



EDITIONS DU P.S.I.

la pratique du ZX SPECTRUM

**1. BASIC APPROFONDI.
INITIATION AU LANGAGE
MACHINE.**

xavier linant de bellefonds

La pratique du ZX SPECTRUM

Tome 1

La collection « **MATERIELS** » s'intéresse à l'utilisation de tel ou tel type d'ordinateur, depuis la première prise de contact jusqu'à l'utilisation la plus pointue. Elle est aujourd'hui constituée de 33 titres.

- Exercices pour Apple II — Frédéric Lévy
- La découverte de l'Apple II — Dominique Schraen et Frédéric Lévy
- La pratique de l'Apple II — volumes I et II — Nicole Bréaud-Pouliquen
- La pratique de l'Apple II — volume III — Nicole Bréaud-Pouliquen et Daniel-Jean David
- La découverte de l'Atari — Daniel-Jean David
- La découverte du Commodore 64 — Daniel-Jean David
- La pratique du Commodore 64 — Daniel-Jean David
- L'assembleur du Commodore 64 — Daniel-Jean David
- Dragon tout feu tout flammes — Trévor Toms et John Phipps — traduit par Olivier Arnaud
- La découverte du FX-702P — Jean-Pierre Richard
- Un ordinateur personnel signé IBM — Pierre Lotigie-Laurent
- La découverte du PB-100 — Pierrick Moigneau
- La découverte du PB-1251 — Jean-Pierre Richard
- La découverte du PC-1500 — Jean-Pierre Richard
- Le petit livre du Spectrum — Trévor Toms — traduit par Allan Keil
- Exercices pour ZX-Spectrum — Julien Lévy
- La pratique du ZX-Spectrum — tome 1 — Xavier Linant de Bellefonds
- La pratique du ZX-Spectrum — tome 2 — Marcel Henrot
- La découverte du T07 — Dominique Schraen et Maurice Charbit
- Exercices pour T07 — Maurice Charbit, Dominique Schraen et Frédéric Lévy
- Les graphiques sur TRS-80 — Donn Inman — traduit par Alain Pinaud
- La découverte de la TI-57 — Xavier de la Tullaye
- La découverte du TI-99/4A — Frédéric Lévy et Dominique Schraen
- Exercices pour TI-99/4A — Frédéric Lévy
- La découverte du Vic — volume 1 — Daniel-Jean David
- La pratique du ZX-81 — tome 1 — Xavier Linant de Bellefonds
- La pratique du ZX-81 — tome 2 — Marcel Henrot
- Le petit livre du ZX-81 — Trévor Toms — traduit par Ghislaine Lapeyre
- La découverte de l'Oric 1 — Daniel-Jean David
- La découverte de l'Alice et du MC10 — Maurice Charbit
- La pratique du Vic — Daniel-Jean David
- La découverte du Dragon — Dominique Schraen et Frédéric Lévy

Autres ouvrages relatifs au ZX Spectrum :

- 102 programmes pour Sinclair — Jacques Déconchat
- Collection « PROGRAMMES »
- Le petit livre du ZX Spectrum — Trévor Toms
- Collection « MATERIELS »
- La pratique du ZX Spectrum — tome 2 — Marcel Henrot
- Collection « MATERIELS »
- Exercices pour ZX Spectrum — Julien Lévy
- Collection « MATERIELS »
- Le ZX Spectrum à l'affiche — Jean-François Séhan
- Collection « PROGRAMMES »
- Clefs pour le ZX Spectrum — Jean-François Séhan
- Collection « MEMENTOS »
- Spectrum pour tous — Marcel Henrot et Jacques Boisgontier
- Collection « ... POUR TOUS »

RAPPELS

Les collections :

Les ouvrages d'Édition du P.S.I. actuellement au nombre de 135, sont répartis en collections :

« **MATERIELS** », « **UTILISATIONS DE L'ORDINATEUR** », « **MEMENTOS** », « **PROGRAMMES** », « **GUIDES PRATIQUES** », « **LANGAGES** », « **METHODES PRATIQUES** », « **TECHNIQUES ET METHODES** », « **LOGIGUIDE** », et pour débutants, les collections « **INITIATION** », « **... POUR TOUS** ».

La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'article 41, d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective », et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause, est illicite » (alinéa 1^{er} de l'article 40).

Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles 425 et suivants du Code Pénal.

La pratique du ZX SPECTRUM

par
Xavier Linant de Bellefonds

Tome 1
Basic approfondi.
Initiation au langage machine



Editions du P.S.I.
1984

PRESENTATION DE L'AUTEUR

Xavier LINANT de BELLEFONDS, 38 ans, dirige le Séminaire d'Informatique juridique de la Faculté de Droit d'Orléans ; il s'est également spécialisé dans le Droit de l'Informatique.

SOMMAIRE

	Pages
CHAPITRE I INTRODUCTION ET RAPPELS	7
La pluralité d'instructions par ligne	8
Le jeu de commandes DATA, READ, RESTORE	9
La définition de fonctions	10
CHAPITRE II TRAITEMENT DES CHAINES ET DES TABLEAUX	13
La définition de caractères	13
La construction d'index automatique	19
Le traitement de texte sur ZX Spectrum	25
Les fichiers du ZX-Spectrum	33
CHAPITRE III HAUTE DEFINITION GRAPHIQUE ET UTILISATION COURANTE DES FONCTIONS MATHEMATIQUES	45
Les grandes familles de courbes	46
Introduction aux graphiques de présentation de données	48
Les regressions simples	70
Les représentations de courbes en trois dimensions	77
CHAPITRE IV LA COULEUR ET LE SON	85
La couleur sur le ZX-Spectrum	86
Le son sur le ZX-Spectrum	102
CHAPITRE V LA MEMOIRE DU ZX-SPECTRUM	113
PEEK, POKE et le travail direct avec les octets de mémoire	114
Les grandes lignes d'organisation de la mémoire vive	117
Les grandes lignes de la ROM du Spectrum	131

Un programme de visualisation de la place mémoire disponible	133
CHAPITRE VI INITIATION AU LANGAGE MACHINE	139
Basic et langage machine	139
L'étude du langage machine	140
La structure du microprocesseur Z80A	141
Développement d'un programme en langage machine	147
ANNEXES CORRIGES DES EXERCICES	156

CHAPITRE I

INTRODUCTION ET RAPPELS

En rédigeant cet ouvrage de programmation approfondie sur le ZX-Spectrum, l'auteur a pensé à deux catégories de lecteurs potentiels :

- ceux qui mènent leur initiation à l'informatique directement sur cet appareil ;
- ceux qui se sont formés à l'informatique sur le remarquable prédécesseur du ZX-Spectrum, le ZX 81, et qui désirent passer à un ordinateur plus puissant avec le ZX-Spectrum.

Ce livre s'adresse donc, dans les deux cas, à un public déjà entraîné à la programmation Basic élémentaire : on tient pour acquis que les premiers ont assimilé la documentation de base de leur ordinateur, et que les seconds savent programmer efficacement leur ZX81.

Seule différence : les possesseurs de ZX81, dont l'initiation au Basic est chose faite, chercheront sans doute à passer à la programmation approfondie sans s'astreindre, au préalable, à la lecture du manuel de base du ZX-Spectrum. C'est à ceux-ci qu'est plus particulièrement destinée la brève présentation de quelques pages qui suit.

Cette introduction n'est pas axée sur les différences primordiales entre les deux ordinateurs telles que le son, la couleur, la haute définition graphique, etc. : ces caractéristiques font chacune l'objet d'un chapitre entier ; elle sera plutôt consacrée à quelques rappels rapides mais nécessaires de différences moins détermi-

nantes ou apparentes, mais constamment présentes dans l'ensemble des logiciels d'application.

Ces rappels (également utiles aux possesseurs de ZX-Spectrum qui voudront actualiser l'innombrable bibliothèque des programmes ZX81 pour les transposer sur leur appareil) porteront sur les points suivants : la pluralité d'instructions par ligne, le jeu de commandes **DATA**, **READ**, **RESTORE**, la définition de fonctions, les codes **ASCII**.

LA PLURALITE D'INSTRUCTIONS PAR LIGNE

Contrairement au ZX81 qui n'acceptait qu'une seule instruction par ligne, ce qui donnait, aux listes de cet ordinateur, un aspect net et personnel, le ZX-Spectrum accepte que, sous un même numéro de ligne, figurent plusieurs instructions séparées par un " : ". Il s'agit là d'une facilité incontestable qui permet de grouper, sur une même ligne, toutes les instructions d'une boucle, plusieurs déclarations simultanées de variables, etc.

Autre utilité appréciable : au cours de la mise au point des programmes, on écrit souvent des routines provisoires dont le but est de mettre en évidence des résultats intermédiaires pour vérifier que les différentes parties d'un programme tournent correctement : il suffit alors d'introduire un seul numéro de ligne pour effacer l'ensemble des instructions devenues inutiles.

On ne discute pas l'intérêt des facilités supplémentaires qu'offre un ordinateur, mais il faut être conscient qu'elles peuvent se retourner contre le programmeur : la pluralité d'instructions par ligne permet de nombreux rattrapages, mais elle aboutit forcément à des listes plus chargées qui rendent les programmes moins lisibles au premier coup d'oeil, et ceci est sensible au moment de l'écriture ; ensuite, elle autorise une décomposition moins fouillée des organigrammes et donc une programmation moins structurée, c'est-à-dire susceptible de renfermer davantage d'erreurs difficilement décelables. Il faut conseiller de réserver la pluralité d'instructions par ligne aux instructions similaires ou solidaires.

Il n'y a qu'un seul cas où cette facilité est une réelle supériorité : c'est au niveau des tests ; en voici un exemple :

```
10 IF A = 0 THEN LET B = 1 : LET C = 2
.....
```

Si le test est positif, le reste de la ligne a valeur de sous-programme conditionnel qui ne sera pas exécuté si le test est négatif. Avec une seule instruction par ligne, il faudrait envoyer à un sous-programme de la forme :

```
10 IF A = 0 THEN GOSUB 1000
.....
1000 LET B = 1
1010 LET C = 2
1020 RETURN
```

LE JEU DE COMMANDES DATA, READ, RESTORE

Il s'agit de trois ordres Basic d'une grande utilité, bien qu'un passage du manuel de base du ZX81 s'efforce de démontrer le contraire. Ces commandes permettent de stocker, dans la zone programme, toutes les variables dont un programme a besoin (**DATA**) qui seront lues par ce programme au moment voulu (**READ**) ; ceci, dans un ordre séquentiel, à partir de la donnée choisie (**RESTORE n**). Pour réaliser l'équivalent avec un ZX81, il faudrait accumuler les instructions déclaratives et faire appel à un tableau pour pouvoir relire ces données à partir d'un certain numéro.

Voici un programme graphique, sans prétention, qui illustre la mise en oeuvre concomitante de ces trois commandes : les lettres du mot "SINCLAIR" viennent s'accoler les unes aux autres, verticalement et horizontalement, pour former un motif coloré.

```
1 DATA "S" : 3
2 DATA "I" : 1
3 DATA "N" : 5
4 DATA "C" : 4
5 DATA "L" : 6
6 DATA "A" : 2
7 DATA "I" : 3
8 DATA "R" : 1
30 FOR L=0 TO 7
40 FOR I=1 TO 8-L
45 READ A$(B)
50 FOR J=31 TO I-1 STEP -1
60 PRINT PAPER B, INK 9; AT L, J
A$;" "
70 NEXT J
80 NEXT I
90 RESTORE (L+2)
100 NEXT L
```


Les instructions **DATA** comprennent deux valeurs : une valeur littérale et une valeur numérique ; elles sont lues par la ligne 45 ; de façon à provoquer un décalage, on réinitialise la série des données à lire avec l'instruction 90. Il faut éviter le compte-rendu d'erreur *Out of Data*.

LA DEFINITION DE FONCTIONS

Les commandes **DEF FN** et **FN** sont très puissantes sur le ZX-Spectrum, puisque l'utilisateur peut définir jusqu'à 52 fonctions (26 fonctions numériques et 26 fonctions littérales).

DEF FN définit la fonction, et **FN** rappelle la fonction en calculant sa valeur.

Sur le ZX81, il était possible de définir des fonctions par l'intermédiaire de la commande **VAL** associée à une chaîne (*figure I-2*).

Une expression est enfermée dans une chaîne littérale ; elle contient une variable ; il suffit de définir la variable pour que l'ordre **VAL** donne la valeur de l'ensemble de l'expression. On peut même, de cette manière, définir des fonctions de plusieurs variables.

L'existence, sur le ZX-Spectrum, d'un jeu de commandes spécialisées, donne plus de souplesse au processus, comme le démontre le court programme de la *figure I-3* qui intègre des possibilités nouvelles.

On s'aperçoit :

- *premièrement*, que la définition peut être placée dans le programme, à n'importe quel endroit ; en effet, sollicité de retrouver une fonction, l'ordinateur ira chercher la formule de cette fonction à n'importe quel endroit du programme, tandis que, dans le ZX81, la formule, enfermée dans une déclaration de chaîne, devait précéder évidemment la lecture ;

- *deuxièmement*, l'attribution d'un argument à la fonction se fait d'une manière directe et non par la prédéfinition d'une variable ; les possibilités sont nombreuses puisque l'on peut calculer directement pour un argument numérique (ligne 53), pour une ou plusieurs variables, ou même pour pas de variable du tout (auquel cas, c'est la première variable qui est prise en compte) ;

```
10 LET A$="2*COS I+(EXP I)/10"
15 INPUT I
20 LET F=VAL A$
30 PRINT F
```

Figure I-2

```
10 INPUT I
30 PRINT FN A(I)
40 INPUT J
50 PRINT FN A(J)
51 PRINT FN B()
52 PRINT FN C(I,J)
53 PRINT FN A(3)
60 DEF FN A(I)=2*COS I+(EXP I)
/10
70 DEF FN B()=2*COS I+(EXP I)
10 80 DEF FN C(I,J)=2*COS I+(EXP
J)/10
```

Figure I-3

- *troisièmement* et surtout, il est possible d'obtenir la valeur d'une fonction pour une variable, même lorsque la fonction a été définie avec le nom d'une autre variable, ce qui était impossible avec le système de la déclaration de chaîne.

Par exemple, pour obtenir l'équivalent de la ligne 50, qui calcule la valeur, pour la variable J, de la fonction A définie à partir de la variable I, il aurait fallu, d'abord, établir l'équivalence : LET I = J, ce qui perturbe les affectations, à moins que l'on n'effectue des sauvegardes provisoires.

LE CODE ASCII

Le langage du ZX81 est spécial tant dans la syntaxe de son Basic que dans le codage de ses caractères ; la syntaxe du ZX-Spectrum reste originale, bien qu'elle se rapproche des Basic standards par l'introduction de certaines commandes, mais se rallie en partie au codage ASCII pour les caractères. On y gagne en normalisation (caractères minuscules, etc.).

On y perd une facilité à laquelle les programmeurs de ZX81 étaient attachés, qui consistait à pouvoir obtenir directement l'inverse des caractères, au clavier ou par calcul. La commande de l'impression inversée d'un caractère se contrôle désormais par l'ordre **INVERSE** ou par **CHR\$ 20**. Il ne suffit donc plus d'ajouter 128 au code d'un caractère pour obtenir son inverse, l'une des propriétés les plus personnelles du ZX81.

CHAPITRE II

TRAITEMENT DES CHAINES ET DES TABLEAUX

Le Spectrum possède un jeu très riche de fonctions et de commandes pour le traitement des chaînes de caractères comme on l'a vu dans l'introduction.

Nous allons nous attacher, dans le cadre de ce chapitre, à en faire l'exploitation systématique avec quatre applications différentes qui recouvrent tous les champs de possibilités de cet ordinateur. La définition de caractères personnalisés, la construction d'index automatique, le traitement de texte sophistiqué et enfin les fichiers à articulation complexe.

LA DEFINITION DE CARACTERES

La définition d'un certain nombre de caractères personnalisés, directement accessibles au clavier par l'utilisateur, est l'une des grandes possibilités offertes par le Spectrum que ne possédait pas son prédécesseur, le ZX 81.

Cette faculté est particulièrement intéressante dans les applications de jeux car l'on peut faire afficher sur une ligne ou une colonne instantanément un motif différent des caractères ASCII, c'est-à-dire faire apparaître par exemple un envahisseur ou un "gobbler" en utilisant autre chose qu'un W ou un G en vidéo inversée.

Nous verrons plus loin, en parlant de la mise en

oeuvre de graphiques à haute définition, que l'on peut avoir accès par la fonction **PLOT** à chacun des 256 x 192 points qui composent l'écran. Il est très facile de construire une routine d'impression qui se positionne sur l'un des coins de l'une quelconque des 32 x 22 cases de l'écran pour **PLOT**er individuellement, selon le dessin recherché, les points choisis parmi les 64 qui composent chacune des cases. Néanmoins cette procédure est extrêmement lourde et voilà pourquoi il est satisfaisant de disposer d'une méthode d'accès direct à partir du clavier.

