A 6C 11 5A 20 D8 C2:F2
4B40:AE 1E 64 65 E5 B1 B9 D8:43
4B45:00 CE 2D FD 6C 09 DE CF:87:60
4B50:48 E0 EB 96 1E 16 EC 88:8A
4B55:0A 37 1A 0A CC 93 35 64:92
4B60:CC 15 B3 36 29 75 7A D1:FB
4B65:69 14 3F 65 26 04 83 9F:38
4B70:35 68 64 B4 F4 6A 81 DA:23
4B75:7A 6B D1 FD B9 DC 30 2F:07
4B80:E1 FD 44 84 CC 05 AD 85:BF
4B85:17 15 10 9F 40 FE 1A 58:5A
4B90:1D 68 05 E8 76 6E 92 10:07
4B95:78 62 59 CA B3 59 F9 80:70
4BA0:C5 64 55 1C 61 F2 5F 5A:E9
4BA5:20 98 A1 4C 0E 44 5F 26:FD
4BB0:55 0B 71 92 97 B6 B6 AF:7D
4BB5:CB C9 1F 18 06 09 B6 8E:AB
4BC0:B7 A3 10 08 C4 03 00 26:AE
4BC5:C9 B2 2D 8B 28 D9 A5:11
4BD0:49 BA BF 48 59 65 00 17:14
4BD5:7E 1D F5 17 52 9F 13 5E:DF
4BE0:C1 E7 17 6B 5A 1C 6A 26:2B
4BE5:0C 36 6C 5E 51 EF 49 01:09:D3
4BF0:81 F9 CA 28 5F 10 77 8A:68
4BF5:7E 70 FC 55 05 6F 83 2D:F9
4C00:11 4C A5 EC F0 68 CD 30:22
4C05:14 49 27 49 50 84 BC 55:19
4C10:B4 D9 2A F2 D0 0A 20 2F:AE
4C15:08 83 D5 93 58 97 0C 83:07
4C20:9F 72 84 33 CE CC 6B 89:CA
4C25:A8 36 88 8A 02 41 37 0C:DA
4C30:E2 CA 5C D2 78 03 6A 3D:39
4C35:49 5F CC AD E8 09 D4 42:47
4C40:17 1A 45 64 9A E3 66 14:F4
4C45:2D C8 1E 9F 68 1C 58 4A:8F
4C50:5C BC D3 DA EB AE 7E 58:2E
4C55:30 CC BA EA 8E 69 32 D1 50:88
4C60:53 7C 76 64 64 82 92 3D:01
4C65:BD 81 7D 64 D4 19 25 DB 3A:B2
4C70:E0 02 7B DB 26 B9 1E:0A
4C75:31 EE C8 E2 F1 D6 8E 0F:CA
4C80:F5 9E 9F 11 9D 7A 9D 87:8E
4C85:05 86 BB 88 AE 2F AD 18:DE
4C90:51 20 13 46 D0 91 EF DE:4D
4C95:08 0E 43 80 80 9C EF 81:03
4CA0:72 9C 6E 44 C3 3C 6A 1A:61
4CA5:20 00 D2 93 4D 79 31 BA:62
4CB0:63 0E D3 C6 21 71 58 B2:4F
4CB5:E7 85 86 52 BF 38 96 A7:AF
4CC0:A3 79 1E BD E1 13 75 9B:D1
4CC5:98 D1 29 FA 0A 8F 7F 11:BE
4CC8:0C EB 94 8B 49 8E 3E A4:39
4CD0:00 4B 1C 24 87 95 3B 55:04
4CD5:FC 3E F7 0C FB F4 37 ED:41
4CE0:30 CF 31 36 81 05 58 F8:D8
4CF0:16 04 36 39 5A 18 5F 5D:29
4CF5:1E 2A 17 EA 5B 53 01 B4:84
4D00:C2 2B 90 5D 9A 27 77 27:12
4D05:8E 52 F1 67 65 54 D4 C9:90
4D10:E6 15 2F 62 66 62 EA 19:19
4D15:A0 65 58 92 49 EA B5 E9:1F
4D20:B3 80 8E 0C 4D 3C 05 05:DF
4D25:C6 C9 36 12 1D 04 9A 3B:8A
4D30:91 15 B2 6E 42 C6 60 6D:1C
4D35:F8 D8 6E 43 6E CA 81 66:10
4D40:CB 8D EA CD 13 79 62 BA:14
4D45:B8 CA 74 55 AD 96 F4 F5:CA
4D50:4C C8 68 C3 EF 82 16 98:EE
4D55:20 74 2B EF 8D 0C 2C 52:0E
4D60:0F 96 F6 98 BF 11 14 F3:25
4D65:22 1C B5 DF EB 8B A8 A1:34
4D70:F2 B4 89 52 BB C3 25 72:C6

DEATHS TICKET

4D78:68	5C	D2	B8	77	BB	CF	F8:59	5018:7C	06	C0	C9	F1	F3	14	A9:9B	52B8:8C	24	8C	8C	8C	8C	8C	EA
4D80:53	C0	52	F6	86	2E	F2	83:11	5020:C6	C0	98	F0	F7	14	A9:92	AD	52C8:8C	8C	8C	C0	C8	C0	C8	C0
4D82:84	33	4F	18	B2	73	3F	B7:AA	5028:C0	20	A0	00	7C	43	00	30:E2	52C8:C8	C0	C8	04	00	04	80	04:12
4D90:2C	3B	1E	E2	58	04	D8	A6:49	5030:00	14	BC	7C	56	29	83	7C:E5	52D0:80	04	80	C0	C0	C0	C0	AA
4D98:83	5F	AE	80	20	C6	CE	61:EE	5038:56	14	BC	24	D8	48	84	8C:AE	52D8:C0	C0	C0	C3	83	C3	C3	C3:44
4DA0:F8	04	C4	C2	03	AB	48	D0:72	5040:4C	24	D8	00	00	4C	30	C0:04	52E0:C3	43	C3	03	43	03	03	03:5E
4DAB:92	2D	35	97	80	74	8F	AA:1D	5048:00	00	48	52	F0	00	40:78	52E8:03	83	03	F3	F3	F3	F3	F3:CE	
4DB0:1F	5F	5F	33	EC	C0	98	01:88	5050:C1	00	00	50	20	10	82	40:78	52F0:F3	F1	F1	F1	F1	F1	F1	F1:32
4DB2:3C	9F	0E	E7	45	1D	49	C6:F1	5058:03	51	20	70	02	00	00	92:47	52F8:F1	F1	F1	F1	F1	F1	F1	F1:10
4DB4:9B	F2	60	2A	E7	ED	6E	7B:3F	5060:C6	20	00	82	00	A2	84	F0:AE	5300:F0	F0	F0	00	00	00	00	F0:F3
4DC8:3F	00	AC	97	65	DB	C7	D3:28	5068:00	82	F0	C0	A0	00	00	92:34	5308:00	00	F0	00	F0	F0	F0	F0:7B
4DD0:4D	9E	82	EB	AE	DB	54	6D:FC	5070:00	00	06	10	82	51	43	50:D7	5310:F0	F0	F0	00	00	00	00	C0:02
4DD8:8B	89	6A	DA	BD	B2	A4	3B:EB	5078:58	44	C1	00	00	00	50	00:DB	5318:C3	00	00	91	33	00	00	C1:79
4DE0:11	7C	03	E4	74	64	B7	AA:AA	5080:00	00	00	00	00	00	40:00	5320:83	00	00	00	00	00	00	00:00	
4DE8:94	FD	E2	3A	11	C0	01	23:BC:AB	5088:70	00	00	00	00	4C	30	00:00	5328:03	40	08	33	8F	00	90	C3:35
4DF0:8C	EA	24	B3	CA	98	EE	FE:41	5090:00	00	48	52	F0	00	40:10	5330:56	00	00	98	33	8F	4C	70	83:40
4DF8:65	C3	FA	81	77	31	8E	26:AD	5098:C1	00	00	50	20	10	82	40:78	5338:56	40	70	67	0A	10	80	83:6F
4E00:04	07	3C	16	1E	10	77	83:3B	50A0:83	51	20	70	02	00	82	C0:DD	5340:BC	10	A0	67	0A	00	00	00:35
4E08:8E	FA	6C	0E	66	55	7E	13:43	50AC:C6	20	00	02	00	84	F0:24	5348:48	00	00	27	05	00	00	00	56:8A
4E10:02	FC	6F	33	25	08	F7	17:D3	50B0:00	00	A2	00	C2	0A	02	70:24	5350:3C	00	00	00	00	00	00	88:8A:BB
4E18:C5	09	1D	57	P5	31	AA	7E:0B	50B8:A0	00	06	10	82	40	00	50:19	5358:C0	98	70	4C	98	30	F0	44:EF
4E20:85	BB	CB	33	F4	B1	51	7F:AF	50C0:88	44	C1	00	00	50	20	00:23	5360:30	F0	A0	40	C0	C0	C0	C0:63
4E28:DA	37	04	F4	11	29	1C	3B:99	50C8:84	52	00	00	00	00	40:60	5368:C0	E2	F3	40	D1	D1	D1	40:81	
4E30:5E	B2	D8	6D	30	7E	59	11:DE	50D0:70	00	00	00	20	10	50:00	5370:C0	C0	C0	C0	C0	E2	F3	40:34	
4E38:14	8C	1E	94	80	00	07	40:1D	50E0:20	50	00	00	A0	00	10:78	5378:D1	D1	D1	A2	84	C0	98	70:40	
4E40:9F	49	84	2A	15	48	9A:DD	00:00	50E8:10	00	20	50	20	50	50:00	5380:98	30	F0	44	30	F0	40	C0:9B	
4E48:5E	C9	F7	F1	77	2E	00:00	00:00	50F0:50	A0	20	00	00	A0	20:00	5388:C0	E2	F3	40	D1	D1	D1	00:A1	
4E50:04	45	00	B3	2B	F1	5B	58:52	50F8:20	A0	A0	B0	B0	18	00:00	5390:98	00	00	C0	C0	C0	C0	C0:8A	
4E58:F1	45	AA	A5	52	7F	5A	50:3F	5100:30	24	70	64	70	70	24:9E	5398:98	30	F0	C4	30	F0	F1	40:40	
4E60:7B	3E	EF	E2	26	97	87	C3:7B	5108:00	00	90	60	70	24	30:71	53A8:F3	F3	A2	00	00	00	00	00:BA	
4E68:9F	82	E6	26	29	5D	C0	A8:6E	5110:24	70	30	04	88	00	00:6D	53B0:00	00	00	00	00	00	00	40:03	
4E70:83	55	AF	BF	77	7E	DA	65:7C	5118:00	00	00	80	B0	00	44:F9	53BA:C0	C0	80	C0	E2	F3	C8:CB		
4E78:C4	4E	2D	E2	71	7E	DA	E5:7C	5120:88	00	00	84	CC	44	88:0D	53C0:D1	D1	F2	44	30	F0	00	00:5B	
4E80:A7	9C	43	0E	84	5F	82	B9:1A	5128:10	00	00	00	8D	10	84:71	53C8:00	00	00	00	00	00	00	00:1B	
4E88:7C	22	EB	64	1B	57	18	73:95	5130:CC	00	00	00	00	84	C0:89	53D0:00	00	00	00	00	00	D1	E2:47	
4E90:70	01	AC	9A	16	DB	FD	30:87	5138:98	44	88	84	18	C0	84:FC	53D8:0C	00	F0	C8	C0	C0	D0	C0:6A	
4E98:0F	9C	FD	B2	49	F0	20	D8:DB	5140:18	10	00	90	CC	84	18	C0:35	53E0:C0	E2	F3	40	D1	D1	A2	F0:00
4EA0:8A	94	2F	D2	48	93	08	18:BB	5148:0C	84	C0	84	CC	84	C0	84:AE	53E8:00	00	00	00	00	00	00	40:39
4EA8:5B	F7	D2	36	73	06	97	3F:3F	5150:18	10	18	C0	C0	C1	83	C0:5C	53F0:C0	C0	A2	C0	D1	F3	F3	40:D5
4EAB:C4	F8	B0	9D	19	00	69:AF	00:00	5158:C0	C1	83	C0	C3	83	43	C0:2A	53F8:CC	00	88	84	00	C0	C0	C0:DC
4EB8:28	49	74	63	C5	0B	00	A9:9A	5160:83	83	43	C3	56	43	C1:87	5400:98	F0	F0	40	C0	C0	C0	C0:00	
4EC0:84	26	D2	14	BB	E8	94	62:A1	5168:93	00	00	00	00	00	00:00	5408:C0	E2	F3	40	D1	D1	D1	A2:40	
4EC8:CA	66	BC	46	FC	5B	80	BE:9E	5170:56	49	C1	83	56	29	83	C3:00	5410:98	00	00	00	00	00	00	C0:02
4ED0:FC	85	2C	5D	8C	A6	3C	24:24	5178:16	49	83	83	A9	43	86:97	5418:84	00	00	5C	84	4C	00	98:4F	
4ED8:52	E4	83	32	3E	DE	44	CF:85	5180:49	43	83	43	C3	56	83:E8	5420:CC	30	F0	C4	30	70	BA	C3:14	
4EE0:93	F3	85	13	BF	70	CF	10:56	5188:43	C3	56	43	C3	03	BC:43	5428:03	FC	3C	41	C3	03	80	00:A8	
4EE8:E7	6A	B1	84	9C	99	18	EE:62	5190:C3	03	00	C0	C3	83	03:EC	5430:84	98	40	4C	70	00	40	20:EC	
4EF0:9E	74	E5	7B	F0	F0	BE	FB:5D	5198:03	03	03	D6	FC	FC	FC:0E	5438:84	98	A0	84	98	A0	40	40:08	
4EF8:77	60	B2	69	EF	4F	33	33:89	51A0:FC	FC	FC	FC	FC	FC:0E	5440:20	40	4C	70	00	84	98	00	10:18	
4F00:9E	3C	75	F3	04	9C	5F:5D	00:00	51A8:FC	FC	56	A9	FC	A9	FC:2D	5448:C0	00	40	D1	80	40	E2	A2:66	
4F08:5E	5A	69	50	C8	0C	6B	7E:F1	51B0:49	FC	56	E9	FC	E9	FC:E2	5450:D1	D1	E2	F3	A2	51	A2	66:6A	
4F10:F1	09	05	1C	C8	47	69	18:12	51B8:E9	FC	56	FC	FC	FC:0E	5458:00	51	51	00	00	A2	00	40	00:00	
4F18:DF	44	66	13	A4	34	71	EE:3F	51C0:FC	FC	56	FC	FC	FC:0E	5460:98	84	98	0C	98	4C	70	00	00:EC	
4F20:BA	E1	5B	27	43	E1	D1	37:9C	51D0:FC	FC	3C	3C	3C	3C:49	5468:70	10	A0	00	08	84	98	58:54		
4F28:3F	65	4E	45	8A	C1	12	6C:0E	51E0:8C	00	3C	00	40	80	00:C9	5470:98	00	F0	D8	10	80	00	40:44	
4F30:46	C0	9A	D4	4D	1D	12	65:AF	51F0:40	40	84	40	84	00	00:AD	5478:98	84	98	00	10	00	50	00:54	
4F38:B6	3F	2E	07	9C	18	A8	69:DD	51E8:00	40	84	84	84	8C	80:00	5480:98	00	A0	00	00	00	00	00:00	
4F40:57	8C	48	70	91	C8	80	E3:B3	51E8:40	84	4C	98	64	8C	48:F9	5488:10	00	10	00	10	00	10	44:8C	
4F48:53	AB	18	60	1F	33	6B	7B:9A	51F0:84	4C	98	98	70	B0	64	8C:F1	5490:00	80	8C	84	00	00	10:00:14	
4F50:AC	B2	4B	9A	2C	95	6C	E9:8E	51F8:4C	98	70	A0	50	B0	44:C5	5498:50	10	A0	C0	C2	90	8B	45:45	
4F58:B7	77	C3	81	A1	0F	37	3C:79	5200:98	70	A0	A0	00	00	50:DA	54A0:6F	22	91	6C	30	78	24	D2:56	
4F60:02	A5	47	BA	0E	9C	D2	DD:BA	5208:70	A0	00	00	00	00	00:8A	54A8:87	56	B6	28	FE	DB	2C	0A:D7	
4F68:BC	FC	58	EA	5B	9E	09	20:53	5210:A0	00	00	00	00	00	00:82	54B0:95	00	5D	94	43	9F	F9	E2:84	
4F70:B6	09	4C	F2	11	3A	78	C9:74	5218:80	00	00	40	48	00	00:00	54B8:BE	3C	AC	BA	4F	4F	4E	76:57	
4F78:74	C8	47	78	F5	63	1C	CE:CF	5220:48	00	00	84	8C	00	84:AE	54C0:CD	6D	B9	46	A6	70	57	60:E3	
4F80:98	E5	34	34	F2	F5	F7	F9:BE	5228:8C	80	00	4C	64	48	40:4C:FA	54D0:CA	72	24</						

5558:F0 20 8E EE 88 33 56 98:33
5559:F6 33 16 1E 88 18 21:B3
5566:A6 B1 05 4E 5C 95 9C:FD:D6
5579:CF 05 D2 8F 71 AB BF 98:30
5578:CD 99 B8 5A CC CT 3B 2C:FF
5580:8D 73 97 BD 86 95 EC 83:2C
5588:2D 78 DE 86 25 E1 DF 1E:AE
5599:6C F4 B2 C3 F3 2A CD D9:B6
5598:CC F4 FE D9 23 07 BA D3:B6
5558:42 99 15 9A 75 85 3C 3E:0B
55A8:DA 2E CA 0D 8E E5 38 01:80
555A:36 AE 7D 95 CF E7 89 2F:A6
55B8:45 FE A3 0B 62 41 E3:88
55C8:25 89 B0 05 A0 CF 00 7F:62
55C8:34 56 3C 71 0D BD CA 1D:70
55D9:E4 EB A2 71 7A 42 C4 0E:D3
55D8:6F 76 DE 16 95 10 CC AD:BF
55E9:61 F5 97 CE 16 8A 8C D1:AF
55E8:DD 93 33 15 6C E1 BA C0:7B
55F9:89 70 B3 EB 83 7E EB 49:93
55F8:69 E6 22 11 20 53 20 2A:EE
5600:91 AD CB 84 90 EA 6B 11:23
5608:6C 9A 49 D8 81 E4 89 23:ED
5610:0F 43 93 5B 50 44 55 CF:53
5618:B2 F8 66 29 D1 57 89 AB:FC
5620:CE BD 61 14 4A BE 59 6C:A7
5628:0F DF D7 48 F2 57 BD FE:55
5639:E3 A4 AE E2 4E 39 64 44:34
5638:7D 22 6D 4F 39 61 89 46:99
5640:F6 EE B9 AF 95 52 B8 53:CA
5648:7B 1F 47 CC 30 C4 05 83:CE
5650:64 33 BC 9E A3 D3 DC 5C:91
5658:83 FA 73 E9 11 43 1C 4A:1D
5660:BF CD F0 B6 07 87 98 6C:8C
5668:88 D8 C9 BB DA A1 BB F8:22
5670:85 55 C6 4B 5C 08 19 40:1E
5678:B2 5E 76 20 92 4B 7C A9:A6
5680:39 48 15 73 F5 7F 10 69:5D
5688:30 3B 12 63 83 54 D2 90:8B
5690:F9 96 FB 89 88 56 5D 4F:16
5698:38 90 7A 3B 0E 5E 6D E2:07
56A8:53 DD 28 95 F9 FA C6 18:85
56A8:AA A9 3D D1 C2 75 FC BE:8F
56B8:7B 45 25 ED 15 6E AB 5A:38
56B8:06 88 BD AB E7 ED 82 CD:42
56C9:16 65 EF 0C DE 78 85 72:04
56C8:6B D2 8B 8A BB 84 06 43:F7
56D0:CD C5 E5 1A 7C B9 47 00:4A
56D8:E3 24 16 71 53 1D 76 93:7E
56E9:74 33 0E E3 9B 97 24 3D:0E
56E8:31 6D 47 F9 FC 9E 7A B0:78
56F9:1E 7A 1F FC C8 2F 80 02:02
56F9:89 0C 81 AD AE 14 88 E6:6C
5709:CF 7B FD 31 96 C5 7E 16:D4
5708:DD 48 D8 F5 3A 50 66 CD:3C
5710:92 74 63 61 87 0C BE CB:53
5718:1D CA FE ED 15 6E AB 5A:38
5729:98 13 19 AF 69 D2 73:4C
5728:33 0E 3C 4C 25 52 88 C2:1F
5730:5A 8B 9B BA 66 31 D5 0F:1F
5738:83 BB BD FD 16 D5 CC CC:FF
5740:7B 79 BC 8C 9E 2D B1 73:31
5748:5E 1E C9 69 8E 38 2D 35:31
5750:85 FE 44 E1 8D F5 8D 53:64
5758:51 39 5A F6 B1 7F 4C DE:CB
5760:92 D8 2E DD 43 B7 P5 94:93
5768:96 FE EE 60 6B 5D ED C2:D0
5770:DA F5 2C 78 DC A0 C2 3C:33
5778:59 21 01 BE 58 E8 89 7E:38
5780:39 D8 FD 74 9A 7A D2 49:87
5788:59 0D 72 D1 97 7E C2 E8:61
5790:73 7C BF 69 E4 26 37 C8:2C
5798:94 CE 5D 3C 0B 66 5D 76:6C
57A0:25 E8 65 3C 90 7A 92:59
57A8:50 D0 04 12 1F 53 25 19:9B
57A9:27 DE 49 68 EF F5 C6 70:7D
57B8:D4 D4 F9 94 06 C5 4A AA:AB
57C8:AE B8 CA BC 80 F3 75 17:A0
57C8:50 2E 72 5B D0 75 E9 8B:12
57D0:55 85 8E 6B AF 66 00 8D:BD
57D8:25 52 32 7D 88 13 9E 8A:5F
57E9:80 F6 35 8C 6D C0 4E 57:1D
57E9:FD E7 B9 BD 30 98 2F:0F
57F9:8D CA E1 67 99 1B F3 86:8F

57F8:A8 C5 9B D6 ED F0 2C CD:87
5800:D3 20 D3 D0 D3 20 D3 20:08
5808:D3 20 D3 D0 D3 20 D3 20:10
5810:D3 20 D3 D0 D3 20 D3 20:18
5818:D3 20 D3 D0 D3 20 D3 20:10
5820:D3 20 D3 D0 D3 20 D3 20:28
5828:D3 20 D3 D0 D3 20 D3 20:30
5830:D3 20 D3 D0 D3 20 D3 20:38
5838:D3 20 D3 D0 D3 20 D3 20:30
5840:D3 20 D3 D0 D3 20 D3 20:40
5848:D3 20 D3 D0 D3 20 D3 20:50
5850:D3 20 D3 D0 D3 20 D3 20:58
5858:D3 20 D3 D0 D3 20 D3 20:50
5860:D3 20 D3 D0 D3 20 D3 20:68
5868:D3 20 D3 D0 D3 20 D3 20:70
5870:D3 20 D3 D0 D3 20 D3 20:78
5878:D3 20 D3 D0 D3 20 D3 20:00
5880:D3 20 D3 D0 D3 20 D3 20:88
5888:D3 20 D3 D0 D3 20 D3 20:90
5890:D3 20 D3 D0 D3 20 D3 20:98
5898:D3 20 D3 D0 D3 20 D3 20:20
58A0:D3 20 D3 D0 D3 20 D3 20:A8
58A8:D3 20 D3 D0 D3 20 D3 20:B0
58B0:D3 20 D3 D0 D3 20 D3 20:B8
58B8:D3 20 D3 D0 D3 20 D3 20:40
58C0:D3 20 D3 D0 D3 20 D3 20:C8
58C8:D3 20 D3 D0 D3 20 D3 20:D8
58D0:D3 20 D3 D0 D3 20 D3 20:D8
58D8:D3 20 D3 D0 D3 20 D3 20:60
58E0:D3 20 D3 D0 D3 20 D3 20:E8
58E8:D3 20 D3 D0 D3 20 D3 20:F8
58F0:D3 20 D3 D0 D3 20 D3 20:F8
58F8:D3 20 D3 D0 D3 20 D3 20:80
5900:D3 20 D3 D0 D3 20 D3 20:09
5908:D3 20 D3 D0 D3 20 D3 20:11
5910:D3 20 D3 D0 D3 20 D3 20:19
5918:D3 20 D3 D0 D3 20 D3 20:A1
5920:D3 20 D3 D0 D3 20 D3 20:29
5928:D3 20 D3 D0 D3 20 D3 20:31
5930:D3 20 D3 D0 D3 20 D3 20:39
5938:D3 20 D3 D0 D3 20 D3 20:C1
5940:D3 20 D3 D0 D3 20 D3 20:49
5948:D3 20 D3 D0 D3 20 D3 20:55
5950:D3 20 D3 D0 D3 20 D3 20:59
5958:D3 20 D3 D0 D3 20 D3 20:E1
5960:D3 20 D3 D0 D3 20 D3 20:69
5968:D3 20 D3 D0 D3 20 D3 20:71
5970:D3 20 D3 D0 D3 20 D3 20:79
5978:D3 20 D3 D0 D3 20 D3 20:01
5980:D3 20 D3 D0 D3 20 D3 20:89
5988:D3 20 D3 D0 D3 20 D3 20:91
5990:D3 20 D3 D0 D3 20 D3 20:99
5998:D3 20 D3 D0 D3 20 D3 20:21
59A0:D3 20 D3 D0 D3 20 D3 20:A9
59A8:D3 20 D3 D0 D3 20 D3 20:B1
59B0:D3 20 D3 D0 D3 20 D3 20:B9
59B8:D3 20 D3 D0 D3 20 D3 20:43
59C0:D3 20 D3 D0 D3 20 D3 20:C9
59C8:D3 20 D3 D0 D3 20 D3 20:D1
59D0:D3 20 D3 D0 D3 20 D3 20:D9
59D8:D3 20 D3 D0 D3 20 D3 20:61
59E0:D3 20 D3 D0 D3 20 D3 20:E9
59E8:D3 20 D3 D0 D3 20 D3 20:F1
59F0:D3 20 D3 D0 D3 20 D3 20:F9
59F8:D3 20 D3 D0 D3 20 D3 20:81
5A00:D3 20 13 21 D3 13 21:9E
5A08:D3 20 13 21 D3 13 21:9E
5A10:D3 20 13 21 D3 13 21:A6
5A18:D3 20 13 21 D3 13 21:2E
5A20:D3 20 13 21 D3 13 21:B6
5A28:D3 20 13 21 D3 13 21:BE
5A30:D3 20 13 21 D3 13 21:C6
5A38:D3 20 13 21 D3 13 21:4E
5A40:D3 20 13 21 D3 13 21:4E
5A48:D3 20 13 21 D3 13 21:DE
5A50:D3 20 13 21 D3 13 21:DE
5A58:D3 20 13 21 D3 13 21:DE
5A60:D3 20 13 21 D3 13 21:F6
5A68:D3 20 13 21 D3 13 21:F6
5A70:D3 20 13 21 D3 20 13 21:0E
5A78:D3 20 13 21 D3 20 13 21:8E
5A80:D3 20 13 21 D3 20 13 21:1E
5A88:D3 20 13 21 D3 20 13 21:1E
5A90:D3 20 13 21 D3 20 13 21:2E

5A98:D3 20 13 21 D3 60 13 21:AE
5AA0:D3 20 13 21 D3 20 13 21:36
5AA8:D3 20 13 21 D3 20 13 21:3E
5AB0:D3 20 13 21 D3 20 13 21:46
5AB8:D3 20 13 21 D3 20 13 21:CE
5AC0:D3 20 13 21 D3 20 13 21:5E
5AC8:D3 20 13 21 D3 20 13 21:5E
5AD0:D3 20 13 21 D3 20 13 21:66
5AD8:D3 20 13 21 D3 20 13 21:EE
5AE0:D3 20 13 21 D3 20 13 21:76
5AE8:D3 20 13 21 D3 20 13 21:7E
5AF0:D3 20 13 21 D3 20 13 21:8E
5AF8:D3 20 13 21 D3 20 13 21:0E
5B00:13 21 13 21 13 21 13 21:1F
5B08:13 21 13 21 13 21 13 21:27
5B10:13 21 13 21 13 21 13 21:2F
5B18:13 21 13 21 13 21 13 21:3F
5B20:13 21 13 21 13 21 13 21:B7
5B28:13 21 13 21 13 21 13 21:47
5B30:13 21 13 21 13 21 13 21:4F
5B38:13 21 13 21 13 21 13 21:D7
5B40:13 21 13 21 13 21 13 21:5F
5B48:13 21 13 21 13 21 13 21:6F
5B50:13 21 13 21 13 21 13 21:6F
5B58:13 21 13 21 13 21 13 21:F7
5B60:13 21 13 21 13 21 13 21:7F
5B68:13 21 13 21 13 21 13 21:8F
5B70:13 21 13 21 13 21 13 21:8F
5B78:13 21 13 21 13 21 13 21:17
5B80:13 21 13 21 13 21 13 21:9F
5B88:13 21 13 21 13 21 13 21:A7
5B90:13 21 13 21 13 21 13 21:AF
5B98:13 21 13 21 13 21 13 21:3F
5BA0:13 21 13 21 13 21 13 21:BF
5BA8:13 21 13 21 13 21 13 21:C7
5BB0:13 21 13 21 13 21 13 21:CE
5BB8:13 21 13 21 13 21 13 21:5F
5BC0:13 21 13 21 13 21 13 21:DE
5BC8:13 21 13 21 13 21 13 21:E7
5BD0:13 21 13 21 13 21 13 21:EF
5BD8:13 21 13 21 13 21 13 21:7F
5BE0:13 21 13 21 13 21 13 21:FF
5BE8:13 21 13 21 13 21 13 21:0F
5BF0:13 21 13 21 13 21 13 21:0F
5BF8:13 21 13 21 13 21 13 21:97
5C00:13 21 13 21 13 21 13 21:28
5C08:13 21 13 21 13 21 13 21:28
5C10:13 21 13 21 13 21 13 21:30
5C18:13 21 13 21 13 21 13 21:B8
5C20:13 21 13 21 13 21 13 21:40
5C28:13 21 13 21 13 21 13 21:48
5C30:13 21 13 21 13 21 13 21:50
5C38:13 21 13 21 13 21 13 21:D8
5C40:13 21 13 21 13 21 13 21:68
5C48:13 21 13 21 13 21 13 21:68
5C50:13 21 13 21 13 21 13 21:78
5C58:13 21 13 21 13 21 13 21:F8
5C60:13 21 13 21 13 21 13 21:88
5C68:13 21 13 21 13 21 13 21:88
5C70:13 21 13 21 13 21 13 21:98
5C78:13 21 13 21 13 21 13 21:10
5C80:13 21 13 21 13 21 13 21:A0
5C88:13 21 13 21 13 21 13 21:A8
5C90:13 21 13 21 13 21 13 21:B0
5C98:13 21 13 21 13 21 13 21:38
5CA0:13 21 13 21 13 21 13 21:C0
5CA8:13 21 13 21 13 21 13 21:C8
5CB0:13 21 13 21 13 21 13 21:D0
5CB8:13 21 13 21 13 21 13 21:58
5CC0:13 21 13 21 13 21 13 21:E8
5CC8:13 21 13 21 13 21 13 21:E8
5CD0:13 21 13 21 13 21 13 21:F8
5CD8:13 21 13 21 13 21 13 21:78
5CE0:13 21 13 21 13 21 13 21:88
5CE8:13 21 13 21 13 21 13 21:88
5CF0:13 21 13 21 13 21 13 21:10
5CF8:13 21 13 21 13 21 13 21:98
5D00:53 21 53 21 53 21 53 21:21
5D08:53 21 53 21 53 21 53 21:21
5D10:53 21 53 21 53 21 53 21:31
5D18:53 21 53 21 53 21 53 21:41
5D20:53 21 53 21 53 21 53 21:49
5D28:53 21 53 21 53 21 53 21:49
5D30:53 21 53 21 53 21 53 21:51