Au-delà des caractères graphiques purement visuels utilisés dans les programmes de jeux, on peut se construire des caractères symboliques (mathématiques) ou alphabétiques (alphabets étrangers ou même alphabet français pour les accentuations). Ces applications "sérieuses" vont beaucoup plus loin que les jeux et l'utilisateur pressé doit pouvoir disposer d'un sous-programme indépendant ou intégré qui lui permette de définir rapidement les caractères dont il a besoin pour les stocker dans la partie de la RAM réservée à cet usage au-dessus de 32600 (ou 65368 pour le Spectrum 48K).

En effet la méthode directe proposée par le manuel, effectuant l'introduction ligne par ligne du caractère à inventer (8 lignes pour 1 caractère) avec l'aide de la fonction **BIN**, est particulièrement fastidieuse dès lors que le caractère est complexe, ce qui exclut d'y avoir recours pour plus de trois caractères. Il faut dessiner manuellement une grille de 8 sur 8, noircir les cases selon le dessin recherché, décomposer la grille en 8 lignes, transposer les noirs en "1" et les blancs en "0" pour obtenir une ligne binaire de 8 chiffres, et ainsi de suite jusqu'à ce que la totalité des lignes soit entrée. Cette méthode est désespérante de lenteur et n'aboutit pas toujours à un résultat satisfaisant. Le caractère imprimé sur l'écran se révèle être un gribouilli illisible et il faut recommencer l'ensemble de l'opération pour remodeler le caractère en gagnant sur la hauteur ou la largeur et en évitant de noircir telle ou telle case superflue. Un caractère affiché sur l'écran n'a pas du tout le même "rapport à l'oeil" que le même caractère décomposé en grand sur le papier.

Le programme qui suit est un programme d'aide à la définition de caractères qui cherche à éviter les différents écueils que nous venons de mentionner. Son introduction ne pose pas de problème car il est très court et le travail est souple et agréable.

Le fonctionnement du programme

A la commande **RUN**, vient s'afficher dans le bas de l'écran une grille de 8 cases sur 8. Un pavé noir "flashe" sur la première des cases. Il est possible de déplacer ce pavé à l'aide des quatre curseurs du Spectrum (les quatre flèches), de façon à le positionner sur les cases désirées. Si l'on veut noircir la case, il suffit alors de presser sur la lettre N. A noter que, et c'est là une des grandes facilités de ce programme, on peut se positionner sur une case déjà noircie qui "flashe" de nouveau : presser à ce moment-là sur la lettre N provoquera l'effet inverse de blanchir cette case dans le cas où il se serait agi d'une erreur.

En regard de la grille s'inscrit un carré qui contient, en dimension réelle (ce qui permet naturellement de rectifier le tir en cas de besoin), le caractère en cours de formation. Lorsque la forme du caractère donne satisfaction, il suffit de presser la touche "0" (comme O.K.) pour que l'ordinateur demande à quelle touche du clavier on souhaite faire correspondre le nouveau caractère. En effet cette place n'est pas indifférente et, par exemple, si l'on cherche à définir un caractère symbolique, comme α , il est naturellement tout indiqué de l'affecter à la touche A. Le chargement des différents octets de mémoire correspondant à la touche choisie se fera automatiquement.

Afin d'éviter de surcharger deux fois la même touche ou de rechercher quelle touche n'a pas encore été affectée, dans le cas d'une redéfinition exhaustive du clavier, l'ensemble de l'alphabet normal vient s'afficher au sommet de l'écran, avec en-dessous des lettres qui ont déjà été affectées, le caractère graphique correspondant. Les caractères graphiques sont donc tous affichés sur la même ligne, ce qui permet de les remodeler les uns par rapport aux autres en taille et en forme.

Détail du logiciel (figure II-1)

Le programme commence par une instruction, l'instruction 8 qui a une importance considérable : c'est l'instruction **OVER 1** qui est une manière simple de permettre l'**inversion blanc-noir** dont nous avons parlé au début. En effet elle signifie que réimprimer un noir sur un noir redonne un blanc. C'est la particularité de cette commande.

```

1 REM ALPHABET
2 REM -----
3 REM ATTENTION-Ne fonctionne
  qu'en mode "CAPS LOCK"
4 OVER 1
5 REM
6 REM Trame
7 REM
8 FOR I=7 TO 78 STEP 8
9 PLOT 8,I: DRAW 64,0
10 PLOT I+1,7: DRAW 0,64
11 NEXT I
12 PLOT 99,70: DRAW 12,0: DRAW
  0,-13: DRAW -13,0: DRAW 0,13
13 DIM A$(8,8)
14 FOR I=1 TO 8
15 LET A$(I)=""
16 NEXT I
17 DIM A(8)
18 LET I=13: LET J=1
19 PRINT FLASH 1:AT I,J;" "
20 LET A=I: LET B=J
21 IF INKEY$="" THEN GO TO 400
22 IF INKEY$="N" THEN GO TO 50
23 PRINT AT A,B;" "
24 LET I=I+(INKEY$="6")-(INKEY$="7")
25 LET J=J+(INKEY$="8")-(INKEY$="5")
26 PRINT FLASH 1:AT I,J;" "
27 PAUSE 100: LET A=I: LET B=J
28 GO TO 400
29 REM
30 REM Noircir les cases
31 REM
32 PRINT AT I,J:CHR$ 143
33 PLOT J+100,80-I
34 LET A$(I-12,J)=STR$(ABS (V
  AL A$(I-12,J)-1))
35 IF (INKEY$="N") OR (INKEY$="
  ") THEN GO TO 520
36 IF INKEY$="O" THEN GO TO 60
37 GO TO 420
38 REM
39 REM Definition des touches
40 REM
41 PAUSE 100
42 PRINT AT 18,10;"Lettre clav
  ier": PRINT AT 19,10;"correspond
  ante ?"
43 INPUT N$: LET N=(CODE N$-65
  )*8
44 FOR I=13 TO 20
45 FOR J=1 TO 8
46 IF A$(I-12,J)="1" THEN LET
  A(I-12)=A(I-12)+2*(8-J)
47 NEXT J
48 POKE 32587+I+N,A(I-12)
49 NEXT I
50 REM
51 REM Affichage caracteres
52 REM
53 CLS
54 FOR M=0 TO 20
55 PRINT AT 0,M:CHR$ (65+M)
56 PRINT AT 1,M:CHR$ (144+M)
57 NEXT M
58 GO TO 100

```

Figure II-1

Les lignes 100 à 160 dessinent la trame de la grille ainsi que celle du petit carré qui encadrera le caractère en cours de formation à droite de la grille.

Les lignes suivantes préparent un tableau de 8 lignes de 8 caractères alphanumériques qui représenteront la matrice du caractère à définir. Les lignes 400 gèrent le curseur à l'aide de fonctions logiques utilisant la fonction **INKEY\$**. Les lignes 500 commandent le noircissement de la case choisie avec, dans le tableau représentatif de la matrice, le basculement pour la case de la valeur "0" à la valeur "1". Une fois achevée la formation du caractère (ligne 530 : pression de la touche "O"), a lieu la définition des touches (lignes 600 et suivantes). Cette partie du programme transforme chacune des lignes du tableau matriciel de 8 x 8 en un nombre décimal converti du nombre binaire équivalent à la succession des "0" et des "1" (ligne 630); la ligne 650, qui fait intervenir une commande dont l'emploi systématique sera longuement expliqué dans un chapitre ultérieur, la commande **POKE**, introduit dans une cellule de mémoire RAM la valeur décimale ainsi obtenue; c'est la partie de la mémoire réservée à la définition des touches graphiques au-dessus de 32600.

Enfin la routine d'affichage des caractères, qui va de 700 à 740, ne pose aucun problème de compréhension. Les lignes 720 et 730 affichent d'une manière superposée la valeur normale de la touche et le correspondant graphique qui lui est affecté.

Utilisation systématique des caractères prédéfinis

L'ordinateur ZX-Spectrum ne réserve pour les graphiques définis par l'utilisateur que 21 touches : celles correspondant aux lettres A à U. Il est naturellement possible, en modifiant la valeur portée à la ligne 650, d'augmenter l'emplacement de mémoire RAM réservé aux différents caractères supplémentaires que l'on voudrait introduire. Mais ceci fait appel à une technique que nous verrons dans un chapitre ultérieur et, de toute façon, l'emplacement artificiellement ainsi créé ne bénéficierait pas de la même "immunité". En effet la zone RAM affectée aux graphiques définis par l'utilisateur (au-dessus de 32600) n'est modifiée ni par la commande **NEW** ni par la commande **CLEAR**, ce qui permet de disposer de ces caractères, même lorsqu'un autre programme a été chargé ultérieurement (sans que naturel-

48K :
65355

ABCDEFGHIJKLMNOPQRSTUVWXYZ
αβγδεϕζηιθκλμνξοπρστυ

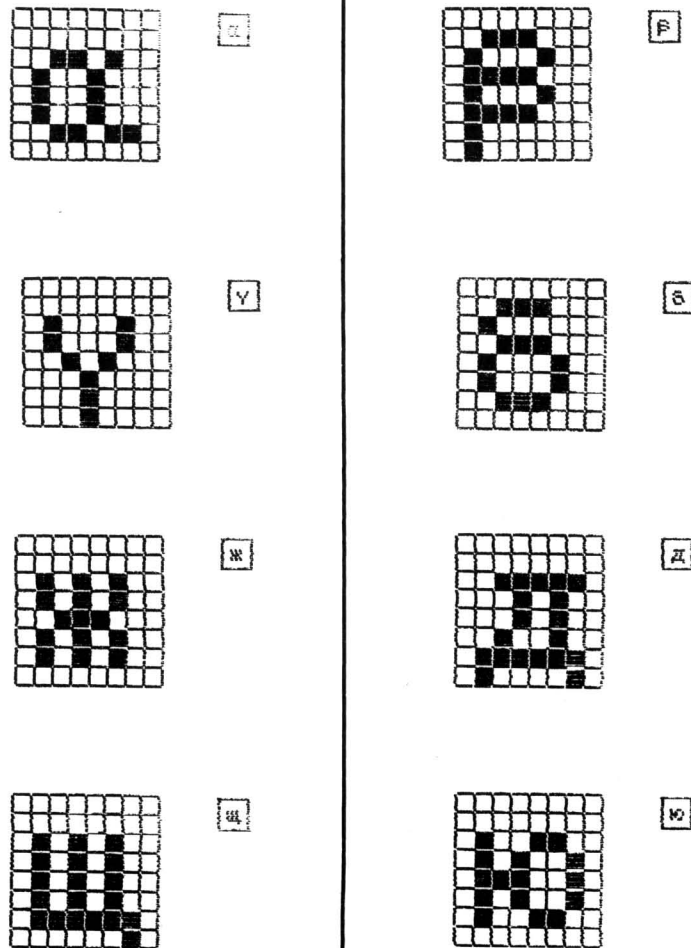


Figure II-2

lement, entretemps, il y ait eu une interruption de l'alimentation). C'est ce qui fait tout l'intérêt de ce programme de définition de caractères.

A titre d'illustration ont été reproduits avec cette méthode les quatres premiers caractères de l'alphabet grec et quatre caractères "étranges" de l'alphabet cyrillique (*figure II-2*).

LA CONSTRUCTION D'INDEX AUTOMATIQUE

Le travail sur les chaînes de caractères est essentiellement effectué par les programmes de fichier et nous présentons un semblable programme dans le cadre de cet ouvrage. Mais il existe un autre cas où l'ordinateur peut avantageusement remplacer le travail sur fiches manuelles : c'est le cas de la construction d'index. Que l'on se reporte à la fin de cet ouvrage et l'on verra l'index des principales matières traitées par ce livre ; cet index a été dressé à l'aide d'un programme interactif qui met en oeuvre toutes les fonctions de chaînes du ZX-Spectrum et qui, à ce titre, présente une bonne valeur illustrative.

La construction d'index

Lorsqu'on parcourt un livre, un rapport, des cours, des notes etc. avec le désir de dresser un fichier des principales questions abordées, la méthode manuelle la plus couramment utilisée consiste à ouvrir un paquet de fiches et à créer une fiche nouvelle chaque fois que l'on rencontre un mot nouveau devant figurer dans l'index. Lorsque le nombre des termes devient important, chaque création de fiche implique qu'on vérifie d'abord qu'aucune fiche n'a été créée pour ce terme ; lorsqu'une fiche a été créée et que, dans les pages suivantes, on rencontre la même notion, il faut ressortir du paquet la fiche en question pour faire porter dessus la nouvelle référence. Ceci implique un nombre de manipulations manuelles considérable qui peut être aisément épargné par l'ordinateur si celui-ci est programmé pour faire défiler très rapidement tous les termes de l'index afin de permettre à l'utilisateur de sélectionner sans fatigue le mot qui l'intéresse et d'ajouter la nouvelle référence. Tous les problèmes de tri, d'insertion, de décalage sont alors pris en charge par l'ordinateur qui peut même, en fin de travail, éditer l'ensemble

de l'index.

L'exemple retenu ici porte sur un index de 100 termes, termes de 25 caractères au plus, chacun de ces termes pouvant être affecté de 10 références. Ceci est nettement inférieur aux capacités du ZX-Spectrum, que l'on peut encore accroître en se contentant d'un nombre inférieur de références par terme et de termes plus courts selon la matière traitée.

Le second intérêt de ce programme est de fournir une initiation aux différentes techniques de défilement des pages d'écran.

Bien qu'il soit relativement court, ce programme de construction d'index comporte une routine de correction intéressante qui permet de modifier en cours de travail l'intitulé d'un terme et également de corriger une référence.

Fonctionnement du programme (figure II-3)

Le programme commence par initialiser en construisant un index fictif : il charge le tableau qui contiendra les données de l'index avec une centaine de lettres ; puis il affiche les 20 premières lignes du tableau ainsi créé. L'utilisateur a à sa disposition les flèches "7" et "6" pour remonter ou redescendre dans l'ordre des pages. Lorsqu'il n'a nulle part rencontré d'un coup d'oeil l'article qui l'intéresse, il procède à la création en pressant sur la touche "0" qui envoie à la routine d'introduction, lignes 5000 et suivantes. Lorsque le mot qui l'intéresse figure dans la page d'écran, en pressant sur la flèche "8", l'utilisateur fait apparaître un curseur qu'il positionne en regard du terme recherché, toujours à l'aide des curseurs "7" et "6" pour monter ou redescendre (figure II-4).

Dans le cas où il faut créer un nouveau terme, l'ordinateur demande quel est ce nouvel article, procède à son insertion dans l'ensemble de la table et envoie à la routine d'addition de page pour que la première référence soit entrée.

Lorsque le terme existe déjà dans la base et que l'utilisateur a placé le curseur devant lui, il a l'option entre l'addition d'une **nouvelle référence** et la **correction**. Il suffit de presser sur les lettres P ou C pour obtenir respectivement l'une de ces deux routines. La routine de correction propose le choix entre la correction de l'article ou la correction de

```

1000 REM CONSTRUCTION D'INDEX
1010 REM -----
1040 REM
1050 REM INITIALISATION
1060 REM
1100 DIM T$(101,25)
1200 DIM P$(101,10)
1210 DIM Q(101)
1220 LET Z$=CHR$ 0
1230 FOR I=1 TO 9
1240 LET Z$=Z$+CHR$ 0
1250 NEXT I
1300 FOR I=1 TO 101
1320 LET T$(I)=CHR$ (I+100)
1330 LET P$(I)=Z$
1340 NEXT I: LET E=1
1900 REM
2000 REM AFFICHAGE
2050 CLS
2100 FOR L=0 TO 20
2120 PRINT TAB 1;T$(E+L)
2140 NEXT L
2900 REM
3000 REM CHANGEMENT D'ECRAN
3050 REM
3100 IF INKEY$="" THEN GO TO 310
3150 IF INKEY$="0" THEN GO TO 50
3200 IF INKEY$="8" THEN GO TO 40
3300 IF INKEY$="7" THEN LET E=E-
20*(E>20): GO TO 2000
3400 IF INKEY$="6" THEN LET E=E+
20*(E<=80): GO TO 2000
3500 GO TO 3100
3900 REM
4000 REM CURSEUR
4005 REM
4010 LET C=10
4020 PRINT AT C,0; FLASH 1;">":
LET D=C
4030 PAUSE 30
4100 IF INKEY$="" THEN GO TO 410
4150 IF INKEY$="P" OR INKEY$="p"
THEN GO TO 6000
4160 IF INKEY$="C" OR INKEY$="c"
THEN GO TO 7000
4200 IF INKEY$="8" THEN GO TO 43
4250 IF INKEY$="7" THEN GO TO 44
4300 LET C=C+1: IF C>21 THEN GO
TO 3400
4350 GO TO 4500
4400 LET C=C-1: IF C<0 THEN GO T
O 3300
4500 PRINT AT D,0;" "
4600 GO TO 4020
4900 REM
5000 REM INTRODUCTION
5050 CLS
5100 PRINT "Nouveau terme?"
5200 INPUT N$
5300 FOR I=1 TO 101
5310 IF N$<T$(I) THEN GO TO 5400

```

Figure II-3


```

5320 NEXT I
5400 FOR K=101 TO I+1 STEP -1
5420 LET T$(K)=T$(K-1): LET P$(K)
)=P$(K-1)
5440 NEXT K
5500 LET T$(I)=N$: LET P$(I)=Z$
5600 GO TO 2000
5900 REM
6000 REM ADDITION D'UNE PAGE
6050 REM
6100 LET I=E+C
6150 CLS
6200 PRINT T$(I)
6210 FOR J=2 TO 10
6220 IF P$(I,J)=CHR$ 0 THEN GO TO 6250
6230 PRINT CODE P$(I,J)
6240 NEXT J
6245 RETURN
6246 PRINT
6250 PRINT "Quelle page ?": INPUT
T P
6300 LET P$(I,J)=CHR$ P: PRINT P
6400 LET P$(I,1)=CHR$ (CODE (P$(
I,1))+1)
6500 PAUSE 50
6600 GO TO 2000
6900 REM
7000 REM CORRECTION
7050 REM
7100 LET I=E+C
7150 CLS: PRINT FLASH 1;"A CORR
IGER:"; PRINT T$(I)
7160 PRINT "Correction de l'arti
cle (A) ou de la reference (R)?"
7170 INPUT X$: IF X$="R" THEN GO
TO 7400
7210 PRINT "Nouvel article?": IN
PUT N$: LET T$(I)=N$
7300 GO TO 2000
7400 PRINT "Quelle reference?":
INPUT Q
7450 PRINT "Nouvelle page?": INP
UT P
7500 LET P$(I,Q+1)=CHR$ P
7600 GO TO 2000
7900 REM
8000 REM IMPRESSION
8050 REM
8100 FOR I=1 TO 101
8150 LET S$=""
8200 FOR J=1 TO 23
8300 IF T$(I,J+1)+T$(I,J+2)=" "
THEN GO TO 8400
8350 LET S$=S$+T$(I,J)
8370 NEXT J
8400 PRINT S$;T$(I,J);: GO SUB 8
500
8430 IF T$(I,1)<>T$(I+1,1) THEN
PRINT
8450 NEXT I
8500 FOR K=2 TO 10
8600 IF P$(I,K)=CHR$ 0 THEN PRIN
T: RETURN
8700 PRINT ", ";CODE P$(I,K);
8800 NEXT K

```

Figure II-3

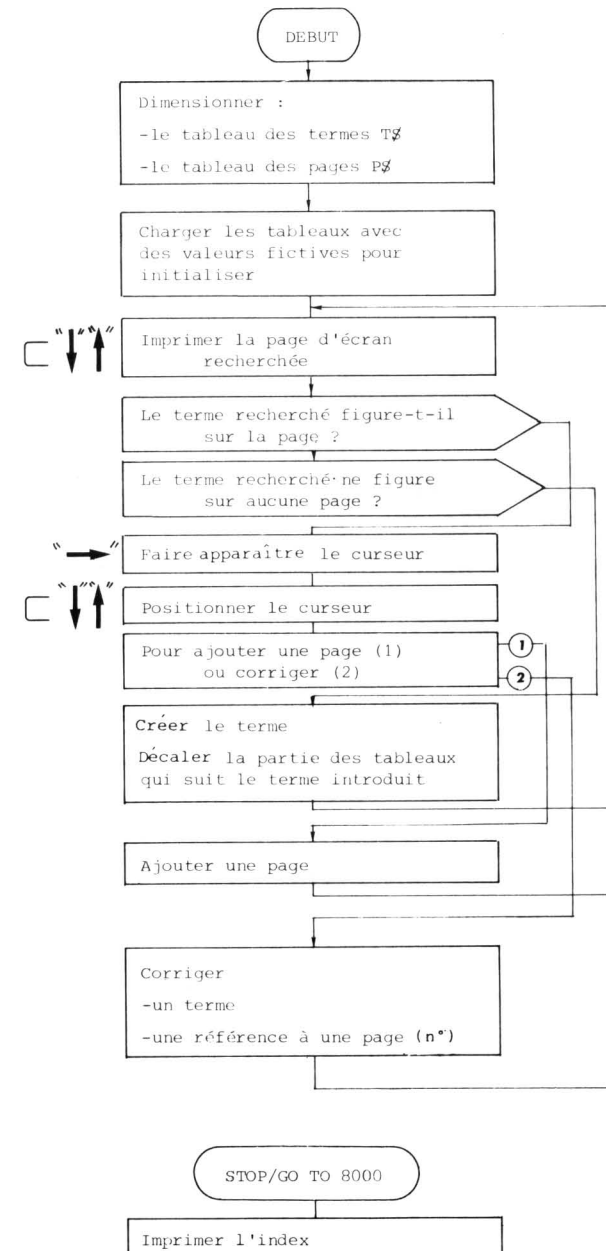


Figure II-4 Programme de construction d'index (organigramme)

la référence n° 1, 2, etc... Il est à noter que cette correction peut difficilement porter sur les toutes premières lettres de l'article car cet article risque de se trouver alors à une place qui n'est pas la sienne dans l'ordre alphabétique. Il faut dans ce cas créer de toutes pièces un nouvel article par la routine d'introduction.

La routine d'impression (8000 et suivantes) imprime la totalité de la table en imprimant des blancs lorsqu'on change de lettre alphabétique, blancs qui peuvent être augmentés ou réduits en ajoutant des **PRINT** ou en les soustrayant à la ligne 8430. Pour ne pas surcharger le programme, il n'a pas été prévu de routine de suppression proprement dite ; l'élaboration de ce sous-programme est confiée au lecteur à titre d'exercice.

Précautions à prendre dans la mise en oeuvre du programme

L'utilisateur devra calculer, en fonction de la place mémoire dont il dispose et qui, il faut bien le reconnaître, est courte dans le ZX-Spectrum 16K à cause de l'importance de la partie RAM consacrée à la gestion de la couleur et de la haute définition, quelle est la place mémoire qui reste disponible et voir s'il préfère plus de termes et moins de références ou moins de termes, chacun étant affecté de davantage de références.

Pour gagner de la place, les références chiffrées ne sont pas inscrites en chiffres mais entrées selon le caractère qui correspond à ce chiffre dans la table ASCII, en sorte qu'il est impossible d'entrer une référence qui serait supérieure à 255 ; le code du "vide" est égal à 0 ce qui n'est pas grave puisqu'il existe rarement une page zéro. Lorsque l'index a été interrompu en cours de constitution et provisoirement sauvegardé sur cassette, il faut éviter la touche **RUN** qui a pour effet d'effacer toutes les données : faire alors **GOTO 2000**. Le codage est effectué par la ligne 6400 et le décodage au stade de l'impression par la ligne 8700 (deuxième instruction).

EXERCICE 2.1

Construire une routine de suppression qui effectue l'écrasement de l'article non souhaité par la remontée d'un cran de l'ensemble des articles qui suivent ;

on s'inspirera pour ce faire de la routine de décalage qui est à la deuxième partie de la routine d'introduction (5000) ; seulement le décalage, au lieu de se faire par le bas, se fera par le haut.

LE TRAITEMENT DE TEXTE SUR ZX-SPECTRUM

A la différence du ZX-81, dont le jeu de caractères ne comporte que des capitales, le ZX-Spectrum met à la disposition de l'utilisateur un alphabet minuscule complet et une ponctuation plus riche (apostrophe en particulier). Il en résulte qu'il est possible de réaliser avec le nouveau Sinclair des éditions de texte assez réussies. Il nous a même paru légitime d'envisager des possibilités de traitement de texte et c'est un semblable programme que nous soumettrons maintenant au lecteur pour lui permettre d'approfondir dans une nouvelle direction les problèmes liés à la manipulation des chaînes de caractères.

Une première précision : bien que la possession d'une imprimante Sinclair soit conseillée pour tirer tout le parti possible de ce programme, il est parfaitement possible de l'utiliser indépendamment d'une imprimante pour obtenir à l'écran des éditions parfaites.

L'objet du traitement de texte

Dans la pratique commerciale du marché de l'informatique, la notion de traitement de texte est extrêmement large et recouvre des réalités souvent très différentes. L'application que nous développons ici est évidemment limitée dans son ambition au traitement de texte au sens strict, c'est-à-dire qu'il s'agit d'obtenir une édition parfaite, avec **double justification à droite et à gauche** d'un texte préalablement entré dans l'ordinateur. A cette double justification, qui constitue l'essentiel du programme, nous avons ajouté quelques routines et proposé à titre d'exercice complémentaire l'introduction de routines de pagination et centrage automatique des titres.

Pour comprendre les différentes fonctions attribuées au programme, on se reportera à la figure (II-5), qui représente le même texte avant traitement et après traitement. Il s'agit en l'occurrence, et pour ne pas changer de centre d'intérêt, des premiers articles de

la célèbre loi française sur l'informatique et les libertés du 6 janvier 1978.

L'informatique doit être au service de chaque citoyen. Son développement doit s'opérer dans le cadre de la coopération internationale. Elle ne doit porter atteinte ni à l'identité humaine, ni aux droits de l'homme, ni à la vie privée, ni aux libertés individuelles ou publiques.

Aucune décision de justice impliquant une appréciation sur un comportement humain ne peut avoir pour fondement un traitement informatisé d'informations donnant une définition du profil ou de la personnalité de l'intéressé.

Toute personne a le droit de connaître et de contester les informations et les raisonnements utilisés dans les traitements automatisés.

L'informatique doit être au service de chaque citoyen. Son développement doit s'opérer dans le cadre de la coopération internationale. Elle ne doit porter atteinte ni à l'identité humaine, ni aux droits de l'homme, ni à la vie privée, ni aux libertés individuelles ou publiques.

Aucune décision de justice impliquant une appréciation sur un comportement humain ne peut avoir pour fondement un traitement informatisé d'informations donnant une définition du profil ou de la personnalité de l'intéressé.

Toute personne a le droit de connaître et de contester les

Figure II-5

Vue de haut et de loin, la différence entre les deux éditions n'est pas marquée. En effet le Sinclair effectue une justification automatique en fin de ligne en coupant purement et simplement au 32ème caractère. Il en résulte que visuellement, sauf si le caractère de fin de ligne est un espace, la marge à droite paraît régulière. Mais, si l'on se penche un peu sur le texte, on s'aperçoit que les mots qui se trouvent à cheval sur

les fins de ligne sont tronqués d'une manière toute à fait arbitraire et non conforme aux règles de l'orthographe. De plus, la justification à gauche n'est pas garantie si le premier caractère de la ligne est lui-même un espace. Enfin les caractères du ZX-Spectrum sont ce qu'ils sont, c'est-à-dire dessinés dans une matrice carrée (et non rectangulaire avec davantage de points matriciels en hauteur qu'en largeur comme dans les ordinateurs spécialisés dans le traitement de texte): ces caractères ne sont en définitive pas très jolis, l'ensemble de l'édition paraît un peu dense et le texte gagnerait à être aéré.

L'édition du texte après traitement est, elle, manifestement plus aérée, puisque le principe de base consiste à insérer dans chaque ligne un certain nombre de blancs de façon que la troncature s'effectue entre deux mots et non pas au cœur d'un mot.

La solution de l'insertion des blancs est classique et tous les programmes de traitement de texte l'incorporent. Mais une difficulté particulière surgit à cause des particularités du ZX-Spectrum : cet ordinateur ne traite qu'un écran de 32 colonnes, tandis qu'il est courant, pour les ordinateurs de traitement de texte, de travailler sur 80 colonnes. Bien que la longueur moyenne des mots en français soit plus longue que dans d'autres langues, comme l'anglais, si longs que soient les mots qu'il faut reporter sur la ligne suivante en gagnant des blancs, lorsque l'on dispose de 80 colonnes il est très facile de trouver entre les mots de la ligne assez d'espace à agrandir pour équilibrer cette ligne. En revanche, lorsque l'on n'a que 32 colonnes, on risque, en utilisant exclusivement la méthode de l'agrandissement des espaces, d'aboutir à une édition un peu trop dispersée : c'est ce que proposent certains programmes de traitement de texte pour le ZX81 ou le ZX-Spectrum, qui n'aboutissent pas à un résultat satisfaisant.

D'où l'idée suivante : en présence d'un mot de fin de ligne trop long pour être intégralement reporté sur la ligne suivante sans risquer de porter atteinte à la continuité visuelle du texte, le programme va proposer à l'utilisateur d'insérer un tiret sur la lettre de son choix afin de provoquer une troncature entre deux syllabes.

Le résultat est une édition aérée, sans être dispersée, à laquelle l'insertion de tirets apporte un petit air professionnel indubitable. L'utilisateur en possession

de l'interface 232 destinée à connecter le ZX-Spectrum à une imprimante, il sera en mesure de réaliser, en juxtaposant deux colonnes, des pages d'une grande "propreté" du type présentation dictionnaire ou revue.

Fonctionnement du programme de traitement de texte (figure II-6)

La première idée est la suivante : le texte à éditer est entré dans la mémoire programme, paragraphe par paragraphe, dans autant d'instructions **DATA**, ceci pour bénéficier de toutes les possibilités d'édition et de correction de l'ordinateur lui-même. S'il existe une faute dans le texte ou que l'on souhaite insérer un terme supplémentaire ou changer un terme pour un autre, il suffit par la fonction d'édition normale de rappeler l'instruction considérée pour la modifier directement au clavier. La plupart des programmes de traitement de texte possèdent ces fonctions mais elles sont beaucoup plus lourdes et complexes à organiser si le texte n'est pas en mémoire programme mais en mémoire variable. Il est à noter que par le biais des fonctions d'édition de l'ordinateur il est non seulement possible de corriger l'intérieur d'un paragraphe mais aussi de supprimer celui-ci. La suppression d'un paragraphe se fait par la suppression directe de l'instruction **DATA** qui le contient et l'insertion d'un nouveau paragraphe se fait par l'addition, sur la ligne choisie, d'une seconde instruction **DATA** puisque l'une des particularités les plus intéressantes du ZX-Spectrum est d'accepter plusieurs instructions par ligne.

```

100 REM TRAITEMENT DE TEXTE
101 REM -----
102 REM
103 DATA "Paragraphe"
104 DATA "Paragraphe"
105 DATA "Paragraphe"
106 DATA "Paragraphe"
107 DATA "Paragraphe"
108 REM etc.
109 REM etc.
110 REM etc.
111 REM
800 REM NOUVEAU PARAGRAPHE
801 REM
810 READ N$
830 LET N=1
890 REM
900 REM NOUVELLE LIGNE
901 REM
905 LET D=0: LET A=0
910 LET P$=N$(N TO )
990 REM
1000 REM SITUATION FIN DE LIGNE
1001 REM

```

```

1120 IF LEN P$<=32 THEN LET L$=P$
1130 LET D=1: GO TO 4100
1140 LET R=0
1160 IF P$(33-R)=" " THEN GO TO 1300
1170 LET R=R+1
1180 GO TO 1160
1190 REM
1300 REM POSSIBILITES
1301 REM
1305 IF R=0 THEN LET L$=P$( TO 3
2): GO TO 4100
1310 LET B=0: LET P=0
1315 LET K=0: LET K1=0
1320 FOR X=3 TO 31-R
1330 IF P$(X)=" " THEN LET B=B+1
1335 IF CODE P$(X)>=33 AND CODE
P$(X)<=64 THEN LET P=P+1
1340 NEXT X
1350 IF R<=B+P THEN GO TO 2000
1360 IF R>B+P THEN GO TO 3000
1900 REM
2000 REM INSERTION DES BLANCS
2001 REM
2050 DIM D$(32): DIM L$(32)
2060 IF P=0 THEN LET D$=P$( TO 3
2): GO TO 2200
2070 LET D$( TO 2)=P$( TO 2)
2080 LET L$( TO 2)=P$( TO 2)
2110 FOR J=3 TO 32-R
2120 LET D$(J+K)=P$(J)
2130 IF CODE P$(J)<=64 AND CODE
P$(J)>=33 AND K<P AND K<R THEN
GO SUB 2300
2150 NEXT J
2160 IF R-P=0 THEN GO TO 4000
2200 LET L$( TO 2)=D$( TO 2)
2210 FOR J=3 TO 32-R+K
2220 LET L$(J+K1)=D$(J)
2230 IF D$(J)=" " AND K1+K<R THE
N GO SUB 2400
2250 NEXT J
2260 GO TO 4100
2300 LET K=K+1
2310 LET D$(J+K)=" "
2320 RETURN
2400 LET K1=K+1
2410 LET L$(J+K1)=" "
2420 RETURN
2900 REM
3000 REM METTRE UN TIRET
3001 REM
3050 LET A=1
3100 INPUT (P$(25 TO 32);"-Tiret
sur: ";P$(33-A);"? 0/N");R$
3200 IF A$="0" THEN GO TO 3250
3210 LET A=A+1
3220 GO TO 3100
3300 LET P$(34-A)=" ": LET P$(33
-A)="-"
3350 GO TO 1140
3400 REM
4000 REM IMPRESSION LIGNE
4001 REM
4005 PRINT D$: GO TO 4200
4100 PRINT L$
4200 IF D=1 THEN GO TO 800
4350 LET N=N+33-R-2*(A<>0)
4400 GO TO 900: REM RETOUR

```

Figure II-6

C'est donc ainsi que débute le programme, en prévoyant entre les lignes 100 à 800 une place considérable pour ce type de texte. En fait, en évaluation minimale avec le ZX-Spectrum 16 K, il est possible de gérer sans problème un texte de 6000 caractères, soit en gros 4 pages de livre normal (figure II-7).