DEATH'S TICKET

5D38:53	21	53	21	53	21:59
5D40:53	21	53	21	53	21:61
5D48:53	21	53	21	53	21:69
5D50:53	61	53	21	53	21:71
5D58:53	21	53	21	53	21:79
5D60:53	21	53	21	53	21:81
5D68:53	21	53	21	53	21:89
5D70:53	21	53	21	53	21:91
5D78:53	21	53	21	53	21:99
5D80:53	21	53	21	53	21:01
5D88:53	61	53	21	53	21:09
5D90:53	21	53	21	53	21:11
5D98:53	21	53	21	53	21:19
5DA0:53	21	53	21	53	21:21
5DA8:53	21	53	21	53	21:29
5DB0:53	21	53	21	53	21:31
5DB8:53	21	53	21	53	21:39
5DC0:53	21	53	21	53	21:41
5DC8:53	21	53	21	53	21:49
5DD0:53	21	53	21	53	21:51
5DD8:53	21	53	21	53	21:59
5DE0:53	21	53	21	53	21:01
5DE8:53	21	53	21	53	21:09
5DF0:53	21	53	21	53	21:11
5DF8:53	61	53	21	53	21:19
5E00:53	61	53	21	53	21:22
5E08:53	21	53	21	53	21:24
5E10:53	21	53	21	53	21:26
5E18:53	21	53	21	53	21:28
5E20:53	21	53	21	53	21:30
5E28:53	21	53	21	53	21:32
5E30:53	61	53	21	53	21:34
5E38:53	21	53	21	53	21:36
5E40:53	21	53	21	53	21:38
5E48:53	21	53	21	53	21:40
5E50:53	21	53	21	53	21:42
5E58:53	21	53	21	53	21:44
5E60:53	21	53	21	53	21:46
5E68:53	61	53	21	53	21:48
5E70:53	21	53	21	53	21:50
5E78:53	21	53	21	53	21:52
5E80:53	21	53	21	53	21:54
5E88:53	21	53	21	53	21:56
5E90:53	21	53	21	53	21:58
5E98:53	21	53	21	53	21:00
5EA0:53	61	53	21	53	21:02
5EA8:53	21	53	21	53	21:04
5EB0:53	21	53	21	53	21:06
5EB8:53	21	53	21	53	21:08
5EC0:93	21	93	21	93	21:10
5EC8:93	21	93	21	93	21:12
5ED0:93	21	93	21	93	21:14
5ED8:93	61	93	21	93	21:16
5EE0:93	21	93	21	93	21:18
5EE8:93	21	93	21	93	21:20
5EF0:93	21	93	21	93	21:22
5EF8:93	21	93	21	93	21:24
5F00:93	21	93	21	93	21:26
5F08:93	21	93	21	93	21:28
5F10:93	61	93	21	93	21:30
5F18:93	21	93	21	93	21:32
5F20:93	21	93	21	93	21:34
5F28:93	21	93	21	93	21:36
5F30:93	21	93	21	93	21:38
5F38:93	21	93	21	93	21:40
5F40:93	21	93	21	93	21:42
5F48:93	61	93	21	93	21:44
5F50:93	21	93	21	93	21:46
5F58:93	21	93	21	93	21:48
5F60:93	21	93	21	93	21:50
5F68:93	21	93	21	93	21:52
5F70:93	21	93	21	93	21:54
5F78:93	21	93	21	93	21:56
5F80:93	61	93	21	93	21:58
5F88:93	21	93	21	93	21:00
5F90:93	21	93	21	93	21:02
5F98:93	21	93	21	93	21:04
5FA0:93	21	93	21	93	21:06
5FA8:93	21	93	21	93	21:08
5FB0:93	21	93	21	93	21:10
5FB8:93	61	93	21	93	21:12
5FC0:93	21	93	21	93	21:14
5FC8:93	21	93	21	93	21:16
5FD0:93	21	93	21	93	21:18
5FD8:93	21	93	21	93	21:20
5FE0:93	21	93	21	93	21:22
5FE8:93	21	93	21	93	21:24
5FF0:93	61	93	21	93	21:26
5FF8:93	21	93	21	93	21:28
6000:93	21	93	21	93	21:30
6008:93	21	93	21	93	21:32
6010:93	21	93	21	93	21:34
6018:93	21	93	21	93	21:36
6020:93	21	93	21	93	21:38
6028:93	61	93	21	93	21:40
6030:93	21	93	21	93	21:42
6038:93	21	93	21	93	21:44
6040:93	21	93	21	93	21:46
6048:93	21	93	21	93	21:48
6050:93	21	93	21	93	21:50
6058:93	21	93	21	93	21:52
6060:93	61	93	21	93	21:54
6068:93	21	93	21	93	21:56
6070:93	21	93	21	93	21:58
6078:93	21	93	21	93	21:00
6080:93	21	93	21	93	21:02
6088:93	21	93	21	93	21:04
6090:93	21	93	21	93	21:06
6098:93	61	93	21	93	21:08
6100:93	21	93	21	93	21:10
6108:93	21	93	21	93	21:12
6110:93	21	93	21	93	21:14
6118:93	21	93	21	93	21:16
6120:93	61	93	21	93	21:18
6128:93	21	93	21	93	21:20
6130:93	21	93	21	93	21:22
6138:93	21	93	21	93	21:24
6140:93	61	93	21	93	21:26
6148:93	21	93	21	93	21:28
6150:93	21	93	21	93	21:30
6158:93	21	93	21	93	21:32
6160:93	21	93	21	93	21:34
6168:93	21	93	21	93	21:36
6170:93	21	93	21	93	21:38
6178:93	21	93	21	93	21:40
6180:93	21	93	21	93	21:42
6188:93	61	93	21	93	21:44
6190:93	21	93	21	93	21:46
6198:93	21	93	21	93	21:48
6200:93	21	93	21	93	21:50
6208:93	21	93	21	93	21:52
6210:93	21	93	21	93	21:54
6218:93	21	93	21	93	21:56
6220:93	21	93	21	93	21:58
6228:93	21	93	21	93	21:00
6230:93	21	93	21	93	21:02
6238:93	21	93	21	93	21:04
6240:93	21	93	21	93	21:06
6248:93	21	93	21	93	21:08
6250:93	21	93	21	93	21:10
6258:93	21	93	21	93	21:12
6260:93	21	93	21	93	21:14
6268:93	21	93	21	93	21:16
6270:93	21	93	21	93	21:18
6278:93	21	93	21	93	21:20
6280:93	21	93	21	93	21:22
6288:93	21	93	21	93	21:24
6290:93	21	93	21	93	21:26
6298:93	21	93	21	93	21:28
6300:93	21	93	21	93	21:30
6308:93	21	93	21	93	21:32
6310:93	21	93	21	93	21:34
6318:93	21	93	21	93	21:36
6320:93	21	93	21	93	21:38
6328:93	21	93	21	93	21:40
6330:93	21	93	21	93	21:42
6338:93	21	93	21	93	21:44
6340:93	21	93	21	93	21:46
6348:93	21	93	21	93	21:48
6350:93	21	93	21	93	21:50
6358:93	21	93	21	93	21:52
6360:93	21	93	21	93	21:54
6368:93	21	93	21	93	21:56
6370:93	21	93	21	93	21:58
6378:93	21	93	21	93	21:00
6380:93	21	93	21	93	21:02
6388:93	21	93	21	93	21:04
6390:93	21	93	21	93	21:06
6398:93	21	93	21	93	21:08
6400:93	21	93	21	93	21:10
6408:93	21	93	21	93	21:12
6410:93	21	93	21	93	21:14
6418:93	21	93	21	93	21:16
6420:93	21	93	21	93	21:18
6428:93	21	93	21	93	21:20
6430:93	21	93	21	93	21:22
6438:93	21	93	21	93	21:24
6440:93	21	93	21	93	21:26
6448:93	21	93	21	93	21:28
6450:93	21	93	21	93	21:30
6458:93	21	93	21	93	21:32
6460:93	21	93	21	93	21:34
6468:93	21	93	21	93	21:36
6470:93	21	93	21	93	21:38
6478:93	21	93	21	93	21:40
6480:93	21	93	21	93	21:42
6488:93	21	93	21	93	21:44
6490:93	21	93	21	93	21:46
6498:93	21	93	21	93	21:48
6500:93	21	93	21	93	21:50
6508:93	21	93	21	93	21:52
6510:93	21	93	21	93	21:54
6518:93	21	93	21	93	21:56
6520:93	21	93	21	93	21:58
6528:93	21	93	21	93	21:00
6530:93	21	93	21	93	21:02
6538:93	21	93	21	93	21:04
6540:93	21	93	21	93	21:06
6548:93	21	93	21	93	21:08
6550:93	21	93	21	93	21:10
6558:93	21	93	21	93	21:12
6560:93	21	93	21	93	21:14
6568:93	21	93	21	93	21:16
6570:93	21	93	21	93	21:18
6578:93	21	93	21	93	21:20
6580:93	21	93	21	93	21:22
6588:93	21	93	21	93	21:24
6590:93	21	93	21	93	21:26
6598:93	21	93	21	93	21:28
6600:93	21	93	21	93	21:30

6518:13	22	13	22	13	62	13	22	D5	67B8:53	22	53	22	53	62	53	22	77	6A58:93	22	93	22	93	22	93	22	93	22	9A
6520:13	22	13	22	13	62	13	22	D5	67C0:53	22	53	22	53	62	53	22	77	6A60:93	22	93	22	93	22	93	22	93	22	A2
6528:13	22	13	22	13	62	13	22	R5	67C8:53	22	53	22	53	62	53	22	82	6A68:93	62	93	22	93	22	93	22	93	22	2A
6530:13	22	13	22	13	62	13	22	D6	67D0:53	22	53	22	53	62	53	22	92	6A70:93	22	93	22	93	22	93	22	93	22	B2
6538:13	22	13	22	13	62	13	22	F5	67D8:53	22	53	22	53	62	53	22	97	6A78:93	22	93	22	93	22	93	22	93	22	3A
6540:13	22	13	22	13	62	13	22	7D	67E0:53	22	53	22	53	62	53	22	1F	6A80:93	22	93	22	93	22	93	22	93	22	C2
6548:13	22	13	22	13	62	13	22	9D	67E8:53	22	53	22	53	62	53	22	A7	6A88:93	22	93	22	93	22	93	22	93	22	CA
6550:13	22	13	22	13	62	13	22	8D	67F0:53	22	53	22	53	62	53	22	2F	6A90:93	62	93	22	93	22	93	22	93	22	52
6558:13	22	13	22	13	62	13	22	15	67F8:53	22	53	22	53	62	53	22	B7	6A98:93	22	93	22	93	22	93	22	93	22	DA
6560:13	22	13	22	13	62	13	22	9D	6800:53	22	53	22	53	62	53	22	A0	6AA0:93	22	93	22	93	22	93	22	93	22	62
6568:13	22	13	22	13	62	13	22	25	6808:53	22	53	22	53	62	53	22	C8	6AA8:93	22	93	22	93	22	93	22	93	22	EA
6570:13	22	13	22	13	62	13	22	AD	6810:53	22	53	22	53	62	53	22	59	6AB0:93	22	93	22	93	22	93	22	93	22	7A
6578:13	22	13	22	13	62	13	22	35	6818:53	22	53	22	53	62	53	22	D6	6AB8:93	22	93	22	93	22	93	22	93	22	8A
6580:13	22	13	22	13	62	13	22	BD	6820:53	22	53	22	53	62	53	22	E0	6AC0:93	22	93	22	93	22	93	22	93	22	92
6588:13	22	13	22	13	62	13	22	45	6828:53	22	53	22	53	62	53	22	F8	6AC8:93	22	93	22	93	22	93	22	93	22	8A
6590:13	22	13	22	13	62	13	22	CD	6830:53	22	53	22	53	62	53	22	79	6AD0:93	22	93	22	93	22	93	22	93	22	12
6598:13	22	13	22	13	62	13	22	55	6838:53	22	53	22	53	62	53	22	F8	6AD8:93	22	93	22	93	22	93	22	93	22	1A
65A0:13	22	13	22	13	62	13	22	DD	6840:53	22	53	22	53	62	53	22	80	6AE0:93	62	93	22	93	22	93	22	93	22	A2
65A8:13	22	13	22	13	62	13	22	65	6848:53	22	53	22	53	62	53	22	98	6AE8:93	22	93	22	93	22	93	22	93	22	2A
65B0:13	22	13	22	13	62	13	22	ED	6850:53	22	53	22	53	62	53	22	90	6AF0:93	22	93	22	93	22	93	22	93	22	B2
65B8:13	22	13	22	13	62	13	22	75	6858:53	22	53	22	53	62	53	22	18	6AF8:93	22	93	22	93	22	93	22	93	22	3A
65C0:13	22	13	22	13	62	13	22	FD	6860:53	22	53	22	53	62	53	22	A0	6B00:93	22	93	22	93	22	93	22	93	22	43
65C8:13	22	13	22	13	62	13	22	85	6868:53	22	53	22	53	62	53	22	B8	6B08:93	62	93	22	93	22	93	22	93	22	CB
65D0:13	22	13	22	13	62	13	22	9D	6870:53	22	53	22	53	62	53	22	20	6B10:93	22	93	22	93	22	93	22	93	22	53
65D8:13	22	13	22	13	62	13	22	95	6878:53	22	53	22	53	62	53	22	38	6B18:93	22	93	22	93	22	93	22	93	22	DB
65E0:13	22	13	22	13	62	13	22	1D	6880:53	22	53	22	53	62	53	22	C0	6B20:93	22	93	22	93	22	93	22	93	22	63
65E8:13	22	13	22	13	62	13	22	A5	6888:53	22	53	22	53	62	53	22	A8	6B28:93	22	93	22	93	22	93	22	93	22	63
65F0:13	22	13	22	13	62	13	22	2D	6890:53	22	53	22	53	62	53	22	D0	6B30:93	22	93	22	93	22	93	22	93	22	63
65F8:13	22	13	22	13	62	13	22	B5	6898:53	22	53	22	53	62	53	22	5E	6B38:93	22	93	22	93	22	93	22	93	22	7B
6600:13	22	13	22	13	62	13	22	35	689A:93	22	93	22	93	62	93	22	70	6B40:93	22	93	22	93	22	93	22	93	22	83
6608:13	22	13	22	13	62	13	22	C6	689B:93	22	93	22	93	62	93	22	7E	6B48:93	22	93	22	93	22	93	22	93	22	9B
6610:13	22	13	22	13	62	13	22	1E	689C:93	22	93	22	93	62	93	22	79	6B50:93	22	93	22	93	22	93	22	93	22	93
6618:13	22	13	22	13	62	13	22	D6	689D:93	22	93	22	93	62	93	22	7F	6B58:93	22	93	22	93	22	93	22	93	22	9B
6620:53	22	53	22	53	62	53	22	5E	689E:93	22	93	22	93	62	93	22	80	6B60:93	22	93	22	93	22	93	22	93	22	23
6628:53	22	53	22	53	62	53	22	B6	689F:93	22	93	22	93	62	93	22	98	6B68:93	22	93	22	93	22	93	22	93	22	AB
6630:53	22	53	22	53	62	53	22	F6	68D0:93	22	93	22	93	62	93	22	10	6B70:93	22	93	22	93	22	93	22	93	22	B3
6638:53	22	53	22	53	62	53	22	E6	68D8:93	22	93	22	93	62	93	22	98	6B78:93	22	93	22	93	22	93	22	93	22	3B
6640:53	22	53	22	53	62	53	22	7E	68E0:93	22	93	22	93	62	93	22	20	6B80:93	22	93	22	93	22	93	22	93	22	C3
6648:53	22	53	22	53	62	53	22	86	68E8:93	22	93	22	93	62	93	22	28	6B88:93	22	93	22	93	22	93	22	93	22	CB
6650:53	22	53	22	53	62	53	22	8E	68F0:93	22	93	22	93	62	93	22	30	6B90:93	22	93	22	93	22	93	22	93	22	53
6658:53	22	53	22	53	62	53	22	16	68F8:93	22	93	22	93	62	93	22	38	6B98:93	22	93	22	93	22	93	22	93	22	DB
6660:53	22	53	22	53	62	53	22	9E	6900:93	22	93	22	93	62	93	22	C1	6BA0:93	22	93	22	93	22	93	22	93	22	E3
6668:53	22	53	22	53	62	53	22	26	6908:93	22	93	22	93	62	93	22	A9	6BA8:93	22	93	22	93	22	93	22	93	22	6B
6670:53	22	53	22	53	62	53	22	5A	6910:93	22	93	22	93	62	93	22	D1	6BB0:93	22	93	22	93	22	93	22	93	22	F3
6678:53	22	53	22	53	62	53	22	36	6918:93	22	93	22	93	62	93	22	59	6BB8:93	22	93	22	93	22	93	22	93	22	FB
6680:53	22	53	22	53	62	53	22	46	6920:93	22	93	22	93	62	93	22	61	6BC0:93	22	93	22	93	22	93	22	93	22	7D
6688:53	22	53	22	53	62	53	22	46	6928:93	22	93	22	93	62	93	22	89	6BC8:93	22	93	22	93	22	93	22	93	22	9B
6690:53	22	53	22	53	62	53	22	CE	6930:93	22	93	22	93	62	93	22	71	6BD0:93	22	93	22	93	22	93	22	93	22	13
6698:53	22	53	22	53	62	53	22	56	6938:93	22	93	22	93	62	93	22	F9	6BD8:93	22	93	22	93	22	93	22	93	22	9B
66A0:53	22	53	22	53	62	53	22	DE	6940:93	22	93	22	93	62	93	22	81	6BE0:93	22	93	22	93	22	93	22	93	22	23
66A8:53	22	53	22	53	62	53	22	66	6948:93	22	93	22	93	62	93	22	89	6BE8:93	22	93	22	93	22	93	22	93	22	2B
66B0:53	22	53	22	53	62	53	22	EE	6950:93	22	93	22	93	62	93	22	11	6BF0:93	22	93	22	93	22	93	22	93	22	B3
66B8:53	22	53	22	53	62	53	22	76	6958:93	22	93	22	93	62	93	22	99	6BF8:93	22	93	22	93	22	93	22	93	22	3B
66C0:53	22	53	22	53	62	53	22	F6	6960:93	22	93	22	93	62	93	22	21	6C00:93	22	93	22	93	22	93	22	93	22	44
66C8:53	22	53	22	53	62	53	22	86	6968:93	22	93	22	93	62	93	22	A9	6C08:93	22	93	22	93	22	93	22	93	22	CC
66D0:53	22	53	22	53	62	53	22	9E	6970:93	22	93	22	93	62	93	22	B1	6C10:93	22	93								

DEATH'S TICKET

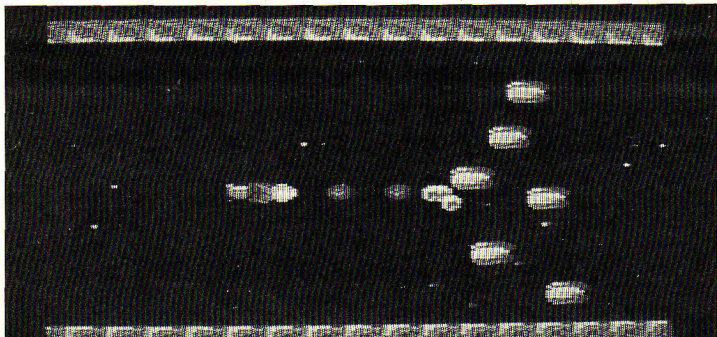
6CF8:D3 22 D3 22 D3 62 D3 22:BC
 4D09:D3 22 D3 22 D3 22 D3 22:45
 6D08:D3 22 D3 22 D3 22 D3 22:4D
 6D19:D3 22 D3 22 D3 62 D3 22:D5
 6D18:D3 22 D3 22 D3 22 D3 22:5D
 6D29:D3 22 D3 22 D3 22 D3 22:65
 6D28:D3 22 D3 22 D3 62 D3 22:ED
 6D39:D3 22 D3 22 D3 22 D3 22:75
 6D38:D3 22 D3 22 D3 22 D3 22:7D
 6D48:D3 22 D3 22 D3 62 D3 22:95
 6D49:D3 22 D3 22 D3 22 D3 22:95
 6D59:D3 22 D3 22 D3 22 D3 22:95
 6D58:D3 22 D3 22 D3 62 D3 A2:1D
 6D69:13 23 13 23 13 23 13:3B
 6D68:13 23 13 23 13 23 13:C1
 6D78:13 23 13 23 13 23 13:C9
 6D79:13 23 13 23 13 23 13:C1
 6D88:13 23 13 23 13 23 13:D9
 6D89:13 23 13 23 13 23 13:D9
 6D98:13 23 13 23 13 23 13:E1
 6D99:13 23 13 23 13 63 13:69
 6D98:13 23 13 23 13 23 13:F1
 6DA9:13 23 13 23 13 23 13:F9
 6DA8:13 23 13 23 13 23 13:01
 6DB9:13 63 13 23 13 23 13:89
 6DB8:13 23 13 23 13 23 13:11
 6DC9:13 23 13 23 13 23 13:19
 6DC8:13 23 13 23 13 63 13:A1
 6DD9:13 23 13 23 13 23 13:29
 6DE9:13 23 13 23 13 23 13:31
 6DE8:13 23 13 23 13 23 13:39
 6DE8:13 23 13 23 13 23 13:C1
 6DF9:13 23 13 23 13 23 13:49
 6DF8:13 23 13 23 13 23 13:51
 6E09:13 23 13 23 13 63 13:DA
 6E08:13 23 13 23 13 23 13:62
 6E18:13 23 13 23 13 23 13:6A
 6E19:13 23 13 23 13 23 13:72
 6E28:13 63 13 23 13 23 13:FA
 6E29:13 23 13 23 13 23 13:82
 6E39:13 23 13 23 13 23 13:8A
 6E38:13 23 13 23 13 63 13:12
 6E49:13 23 13 23 13 23 13:9A
 6E48:13 23 13 23 13 23 13:A2
 6E59:13 23 13 23 13 23 13:AA
 6E58:13 23 13 23 13 23 13:AA
 6E69:13 23 13 23 13 23 13:32
 6E68:13 23 13 23 13 23 13:C2
 6E79:13 23 13 23 13 63 13:4A
 6E78:13 23 13 23 13 23 13:D2
 6E89:13 23 13 23 13 23 13:DA
 6E88:13 23 13 23 13 23 13:E2
 6E99:13 63 13 23 13 23 13:EA
 6E98:13 23 13 23 13 23 13:F2
 6EA9:13 23 13 23 13 23 13:FA
 6EA8:13 23 13 23 13 63 13:82
 6EB9:13 23 13 23 13 23 13:0A
 6EB8:13 23 13 23 13 23 13:12
 6EC9:13 23 13 23 13 23 13:1A
 6EC8:13 23 13 23 13 23 13:A2
 6ED9:13 23 13 23 13 23 13:2A
 6ED8:13 23 13 23 13 23 13:32
 6EE9:13 23 13 23 13 63 13:3A
 6EE8:13 23 13 23 13 23 13:42
 6EF9:13 23 13 23 13 23 13:4A
 6EF8:13 23 13 23 13 23 13:52
 6F09:13 63 13 23 13 23 13:DB
 6F08:13 23 13 23 13 23 13:63
 6F19:13 23 13 23 13 23 13:6B
 6F18:13 23 13 23 13 63 13:3F
 6F29:13 23 13 23 13 23 13:7B
 6F28:13 23 13 23 13 23 13:83
 6F39:13 23 13 23 13 23 13:8B
 6F38:13 63 13 23 13 23 13:13
 6F49:13 23 13 23 13 23 13:9B
 6F48:13 23 13 23 13 23 13:A3
 6F59:13 23 13 23 13 63 13:2B
 6F58:13 23 13 23 13 23 13:3B
 6F69:13 23 13 23 13 23 13:3B
 6F68:13 23 13 23 13 23 13:C3
 6F79:13 63 13 23 13 23 13:4B
 6F78:13 23 13 23 13 23 13:D3
 6F89:13 23 13 23 13 23 13:DB
 6F88:13 23 13 23 13 63 13:63
 6F99:13 23 13 23 13 23 13:EB

6F98:13 23 13 23 13 23 13:23
 6FA9:13 23 13 23 13 23 13:23
 6FAB:13 63 13 23 13 23 13:23
 6FBB:13 23 13 23 13 23 13:23
 6FBA:13 23 13 23 13 23 13:23
 6FC9:13 23 13 23 13 63 13:23
 6FC8:13 23 13 23 13 23 13:23
 6FD9:13 23 13 23 13 23 13:23
 6FDA:13 23 13 23 13 23 13:23
 6FEE:13 63 13 23 13 23 13:BB
 6FEB:13 23 13 23 13 23 13:43
 6FEF:13 23 13 23 13 23 13:4B
 6FF8:13 23 13 23 13 E3 13:23

8000:CD E2 8D CD B0 8C 01 0D:13
 8009:BC ED 49 01 00 BD ED 49:2E
 8010:01 0C BC ED 49 01 39 BD:3C
 8018:ED 49 21 0A 95 CD EB 8C:26
 8019:ED 21 1B 95 CD EB 8C 21 35:58
 8028:95 CD EB 8C 21 4F 95 CD:C2
 8039:EB 8C 21 6A 95 CD EB 8C:42
 8038:21 7D 95 CD EB 8C 21 97:44
 8049:95 CD EB 8C 21 A9 95 CD:F6
 8048:EB 8C 21 6A 8E 3E 40 CD:3C
 8059:48 8E C2 63 80 3E 41 CD:29
 8058:48 8E C2 7C 80 CD A3 8E:A5
 8069:04 3A 4A 8E 3E 48 32 F0 8E:43
 8068:3A 32 F1 8E 3E 32 5E 0D:7E
 8079:3C 3E 8E 90 3C 32 82:EB
 8078:8E 3C 92 89 AF 3E 8E:59
 8089:3C 32 0F 8E 3C 32 1E:04
 8088:3E 0F 8E 3E 3E 3E 21:82
 8099:F2 8E CD B0 8C 21 07 8F:70
 8098:11 08 8F 01 AF 04 36 00:E7
 80A9:ED B0 AF 32 EC 8E 32 ED:F6
 80A8:8E 32 F6 8E 32 F3 8E:52
 80B9:F9 8E 32 02 8F 32 FE:3C
 80B8:32 FF 8E 32 00 8F 32 01:9A
 80C9:8F 8E 32 FA 8E 32 FB 8E:3A
 80C8:8F 8E 32 FD 8E 32 95 8F:77
 80D9:32 EA 8E 32 EB 8E 3E 01:6D
 80D8:32 FB 8E 32 90 93 8E:C4
 80E9:32 8E 32 90 8E 32 00:95
 80E8:42 22 DA 8E 22 D6 8E CD:9A
 80F9:68 8E 21 D8 95 07 8D:F6
 80F8:21 FB 95 CD 07 8D CD 4D:F6
 8109:8D CD 76 8D CD 6A 8E CD:05
 8108:A3 8E 3E 42 CD 48 8E:C1A4
 8119:03 80 3A F3 8E B7 C2 81:AD
 8118:8E 3A EE 8E CD 48 8E:C48
 8129:F2 82 3A EF 8E CD 48 8E:F6
 8128:04 01 83 3A F0 8E CD 48:F6
 8139:8E C4 35 83 3A F1 8E CD:8A
 8138:48 8E C4 12 83 3A F2 8E:8A
 8149:CD 48 8E 28 10 3A FE 8F:B6
 8148:B7 20 8E 3E 01 32 02 8F:99
 8159:8E 8E 8F 28 07 3D 32 00:D8
 8158:8F 2A EC 8E 23 22 EC:17
 8169:7C FE 39 CA 66 8E 3A F5:71
 8168:8E B7 28 04 CD 32 F5 8E:ED
 8179:CD A4 89 2A D4 8E 11 00:35
 8178:20 01 07 07 CD 71 82 CD:E9
 8189:33 8B CD 38 89 CD 3A 82:B2
 8188:3A 02 8F B7 CA D8 89 3A:43
 8199:FD 8E EE 01 32 FD 8E 28:02
 8198:0D 3A 01 8F B7 28 07 3D:75
 81A9:32 01 8F CA 71 84 3A FC:D5
 81A8:8E EE 01 32 FC 8E 20 0D:E6
 81B9:3A 09 8F B7 28 07 3D 32:21
 81B8:00 8F CA 22 84 3A FB 8E:68
 81C9:EE 01 32 FB 8E 29 8D 3A:64
 81C8:FF 8E B7 28 07 3D 32 FF:10
 81D9:8E CD D3 83 3A FA 8E:3C
 81D8:01 32 FA 8E 28 0D 3A FE:80
 81E9:8E B7 28 07 3D 32 FE:80
 81E8:C4 D3 83 06 F5 ED 78 1F:0B
 81F9:30 FB 3A 03 8F EE 20 32:20
 81F8:03 8F 01 0D BC ED 49 01:12
 8209:00 8D ED 49 01 0C BC ED:C0
 8208:49 04 ED 79 21 90 0E 2B:EA
 8219:7D BA 20 FB 2A D6 8E 01:83
 8218:07 07 CD C3 82 CD FD 88:A5
 8229:CD 83 88 2A D4 8E 22 D6:CB

8228:8E 3A F6 8E B7 C4 9F 86:8A
 8239:AF 32 F6 8E 32 07 8F C3:EA
 8238:94 81 3A 03 8F 07 1E:90
 8249:00 2E C1 67 8E 07 57 CD:2C
 8248:59 82 3A 03 8F 07 1E:F7
 8259:80 2E 81 P6 04 8E 80:99
 8258:57 06 08 05 C5 05 D1 01:3F
 8269:00 ED B0 D1 3E 08 82 57:CB
 8268:8E 31 3E 84 67 C1 19 EB:C0
 8279:C9 79 32 82 82 3A 03 8F:BE
 8278:07 07 0E B0 84 67 C5 8E:52
 8289:D5 06 0A 1B 78 28 15 E6:91
 8288:AA 28 06 4F 7E E6 55 B1:07
 8299:77 1A 1E 55 28 06 4F 7E:C8
 8298:EA AA 11 77 23 7D E6 3F:22
 82A9:28 03 10 0D D1 3A 82:AB
 82A8:82 6F 26 09 19 EB 81 7C:02
 82B9:C6 08 67 E6 38 28 7D:CD
 82B8:C6 40 67 FC CE C0 67 C1:1C
 82C9:10 BC C9 79 32 D3 82 3A:53
 82C8:03 8F 07 07 EE 80 B4 67:66
 82D9:C5 05 01 00 5D 54 13:16
 82D8:36 00 0D ED B0 E1 7C 6A:AF
 82E9:08 67 E6 38 20 08 7D C6:35
 82E8:40 67 FC CE C0 67 C1 10:25
 82F9:DF C9 2A D4 8E 7D B6 3F:A7
 82F8:C8 2D 22 D4 8E C3 8E:BC
 8309:C9 2A D4 8E 7D E6 3F FE:D2
 8308:3A C8 C2 22 D4 8E CD A3:8E
 8319:8E C9 2A D4 8E 7D E6 C9:7F
 8318:FE 40 28 06 7E 06 FE:82
 8329:04 C8 CD B0 8C CD DB 8C:F1
 8328:CD DB 8C CD DB 8C D4:23
 8339:8E CD A3 8E C9 2A 84 8E:21
 8338:7D E6 C9 04 7C FE 41:BA
 8349:CB CD CB CD CB 8C CD:B5
 8348:CB 8C CD CB 8C 2D 84 8E:05
 8359:CD A3 8E C9 CD A3 8E E6:99
 8358:03 CA 69 83 FC 01 CA 6F:B3
 8369:8F FE 02 CA 75 83 7B:18
 8368:83 3E 08 32 7E 8E C9 3E:83
 8379:08 32 FF 8E C9 3E 08 32:BD
 8378:00 8F C9 3E 08 32 01 8F:3F
 8389:C9 CD B7 89 78 B7 C8 CD:31
 8388:A3 8E 0E 0F FE 0E 30 F7:0A
 8399:00 8D 8E 00 3D 8E 28 06:26
 8398:00 99 3D 30 FC DD 75 01:89
 83A9:DD 74 02 DD 75 07 DD 74:80
 83A8:08 11 47 24 DD 73 03 DD:BA
 83B9:72 04 01 03 08 DD 71 05:51
 83B8:DD 70 06 AF DD 77 0A DD:DF
 83C9:77 02 3E 01 DD 79 09 3E:0C
 83C8:07 DD 77 00 3A 8E 8D:2D
 83D9:77 0C C9 CD B7 89 8E 8D:2D
 83D8:C8 CD A3 8E 06 0F FE 0E:18
 83E9:30 F7 01 21 3F 41 07 07:3B
 83E8:01 40 00 89 3D 20 FC DD:07
 83F9:75 01 DD 74 02 DD 75 07:CD
 83F8:DD 74 08 11 2F 24 DD 73:02
 8409:DD 74 08 04 01 03 DD 72:DE
 8408:71 00 DD 76 AF DD 77 09:99
 8419:0A 3E 01 DD 77 09 3E 06:F6
 8418:DD 77 00 3A F8 8E DD 77:3E
 8429:0C C9 CD B7 89 78 B7 C8:43
 8428:CD A3 8E 06 0F FE 0E 30:22
 8439:F7 01 21 3F 41 26 07 01:E6
 8438:40 00 09 3D 20 FC DD 75:46
 8449:01 DD 74 08 12 DD 75 07 DD:0B
 8448:74 04 01 0F 24 DD 73 03:DE
 8459:DD 72 04 01 0A 08 DD 71 7C:70
 8458:95 DD 70 06 AF DD 77 0A:2D
 8469:3E 01 DD 77 09 3E 05 DD:43
 8468:77 3A F8 8E DD 77 0C:8E
 8479:C9 CD B7 89 78 B7 C8 CD:22
 8478:A3 8E 0E 0F FE 0E 30 F7:0A
 8489:DD 74 08 01 03 DD 75 01:89
 8488:00 99 3D 20 FC DD 75 01:89
 8499:DD 74 02 DD 75 07 DD 74:71
 8498:08 11 6F 2D DD 73 03 DD:1F
 84A9:72 04 01 0A 08 DD 71 05:46
 84A8:DD 70 06 AF DD 77 0A 3E:08
 84B9:01 DD 77 09 3E 03 DD 77:83
 84B8:00 3A F8 8E DD 77 0C 09:87
 84C9:CD CD 85 DD 6E 01 86:15

84C8:02 7D E6 3F CA C8 85 2B:93	8648:DB 8C CD DB 8C CD 6B 87:43	87C0:DD 77 00 3E 01 DD 77 09:C6
84D0:7D E6 3F CA C8 85 2B 7D:9D	8650:E1 23 23 23 CD 6B 87:DD:0E	87C8:DD 77 0A DD 77 0B C9:DD:A8
84D8:E6 3F CA C8 85 2B DD 7E:D4	8658:E1 21 00 42 22 DA 8E C9:55	87D0:7E 0B 3C E6 01 DD 77 0B:03
84E0:9A B7 28 32 3D DD 77 0A:0C	8660:9C 01 32 F3 8E C9 21 E3:03	87D8:C8 DD 7E 0A 3C FE 06 28:0D
84EB:DD 7E 0D B7 28 14 7D E6:23	8668:94 CD EB 8C 21 F6 94 CD:EA	87E0:14 DD 77 0A DD 6E 03 DD:A4
84F0:C0 FE C0 20 08 7C E6 47:82	8670:EB 8C 06 14 21 00 00 2B:58	87E8:66 04 01 0C 00 09 DD 75:F9
84F8:FE 41 CA 65 87 CD DB 8C:BC		87F0:03 DD 74 04 C9 AF DD 77:6A