Puis l'ordinateur procède à la segmentation en prenant les 32 premiers caractères de chaque paragraphe. Il analyse d'abord la situation en fin de ligne pour voir de combien de caractères il faut faire remonter le texte pour arriver au premier blanc : il va falloir "dilater" le texte de la ligne de N caractères + 1 blancs. Il va alors calculer si la "dilatation" est possible (lignes 1300 et suivantes) et, pour ce faire, analyser le texte de la ligne en repérant :

- premièrement, le nombre de ponctuations
- deuxièmement, le nombre d'espaces déjà existants.

(B = le nombre d'espaces, P = le nombre de ponctuations), ceci afin d'établir une **hiérarchie** dans l'insertion des blancs.

En effet, *a priori*, les espaces qu'il est le plus logique d'augmenter sont ceux qui suivent immédiatement un signe de ponctuation. Si le nombre de blancs à trouver est inférieur aux signes de ponctuation, seuls les blancs qui suivent ceux-ci seront augmentés à concurrence du nombre de blancs à trouver. Si en revanche il y a plus de blancs que de signes de ponctuation, il faudra revenir sur la ligne pour augmenter le premier blanc "libre". Enfin, troisième hypothèse, si le nombre des caractères à remonter est supérieur à la somme des blancs et des signes de ponctuation, ce qui signifie que, même en augmentant chaque espace libre de une case et chaque espace suivant un signe de ponctuation de deux cases, on n'arrive pas au nombre de caractères nécessaires, alors il faudra déclencher la procédure de demande d'insertion de tirets.

Cette procédure d'insertion de tirets est interactive, elle fait l'objet de la routine qui commence à la ligne 3000. On a renoncé ici à l'utilisation de la fonction **INKEY\$**, trop rapide, pour une introduction systématique de **INPUT**, qui permet à l'utilisateur de se concentrer davantage sur la lettre à choisir pour ne pas commettre d'erreurs d'orthographe.

Enfin, une fois la transaction effectuée sur la ligne, celle-ci est envoyée à une routine d'impression, très

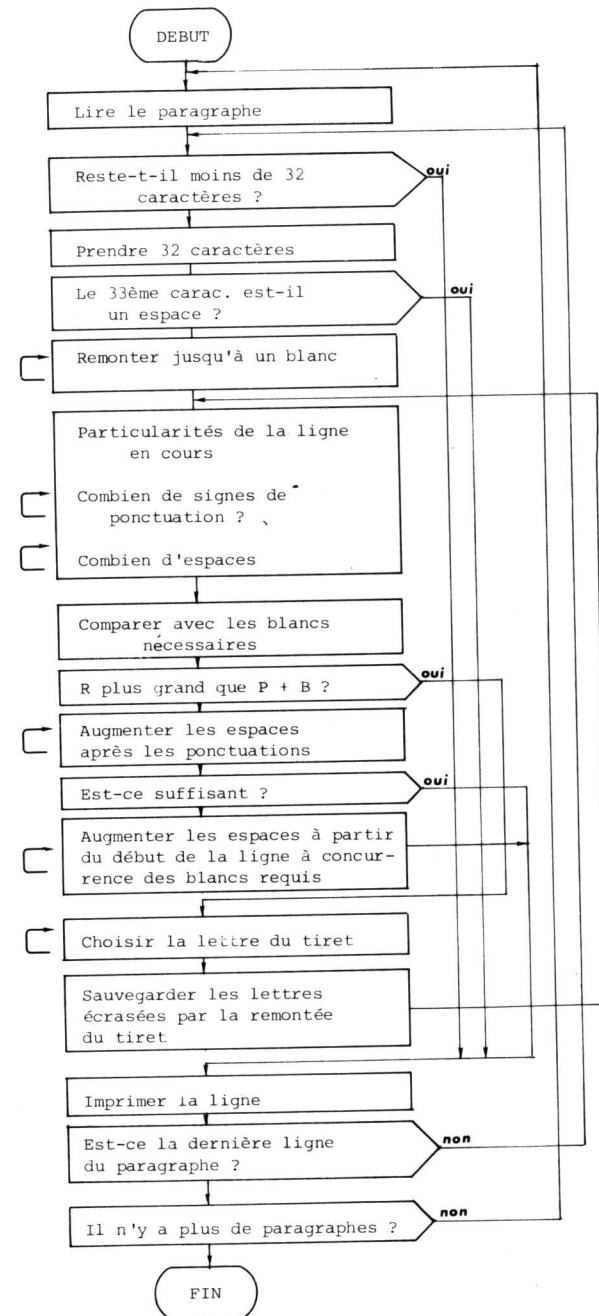


Figure II-7

simple, qui commence à la ligne 4000, et l'on passe à l'étude des 32 caractères suivants.

L'ordinateur travaille constamment sur deux chaînes de caractères, une première chaîne N\$ absorbant l'intégralité du paragraphe, une deuxième chaîne P\$ représentant la portion du paragraphe en cours d'examen, c'est-à-dire chaque fois le paragraphe N\$ amputé au début de ce qui a déjà été imprimé.

Détail de quelques instructions

On notera à ce propos la façon tout à fait originale dont le Sinclair traite les chaînes, façon illustrée par les instructions 1305, 2070, 2080, 2200 etc... La fonction de découpage de chaîne (M TO N) peut se passer de l'un des deux arguments lorsque le découpage est commandé à partir d'une extrémité, ce qui est une simplification intéressante par rapport au Basic du ZX 81. Lorsque l'on commande le découpage à partir de l'extrémité gauche de la chaîne, il est possible d'omettre le 1 (instruction 1305). Lorsque l'on commande le découpage à partir d'un indice jusqu'à la fin de la chaîne, il est inutile de faire suivre TO du numéro d'emplacement du dernier caractère de la chaîne ; cette deuxième facilité est plus intéressante que la première car, dans bien des cas, elle dispense de calculer la longueur de la chaîne ; c'est cette facilité que nous utilisons ici, en particulier dans l'instruction 910 où l'on commande la création d'une chaîne à partir d'un certain caractère d'un paragraphe jusqu'à la fin du paragraphe sans être obligé de calculer la longueur de celui-ci.

L'instruction 2630 se réfère à la table générale des codes ASCII pour reconnaître si un caractère appartient à la plage des signes de ponctuation. Même chose pour la ligne 1335 qui décompte l'ensemble des signes de ponctuation sur une ligne.

Les lignes 3200 et suivantes ont une raison d'être importante : il faut pouvoir mettre en réserve, en cas de tiret, les caractères qui ont été écrasés afin qu'ils se retrouvent au début de la ligne suivante.

Explication des différentes variables

N\$: paragraphe en cours.

N : emplacement dans le paragraphe de la première

lettre de la ligne en cours.

P\$: portion du paragraphe en cours restant à examiner.

R : nombre de caractères à remonter sur la ligne en cours pour créer autant d'espaces.

B : nombre de blancs existant originellement dans la ligne en cours.

P : nombre de signes de ponctuation.

D\$, L\$: chaînes de travail.

K, Kl, X, J : compteurs divers.

D : drapeau de fin de paragraphe (D=1 signifie nouveau paragraphe).

A : nombre de caractères écrasés par le tiret.

EXERCICE 2.2

Ajouter au programme deux nouvelles routines, une première commandant la copie automatique sur l'imprimante toutes les 22 lignes, une seconde permettant la transformation d'un paragraphe de moins de deux lignes en un titre par la simple insertion de la lettre minuscule "t" en début de paragraphe (figure II-8).

LA LEGISLATION INFORMATIQUE ET LIBERTES.

L' informatique doit etre au service de chaque citoyen. Son developpement doit s'operer dans le cadre de la cooperation internationale. Elle ne doit porter atteinte ni a l' identite

Figure II-8

LES FICHIERS DU ZX-SPECTRUM

Les fichiers sont évidemment l'une des grandes indications du traitement de données par ordinateur. Il était donc nécessaire d'inclure dans ce chapitre sur le traitement de chaînes une illustration de ce type.

Cependant il faut avoir à l'esprit que le ZX-Spectrum, en dépit de sa grande rapidité par rapport au ZX81, souffre d'un certain nombre de faiblesses dans le domaine du traitement de fichiers. Premièrement, le ZX-Spectrum manque de capacité dans sa version de base, 16 K. En effet, sur ces 16 K de mémoire accessible à l'utilisateur (RAM), presque la moitié est mobilisée par la gestion des couleurs et de la haute définition graphique, éléments de l'ordinateur qui ne sont pas directement sollicités par les applications de fichier : ce qui ne laisse pour le stockage durable des données proprement dites que 9 K, étant entendu que le programme de fichier lui-même doit aussi tenir dans ces 9 K. Il est certain que, du point de vue de la capacité, le ZX-Spectrum ne sera à la hauteur de ses possibilités en matière de fichier qu'avec l'extension de mémoire 48K et une ou plusieurs unités de microdisquettes (microdrives).

Ensuite l'organisation même du stockage de données, basée sur l'utilisation systématique de l'instruction de dimensionnement **DIM** qui fait de la place en mémoire pour les données qui vont être introduites, implique que tous les enregistrements successifs rangés dans un même tableau soient de même longueur. Les enregistrements plus longs que la dimension prévue sont tronqués, les enregistrements plus courts sont allongés selon la méthode dite de l'"affectation de Procuste" qui était déjà propre au ZX81. Il en résulte que, sauf pour les enregistrements codés qui peuvent être tous de la même longueur, la gestion de tableaux indicés, qui est la méthode la plus simple en matière de fichiers, ne peut être considérée comme la méthode de base et qu'il faut automatiquement y associer, à partir du moment où l'on a affaire à des enregistrements de longueurs variables, des routines plus ou moins complexes de gestion de chaînes dans lesquelles les divers enregistrements sont placés les uns à la suite des autres pour éviter toute perte, et de place et de texte.

Nous renvoyons volontiers le lecteur à notre ouvrage sur le ZX81 dans lequel un développement est consacré à des exemples des différentes méthodes possibles pour articuler ainsi chaînes et tableaux dans des programmes de fichiers adaptés à l'utilisateur; Nous nous contenterons de développer ici un seul exemple, orienté vers la recherche de l'encombrement mémoire minimum, dans lequel sont utilisées simultanément quatre techniques de base essentielles qu'il est nécessaire d'étudier à fond pour dépasser le stade de l'amateurisme en matière

de fichier, vis-à-vis d'un ordinateur aux performances élevées.

L'organisation du fichier

Le fichier que nous proposons ici, très simple et très efficace, est relativement "rustique", en ce sens que pour éviter toute perte de place il a été dépouillé de toutes les routines dites "de confort" (validation, vérification, alertes diverses pour renseigner sur l'écrasement des enregistrements situés aux extrémités du tableau, etc.). Seuls des commentaires d'**INPUT** relativement simples renseignent l'utilisateur sur l'opération à mener et le fichier ne comprend qu'un seul menu principal qui gère les trois routines essentielles de création, recherche et visualisation.

Dans tout programme de fichier figure un sous-programme important qui est celui de la suppression. On choisit ainsi de supprimer un certain enregistrement pour faire de la place à un enregistrement jugé plus important. Ce programme d'effacement ne figure pas dans la formule de base, mais est proposé à titre d'exercice qu'il sera loisible au lecteur de réintroduire si la réponse lui paraît convenir.

Les différents éléments du fichier sont les suivants :

- **un tableau** dans lequel vont venir se ranger les différents mots-clés en fonction de l'ordre alphabétique (figure II-9);

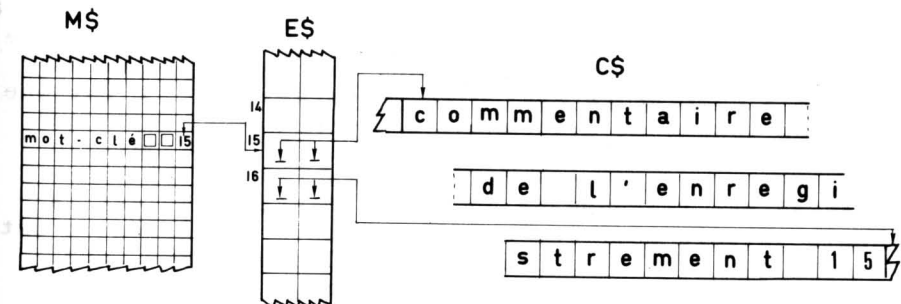


Figure II-9

- **une chaîne unique**, très longue, dans laquelle les divers enregistrements se suivent.

Chaque mot clé est suivi du numéro du ou des enregistrements auquel il se réfère, cela dans les limites ren-

dues possibles par la dimension du tableau ; un mot-clé bref et sans doute plus courant pourra être suivi par un plus grand nombre de références qu'un mot très long et probablement pour cette raison moins usité. Il ne s'agit là bien entendu que de présomptions mais c'est une méthode pour éviter les blancs qui suivent les mots brefs. C'est ce tableau de mots-clés qui sera testé au moment de la recherche. La chaîne des enregistrements/commentaires qui correspondent aux différents mots-clés est inerte, elle ne peut être que lue.

L'interfaçage entre le tableau des mots-clés et la chaîne des commentaires est effectué par l'intermédiaire d'un tableau littéral comportant des pointeurs codés. Chaque ligne de ce tableau correspond à un numéro d'enregistrement et contient le numéro de l'octet de départ du commentaire dans la chaîne.

Avant d'entrer dans le détail du fonctionnement du fichier, il faut consacrer une digression importante à la seule sophistication dont ce fichier est l'objet : le recours à la **méthode de classement dichotomique** tant pour la création que pour la recherche, qui accélère considérablement les opérations par rapport à une simple méthode séquentielle.

Le principe de la procédure dichotomique

Si l'on demande à l'ordinateur de rechercher un certain enregistrement dans une suite, la démarche la plus simple consistera à initialiser sa recherche à l'enregistrement **numéro 1** et à lui demander de comparer tour à tour tous les enregistrements qui se suivent jusqu'à ce qu'il tombe sur celui recherché. Cette méthode, très simple à comprendre, est dite "**séquentielle**". C'est une méthode longue et évidemment d'autant plus longue que la base de données est plus importante. Il s'agit en revanche d'une méthode simple à programmer. Si cette méthode est fréquemment en usage. C'est que c'est la seule valable pour les enregistrements **non triés**, c'est-à-dire qui n'ont pas été rangés selon un ordre, que cet ordre soit alphabétique, numérique etc...

Pour pouvoir gagner beaucoup de temps dans une recherche, il faut pouvoir **trier** les enregistrements au moment de la constitution de la base. Au moment de la recherche, le temps gagné sera considérable car une série de tests de combinaison "emboîtés" réduira massivement le nombre des comparaisons nécessaires en

orientant rapidement l'ordinateur vers la région intéressante.

Le lecteur qui voudrait se rendre compte de la différence de rapidité peut introduire le court programme relatif à la méthode dichotomique qui lui permettra de surcroît d'en comprendre le principe puisque la routine a été réintroduite avec de légères modifications dans le programme de fichier. (*figure II-10*).

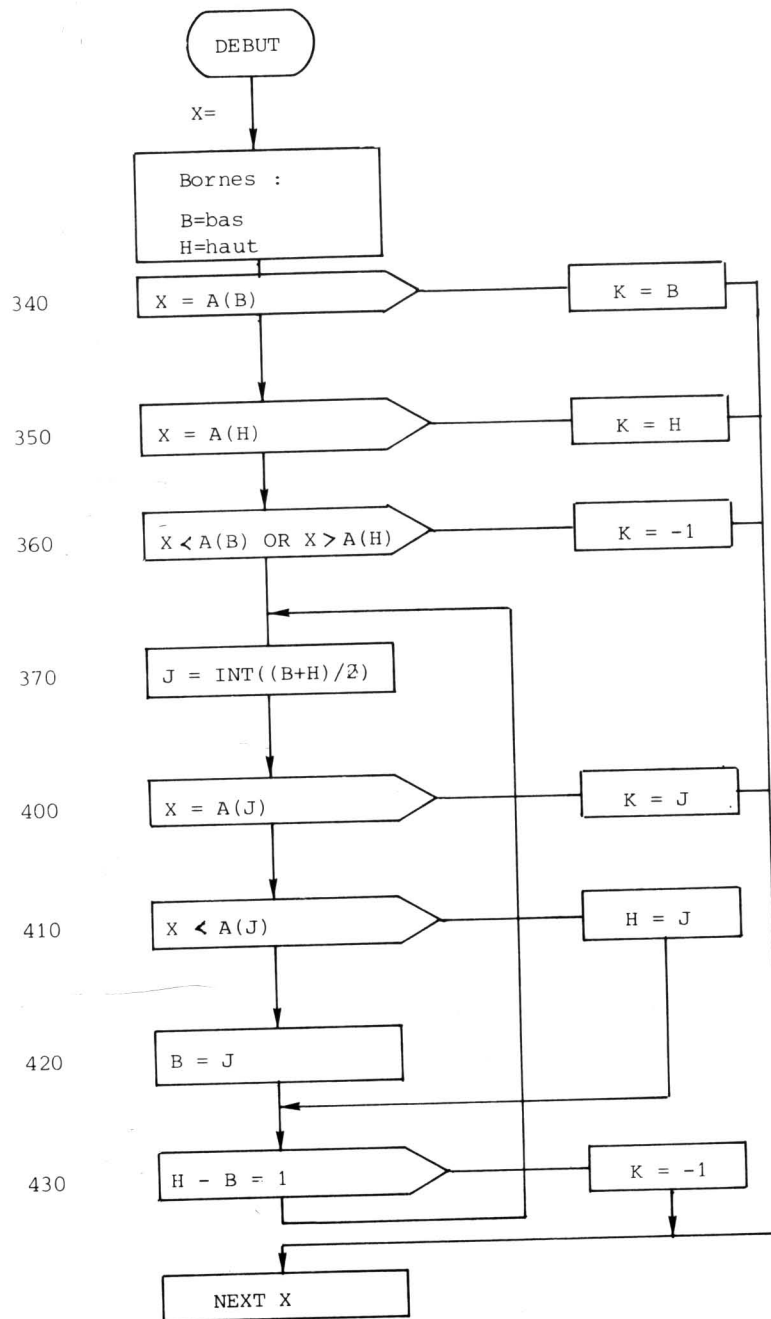
```

100 REM GENERATION DE NOMBRE
110 REM
120 DIM A(100)
130 FOR I=1 TO 100
140 LET A(I)=INT (I*10+RND*9)
150 NEXT I
160 REM
200 REM METHODE SEQUENTIELLE
210 REM
220 FOR X=100 TO 900 STEP 21
230 FOR I=1 TO 100
240 IF X=A(I) THEN GO TO 270
250 NEXT I
260 NEXT X
270 PRINT X: NEXT X
280 STOP
290 REM
300 REM METHODE DICHOTOMIQUE
310 REM
320 FOR X=100 TO 900 STEP 21
330 LET B=1: LET H=100
340 IF X=A(B) THEN LET K=B: GO
TO 450
350 IF X=A(H) THEN LET K=H: GO
TO 450
360 IF X<A(B) OR X>A(H) THEN LET
K=-1: GO TO 450
370 LET J=INT ((B+H)/2)
400 IF X=A(J) THEN LET K=J: GO
TO 450
410 IF X<A(J) THEN LET H=J: GO
TO 430
420 LET B=J
430 IF H-B>1 THEN GO TO 370
440 LET K=-1
450 IF K=-1 THEN NEXT X
460 PRINT X: NEXT X

```

Figure II-10

Ce programme génère 100 nombres compris entre 10 et 1000, rangés dans l'ordre croissant. Puis on demande à l'ordinateur de retrouver une trentaine de nombres (ligne 220) dans la table ainsi créée : d'abord selon la méthode séquentielle (200), puis selon la méthode dichotomique (300 et suivantes). Le lecteur n'a pas



besoin de beaucoup de patience car l'ensemble s'exécute très rapidement. La méthode dichotomique sur une table de 100 noms est environ quatre fois plus rapide. Le principe de la méthode est simple. La longueur du tableau à tester est divisée par deux avant toute comparaison. Comme l'élément à rechercher est forcément dans une moitié ou dans l'autre, il n'y a déjà plus que la moitié du tableau à tester. Dans cette moitié qui reste on procède de même pour déterminer dans quel quart du tableau se situe la valeur qui intéresse, et ainsi de suite jusqu'à ce que l'on tombe effectivement sur cette valeur. Pour donner une idée de l'efficacité de la méthode, un tableau de 1000 éléments ne peut pas donner lieu à plus de 10 divisions par deux. Les choses vont donc très vite. Les variables de ce court programme sont simples à comprendre : **B** signifie "bas", **H** signifie "haut". Si la valeur recherchée correspond à l'indice d'une de ces deux valeurs, on sort du circuit comparatif, sinon on divise le tableau (ligne 370) pour obtenir une nouvelle valeur supérieure (ligne 410), etc... La moyenne des deux limites précédentes peut à son tour devenir la limite inférieure dans un nouveau test (ligne 420) (*figure II-11*).

Il existe des méthodes de recherche plus efficaces dans une table triée, mais elles impliquent une programmation plus lourde et, pour la capacité du ZX-Spectrum, la méthode dichotomique simple suffit largement à améliorer considérablement les performances par rapport à la méthode séquentielle.

Le recours à la méthode dichotomique au moment de la recherche implique naturellement une méthode d'insertion elle-même dichotomique au moment de la création de la base. Voici pourquoi, dans le programme de fichier que nous proposons (*figure II-12*), cette procédure dichotomique est placée en début de programme en routine unique mais avec des entrées et des sorties différentes selon qu'elle intervient au moment de la création du fichier ou de la recherche d'un mot-clé. Sa présence explique en particulier qu'il soit nécessaire de recourir à un artifice au moment de la création en créant d'abord un enregistrement inutile avec deux mots clef dont la valeur en code ASCII soit plus petite (par ex: "AAAA") et plus grande (par ex: "zzzz") que respectivement le plus petit et le plus grand mot clef possible de la base. De cette manière, au moment de l'initialisation, le premier enregistrement normal viendra trouver sa place par insertion entre les deux sans que se posent des problèmes d'extrémités.

```

100 REM FICHIER
101 REM -----
102 REM
110 REM MENU PRINCIPAL
115 CLS
120 PRINT "VOULEZ-VOUS:"
121 PRINT
130 LET K$="enregistrement"
131 LET F$="fichier"
140 PRINT "-initialiser le ";F$
141 PRINT "1"
150 PRINT "-entrer un ";K$;" ?"
151 PRINT "2"
160 PRINT "-rechercher un ";K$;" ?"
161 PRINT "3"
180 PRINT "-visualiser le ";F$;" ?"
181 PRINT "5"
190 INPUT A: GO TO A*1000
500 LET Z$=CHR$ 0
510 FOR I=1 TO 14
520 LET Z$=Z$+CHR$ 0
530 NEXT I: RETURN
999 REM
1100 REM INITIALISATION
1101 REM
1110 INPUT INVERSE 1;"DIMENSION
DE LA CHAÎNE ?";C: DIM C$(C)
1111 LET FC=1: LET FM=2
1112 GO SUB 500
1115 INPUT INVERSE 1;"NOMBRE D'
ENREGISTREMENTS ?";E: DIM E$(E,2)
1116 LET FE=0
1118 LET E$(1)=CHR$ 0+CHR$ 1
1120 DEF FN Y(I)=(256*CODE E$(I,
1)+CODE E$(I,2))
1125 DEF FN X(I)=INT (I/256)
1130 DEF FN Z(I)=256*(I/256-FN X
(I))
1150 INPUT INVERSE 1;"COMBIEN DE
MOTS-CLEFS ?";N: DIM M$(N,15)
1160 LET M$(1)="#####"
1170 LET M$(2)="#####"
1180 GO TO 2000
1200 GO SUB 1500
1210 GO SUB (1700-100*S)
1220 RETURN
1300 GO SUB 1500
1310 GO SUB (1900-100*S)
1320 RETURN
1400 REM
1500 REM PROC.DICHOTOMIQUE
1501 REM
1510 LET D=1: LET F=FM
1515 LET K=0: LET S=0
1520 IF A$=M$(D, TO L) THEN LET
K=D: RETURN
1530 IF A$=M$(F, TO L) THEN LET
K=F: RETURN
1540 LET K=INT ((D+F)/2)
1550 IF A$=M$(K, TO L) THEN RETU
RN
1560 IF A$<M$(K, TO L) THEN LET
F=K: GO TO 1540
1570 LET D=K
1580 IF F-D>1 THEN GO TO 1540
1590 LET S=1: RETURN
1599 REM
1600 REM DECALAGE-INSERTION
1601 REM
1605 LET FM=FM+1
1610 FOR J=FM TO K+2 STEP -1

```

Figure II-12

```

1620 LET M$(J)=M$(J-1)
1630 NEXT J
1640 LET M$(K+1, TO L)=A$: LET M
$(K+1, L+1 TO )=Z$(L+1 TO )
1690 REM
1700 REM ADDITION REFERENCE
1701 REM
1710 FOR J=15 TO 1 STEP -1
1720 IF M$(K+S,J)<>CHR$ 0 THEN N
EXT J
1730 IF M$(K+S,J-1)<>CHR$ 0 THEN
RETURN
1740 LET M$(K+S,J)=CHR$ (FE+1)
1750 RETURN
1800 PRINT "NON TROUVE": PAUSE 2
00: GO TO 100
1890 REM
1900 REM ENVOI IMPRESSION
1901 REM
1910 FOR J=15 TO 1 STEP -1
1920 IF M$(K,J)=CHR$ 0 THEN RETU
RN
1930 LET P0=CODE M$(K,J): GO SUB
5100
1940 NEXT J: RETURN
1990 REM
2000 REM CREATION
2001 REM
2100 INPUT INVERSE 1;"COMBIEN D
E MOTS-CLEFS POUR L'ENREGISTREME
NT ";FE+1;" ?";A
2120 FOR I=1 TO A: INPUT (I;"")
:A$: LET L=LEN A$
2150 GO SUB 1200
2160 NEXT I
2505 LET FE=FE+1
2510 INPUT INVERSE 1;"ENREGISTR
EMENT NO. ";FE):B$
2520 LET L1=LEN B$
2530 LET P1=FN Y(FE)
2560 LET C$(P1 TO P1+L1-1)=B$
2570 LET P0=FE+1
2575 LET P1=P1+L1
2576 LET FC=FC+P1
2580 LET E$(P0)=CHR$ FN X(P1)+CH
R$ FN Z(P1)
2590 GO TO 100
2900 STOP
2990 REM
3000 REM RECHERCHE
3001 REM
3100 INPUT INVERSE 1;"MOT-CLEF ?
":A$: LET L=LEN A$
3200 GO SUB 1300
3300 GO TO 100
3900 STOP
5000 REM VISUALISATION
5001 REM
5010 GO TO 5200
5110 LET P1=FN Y(P0)
5120 LET P2=FN Y(P0+1)-1
5130 PRINT C$(P1 TO P2)
5135 INPUT INVERSE 1;"Voulez-vo
us l'";K$;" SUIVANT ? (O/N) ";Q$
5136 IF Q$="N" THEN GO TO 100
5140 RETURN
5200 FOR P=1 TO FE
5220 LET P0=P
5250 GO SUB 5100
5260 NEXT P

```

Figure II-12

Analyse du programme

Le programme commence par un menu qui commande chacune des routines principales (ligne 190). L'emplacement a été prévu pour une routine n° 4 d'effacement qui sera donnée plus loin en exercice. La routine d'initialisation (ligne 1100) est composée des différents dimensionnements, de la chaîne, du tableau des enregistrements et du tableau des mots-clés, de la définition de trois fonctions, de conversion de décimales en code ASCII et inversement, ces différentes fonctions étant destinées à la manipulation des pointeurs de début d'enregistrement dans la chaîne. La fonction Y transforme une chaîne de deux caractères ASCII en un nombre décimal, les fonctions X et Z font l'inverse pour transformer un nombre décimal en une chaîne littérale de deux octets, X pour le premier octet, Z pour le second octet.

Après l'initialisation sont placées les routines les plus fréquemment appelées par l'ordinateur, de façon à lui faire perdre le moins de temps possible dans le transfert. Il s'agit pour l'essentiel de la procédure dichotomique qui va être utilisée tant au stade de la création qu'à celui de la recherche. Le résultat du passage est différent selon que l'on est dans le stade de la création ou de la recherche :

- **au stade de la création** : si le mot-clé introduit pour servir de clé à un enregistrement n'existe pas déjà dans le tableau parce que l'ensemble des comparaisons a été épuisé sans que le mot ait été reconnu, il y a passage immédiat à la routine de **décalage-insertion**. Une fois le mot installé à sa place, le numéro de l'enregistrement auquel il doit renvoyer est codé dans la même chaîne que lui en commençant par la fin, par la routine **d'addition-référence**. Dans le cas où le mot aurait été reconnu, la routine d'addition-référence est immédiatement appelée après la procédure dichotomique.

- symétriquement, lorsqu'il s'agit **du stade de la recherche**, si la procédure dichotomique est épuisée sans reconnaissance, la routine 1800 est appelée (non trouvé). En revanche, en cas de reconnaissance, le mot est lu, la référence est retrouvée dans sa chaîne et la routine d'impression appelée pour imprimer l'enregistrement afférent.

La même procédure ayant ainsi deux entrées selon la fonction et quatre sorties différentes selon le résultat, il est nécessaire de la faire précéder d'une batterie d'aiguillage : c'est l'objet des différentes instructions

1200, 1210, 1300, 1310 (on a utilisé S comme drapeau).

Cette routine d'analyse dichotomique effectuant l'essentiel du travail, tant au stade de la création qu'au stade de la recherche, il n'est pas surprenant que les routines de création et de recherche proprement dites (lignes 2000 et 3000) soient brèves. L'essentiel de la routine de création consiste à actualiser une série de compteurs avec la manipulation de variables dont le détail sera examiné ci-dessous. La routine de **visualisation** (lignes 5000 et suivantes) peut être appelée de deux manières, soit par programme, pour éditer un ou plusieurs enregistrements à l'issue de la procédure de recherche sur mots-clés, soit directement pour l'ensemble du fichier séquentiellement si l'utilisateur le désire.

Le travail de ce sous-programme est simple : il repère dans le tableau des pointeurs le début et la fin de l'enregistrement concerné et affiche la partie de la chaîne **C%** qui va du premier au second pointeur, en vertu d'une technique symétrique de celle qui a lieu au stade de la création (ligne 2560). Cette ligne 2560 appelle un bref commentaire : il est de loin préférable de procéder à la création d'une partie de la chaîne par l'utilisation de la fonction de découpage et non par une simple concaténation du type **LET C% = C% + B%** ; en effet ce type d'instruction provoque en mémoire le dédoublement de la chaîne de base pour pouvoir effectuer l'addition, ce qui pourrait rapidement conduire à un dépassement de capacité si la chaîne choisie est longue.

Détail des variables

- **Z%** : ligne vide créée par les lignes 500 et suivantes. En effet, contrairement au Basic du ZX-81, l'espace ne correspond pas à la valeur 0, mais à la valeur 32. La présence d'un octet non occupé pourrait laisser penser qu'il s'agit de l'enregistrement 32 ; il faut donc créer une ligne vide avec le code représentatif de l'absence 0.
- **FC** : pointeur de fin de chaîne.
- **FM** : pointeur de fin de tableau des mots-clés.
- **C%** : chaîne des enregistrements.
- **E%** : tableau des pointeurs d'enregistrement.
- **FE** : pointeur de fin du tableau des enregistrements.
- **M%** : tableau des mots-clés.

Les autres variables sont des variables de travail provisoires : I, J, K, H, K 1, L, P1, P2, P0, etc...

EXERCICE 2.3

Construire une routine de suppression d'un enregistrement. Cette routine devra :

- 1) demander le numéro de l'enregistrement à supprimer ;
- 2) rechercher dans le tableau des pointeurs E\$ le début et la fin de l'enregistrement dans la chaîne C\$;
- 3) faire remonter la fin de la chaîne C\$ d'un nombre d'octets correspondant aux octets supprimés ;
- 4) soustraire ce nombre d'octets de tous les pointeurs qui, dans le tableau des pointeurs, sont situés postérieurement aux pointeurs de l'enregistrement supprimé ;
- 5) prévoir une routine qui analyse séquentiellement l'ensemble des références du tableau des mots-clés pour :
 - supprimer l'enregistrement concerné là où il se trouve
 - soustraire d'une unité toutes les références plus élevées, puisque les enregistrements d'indices plus élevés sont tous descendus d'une unité.

Il sera bon, pour gagner du temps, de faire tester le tableau des mots-clés par la "droite" plutôt que par la "gauche" en faisant sauter au mot-clé suivant dès que l'on rencontre un blanc.

NB : Ne pas oublier dans le programme de fichier ci-dessus de créer un premier enregistrement avec les deux mots-clés d'extrémité (voir page 39).

CHAPITRE III

HAUTE DEFINITION GRAPHIQUE ET UTILISATION COURANTE DES FONCTIONS MATHÉMATIQUES

La haute définition graphique, jointe à la couleur et au son, fait du ZX-Spectrum un ordinateur de jeux particulièrement performant. Pourtant ce n'est pas dans le domaine "ludique" que la haute définition graphique prend toute son importance. C'est associée à l'utilisation des fonctions mathématiques qu'elle offre tout son intérêt. Gérer sur l'écran une courbe très fine après une initiation rapide à la puissance de calcul du ZX-Spectrum multiplie considérablement les possibilités du programmeur qui accepte cet effort. D'ailleurs, même si les logiciels de jeux retiennent principalement son attention, c'est en y incorporant les diverses techniques abordées dans ce chapitre qu'il obtiendra les effets les plus brillants, les plus rapides.

Ce chapitre se propose de démontrer très progressivement l'interdépendance étroite qui existe entre la haute définition et la maîtrise de quelques fonctions mathématiques en quatre temps : d'abord une initiation aux familles de courbes les plus couramment utilisées ; une introduction aux graphiques de présentation de données illustre ensuite les techniques de haute définition ; en troisième lieu un exposé rapide de la méthode, très utile en pratique, de la régression linéaire propose un exemple simple de fusion entre haute définition graphique et fonctions mathématiques ; enfin, en manière de récompense à ce chapitre un peu ardu une introduction aux représentations graphiques en trois dimensions décompose une méthode simple pour produire rapidement

ces graphiques "3D" très recherchés pour leur caractère suggestif.