SCORE: 0000 LIFE: 09 RADIO: 00
TYPE: SIMPLE

8500:18 26 7D E6 C0 FE 40 20:6C	8678:7D B4 20 FB 10 F6 C3 03:B0	87F8:00 C9 DD 7E 00 FE 03 20:A9
8508:08 7C E6 97 FE 04 CA 65:17	8680:80 21 C5 95 CD EB 8C 06:F2	8800:10 21 03 96 1E 78 3E 06:83
8510:07 CD CB 8C 18 12 E5 CD:D6	8688:06 21 00 00 2B 7D B4 20:07	8808:CD C8 8D 21 03 96 C3 07:38
8518:A3 8E E6 91 DD 77 0D CF:F9	8690:7B 19 F6 C3 03 89 DD 6E:A9	8810:8D FE 05 20 10 21 03 96:8B
8520:A3 8E E1 E6 0F DD 77 0A:A9	8698:01 DD 66 02 22 EA 8E DD:DF	8818:1E 96 3E 06 CD C8 8D 21:50
8528:DD 75 01 DD 74 02 C9 CD:22	86A0:22 E8 8E CD 4C 88 ED 53:B5	8820:03 96 C3 07 8D FE 06 20:1B
8530:CD 85 DD 6E 01 DD 66 02:E8	86A8:E2 8E DD 21 07 8F 06 64:0E	8828:10 21 03 96 1E C8 3E 06:8B
8538:7D E6 3F CA C8 85 2B DD:0A	86B0:C5 DD 7E 00 FE 03 20 05:F3	8830:CD C8 8D 21 03 96 C3 07:60
8540:7E 0A 3C FE 10 DD 77 0A:12	86B8:CD DD 86 18 17 FE 05 20:41	8838:8D FE 07 C0 21 03 96 1E:1F
8548:38 15 7D E6 3F CA C8 85:D4	86C0:05 CD DD 86 18 0E FE 06:82	8840:FF 3E 06 CD C8 8D 21 03:BE
8550:2B DD 7E 0A FE 1D 38 07:C0	86C8:20 05 CD DD 86 18 0E FE:94	8848:96 C3 07 8D 7D E6 3F 5F:BB
8558:7D E6 3F CA C8 85 2B DD:26	86D0:07 CC DD 86 01 0F 0D DE:EB	8850:7C E6 07 87 87 87 07:74
8560:75 01 DD 74 02 C9 CD CD:86	86D8:09 C1 19 DA C9 DD 66 0E:8E	8858:87 57 7C E6 38 0F 9F 0F:74
8568:85 DD 6E 91 DD 66 02 7D:25	86E0:DD 66 02 CD 4C 88 2A E2:2B	8860:82 57 7D E6 C0 0F 9F 0F:22
8570:E6 3F 28 54 2B DD 7E 0A:E8	86E8:8E 7D C6 02 4F 7B B9 D0:4C	8868:82 57 C9 DD 21 5E 96 06:13
8578:3C E6 CD DD 77 0A 20 05:17	86F0:7B DD 86 05 4F 7D B9 D0:59	8870:1D C5 CD C3 8E E6 55 DD:F7
8580:CD CB 8C 18 07 FE 03 28:D8	86F8:7C C6 04 4F 7C B9 D0:96	8878:77 03 01 06 00 DD 09 C1:0D
8588:E7 CD DB 8C DD 75 01 DD:5D	8700:DD 86 04 4F 7C B9 D0:18	8880:10 EF C9 DD 21 5E 96 06:E8
8590:74 02 C9 CD CD 85 DD 6E:B6	8708:22 E6 8E CD FA 87 01 FF:66	8888:1D C5 DD 6E 04 DD 66 05:3A
8598:01 DD 66 02 7D E6 3F 28:E0	8710:FF DD E5 CD 97 87 DD E1:63	8890:3A C0 8F 07 07 EE 80 B4:F8
85A0:27 2B DD 75 01 DD 74 02:7C	8718:01 FE FF CD 97 87 CD B7:3D	8898:67 36 00 01 06 00 DD 09:68
85A8:DD 6E 03 DD 66 04 01 20:80	8720:89 2A E4 8E CD DB 8C CD:C7	88A0:0C 10 E6 C9 DD 21 5E 96:38
85B0:00 09 DD 7E 0A 3C FE 06:92	8728:DB 8C CD DB 8C CD DB 8C:5C	88A8:06 1D DD 6E 00 DD 66 94:94
85B8:00 04 AF 21 6F 23 DD 75:A6	8730:CD A0 87 DD 2A E8 8E DD:D9	88B0:01 CD 75 04 08 74 25 DD:87
85C0:03 DD 74 04 DD 77 0A C9:17	8738:46 0D DD 2A E6 8E DD 7E:29	88B8:7E 02 DD E6 4A 03 FE 02 28:C2
85C8:AF DD 77 00 C9 3A F5 8E:87	8740:0C 4F B8 38 03 98 18 02:D0	88C0:11 7E 02 AA 20 07 CB 10:7A
85D0:7B C0 DD 6E 01 DD 66 02:E8	8748:3E 00 DD 77 0C B7 20 04:06	88C8:DD 70 03 18 1D CB 18 DD:59
85D8:22 E4 8E CD 4C 88 DD 53:E4	8750:AF DD 77 00 DD 2A E8 8E:BA	88D0:70 03 7D E6 3F 28 09 2B:F9
85E0:E2 8E 2A D4 8E CD 4C 88:19	8758:DD 7E 9C 99 DD 77 0C B7:67	88D8:DD 75 00 DD 74 01 18 0A:DD
85E8:2A E2 8E 7D DD 86 05 4F:09	8760:C9 DD 77 00 C9 DD 6E 0E:2A	88E0:7D F0 3F 6F DD 75 00 DD:41
85F0:7B B9 D0 7D C6 06 07 4F 7D:D7	8768:DD 66 02 E5 01 FF 0F 09:72	88E8:74 01 3A 03 8F 07 EE 3E:36
85F8:B9 D0 7C DD 86 06 4F 7A:79	8770:E5 CD B7 89 E1 CD A8 87:8A	88F0:00 B4 67 70 01 06 00 DD:66
8600:09 D0 7A C6 07 4F 7C B9:8E	8778:CD B7 89 E1 E5 01 FE FF:C2	88F8:09 C1 10 AE C9 06 64 DD:A8
8608:DD 3E 01 32 F6 8E C9 3A:16	8780:09 CD A0 87 CD DB 8C CD:DB	8900:21 07 8E C5 DD 7E 09 B7:85
8610:F4 8E B7 28 4B 3D F4:4E	8788:CD DB 8C CD DB 8C CD DB:6C	8908:28 25 DD 7E 00 B7 20 04:DC
8618:8E 21 0C 9C 3E 02 1E 01:06	8790:8C CD DB 8C C3 A0 87 C5:66	8910:AF DD 77 09 DD 6E 07 DD:89
8620:CD AE 8D 21 0C 9C 06 07:8D	8798:CD B7 89 E1 2A E4 8E 09:4D	8918:66 08 DD 4E 05 DD 64 06:47
8628:8D 3E 28 32 F5 8E 2A D4:DA	87A0:DD 75 01 DD 74 02 DD 75:68	8920:CD C3 82 DD 6E 01 DD 75:D5
8630:9E 05 E5 E5 E5 E5 CD 6B:0B	87A8:07 DD 74 08 11 5F 2A DD:DF	8928:07 DD 66 02 DD 74 08 01:F5
8638:87 E1 23 CD CB 8C CD CB:D6	87B0:73 03 DD 72 04 01 02 06:67	8930:0F 00 DD 09 C1 10 CC C9:84
8640:8C CD 68 87 E1 23 23 CD:DD	87B8:DD 71 05 DD 70 06 3E 04:A7	8938:06 64 DD 21 07 8F C5 DD:72

DEATHS TICKET

8940:7E 00 B7 28 3F FE 01 20:42
 8946:05 CD D3 8A 18 36 FE 02:CF
 8950:20 05 CD 64 8C 18 2D FE:71
 8956:03 20 05 CD 93 85 18 24:2C
 8960:FE 04 20 05 CD 87 18:AF
 8966:1B FE 05 20 05 CD 66 85:70
 8972:18 12 FE 06 20 05 CD 2F:18
 8978:85 18 09 FE 07 20 05 CD:37
 8980:CB 84 18 09 01 0F DD:60
 8986:09 C1 10 B2 06 64 DD 21:1D
 8989:07 8F 8C 5D DD 7E 09 B7 28:B8
 8996:15 DD 06 11 DD 66 02 DD:E9
 8998:5E 03 DD 56 04 DD 4E 05:98
 8998:DD DD 46 06 CD 71 82 01 0F:AF
 8999:04 DD 83 01 10 DC 09 06:99
 89B5:54 DD 21 07 8F DD 09:91
 89C0:BF C8 11 9F 00 DD 19 19:5C
 89C8:F4 C9 38 32 13 96 32:67
 89D0:14 F9 11 11 96 C3 07 8D:59
 89D8:3A F9 8E FE 04 38 02 C6:D1
 89E0:FE FD 5F 47 F4 F7 8E B7 28:49
 89E8:ED 3D 32 F7 8E 10 F4 21:18
 89F0:11 96 3E 02 CD AE 8D 21:70
 89F8:11 96 07 07 8D CD 8E:B5
 8A00:3A F9 8E FE 01 CA 21 8A:50
 8A06:FE 02 CA 25 8A FE 03 CA:91
 8A10:2C 8A FE 04 CA 21 8A FE:52
 8A18:95 CA 25 8A FE 06 CA 2C:D2
 8A20:8A CD 36 C6 C9 CD 45 8A:8E
 8A28:CD 54 8A C9 CD 36 8A CD:5C
 8A30:45 8A CD 54 8A C9 2A D4:F8
 8A38:8E 01 82 00 C5 01 06 09:61
 8A40:09 E5 C3 81 8A 2A D4 8E:D4
 8A48:01 02 02 C5 01 06 09:62
 8A50:E5 C3 81 8A 2A D4 8E:1A
 8A58:02 82 C5 01 06 09 09:E5
 8A60:C3 81 8A 2A D4 8E 01 02:84
 8A68:04 C5 01 06 09 09 E5 C3:2C
 8A70:81 8A 2A D4 8E 01 02 84:57
 8A78:C5 01 06 09 09 E5 C3 81:03
 8A80:8A CD 07 89 78 07 20:11
 8A88:EI C1 C9 E1 CD DB 8C DD:33
 8A90:75 01 DD 74 02 DD 15 07:9E
 8A98:DD 74 08 3A F9 8E FE 04:2A
 8A98:3E 08 11 3B 20 01 02 04:69
 8A98:3E 08 11 3B 20 01 3F:3F
 8A98:02 03 CD 08 11 3B 20:01
 8A98:72 04 DD 71 05 DD 79 06:9E
 8AC9:DD 77 0C 3E 11 0A DD:62
 8AC8:70 0B 01 DD 77 0D DD:99
 8AD0:77 09 C9 DD 6E 01 DD 66:19
 8AD8:02 7D E6 3F FE 3C 30 4E:2A
 8AE0:7D E6 C9 FE 2E 06 05 7C:6E
 8AEB:E6 07 28 42 7D E6 C9 FE:EB
 8AF0:80 20 07 7C E6 07 FE 04:F9
 8AF8:28 34 DD 71 0A 85 6F DD:E2
 8B00:7E 0B 47 B7 28 0E CB 78:39
 8B08:28 07 CB 8C DD 22 8B 18:64
 8B10:83 CD 28 08 DD 75 01 DD:DA
 8B18:74 02 DD E5 CD 96 86 DD:5D
 8B20:EI C1 CD DB 8C 10 FB C9:32
 8B28:07 CD C9 8C BB C9 9C E6:3B
 8B30:77 09 C9 3A 05 8F C3 E6:3B
 8B38:07 32 05 8F FE 01 20 46:85
 8B40:2A 8D 8E 01 04 09 22:B6
 8B48:D8 8E 5E 23 56 CB 7A CB:8B
 8B50:BA D5 D5 06 00 CA 26 8C:08
 8B58:EI D1 C8 72 CB D2 D5 E5:AD
 8B60:CA 54 83 EI D1 23 ED 53:5E
 8B68:DA 8E ED 53 DE 8E 5E 23:50
 8B70:56 CB 7A CB BA D5 E5 06:94
 8B78:01 ED 24 8C EI D1 ED 53:8C
 8B80:DC 8E 5D 53 E0 8E 3A 05:88
 8B88:8F 3D E6 07 FE 07 ED 5B:5D
 8B90:DA 28 09 2A ED 5B DE 8E:DE
 8B98:2A DD 8E 22 DE 58 5F 8E:54
 8BA0:7A CE 09 57 21 FF 78 15:54
 8BA8:03 8F 07 07 EE 80 BA 67:4F
 8BB0:BE CD FE 8B 3A 05 8F 3D:EE
 8BB8:E6 07 07 ED 5B DC 8E:84
 8BC0:20 0A ED 5B 0E 8E 7A DC:6C
 8BC8:8E 22 E0 8E 83 5F 2A CE:8C
 8BD0:00 07 21 BF 44 3A 03 8F:AS
 8BD8:07 57 EE 80 B4 67 EB CD:05

8BE0:83 8B C9 3E 08 ED A0 1B:A5
 8BE8:23 23 23 23 23 23 14:ET
 8BF9:14 14 14 14 14 14 14:93
 8BF8:20 EB C9 3E 08 ED A0 1B:A5
 8C00:23 23 23 23 23 23 15:08
 8C08:15 15 15 15 15 15 37:8C
 8C10:20 EB C9 3E 63 32 37 8E:31
 8C18:3E 39 32 13 96 32 14 96:8C
 8C20:21 11 96 C3 07 8D C5 CD:09
 8C28:87 89 78 C1 B7 C0 DD 70:B7
 8C30:0E 3E 3E DD 77 08 3E 01:92
 8C38:DD 77 09 21 3F 42 DD 75:AS
 8C40:01 DD 74 02 DD 75 07 DD:13
 8C48:74 08 B7 B7 11 43 20 28:A3
 8C50:93 11 85 20 DD 73 83 DD:22
 8C58:07 01 06 00 DD 71 09:32
 8C60:DD 70 84 C9 DD 6E 01 DD:B3
 8C68:66 02 7D E6 3F 28 CB:94
 8C70:DD 75 01 DD 74 02 CD 4C:85
 8C78:88 D5 2A D4 8E CD 4C 88:EC
 8C80:EI 7D C6 07 47 7B B9 D0:7C
 8C88:7B C6 04 4F 7D B9 D0:7C
 8C90:CE 07 4F 7A B9 D0 7A C6:C8
 8C98:0C 4F 7C B9 D0 DD 7E 0E:46
 8CA0:BF 28 05 CD 13 8C 03 DD:DD
 8CA8:CD 76 8D AF DD 77 08 C9:B3
 8CB0:21 00 40 ED FE 3F 11 01:10
 8CB8:40 36 09 01 B0 21 00 C0:DA
 8CC0:91 FE 3F 11 01 C9 36 09:49
 8CC8:ED B0 C9 7C C6 F8 67 E6:99
 8CD0:48 C9 7D C6 C6 08 67 C6:CD
 8CD8:3F 67 04 C9 C6 08 67 E6:CD
 8CE0:30 C8 7D C6 40 6F 7C CE:61
 8CE8:C0 67 C9 E5 23 23 0E 00:D4
 8CF0:7E B7 28 04 0C 23 18 F8:66
 8CF8:79 B7 EI C8 5E 26 53:48
 8D00:3E 20 91 B3 5F 18 04 5E:20
 8D08:23 56 23 7E B7 C8 FE 01:02
 8D10:23 CA 07 8D 2B D5 D5:D6
 8D18:20 6F 26 00 29 29 29:3F
 8D20:01 00 10 09 CB FA D5 D5:DD
 8D28:CD 39 8D E1 D1 CB 18 18:0A
 8D30:39 8D E1 23 D1 13 13:70
 8D38:2B 3E 08 ED A0 ED A0 1B:C5
 8D40:1B 14 14 14 14 14 14:A4
 8D48:14 3D 20 EF C9 21 95 96:00
 8D50:09 05 09 11 06 96 3E 30:00
 8D58:02 7E 5D 8E 32 42 42:8E
 8D60:32 14 96 3E 39 32 94 96:75
 8D68:32 04 96 3E 33 32 37 8E:55
 8D70:3E 03 32 F4 8E C9 3A F9:81
 8D78:8E FE 8C 8C 3C 32 F9 8E:58
 8D80:FE 01 21 16 96 CA 07 8D:0B
 8D88:FE 02 21 16 96 CA 07 8D:39
 8D90:FE 03 21 28 96 CA 07 8D:67
 8D98:FE 04 21 31 96 CA 07 8D:95
 8DA0:FE 05 21 40 96 CA 07 8D:DB
 8DA8:21 4F 96 C3 07 8D 4F 06:8C
 8DB0:00 09 23 23 41 E5 2B 7E:04
 8DB8:3D 77 FE 2F 20 05 3E 39:5E
 8DC0:77 10 F3 D1 E1 2B ED C9:19
 8DC8:00 09 23 41 E5 2B 7E:04
 8DD0:2B 7E 3C 77 FE 8E 4E 6E:AE
 8DD8:3E 30 77 10 F3 D1 E1 2B:3C
 8DE0:ED C9 F3 21 47 8E 22 39:86
 8DE8:00 01 01 BC ED 49 01 20:C8
 8DF0:BF ED 49 01 02 BC ED 49:28
 8DF8:01 2B DD ED 49 01 06:CA
 8E00:ED 49 01 19 BD ED 49 01:BA
 8E08:07 BC ED 49 01 1D BD ED:46
 8E10:49 01 0D BC ED 49 01 00:5E
 8E18:BD ED 49 01 0C BC ED 49:83
 8E20:01 30 BD ED 49 06 7E 0E:74
 8E28:1B ED 49 0E 54 ED 49 21:2E
 8E30:BD 8E 9E 00 16 10 ED 49:50
 8E38:7E 23 F6 40 ED 9C 15:DF
 8E40:00 00 00 00 00 00 00:00
 8E48:21 B1 8E F5 E6 F8 0F 0F:06
 8E50:0F 5F 16 00 19 F1 E5 21:5B
 8E58:62 8E E6 07 5F 19 7E EI:1D
 8E60:A6 C9 01 02 04 08 10 20:E5
 8E68:40 80 01 01 8E 01 01:FA
 8E70:ED 49 06 F6 ED 78 E6 30:A2
 8E78:4F C9 ED 79 ED 49 04:3F

8E80:3E 92 ED 79 C5 CB F1 06:7D
 8E88:F6 ED 49 06 F4 ED 78 2F:FE
 8E90:77 23 C9 79 E6 0F FE 0A:FD
 8E95:20 ED C1 3E 82 ED 79 05:EA
 8EA0:ED 49 C9 06 08 21 CB 8E:0B
 8EA8:4E 7E 81 77 4F 23 10 F9:74
 8EAC:C9 00 00 00 00 00 00:07
 8EB0:00 00 00 14 0B 0A 0E 0C:CB
 8EC0:1C 04 15 1F 13 16 12 0E:92
 8EC8:00 14 18 7B 2A 08 4A F3:52
 8ED0:0D 59 47 20 00 00 00:02
 94E0:00 00 00 00 C2 43 4F 4E:69
 94E8:47 52 41 54 55 4C 41 54:52
 94F0:49 4F 4E 20 21 00 00 C3:92
 94F8:5A 45 47 47 4E 20 45 53:5E
 9500:20 4E 45 57 20 46 52 45:44
 9508:45 00 00 C0 44 45 41 54:3B
 9510:148 07 53 20 54 49 43 4B:3B
 9518:45 54 00 00 C1 31 20 2D:CD
 9520:20 53 54 41 52 54 20 57:A5
 9528:49 54 48 20 4A 4F 59 53:59
 9530:54 49 43 4B 00 00 C2 32:7E
 9538:20 2D 20 53 54 41 52 54:FA
 9540:20 57 49 54 48 20 4B 45:2B
 9548:59 42 4E 41 52 44 00 00:DD
 9550:04 44 45 41 54 48 27 53:D1
 9558:20 54 49 43 4B 45 54 20:FD
 9560:43 59 59 52 49 47 48:AB
 9568:54 00 4A 52 4E 47 2E:68
 9570:53 20 28 53 48 46 54 20:DB
 9578:31 39 39 30 00 00 C4 50:F7
 9580:52 4F 47 52 41 4D 20 41:1D
 9588:4E 44 20 47 52 41 50 48:FF
 9590:49 43 53 20 42 59 00 C0:CD
 9598:04 52 4F 59 45 52 20 53:A3
 95A0:45 42 41 53 54 49 45 4E:BA
 95A8:00 00 C5 45 53 50 45 43:1A
 95B0:49 41 4C 59 20 46 4F 52:55
 95B8:20 4D 49 43 52 4F 20 2D:AA
 95C0:20 4D 41 47 00 80 42 47:F4
 95C8:20 41 20 4D 20 45 20 20:B1
 95D0:4F 20 56 45 20 52 00 CD:CD
 95D8:C0 44 53 43 4F 52 45 3A:E4
 95E0:20 30 30 30 30 30 30:AS
 95E8:0C 49 45 47 49 3F 30 30:00
 95F0:20 45 47 49 3F 30 30:00
 95F8:39 39 00 15 44 59 50:0C
 9600:45 3A 0E C5 44 30 30 CB:CB
 9608:30 30 30 01 E8 44 30 33:CA
 9610:01 FC 44 39 00 21 45 7B:04
 9618:53 49 4D 50 4C 45 00 21:DC
 9620:45 44 45 52 42 4C 45 00:2B
 9628:21 45 54 52 49 50 45 45:36
 9630:00 21 45 53 55 50 45 52:1F
 9638:20 53 49 4D 50 4C 45 00:DE
 9640:21 45 53 55 50 45 52 20:3A
 9648:44 45 53 45 42 4C 45 00 21:E9
 9650:45 53 55 50 45 52 20 54:D5
 9658:52 49 50 4C 45 00 6E 42:5D
 9660:02 41 45 42 9E 42 02 41:84
 9668:9E 42 0E 4F 41 0F 44:74
 9670:3E 41 02 41 3E 41 22 43:92
 9678:01 54 22 43 47 42 02 41:2E
 9680:47 42 44 43 02 41 44 43:FD
 9688:69 43 01 54 69 43 17 44:C0
 9690:02 41 17 44 FC 42 01 54:1E
 9698:FC 42 A0 43 01 54 A0 43:0F
 96A0:C9 42 02 41 C9 42 55 41:61
 96A8:01 54 55 41 61 42 01 54:92
 96B0:81 42 58 43 02 41 58 43:9F
 96B8:38 41 21 54 38 41 68 43:E9
 96C0:01 54 68 43 18 42 02 41:6C
 96C8:1B 42 DB 43 02 41 DB 43:3F
 96D0:45 42 02 41 45 42 16 44:DB
 96D8:02 41 16 44 07 41 0F 44:74
 96E0:57 41 16 44 01 54 01 49:94
 96E8:6B 42 02 41 6B 42 48 44 32:32
 96F0:01 54 48 41 60 44 41 54:36
 96F8:60 42 42 43 02 41 42 43:BE
 9700:68 42 02 41 68 42 54 42:7D
 9708:02 41 54 42 39 43 02 41:EC
 9710:39 43 86 41 01 54 86 41:8B


```

13,24,18,23,26,6
450 INK 7,7,18:tr$=CHR$(22)+CHR$( [3976]
1):nr$=CHR$(22)+CHR$(0)
460 DEF FN lt(x,y)=%C00+(y-1)*80 [1711]
+(x-1)*2
470 ali3$=CHR$(209)+CHR$(210)+CHR [2257]
$(211)+CHR$(212)
480 ali2$=CHR$(205)+CHR$(206)+CHR [2118]
$(207)+CHR$(208)
490 ali1$=CHR$(201)+CHR$(202)+CHR [3744]
$(203)+CHR$(204)
500 DIM sp(20):FOR H=0 TO 15:SP(H [4722]
+)=&9700+(H*64):NEXT:SP(20)=&A40
0
510 DIM JE(16,13):WINDOW #2,17,19 [2387]
,18,23
520 SC=0:POKE &A039,200:TTT=210:T [1467]
A=1
530 ON BREAK GOSUB 2670:REM ligne [2858]
a n'entrer que quand prog au poi
nt
540 GOTO 720 [421]
550 REM : [1736]
560 REM : [419]
570 REM : DECOR DE BASE : [1019]
580 REM : [419]
590 REM : [1736]
600 FOR H=0 TO 30:CALL &A032,&C00 [3150]
0+H:CALL &A032,&C03B-H:NEXT
610 DEF FN po(x,y)=%C00+(y-1)*24 [2654]
+(x-1)*6:POKE &A01A,24:POKE &A01
E,6
620 RESTORE 660:FOR g=1 TO 4:READ [3312]
a$:FOR h=1 TO 10
630 b$=MIDS(a$,h,1):a=VAL(b$):IF [2963]
a=0 THEN 650
640 CALL &A00D,FN po(h,g),&9BA0 [1829]
650 NEXT h,g [382]
660 DATA 111100011111,11100011111 [2523]
1,110000000011,100000000010
670 RESTORE 710:FOR g=5 TO 8:READ [2310]
a$:FOR h=1 TO 10
680 b$=MIDS(a$,h,1):a=VAL(b$):IF [30216]
a=0 THEN 700
690 CALL &A00D,FN po(h,g)+80,&9BA [2390]
0
700 NEXT h,g:POKE &A01A,16:POKE & [1180]
A01E,4:RETURN
710 DATA 0000001000,0000111000,00 [2737]
01111000,0011111000
720 PLOT -10,-10,11:TAG:MOVE 498, [5078]
368:PRINT ali1$:MOVE 498,352:PRI
NT ali2$:MOVE 498,336:PRINT ali3
$:TAGOFF
730 DATA 16,1,5,16,5,5,16,7,5,16, [3041]
9,5,16,12,5,16,14,5,16,17,5,16,24
,5
740 DATA 16,1,5,20,1,5,16,7,3,20, [3716]
7,3,16,12,3,20,12,3,16,17,8,20,17
,8
750 e1=12:e2=9:RESTORE 730:FOR i= [5745]
1 TO 16:GOSUB 760:NEXT:GOTO 840
760 READ X,Y,L:x1=14+(X-1)*32:y1= [2428]
388-(Y-1)*16
770 IF i>8 THEN 810 [1007]
780 FOR J=0 TO 2 STEP 2:PLOT x1,y
1+j,e1:DRAW x1+(1-1)*32,y1+j:NEXT
790 FOR J=4 TO 6 STEP 2:PLOT x1,y [4175]
1+j,e2:DRAW x1+(1-1)*32,y1+j:NEXT
800 RETURN [555]
810 PLOT x1+4,y1+2,e1:DRAW x1+4,y [2465]
1+2-(1-1)*16
820 PLOT x1,y1+2,e2:DRAW x1,y1+2- [1096]
(1-1)*16
830 RETURN [555]
840 LOCATE 1,1:PRINT tr$:RESTORE [4834]
850:FOR h=1 TO 16:GOSUB 870:NEXT:
GOTO 890
850 DATA 16,1,16,5,20,1,20,5,16,7 [3113]
,20,7,16,9,20,9,16,12,20,12
860 DATA 16,14,20,14,16,17,20,17, [1479]
16,24,20,24
870 READ x,y:PEN 12:LOCATE x,y:PR [3520]
INT CHR$(170):PEN 9:LOCATE x,y
880 PRINT CHR$(171):RETURN [1611]
890 RESTORE 900:FOR h=1 TO 5:READ [3477]
ZL,YL,PL,a$:GOSUB 910:NEXT:GOTO
940
900 DATA 32,8,0,"000000",32,10,1, [3729]
"SCORE",32,13,0,"000000",32,15,1,
"TEMPS",31,25,0,"CHALLENGE"
910 AS=UPPER$(A$):FOR T=1 TO LEN( [2706]
A$):SP=(ASC(MIDS(A$,T,1)))-48
920 IF sp<0 THEN sp=43 [1071]
930 CALL &9D00,FN LT(ZL,YL)+(T*2) [4127]
+PL,&9D20+(SP*16):NEXT:RETURN
940 A$=" ALI":FOR T=1 TO LEN(A$ [3598]
):SP=(ASC(MIDS(A$,T,1)))-48
950 IF sp<0 THEN sp=43 [1071]
960 CALL &9D00,&C1CB+(T*2),&9D20+ [2869]
(SP*16):NEXT:GOSUB 3490
970 REM : [1736]
980 REM : [419]
990 REM : DESSIN TABLEAU : [1952]
1000 REM : [419]
1010 REM : [1736]
1020 GOSUB 600:ON TA GOTO 1030,10 [5084]
40,1050,1060,1070,1080,1090,1100,
1110,1120,1130,1140,1150
1030 RESTORE 2730:GOSUB 1170:GOTO [2417]
1410
1040 RESTORE 2790:GOSUB 1170:GOTO [2779]
1410
1050 RESTORE 2850:GOSUB 1170:GOTO [1923]
1410
1060 RESTORE 2910:GOSUB 1170:GOTO [2767]
1410
1070 RESTORE 2970:GOSUB 1170:GOTO [1923]
1410
1080 RESTORE 3030:GOSUB 1170:GOTO [2063]
1410
1090 RESTORE 3090:GOSUB 1170:GOTO [2672]
1410
1100 RESTORE 3150:GOSUB 1170:GOTO [2063]
1410
1110 RESTORE 3210:GOSUB 1170:GOTO [1666]
1410
1120 RESTORE 3270:GOSUB 1170:GOTO [2063]
1410
1130 RESTORE 3330:GOSUB 1170:GOTO [1229]
1410
1140 RESTORE 3390:GOSUB 1170:GOTO [2005]

```



```

1410
1150 TA=1:GOTO 1020 [743]
1160 REM : : : DESSIN TABLEAU : : : [1859]
1170 POKE &A020,0:DEF FN po(x,y)= [1592]
&E000+(y-1)*160+(x-1)*4
1180 ERASE JE:DIM JE(16,13):FOR H [3702]
=1 TO 16:JE(H,1)=20:JE(H,13)=20:N
EXT
1190 FOR H=2 TO 12:JE(1,H)=20:JE( [2212]
16,H)=20:NEXT:
1200 FOR g=1 TO 12:READ a$:FOR h= [2105]
1 TO 15
1210 b$=MID$(a$,h,1):IF B$="X" TH [4255]
EN JE(H,G)=20:GOTO 1270
1220 IF B$="G" THEN A=16:GOTO 124 [1216]
0
1230 a=VAL("&"+B$):IF A=0 THEN 12 [2065]
60
1240 IF A=7 OR A=12 THEN mx=h:my= [921]
g
1250 CALL &A00D, FN PO(H,G), SP(A): [2548]
JE(H,G)=A:GOTO 1270
1260 CALL &A00D, FN PO(H,G), &9C50: [2282]
JE(H,G)=0
1270 NEXT H,G [382]
1280 READ X,Y:POKE &A020, &AE:CALL [2858]
&A00D, FN PO(X,Y), SP(5)
1290 DEF FN po(x,y)=&C000+(y-1)*8 [2532]
+(x-1)*2:POKE &A01A,8:POKE &A01E
,2
1300 CLS #2:FOR G=19 TO 23:READ A [2199]
$:FOR H=1 TO 5
1310 b$=MID$(a$,h,1):a=VAL("&"+b$ [2900]
):IF A=0 THEN 1330
1320 CALL &A00D, FN PO(H+33,G)-1, & [2640]
9AF0+(A*16)
1330 NEXT H,G:POKE &A01A,16:POKE [1020]
&A01E,4
1340 LOCATE 1,1:PRINT NR$:TTT=TTT [1492]
-10:tien=0
1350 fin=0:GAN=0:AF=5:READ tts,MU [3432]
L:REC=TTT:GOSUB 2410:RETURN
1360 REM : : : : : [1736]
1370 REM : : : : : [419]
1380 REM : : : ROUTINE PRINCIPALE : [2225]
1390 REM : : : : : [419]
1400 REM : : : : : [1736]
1410 POKE &A020, &AE:DEF FN po(x,y [2120]
)=&E000+(y-1)*160+(x-1)*4
1420 EVERY 250,0 GOSUB 2470 [492]
1430 SON=0:IF FIN=1 THEN 2540 [1869]
1440 IF INKEY(0)*INKEY(72)=0 THEN [2029]
1510
1450 IF INKEY(2)*INKEY(73)=0 THEN [943]
1640
1460 IF INKEY(8)*INKEY(74)=0 THEN [1330]
1770
1470 IF INKEY(1)*INKEY(75)=0 THEN [838]
1890
1480 IF INKEY(9)*INKEY(76)=0 THEN [1257]
2050
1490 GOTO 1430 [341]
1500 REM : : : VERS LE HAUT : : : [881]
1510 IF je(x,y-1)>19 THEN 1430 [594]
1520 IF TIEN=1 THEN 1580 [1413]
1530 GOSUB 1540:y=y-1:GOTO 1430 [1985]
1540 CALL &A00D, FN po(x,y), sp(AF) [2057]
:CALL &A052, FN po(x,y)
1550 FOR I=1 TO 2:CALL &A05C, sp(A [1279]
F)
1560 FOR t=1 TO 20:NEXT t:CALL &A [2294]
06E, sp(AF):NEXT I
1570 CALL &A05C, sp(AF):RETURN [2223]
1580 IF je(x,y-1)<>0 THEN 1430 [820]
1590 JE(X,Y)=0:GOSUB 1540:Y=Y-1:J [3354]
E(X,Y)=ACT
1600 SON=SON+1:SOUND 1,40+(SON*10 [2367]
),3,5,,,6
1610 IF act=7*MUL THEN mx=x:my=y: [4667]
GOTO 1510 ELSE 1510
1620 REM : : : VERS LE BAS : : : [1275]
1630 GOTO 1510 [397]
1640 IF je(x,y+1)>19 THEN 1430 [788]
1650 IF TIEN=1 THEN 1720 [1417]
1660 GOSUB 1670:y=y+1:GOTO 1430 [1321]
1670 CALL &A00D, FN po(x,y), sp(AF) [2057]
:CALL &A052, FN po(x,y)
1680 FOR I=1 TO 3:CALL &A07B, sp(A [1476]
F)
1690 FOR t=1 TO 20:NEXT t:CALL &A [2294]
06E, sp(AF):NEXT I
1700 CALL &A07B, sp(AF):RETURN [1288]
1710 REM : : : AVANCE bas [1603]
1720 IF je(x,y+1)<>0 THEN 1430 [559]
1730 JE(X,Y)=0:GOSUB 1670:Y=Y+1:J [1182]
E(X,Y)=ACT
1740 SON=SON+1:SOUND 1,40+(SON*10 [2367]
),3,5,,,6
1750 IF act=7*MUL THEN mx=x:my=y: [4045]
GOTO 1640 ELSE 1640
1760 REM : : : A GAUCHE : : : [1052]
1770 IF je(x-1,y)>19 THEN 1430 [598]
1780 IF TIEN=1 THEN 1840 [1409]
1790 GOSUB 1800:x=x-1:GOTO 1430 [2098]
1800 CALL &A00D, FN po(x,y), sp(AF) [2596]
: add=FN po(x,y)
1810 FOR I=1 TO 3:CALL &A00D, add- [1631]
I, sp(AF)
1820 FOR t=1 TO 20:NEXT t:CALL &A [3227]
00D, add-I, sp(AF):NEXT I
1830 CALL &A00D, add-4, sp(AF):RETU [1800]
RN
1840 IF je(x-1,y)<>0 THEN 1430 [818]
1850 JE(X,Y)=0:GOSUB 1800:X=X-1:J [1853]
E(X,Y)=ACT
1860 SON=SON+1:SOUND 1,40+(SON*10 [2367]
),3,5,,,6
1870 IF act=7*MUL THEN mx=x:my=y: [4182]
GOTO 1770 ELSE 1770
1880 REM : : : A DROITE : : : [1375]
1890 IF je(x+1,y)>19 THEN 1430 [1206]
1900 IF TIEN=1 THEN 1960 [1417]
1910 GOSUB 1920:x=x+1:GOTO 1430 [1801]
1920 CALL &A00D, FN po(x,y), sp(AF) [2596]
: add=FN po(x,y)
1930 FOR I=1 TO 3:CALL &A00D, add+ [1370]
I, sp(AF)
1940 FOR t=1 TO 20:NEXT t:CALL &A [2536]
00D, add+I, sp(AF):NEXT I
1950 CALL &A00D, add+4, sp(AF):RETU [3125]
RN
1960 IF je(x+1,y)<>0 THEN 1430 [942]

```

PROGRAMMATION

```

1970 JE(X,Y)=0:GOSUB 1920:X=X+1:J [2639]
E(X,Y)=ACT
1980 SON=SON+1:SOUND 1,40+(SON*10 [2367]
),3,5,...,6
1990 IF act=7*MUL THEN mx=x:my=y: [3801]
GOTO 1890 ELSE 1890
2000 REM :::::::::::::::::::::::::::: [1736]
2010 REM : [419]
2020 REM : PRISE / LACHER : [1603]
2030 REM : [419]
2040 REM :::::::::::::::::::::::::::: [1736]
2050 IF JE(X,Y)<7 OR JE(X,Y)>17 T [1850]
HEN 1430
2060 IF TIEN=1 THEN 2130 [1415]
2070 ENT 3,100,-50,30:SOUND 1,600 [1406]
,10,6,0,3
2080 CALL &A00D,FN PO(X,Y),SP(AF) [3068]
:CALL &A00D,FN PO(X,Y),SP(6)
2090 CALL &A090,FN PO(X,Y):TIEN=1 [2454]
2100 ACT=JE(X,Y):WHILE INKEY$<" " [1419]
:WEND
2110 FOR T=1 TO 500:NEXT:AF=20:GO [2413]
TO 1430
2120 REM :: LACHER :: [317]
2130 ENT 3,100,-50,30:SOUND 1,600 [1406]
,10,6,0,3
2140 CALL &A00D,FN PO(X,Y),SP(6): [2675]
CALL &A00D,FN PO(X,Y),SP(5)
2150 TIEN=0:WHILE INKEY$<" " :WEND [1159]
2160 GOSUB 2220:AF=5:GOTO 1430 [1518]
2170 REM :::::::::::::::::::::::::::: [1736]
2180 REM : [419]
2190 REM : MOLECULE CORRECTE ? : [1304]
2200 REM : [419]
2210 REM :::::::::::::::::::::::::::: [1736]
2220 DATA -1,0,-1,-1,0,-1,1,-1,1, [2082]
0,1,1,0,1,-1,1
2230 DATA -2,-1,-2,0,-2,1,2,-1,2, [2260]
0,2,1
2240 tout=0:RESTORE 2220:FOR h=1 [1933]
TO 14:READ px,py
2250 px=px+mx:py=py+my:sm=je(px,p [2501]
y)
2260 IF sm<7 OR SM>16 THEN tot=0 [3202]
ELSE tot=sm^h
2270 tout=tout+tot:NEXT:TOUT=TOUT [2594]
+JE(MX,MY)
2280 IF TOUT=TTS THEN FIN=1:GAN=1 [2058]
2290 RETURN [555]
2300 REM :::::::::::::::::::::::::::: [1736]
2310 REM : [419]
2320 REM : GESTION COMPTEURS : [2235]
2330 REM : [419]
2340 REM : [1736]
2350 IF sc=0 THEN a$="000000":ZL= [2425]
3:YL=8:GOSUB 910:RETURN
2360 a$=STR$(sc):a$=RIGHT$(a$,LEN [2868]
(a$)-1)
2370 IF sc<10 THEN ZL=36:YL=8:GOS [2766]
UB 910:RETURN
2380 IF sc<100 THEN ZL=35:YL=8:GO [3486]
SUB 910:RETURN
2390 IF sc<1000 THEN ZL=34:YL=8:G [2136]
OSUB 910:RETURN
2400 ZL=33:YL=8:GOSUB 910:RETURN [912]
2410 IF REC=0 THEN a$="000000":ZL [1740]
=32:YL=13:GOSUB 910:RETURN
2420 a$=STR$(REC):a$=RIGHT$(a$,LE [2806]
N(a$)-1)
2430 IF REC<10 THEN ZL=33:YL=13:a [2307]
a$="000"+a$:GOSUB 910:RETURN
2440 IF REC<100 THEN ZL=33:YL=13: [4483]
a$="00"+a$:GOSUB 910:RETURN
2450 IF REC<1000 THEN ZL=33:YL=13 [1853]
:a$="0"+a$:GOSUB 910:RETURN
2460 REM :::: TEMPO [448]
2470 DI:REC=REC-5:GOSUB 2410 [845]
2480 IF REC<5 THEN FIN=1:EI:RETUR [3633]
N ELSE EI:RETURN
2490 REM :::::::::::::::::::::::::::: [1736]
2500 REM : [419]
2510 REM : GAGNE/PERDU [1324]
2520 REM : [419]
2530 REM :::::::::::::::::::::::::::: [1736]
2540 FOR H=0 TO 3:MU=REMAIN(H):NE [2096]
XT
2550 IF gan=0 THEN 2620 [638]
2560 FOR H=1 TO 16:BORDER h:SOUND [3415]
1,h*15,10,14:FOR t=1 TO 200
2570 NEXT t,h:BORDER 0 [2440]
2580 ENV 10,15,-1,1:FOR H=1 TO IN [3773]
T(REC/5):sc=sc+1:GOSUB 2350
2590 SOUND 1,0,15,15,10,,15:NEXT: [3944]
REM CLAUDE TEL 96 38 94 24
2600 TA=TA+1:REC=0:GOSUB 2410:GOT [1738]
O 1020
2610 REM :::: PERDU :::: [1027]
2620 BORDER 26:INK 0,26:SOUND 4,1 [2203]
500,50,7,0,0,10
2630 FOR t=1 TO 4000:NEXT t:BORDE [2617]
R 0:INK 0,0
2640 SOUND 1,239,20,6:SOUND 1,0,2 [7574]
,6:SOUND 1,239,20,6:SOUND 1,319,2
0,6:SOUND 1,213,20,6:SOUND 1,239,
40,6:SOUND 1,319,20,6
2650 PRINT 11:FOR H=1 TO 5:LOCATE 1 [7462]
8,3:PRINT CHR$(214):FOR T=1 TO 40
0:NEXT T:LOCATE 18,3:PRINT CHR$(2
13):FOR T=1 TO 200:NEXT T,H
2660 FOR T=1 TO 2000:NEXT T:TA=TA [3646]
-1:ttt=ttt+10:GOTO 2600
2670 fin=1:gan=0:RETURN [960]
2680 REM :::::::::::::::::::::::::::: [1736]
2690 REM : [419]
2700 REM : DATA 12 TABLEAUX : [1172]
2710 REM : [419]
2720 REM :::::::::::::::::::::::::::: [1736]
2730 DATA XXXXXXXXXXXXXXXXX,XX11111 [4344]
XXXXXXXXXX,XX10001111111111111111
2740 DATA XX10920000001XXX,XX10200 [3031]
07001XXX,XX1020000033011XX
2750 DATA XX1100002030011XX,XX10000 [2720]
203001XX,XX113B30000011XX
2760 DATA XXX10000000011XX,XXX1111 [3916]
1111111XX,XXXXXXXXXXXXXXXXXX
2770 DATA 6,6,00000,05130,00000,0 [1803]
0000,00000,63,0
2780 REM :::: TAB 2 :::: [329]
2790 DATA XXXXXXXXXXXXXXXXX,X44444X [4198]
XXXXXXXXXX,X40004XXX44444X
2800 DATA X400B333330084X,X400200 [2591]
0000034X,X4020003700004X

```


PROGRAMMATION

```

2810 DATA X4000000000004X,X400000 [3479]
0030004X,X4900033333334X
2820 DATA X111111XXXXXXX,XXXXXXX [3880]
XXXXXXX,XXXXXXX
2830 DATA 8,6,0,0000,02200,05130,0 [1457]
0000,00000,87,0
2840 REM :: TAB 3 :: [623]
2850 DATA XXXXXXXXXXXXXXXX,X22222X [4496]
XXXXXXXX,X300B3XXXXXXXXXX
2860 DATA X300032222222X,X300030 [2619]
0071003X,X301030000103X
2870 DATA X300000000003X,X300000 [3048]
0100003X,X3000000A01803X
2880 DATA X301110000103X,X300910 [3037]
0222222X,X222222XXXXXX
2890 DATA 6,7,00000,02200,05130,0 [1996]
0400,00000,157,0
2900 REM :: TAB 4 :: [569]
2910 DATA XXXXXXXXXXXXXXXX,XXXXXXX [3521]
X323232X,X32323XX200003X
2920 DATA X20003XX300C02X,X30F023 [3574]
2300003X,X2000000000002X
2930 DATA X30003232300032,X204000 [2986]
00200003,X30G0000E300002
2940 DATA X23232323200032,XXXXXXX [3193]
XX323232,XXXXXXXXXXXXXX
2950 DATA 5,7,00000,0A070,00600,0 [1834]
0900,00000,300,5
2960 REM :: TAB 5 :: [545]
2970 DATA XXXXXXXXXXXXXXXX,X44444X [4652]
XXXXXXXXX,X40004XXXXXXX
2980 DATA X400C4444444XX,X400020 [2857]
0020444X,X4020000029004X
2990 DATA X4000000020004X,X400222 [3298]
0002040X,X4000000D00004X
3000 DATA X4000000200204X,X444444 [3001]
000B2F4X,XXXXXX4444444X
3010 DATA 5,7,00000,00073,00600,5 [2191]
9000,00000,413,5
3020 REM :: TAB 6 :: [679]
3030 DATA XXXXXXXXXXXXXXXX,XXX1212 [3415]
1212XXXX,XXX20000001XXXX
3040 DATA XXX1000B02XXXX,XXX2000 [2963]
33333333,XXX121000C4D002
3050 DATA X212B2000440001,X1F0010 [2603]
0000002,X20002003333333
3060 DATA X100000000E0C2,X2000G0 [2767]
00000001,X4444444444444
3070 DATA 7,7,00000,5A007,00660,5 [1860]
9008,00000,469,5
3080 REM :: TAB 7 :: [591]
3090 DATA XXXXXXXXXXXXXXXX,X111111 [3284]
111111XX,X17000090001XX
3100 DATA X100020220001XX,X102020 [1909]
000001XX,X100000001111X
3110 DATA X11110A2090B1X,X108000 [2853]
0202001X,X1102200000071X
3120 DATA X100000001111X,X170000 [3581]
02A1XXXX,X11111111XXXX
3130 DATA 6,3,00000,02200,51113,0 [2155]
4400,00000,330,0
3140 REM :: TAB 8 :: [664]
3150 DATA XXXXXXXXXXXXXXXX,X333333 [4336]
333333XX,X3000004F0033XX
3160 DATA X300400000C3XXX,X300404 [2589]
444033XX,X30G4040E4003XX
3170 DATA X304400004D03XX,X3C0000 [3476]
004003XX,X3004000000F3XX
3180 DATA X3004400000443XX,X300G40 [3369]
000003XX,X333333333333XX
3190 DATA 6,3,00000,0A07,00660,0 [1646]
9908,00000,729,5
3200 REM :: TAB 9 :: [640]
3210 DATA XXXXXXXXXXXXXXXX,X444444 [3453]
44444444,X4F030000000004
3220 DATA X4G030000000004,X4C0300 [1863]
0000004,X4C030000000004
3230 DATA X4703000000004,X480000 [2980]
0000004,X4000000000004
3240 DATA X4000000000004,X49A300 [2806]
00000004,X4444444444444
3250 DATA 6,3,00000,A0020,06613,9 [2370]
0040,00000,577,5
3260 REM :: TAB 10 :: [724]
3270 DATA XXXXXXXXXXXXXXXX,X333333 [3249]
33333333,XX400829002CB4X
3280 DATA XXX4000220004XX,XXX4A0 [3001]
02074XX,XXXXX400004XXXX
3290 DATA XXXXX4004XXXXX,XXXXXX4 [3184]
004XXXXX,XXXXX400004XXXX
3300 DATA XXXX40000A433XX,XXX4800 [2809]
200074XX,X333333333333X
3310 DATA 4,3,00000,02020,51613,0 [2060]
4040,00000,469,5
3320 REM :: TAB 11 :: [732]
3330 DATA XXXXXXXXXXXXXXXX,XXXXXXX [4752]
XXXXXXXXX,XXX344444XXXXX
3340 DATA XXXX1C77881XXXX,XXX1AA [2653]
9B01XXXX,XXXX1DEF001XXXX
3350 DATA XXXX101011XXXX,XXX122 [3178]
2221XXXX,XXX344444XXXXX
3360 DATA XXXXXXXXXXXXXXXX,XXXXXXX [4457]
XXXXXXXXX,XXXXXXXXXXXXXXXXX
3370 DATA 8,7,00000,2A072,13651,4 [1881]
9084,00000,943,5
3380 REM :: TAB 12 :: [772]
3390 DATA XXXXXXXXXXXXXXXX,X111111 [3263]
1111111X,X1200000000021X
3400 DATA X1000C77880001X,X1000AA [3023]
9B00001X,X10000DEF00001X
3410 DATA X1000000000001X,X10000 [2716]
0000021X,X100000000001X
3420 DATA X100000000001X,X120000 [3088]
0000021X,X111111111111X
3430 DATA 8,7,00000,2A072,13651,4 [1881]
9084,00000,943,5
3440 REM :::::::::::::::::::::::::::: [1736]
3450 REM :::::::::::::::::::::::::::: [419]
3460 REM PRESENTATION :::::::::::::: [1647]
3470 REM :::::::::::::::::::::::::::: [419]
3480 REM :::::::::::::::::::::::::::: [1736]
3490 a$=" LA MOLEULE":ZL= [2254]
1:YL=1:GOSUB 910
3500 a$=" ::::::::::::::::::::":ZL= [3302]
1:YL=2:GOSUB 910
3510 a$=" EN VOUS SERVANT DES AT [3118]
OMES":ZL=1:YL=4:GOSUB 910
3520 a$="DISSEMINES DANS CHAQUE E [3004]
CRAN":ZL=1:YL=6:GOSUB 910
3530 a$="A VOUS DE RECONSTITUER [4605]
LA":ZL=1:YL=8:GOSUB 910
3540 a$="MOLEULE DONT LE SCHEMA [1945]

```

PROGRAMMATION

```

VOUS":ZL=1:YL=10:GOSUB 910
3550 a$="EST FOURNI<":ZL=1:YL=12: [2555]
GOSUB 910
3560 a$=" POUR CELA VOUS UTILIS [1813]
EREZ":ZL=1:YL=14:GOSUB 910
3570 a$="LE TELEPORTEUR<":ZL=1:YL [3102]
=16:GOSUB 910
3580 a$=" LA FIN D>UN DEPLACEMENT [3257]
T NE":ZL=1:YL=18:GOSUB 910
3590 a$="POUVANT SE FAIRE QU>APRE [3799]
S LA":ZL=1:YL=20:GOSUB 910
3600 a$="RENCONTRE D>UN OBSTACLE< [3182]
":ZL=1:YL=22:GOSUB 910
3610 a$=" BONNE CHANCE":ZL [2665]
=1:YL=24:GOSUB 910
3620 WHILE INKEY$="" :WEND:MU=REMA [3116]
IN(1):RETURN
3630 DIM b(62):RESTORE 3640:FOR n [3396]
=1 TO 62:READ b(n):NEXT
3640 DATA 239,179,179,179,142,159 [11129]
,179,159,142,179,179,142,119,106,
106,106,119,142,179,159,179,159,1
42,179,213,213,239,179,179,106,11
9,142,142,179,159,179,159,106,119
,142,142,119,106,106,106,89,119,1
42,142,179,159,179,159,142,179,21
3,213,239,179,179,179,0
3650 ENV 1,1,15,1,5,-3,2:ENV 2,15 [4390]
,-1,10:ENT-1,10,-1,1:ENT-2,1,-127
,1,5,24,1
3660 EVERY 10,1 GOSUB 3870 [663]
3670 TRS=CHR$(23)+CHR$(1):NR$=CHR [2028]
$(23)+CHR$(0)
3680 INK 0,0:INK 15,6:BORDER 0:MO [1886]
DE 0:FOR A=1 TO 14:INK A,26-A:NEX
T
3690 FOR A=0 TO PI STEP 0.05:X=SI [5554]
N(A)*40:PLOT 320+X,200+COS(A)*40,
1:DRAWR -X*2,0:NEXT
3700 PLOT 470,50:P=2:FOR A=PI/2 T [5022]
O 2.5*PI STEP 0.09:X=SIN(A)*150:Y
=COS(A)*60:DRAW 320+X+Y,200+Y-X,P
:=P+1+14*(P=15):GOSUB 3740:NEXT
3710 P=2:PLOT 260,260:FOR A=0 TO [8140]
2*PI STEP 0.045:X=SIN(A)*150:Y=CO
S(A)*60:DRAW 320-X-Y,200+Y-X,P:P=
P+1+14*(P=15):GOSUB 3740:NEXT
3720 P=2:PLOT 390,200:FOR A=0 TO [5069]
2*PI STEP 0.0225:X=SIN(A)*170:Y=C
OS(A)*70:DRAW 320+Y,200+X,P:P=P+1
+14*(P=15):GOSUB 3740:NEXT:GOTO 3
750
3730 FOR A=2 TO 14:INK A,26:CALL [3523]
&BD19:INK A,0:NEXT:GOTO 3730
3740 IF P=15 THEN P=8:RETURN ELSE [2371]
RETURN
3750 ORIGIN 0,0:INK 4,0:PLOT -10, [2125]
-10,4:a$="LA MOLECULE"
3760 ENC=1:EN2=15:Y=330:X=124:GOS [2275]
UB 3790
3770 PEN 15:LOCATE 10,19:PRINT TR [3906]
$: "BY":a$="CLAUDE LE MOULLEC"
3780 PLOT -10,-10,1:TAG:MOVE 46,4 [5043]
0:PRINT A$:PLOT -10,-10,15:MOVE
50,36:PRINT A$:TAGOFF:PRINT NRS:
GOTO 3840
3790 TAG:FOR H=1 TO LEN(A$):BS=MI [3104]
DS(A$,H,1):MOVE (H*36)-36,14:PRIN
T BS:NEXT:TAGOFF
3800 FOR G=0 TO 14 STEP 2:FOR H=0 [1911]
TO 432 STEP 2
3810 IF TEST(H,2+G)=4 THEN PLOT H [3439]
+X,Y+(G*2),ENC:PLOT H+X,Y+2+(G*2)
,EN2
3820 NEXT H,G:LOCATE 1,25:PRINT S [3898]
PACES(18):INK 4,26:RETURN
3830 PLOT H+X,Y+(G*2),ENC:PLOT H+ [1582]
X,Y+2+(G*2),EN2:RETURN
3840 a=1 [327]
3850 a=a+1:IF a>14 THEN a=2 [1842]
3860 INK A,26:CALL &BD19:INK A,0: [3783]
a$=INKEY$:IF a$="" THEN 3850 ELSE
RETURN
3870 qq=qq+1:IF qq=62 THEN qq=1:w [2097]
=1
3880 IF qq=32 THEN w=3 [276]
3890 SOUND 4,b(qq),19,14,w=2,0:1:IF [6656]
w=5 THEN SOUND 2,0,10,15,w=2,0:2:w
=1:RETURN ELSE SOUND 1,w*150,6,15
,1,2,2:RETURN
3900 RETURN [555]

```

```

9700:F3 F3 F3 F3 F3 F3 E6:5B
9708:F3 9C 79 CC F3 9C 79 CC:7B
9710:F3 9C 79 CC F3 9C 79 CC:83
9718:F3 9C 79 CC F3 9C 79 CC:8B
9720:F3 9C 79 CC F3 9C 79 CC:93
9728:F3 9C 79 CC F3 9C 79 CC:9B
9730:F3 9C 79 CC F3 9C 79 CC:A3
9738:F3 F3 F3 F3 F3 F3 CC:EB
9740:F3 F3 F3 F3 F3 F3 E6:9E
9748:F3 F3 F3 F3 F3 F3 CC:37
9750:F3 9C 79 CC F3 9C 79 CC:C3
9758:F3 9C 79 CC F3 9C 79 CC:CB
9760:F3 9C 79 CC F3 9C 79 CC:D3
9768:F3 9C 79 CC F3 9C 79 CC:DB
9770:F3 F3 F3 F3 F3 F3 CC:64
9778:E6 CC CC CC CC CC CC:09
9780:F3 F3 F3 F3 F3 F3 E6:DE
9788:B7 3D 3D 6C B6 3E 3E 6E:DE
9790:B7 3D 3D 6C B6 3E 3E 6E:E3
9798:B7 3D 3D 6C B6 3E 3E 6E:EB
97A0:B7 3D 3D 6C B6 3E 3E 6E:F3
97A8:B7 3D 3D 6C B6 3E 3E 6E:FB
97B0:B7 3D 3D 6C B6 3E 3E 6E:03
97B8:E6 CC CC CC CC CC:09

```

```

97C0:F3 E6 F3 E6 B6 E6 B6 E6:A3
97C8:B6 E6 B6 E6 B6 E6 B6:E7
97D0:B6 E6 B6 E6 B6 E6 B6:BF
97D8:F3 E6 F3 E6 CC CC CC:57
97E0:F3 E6 F3 E6 B6 E6 B6:E3
97E8:B6 E6 B6 E6 B6 B6 E6:DF
97F0:F3 E6 F3 E6 CC CC CC:77
9800:0C 0C 0C 0C 0C 0C 04:58
9808:08 00 00 04 08 00 04:00
9810:08 00 00 04 08 00 04:08
9818:08 00 00 04 08 00 04:10
9820:08 00 00 04 08 00 04:18
9828:08 00 00 04 08 00 04:20
9830:08 00 00 04 0C 0C 0C:20
9840:00 54 A8 00 A8 54 00:B4
9848:54 00 00 A8 A8 00 00:5A
9850:A8 00 00 54 A8 00 00:5C
9858:A8 00 00 54 A8 00 00:5D
9860:A8 00 00 54 A8 00 00:5D
9868:A8 00 00 54 A8 00 00:5E

```

```

9870:A8 00 00 54 54 00 00:A8:E4
9878:00 A8 54 00 00 54 A8:00:EC
9880:00 51 0A 00 00 51 0A:00:04
9888:00 41 82 00 00 C3 C3:00:0F
9890:00 C3 C3 00 41 93 33:22:73
9898:41 D6 76 22 4B D6 FC:73:FE
98A0:E3 76 76 27 41 76 76:8A:9E
98A8:41 33 67 8A 00 33 67:00:47
98B0:00 33 CC 00 00 45 8A:00:7F
98B8:00 51 2A 00 00 51 A8:00:3C
98C0:00 15 A4 00 00 3F 0C:00:CE
98C8:15 2E 0C 08 15 0C 08:08:5A
98D0:2E 5D FF 0C 2E 0C AE:0C:CD
98D8:2E 0C AE 0C 2E 0C AE:0C:40
98E0:2E 0C AE 0C 4D FF 08:08:73
98E8:04 0C 08 08 00 0C 00:7C:7C
98F0:00 0C 0C 00 00 04 08:00:14
98F8:00 51 0A 00 00 51 A8:00:7C
9900:00 04 F3 00 00 0C 59:A2:41
9908:00 0C 0C 0C 2A 0C 0C:F3:AD
9910:04 0C 0C 59 04 5D FF:59:5D
9918:04 0C AE 59 2E 0C AE:59:F3
9920:0E 0C AE 59 04 0C AE:59:33
9928:04 5D FF 59 04 0C 0C:59:58

```


CPC

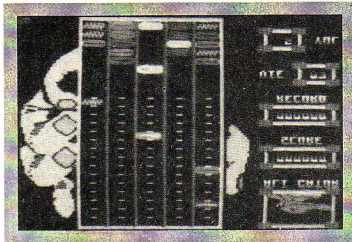
PROGRAMMATION

9930:00	0C	0C	08	00	0C	0C	08:01
9938:00	04	0C	08	00	04	0C	08:67
9948:00	15	0A	00	15	00	00:25	
9948:00	04	08	00	0C	0C	00:9D	
9950:00	0C	0C	00	04	0C	05:15	
9958:04	FF	AE	0C	5D	0C	1D:3C	
9960:0C	5D	0C	1D	0C	5D	1D:FD	
9968:0C	5D	0C	1D	0C	FF	AE	1D:3F
9970:04	0C	0C	2A	04	0C	1D:68	
9978:00	0C	3F	00	15	2A	00:8A	
9980:00	F3	08	00	51	A6	00:EA	
9988:51	0C	0C	00	F3	0C	08:49	
9990:A6	0C	0C	08	A6	FF	AE	08:65
9998:A6	5D	0C	08	A6	5D	0D:FD	
99A0:A6	5D	0C	59	A6	5D	08:21	
99A8:A6	FF	AE	08	A6	0C	08:29	
99B0:A6	0C	0C	00	04	0C	09:39	
99B8:00	0C	08	00	0C	08	09:91	
99C2:2A	00	15	1F	00	2F:EA		
99C8:05	07	0A	09	03	FF	09:C4	
99D2:01	57	FF	09	01	BA	30	AA:15
99D8:01	BA	FF	AA	01	BA	FF	AA:35
99E0:01	BA	FF	AA	01	BA	FF	AA:35
99E8:01	BA	30	AA	01	57	FF	09:36
99F0:00	03	FF	00	05	0B	07	0A:68
99F8:1F	00	2F	2A	00	00	15:EA	
9A00:00	00	20	00	00	51	F3	00:25
9A08:00	F3	0C	88	00	E6	0C	88:44
9A10:51	0C	0C	88	51	D8	FC	00:AC
9A18:51	0C	E4	CC	51	0C	E4	CC:7C
9A20:51	0C	E4	CC	51	0C	E4	CC:7C
9A28:51	D8	FC	CC	51	CC	CC	11:14
9A30:00	E6	0C	88	85	E6	0C	88:6B
9A38:51	0A	0C	2A	2A	0C	09:17	
9A40:2A	00	00	00	51	E6	0C	88:6F
9A48:05	E6	0C	88	00	E6	0C	88:6F
9A50:51	0C	CC	CC	51	D8	FC	00:FC
9A58:51	0C	E4	CC	51	0C	E4	CC:80
9A60:51	0C	E4	CC	51	0C	E4	CC:80
9A68:51	D8	F0	CC	51	0C	88:34	
9A70:00	E6	0C	88	00	F3	0C	88:80
9A78:00	51	F3	00	00	00	00:8D	
9A80:00	00	15	00	D9	A2	2F:6A	
9A88:44	0C	D9	0A	44	0C	D9	08:80
9A90:0C	0C	A2	CC	F0	E4	A2:62	
9A98:0C	D8	CC	A2	CC	D8	CC	A2:6A
9AA0:0C	D8	CC	A2	CC	D8	CC	A2:52
9AA8:0C	F0	E4	A2	44	CC	A2:82	
9AB0:44	0C	D9	0A	44	0C	F3	07:72
9AB8:00	F3	A2	00	00	00	00:1E	
9AC0:00	00	00	00	F3	A2	09:7A	
9AC8:44	0C	F3	00	44	D9	09:22	
9AD0:44	0C	CC	A2	CC	F0	E4	A2:1A
9AD8:0C	D8	CC	A2	CC	D8	CC	A2:8A
9AE0:0C	D8	CC	A2	CC	D8	CC	A2:92
9AE8:0C	F0	E4	A2	CC	CC	A2:CA	
9AF0:44	0C	D9	0A	44	0C	D9	0A:4C
9AF8:00	D9	A2	2F	00	00	15:8E	
9B00:05	A2	05	A2	93	63	1B	73:91
9B08:B3	27	93	63	51	0A	51	0A:41
9B10:51	A2	F3	0C	A6	0C	0C	88:93
9B18:00	A2	0C	0C	05	A2	05	A2:43
9B20:04	A2	04	F3	86	06	59:2E	
9B28:0E	59	64	94	0C	04	09:31	
9B30:51	0A	01	0C	0C	0C	0C:83	
9B38:0C	0C	A6	0C	F3	0C	51	A2:67
9B40:51	F3	08	A6	49	A6	0D:1B	
9B48:A6	59	A6	49	0C	08	0A	08:19
9B50:B2	71	1A	25	7A	75	BA	09:50
9B58:75	BA	75	BA	1A	25	B2	71:EF
9B60:00	88	44	CC	44	CC	44	CC:5F
9B68:44	0C	44	CC	0A	88	A2	08:AB
9B70:A2	0C	0A	88	44	CC	44	CC:43
9B78:44	0C	44	CC	44	CC	08:47	
9B80:00	51	44	05	CC	88	CC	88:9D
9B88:00	88	CC	88	CC	88	44	09:8B
9B90:44	CC	88	CC	88	CC	88:83	
9B98:00	88	CC	88	44	95	00	51:8D
9BA0:00	00	00	00	E7	00	00:15	
9BA8:00	00	00	E7	00	00	00:DF	
9BB0:51	8E	00	00	00	00	51	8E:5F
9BB8:00	00	00	00	00	00	00:04	
9BC0:00	00	00	00	00	00	00:0C	
9BC8:8E	AA	00	00	51	8E	AA:5D	
9BD0:00	00	00	00	E7	5D	00:00:08	
9BD8:00	E7	5D	00	00	00	51	8E:5F
9BE0:AA	00	00	00	51	8E	AA	00:00:BA
9BE8:00	00	E7	5D	00	00	00:00:AC	
9BF0:E7	5D	00	00	00	51	8E	AA:44
9BF8:00	00	00	51	8E	AA	00:00:99	
9C00:00	E7	5D	00	00	00	E7:89	
9C08:5D	00	00	51	8E	AA	00:00:29	
9C10:00	00	51	8E	AA	00:00:29		
9C18:E7	5D	00	00	00	E7	5D:8E	
9C20:00	00	00	8E	AA	00	00:00:7E	
9C28:00	00	8E	AA	00	00	00:14	
9C30:00	00	00	00	00	00	00:00:CC	
9C38:00	00	00	00	00	00	00:00:DA	
9C40:00	00	00	00	00	00	00:00:DC	
9C48:00	00	00	00	00	00	00:00:EC	
9C50:00	00	00	00	00	00	00:00:EC	
9C58:00	00	00	00	00	00	00:00:PC	
9C60:00	00	00	00	00	00	00:00:FC	
9C68:00	00	00	00	00	00	00:00:04	
9C70:00	00	00	00	00	00	00:00:0C	
9C78:00	00	00	00	00	00	00:00:1A	
9C80:00	00	00	00	00	00	00:00:1C	
9C88:00	00	00	00	00	00	00:00:2A	
9C90:00	00	00	00	00	00	00:00:2C	
9C98:00	00	00	00	00	00	00:00:3A	
9CA0:00	00	00	00	00	00	00:00:3C	
9CA8:00	00	00	00	00	00	00:00:44	
9CB0:00	00	00	00	00	00	00:00:4C	
9CB8:00	00	00	00	00	00	00:00:54	
9CC0:00	00	00	00	00	00	00:00:5C	
9CC8:00	00	00	00	00	00	00:00:64	
9CD0:00	00	00	00	00	00	00:00:6A	
9CD8:00	00	00	00	00	00	00:00:7A	
9CE0:00	00	00	00	00	00	00:00:7C	
9CE8:00	00	00	00	00	00	00:00:84	
9CF0:00	00	00	00	00	00	00:00:8C	
9CF8:00	00	00	00	00	00	00:00:94	
9D00:DD	5E	00	DD	56	01	DD	6E:9D
9D08:02	DD	66	03	06	08	05	8E:78
9D10:06	92	1A	77	23	13	10	FA:42
9D18:EI	CD	26	BC	C1	10	EF	9C:88
9D20:15	3F	15	3F	45	45	45	45:8D
9D28:04	04	04	04	05	FF	00	00:90
9D30:02	2A	15	2A	00	0A	00	00:98
9D38:00	08	00	08	55	FF	00	00:AA
9D40:15	3F	15	15	00	45	00	45:9C
9D48:00	08	04	00	45	FF	00	00:AA
9D50:15	3F	15	15	00	45	00	45:9D
9D58:00	0C	00	04	55	FF	00	00:00
9D60:00	15	00	3F	00	CF	45	45:98
9D68:04	0C	04	04	00	55	00:00:2F	
9D70:15	3F	15	00	45	00	00:CF:80	
9D78:00	04	04	04	05	FF	00	00:DC
9D80:15	3F	15	15	00	45	00	45:CF:87
9D88:04	04	04	04	45	FF	00	00:F0
9D90:15	3F	00	15	00	45	00	45:8A:92
9D98:00	08	04	00	55	00	00:FA	
9DA0:15	3F	15	15	00	45	00	45:CF:95
9DA8:04	04	04	04	05	FF	00	00:10
9DB0:15	3F	15	15	00	45	00	45:CF:CS
9DB8:00	04	00	04	55	FF	00	00:10
9DC0:00	00	00	00	00	00	00:00:5D	
9DC8:00	00	00	00	00	AA	00:00:71	
9DD0:00	00	00	00	00	00	00:00:6D	
9DD8:00	00	00	00	AA	55	00:00:DA	
9DE0:00	00	00	00	00	00	00:00:7D	
9DE8:00	00	00	00	00	AA	00:00:A1	
9DF0:00	00	00	00	00	00	00:00:8D	
9DF8:00	00	00	00	00	AA	55	00:00:94
9E00:00	2A	2A	2A	45	00	00:00:F3	
9E08:00	00	00	00	00	00	00:00:A6	
9E10:00	2A	15	15	45	45	00:00:45	
9E18:00	00	00	00	00	00	00:00:C2	
9E20:00	2A	00	2A	00	00:8A:66		
9E28:00	00	00	00	00	AA	00:00:02	
9E30:15	3F	15	3F	45	45	45	45:9E
9E38:04	0C	04	04	55	55	00:00:85	
9E40:15	3F	15	3F	45	45	45	45:AE
9E48:04	08	04	04	55	55	00:00:89	
9E50:15	3F	15	3F	45	45	45	45:F8
9E58:04	00	04	00	55	55	00:00:A9	
9E60:15	2A	15	3F	45	45	45	45:AA
9E68:04	04	04	04	55	55	00:00:D1	
9E70:15	3F	15	3F	45	45	45	45:08
9E78:04	08	04	00	55	55	00:00:D9	
9E80:15	3F	15	3F	45	45	45	45:28
9E88:04	08	04	00	55	55	00:00:DF	
9E90:15	3F	15	3F	45	45	45	45:06
9E98:04	00	04	05	55	55	00:00:F9	
9EA0:15	15	15	15	15	45	45	45:12
9EA8:04	0C	04	04	55	55	00:00:25	
9EB0:15	3F	00	2A	00	8A	00	8A:15
9EB8:00	08	00	00	55	55	00:00:29	
9EC0:15	3F	00	2A	00	8A	00	8A:25
9EC8:00	08	00	00	55	AA	00	00:3B
9ED0:15	15	15	15	15	45	45	45:8A:6A
9ED8:04	08	04	04	55	55	00:00:4D	
9EE0:15	00	15	00	45	00	45	00:00:9E
9EE8:04	04	04	00	55	FF	00:00:39	
9EF0:15	15	15	3F	45	CF	45	45:46
9EF8:04	04	04	04	55	55	00:00:65	
9F00:15	15	15	15	15	45	CF	45:CF:FF
9F08:04	0C	04	04	55	55	00:00:86	
9F10:15	3F	04	15	05	45	45	45:45:07
9F18:04	04	15	04	55	FF	00:00:82	
9F20:15	3F	15	3F	45	45	45	45:45:8F
9F28:04	0C	04	00	55	00	00:00:98	
9F30:15	3F	15	3F	45	45	45	45:9F
9F38:04	04	04	04	55	FF	00:AA:FF	

Alignez-vous ...