LES GRANDES FAMILLES DE COURBES

Cette section purement introductive se concentre sur les quatre principales familles de courbes. Il suffit au lecteur, d'introduire le court programme ci-après pour voir se dessiner sur l'écran, chacune à son tour, une courbe représentative de l'une des quatre familles avec lesquelles le programmeur a souvent affaire pour ses effets graphiques (*figures III-1, III-2*).

$$Y = A \cdot X + B$$

Les lignes 100 et suivantes tracent une droite : à chaque valeur de X sur l'axe horizontal correspond une valeur de Y égale à la multiplication de X par une constante (et éventuellement l'addition d'une constante) ; cette forme est la plus simple : A est la pente de la courbe et B est le point où elle rencontre l'axe des Y.

La croissance exponentielle

Imaginons un deuxième cas de figure : la valeur initiale de X est multipliée par un certain coefficient ; le résultat de cette multiplication est lui-même remultiplié par le coefficient et ainsi de suite. Si le coefficient est supérieur à 1 la courbe ne fera que croître de plus en plus vite, s'il est inférieur à 1 elle diminuera de plus en plus lentement. Dans les deux cas le phénomène est essentiellement cumulatif : il s'agit d'une croissance exponentielle. On le voit à la ligne 220 : la valeur I est constamment remultipliée par un certain coefficient.

La croissance logarithmique

Regardons la courbe exponentielle ainsi tracée et essayons d'imaginer que les axes X et Y aient été intervertis, c'est-à-dire que les Y augmentent mais de moins en moins vite par rapport aux X : cette courbe est une courbe logarithmique (étroitement apparentée à la courbe exponentielle). Prenons un exemple : le logarithme décimal de 10 est égal à 1, le logarithme décimal

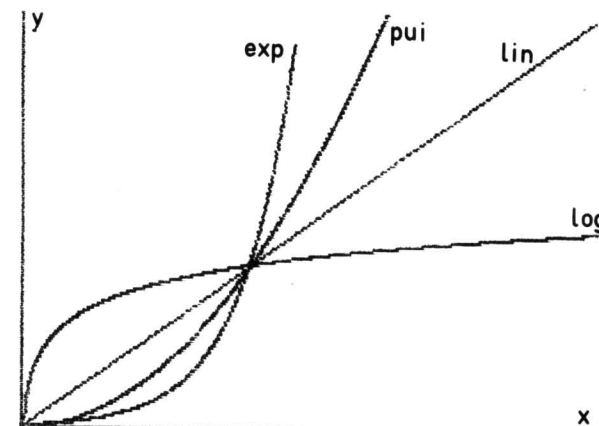


Figure III-1

```

1 REM LES FAMILLES DE COURBES
2 REM -----
10 REM AXES
20 PLOT 0,0: DRAW 255,0
30 PLOT 0,0: DRAW 0,175
40 REM
100 REM Lineaire
110 REM FOR I=1 TO 250
120 REM PLOT I,175/255*I
130 REM NEXT I
140 REM Equivalent:
150 PLOT 0,0: DRAW 250,175/255*
250
160 REM
170 REM
200 REM Exponentiel
205 LET Y=1
210 FOR I=1 TO 119 STEP .1
220 LET Y=Y*1.004315233
230 PLOT I,Y
240 NEXT I
250 REM Equivalent:
260 REM FOR I=1 TO 174: PLOT I,
1.03*I: NEXT I
290 REM
300 REM Logarithmique
310 FOR I=1 TO 255 STEP .1
320 PLOT I, LN I*15
330 NEXT I
390 REM
400 REM Puissance
410 FOR I=1 TO 200 STEP .1
420 PLOT I,I*I/145
430 NEXT I

```

Figure III-2

de 100 est égal à 2, le logarithme décimal de 1000 est égal à 3 ; c'est comme si l'on faisait correspondre à un nombre son ordre de grandeur : il y a beaucoup plus de nombres entiers entre 100 et 1000 qu'entre 10 et 100 et pourtant 101 et 999 ont le même ordre de grandeur, de même 10 et 99 (soit 2).

Les puissances

Les courbes puissances sont d'une nature différente : ce sont des croissances plus que proportionnelles mais qui ne sont pas de forme cumulative. Pour toute valeur de X la valeur correspondante de Y sera obtenue en multipliant la valeur de X par elle-même (carré) ou encore par elle-même (cube) etc...

Ce qu'il faut noter c'est que les trois dernières courbes peuvent être ramenées **graphiquement** à un schéma linéaire : la puissance 1 ramène à une droite (4 puissance 1 égale 4) ; une courbe exponentielle redonne un tracé linéaire si, au lieu de Y, on porte sur l'axe des Y le logarithme de Y, et la courbe logarithmique devient linéaire si l'on porte sur l'axe des X non pas X mais le logarithme de X.

EXERCICE 3.1

Ligne 260 : faire correspondre à I non plus $1.03 \cdot I$ mais le logarithme de cette valeur multiplié par la constante 30. Quelle est la nouvelle allure de la courbe ?

INTRODUCTION AUX GRAPHIQUES DE PRESENTATION DE DONNEES

Les applications graphiques qui vont suivre sont des graphiques au sens propre du terme, c'est-à-dire la présentation visuelle d'un certain nombre de données, généralement statistiques. L'utilité de ces différents programmes est évidente : ils peuvent être intégrés très facilement dans une grande variété de programmes plus importants, d'analyse de données ou de gestion.

En effet, quiconque a essayé d'illustrer un rapport ou une étude avec des graphiques explicites a pu remarquer le temps énorme qu'absorbe le calcul de chaque valeur et son report sur le papier ; ceci sans trop

savoir si le résultat final sera agréable à l'oeil ou même seulement explicite.

L'élaboration des graphiques par ordinateur permet, non seulement le calcul des valeurs sans la moindre faute ni le moindre oubli, mais, dans le cas où le résultat final n'est pas satisfaisant, d'apporter immédiatement une modification dans les paramètres de hauteur ou de largeur afin d'aboutir à une présentation mieux équilibrée.

Outre leur utilité en tant qu'application, le montage de graphiques offre un intérêt pédagogique certain dans le cadre d'une initiation aux possibilités du ZX-Spectrum.

En effet seule une bonne maîtrise simultanée des fonctions mathématiques d'une part et des commandes graphiques de l'ordinateur d'autre part permet de dresser un graphique un tant soit peu élaboré. Les mathématiques interviennent dans le choix des proportions, dans le calcul des paramètres des différents axes, mais surtout dans les opérations de grisé. Leur usage nécessite, spécialement dans le cas des graphiques circulaires, le maniement de formules mathématiques simples pour correctement calculer le point de départ et le point d'arrivée de chaque segment.

Nous verrons, à la suite de la section consacrée aux graphiques, une application d'un intérêt considérable qui pousse encore plus loin le mariage entre les techniques mathématiques et les présentations graphiques : il s'agit des courbes de régression. Dans ce cas, les programmes calculeront la structure des données pour voir si elles se ramènent à une courbe relativement simple et c'est cette courbe qui constitue ensuite la base des reports graphiques.

Graphique-barres et graphiques-bandes

Le graphique-barre simple

Le graphique de base le plus simple qu'on puisse imaginer consiste en la juxtaposition de barres verticales dont la hauteur est proportionnelle à la valeur de la donnée reportée. Ce type est reproduit à la *figure III-3*, qui est la traduction graphique de la statistique ci-dessous représentant les taux de croissance respectifs des capacités disques des ordinateurs de 1978 à 1986 (base 100 en 1978), tableau (*figure III-4*) comprenant donc une interpolation pour l'avenir à court terme.

(source : *Le Monde Informatique*, 17 janvier 1983).

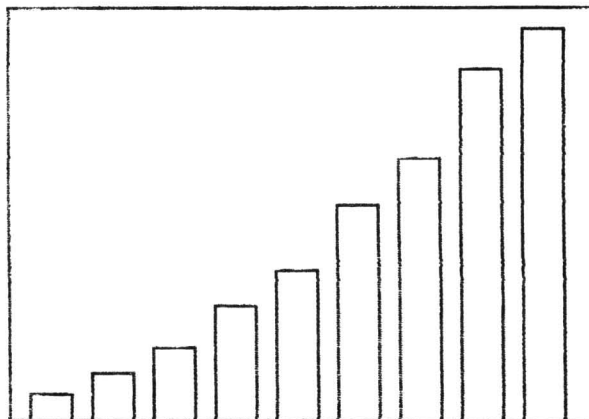


Figure III-3

Année	Disque
1978	100
1979	172
1980	274
1981	433
1982	566
1983	800
1984	968
1985	1310
1986	1461

Les taux de croissance respectifs des capacités-disques de 1978 à 1986 (source : *Le Monde Informatique*, 17 janvier 1983)

Figure III-4

Si simple qu'apparaisse l'illustration, la structure du programme est pourtant déjà complexe. Il faut en effet successivement prévoir le nombre de données pour

ouvrir un tableau (ligne 110 du programme) (figure III-5), calculer la largeur des barres et l'intervalle entre les barres sur la base : largeur M1 = 2 intervalles M.

```

10 REM GRAPHIQUES BARRES
11 REM -----
100 PRINT "Combien de données?"
110 INPUT N: DIM T(N): LET M=IN
  T (255/(3*N+1)): LET M1=2*M+INT
  ((255-M-N*(3*M))/N)
200 REM Introduction données
210 FOR I=1 TO N: PRINT "Donnée
  No: "; I: INPUT T(I): PRINT I,T(I)
  : NEXT I
300 REM Rech. + grande donnée
310 LET T1=0
320 FOR I=1 TO N
330 IF T(I)>T1 THEN LET T1=T(I)
340 NEXT I: LET T1=(175-M)/T1:
CLS
400 REM Axes
410 PLOT 0,0: DRAW 255,0: DRAW
  0,175: DRAW -255,0: DRAW 0,-175
500 FOR I=1 TO N
510 PLOT M*I+M1*(I-1),0
520 DRAW 0,T(I)*T1: DRAW M1,0:
  DRAW 0,-T(I)*T1
530 NEXT I

```

Figure III-5

Il faut prévoir un ajustement de l'épaisseur des barres pour éviter que ne se produise à droite un blanc trop important résultant de la division de la largeur de l'écran par 3 : c'est là l'objet de la quatrième instruction de la ligne 110 qui calcule, en partant de la largeur des bases, de combien de points celles-ci pourraient augmenter sans dépasser le cadre. Puis l'on procède à l'introduction des données du tableau selon une technique sans difficulté (ligne 210). Une fois le tableau intégralement entré, on recherchera quelle est la donnée la plus grande pour dimensionner le cadre en hauteur au moyen d'un tri de comparaison séquentielle très simple (ligne 330). Une fois la donnée la plus grande trouvée, elle est légèrement diminuée de la valeur d'un intervalle pour ne pas "défoncer le plafond" et l'on peut alors procéder au dessin tant du cadre (ligne 410) que des différentes barres, représentées chacune par deux segments verticaux et un segment horizontal, les segments verticaux étant commandés par les valeurs du tableau (520), et horizontaux par la largeur de chaque barre.

Les graphiques bande en pourcentage

Le type de graphique-barre illustré ci-dessus est tout à fait adapté à la présentation des séquences statistiques montrant la succession dans le temps des différentes valeurs d'une même donnée. Lorsqu'il s'agit de figurer l'importance relative de différents éléments qui composent un ensemble, il existe une présentation plus explicite encore, celle du graphique de pourcentages : une seule et même bande (barre) est divisée en sections à largeur (hauteur) proportionnelle au pourcentage de la valeur représentée par cette section dans l'ensemble. Naturellement la largeur (hauteur) totale est représentative des 100%. Il est clair que pour bénéficier de la plus grande définition d'écran par rapport aux différents pourcentages, il est plus avantageux de choisir une représentation horizontale que verticale, un peu plus écrasée. La figure III-6 traduit sur un graphique de ce type une statistique relative à la répartition des semi-conducteurs/microprocesseurs entre les 10 entreprises les plus puissantes du monde dans ce domaine, leur production globale étant quasiment équivalente à 100% du marché, appréciée non en unités mais en valeur (source : *Le Monde Informatique*, 17 janvier 1983).

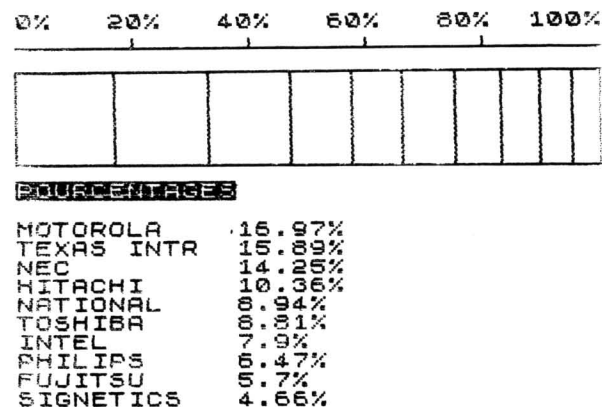


Figure III-6

Le programme (figure III-7), qui correspond à ce graphique est relativement simple. Il ouvre deux tableaux : nom et valeur de chaque observation (ligne 110), tableau représenté par la boucle 205-240 ; le cadre

global est tracé sans difficulté (ligne 310), ainsi que l'échelle de graduation (320-380) ; les sections sont ensuite définies par un trait vertical (une double boucle I et J, boucle J indispensable ayant pour effet de calculer pour chaque section la somme des sections antérieures de façon à progresser sur la droite).

```

10 REM GRAPHIQUE BANDE EN %
20 REM -----
100 PRINT "Combien de donnees?"
110 INPUT N: DIM T(N): DIM T$(N)
120: LET S=0
130 REM
140 REM Introduction donnees
150 FOR I=1 TO N
210 PRINT INVERSE 1;"Donnee no:
":I: INPUT T$(I): INPUT T(I)
220 PRINT T$(I),T(I)
230 LET S=S+T(I)
240 NEXT I
250 CLS
260 REM
270 REM Cadre
310 PLOT 0,150: DRAW 255,0: DRA
W 0,-40: DRAW -255,0: DRAW 0,40
320 PLOT 0,160: DRAW 255,0
330 PRINT AT 0,0;"0%": PRINT AT
0,28;"100%"
340 FOR I=2 TO 8 STEP 2
360 PLOT I*25,5,160: DRAW 0,4
370 PRINT AT 0,3*I-1,I*10,"%
380 NEXT I
390 REM
400 REM Sections
405 FOR I=1 TO N-1: LET X=0
410 FOR J=1 TO I: LET X=X+T(J)
NEXT J
420 PLOT 255*X/S,110: DRAW 0,40
430 NEXT I
440 REM
450 REM Tableau
505 PRINT INVERSE 1;AT 9,0;"POU
RCENTAGES"
510 FOR I=1 TO N
520 PRINT AT 10+I,0,T$(I),TAB 1
2;INT (T(I)/S*10000)/100,"%
530 NEXT I

```

Figure III-7

L'argument de la commande **DRAW** étant relatif, on aurait aussi pu se contenter d'une boucle simple demandant chaque fois un démarrage, alternativement à partir de la base ou du sommet du montant précédent. Enfin les dernières lignes du programme impriment en dessous de la bande les données du tableau.

L'introduction des grisés dans le graphique

La lisibilité des graphiques ci-dessus présentés s'accroît de façon spectaculaire lorsqu'il est possible de différencier les sections avec un grisage plus ou moins soutenu. Ce grisage, répétons-le, est en quelque sorte un succédané de couleur, laquelle est difficilement utilisable, tant à cause de l'imprimante, qui ne rend pas les couleurs, que des capacités du ZX-Spectrum lui-même, la détermination des couleurs ne s'effectuant pas point par point mais par pavés de 8 x 8 points. En revanche il est parfaitement loisible d'obtenir, pour l'intégralité de la bande, une couleur de fond différente de la couleur du reste de l'écran. En effet, la bande, dans son dessin intégral, recouvre un nombre entier de colonnes et de lignes : 5 lignes et 32 colonnes. Il suffit d'ajouter une instruction complémentaire au dessin du cadre.

La figure III-8 représente la même statistique que précédemment, mais chacune des sections a fait l'objet d'un grisage de plus en plus clair.