MANO NÉGRA

*Savez-vous réussir un Yo!
En d'autres termes, êtes-vous capable
d'aligner dans un sens quelconque,
3 Zors de couleur identique?*



Ce jeu de pure stratégie, réclame un sang-froid hors pair. L'action consiste à disposer judicieusement, afin de constituer des alignements, les pions multicolores qui chutent de rangées verticales. Ne vous y fiez pas, on est très vite débordé!.

Sauvegarde

Sauvez sous un nom de votre choix, le programme Basic principal. Entrez ensuite par Amsaisie V.2 en vous reportant à son mode d'emploi, les deux

listings de codes hexadécimaux.

Nom	Adr. déb.	Long
MANOSPRI	&8000	&D68
MANOROUT	&9D20	&35C

La longueur est ici précisée à l'attention de ceux qui envisagent de morceler leur travail en plusieurs fichiers qui devront ultérieurement être réunis en deux fichiers définitifs.

Claude Le Moullec

100 REM	:	:	[1736]
20 REM	:	:	[419]
30 REM	:	MANO NEGRA	[619]
40 REM	:	:	[419]
50 REM	:	:	[1736]
60 REM	:	:	[419]
70 REM	:	LMC SOFTWARE	[538]
80 REM	:	Claude LE MOULLEC	[2031]

90 REM	:	:	[419]
100 REM	:	:	[1736]
110 REM	:	:	[419]
120 REM	:	REDEFINITION	[1622]
130 REM	:	:	[419]
140 REM	:	:	[1736]
150 SYMBOL	AFTER 169	:	[1435]
160 SYMBOL	170,170,170,170,170,17	:	[2823]
170 SYMBOL	171,0,84,84,84,84,84,8	:	[1597]
180 SYMBOL	201,0,0,0,0,1,1,0,1	:	[1667]
190 SYMBOL	202,14,31,123,245,226,	:	[2905]
200 SYMBOL	203,128,224,192,224,25	:	[3220]
210 SYMBOL	204,0,0,0,58,239,255,2	:	[2001]
220 SYMBOL	205,3,3,1,0,1,3,15,62	:	[1814]
230 SYMBOL	206,108,113,191,220,17	:	[2683]
240 SYMBOL	207,15,252,128,0,192,2	:	[2530]
250 SYMBOL	208,240,0,0,0,0,0,12	:	[1947]
260 SYMBOL	209,124,249,0,0,0,0,0,	:	[1555]
270 SYMBOL	210,231,223,0,0,0,0,0,	:	[2016]
280 SYMBOL	211,63,255,0,0,0,0,0,0	:	[2345]
290 SYMBOL	212,192,192,0,0,0,0,0,	:	[1836]
300 SYMBOL	213,254,188,94,47,255,	:	[2394]
310 SYMBOL	214,254,252,30,15,255,	:	[2281]
320 MEMORY	&7FFF:LOAD "IMANOSPRI"	:	[1098]
330 LOAD	"IMANOROUT",&9D20	:	[1427]
340 REM	:	:	[1736]
350 REM	:	:	[419]
360 REM	:	VARIABLES DE BASE	[2081]
370 REM	:	:	[419]
380 REM	:	:	[1736]
390 DEFINT	a-z:MODE 0:BORDER 0	:	[1453]
400 RESTORE	410:FOR h=0 TO 15:REA	:	[1813]
410 DATA	0,1,15,5,3,22,14,7,16,9,	:	[1985]
420 DEF FN	pt(x,y)=&C000+(y-1)*80	:	[2454]
430 DEF FN	po(x,y)=&E003+(y-1)*80	:	[1359]
440 DIM	sp(15):FOR h=0 TO 10:sp(h	:	[3664]


```

+1)=&8BEE+(h*42):NEXT
450 DIM je(9,26):DIM cx(2000):DIM [2824]
  cy(2000):DIM cp(2000)
460 FOR H=1 TO 5:JE(H,19)=10:NEXT [1313]
470 FOR h=1 TO 9:je(h,25)=10:je(h [3360]
  ,26)=10:je(h,19)=10:je(h,18)=10:N
EXT
480 FOR h=18 TO 26:je(1,h)=10:je( [3770]
  2,h)=10:je(9,h)=10:je(8,h)=10:NEX
T
490 ali3$=CHR$(209)+CHR$(210)+CHR [2257]
  $(211)+CHR$(212)
500 ali2$=CHR$(205)+CHR$(206)+CHR [2118]
  $(207)+CHR$(208)
510 ali1$=CHR$(201)+CHR$(202)+CHR [3744]
  $(203)+CHR$(204)
520 tr$=CHR$(22)+CHR$(1):nr$=CHR$ [1849]
  (22)+CHR$(0)
530 ENT 2,200,20,5: BUT=0: VIE=3 [2535]
540 ON BREAK GOSUB 2850 [755]
550 REM :::::::::::::::::::: [1736]
560 REM : [419]
570 REM : DESSIN DECOR [1958]
580 REM : [419]
590 REM : [1736]
600 POKE &A01A,165:POKE &A01E,16: [2681]
  CALL &A00D, FN pt(2,2)-1,&8000
610 POKE &A01A,69:POKE &A01E,6:CA [2723]
  LL &A00D, FN pt(27,5),&8A50
620 FOR h=128 TO 140 STEP 4:PLOT [2757]
  h,0: DRAW h,400:NEXT
630 PLOT 128,1,10: DRAW 128,398:DR [5171]
  AW 414,398: DRAW 414,1: DRAW 128,1:
  PLOT 128,3: DRAW 414,3: PLOT 128,39
  6: DRAW 414,396
640 PLOT 132,5,6: DRAW 132,394: DRA [3547]
  W 410,394: DRAW 410,5: DRAW 132,5: P
  LOT 132,7: DRAW 410,7: PLOT 132,392
  : DRAW 410,392
650 PLOT -10, -10, 11: TAG: MOVE 498, [5078]
  368: PRINT ali1$: MOVE 498,352: PRI
  NT ali2$: MOVE 498,336: PRINT ali3
  $: TAGOFF
660 DATA 16,1,5,16,5,5,16,8,5,16, [2885]
  10,5,16,13,5,16,15,5,18,18
670 DATA 3,18,20,3,16,22,3,16,24, [2513]
  3,16,1,5,20,1,5,16,8,3,20
680 DATA 8,3,16,13,3,20,13,3,18,1 [2824]
  8,3,20,18,3,16,22,3,18,22,3
690 e1=10:e2=6:RESTORE 660:FOR i= [3830]
  1 TO 20:GOSUB 700:NEXT:GOTO 700
700 READ X,Y,L: x1=14+(X-1)*32:y1= [2428]
  388-(Y-1)*16
710 IF i>10 THEN 750 [1058]
720 FOR J=0 TO 2 STEP 2:PLOT x1,y [2455]
  1+j,e1: DRAW x1+(1-1)*32,y1+j:NEXT
730 FOR J=4 TO 6 STEP 2:PLOT x1,y [4175]
  1+j,e2: DRAW x1+(1-1)*32,y1+j:NEXT
740 RETURN [555]
750 PLOT x1+4,y1+2,e1: DRAW x1+4,y [2465]
  1+2-(1-1)*16
760 PLOT x1,y1+2,e2: DRAW x1,y1+2- [1096]
  (1-1)*16
770 RETURN [555]
780 LOCATE 1,1:PRINT tr$:RESTORE [4715]
790:FOR h=1 TO 20:GOSUB 810:NEXT:
GOTO 830
790 DATA 16,1,16,5,20,1,20,5,16,8 [3014]
  ,20,8,16,10,20,10,16,13,20,13
800 DATA 16,15,20,15,18,18,20,18, [1874]
  18,20,20,20,16,22,18,22,16,24,18,
  24
810 READ x,y: PEN 9: LOCATE x,y: PRI [3811]
  NT CHR$(170): PEN 12: LOCATE x,y
820 PRINT CHR$(171): RETURN [1611]
830 RESTORE 840: FOR h=1 TO 9: READ [4407]
  ZL,YL,mo,a$: GOSUB 850: NEXT: GOTO
  940
840 DATA 32,9,0,"000000",32,11,1, [7666]
  "SCORE",32,14,0,"000000",36,19,0,
  "00",32,23,0,"00",36,23,1,"YOL",3
  0,19,0,"VIE",30,6,1,"ALI GATOR",3
  2,16,0,"RECORD"
850 A$=UPPER$(A$):FOR T=1 TO LEN( [2706]
  A$): SP=(ASC(MID$(A$,T,1)))-48
860 IF SP<0 OR SP>43 THEN SP=43 [1734]
870 CALL &9D20, FN PT(ZL,YL)+(T*2) [3746]
  +MO,&9D40+(SP*16): NEXT
880 RETURN [555]
890 REM :::::::::::::::::::: [1736]
900 REM : [419]
910 REM : DEBUT DE PARTIE [4179]
920 REM : [419]
930 REM : [1736]
940 GOSUB 2870 [901]
950 BUT= BUT+1 [801]
960 TA=(BUT*7):TA1=TA:GOSUB 2550: [2127]
  GOSUB 2570
970 VIT1=200-TA: IF VIT1<100 THEN [1775]
  VIT1=100
980 VIT2=50-(BUT*3): IF VIT2<27 TH [2424]
  EN VIT2=50
990 GOSUB 2830:A$="EPREUVE NR"+ST [4188]
  R$(BUT):ZL=11:YL=13:GOSUB 850
1000 RESTORE 2720:GOSUB 2690:FOR [4133]
  T=1 TO 4000:NEXT:GOSUB 2830:GOSUB
  2790
1010 FIN=0:PER=0:PEL=0:POKE &A01A [2075]
  ,7:POKE &A01E,6
1020 CALL &A00D, FN po(5,19),sp(9) [1614]
  :xj=5:yj=16
1030 c=INT(RND*8)+1:x=INT(RND*5)+ [4824]
  3:CALL &A00D, FN po(x,1),sp(c)
1040 cx(1)=x:cy(1)=1:cp(1)=c:cc=1 [2462]
  :dp=1
1050 REM :::::::::::::::::::: [1736]
1060 REM : [419]
1070 REM : ROUTINE PRINCIPALE [2225]
1080 REM : [419]
1090 REM : [1736]
1100 EVERY VIT1,3 GOSUB 1380 [1529]
1110 EVERY VIT2,2 GOSUB 1410 [1800]
1120 IF PEL=4 THEN 1350 [582]
1130 IF fin=1 THEN DI:GOTO 2270 [689]
1140 IF INKEY(8)*INKEY(74)=0 AND [2582]
  xj>3 THEN 1190
1150 IF INKEY(1)*INKEY(75)=0 AND [1457]
  xj<7 THEN 1220
1160 IF INKEY(9)*INKEY(76)=0 THEN [1257]
  1250
1170 GOTO 1120 [347]
1180 REM :: A GAUCHE :: [1052]

```

PROGRAMMATION

```

1190 DI:CALL &A04B, FN po(xj-1, yj) [2724]
.FN po(xj, yj):xj=xj-1
1200 FOR t=1 TO 30:NEXT:SOUND 1,1 [1893]
00,2,5,EI:GOTO 1120
1210 REM ::: A DROITE ::: [1375]
1220 DI:CALL &A04B, FN po(xj+1, yj) [4305]
.FN po(xj, yj):xj=xj+1
1230 FOR t=1 TO 30:NEXT:SOUND 1,1 [1893]
00,2,5,EI:GOTO 1120
1240 REM ::: LACHER ::: [317]
1250 DI:IF pel=0 THEN EI:GOTO 112 [2611]
0
1260 A=0:FOR H=1 TO PEL:IF JE(XJ, [2488]
19+H)<>0 THEN A=1
1270 NEXT:IF A=1 THEN SOUND 1,150 [2659]
,15,7,EI:GOTO 1120
1280 CALL &A00D, FN po(xj, 19), sp(9 [1635]
)
1290 FOR h=1 TO pel:p=pl(h):CALL [2733]
&A00D, FN po(xj, 19-h), SP(P)
1300 FOR g=20 TO 24:IF je(xj, g)=0 [1946]
THEN a=g
1310 NEXT:CALL &A00D, FN po(xj, a), [2305]
sp(p):je(xj, a)=p
1320 GOSUB 1620:NEXT h:CALL &A00D [3052]
.FN po(xj, 19), sp(9)
1330 PEL=0:EI:GOTO 1120 [845]
1340 REM 4 ZORS SUR LA RAQUETTE [1258]
1350 A=0:FOR H=1 TO 4:IF JE(XJ, 19 [2639]
+H)<>0 THEN A=1
1360 NEXT:IF A=1 THEN fin=1:PER=1 [4316]
:GOTO 1280 ELSE 1280
1370 REM : APPARTION D'UN ZOR : [1858]
1380 DI:c=INT(RND*8)+1:x=INT(RND* [2932]
5)+3:CALL &A00D, FN po(x, 1), sp(c)
1390 cc=cc+1:cx(cc)=x:cy(cc)=1:cp [1885]
(cc)=c:EI:RETURN
1400 REM : CHUTE DES ZORS : [2353]
1410 DI:FOR h=dp TO cc [874]
1420 x1=cx(h):y1=cy(h):p=cp(h) [1785]
1430 CALL &A032, FN po(x1, y1):y1=y [2269]
1+1:IF y1=16 THEN 1460
1440 CALL &A00D, FN po(x1, y1), sp(p [2326]
):cy(h)=y1
1450 NEXT:EI:RETURN [861]
1460 IF x1=xj THEN 1520 [1206]
1470 IF je(x1, 20)<>0 THEN CALL &A [4107]
00D, FN po(x1, 19), sp(p):fin=1:PER=
1:GOTO 1450
1480 dp=dp+1:FOR g=20 TO 24:IF je [4027]
(x1, g)=0 THEN a=g
1490 NEXT:CALL &A00D, FN po(x1, a), [3274]
sp(p):je(x1, a)=p
1500 GOSUB 1610:GOTO 1450 [1308]
1510 REM CHUTE SUR LA RAQUETTE [1917]
1520 dp=dp+1:pel=pel+1 [2473]
1530 CALL &A00D, FN po(x1, 19-pel), [3002]
sp(p):pl(pel)=p:GOTO 1450
1540 REM ::::::::::::::::::: [1736]
1550 REM : [419]
1560 REM : ALGORITHME POUR [1312]
1570 REM : CALCUL D'ALIGNEMENT : [1039]
1580 REM : (quel boulot !) [1291]
1590 REM : [419]
1600 REM ::::::::::::::::::: [1736]
1610 XF=X1:YF=A:GOTO 1630 [1419]
1620 XF=XJ:YF=A [937]
1630 IF JE(XF, YF+1)=P AND JE(XF, Y [2102]
F+2)=P THEN 1820
1640 IF JE(XF+1, YF)=P AND JE(XF+2 [2007]
,YF)=P THEN 1740
1650 IF JE(XF-1, YF)=P AND JE(XF-2 [3039]
,YF)=P THEN 1780
1660 IF JE(XF-1, YF)=P AND JE(XF+1 [2748]
,YF)=P THEN XF=XF-1:GOTO 1740
1670 IF JE(XF-1, YF-1)=P AND JE(XF [2284]
-2, YF-2)=P THEN 1850
1680 IF JE(XF+1, YF+1)=P AND JE(XF [1491]
+2, YF+2)=P THEN 1890
1690 IF JE(XF+1, YF-1)=P AND JE(XF [2365]
+2, YF-2)=P THEN 1930
1700 IF JE(XF-1, YF+1)=P AND JE(XF [1770]
-2, YF+2)=P THEN 1970
1710 IF JE(XF+1, YF-1)=P AND JE(XF [1820]
-1, YF+1)=P THEN 2010
1720 IF JE(XF-1, YF-1)=P AND JE(XF [2983]
+1, YF+1)=P THEN 2050
1730 RETURN [555]
1740 JE(XF, YF)=0:COX=XF:GOSUB 214 [1626]
0
1750 JE(XF+1, YF)=0:COX=XF+1:GOSUB [1592]
2140
1760 JE(XF+2, YF)=0:COX=XF+2:GOSUB [1019]
2140
1770 TA=TA-1:GOSUB 2570:IF TA=0 T [2763]
HEN FIN=1:RETURN ELSE RETURN
1780 JE(XF, YF)=0:COX=XF:GOSUB 214 [1626]
0
1790 JE(XF-1, YF)=0:COX=XF-1:GOSUB [2525]
2140
1800 JE(XF-2, YF)=0:COX=XF-2:GOSUB [2612]
2140
1810 TA=TA-1:GOSUB 2570:IF TA=0 T [2763]
HEN FIN=1:RETURN ELSE RETURN
1820 JE(XF, YF)=0:JE(XF, YF+1)=0:JE [3254]
(XF, YF+2)=0
1830 COX=XF:GOSUB 2140 [1167]
1840 TA=TA-1:GOSUB 2570:IF TA=0 T [2763]
HEN FIN=1:RETURN ELSE RETURN
1850 JE(XF, YF)=0:COX=XF:GOSUB 214 [1626]
0
1860 JE(XF-1, YF-1)=0:COX=XF-1:GOS [1845]
UB 2140
1870 JE(XF-2, YF-2)=0:COX=XF-2:GOS [2975]
UB 2140
1880 TA=TA-1:GOSUB 2570:IF TA=0 T [2763]
HEN FIN=1:RETURN ELSE RETURN
1890 JE(XF, YF)=0:COX=XF:GOSUB 214 [1626]
0
1900 JE(XF+1, YF+1)=0:COX=XF+1:GOS [1730]
UB 2140
1910 JE(XF+2, YF+2)=0:COX=XF+2:GOS [1552]
UB 2140
1920 TA=TA-1:GOSUB 2570:IF TA=0 T [2763]
HEN FIN=1:RETURN ELSE RETURN
1930 JE(XF, YF)=0:COX=XF:GOSUB 214 [1626]
0
1940 JE(XF+1, YF-1)=0:COX=XF+1:GOS [2692]
UB 2140
1950 JE(XF+2, YF-2)=0:COX=XF+2:GOS [2434]
UB 2140
1960 TA=TA-1:GOSUB 2570:IF TA=0 T [2763]

```


PROGRAMMATION

```

HEN FIN=1:RETURN ELSE RETURN
1970 JE(XF,YF)=0:COX=XF:GOSUB 214 [1626]
0
1980 JE(XF-1,YF+1)=0:COX=XF-1:GOS [3337]
UB 2140
1990 JE(XF-2,YF+2)=0:COX=XF-2:GOS [2641]
UB 2140
2000 TA=TA-1:GOSUB 2570:IF TA=0 T [2763]
HEN FIN=1:RETURN ELSE RETURN
2010 JE(XF,YF)=0:COX=XF:GOSUB 214 [1626]
0
2020 JE(XF+1,YF-1)=0:COX=XF+1:GOS [2692]
UB 2140
2030 JE(XF-1,YF+1)=0:COX=XF-1:GOS [3337]
UB 2140
2040 TA=TA-1:GOSUB 2570:IF TA=0 T [2763]
HEN FIN=1:RETURN ELSE RETURN
2050 JE(XF,YF)=0:COX=XF:GOSUB 214 [1626]
0
2060 JE(XF-1,YF-1)=0:COX=XF-1:GOS [1845]
UB 2140
2070 JE(XF+1,YF+1)=0:COX=XF+1:GOS [1730]
UB 2140
2080 TA=TA-1:GOSUB 2570:IF TA=0 T [2763]
HEN FIN=1:RETURN ELSE RETURN
2090 REM : : : : : [1736]
2100 REM : : : : : [419]
2110 REM : MISE A JOUR COLONNE : [1137]
2120 REM : : : : : [419]
2130 REM : : : : : [1736]
2140 POKE &A06F,40:CALL &A06E,FN [2863]
po(COX,20):POKE &A06F,32
2150 G1=0:FOR I=24 TO 20 STEP -1 [1692]
2160 IF JE(COX,I)=0 THEN 2170 ELS [3330]
E G1=G1+1:G(G1)=JE(COX,I)
2170 NEXT [350]
2180 IF G1=0 THEN RETURN [1135]
2190 FOR H5=20 TO 24:JE(COX,H5)=0 [2882]
:NEXT
2200 FOR I=1 TO G1:CALL &A00D,FN [2705]
PO(COX,25-I),SP(G(I))
2210 JE(COX,25-I)=G(I):NEXT:RETUR [1235]
N
2220 REM : : : : : [1736]
2230 REM : : : : : [419]
2240 REM : FIN DE PARTIE [858]
2250 REM : : : : : [419]
2260 REM : : : : : [1736]
2270 FOR H=0 TO 3:MO=REMAIN(H):NE [2096]
XT
2280 RESTORE 2670:GOSUB 2690 [1762]
2290 ERASE JE,CX,CY,CP:DIM je(9,2 [2724]
6):DIM cx(2000):DIM cy(2000):DIM
cp(2000)
2300 IF PER=1 THEN 2350 [911]
2310 FOR H=1 TO TA1:SC=SC+5:ENV 1 [3429]
0,15,-1,1:SOUND 1,0,15,15,10,15
2320 GOSUB 2430:NEXT:IF SC>REC TH [3948]
EN REC=SC:GOSUB 2490
2330 EI:GOTO 950 [321]
2340 REM : : PERDU : : [1027]
2350 VIE=VIE-1:BUT=BUT-1:IF SC>RE [3288]
C THEN REC=SC:GOSUB 2490
2360 SC=0:GOSUB 2430:IF VIE=0 THE [1800]
N BUT=0:VIE=3
2370 EI:GOTO 950 [321]
2380 REM : : : : : [1661]
2390 REM : : : : : [419]
2400 REM : GESTION SCORES : [1349]
2410 REM : : : : : [419]
2420 REM : : : : : [1661]
2430 IF sc=0 THEN a$="000000":ZL= [2483]
32:YL=9:GOSUB 850:RETURN
2440 a$=STR$(sc):a$=RIGHT$(a$,LEN [2868]
(a$)-1)
2450 IF sc<10 THEN ZL=35:YL=9:GOS [3048]
UB 850:RETURN
2460 IF sc<100 THEN ZL=34:YL=9:GO [2041]
SUB 850:RETURN
2470 IF sc<1000 THEN ZL=33:YL=9:G [2172]
OSUB 850:RETURN
2480 ZL=32:YL=9:GOSUB 850:RETURN [2194]
2490 IF REC=0 THEN a$="000000":ZL [2264]
=32:YL=14:GOSUB 850:RETURN
2500 a$=STR$(REC):a$=RIGHT$(a$,LE [2806]
N(a$)-1)
2510 IF REC<10 THEN ZL=35:YL=14:G [2332]
OSUB 850:RETURN
2520 IF REC<100 THEN ZL=34:YL=14: [3185]
GOSUB 850:RETURN
2530 IF REC<1000 THEN ZL=33:YL=14 [1937]
:GOSUB 850:RETURN
2540 ZL=32:YL=14:GOSUB 850:RETURN [1660]
2550 a$=STR$(vie):a$=RIGHT$(a$,LE [1991]
N(a$)-1)
2560 ZL=37:YL=19:GOSUB 850:RETURN [2079]
2570 a$=STR$(ta):a$=RIGHT$(a$,LEN [1517]
(a$)-1)
2580 SOUND 1,150,50,7,0,2:IF ta<1 [2185]
0 THEN a$=" "+a$
2590 IF TA<100 THEN ZL=32:YL=23:G [3340]
OSUB 850:RETURN
2600 ZL=31:YL=23:GOSUB 850:RETURN [2629]
2610 REM : : : : : [1661]
2620 REM : : : : : [419]
2630 REM : SOUS PROG DIVERS : [1142]
2640 REM : : : : : [419]
2650 REM : : : : : [1661]
2660 REM : : MUSIQUES : : [1065]
2670 DATA 1,213,2,1,0,1,1,213,1,1, [5592]
0,1,1,213,1,1,0,1,17,169,1,10,21
3,1,1,0,1,1,213,1,1,0,1,17,169,1,
10,213,1,1,0,1,49,142,1,42,169,1,
28,213,1,
2680 DATA 1,0,1,49,106,2,42,142,2 [6794]
,28,169,2,1,0,1,49,106,1,42,142,1
,28,169,1,1,0,1,49,106,1,42,142,1
,28,169,1,1,0,1,49,106,4,42,142,4
,28,169,4,1,0,1,0,0,0
2690 ca=1:WHILE ca<0 [1244]
2700 READ ca,p,z:SOUND ca,p,z*10, [2058]
15
2710 WEND:RETURN [935]
2720 DATA 49,159,2,42,190,2,28,23 [1045]
9,2
2730 DATA 17,190,1,10,239,1,49,17 [7440]
9,2,42,213,2,28,239,2,49,159,4,42
,190,4,28,239,4,49,159,2,42,190,2
,28,239,2,49,142,2,42,179,2,28,23
9,2,49,159,2,42,190,2,28,239,2,49
,142,2,42,179,2
2740 DATA 28,239,2,49,134,2,42,15 [8999]

```


Les modes d'adressage (suite)

AUTO-FORMEZ-VOUS A L'ASSEMBLEUR 68000 (5^e partie)

Mais avant cela, il apparaît nécessaire de définir ce qu'est une adresse effective.

L'adresse effective

On utilise cette expression pour désigner les opérandes utilisés dans les instructions 68000. C'est ainsi qu'une adresse effective englobe tous les types d'opérandes connus. Voyez le tableau ci-contre résumant les 15 différentes adresses effectives.

On utilise couramment le symbole <ae> pour caractériser une adresse effective. Une instruction MOVE peut dès lors s'écrire:

MOVE.f <ae>, <ae>

Une adresse effective sera donc un registre quelconque du 68000 ou un emplacement mémoire directement, voire indirectement défini. Reste à découvrir ce que signifient les autres opérandes.

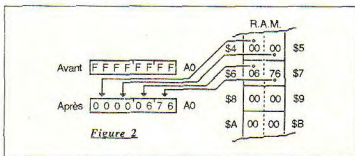
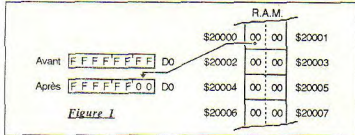
L'adressage absolu, long et court

Nous allons faire intervenir la mémoire dans nos propos. En effet, une adresse absolue (valeur 32 bits) est tout simplement l'une des cases de nos mémoires. Ce n'est donc plus un transfert registre-registre qui a lieu, mais bien un échange entre la mémoire interne du 68000 qui contient les registres, et les mémoires externes telles

L'adressage direct étant d'ores et déjà connu de tous (Micro-Mag n°12), passons au niveau supérieur, j'ai nommé l'adressage absolu long et court.

Symbole	Mode	Symbole	Mode	Symbole	Mode
Dn	Direct	(An)	Indirect	d16 (PC)	Indirect
An	Direct	(An) +	Indirect	d8 (PC, Rn)	Indirect
(x).L	Absolu	-(An)	Indirect	SR	Immédiat
(x).W	Absolu	d16 (An)	Indirect	CCR	Immédiat
#x	Immédiat	d8 (An, Rn)	Indirect	USP	Direct

Les types d'adresses effectives



que la RAM et la ROM. Voyons maintenant le format d'une instruction MOVE correspondante :

MOVE.f (x).f, <ae>

Opération effectuée:

(x).f -> <ae>

Ça se corse ! Examinons un exemple concret (figure 1):

MOVE.B \$20000.L, D0

Le (x) correspond, comme dit précédemment, à une valeur 16 ou 32 bits, tandis que f correspond au format de cette adresse. L'intérêt de ce f est simple: pourquoi aurions-nous besoin de 32 bits pour désigner une valeur? 16 bits suffisent largement. Suivant la grandeur de l'adresse, il paraît normal de pouvoir choisir le format de celle-ci; les conséquences seront un gain de vitesse et de mémoire. A ceci, nous indiquerons que le format f consiste en 2 différentes valeurs. Un W indique une adresse codée sur 16 bits et un L une adresse sur 32 bits. Il est donc possible d'adresser la mémoire de \$F8000 à \$007FFF en utilisant le format 16 bits (adressage absolu court). Si le format est de type long (sur 32 bits), alors l'adresse n'est pas signée et on accède ainsi à la totalité de la mémoire (\$0 à \$FFFFFF). Une astuce: si l'adresse est codée sur 32 bits, le L peut être omis. Le codage d'une adresse sur 8 bits n'existe pas, donc pas de MOVE.L \$6.B, D0. Voici un exemple d'utilisation d'une adresse 16 bits:

MOVE.L \$4.W, A0 (figure 2)

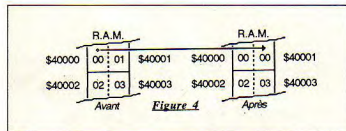
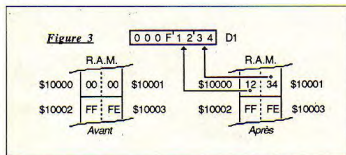
Quelques explications à propos de ce mode d'adressage: Rappelez-vous que chaque adresse d'une mémoire externe contient un octet. Le 68000 commence par déterminer

l'adresse effective de l'opérande source, ici une adresse absolue, puis celle de l'opérande destination, un registre d'adresses. Le 68000 fait ensuite un accès mémoire et récupère, à l'adresse absolue indiquée, le nombre d'octets défini dans le format *f* de l'instruction MOVE. Enfin, il dépose ces valeurs dans le registre d'adresses.

A ce stade, précisons qu'il n'est pas permis d'utiliser un format autre que 8 bits pour l'instruction MOVE, si l'adresse absolue de l'opérande source et/ou destination est impaire. Ainsi, MOVE.W #20001.L, D0 est impossible. En d'autres termes, on accède à une adresse impaire si (et seulement si) l'accès se fait seulement sur 8 bits comme MOVE.B #455.W.D.

D'un autre côté, on peut évidemment utiliser l'adressage absolu avec un opérande de destination ou même avec les 2 opérandes. Voici 2 exemples.

MOVE.W D1,\$1000.W
MOVE.B \$40000,\$40001
(figures 3 et 4)



Faites un petit programme de test (voir article précédent) et amusez-vous. N'omettez pas le RTS final après vos MOVES, sous peine... d'un plantage. Remarquez quand même que le «bas» de la RAM - espace

mémoire allant de \$0000 à \$03FF - est rempli de valeurs système telles les vectorisations pour les exceptions, les interruptions, etc. Une promenade de santé dans les adresses du 68000 serait donc imprudente, tout du moins pour l'instant...

Il eut été judicieux de comparer les temps d'exécution avec les modes d'adressage utilisés, mais réservons cela pour plus tard. Il faut auparavant maîtriser l'ensemble des modes et instructions du 68000.

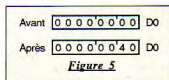
Le mode d'adressage immédiat

L'adressage immédiat est caractérisé par le fait que l'opérande source est une valeur statique, fixée une fois pour toute à la suite du code opératoire de l'instruction utilisée.

Instruction: MOVE.f #x,<ae>
Opération effectuée:
#x -> <ae>

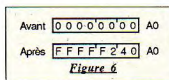
L'opérande source au format

MOVE.B # \$40.D0
(figure 5)



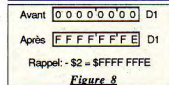
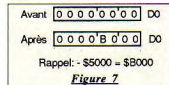
Les 3 formats B, W et L sont possibles. Si le format W est utilisé et si l'opérande destination est un registre d'adresses, alors une extension de signe 32 bits sera effectuée. Le format B est interdit avec un registre d'adresse. Les valeurs de départ des registres et adresses dans nos exemples sont purement fortuites, elles servent juste à bien préciser le changement d'état. Testons ceci avec:

MOVEA.W #SF240.A0
(figure 6)



Voyons également, à l'aide de 2 exemples, ce que l'on obtiendrait en utilisant un nombre négatif:

MOVE.W #- \$5000.D0 et
MOVE.L #-2,D1
(figure 7 et 8)



Dernière chose: il est bien sûr inutile d'essayer de mettre une adresse effective de type immédiat pour l'opérande destination. Ceci donnerait: MOVE.f #x,#y ce qui, vous en conviendrez, n'a aucun sens. Le mode d'adressage immédiat d'un opérande destination est donc absolument prohibé

sous peine d'une mémorable exception «Illegal Instruction», et cela quelle que soit l'adresse effective de l'opérande source (exception faite des instructions LINK et BTST en mode dynamique.)

• Les flags

Représentés par l'octet utilisateur (CCR) du registre SR, ils sont en constante «mouvance». Ainsi, lorsque l'opérande source est nul, le flag Z (Zéro) sera positionné. Si l'opérande est négatif, le flag N sera mis à 1. On peut donc, sans même faire de test de comparaison supplémentaire, effectuer un branchement.

• L'adressage immédiat accéléré

Il existe en effet une variante du mode immédiat. Il s'agit d'utiliser une instruction de type MOVEQ (contraction de MOVE-Quick). Dans ce cas, l'opérande source est obligatoirement immédiat et l'opérande destination un registre de données. La valeur source est un nombre compris entre -128 et +127, le 68000 effectue une extension 32 bits avant de stocker la valeur finale dans Dn. L'intérêt de cette instruction semble limité au premier abord, mais elle met très peu de temps à être exécutée et l'opérande source est stocké dans le code opératoire même de l'instruction. Voici un exemple de comparaison: MOVEQ #120,D0 et MOVE.L #120,D0 donnent un résultat rigoureusement identique.

- 1- l'instruction MOVEQ occupe 2 octets en RAM, et demande 4 cycles d'exécution.
- 2- l'instruction MOVE.L, elle, occupe 6 octets, et 12 cycles pour son exécution.