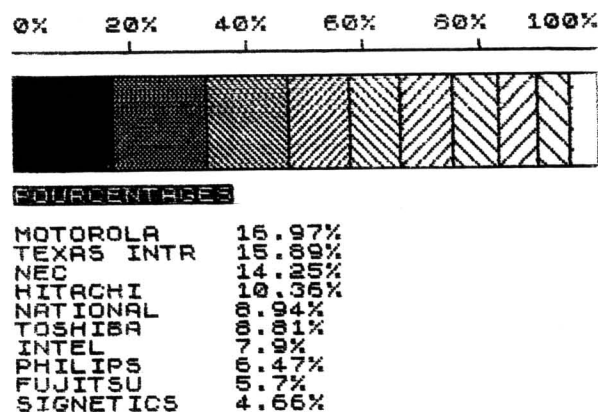


Figure III-8

La structure du programme (figure III-9) qui réalise ce type de graphique reprend dans sa première partie le programme antérieur ; sa deuxième partie (ligne 500-1090) contient les deux sous-programmes de grisage. Un seul sous-programme aurait pu suffire pour effectuer

```

10 REM GRAPHIQUE BANDE EN %
20 REM -----
100 PRINT "Combien de donnees?"
110 INPUT N: DIM T(N): DIM T$(N)
120 DIM X(N)
130 LET S=0: LET X(1)=0
140 REM
200 REM Introduction donnees
205 FOR I=1 TO N
210 PRINT INVERSE 1;"Donnee no:
";I: INPUT T$(I): INPUT T(I)
220 PRINT T$(I),T(I)
230 LET S=S+T(I)
240 NEXT I
250 CLS
260 REM
300 REM Cadre
310 PLOT 0,150: DRAW 255,0: DRA
W 0,-40: DRAW -255,0: DRAW 0,40
320 PLOT 0,160: DRAW 255,0
330 PRINT AT 0,0;"0%": PRINT AT
0,28;"100%"
340 FOR I=2 TO 8 STEP 2
350 PLOT I*25,5,160: DRAW 0,4
370 PRINT AT 0,3*I-1,I*10;"%"
380 NEXT I
390 REM
400 REM Sections
405 FOR I=1 TO N-1: LET X=0
410 FOR J=1 TO I: LET X=X+T(J):
NEXT J
415 LET X(I+1)=X/S*255
420 PLOT 255*X/S,110: DRAW 0,40
425 GO SUB 500+(I/2-INT (I/2))*
1000
430 NEXT I
450 GO TO 1500
460 REM
500 REM Grisage-0
510 LET L=X(I+1)-X(I)
520 FOR K=110 TO 150 STEP I
525 LET Q=150-K: LET L1=L*(Q)=L
)+0*(Q<L)
530 PLOT X(I),K: DRAW L1,L1
540 NEXT K
550 FOR X=X(I) TO X(I+1) STEP I
555 LET L=X(I+1)-X: LET L1=L*(L
<40)+40*(L>=40)
560 PLOT X,110: DRAW L1,L1
570 NEXT X: RETURN
580 REM
1000 REM Grisage-1
1010 LET L=X(I+1)-X(I)
1020 FOR K=150 TO 110 STEP -I
1025 LET Q=K-110: LET L1=L*(Q)=L
)+0*(Q<L)
1030 PLOT X(I),K: DRAW L1,-L1
1040 NEXT K
1050 FOR X=X(I) TO X(I+1) STEP I
1055 LET L=X(I+1)-X: LET L1=L*(L
<40)+40*(L>=40)
1060 PLOT X,150: DRAW L1,-L1
1070 NEXT X: RETURN
1080 REM
1500 REM Tableau
1505 PRINT INVERSE 1;AT 9,0;"POU
RCENTAGES"
1510 FOR I=1 TO N
1520 PRINT AT 10+I,0:T$(I);TAB 1
2;INT (T(I)/S*10000)/100;"%"
1530 NEXT I

```

Figure III-9

des grisés progressivement plus clairs en augmentant à chaque fois le pas de la trace oblique, mais, pour améliorer la qualité générale de l'effet, et surtout pour mieux différencier les sections très claires (lorsque le pas devient important la distinction est plus délicate à faire), on a trouvé intéressant de modifier alternativement l'orientation de la trace oblique ; d'où, pour simplifier, deux sous-programmes : un sous-programme de grisage "0" correspondant aux sections paires (traces de bas en haut) et un grisage "1" correspondant aux sections impaires (grisage de haut en bas), respectivement lignes 500 et suivantes, lignes 1000 et suivantes, commandées par l'instruction de transfert 425. Ces deux routines sont modulairement identiques ; elles ne diffèrent que par les paramètres d'entrée, haut ou bas, etc...

Il est parfaitement possible de fondre ces deux routines en une routine unique à l'aide de fonctions logiques, ce qui est d'ailleurs proposé au lecteur à titre d'exercice un peu plus loin. Pour chaque section, indépendamment du tracé de grisage, le grisage s'effectue en deux temps : d'un coin gauche de la section (coin supérieur ou inférieur), le grisage va d'abord vers le côté opposé de la bande, puis, à partir du même coin, rejoint le début de la section suivante. On notera que, le tracé s'effectuant à 45 degrés, les arguments de la fonction DRAW ont des valeurs absolues identiques. La longueur du trait est soumise à une double limitation : prendre la plus petite des deux distances avec le bord ou la séparation de la section suivante.

EXERCICE 3.2

Construire, en faisant une utilisation judicieuse des fonctions logiques du Sinclair, une routine unique visant à remplacer les deux routines de grisage.

Les graphiques-barres à secteur de pourcentage grisé

Il est possible de se lancer dans la construction de graphiques plus complexes, contenant beaucoup d'informations et de lisibilité parfaite : ce sont les graphiques-barres verticaux à secteur grisé. Ces graphiques permettent par exemple de suivre l'évolution d'une distribution en pourcentages dans le temps.

Le programme (figure III-10) permet de construire en

```

100 REM GRAPHIQUE 2*D
110 REM -----
120 REM
1000 REM Introduction donnees
1100 PRINT "Combien d'etapes?"
1110 INPUT N
1120 PRINT "* par Combien d'obse
rvations?"
1130 INPUT P
1140 DIM T(N,P+1)
1200 FOR I=1 TO N
1205 PRINT INVERSE 1;"Etape no.:"
;I
1210 FOR J=1 TO P: INPUT T(I,J)
1220 PRINT T(I,J): NEXT J
1230 PRINT "Etape correcte?(O/N)"
; INPUT R#
1240 IF R#<>"O" THEN GO TO 1205
1250 NEXT I
1290 REM
1300 REM Sommes par etapes
1310 FOR I=1 TO N: LET S=0
1320 FOR J=1 TO P
1330 LET S=S+T(I,J): NEXT J
1340 LET T(I,P+1)=S: NEXT I
1350 REM
2000 REM Recherche de la +grande
somme
2050 LET S1=0
2100 FOR I=1 TO N
2110 IF T(I,P+1)>S1 THEN LET S1=
T(I,P+1)
2120 NEXT I: LET S=S1
2130 CLS
3000 REM Cadre et barres
3100 PLOT 0,0: DRAW 255,0: DRAW
0,175: DRAW -255,0: DRAW 0,-175
3190 REM
3200 REM Espacement des barres
3210 LET E=INT (255/(3*N+1))
3220 LET E1=2*E+INT ((255-E-N*3*
E)/N): LET S1=(175-E)/S
3300 FOR I=1 TO N
3305 DEF FN E(I)=E*I+E1*(I-1)
3310 PLOT FN E(I),0
3320 DRAW 0,T(I,P+1)*S1
3330 DRAW E1,0: DRAW 0,-T(I,P+1)
*S1: NEXT I
3900 REM
4000 REM Sections
4050 DIM Y(N,P)
4100 FOR B=1 TO N
4110 FOR I=1 TO P-1: LET Y=0
4120 FOR J=1 TO I: LET Y=Y+T(B,J)
; NEXT J
4130 LET Y(B,I+1)=Y*S1
4140 PLOT FN E(B),Y(B,I+1): DRAW
E1,0
4150 LET D=FN E(B): LET F=D+E1
4170 LET H=Y(B,I+1)-Y(B,I)
4180 GO SUB 5000+(I/2-INT (I/2))
*2000
4190 NEXT I: NEXT B
4900 STOP
4950 REM
5000 REM Grisage-0
5010 REM
5140 FOR K=D TO F STEP I
5150 LET O=F-K: LET H1=H*(O>=H)+

```

```

Q*(Q<H)
5180 PLOT K,Y(B,I): DRAW H1,H1
5200 NEXT K
5300 FOR Y=Y(B,I) TO Y(B,I+1) ST
EP I
5320 LET H=Y(B,I+1)-Y: LET H1=H*
(H<E1)+E1*(H>=E1)
5340 PLOT D,Y: DRAW H1,H1
5360 NEXT Y: RETURN
5900 REM
6000 REM Grillage-1
6140 FOR K=F TO D STEP -1
6160 LET Q=K-D: LET H1=H*(Q>=H)+
Q*(Q<H)
6180 PLOT K,Y(B,I): DRAW -H1,H1
6200 NEXT K
6300 FOR Y=Y(B,I) TO Y(B,I+1) ST
EP I
6320 LET H=Y(B,I+1)-Y: LET H1=H*
(H<E1)+E1*(H>=E1)
6340 PLOT F,Y: DRAW -H1,H1
6360 NEXT Y: RETURN

```

Figure III-10

quelques secondes une présentation graphique dont la confection demanderait un temps considérable en exécution manuelle, comme par exemple la figure III-11 à partir de données aléatoires. A cause de son caractère très explicite, ce type de graphique est d'une application très générale dans la littérature scientifique ou dans la presse technique, quel qu'en soit le domaine : visualiser d'un coup d'oeil des chiffres qui paraîtraient beaucoup moins significatifs dans les tableaux statistiques ordinaires. On a ainsi traité (figure III-12) le tableau publié par *Le Monde Informatique* du 13 septembre 1982 concernant l'évolution du parc français des ordinateurs (figure III-13).

Le premier graphique présente la série sur la base des unités d'ordinateurs, le deuxième graphique sur la base de la valeur en MF. La comparaison des deux graphiques permet ainsi de mettre en évidence l'opposition nombre-prix qui sépare les petits des gros matériels. Les tableaux sont particulièrement parlants.

La structure du programme est donc plus complexe puisqu'il récupère, en qualité de sous-programme, l'essentiel des programmes précédemment développés. Ce programme commence par ouvrir un tableau à deux dimensions, d'où le nom de graphique 2*D. On notera que la dimension verticale est augmentée d'une unité (ligne 1440), ceci pour ménager une place aux sommes partielles de chaque étape. En effet, pour arrêter la hauteur maximum du graphique, le programme est obligé de rechercher, non plus la plus grande donnée, mais la plus grande

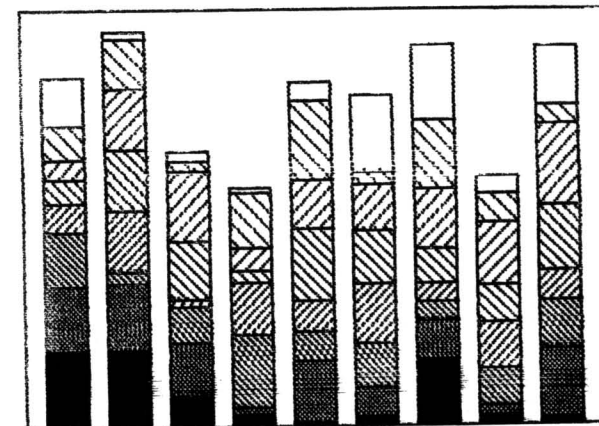


Figure III-11

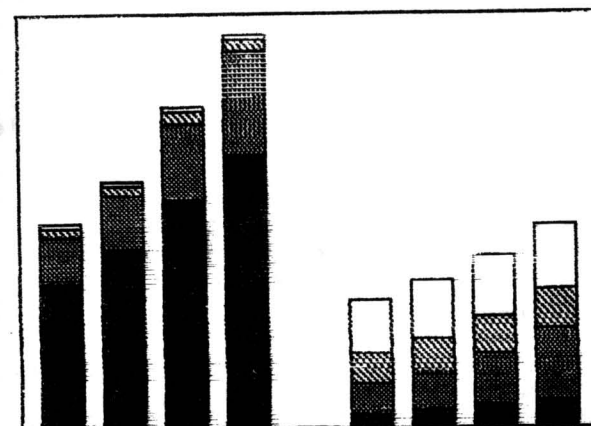


Figure III-12

Evolution du parc français des ordinateurs

Années au 1 ^{er} janv.	En nombre					Parc total
	Très petits de 50 à 250 kF	Petits de 0,25 à 1,6 MF	Moyens de 1,6 à 7 MF	Grands et très grands + de 7 MF		
1979	43 419	12 912	2 748	1 088		60 167
1980	52 848	15 529	3 103	1 221		72 701
1981	67 376	22 287	3 141	1 271		94 075
1982	80 404	29 807	3 571	1 346		115 128
Progres- sion 81/82	+ 19,3 %	+ 33,7 %	+ 13,6 %	+ 5,9 %		+ 22,3 %

Figure III-13

Années au 1 ^{er} janv.	En valeur (MF)					Parc total
	Très petits de 50 à 250 kF	Petits de 0,25 à 1,6 MF	Moyens de 1,6 à 7 MF	Grands et très grands + de 7 MF		
1979	4 340	9 040	8 795	15 230		37 405
1980	5 285	10 870	9 930	17 095		43 180
1981	6 740	15 600	10 050	17 790		50 180
1982	8 040	20 870	11 430	18 845		59 185
Progres- sion 81/82	+ 19,2 %	+ 33,7 %	+ 13,7 %	+ 5,9 %		+ 17,9 %
	+ 1 300	+ 5 270	+ 1 380			

somme de données partielles. Cette recherche est effectuée par les lignes 1300 à 2130. Le cadre et les barres sont ensuite dessinés et les barres espacées conformément à la technique développée précédemment. Une de ces barres est ensuite considérée comme une bande verticale traitée individuellement selon la technique illustrée plus haut.

Toutefois les deux routines de grisage sont un peu plus complexes, car le point de départ et le point d'arrivée de chaque tracé oblique n'est pas dans ce cas une constante, mais une variable, D et F (Début-Fin) : ligne 4150. Dans ce cas également les deux routines de grisage peuvent être fondues en une seule selon la technique suggérée à l'exercice 3.2.

Remarque : il est possible de juxtaposer deux graphiques dans le même cadre avec le subterfuge suivant, à condition que dans le premier et le second graphiques le nombre d'observations par étape soit identique : il suffit d'ajouter entre les deux séries une barre supplémentaire dont toutes les valeurs sont égales à 0, ce qui a été fait sur la figure III-12.

Cette formule graphique est extrêmement souple et peut se plier à des applications très différenciées.

Les graphiques circulaires

Le type des graphiques circulaires, appelés familièrement "graphiques camembert" est souvent utilisé car il présente une excellente lisibilité. On peut même dire qu'en matière de pourcentages ils sont plus explicites que les graphiques-bandes étudiés précédemment car le cercle suggère mieux l'idée de l'ensemble divisé en un certain nombre de parts. Ils sont enfin, et cela n'est pas négligeable, sans doute plus esthétiques.

Leur construction est en revanche nettement plus délicate que celle des précédents, en ce sens qu'il est nécessaire de faire appel aux fonctions circulaires (trigonométriques), et de bien connaître la réalité du radian en tant que mesure. Il y a 2π radians dans un cercle, ce qui fait environ 57 degrés pour un radian (57,29). Ces graphiques impliquent qu'il soit fait usage de la fonction **CIRCLE** du Sinclair pour d'abord dessiner le cercle.

Les graphiques circulaires grisés par points

Il y a une variété de graphique circulaire très facile à construire, mais peu spectaculaire et surtout très lente, qui constitue une bonne entrée en matière pour cette section, car elle illustre la notion de **coordonnées polaires**. En effet il va être fait appel à une génération aléatoire pour déterminer l'apparition de points dans les différents secteurs qui représentent les pourcentages d'une série de données par rapport à leur ensemble. L'illustration de ce type de graphiques est fournie figure III-14.

Le programme qui réalise ce type de graphiques est représenté figure III-15. Il se compose de deux parties : une première qui va de la ligne 10 à la ligne 330 intègre les données dans un tableau ; la deuxième partie (1000 à 1700) est proprement graphique.

La première calcule naturellement les pourcentages mais surtout les transforme en angles. L'origine des angles dans un cercle est l'axe horizontal, mais il est plus conforme à l'habitude visuelle de faire le décompte de secteurs à partir du haut du cercle ; aussi l'on procède comme s'il existait une donnée 0 de la valeur d'un quart de cercle, objet de l'instruction 200. Puis les différents secteurs sont tracés avec des rayons distincts à partir de l'origine, comme si chaque angle était la somme de lui-même et de tous les angles qui le précèdent. Ces rayons seront tracés à l'aide de l'instruction **DRAW** dont les arguments sont : différence des points d'abscisse, différence des points d'ordonnée.

Chaque rayon étant dessiné à partir du centre, son extrémité sera facilement déterminée par les règles de la trigonométrie, la différence des abscisses correspondant au cosinus de l'angle. L'instruction 1300 est explicite à ce sujet puisque l'on y voit la valeur du rayon en longueur (87) successivement multipliée par le cosinus de l'angle puis par le sinus de l'angle (figure III-16).