L'important chez un programmeur étant l'optimisation (tant en vitesse qu'en mémoire) de son programme, nous vous incitons plus que jamais à choisir au mieux vos instruc-

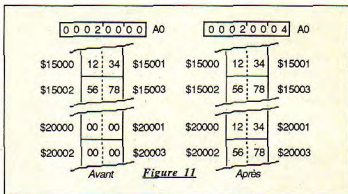
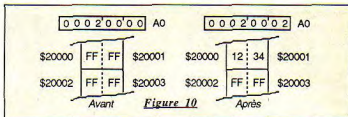
tions, afin de réaliser un programme tendant vers la perfection, encore très rare de nos jours...

Il n'est pas trop tard pour expliquer ce qu'est un cycle: La fréquence d'un 68000 classique est 7.16 Mhz, c'est-à-dire qu'il se produit 7160000 fronts d'horloge par seconde. Ces fronts étant obtenus à l'aide d'un oscillateur ainsi que d'un circuit externe de décodage. L'exécution des instructions est fixé par rapport à un nombre défini de cycles. Vous imaginez facilement que plus le 68000 a d'accès mémoire à faire, plus le nombre de cycles sera élevé. Si, en revanche, il a juste quelques opérations à réaliser avec ses registres, alors la vitesse sera d'autant plus rapide. En moyenne, une instruction 68000 met 20 cycles à être exécutée, le 68000 peut donc exécuter 358000 instructions par seconde. Et encore, il est généré en temps normal par le DMA qui lui «pique» pas mal de cycles dans ses accès à la RAM. Une adresse effective, à elle seule, peut occuper jusqu'à 16 cycles (sans considérer le DMA), alors que l'utilisation de registres internes amène un temps d'exécution presque

Une fois l'instruction tapée, quelle que soit l'adresse où elle se trouve logée, son exécution sera possible sans modification préalable. D'où une puissance du 68000 qui ne cessera de s'accroître, vous le verrez !

Instruction générale:
MOVE.f <ae>.(An)
Opération effectuée:
<ae> -> (An)

Ah, du nouveau sur le front de l'adressage, nous apercevons effectivement que l'opérande destination est «entre parenthèses». C'est en fait pour une raison simple: A quoi sert un registre d'adresses? A adresser!... Bien! Il contient donc une valeur 32 bits indiquant l'adresse où le 68000 ira faire son travail. Cette instruction est radicalement différente du MOVE.f <ae>.An. Cette dernière charge dans An une certaine valeur calculée à partir de l'adresse effective de l'opérande de source. Avec l'adressage indirect, ce n'est pas An qui sera modifié, mais l'emplacement mémoire sur lequel il pointe. Pour clarifier ceci, voici un exemple concret.



2,\$20000 reviendrait au même). Avec l'adressage indirect, le 68000 occupe 24 cycles et seulement 20 avec l'adressage absolu.

Donc, c'est en contradiction avec ce qui a été dit plus haut, mais il faut savoir que dans le 1er cas, le registre A0 est conservé et utilisable pour un prochain accès, ce qui est différent du second cas où aucun registre n'est initialisé.

MOVE.W #1234,(A0)+ ; format W donc A0 <- A0+2
MOVE.L #15000,(A0)+ ; format L donc A0 <- A0+4 (figures 10 et 11)

Ce type d'adressage est fortement utilisé lorsqu'il s'agit de «boucles», par exemple :

```
MOVE.L #30000,A0 ; A0=$30000
MOVEQ #10,D0 ; D0=10; 10 fois
LOOP: MOVE.B D0,(A0)+ ; transfert
      DBF D0,LOOP ; bouclage
      RTS ; retour éditeur
```

Suite à ce programme, le contenu en mémoire sera le suivant :

- \$30000 -> 10
- \$30001 -> 9
- \$30002 -> 8
- \$30003 -> 7
- \$30004 -> 6
- \$30005 -> 5
- \$30006 -> 4
- \$30007 -> 3
- \$30008 -> 2
- \$30009 -> 1
- \$3000A -> 0

Adressage indirect de registre d'adresses avec postincrément

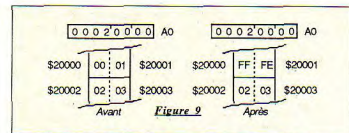
Instruction MOVE:
MOVE.f <ae>.(An)+

Opérations effectuées:
<ae> -> (An)
An <- An+f

Ici, non seulement An sert de pointeur sur l'adresse mémoire à affecter, mais en plus, APRES l'opération (d'où postincrément), An se verra augmenté d'une certaine valeur calculée en fonction du format de l'opérande source. L'opérande source ne sera pas affecté. 2 exemples :

D0 contiendra la valeur \$FFFFFFF (-1), et A0 \$3000B.

Là encore, la dynamique des instructions de notre bon vieux 68000 transforme la programmation en un véritable jeu d'enfant...



immédiat. Il est de ce fait déconseillé d'utiliser à tort et à travers l'adressage absolu. On lui préférera en effet l'adressage indirect, plus rapide et plus court en mémoire.

Mode d'adressage indirect de registre d'adresses

La spécification «indirect» désigne en fait le caractère dynamique de l'adressage.

MOVE.L #20000,A0 ; -> A0=\$20000
MOVE.W #5FFF,(A0) ; -> \$20000 <- 5FF, \$20001 <- 5FE (figure 9)

Notez au passage que ceci ne modifie en rien l'état de An. Le flag N sera positionné suite à ces 2 instructions. Maintenant que nous connaissons l'adressage absolu, nous pourrions effectuer la même opération que la précédente : MOVE.W #5FFF,\$20000.L (notez que taper MOVE #-

PROGRAMMATION
INITIATIONAdressage indirect de
registre d'adresses avec
prédécimation

Inutile de faire tout un discours sur ce mode d'adressage. Le précédent permettait une POSTincrémentation, celui-ci effectue une PRÉ-décimation. C'est à dire que An sera d'abord décrementé puis le transfert aura lieu.

Instruction MOVE :
MOVE.f <ae>,-(An)
Opérations effectuées:
An <- An-f
<ae> -> (An)

Un exemple:
MOVE.L #\$40000,A0
MOVE.W #\$5678,-(A0)
(figure 12)

Quelques remarques à propos de l'adressage indirect de registre d'adresses:

- Il n'est pas permis de faire de même avec des registres de données.

• L'inc/décimation est réelle, le résultat interne de l'opération effectuée par le 68000 est ensuite chargé dans An.

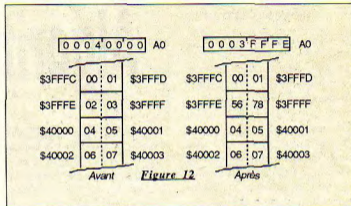
- Aucune extension de signe n'est effectuée.
- Les 3 formats B,W et L sont autorisés.

Bien sûr, pour nos explications, seul l'opérande destination s'est vu affecté d'une adresse effective indirecte. Il est évident que cela peut également s'appliquer dans d'autres cas. De plus, là aussi, les 3 formats sont possibles.

Avant de se quitter, examinons un dernier exemple (ci-contre) qui correspond à un programme de copie de zones mémoires. Tous les types d'adressage indirects sont utilisés.

Quelques modes d'adressage restent à voir prochainement. En attendant, assemblez-bien !

Stéphane Rodriguez



```

MOVE.L #$20000,A0 ; 1ère adresse de départ.
MOVE.L #$22000,A1 ; 2ème adresse de départ.
MOVE.L #$30000,A2 ; 3ème adresse de départ.
MOVE.W #$1000,D0 ; Nombre de boucles
LOOP: MOVE.B (A0)+,(A1) ; 1ère copie (Source postincrémentée)
      MOVE.B -(A1),D1 ; 2ème copie (Source prédécémentée)
      NOTL D1 ; Inversement de D1
      MOVE.B D1,(A2)+ ; Copie en A2 (Destination postincrémentée)
      NOTL D1 ; Ré-inversement.
      DBF D0,LOOP ; Bouclage
      RTS ; Retour
  
```

Algorithmique

LA RECURSIVITÉ

*Voisine de la récurrence mathématique,
la récursivité n'est pas seulement
une méthode de programmation. C'est aussi
une méthode de réflexion qui permet
d'aborder des problèmes a priori
insolubles comme ceux posés par
l'intelligence artificielle.*

Actuellement, la plupart des langages évolués C, Pascal, Lisp, certains Basic, ... etc acceptent des fonctions récursives et proposent ainsi une facilité dont il serait dommage de se passer. L'idée essentielle consiste à transformer un problème complexe en plusieurs sous-problèmes SIMILAIRES mais plus simples et AINSI DE SUITE jusqu'au moment où on est ramené à des cas suffisamment simples pour être traités directement. Exprimé de façon guerrière, ceci s'appelle «Diviser pour régner». Voyons les premières conséquences:

- Il faut trouver le moyen de décomposer le problème, ce n'est pas toujours facile.
- Il est impératif de s'assurer que les sous-problèmes sont calculables, c'est-à-dire qu'on aboutira après un certain nombre d'étapes à un cas trivial. Sinon, soit on a affaire à un cercle vicieux, soit le nombre des sous-problèmes se met à croître vertigineusement et finit par faire planter l'ordinateur.

Premiers pas

Voici quelques exemples illustrant cette manière de penser:

- Monter un escalier de n marches se remène à :
1- Si l'escalier ne comporte qu'une marche, on monte d'une marche et c'est fini (le cas simple).
2- Sinon, il suffit de monter d'une marche (cas simple) et de recommencer l'algorithme avec un escalier de $n-1$

marches (sous-problème similaire).

- Donner la liste des ancêtres d'une personne (en négligeant les collatéraux) se ramène à :
1- Si on connaît le père.
a) Afficher le nom du père (on sait faire).
b) Recommencer avec la liste des ancêtres du père (problème similaire).
2- Si on connaît la mère.
a) Afficher le nom de la mère (on sait faire).
b) Recommencer avec la liste des ancêtres de la mère (problème similaire).
3) - Si on ne connaît ni père, ni mère, c'est fini (cas simple).

- Parfois, on n'est pas sur d'atteindre un cas simple, il faut alors une condition d'arrêt pour échapper à une régression infinie ou dépassant les capacités de l'ordinateur ou encore, qui serait inacceptable en temps machine. Ce problème se rencontre (même pour un humain) au cours d'une partie d'échecs, de dames, etc... Pour chaque déplacement envisagé, il faut évaluer la position, imaginer la réponse de l'adversaire et...ainsi de

suite jusqu'à une profondeur donnée (L'algorithme classique dans les problèmes à deux joueurs s'appelle le minimax, car l'un des joueurs tente de minimiser le gain espéré alors que l'autre le maximise).

- On peut encore citer de nombreux exemples où la notion de récursivité est omniprésente, comme les systèmes experts, l'analyse syntaxique, le calcul symbolique traité par ordinateur, etc. En résumé, l'énoncé d'un problème dans lequel on trouve l'expression "...et ainsi de suite jusqu'à ce que telle condition soit remplie" est de nature essentiellement récursive (La boucle do...while en est d'ailleurs le cas le plus simple). Voyons maintenant comment traiter quelques cas pratiques simples...
L'exemple «bateau» est celui de la fonction factorielle. On sait qu'elle est définie par :
 $fact(n) = n! = n \cdot (n-1) \cdot (n-2) \cdot \dots \cdot 1$
et on peut la calculer à l'aide de la fonction suivante:

```
{
  int f = n ;
  while( -n > 0 )
    f = f * n ;
  return(f) ;
}
```

N. B. : Ici, comme dans la suite et pour éviter les problèmes d'intendance, on considérera qu'on se borne à des valeurs de n suffisamment petites pour ne pas devoir faire intervenir les entiers longs. Mais on peut essayer une approche récursive en remarquant que:
 $fact(1) = 1$ et
 $fact(n) = n * fact(n-1)$
Ayant lu attentivement ce qui précède et sachant en particulier que le C est récursif, on tape avec confiance :

```
fact( n )
int n ;
{
  if( n <= 1 )
    return( 1 ) ;
  else
    return( n * fact( n-1 ) ) ;
}
```

Et ça marche! Examinons un cas similaire mais un chouïa plus compliqué. Soit la fonction donnée par:

$$f(1) = f(2) = 1 \text{ et } f(n) = f(n-1) + f(n-2) \text{ si } n > 2$$

La solution récursive s'écrit immédiatement :

```
f( n )
int n ;
{
  if( n < 3 )
    return( 1 ) ;
```



```
else
return( f(n-1) + f(n-2) );
}
```

Cet exemple est encore relativement facile à écrire de façon non récursive. Il suffit de calculer les valeurs de 1 à n en gardant en mémoire les deux dernières, dans un tableau ou dans des variables (f(n) donne les valeurs successives de la suite dite «de Fibonacci» et qui sont: 1 1 2 3 5 8 13...). Mais si vous cherchez à trouver une solution non récursive à la prochaine, vous allez jongler...

```
ack( m, n )
int m, n ; /* de petites valeurs */
{
if( m==0 )
return( n+1 );
else if( n==0 )
return( ack( m-1, 1 ) );
else
return( ack( m-1, ack( m, n-1 ) ) );
}
```

Comment ça marche?

Euh...Comme si vous le faisiez vous-même... Reprenons la factorielle. Pour calculer sa valeur avec $n = 5$, on fait :

```
fact(5) = 5*( fact(4) ) = 5*( 4 *
fact(3) ) et en fin de compte :
fact(5) = 5*( 4*( 3*( 2*( 1 ) ) ) )
```

Appelons «incarnation», l'ensemble généré par l'appel d'une fonction. C'est la zone - au sommet de la pile - contenant paramètres et variables locales. Elle est détruite en fin d'exécution et la pile est restaurée après que la procédure appelante aie récupéré le résultat. Le calcul de fact(n) va donc commencer par l'ouverture de n-1 incarnations successives avant de pouvoir renvoyer une valeur qui sera celle de fact(1) = 1. C'est seulement alors, que toutes ces incarnations vont se refermer en cascade en effectuant les multiplications. On peut dire qu'on a affaire à une remontée

Listing 1

Désassemblage de la fonction fibo() générée par le compilateur du Mégamax C...

```
fibo:  global %l
LINK  A6,L$6      /* Pas de variables locales => L$6 = 0 */
CMPI  #2,B(A6)    /* 8(A6) contient n */
BGT   L8(PC)     /* Si n > 2, on va en L8 */
MOVE  #1,D0      /* La valeur de retour est mise dans D0 */
BRA   L7(PC)     /* Et on sort */

L8:
MOVE  8(A6),D0   /* On récupère n */
SUBQ  #1,D0      /* D0 = n-1 */
MOVE  D0,-(A7)   /* Passage de n-1 par la pile */
JSR   fibo(PC)   /* Exécute fibo(n-1) */
ADDQ.L #2,A7     /* Restauration de la pile */
MOVE.L D0,-(A7) /* Remplit le résultat */

MOVE  8(A6),D0   /* fibo(n-2) est dans D0 */
SUBQ  #2,D0
MOVE  D0,-(A7)
JSR   fibo(PC)
ADDQ.L #2,A7

MOVE.L D0,D1    /* Passage dans D1 */
MOVE.L (A7)+,D0 /* Récupération de fibo(n-1) */
ADD  D1,D0      /* D0 = fibo(n-1) + fibo(n-2) */
BRA   L7(PC)    /* Kolossale finesse du compilé */

L7:
UNLK  A6        /* Restauration de la pile */
RTS   /* Retour */

L$6:  .EQU  #0
```

de l'information vers les couches les plus hautes ou, en utilisant le langage des arbres, que la valeur du noeud terminal - et des autres - est envoyée à la racine. Concrétisons les choses (si possible)... Imaginons 5 poupées chinoises emboîtées dont la plus interne a la valeur 1 écrite dessus. Sur le couvercle de la première, on écrit :

```
fact(5) = 5 * ... = ...
sans savoir pour le moment par quoi remplacer les «...». Puis on retire soigneusement le couvercle pour faire apparaître la suivante sur laquelle on écrit :
```

```
fact(4) = 4 * ... = ...
Et ainsi de suite :
```

```
fact(3) = 3 * ... = ...
fact(2) = 2 * ... = ...
fact(1) = 1
Ayant obtenu l'information nécessaire, il ne reste plus qu'à
```

remettre les couvercles dans le bon ordre en remplissant les «...». Ce qui donne la suite d'opérations :

```
fact(2) = 2 * 1 = 2
fact(3) = 3 * 2 = 6
fact(4) = 4 * 6 = 24
fact(5) = 5 * 24 = 120
```

Bien entendu, le cas de la factorielle est particulièrement simple et se résout à l'aide d'une seule série de poupées. Mais pensez au calcul de la suite de Fibonacci... Les fanatiques de l'assemblage en trouveront un désassemblage commenté dans l'encadré 1, montrant comment un compilateur peut réagir face à la récursivité.

Tout n'est pas rose

Si la mise en oeuvre de la récursivité se révèle souvent nécessaire, puissante et facile à coder, on ne peut passer sous silence les problèmes qu'elle entraîne et qui doivent inciter à une utilisation prudente. Nous avons vu qu'un appel récursif se traduisait par une réservation d'espace sur la pile. Mais, outre la mémoire, il faut aussi penser que cette opération demande un temps certain. On peut en conclure qu'un algorithme récursif n'est pas, en général, le moyen le plus efficace d'arriver au but. Penchons nous un peu plus sur la suite de Fibonacci. De la relation :

$$f(n) = f(n-1) + f(n-2),$$

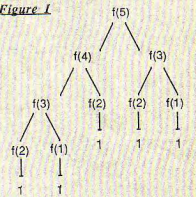
on tire en développant :

$$f(n) = [f(n-2) + f(n-3)] + [f(n-3) + f(n-4)]$$

et le fameux «ainsi de suite»... Deux remarques s'imposent :

- Il va falloir - inutilement! - recalculer plusieurs fois les mêmes valeurs (voir figure 1).
- Comme chaque calcul se subdivise en deux autres, la complexité de l'algorithme va être en $2^{(n-2)}$. C'est-à-dire que le temps d'exécution va croître exponentiellement en fonction de n. Inacceptable, alors qu'on dispose d'un algorithme itératif simple qui est en $O(n)$ et que voici :

Figure 1



L'arbre engendré par le calcul de fibo(5). Le parcours se fait en préordre.

```

fibonacci(n)
int n;
{
  int u1 = 1, u2 = 1, tampon, i;
  if (n < 3)
    return(1);
  for (i = 3; i <= n; i++)
  {
    tampon = u2;
    u2 += u1;
    u1 = tampon;
  }
  return(u2);
}

```

Moralité, ne pas sauter sur la première solution récursive venue! Si on s'aperçoit que la complexité spatiale ou temporelle du programme va poser des problèmes, il est bon d'essayer de trouver une solution simplement itérative... si elle

existe. Une question se pose: Peut-on toujours éliminer la récursivité? Disons qu'il est possible de l'éviter formellement en remplaçant l'appel récursif - et la sauvegarde de l'environnement dans la pile - par la sauvegarde des données importantes dans un ou plusieurs tableaux sur lesquels se promènent des pointeurs, correspondant au niveau d'imbrication atteint. Cette technique revient en fait à gérer manuellement la récursivité, comme l'ont fait des générations de programmeurs avec des langages ne leur proposant pas ce cadeau. Même combat pour celui qui ne s'est pas rendu compte de la récursivité sous-jacente de son problème et se retrouve dans la situation de

M. Jourdain... Il n'est pas évident que ce genre de solution fasse gagner beaucoup de temps à l'exécution, bien qu'on puisse ici sélectionner les variables à sauver. Quant au coût du développement... On ne peut donc vraiment parler d'élimination de récursivité que lorsqu'on a fait disparaître un ou plusieurs de ces tableaux - ou piles - Les moyens (excepté parfois... le bon sens) qui le permettent ne sont absolument pas immédiats et je préfère renvoyer à la bibliographie en fin d'article, où sont examinés à fond certains exemples non triviaux. Concluons avec quelques graphismes un peu plus «bandants» que la suite de Fibonacci et autres factorielles.

Des dessins récursifs

Les programmes proposés ont été écrits pour tourner sur NT en moyenne résolution (620x200). Il utilise tous le fichier header <graphset.h>, qui contient quelques fonctions graphiques rudimentaires obtenues à partir des appels VDI. Pour ceux qui voudraient s'en passer ou utiliser des moyens plus rapides comme les LINEA, voici le détail des routines :

- Le système des coordonnées est modifié, l'origine est en bas à gauche et les «y» croissent vers le haut.
- openw() ouvre une station de travail qu'il FAUT fermer à la fin avec :
- closew()
- clearw() efface l'écran.
- hidecur() cache la souris.
- showcur() montre la souris.
- origin(x, y) déplace l'origine de l'écran au point (x, y) et permet d'utiliser des coordonnées négatives.
- setcolor(c) fixe la couleur des points et droites.
- plot(x, y) affiche un point relativement au repère.
- move(x, y) positionne le pointeur graphique relativement au repère.
- draw(x, y) trace une droite entre le pointeur graphique (x0,y0) et le point (x, y).

<pre> #define clearw() v_clrwk(handle) #define closew() v_cisvwk(handle) #define showcur() v_show_c(handle,0) #define hidecur() v_hide_c(handle) #define HAUTEUR 199 int contr[12], lntin[128], ptsin[128], lntout[128], ptsout[128]; int handle, work_out[57], work_in[12]; int v_offx, v_offy, v_graphmat[10]; openw() { int i; for (i = 0; i < 10; work_in[i++] = 1); work_in[10] = 2; v_opnvwk(work_in, &handle, work_out); vsf_perimeter(handle, 1); vsf_interior(handle, 2); vsf_color(handle, 1); vswr_mode(handle, 1); vsf_style(handle, 1); } setcolor(x) int x; { x &= 3; vsm_color(handle,x); vsf_color(handle,x); } origin(x, y) int x, y; { </pre>	<pre> v_offx = x; v_offy = y; } move(x, y); } move(x, y) int x, y; { register int *ptr = v_graphmat; *ptr++ = x + v_offx; *ptr = HAUTEUR - (y + v_offy); } draw(x, y) int x, y; { register int *ptr = v_graphmat + 2; *ptr++ = x; *ptr = HAUTEUR - (y + v_offy); v_pline(handle, 2, v_graphmat); move(x, y); } plot(x, y) int x, y; { register int *ptr = v_graphmat + 2; *ptr++ = x + v_offx; *ptr = HAUTEUR - (y + v_offy); v_pmarker(handle, 1, ptr); } </pre>
--	---

Listing 2

Le fichier <graphset.h>

Une cercle bizarre

Le programme en encadré 3 s'appelle cercle. c. L'idée est de tracer un cercle de rayon r mais dont chaque point serait lui-même le centre d'un cercle de rayon inférieur. Et ainsi de suite... Pour aller suffisamment vite, il faut travailler en entiers et précalculer un tableau de sinus et de cosinus pour chaque point dont le nombre est fixé à 35 par cercle. Les valeurs trigonométriques sont multipliées par 256 afin de ne pas trop perdre de précision et on redivise ensuite pour le résultat final. Les appels récursifs cessent quand le rayon devient inférieur à 25,

c'est la condition d'arrêt. On remarque que les ordonnées sont divisées par 2 avant d'af-

ficher un point. C'est pour corriger le phénomène d'éirement vers haut dû à la moyen-

ne résolution (640x200). Il est facile de construire des variations sur ce thème : Par exemple, en remplaçant les cercles par des carrés, etc.

Cette...pathologie semble surgir des cauchemards d'un mathématicien à la cervelle dévastée par les x. Jugez plutôt :

- La longueur d'une partie quelconque de la courbe est infinie.
- La courbe est partout continue et nulle part dérivable.

```

Listing 3
Cercles.c

setcolor(2);
circle(0,0,80);
Crawcin();
showcur();
closew();
}

circle(x,y,r)
int x,y,r;
{
register int l,dx,dy,dr;
register int *ptrco,*ptrsi;

plo(x,(y>>1));
if(r<25)
return;
for(ptrco=co,ptrsi=si,l=0;l<NB;l++)
{
dr=(r*100)/150;
dx=(r*ptrco++)>>8;
dy=(r*ptrsi++)>>8;
circle(x+dx,y+dy,dr);
}
}

#include <math.h>
#include <osbind.h>
#include <graphset.h>

#define pi2 6.28
#define NB 35

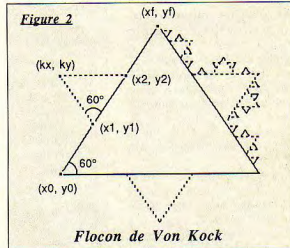
int si[NB+5], co[NB+5];

main()
{
double k,pas;
int l=0;

openw();
clearw();
origin(320,100); /* centre de l'écran */
hidecur();
pas=pi2/(double)NB;
for(k=0;k<pi2;k+=pas)
{
si[l]= (int)(256.0*sin(k));
co[l++]= (int)(256.0*cos(k));
}
}

```

Le flocon de Von Kock



```

Listing 4
Flocon de Von Kock

long lim = 200L;

main()
{
openw();
origin(0,0);
do {
clearw();
flocon();
lim /= 3L;
Crawcin();
} while (lim > 0L);
closew();
}

cote(x0,y0,xf,yf) /* trace un des cotés */
int x0,y0,xf,yf;
{
int x1,y1,x2,y2,nx,ny,dx,dy;
register long kx,ky,si=113511,co=65536;

dx=(xf-x0)/3;
dy=(yf-y0)/3;
if(dx*dx+dy*dy>lim*lim) /* pythagore */
{
x1=x0+dx;y1=y0+dy;
x2=xf-dx;y2=yf-dy;
kx=(co*dx-si*dy)>>17;
nx=x1+(int)kx;
ky=(si*dx+co*dy)>>17;
ny=y1+(int)ky;
/* Le sommet du nouveau triangle est trouvé par */
/* une rotation de 60 degrés */
cote(x0,y0,x1,y1);
cote(x1,y1,nx,ny);
cote(nx,ny,x2,y2);
cote(x2,y2,xf,yf);
/* Les appels récursifs sur les 4 nouveaux */
/* segments */
}
else
{
move(x0,y0>>1);
draw(xf,yf>>1);
/* Quand on a atteint la condition d'arrêt, il ne */
/* reste plus qu'à tracer */
}
}

flocon()
{
int x0,y0,x1,y1,x2,y2;

x0=100;y0=90;
x1=260;y1=365;
x2=420;y2=90;
cote(x0,y0,x1,y1);
cote(x1,y1,x2,y2);
cote(x2,y2,y0,y0);
/* un flocon comporte 3 côtés */
}
}

```

- La surface qu'elle englobe est finie.

Pour la construire, on utilise l'algorithme suivant (voir figure 2) :

- On prend un triangle équilatéral qu'on appelle «graine».
- Au centre de chacun des segments de droite ainsi définis, on fait germer un petit triangle équilatéral dont chaque côté fait le tiers du segment initial.
- Et ainsi de suite avec tous les segments de droite présents sur le dessin.
- La condition d'arrêt est donnée par la variable globale lim, qui représente la longueur minimale d'un côté.

Ces types de courbes s'appellent «fractales» et ont été étudiées sérieusement pour la première fois par Benoît Mandelbrot. Elles permettent de prendre en compte le côté «non euclidien» de certains phénomènes naturels (voir bibliographie). Compte tenu de la résolution d'un ordinateur, on ne peut en donner qu'une approximation gra-

phique. Mais cette dernière en montre bien le caractère intrinsèquement récuratif, qui n'est pas difficile à mettre en oeuvre quand on en a saisi le principe. Le seul problème consiste à établir les relations mathématiques permettant le passage de l'étape n à l'étape n+1.

La courbe de Péano

La présente courbe se programme de la même manière que le flocon. A partir de la graine donnée (voir figure 3), on cherche les coordonnées des points conduisant à l'étape suivante et ainsi de suite jusqu'à la condition d'arrêt. Notons que ce n'est pas la meilleure façon d'aller de A à B...

Les deux courbes suivantes sont très proches l'une de l'autre par leur construction (figures 4 et 5) qui consiste à décomposer chaque segment de façon simple. Par contre et au lieu de passer les coordonnées début et fin de chaque segment, on se contente d'en donner l'origine et d'indiquer longueur et direction. Au même titre que la courbe de Péano, elles donnent une bien mauvaise réponse au problè-

```
#include <osbind.h>
#include <graphset.h>

long lim = 250L, si = -20L, co = 40L;

main()
{
    nt l, x0 = 460, y0 = 60, x1 = 180, y1 = 60;

    openw();
    hidecur();
    clearw();
    selcolor(1);
    origin(0, 0);
    do
    {
        peano(x0, y0, x1, y1);
        lim /= 2L;
        Crawlinc();
        clearw();
    } while (lim > 1L);
    showcur();
    closew();
}

peano(x0, y0, xf, yf)
int x0, y0, xf, yf;
```

```
{
    int v1x, v1y, v2x, v2y, dx, dy, x1, y1, x2, y2, x3, y3, x4, y4;
    long kx, ky, ldx, ldy;

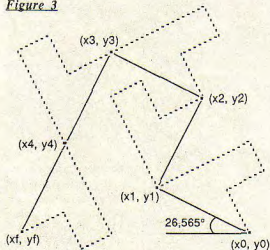
    ldx = (long) ( dx * xf - x0 );
    ldy = (long) ( dy * yf - y0 );
    if ( ldx*ldx + ldy*ldy > lim*lim )
    {
        kx = ( co*dx - si*dy )/100L; v1x = (int) kx;
        ky = ( si*dx + co*dy )/100L; v1y = (int) ky;
        x1 = x0 + v1x; y1 = y0 + v1y;
        v2x = v1y; v2y = -v1x;
        x2 = x1 + v2x; y2 = y1 + v2y;
        x3 = x2 + v1x; y3 = y2 + v1y;
        x4 = x1 + v1x; y4 = y1 + v1y;
        peano(x0, y0, x1, y1);
        peano(x2, y2, x1, y1);
        peano(x2, y2, x3, y3);
        peano(x3, y3, x4, y4);
        peano(xf, yf, x4, y4);
    }
    else {
        move(x0, y0 >> 1);
        draw(xf, yf >> 1);
    }
}
```

Listing 5
Courbe de Péano

me du plus court chemin d'un point à un autre...

Vous verrez, en exécutant ces petits programmes, que ces courbes fractales ont de curieuses propriétés. Par exemple, si on prélève une partie et qu'on la grossisse convenablement, on obtient

Figure 3



Les sinus et cosinus donnés dans le programme sont multipliés par 100 $\sqrt{5}$.

Courbe de péano

```
#include <osbind.h>
#include <graphset.h>

#define qrt 0.707106781
#define sqrt2 1.41421
#define LMIN 2.0

float x0, y0, length;

float tab[S][2] = {
    { 1.0, 0.0 }, { qrt, qrt }, { 0.0, 1.0 }, { -qrt, qrt },
    { -1.0, 0.0 }, { -qrt, -qrt }, { 0.0, -1.0 }, { qrt, -qrt }
}; /* Tableau de sinus et cosinus pour gagner de temps */

main()
{
    openw();
    origin(0, 0);
    length = 128.0;
    while (length >= LMIN)
    {
        clearw();
        move((int) (x0 = 450.0), (int) (y0 = 150.0));
        Curvef(128.0, 4);
        Crawlinc();
        length /= sqrt2;
    }
    length = 160.0;
    while (length >= LMIN)
    {
        clearw();
        move((int) (x0 = 170.0), (int) (y0 = 70.0));
        dragon(160.0, 0, 1);
    }
}
```

(Suite du listing page suivante)


```

Crawcin();
length /= sqrt2 ;
}
close();
}

dragon(len, ang, sign)
float len;
int ang, sign;
{
    float l;
    if ( len <= length )
    {
        ang &= 7;
        if ( ang <0 ) ang += 8;
        polaire(len, ang );
    }
    else
    {
        l=len/sqrt2;
        dragon( l, ang+sign, 1 );
        dragon( l, ang-sign, -1 );
    }
}

Ccurve(len, ang)
float len;
int ang;
{
    float l;
    if ( len <= length )
    {
        ang &= 7;
        if ( ang <0 ) ang += 8;
        polaire(len, ang );
    }
    else
    {
        l=len/sqrt2;
        Ccurve( l, ang+1 );
        Ccurve( l, ang-1 );
    }
}

/* Polaire trace un segment d'origine ( x0, y0 ), de longueur */
/* len et faisant un angle ang avec l'horizontale */
polaire( len, ang )
float len;
int ang;
{
    static int col;

    setcolor( ++col );
    if( col%4 == 3 ) setcolor( col++ );
    /* du fond */
    x0 += 2*tab[ang][0]*len;
    y0 += tab[ang][1]*len;
    draw( ( int ) x0, ( int ) y0 );
}

```

une courbe semblable au tout. Il est aussi possible de créer de nouveaux monstres en introduisant des valeurs aléatoires ou par tout autre moyen ! ET AINSI DE SUITE, JUSQU'À EPUISEMENT...

Jean-Yves Trétout

Bibliographie

Sur la récursivité :

«Les bases de la programmation» par J. Arsac chez DUNOD.

«Méthodes de programmation» par B. Meyer et C. Baudouin chez EYROLLES.

Au sujet des fractals :

«Les objets fractals» par B. Mandelbrot chez FLAMMARION.

«Fractals: Dimension non-entière et applications» sous la direction de G. Cherbit chez MASSON.

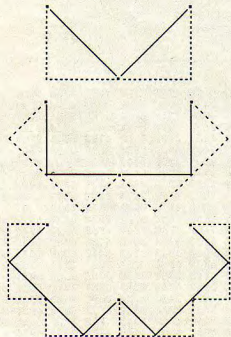
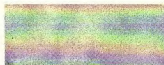


Figure 4 La C curve

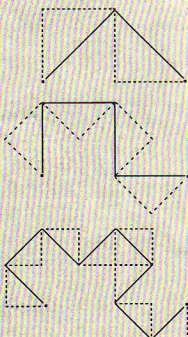


Figure 5 Le Dragon

Super copieur

SCOPY

*Possesseurs d'un seul drive,
soyez heureux! Fini le rituel grille-pain
de la copie en plusieurs passes via le
Disk RAM.*

S COPY (contraction de Super-Copy) apporte 2 éléments nouveaux dans la manière d'effectuer vos copies.

- Il traite des fichiers de n'importe quelle longueur. Au préalable, il permet d'apprécier la quantité de mémoire restante utilisable à cet effet.
- Il fait l'impasse sur le Disk RAM limité à 200 Ko. La copie s'effectue directement.

Utilisation

SCOPY se divise en 2 parties.

- la partie INFO où l'on accède en tapant SCOPY sous CLI sans y adjoindre le moindre paramètre. Après le copyright traditionnel, s'affiche un nombre hexadécimal précisant la quantité de mémoire CHIP disponible. Si les heureux possesseurs d'au moins 1 méga de

RAM n'ont guère l'usage de cet utilitaire, les 512K-istes en revanche, sont vivement intéressés par cette valeur. Tant que la longueur d'un fichier est inférieure à la valeur indiquée, SCOPY est utilisable. Viennent ensuite les précautions à prendre...

- la fonction EFFECTIVE, où l'option copie de SCOPY qui s'obtient par l'ajout d'un paramètre: le nom du fichier que vous désirez copier (Exemple, SCOPY d0:c/SetMap). A ce stade, précisons que le nom du drive utilisé doit obligatoirement être spécifié (ici «D0:»). Pour le reste, le nom du fichier

est on ne peut plus classique et les règles que vous connaissez déjà restent les mêmes. Un message apparaît ensuite vous demandant d'insérer la disquette source. Changez de disquette le cas échéant, puis pressez en connaissance de cause l'un des deux boutons de la souris.

- Le bouton droit sert à sortir du programme en cas de lancement erroné de SCOPY, ou d'erreur dans la saisie du nom du fichier.

- Le bouton gauche avertit l'Amiga que vous êtes prêt. SCOPY charge le fichier puis demande d'insérer la disquette

destination (celle où vous désirez copier votre précieux fichier).

Là encore, pressez l'un des 2 boutons après avoir inséré la disquette susdite qui doit contenir suffisamment d'espace libre pour héberger le fichier. La sauvegarde effectuée, retour au CLI comme par enchantement.

Sauvegarde

Les codes hexadécimaux du listing sont à entrer par l'utilitaire *Amiga Saisie* en vous reportant, bien sûr, à son mode d'emploi (Micro-Mag n°10). Longue en octets à spécifier: 1796.

Stéphane Rodriguez

```

00001:0000 03F3 0000 0000 0000 0002 0000 0000:03F5
00002:0000 0001 0000 AEEA 0000 0001 0000 03E9:B2D5
00003:0000 0191 5340 6704 6000 0072 2C78 0004:48C3
00004:7802 4EAE FF28 613A 43FA 95CC 4EAE FE68:B7EE
00005:23C0 0000 04C2 6700 0028 2C40 4EAE FFC4:0A5C
00006:6712 2200 243C 0000 0212 263C 0000 02F7:D7CB
00007:4EAE FFD0 2C78 0004 227A 047C 4EAE FE62:EF00
00008:4E75 41FA 0241 7207 E998 2400 0200 0000:14CF
00009:000F 0C00 000A 6D06 0600 0037 6004 0600:E55A
00010:0030 10C0 2002 51C9 FFE0 4E75 0C18 0020:DD48
00011:6604 51C8 FFF8 43FA 056A 41E8 FFFF 5340:964F
00012:12D8 51C8 FFFC 2C78 0004 43FA 054A 4EAE:290A
00013:F5E8 23C0 0000 04C2 67A6 2C40 4EAE FFC4:0942
00014:23C0 0000 05DE 2200 243C 0000 0441 263C:9A57
00015:0000 003C 4EAE FFD0 0039 000A 00DF F016:47F2
00016:6700 0134 0839 0006 05E0 E001 66EA 2C79:E496
00017:0000 04C2 223C 0000 05EE 243C FFFF FFFE:5125
00018:4EAE FFAC 23C0 0000 04C6 6700 019A 2C79:0B63
00019:0000 04C2 2200 243C 0000 04CA 4EAE FFA7:9E10
00020:4A80 6700 0002 41FA 0336 2028 097A 3C04:3C84
00021:0000 05DA 223C 0001 0002 2C78 0004 4EAE:A343
00022:FF3A 23C0 0000 05D6 6700 000C 2C79 0000:BD15
00023:04C2 223C 0000 05EE 243C 0000 03ED 4EAE:A3C3
00024:FFE2 23C0 0000 05CE 6700 0008 2200 2439:D741
00025:0000 05D6 2639 0000 65DA 4EAE FFD0 2239:A2A6
00026:0000 05CE 4EAE FFFC 2239 0000 0027:0000
00027:0000 047D 263C 0000 0043 4EAE FFD0 0839:81B3
00028:000A 00DF F016 675A 0839 0006 00BF E001:4158
00029:66EC 223C 0000 05EE 243C 0000 03EE 4EAE:05EE
00030:FFE2 23C0 0000 05D2 6700 0038 2239 0000:02E5
00031:05D2 2439 0000 05D6 2639 0000 05DA 4EAE:AA82
00032:FFD0 2239 0000 05D2 4EAE FFFC 2239 0000:989E
00033:05DE 243C 0000 04C0 263C 0000 0002 4EAE:A3C6
00034:FFD0 2279 0000 05D6 2039 0000 05DA 2C78:7AA8
00035:0004 4EAE FF2E 2C78 0004 2279 0000 04C2:8197
00036:4EAE FE62 4E75 0D0A 9B30 3B33 313B 3430:E45D
00037:6D41 6E6F 7468 6572 2070 6E77 6572 6675:1158
00038:6C20 746F 6F6C 2064 6F6E 6520 6279 209E:C801
00039:343B 3333 3B34 306D 416C 646F 2052 6573:FEAF
00040:6574 9B30 3B33 313B 3430 6D2F 9B31 3B33:EA05
00041:323B 3430 6D4D 4354 903B 3B33 313B 3430:52DA
00042:6D20 696E 2031 3939 392E 000A 0A41 7641:EDD2
00043:696C 6162 6C65 2043 6869 702D 4D65 6D6F:EAE9
00044:7279 3A20 2400 0000 0000 0000 0020 6279:3332
00045:7465 732E 0D0A 0A48 6E77 2074 6E20 7570:7363
00046:6520 7468 6520 5343 4E00 5F20 636E 6D4D:9B37
00047:6166 643A 0D0A 4A75 7374 2067 6976 6520:7F98
00048:7468 6520 6669 6C65 6E61 6D65 2C20 492E:1D6A
00049:652E 209F 373B 3331 3B34 306D 5343 4EAE:FE49
00050:5920 6466 303A 632F 5365 746D 6170 9B30:1561
00051:3B33 313B 3430 6D0D 0A48 6974 203C 5265:F408

```


AMIGA

PROGRAMMATION

00052:7475	726E	3E2C	7468	656E	2069	6E73	6572:F333	00083:0000	0000	0000	0000	0000	0000	0000	0000	0000:0000
00053:7420	7468	6520	9B33	3B33	333B	3430	6D73:F8EC	00084:0000	0000	0000	0000	0000	0000	0000	0000	0000:0000
00054:6F75	7263	6520	9B30	3B33	313B	3430	6D64:F02A	00085:0000	0000	0000	0000	0000	0000	0000	0000	0000:0000
00055:6973	6B20	7768	6963	6820	6375	7272	656E:58D3	00086:0000	0000	0000	0000	0000	0000	0000	0000	0000:0000
00056:746C	790D	0A63	6F6E	7461	696E	7320	7468:2CA1	00087:0000	0000	0000	0000	0000	0000	0000	0000	0000:0000
00057:6520	6669	6C65	2E50	7265	7373	206D	6F75:DBF8	00088:0000	0000	0000	0000	0000	0000	0000	0000	0000:0000
00058:7365	2D62	7574	746F	6E2C	2052	6967	6874:EB03	00089:0000	0000	0000	0000	0000	0000	0000	0000	0000:0000
00059:2062	7574	746F	6E20	746F	2071	7569	742E:F6DC	00090:0000	0000	0000	0000	0000	0000	0000	0000	0000:0000
00060:0D0A	596F	7572	2066	696C	6520	6973	206C:54BC	00091:0000	0000	0000	0000	0000	0000	0000	0000	0000:0000
00061:6F61	6465	642C	7468	656E	2069	6E73	6572:0616	00092:0000	0000	0000	0000	0000	0000	0000	0000	0000:0000
00062:7420	7468	6520	9B33	3B33	333B	3430	6D64:F8DD	00093:0000	0000	0000	0000	0000	0000	0000	0000	0000:0000
00063:6573	7469	6E61	7469	6F6E	9B30	3B33	313B:33B2	00094:0000	0000	0000	0000	0000	0000	0000	0000	0000:0000
00064:3430	6D20	6469	736B	2061	6E64	0D0A	646F:7962	00095:0000	0000	0000	0000	0000	0000	0000	0000	0000:0000
00065:2061	7320	6265	666F	7265	2E20	596F	7572:CBBB	00096:0000	0000	0000	0000	0000	0000	0000	0000	0000:0000
00066:2066	696C	6520	6973	206E	6F77	2063	6F70:781D	00097:0000	0000	0000	0000	646F	732E	6C69	6272	6172:07EA
00067:6965	6420	6F6E	2074	6865	2064	6973	6B2E:BAD1	00098:7900	0000	0000	0000	0000	0000	0000	0000	0000:7900
00068:0D0A	5761	726E	696E	673A	2064	6973	6B20:9C78	00099:0000	0000	0000	0000	0000	0000	0000	0000	0000:0000
00069:6E61	6D65	2028	6578	2E20	6466	303A	2920:4D46	00100:0000	0000	0000	0000	0000	0000	0000	0000	0000:0000
00070:6D75	7374	2062	6520	7370	6563	6966	6965:1209	00101:0000	0000	0000	0000	0000	0000	0000	0000	0000:0000
00071:642E	0D0A	0A49	6E73	6572	7420	9B33	3B33:F9EC	00102:0000	0000	0000	0000	0000	0000	0000	0000	0000:0000
00072:313B	3430	6D73	6E75	7263	6520	9B30	3B33:F039	00103:0000	7965	735F	636F	0000	03EC	0000	001F:543E	
00073:313B	3430	6D64	6973	6B20	616E	6420	7072:DD62	00104:0000	0000	0000	001E	0000	0032	0000	00A0:00F0	
00074:6573	7320	6D6F	7573	6520	6275	7474	6F6E:66EC	00105:0000	00AE	0000	00B6	0000	00DC	0000	00E2:0322	
00075:2E0D	0A49	6E73	6572	7420	9B33	3B33	313B:87FC	00106:0000	00F2	0000	00FC	0000	0104	0000	011C:040E	
00076:3430	6D64	6573	7469	6E61	7469	6F6E	209B:EE43	00107:0000	0130	0000	013A	0000	0140	0000	0150:04FA	
00077:303B	3331	3B34	306D	6469	736B	2061	6E64:35A6	00108:0000	015C	0000	0162	0000	016C	0000	0176:05A0	
00078:2070	7265	7373	206D	6F75	7365	2062	7574:9F65	00109:0000	017C	0000	01A0	0000	01B0	0000	01BA:0686	
00079:746F	6E2E	0D0A	0000	0000	0000	0000	0000:EFA7	00110:0000	01C0	0000	01C6	0000	01D0	0000	01DA:0730	
00080:0000	0000	0000	0000	0000	0000	0000	0000:0000	00111:0000	01E0	0000	01F0	0000	01F6	0000	0208:07CE	
00081:0000	0000	0000	0000	0000	0000	0000	0000:0000	00112:0000	0000	0000	03F2	0000	03EB	0000	0001:07DE	
00082:0000	0000	0000	0000	0000	0000	0000	0000:0000	00113:0000	03F2	0000	0000	0006	9808	0000	7068:0C68	

Du vrai cinéma

ANIMATIX

*Créer des sprites multicolores c'est bien,
les animer c'est mieux. Surtout par le
biais d'un outil à la hauteur.*

C et utilitaire en GFA 3.0 permet la création et l'animation de sprites. Le second programme est un exemple d'utilisation des fichiers préalablement créés. L'éditeur d'Animatix propose l'élaboration de séquences d'une quinzaine de sprites (16 x 16, 24 x 24 ou 32 x 32 pixels). L'emploi est aisé par l'utilisation de menus déroulants dont voici le détail.

Project

- **Save:** sauvegarde la case en cours.
- **Load:** chargement d'une case.
- **Save animation:** sauvegarde d'une animation. Exemple, 15 cases en une seule fois.
- Sélectionner «Animation select».
- Cliquer les 15 cases les unes après les autres.
- Sélectionner «Save animation».
- Entrer le nom du fichier («Essai» par exemple).
- Le programme sauvegardera alors les 15 cases sous les noms de «Essai1, Essai2,... Essais15».
- **Load animation:** charge une animation. Entrer le nom du fichier, plus le numéro 1. Pour recharger l'animation «Essai» de 1 à 15.
- Sélectionner «Load animation».

- Entrer «Essai», le programme chargera le reste.
N. B. : pour «Save animation» et «Load animation», on doit impérativement avoir sélectionné au moins une animation, sinon, il ne se passera rien.

- **Palette:** lors de la sélection de cette option:
 - Pour «Load animation», le programme chargera la palette de couleurs si elle existe.
 - Pour «Save animation», le programme sauvegardera la palette de couleurs.
- **Quit:** quitter le programme.

Utilité

- **Move:** déplace une case vers une autre. Après sélection, cliquer la case destination. La case destination est remplacée par la case actuelle qui s'efface.
- **Swap:** échange le contenu de 2 cases. Après sélection, cliquer la seconde case.
- **Copy:** duplique la case sur une autre. Après sélection, cliquer la case destination. Celle-ci est remplacée.
- **Inversion H:** inverse horizontalement le contenu de la case.

- **Inversion V:** inverse verticalement le contenu de la case.
- **Clear:** détruit la case actuelle.

Animation

- **Animation select:** sélection d'une animation. Sélectionner les cases les unes après les autres (le programme n'accepte que 20 animations maxi). Quitter cette fonction en cliquant sur le bouton droit de la souris. Chaque nouvelle animation efface la précédente.
- **Animation start:** active l'animation sur l'écran. Si aucune n'a été sélectionnée, il ne se passera rien. Quitter cette fonction en cliquant longuement sur le bouton gauche. Régler la vitesse d'animation en déplaçant la souris de gauche à droite.

Format

- **16 x 16:** format des cases par défaut, correspondant à 16 pixels sur 16.
- **24 x 24:** correspond à 24 pixels/24.
- **32 x 32:** correspond à 32 pixels/32. Le plus grand format du programme.

mat du programme.
• **Restore palette:** restaure les couleurs à leur état par défaut.

Divers

- Pour dessiner dans différentes cases, sélectionner la case en question.
- Pour chaque couleur sélectionnée, on peut modifier les niveaux de rouge, vert et bleu en cliquant dans les 3 cases réservées à cet effet.
- Le programme utilise les couleurs de 0 à 3. Ne pas les modifier afin d'éviter tout compromis avec l'affichage de présentation.
- Une petite case au-dessus de la palettisation indique:
 - L'attente (Wait).
 - Que l'on doit cliquer une autre case (Pick).
 - Le numéro de sélection d'animation.
- Les cases sont sauvegardées sous le format «Get» et «Put» avec, en début de fichier, différents paramètres (voir commentaires du second programme).
- Il est possible de charger sur un format supérieur, des cases sauvegardées sous un format inférieur, mais pas l'inverse.
- Les images sont sauvegardées sous 5 plans de bits 32 couleurs.

Daniel Provenier

```

. * indique l'endroit où
. * vous devez frapper Return.
. *
. * .....
. * ANIMATIX
. *
. * Writing by
. *
. * PROVENIER DANIEL
. *
. * LE 12/02/1990
. * .....
    
```

```

RESERVE 30000.
DEFWRD "a.z".
@init.
EDIT.
PROCEDURE principal.
FOR q=0 TO 2.
    @aff_palette(q).
NEXT q.
ON MENU GOSUB quel_menu.
REPEAT.
    ON MENU 0.1.
        MOUSE x,y,k.
    
```

```

IF k=1.
    @clique.
ENDIF.
UNTIL fini=1.
RETURN.
PROCEDURE init.
DIM men$(25),co(31,2),an$(20).
di=1.
OPENS 1,0,0,320,256,5,0.
OPENW 1,0,0,320,256,0,0,1.
CLEARW 1.
TITLEW #1,"".
    
```



```

FOR n=0 TO 25.
  READ men$(n).
NEXT n.
MENU men$(.).
@restore_co.
@init2.
@get.
cl$=lm$.
@principal.
pal=0.
valm%=&H52.
DATA Project,Save,Load,Save animation,L
oad animation,Palette,Quit,"".
DATA Utile,Move,Swap,Copy,Inversion,H,I
nversion,V,Clear,"".
DATA Anime,Animation select,Animation s
tart,"".
DATA Format,16*16,24*24,32*32,Restore p
alette,"".
palette:.
DATA 0,0,0.
DATA 7,7,7.
DATA 5,15,15.
DATA 15,5,5.
DATA 15,15,15.
DATA 11,11,11.
DATA 6,6,6.
DATA 2,2,2.
DATA 15,0,0.
DATA 15,4,0.
DATA 15,8,0.
DATA 14,12,0.
DATA 13,14,0.
DATA 8,14,0.
DATA 4,14,0.
DATA 0,14,0.
DATA 0,15,0.
DATA 0,15,4.
DATA 0,15,8.
DATA 0,14,12.
DATA 0,12,14.
DATA 0,8,14.
DATA 0,4,14.
DATA 0,0,14.
DATA 0,0,15.
DATA 4,0,15.
DATA 9,0,15.
DATA 13,0,15.
DATA 15,0,13.
DATA 15,0,9.
DATA 15,0,4.
DATA 15,7,15.
RETURN.
PROCEDURE init1.
ALERT 0,"ATTENTION|toutes les cases von
t etre|DETRUITES",1,"CONTINUE|OUBLIE !!".
.a.
IF a=1.
  di=di1.
  @init2.
ENDIF.
RETURN.
PROCEDURE init2.
DEFFILL 1,1,1.
PBOX 0,0,320,256.
@box(5,0,10,192,202).
@box(3,200,213,247,221).
@box(3,200,228,247,236).
@box(3,200,243,247,251).
@box(3,0,218,194,244).
@box(3,275,213,310,223).
COLOR 3.
FOR n=0 TO 192 STEP 12+(4*(di=2))+(6*(d
i=3)).
  LINE n,10,n,202.
  LINE 0,n+10,192,n+10.
NEXT n.
COLOR 3.
BOX 0,10,192,202.
di=(di-1)*8.
FOR n=0 TO 2.
  FOR m=0 TO 4.
    @box(3,200+(n*40),10+(m*40),216+(di
-1)*8+(n*40)+1,26+(m*40)+(di-1)*8+1).
  NEXT m.
NEXT n.
c=0.
FOR n=0 TO 1.
  FOR m=0 TO 15.
    DEFFILL c,1,1.
    PBOX m*12+1,n*12+200,m*12+11,n*12+
30.
    c=c+1.
    NEXT m.
  NEXT n.
  COLOR 1.
  c1=219.
  c2=231.
  c=0.
  BOX 0,219,12,231.
  COLOR 2,1.
  TEXT 253,221,"R".
  TEXT 253,236,"V".
  TEXT 253,251,"B".
  bt1=0.
  bt2=0.
  COLOR 3.
  BOX 200,10,216+(di-1)*8+1,26+(di-1)*8+1.
  .
  nor=12+(4*(di=2))+(6*(di=3)).
  n2=8+(8*di).
RETURN.
PROCEDURE box(noir,x1,y1,x2,y2).
DEFFILL 0,1,1.
PBOX x1+noir,y1+noir,x2+noir,y2+noir.
DEFFILL 0,1,1.
PBOX x1,y1,x2,y2.
DEFFILL 2.
BOX x1,y1,x2,y2.
RETURN.
PROCEDURE clique.
IF x<192 AND y<202 AND y>11.
  @point.
ELSE IF x<194 AND y>218 AND y<244.
  @choix_couleur.
ELSE IF x>200 AND y>10 AND y<200.
  @quel_boite.
  @restIt.
ELSE IF x>200 AND x<247 AND y>213 AND y
<251.
  @choix_palette.
ENDIF.
RETURN.
PROCEDURE point.

```

```

COLOR c.
x1=INT(x/nor)*nor.
y1=INT((y-10)/nor)*nor+10.
PBOX x1+1,y1+1,x1+nor-1,y1+nor-1.
PLOT 201+x1/nor+40*bt1,11+INT(y1-10)/no
r+40*bt2.
RETURN.
PROCEDURE choix_couleur.
COLOR 0.
BOX (c+(16*(c>15)))*12,c1,(c+(16*(c>15)
))*12+12,c2.
c=POINT(x,y).
IF c<16.
c1=219.
c2=231.
ELSE.
c1=231.
c2=243.
ENDIF.
COLOR 1.
BOX (c+(16*(c>15)))*12,c1,(c+(16*(c>15)
))*12+12,c2.
FOR q=0 TO 2.
@aff_palette(q).
NEXT q.
RETURN.
PROCEDURE quel_boite.
IF x<240.
bts1=0.
ELSE IF x<280.
bts1=1.
ELSE.
bts1=2.
ENDIF.
IF y<50.
bts2=0.
ELSE IF y<90.
bts2=1.
ELSE IF y<130.
bts2=2.
ELSE IF y<170.
bts2=3.
ELSE.
bts2=4.
ENDIF.
COLOR 2.
BOX 200+(bt1*40),10+(bt2*40),216+(di-1)
*8+(bt1*40)+1,26+(bt2*40)+(di-1)*8+1.
COLOR 3.
BOX 200+(bts1*40),10+(bts2*40),216+(di-1)
*8+(bts1*40)+1,26+(bts2*40)+(di-1)*8+1.
bts1=bts1.
bts2=bts2.
RETURN.
PROCEDURE restit.
COLOR 3,0.
TEXT 276,221,"WAIT".
FOR n=0 TO n2-1.
n1=n+201+(40*bt1).
n3=n*nor.
FOR m=0 TO n2-1.
cp=POINT(n1,m+11+(40*bt2)).
COLOR cp.
PBOX n3+1,(m*nor)+11,(n3+nor)-1,(m*
nor+nor)+9.
NEXT m.
NEXT n.
TEXT 276,221," ".
RETURN.
PROCEDURE quel_menu.
ON MENU(0) GOSUB save,load,save_an,load
_an,pale,quit,rien,rien,move,swap,copy,i
nverh,inverv,clear,rien,rien,ansel,ansta
rt,rien,rien,dim1,dim2,dim3,restore_co.
RETURN.
PROCEDURE quit.
fini=1.
RETURN.
PROCEDURE move.
COLOR 3,0.
TEXT 276,221,"PICK".
@get.
ci$=im$.
im$=ci$.
test=0.
k=0.
REPEAT.
WHILE k<>1.
MOUSE x,y,k.
WEND.
IF x>200 AND y>10 AND y<200.
@put.
im$=ci$.
@quel_boite.
test=I.
ENDIF.
UNTIL test=1.
@put.
@relache.
COLOR 3,0.
TEXT 276,221," ".
RETURN.
PROCEDURE swap.
COLOR 3,0.
TEXT 276,221,"PICK".
@get.
ci$=im$.
test=0.
k=0.
REPEAT.
WHILE k<>1.
MOUSE x,y,k.
WEND.
IF x>200 AND y>10 AND y<200.
bt11=bt1.
bt22=bt2.
@quel_boite.
@get.
SWAP bt1,bt11.
SWAP bt2,bt22.
@put.
SWAP bt1,bt11.
SWAP bt2,bt22.
im$=ci$.
@put.
test=1.
ENDIF.
UNTIL test=1.
@relache.
COLOR 3,0.
TEXT 276,221," ".
RETURN.
PROCEDURE copy.

```



```

COLOR 3,0.
TEXT 276,221,"PICK".
@get.
test=0.
k=0.
REPEAT.
  WHILE k<>1.
    MOUSE x,y,k.
  WEND.
  IF x>200 AND y>10 AND y<200.
    @quel_boite.
    @put.
    test=1.
  ENDIF.
UNTIL test=1.
@relache.
COLOR 3,0.
TEXT 276,221,"  ".
RETURN.
PROCEDURE clear.
  ALERT 0,"ATTENTION|cette case va etre|d
  etruite",1,"CONTINUE|OUBLIE !!",a.
  IF a=1.
    im$=cl$.
    @put.
    @restit.
  ENDIF.
RETURN.
PROCEDURE inverh.
  n3=40*bt2+11.
  FOR n=0 TO n2-1.
    n1=n+201+(40*bt1).
    FOR m=0 TO (n2-1)/2.
      cp1=POINT(n1,m+n3).
      cp2=POINT(n1,n3+n2-m-1).
      COLOR cp1.
      PLOT n1,n3+n2-m-1.
      COLOR cp2.
      PLOT n1,m+n3.
    NEXT m.
  NEXT n.
  @restit.
  COLOR c.
RETURN.
PROCEDURE inverv.
  n3=40*bt2+11.
  FOR n=0 TO (n2-1)/2.
    n1=n+201+(40*bt1).
    FOR m=0 TO n2-1.
      cp1=POINT(n2+200+(40*bt1)-n,m+n3).
      cp2=POINT(n1,m+n3).
      COLOR cp1.
      PLOT n1,m+n3.
      COLOR cp2.
      PLOT n2+200+(40*bt1)-n,m+n3.
    NEXT m.
  NEXT n.
  @restit.
  COLOR c.
RETURN.
PROCEDURE ansel.
  ja=0.
  bt1=bt1.
  bt2=bt2.
  WHILE MOUSEK<>2.
    MOUSE x,y,k.

```

```

  IF k=1.
    IF x>200 AND y>10 AND y<200 AND ja<
20.
    @quel_boite.
    @get.
    an$(ja)=im$.
    INC ja.
    @relache.
    COLOR 3,0.
    TEXT 285,221,STR$(ja).
  ENDIF.
ENDIF.
WEND.
TEXT 285,221,"  ".
COLOR 2.
BOX 200+(bt1*40),10+(bt2*40),216+(di-1)
*8+(bt1*40)+1,26+(bt2*40)+(di-1)*8+1.
COLOR 3.
bt1=bt1.
bt2=bt2.
BOX 200+(bt1*40),10+(bt2*40),216+(di-1)
*8+(bt1*40)+1,26+(bt2*40)+(di-1)*8+1.
RETURN.
PROCEDURE anstart.
  IF ja<>0.
    GET 76,76,116,116,a$.
    @box(3,76,76,109,109).
    WHILE MOUSEK<>1.
      FOR n=0 TO ja-1.
        FOR tt=0 TO MOUSEX*10.
          NEXT tt.
          PUT 77,77,an$(n).
        NEXT n.
      WEND.
      PUT 76,76,a$.
    ENDIF.
  RETURN.
PROCEDURE get.
  GET 201+(bt1*40),11+(bt2*40),216+(di-1)
*8+(bt1*40),26+(bt2*40)+(di-1)*8,im$.
  RETURN.
PROCEDURE put.
  PUT 201+(bt1*40),11+(bt2*40),im$.
  RETURN.
PROCEDURE relache.
  WHILE MOUSEK.
  WEND.
  FOR j=0 TO 500.
    NEXT j.
  WHILE MOUSEK.
  WEND.
  RETURN.
PROCEDURE dim1.
  IF di<>1.
    di1=1.
    @init1.
  ENDIF.
  RETURN.
PROCEDURE dim2.
  IF di<>2.
    di1=2.
    @init1.
  ENDIF.
  RETURN.
PROCEDURE dim3.
  IF di<>3.

```

AMIGA

PROGRAMMATION

```

di1=3.
@init1.
ENDIF.
RETURN.
PROCEDURE restore_co.
RESTORE palette.
FOR n=0 TO 31.
  FOR m=0 TO 2.
    READ co(n,m).
  NEXT m.
  SETCOLOR n,co(n,0),co(n,1),co(n,2).
NEXT n.
FOR q=0 TO 2.
  @aff_palette(q).
NEXT q.
RETURN.
PROCEDURE aff_palette(q).
SELECT q.
CASE 0.
  @box(3,200,213,247,221).
CASE 1.
  @box(3,200,228,247,236).
CASE 2.
  @box(3,200,243,247,251).
ENDSELECT.
COLOR 3.
IF co(c,q).
  PBOX 201,214+15*q,201+3*co(c,q),214+
5*q+6.
ENDIF.
COLOR c.
RETURN.
PROCEDURE choix_palette.
x1=FIX((x-201)/73).
y1=y-214.
IF y1>=0 AND y1<8.
  IF co(c,0)<>x1.
    co(c,0)=x1.
    @aff_palette(0).
  ENDIF.
ELSE IF y1>=15 AND y1<23.
  IF co(c,1)<>x1.
    co(c,1)=x1.
    @aff_palette(1).
  ENDIF.
ELSE IF y1>=30 AND y1<38.
  IF co(c,2)<>x1.
    co(c,2)=x1.
    @aff_palette(2).
  ENDIF.
ENDIF.
SETCOLOR c,co(c,0),co(c,1),co(c,2).
RETURN.
PROCEDURE save.
@get.
FILESELECT "SAUVE CASE","SAUVER","DF0:"
,nom$.
IF nom$<>"".
  a=LEN(im$).
  OPEN "O",#1,nom$.
  PRINT #1,"ANIMATE".
  PRINT #1,a.
  PRINT #1,di.
  FOR n=1 TO a.
    b=ASC(MID$(im$,n,1)).
    OUT #1,b.

```

```

NEXT n.
CLOSE #1.
ENDIF.
RETURN.
PROCEDURE load.
FILESELECT "LOAD CASE","CHARGER","DF0:"
,nom$.
IF nom$<>"".
  OPEN "I",#1,nom$.
  INPUT #1,sor$.
  IF sor$="ANIMATE".
    INPUT #1,a.
    INPUT #1,di1.
    IF di1<di.
      im$="".
      FOR n=1 TO a.
        b=INP(#1).
        im$=im$+CHR$(b).
      NEXT n.
    ELSE.
      ALERT 0,"CETTE CASE|A ETE SAUVEGA
RDEE|SOUS UN FORMAT|SUPERIEUR",1,"OK",
a.
    ENDIF.
  ENDIF.
  CLOSE.
  @put.
  ENDIF.
RETURN.
PROCEDURE load_an.
FILESELECT "LOAD ANIMATION","CHARGER","
DF0:",nom$.
IF nom$<>"" AND RIGHT$(nom$,1)="1".
  nom$=MID$(nom$,1,LEN(nom$)-1).
  bts1=bt1.
  bts2=bt2.
  FOR bt1=0 TO 2.
    FOR bt2=0 TO 4.
      IF EXIST(nom$+STR$(bt2+(5*bt1)+1)
).
        OPEN "I",#1,nom$+STR$(bt2+(5*bt
1)+1).
        INPUT #1,sor$.
        IF sor$="ANIMATE".
          INPUT #1,a.
          INPUT #1,di1.
          IF di1<di.
            im$="".
            FOR n=1 TO a.
              b=INP(#1).
              im$=im$+CHR$(b).
            NEXT n.
          ELSE.
            ALERT 0,"CETTE CASE|A ETE S
AUEVGARDEE|SOUS UN FORMAT|SUPERIEUR",1,"
OK",a.
          ENDIF.
        ENDIF.
        CLOSE.
        @put.
        ENDIF.
        NEXT bt2.
        NEXT bt1.
        IF pal=1.
          IF EXIST(nom$+".PALETTE").
            OPEN "I",#1,nom$+".PALETTE".

```


PROGRAMMATION

```

FOR n=0 TO 31.
  INPUT #1,co(n,0);co(n,1);co(n,2
),
  SETCOLOR n,co(n,0),co(n,1),co(n
),2).
  NEXT n.
  CLOSE #1.
  ENDIF.
  ENDIF.
  bt1=bts1.
  bt2=bts2.
  @restit.
  ENDIF.
  RETURN.
PROCEDURE save_an.
  IF ja<>0.
    FILESELECT "SAVE ANIMATION","SAUVER",
"DF0:",nom$.
    IF nom$<>"".
      IF RIGHTS(nom$,1)="1".
        nom%=MID$(nom$,1,LEN(nom$)-1).
      ENDIF.
      FOR m=0 TO ja-1.
        a=LEN(an$(m)).
        OPEN "O",#1,nom$+STR$(m+1).
        PRINT #1,"ANIMATE".
        PRINT #1,a.
      PRINT #1,di.
      FOR n=1 TO a.
        FOR n=1 TO a.
          b=ASC(MID$(an$(m),n,1)).
          OUT #1,b.
        NEXT n.
      CLOSE #1.
    NEXT m.
  IF pal=1.
    OPEN "O",#1,nom$+".PALETTE".
    FOR n=0 TO 31.
      WRITE#1,co(n,0),co(n,1),co(n,2)
    NEXT n.
  CLOSE #1.
  ENDIF.
  ENDIF.
  ENDIF.
  RETURN.
PROCEDURE pale.
  pal=pal XOR 1.
  IF pal=1.
    valm%=valm% OR &H151.
  ELSE IF pal=0.
    valm%=valm% AND &HFE.
  ENDIF.
  MENU (5),valm%.
  RETURN.
' . indique l'endroit où
' vous devez frapper Return.
RESERVE 10000.
DEFWRD "a,z".
@init.
@load_an.
@animation_start.
CLOSES 1.
EDIT.
PROCEDURE init.
  DIM an$(20).
  OPENS 1,0,0,320,256,5,0.
  OPENW 1,0,0,320,256,0,0,1.
  CLEARW 1.
  TITLEW #1,"".
  RETURN.
PROCEDURE animation_start.
  IF ja<>0.
    WHILE MOUSEK=0.
      FOR n=0 TO ja-1.
        FOR tt=0 TO MOUSEX*5 !--->VARIATI
ON DE LA VITESSE AVEC LA SOURIS.
          NEXT tt.
          PUT 77,77,an$(n).
        NEXT n.
      WEND.
      CLEARW 1.
    ENDIF.
  RETURN.
PROCEDURE load_an.
  FILESELECT "LOAD ANIMATION","CHARGER",
"DF0:",nom$.
  IF nom$<>"" AND RIGHTS(nom$,1)="1" | RE
NTRE LE NOM DE LA 1 er IMAGE.
    nom%=MID$(nom$,1,LEN(nom$)-1) | LE
PROGRAMME CHARGERÀ LE RESTE.
    FOR m=0 TO 14.
      IF EXIST(nom$+STR$(m+1)).
        OPEN "I",#1,nom$+STR$(m+1).
        INPUT #1,sor$ | -----> VERIF
ICATION SI BON FICHIER.
        IF sor$="ANIMATE" | -----> LE PR
EMIER CODE EST "ANIMATE".
          INPUT #1,a | -----> LONGU
EUR DU FICHIER.
          INPUT #1,di1 | -----> TEST
POUR LE FORMAT D'IMAGE.
          1=
          16*16.
          2=
          24*24.
          3=
          32*32.
          FOR n=1 TO a | -----> CHARG
EMENT DE L'IMAGE.
            b=INP(#1).
            an$(ja)=an$(ja)+CHR$(b).
          NEXT n.
          ja=ja+1 | --->NOMBRE
D'IMAGES.
        ENDIF.
      CLOSE #1.
    ENDIF.
  NEXT m.
  IF EXIST(nom$+".PALETTE") | ---> VER
IFICATION POUR SAVOIR SI LA PALETTE.
  OPEN "I",#1,nom$+".PALETTE" | ES
T PRESENTE SUR LA DISQUETTE.
  FOR n=0 TO 31.
    INPUT #1,c1,c2,c3.
    SETCOLOR n,c1,c2,c3.
  NEXT n.
  CLOSE #1.
  ENDIF.
  ENDIF.
  RETURN.

```

Plein gaz!

UNE MANETTE DES GAZ SUR PC

Le principe du montage est simple: Une carte Joystick possède quatre ports. Deux sont utilisés dans un joystick classique (coord. X et Y), et donc ... deux sont inutilisés! Deux? non, un est réservé par certains jeux pour contrôler la vitesse. Lesquels? Ceux compatibles Maxx™ (un manche à balai superbe ... mais cher et qui possède un contrôle des gaz) soit, entre autres, *Flight Simulator III™*, *Chuck Yeager's advanced flight trainer™*, *Silent Service™*, etc.

La Carte Joystick

Pour brancher une manette de jeux, il faut utiliser une carte adaptateur, laquelle autorise quatre entrées analogiques (c'est-à-dire pouvant prendre des valeurs de 0 à n) et deux entrées de type «tout ou rien». En théorie, cette carte est prévue pour deux joysticks, chacun devant donc utiliser deux entrées analogiques, ainsi qu'une entrée numérique. En fait, un joystick utilise deux boutons de mise à feu et occupe de ce fait les deux entrées «tout ou rien». Le principe de mesure des entrées analogiques est le suivant: aux bornes de chacune d'elles se trouve un potentiomètre (résistance variable). Suivant la position de ce dernier, la résistance varie et est évaluée par une temporisation plus ou moins longue mesurée à tout instant par le jeu.

Dingues des simulateurs de vol et mordus de jeux vidéo, ce montage est fait pour vous. Modique (60 F environ) et facile à réaliser, il permet le contrôle aisé de l'engin que vous pilotez. Condition sine qua non: que le jeu prenne en compte ce périphérique supplémentaire.

Principe

Il suffit d'intercaler un potentiomètre entre les bornes de la troisième entrée analogique et le tour est joué! Mais voyons d'abord les différentes cartes Joystick. En fait, deux types de cartes: les cartes à port et les cartes deux ports qui permettent le branchement d'un second joystick.

Il est beaucoup plus facile d'exécuter le montage sur un carte deux ports. En effet, le premier étant pris par le joystick, le montage se fera sur le deuxième, ce qui est plus simple à réaliser.

Si l'on regarde le premier schéma (Carte joystick 1 port), on peut noter qu'il y a une rangée de broches réservée pour le joystick 1 et une autre pour le joystick 2. Oui, mais si l'on peut brancher un joystick (semblable aux autres) sur le second port, c'est que le brochage est inversé (d'où le second schéma)

Réalisation

Sans danger pour l'ordinateur, elle ne pose aucun problème pour quiconque n'est pas trop maladroit. Toutefois, il est conseillé d'utiliser un fer à souder de faible puissance (inférieure à 50 W) et de la soudure électronique à 60%.

D'abord, il faut souder les deux brins (fils) du câble sur les broches 1 et 6 de la prise db15. Inutile de compter, les broches sont numérotées. Si possible, utiliser un étai pour fixer le connecteur. Sinon, faites tenir la prise par un copain obligeant armé d'une paire de pinces isolantes (gaffe aux brûlures!).

A l'autre extrémité du câble «scotchez» les brins sur deux pattes côte à côte du potentiomètre. Branchez et essayez (sur un jeu qui accepte). Ca marche? Alors soudez le potentiomètre. Dans le cas contraire, vérifiez le montage et assurez-vous que la manette est au zéro (Sinon, *Chuck Yeager*, par exemple, ne la

reconnaitra pas). De toute façon, ÇA DOIT MARCHER!!! Il ne reste plus qu'à loger votre manette dans un boîtier (ou la fixer au clavier), puis jouer, enfin.

Facile non? Hélas, trop peu de jeux exploitent cette possibilité. Imaginez une bourre sur *Falcon™* avec ce genre d'engin... A vous les manœuvres *Top Gun ...* Quand *Spectrum Holobyte* daignera le prévoir! De plus, il reste le quatrième port. Il n'existe pas de norme établie, mais *Flight Simulator 3* et *4* utilisent ledit port pour le palonnier (les pédales qui doivent être conjuguées avec le manche)... Fanas de réalisme et bidouilleurs de tous poils, vous savez ce qu'il vous reste à faire.

Jean-François Six

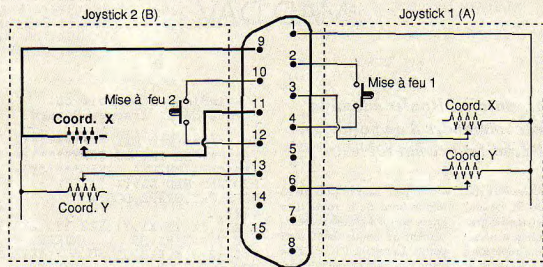
Liste du matériel

- 1m 50 de câble deux brins.
- 1 potentiomètre droit linéaire variant de 0 à 100 kilohms (kΩ).
- 1 connecteur mâle DB 15 (15 broches).

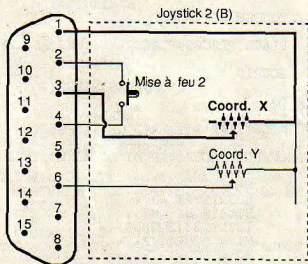
Accessoirement :

- 1 capot pour DB15
- 1 boîtier pour le potentiomètre.

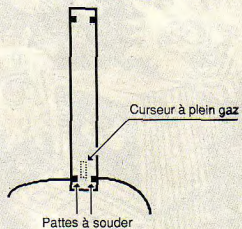
Carte joystick 1 port



Carte Joy. 2 ports : 2nd port



Montage du potentiomètre
(Vue de dessous)



Sus aux bagnoles!

MAD DAV

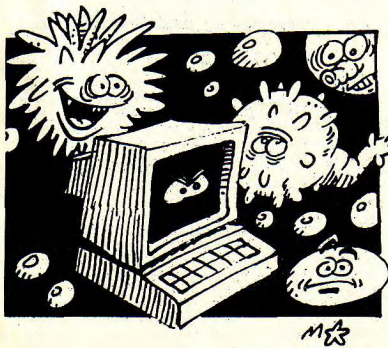
Excellent exutoire! Rouler à fond la caisse aux commandes d'un bolide, en canardant tout ce qui roule...

Programmé en Turbo Pascal (4 ou 5), ce jeu, tournant en mode graphique CGA minimum, nécessite dans le même répertoire les fichiers CGA.BGI, GOTH.CHR, LITT.CHR et GRAPH.TPU.

Il se joue au joystick (ne choisir cette option qu'en possession de la carte sous peine de plantage), à la souris ou avec les chiffres. Les indications nécessaires sont incluses.

On ne peut heurter plus de 10 fois le bord de la route. Les forces de 1 à 4 offrent respectivement un bonus de 10 à 25 points. La touche ESC, comme à son habitude, permet de sortir.

David Farenzena



* indique l'endroit où vous devez frapper Return.

```

{***** MAD DAV *****}.
{*****EN TP4 OU 5*****}.
{*****FARENZENA DAVID*****}.
PROGRAM MAD DAV;
USES CRT, GRAPH, DOS;
VAR
  X, Y, XX, YY, X1, Y1, XX2, YY2, XX3, XX4, YY3, YY
4, U, V, A, B, B1, B2 : INTEGER;
  I, I2, J, K, L, M, J1, SC, SC1, KL, KL1, KL2, KL3,
SCT, SCO, SCOR : INTEGER;
  O, O1, O2, O4, O5, P, Q, R, S, T, T1, LEN, DRI, MOD
E, COMPT, MUST : INTEGER;
  TAILLE, TAILLE2, TAILLE3, TV1, TV2, TV3, TV4
, FORCE
HEU, MIN, SEC, CEN, JOY1, BO, SC2, SC3
: WORD;
CO, COI : ARCCOORDSTYPE;
ZA, ZE : BOOLEAN;
L_, T_, V_, R_ : STRING[3];
MIN_, SEC_, NOM, SC1_, KL : STRING[10];
TOUCHE, JOY, FORCE : CHAR;
PLACE, PLACE2, PLACE3, PF1, PF2, PF3, PF4 : POINTER;
SOUSIS : REGISTERS;
O3 : REAL;
SCORT: ARRAY[1..40] OF INTEGER;
)-----).
PROCEDURE SOUR(VAR M1, M2, M3, M4: INTEGER);
BEGIN
  SOURIS.AX:=3;
  SOURIS.BX:=M2;
  INTR($33, SOURIS);
  M1:= SOURIS.AX;
  M2:= SOURIS.BX;
  M3:= SOURIS.CX;
  M4:= SOURIS.DX;
END;
)-----).
PROCEDURE STICK(VAR X, Y, X1, X2: INTEGER);

```



```

BEGIN=
INLINE=
($BA/$01/$02/SEC/$A8/$03/$75/$FB/$B
9/$2C/$01/$FA/SEE/SEC/$A8/$01/.
$E0/$FB/$BB/$2C/$01/$2B/$D9/$C4/$BE
/X/$26/$89/$1D/$2B/$DB/$A8/$10/.
$75/$03/$83/$C3/$01/$C4/$BE/K1/$26/
$89/$1D/$2B/$DB/$A8/$20/$75/$03/.
$83/$C3/$01/$C4/$BE/K2/$26/$89/$1D/
SEC/$A8/$03/$75/$FB/$B9/$2C/$01/.
$EE/SEC/$A8/$02/$E0/$FB/$FB/$BB/$2C
/$01/$2B/$D9/$C4/$BE/Y/.
$26/$89/$1D);=
END;=
{-----}.
PROCEDURE CADRE;=
VAR I: INTEGER;=
BEGIN=
GOTOXY(1,1);WRITE(CHR(201));=
FOR I:=2 TO 79 DO=
BEGIN=
GOTOXY(I,1);WRITE(CHR(205));=
END;=
GOTOXY(80,1);WRITE(CHR(187));=
FOR I:=2 TO 23 DO=
BEGIN=
GOTOXY(80,I);WRITE(CHR(186));=
END;=
GOTOXY(80,24);WRITE(CHR(188));=
FOR I:=2 TO 79 DO=
BEGIN=
GOTOXY(I,24);WRITE(CHR(205));=
END;=
GOTOXY(1,24);WRITE(CHR(200));=
FOR I:=2 TO 23 DO=
BEGIN=
GOTOXY(1,I);WRITE(CHR(186));=
END;=
END;=
{-----}.
PROCEDURE ENTREE;=
BEGIN=
TEXTCOLOR(1);=
CADRE;=
TEXTCOLOR(4);=
GOTOXY(36,1);WRITE('MAD DAV');=
GOTOXY(35,2);WRITE('=====');=
TEXTCOLOR(2);=
GOTOXY(4,4);WRITE('VOUS ETES L' AIGLE D
E LA ROUTE DETERMINE A ROULER LE PLUS VI
TE');=
GOTOXY(4,5);WRITE('POSSIBLE ET A PARCOU
RIR LE PLUS GRAND NOMBRE DE KILOMETRES.'
);=
GOTOXY(4,6);WRITE('POUR CELA VOUS ETES
PRET A DETRUIRE TOUT ENNEMI VENANT EN SE
NS INVERSE.');
```

```

GOTOXY(4,11);WRITE('AVEC UN JOYSTICK OU
A LA SOURIS : (UN DES BOUTONS) POUR TIRE
R.');
```

```

GOTOXY(4,12);WRITE('LA MANETTE OU MA S
OURIS VERS SOI) POUR FREINER.');
```

```

GOTOXY(4,13);WRITE('AVEC LES CHIFFRES:(
7 OU 9) POUR TIRER.(8) POUR AVANCER.(2)
POUR FREINER.');
```

```

GOTOXY(4,14);WRITE('(6) A DROITE.(4) A
GAUCHE.(5) RETOUR AU MILIEU.');
```

```

TEXTCOLOR(3);=
GOTOXY(16,17);WRITE('VOTRE FORCE (1/4):
');
```

```

READLN(FORCE);=
CASE FORCE OF=
'1':FORCE:=25;=
'2':FORCE:=20;=
'3':FORCE:=15;=
'4':FORCE:=10;=
ELSE BEGIN FORCE_:=4;FORCE:=10;END;=
END;=
GOTOXY(30,17);WRITE(' ');=
GOTOXY(16,17);WRITE('VOTRE PRENOM
');
```

```

READLN(NOM);=
GOTOXY(16,17);WRITE('
');
```

```

LEN:=(LENGTH(NOM)) DIV 2;=
GOTOXY(30-LEN,17);WRITE('CHER(E)',NOM,
' AVEZ VOUS ');=
GOTOXY(30-LEN,18);WRITE('UN JOYSTICK
:(1)');
```

```

GOTOXY(30-LEN,19);WRITE('UNE SOURIS
:(2)');
```

```

GOTOXY(30-LEN,20);WRITE('LES CHIFFRES
:(3)');
```

```

TEXTCOLOR(4);=
GOTOXY(30-LEN,21);WRITE('-----
-----');
```

```

GOTOXY(30-LEN,22);WRITE('VOTRE REPONSE
: ');=
READLN(JOY);=
CASE JOY OF=
'1':JOY1:=1;=
'2':JOY1:=2;=
'3':JOY1:=3;=
ELSE JOY1:=3;=
END;=
TEXTCOLOR(6);=
GOTOXY(29-LEN,24);WRITE(NOM,' FRAPPEZ
(ENTER)');
```

```

READLN;=
END;=
{-----}.
PROCEDURE SORTIE;=
VAR J:STRING;I,CRA:INTEGER;=
BEGIN=
INC(COMPT);=
CASE FORCE OF=
'1':CRA:=10;=
'2':CRA:=15;=
'3':CRA:=20;=
'4':CRA:=25;=
END;=
SCO:=(SC*10);=
SCOR:=KL+(CRA-SCT);=
```

PROGRAMMATION

```

SCORT[COMPT]:=SCOR:
(***SYSTEME DE TRI***)
FOR I:=1 TO COMPT DO
BEGIN
IF SCORT[I]>SCORT[COMPT] THEN MUST:=SC
ORT[I]:
END:
IF MUST<SCOR THEN MUST:=SCOR:
CASE JOY1 OF
1:J:='UN JOYSTICK':
2:J:='UNE SOURIS':
3:J:='LES FLECHES':
END:
CLRSR:
CADRE:
GOTOXY(17-LEN,4):
WRITE('FORCE=' ,FORCE_ , ' SOIT UN BONUS
DE ' ,CRA_ , ' POINTS.'):
GOTOXY(17-LEN,6):
WRITE('MEILLEUR SCORE=' ,MUST_ , ' POINTS.
'):
GOTOXY(17-LEN,8):
WRITE('VOTRE SCORE=' ,SCOR_ , ' POINTS.
'):
TEXTCOLOR(2):
GOTOXY(36,1):WRITE('MAD DAV'):
GOTOXY(35,2):WRITE('====='):
GOTOXY(17-LEN,10):
WRITE('CHER(E) ' ,NOM_ , ' VOUS AVEZ PARCOU
RU ' ,XL_ , ' KILOMETRES ' ,SCO_ , ' METRES'):
GOTOXY(17-LEN,12):
WRITE('EN COGNANT ' ,SCT_ , ' FOIS LA ROUTE
,VOUS AVEZ TUE ' ,KL_ , ' ENNEMIS.'):
GOTOXY(17-LEN,14):
WRITE('ET TOUT CELA AVEC ' ,J_ , ' SALUT AI
GLE DE LA ROUTE.'):
TEXTCOLOR(4):
GOTOXY(30-LEN,23):WRITE('UNE AUTRE ' ,NO
M_ , ' (O/N)'):
READLN(TOUCHE):
END:
(-----).
PROCEDURE DESSIN_MOB:
BEGIN
(---DESSINS DES MOBILES---).
(---ENNEMI1---).
SETFILLSTYLE(1,2):
SETCOLOR(2):
RECTANGLE(158,30,162,36):RECTANGLE(159,
31,161,33):
SETCOLOR(3):
RECTANGLE(155,33,157,37):RECTANGLE(163,
33,165,37):
SETCOLOR(1):PUTPIXEL(159,35,1):PUTPIXEL
(161,35,1):
PUTPIXEL(157,32,1):PUTPIXEL(163,32,1):
TV1:=IMAGESIZE(155,30,165,37):
GETMEM(PF1,TV1):
GETIMAGE(155,30,165,37,PF1^):
PUTIMAGE(155,30,PF1^,1):
(---ENNEMI2---).
SETCOLOR(2):
RECTANGLE(154,40,166,58):RECTANGLE(157,
43,163,49):
FLOODFILL(160,53,2):
SETCOLOR(3):
RECTANGLE(145,52,153,61):RECTANGLE(167,
52,175,61):
SETCOLOR(1):CIRCLE(157,55,2):CIRCLE(163
,55,2):
FILLELLIPSE(151,46,2,2):FILLELLIPSE(169
,46,2,2):
TV2:=IMAGESIZE(145,40,175,61):
GETMEM(PF2,TV2):
GETIMAGE(145,40,175,61,PF2^):
PUTIMAGE(145,40,PF2^,1):
(---ENNEMI3---).
SETCOLOR(2):
RECTANGLE(142,50,178,104):RECTANGLE(151
,59,169,75):
FLOODFILL(160,86,2):
SETCOLOR(3):
RECTANGLE(115,86,141,113):RECTANGLE(179
,86,205,113):
SETCOLOR(1):CIRCLE(151,95,6):CIRCLE(169
,95,6):
FILLELLIPSE(133,68,6,6):FILLELLIPSE(187
,68,6,6):
TV3:=IMAGESIZE(115,50,205,113):
GETMEM(PF3,TV3):
GETIMAGE(115,50,205,113,PF3^):
PUTIMAGE(115,50,PF3^,1):
(---ENNEMI4---).
SETCOLOR(2):
SETLINESTYLE(0,0,THICKWIDTH):
RECTANGLE(130,70,190,160):RECTANGLE(145
,85,175,115):
FLOODFILL(160,130,2):
SETCOLOR(3):
RECTANGLE(85,130,128,175):RECTANGLE(192
,130,235,175):
SETCOLOR(1):CIRCLE(145,145,9):
SETLINESTYLE(0,0,THICKWIDTH):CIRCLE(175
,145,9):
FILLELLIPSE(115,100,9,9):FILLELLIPSE(20
5,100,9,9):
TV4:=IMAGESIZE(84,69,236,176):
GETMEM(PF4,TV4):
GETIMAGE(84,69,236,176,PF4^):
PUTIMAGE(84,69,PF4^,1):
(---VOLANT DROIT---).
SETCOLOR(1):
CIRCLE(160,195,55):
CIRCLE(160,195,60):
SETCOLOR(2):
SETLINESTYLE(0,0,THICKWIDTH):
LINE(160,195,130,160):
FILLELLIPSE(160,197,8,6):
TAILLE:=IMAGESIZE(100,140,225,199):
GETMEM(PLACE,TAILLE):
GETIMAGE(100,140,225,199,PLACE^):
PUTIMAGE(100,140,PLACE^,1):
(---VOLANT GAUCHE---).
SETCOLOR(1):
CIRCLE(160,195,55):
CIRCLE(160,195,60):
SETCOLOR(2):
SETLINESTYLE(0,0,THICKWIDTH):
LINE(160,195,190,160):
FILLELLIPSE(160,197,8,6):
TAILLE2:=IMAGESIZE(100,140,225,199):
GETMEM(PLACE2,TAILLE2):

```

PROGRAMMATION

```

GETIMAGE(100,140,225,199,PLACE2^);
PUTIMAGE(100,140,PLACE2^,1);
{---VOLANT CENTRE---}
SETCOLOR(1);
CIRCLE(160,195,55);
CIRCLE(160,195,60);
SETCOLOR(2);
SETLINESTYLE(0,0,THICKWIDTH);
LINE(160,195,214,190);
LINE(160,195,106,190);
FILLELLIPSE(160,197,8,6);
TAILLE3:=IMAGESIZE(100,140,225,199);
GETMEM(PLACE3,TAILLE3);
GETIMAGE(100,140,225,199,PLACE3^);
PUTIMAGE(100,140,PLACE3^,1);
END;
{-----}
PROCEDURE INITIALISATION;
BEGIN
  X:=160;Y:=170;ZA:=TRUE;TOUCHE:=#0;I:=33
  0;K:=12;L:=11;L:=' ':U:=1;
  A:=14;M:=0;B:=0;XX2:=140;YY2:=1;O:=0;P:
  =0;Q:=0;R:=1;T:=1;S:=1;
  V:=0;DRI:=1;MODE:=2;J1:=0;X1:=180;Y1:=1
  ;O1:=0;SC:=0;SC1:=10;KL:=0;
  SCT1:='10';KL:='0';BO:=1;SC2:=0;SC3:=0
  ;T1:=1;KL1:=0;KL2:=0;KL3:=0;
  SCT:=0;O5:=0;SCO:=0;SCOR:=0;
END;
{-----}
PROCEDURE TACHI;
BEGIN
  IF ZA AND (I>20) THEN
  BEGIN
    DEC(I,5);
  END;
  IF (NOT ZA) AND (I<320) THEN INC(I,5);
  SOUND(410-I);DELAY(10);NOSOUND;
  SETCOLOR(0);
  LINE(CO.XEND,CO.YEND,CO.X,CO.Y);
  END;
{-----}
PROCEDURE DESSIN_ECRAN;
BEGIN
  {---DESSIN DU TACHIMETRE---}
  SETTEXTJUSTIFY(1,1);
  SETTEXTSTYLE(2,0,2);
  SETCOLOR(3);
  FOR M:=6 TO 8 DO CIRCLE(X,Y,A+M);
  {---DESSIN VOITURE---}
  SETCOLOR(2);
  SETFILLSTYLE(7,3);
  SETLINESTYLE(0,0,THICKWIDTH);
  BAR(35,150,80,199);BAR(240,150,285,199)
  ;
  RECTANGLE(35,150,80,200);RECTANGLE(240,
  150,285,200);
  MOVETO(90,199);LINETO(90,175);LINETO(12
  0,150);MOVETO(200,150);
  LINETO(230,175);LINETO(230,199);ELLIPSE
  (160,150,0,180,39,9);
  SETCOLOR(1);
  SETLINESTYLE(0,0,THICKWIDTH);
  LINE(1,199,319,199);
  SETFILLSTYLE(1,1);
  SETLINESTYLE(0,0,0);
  BAR(83,190,87,195);BAR(233,190,237,195)
  ;
  SETCOLOR(3);
  LINE(1,19,320,19);
  {---DESSIN DES COMPTEURS---}
  SETCOLOR(2);
  SETFILLSTYLE(1,1);
  FILLELLIPSE(194,182,12,7);
  OUTTEXTXY(194,182,SC1);
  OUTTEXTXY(194,170,'CRACH');
  FILLELLIPSE(126,182,12,7);
  OUTTEXTXY(126,182,KL);
  OUTTEXTXY(126,170,'KILO');
  WHILE K<360 DO
  BEGIN
    SETCOLOR(0);
    ARC(X,Y,K,K+1,A);
    GETARCCOORDS(CO);
    IF K>80 THEN SETCOLOR(1) ELSE SETCOLOR(
    2);
    INC(K,30);
    STR(L,L);
    OUTTEXTXY(CO.XEND,CO.YEND,L_);
    DEC(L);
  END;
  ZE:=FALSE;
  END;
{---PROCEDURES DE DESSIN DES VOLANTS---}
PROCEDURE VOLANTC;
BEGIN
  IF (B=3) THEN PUTIMAGE(100,140,PLACE^,1
  );
  IF (B=2) THEN PUTIMAGE(100,140,PLACE2^,
  1);
  PUTIMAGE(100,140,PLACE3^,1);
  B:=1;
  END;
{-----}
PROCEDURE VOLANTD;
BEGIN
  IF R<4 THEN INC(R) ELSE R:=1;
  IF B=1 THEN PUTIMAGE(100,140,PLACE3^,1)
  ;
  PUTIMAGE(100,140,PLACE^,1);
  B:=3;
  END;
{-----}
PROCEDURE VOLANTG;
BEGIN
  IF R>1 THEN DEC(R) ELSE R:=4;
  IF B=1 THEN PUTIMAGE(100,140,PLACE3^,1)
  ;
  PUTIMAGE(100,140,PLACE2^,1);
  B:=2;
  END;
{-----}
PROCEDURE ROUTE1;
BEGIN
  SETCOLOR(1);
  SETLINESTYLE(3,0,0);
  LINE(1,165,XX2+O,20);
  LINE(319,165,X1+O,20);
  END;
{-----}

```


PROGRAMMATION

```

PROCEDURE ROUTE0;
  BEGIN
    SETCOLOR(0);
    LINE(1,165,XX2+O,20);
    LINE(319,165,X1+O,20);
  END;
  {-----}
PROCEDURE FLECHES;
  BEGIN
    CASE TOUCHE OF
      '8':BEGIN YY:=10;XX:=160;END;
      '2':BEGIN YY:=195;I:=320;END;
      '4':XX:=5;
      '6':XX:=300;
      '5':XX:=160;
    END;
  END;
  {-----}
PROCEDURE SOU;
  BEGIN
    FOR J:=125 TO 565 DO
      BEGIN
        SOUND(J);DELAY(10);
      END;
    NOSOUND;
  END;
  {-----}
PROCEDURE CHOC;
  BEGIN
    FOR J:=300 TO 500 DO BEGIN
      SOUND(J);DELAY(5);END;NOSOUND;
    SETCOLOR(0);
    LINE(1,165,XX2+O,20);
    LINE(319,165,X1+O,20);
    O1:=0;O:=0;XX2:=130;X1:=190;
    DEC(SC1);
    SETCOLOR(2);
    STR(SC1,SC1);
    SETFILLSTYLE(1,1);
    FILLELLIPSE(194,182,12,7);
    OUTTEXTXY(194,182,SC1);
  END;
  {-----}
PROCEDURE CHANGEPOS;
  BEGIN
    CASE B OF
      2:BEGIN INC(O,6);END;
      3:BEGIN DEC(O,6);END;
    END;
    IF (X1+O>175) AND (B=1) THEN INC(O,4);
    IF (XX2+O<145) AND (B=1) THEN DEC(O,4);
    INC(I2);
    IF I2>100 THEN BEGIN
      RANDOMIZE;
      I2:=0;
      O1:=RANDOM(3);
      CASE O1 OF
        0:O1:=-5;
        1:O1:=0;
        2:O1:=5;
      END;
    END;
    O:=O+O1;
  END;
  {-----}
PROCEDURE PERDU;
  BEGIN
    CLEARDEVICE;
    SETCOLOR(2);
    SETTEXTSTYLE(4,0,8);
    RECTANGLE(1,1,319,199);
    SETCOLOR(1);
    OUTTEXTXY(160,50,'FINI');
    SETCOLOR(3);
    SETTEXTSTYLE(4,0,5);
    OUTTEXTXY(160,130,NOM);
    SOUND(250);DELAY(1000);NOSOUND;SOUND(150);
    DELAY(1000);NOSOUND;
    SOUND(250);DELAY(1000);NOSOUND;SOUND(150);
    DELAY(1000);NOSOUND;
    SC1:=0;
  END;
  {-----}
PROCEDURE ENNEMI;
  BEGIN
    SETFILLSTYLE(0,0);
    O3:=O/112;
    CASE BO OF
      1:O4:=155+O;
      2:O4:=ROUND(145+(83*O3));
      3:O4:=ROUND(115+(63*O3));
      4:O4:=ROUND(85+(43*O3));
    END;
    O5:=11;
    CASE BO OF
      1:PUTIMAGE(O4,O5,PF1^,1);
      2:BEGIN BAR(O2,11,O2+15,18);O5:=50;PUTI
        MAGE(O4,O5,PF2^,1);END;
      3:BEGIN BAR(O2,50,O2+35,71);O5:=70;PUTI
        MAGE(O4,O5,PF3^,1);END;
      4:BEGIN BAR(O2,70,O2+95,133);O5:=90;PUT
        IMAGE(O4,O5,PF4^,1);SOU;
      SC1:=0;PUTIMAGE(O4,O5,PF4^,1);END;
    END;
    O2:=O4;
  END;
  {-----}
PROCEDURE ENNEMI1;
  BEGIN
    SETFILLSTYLE(0,0);
    CASE BO OF
      1:BAR(O2,3,O2+15,18);
      2:BAR(O2,43,O2+35,71);
      3:BAR(O2,64,O2+95,133);
    END;
    TOUCHE:=#0;
  END;
  {-----}
PROCEDURE TIR;
  VAR O6,O7:INTEGER;
  BEGIN
    ROUTE1;
    SETCOLOR(3);
    LINE(135,141,155+O,21);LINE(185,141,165
    +O,21);
    CASE O5 OF
      11:BEGIN O6:=O4+5;O7:=O5-4;END;
      50:BEGIN O6:=O4+15;O7:=O5-4;END;
      70:BEGIN O6:=O4+45;O7:=O5-4;END;
    END;
    IF SC2=1 THEN BEGIN SETCOLOR(1);OUTTEXT
  
```

PROGRAMMATION

```

XY(O6,O7,'PAF');END;.*
FOR J:=600 TO 700 DO BEGIN.*
  SOUND(J);DELAY(5);.*
END;.*
NOSOUND;.*
IF (SC2=1) THEN BEGIN I:=320;SC2:=0;ENN
EM1;END;.*
SETCOLOR(0);.*
LINE(135,141,155+O,21);LINE(185,141,165
+O,21);.*
TOUCHE:=#0;.*
END;.*
{-----}.*
PROCEDURE COMENNEMIS;.*
  BEGIN.*
  {---SYSTEME ACCELERATEUR DES ENNEMIS---}.*
  INC(KL1);KL3:=FORCE-KL;.*
  IF KL3<2 THEN KL3:=2;.*
  IF KL1>=KL3 THEN BEGIN KL1:=0;INC(T1);K
L2:=1;END;.*
  IF KL2=1 THEN BEGIN.*
  CASE T1 OF.*
    1:BEGIN BO:=1;ENNEMI;END;.*
    2:BEGIN BO:=2;ENNEMI;END;.*
    3:BEGIN BO:=3;ENNEMI;END;.*
    4:BEGIN BO:=4;ENNEMI;END;.*
  END;.*
  KL2:=0;.*
  END;.*
  END;.*
  {-----}.*
PROCEDURE MOTEUR;.*
  BEGIN.*
  CASE JOY1 OF.*
    1: STICK(XX,YY,B1,B2);.*
    2: SOUR(B1,B2,XX,YY);.*
  END;.*
  WHILE (TOUCHE<#27) AND (SC1>0) DO.*
  BEGIN.*
  TOUCHE:=UPCASE(READKEY);.*
  WHILE (NOT KEYPRESSED) AND (SC1>0) DO.*
  BEGIN.*
  IF YY>190 THEN I:=320;.*
  IF (B1=1) OR (TOUCHE='7') OR (B2=1) OR
(TOUCHE='9') THEN TIR;.*
  {---COMPTEUR KILOMETRIQUE---}.*
  IF I<250 THEN INC(SC);.*
  IF SC=100 THEN.*
  BEGIN.*
  SETCOLOR(2);.*
  INC(KL);.*
  STR(KL,KL);.*
  SETFILLSTYLE(1,1);.*
  FILLELLIPSE(126,182,12,7);.*
  OUTTEXTXY(126,182,KL);.*
  SC:=0;SC2:=1;T1:=0;.*
  END;.*
  SCT:=10-SC1;.*
  ROUTE1;.*
  {---CHOIX DES SYSTEMES---}.*
  CASE JOY1 OF.*
    1: STICK(XX,YY,B1,B2);.*
    2: SOUR(B1,B2,XX,YY);.*
    3: FLECHES;.*
  END;.*
  {---POSITIONS DES VOLANTS---}.*
  CASE JOY1 OF.*
    1:BEGIN.*
    IF (YY<20)THEN ZA:=TRUE ELSE ZA:=FALSE;.*
    IF (XX>180) AND (B=1) THEN VOLANTD;.*
    IF (XX<10) AND (B=1) THEN VOLANTG;.*
    IF (XX>20) AND (XX<170) AND (B<>1) THEN
VOLANTC;END;.*
    2:BEGIN.*
    IF (YY<80)THEN ZA:=TRUE ELSE ZA:=FALSE;.*
    IF (XX>180) AND (B=1) THEN VOLANTD;.*
    IF (XX<140) AND (B=1) THEN VOLANTG;.*
    IF (XX>140) AND (XX<180) AND (B<>1) THE
N VOLANTC;END;.*
    3:BEGIN.*
    IF (YY<20)THEN ZA:=TRUE ELSE ZA:=FALSE;.*
    IF (XX>290) AND (B=1) THEN VOLANTD;.*
    IF (XX<10) AND (B=1) THEN VOLANTG;.*
    IF (XX>20) AND (XX<280) AND (B<>1) THEN
VOLANTC;END;.*
  END;.*
  {---DESSIN DU TACHIMETRE---}.*
  SETCOLOR(0);.*
  ARC(X,Y,0,1,A-5);.*
  GETARCCOORDS(CO);.*
  SETCOLOR(1);.*
  SETLINESTYLE(0,0,0);.*
  LINE(CO.X,CO.Y,CO.XEND,CO.YEND);.*
  TACHI;.*
  IF (GETPIXEL(34,149)=1) OR (GETPIXEL(28
5,149)=1) THEN CHOC;.*
  ROUTE0;.*
  IF SC2=0 THEN CHANGEPOS;.*
  IF (SC2=1) AND (I<250) THEN COMENNEMIS;.*
  END;.*
  END;.*
  END;.*
  {-----PROGRAMME PRINCIPAL-----}.*
  BEGIN.*
  CLRSCR;.*
  ENTREE;.*
  ZE:=TRUE;COMPT:=0;MUST:=0;.*
  FOR I:=1 TO 10 DO BEGIN SCORT[I]:=0;END
;.*
  REPEAT.*
  BEGIN.*
  {---INITIALISATION DES VARIABLES---}.*
  INITIALISATION;.*
  {---INITIALISATION DU MODE---}.*
  INITGRAPH(DRI,MODE,'');.*
  {---DESSIN DES MOBILES---}.*
  IF ZE THEN DESSIN_MOB;.*
  DESSIN ECRAN;.*
  MOTEUR;.*
  IF SC1=0 THEN PERDU;.*
  CLOSEGRAPH;.*
  SORTIE;.*
  END;.*
  UNTIL (TOUCHE='N') OR (TOUCHE='n');.*
  CLRSCR;.*
  END.{---FIN DE PROGRAMME---}.*
  
```