On procède ensuite au grisage par points du graphique selon la procédure suivante : un certain nombre de points vont être tracés à une distance du centre aléatoirement choisie entre 0 et 87 et selon un angle aléatoirement pris entre 90° et 360° moins la valeur du dernier angle. Les deux valeurs qui vont déterminer l'emplacement de chaque point ne sont autres que les coordonnées polaires de ce point. L'éloignement du

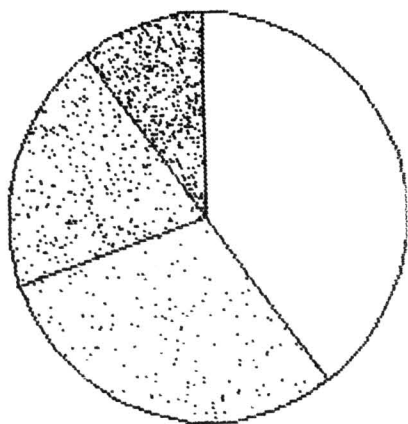


Figure III-14

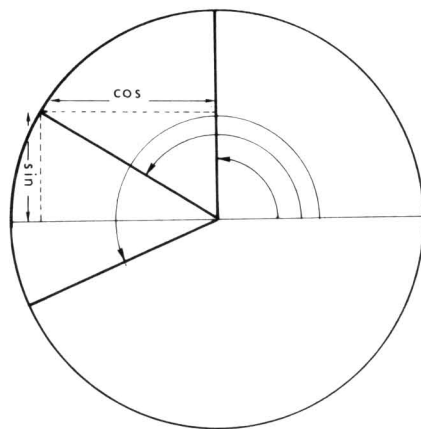


Figure III-16

```

10 REM CAMEMBERT
20 REM Grise a points
30 REM -----
80 PRINT "Combien de valeurs?"
100 INPUT A: DIM T(A,2)
105 LET S=0
110 FOR I=1 TO A
120 PRINT "Valeur no: "; I: INPUT
T(I,1)
125 LET S=S+T(I,1)
130 NEXT I
200 LET T(1,2)=PI/2
210 FOR I=2 TO A
220 LET T(I,2)=T(I-1,2)+T(I-1,1)
) /S*2*PI
230 NEXT I
300 CLS
310 FOR I=1 TO A
325 PRINT INT (T(I,1)/S*10000) /
100; " %"
330 NEXT I
1000 CIRCLE 168,88,87

1200 FOR I=1 TO A
1300 PLOT 168,88: DRAW 87*COS T(
I,2),87*SIN T(I,2)
1500 NEXT I
1550 FOR I=2 TO A
1600 LET R=RD*(T(I,2)-PI/2)+PI/
2
1630 LET H=SQR (2*RD*3784.5)
1650 LET X=COS R: LET Y=SIN R
1660 PLOT H*X+168,H*Y+88
1670 NEXT I
1700 GO TO 1550

```

Figure III-15

centre au module est défini par la distance H (comme Hasard) avec l'instruction 1630, et l'angle que forme avec le rayon d'origine le segment qui joint ce point au centre est aléatoirement défini par l'instruction 1600.

Pour obtenir un dégradé dans les grisés il est nécessaire que certaines régions soient plus fréquemment incluses que d'autres dans la partie du cercle à griser. C'est pour cette raison que cette procédure de grisage par points est incluse dans une boucle qui prend successivement en compte, non pas la surface de chacun des angles, mais la surface de chacun des angles augmentée de tous ceux qui le précèdent. Cette procédure de grisage n'a qu'une valeur illustrative et est beaucoup trop longue pour permettre de disposer d'un instrument de représentation efficace.

Une technique nettement plus rapide et plus belle est fournie par le programme "camembert" dont les secteurs sont grisés par lignes.

Les graphiques circulaires à grisage par lignes

Ce type de graphique illustré par la figure III-17 est d'une réalisation nettement plus délicate.

2.77 %
5.55 %
8.33 %
11.11 %
13.88 %
16.66 %
19.44 %
22.22 %

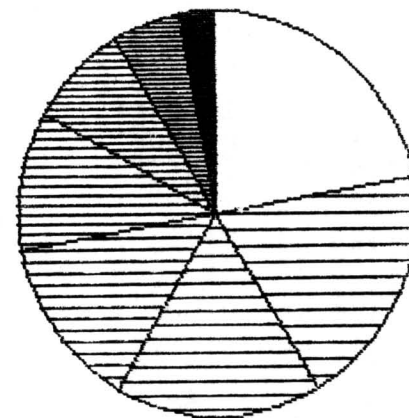


Figure III-17

Seule la première partie est comparable au programme précédent, c'est-à-dire jusqu'au module de comptage (instruction 500). Mais les choses se compliquent à

```

10 REM CAMEMBERT
20 REM Lignes
30 REM -----
80 PRINT "Combien de valeurs?"
100 INPUT A: DIM R(A,2)
105 LET S=0
110 FOR I=1 TO A
120 PRINT "Valeur no: "; I: INPUT
R(I,1)
125 LET S=S+R(I,1)
130 NEXT I
200 LET R(1,2)=PI/2
210 FOR I=2 TO A
220 LET R(I,2)=R(I-1,2)+R(I-1,1)
) / S * 2 * PI
230 NEXT I
300 CLS
310 FOR I=1 TO A
325 PRINT INT (R(I,1) / S * 10000) /
100; " %"
330 NEXT I
500 REM Module de comptage
510 DIM T(A+3,2): LET C=PI: LET
B=1
515 LET T(1,2)=PI/2
520 FOR K=2 TO A
525 LET B=B+1
530 IF R(K,2) < C THEN LET T(B,2)
=R(K,2): LET T(B-1,1)=K: NEXT K:
GO TO 800
540 IF R(K,2) = C THEN LET T(B-1,
1)=K: LET T(B,2)=R(K,2): LET C=C
+PI/2: NEXT K: GO TO 800
560 IF R(K,2) > C THEN LET T(B,2)
=C: LET C=C+PI/2: LET T(B-1,1)=K
: GO TO 525
1000 CIRCLE 168,88,87
1200 FOR I=1 TO A
1300 PLOT 168,88: DRAW 87*COS R(
I,2),87*SIN R(I,2)
1500 NEXT I
4000 REM Module de grisage
4050 FOR K=1 TO B-1
4100 REM Parametres
4105 LET Z=(T(K,2)+T(K+1,2))/2
4110 LET D=(COS Z*SIN Z<0)
4120 LET E=87*SIN T(K+D,2)
4130 LET F=87*SIN T(K+1-D,2)
4140 LET U=COS T(K+1-D,2)/SIN T(
K+1-D,2)
4160 LET P=(T(K,1)-1)*SGN SIN Z
4170 LET W=SGN COS Z
4180 IF E=0 THEN LET I=0: GO TO
4300
4190 LET V=COS T(K+D,2)/SIN T(K+
D,2)
4200 REM Secteurs
4210 FOR I=0 TO E STEP P
4220 LET L=I*V-I*U
4230 PLOT 168+I*U,I+88: DRAW L,0
4250 NEXT I
4300 FOR I=I TO F STEP P
4310 LET L=U*SQR (7559-I*I)-U*I
4320 PLOT 168+I*U,I+88: DRAW L,0
4350 NEXT I
5000 NEXT K

```

Figure III-18

partir de ce moment-là, car l'utilisation de la fonction **DRAW**, qui permet de tracer chacun des segments qui vont griser les secteurs, est spécialement délicate à mettre en oeuvre bien que l'un des paramètres, le paramètre de hauteur, soit constamment égal à 0 (figure III-18).

La procédure est la suivante : le tracé de chaque ligne commence au rayon de départ de l'angle concerné et son aboutissement est, soit le rayon suivant, soit le rebord du cercle. Il est donc nécessaire de calculer à chaque fois, angle par angle, à quel moment l'aboutissement de la ligne ne sera plus le rayon suivant mais le cercle.

Une autre difficulté résulte de ce que les secteurs ne sont pas forcément contenus dans le même quart de cercle, c'est-à-dire qu'ils peuvent être à cheval sur un rayon horizontal ; alors l'incrémentation qui produit la succession des lignes est plus difficile à organiser puisqu'il ne suffit plus de faire remonter le point de départ de chaque ligne sur un seul rayon. Il va donc être nécessaire, pour simplifier globalement le problème, de décomposer chaque secteur qui se trouvera ainsi à cheval sur deux ou plusieurs quarts de cercle en autant de sous-secteurs qu'il y a de quarts de cercle concernés, de façon à pouvoir traiter le grisage des régions ainsi définies sans se heurter au problème résultant de l'inversion du signe du sinus (à cheval sur un rayon horizontal) ou du cosinus (à cheval sur un rayon vertical). Naturellement lorsqu'un secteur sera assez vaste pour concerner plusieurs quarts de cercle, les sous-secteurs qui le composent devront être grisés d'une manière analogue et sans discontinuité (figure III-19). D'où la nécessité de construire un second tableau de données comportant un nombre plus élevé de secteurs et calculant les densités de grisage de telle sorte que les sous-secteurs éventuels d'un même secteur soient de la même densité. C'est le module de comptage, qui va des lignes 500 à 560, qui effectue ce traitement.

Le module de grisage qui suit et qui occupe les lignes 4000 se décompose en deux parties : une première partie calcule les différents paramètres correspondant aux caractéristiques du grisage pour cinq secteurs ; et la deuxième partie (4200-4360) comprend la routine de grisage proprement dite, elle-même divisée en deux sous-routines, une routine calculant les lignes de rayon à rayon (4210 à 4250) et une sous-routine calculant les lignes de rayon à bord du cercle (4300-4350).

Les variables qui définissent les paramètres sont les suivantes : F est la fin du grisage pour chaque secteur, E détermine l'endroit où commence sur chaque rayon le second mode de calcul de la longueur du tracé, P calcule l'incrémentation qui donnera la densité du grisage Z, D, U et W sont des variables de travail qui visent à déterminer quel est

le signe des valeurs trigonométriques utilisées pour savoir essentiellement dans quel sens on tracera la ligne : par exemple vers la gauche (moitié du cercle où le cosinus est négatif) ou vers la droite (moitié du cercle où le cosinus est positif), la succession des lignes se faisant vers le haut (moitié supérieure du cercle) ou vers le bas (moitié inférieure du cercle : sinus négatif).

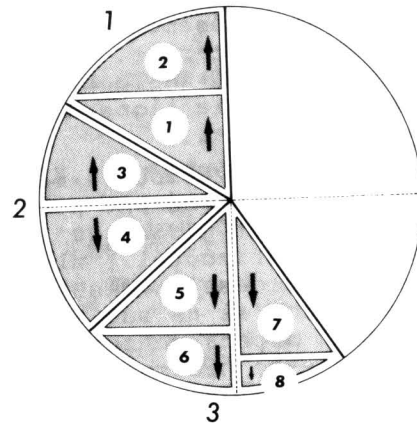


Figure III-19

Il est parfaitement possible d'augmenter le contraste entre les différents secteurs en jouant sur la variable P, sans se contenter de l'incrémenter purement et simplement. En revanche on ne pourra pas toucher à son signe : il devra toujours être positif pour la moitié de cercle supérieure et négatif pour la moitié de cercle inférieure.

EXERCICE 3.3

Trouver une brève formule qui introduise de la variété dans la définition du pas d'incrémentation afin de produire des grisés contrastés en alternance.

Les graphiques en ligne brisée

Les graphiques en ligne brisée, dont un exemple est figuré III-20, représentent la catégorie la plus ordinaire de graphiques. Ces graphiques sont nettement moins esthétiques que les précédents et l'économie de temps réalisée par l'ordinateur dans ce cas n'est pas

aussi considérable car il est aisé de prendre un papier millimétré pour reporter les différentes valeurs d'une série statistique par exemple.

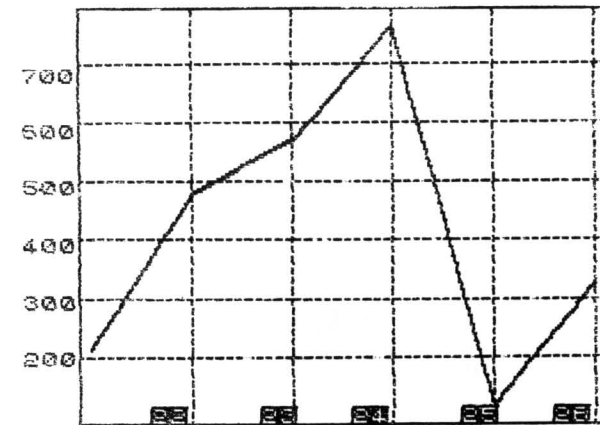


Figure III-20

Mais à côté de leur simplicité, ces graphiques présentent une supériorité sur les précédents : ils sont les seuls à accepter la superposition de plusieurs séries de données sans engendrer de confusion visuelle.

Le programme qui réalise ce type de graphique est présenté figure III-21. On remarquera que ce programme est assez long, ce qui peut paraître surprenant étant donné la relative simplicité des graphiques proposés. La longueur du programme tient essentiellement au fait qu'on lui confie entièrement le choix de l'échelle de représentation, tant dans la dimension verticale que dans la dimension horizontale. Le programme calcule automatiquement les données extrêmes sur chacun des deux axes, c'est-à-dire qu'il va choisir l'unité graphique sur chacun de ces deux axes (lignes 500 et suivantes). Et, deuxièmement, la longueur du logiciel est due à ce qu'il effectue également le dessin des graduations qui donneront plus de lisibilité au graphique.

Le choix de ces graduations a impliqué une routine relativement délicate : il faut en effet choisir pour elles l'ordre de grandeur le plus adapté aux dimensions de l'écran, ne pas surcharger le graphique avec des

```

100 REM GRAPHIQUE SIMPLE
110 REM -----
120 REM
200 REM Introduction donnees
210 PRINT "Combien d'etapes ?"
220 INPUT N: DIM T(N,2)
300 FOR I=1 TO N
310 PRINT INVERSE 1; "Donnee no: "; I
320 PRINT "X="; INPUT T(I,1)
330 PRINT T(I,1);
340 PRINT "Y="; INPUT T(I,2)
350 PRINT T(I,2)
360 NEXT I
390 REM
400 REM Calcul des extremes
410 LET G=T(1,1): LET D=T(1,1)
420 LET H=T(1,2): LET B=T(1,2)
430 FOR I=1 TO N
440 LET T2=T(I,2)
445 LET T1=T(I,1)
450 IF T2<B THEN LET B=T2
460 IF T2>H THEN LET H=T2
470 IF T1>D THEN LET D=T1
480 IF T1<G THEN LET G=T1
485 NEXT I
490 CLS
495 REM
500 REM Parametres
505 LET CY=160/(H-B): LET CX=22
510 LET D-G
530 LET B$=STR$ INT B
540 LET H$=STR$ INT (H-B)
550 LET G$=STR$ INT G
560 LET D$=STR$ INT (D-G)
570 LET LY=LEN H$: LET LX=LEN D$
580 LET SX=10+(LX-1): LET SY=10+(LY-1)
590 REM
600 REM Cadre
610 PLOT 25,0: DRAW 0,175: DRAW 230,0: DRAW 0,-175: DRAW -230,0
615 REM
620 REM Graduation horizontale
630 PLOT 25,0: DRAW 0,175: DRAW 230,0: DRAW 0,-175: DRAW -230,0
640 LET A=(INT (B/SY)+1)*SY: LET C=A+(VAL H$(1))*SY: IF C>=H THEN LET C=C-SY
650 FOR I=A TO C STEP SY
660 LET Y=(I-B)*CY+8
670 LET W=21-Y/8.33: PRINT AT W,0; I
680 FOR J=25 TO 255 STEP 1.3
690 PLOT J,Y: NEXT J: NEXT I
695 REM
700 REM Graduation verticale
710 LET A=(INT (G/SX)+1)*SX: LET C=A+(VAL D$(1))*SX: IF C>=D THEN LET C=C-SX
710 FOR I=A TO C STEP SX
720 LET X=(I-G)*CX+30
730 LET W=X/8-2: PRINT INVERSE 1,AT 21,W; I
740 FOR J=0 TO 175 STEP 1.3
750 PLOT X,J: NEXT J: NEXT I
790 REM
800 REM Ligne du graphique
810 DEF FN Y(I)=(T(I,2)-B)*CY+8

```

```

820 DEF FN X(I)=(T(I,1)-G)*CX+30
830 FOR I=1 TO N-1
840 PLOT FN X(I),FN Y(I)
850 DRAW FN X(I+1)-FN X(I),FN Y(I+1)-FN Y(I)
860 PLOT FN X(I),FN Y(I)-1
870 DRAW FN X(I+1)-FN X(I),FN Y(I+1)-FN Y(I)
880 NEXT I

```

Figure III-21

graduations trop nombreuses sous peine de le rendre illisible, ne pas non plus les négliger, faute de quoi le graphique cesse d'être parlant. Pour ce faire, le nombre des graduations ne doit pas être supérieur à la dizaine ou inférieur à l'unité. La base de la décision est évidemment la différence entre la donnée la plus faible et la plus élevée, en calculant la limite supérieure par excès et la limite inférieure par défaut. Le calcul de la graduation verticale (le long de l'axe horizontal) est plus simple en vue de donner une certaine liberté à l'utilisateur, pour comprimer ou élargir son graphique dans la largeur.

Revenons rapidement sur chacune de ces routines :

- 400, le calcul des extrêmes : même principe que précédemment, mais avec 4 variables, B (bas), D (droit), H (haut), G (gauche).
- 500 et suivantes, le calcul des paramètres : pour définir l'ordre de grandeur des valeurs extrêmes, il est fait usage de la fonction **STR\$** affectant la valeur entière (Integer) des valeurs extrêmes. La longueur de la chaîne ainsi obtenue est calculée par la fonction **LEN** à la ligne 570. C'est cette longueur qui fournira la puissance de 10 qui servira d'ordre de grandeur au calcul des échelles (ligne 570 : élévation à la puissance dans la 3e instruction de la ligne 570).

Les variables pivot pour le dessin des différentes graduations sont les suivantes : **SX**, pas de la graduation verticale, **SY**, pas de la graduation horizontale, **CX**, coefficient d'unité graphique sur l'axe des X, **CY**, coefficient d'unité graphique sur l'axe des Y. **CX** et **CY** ont pour objet de faire respectivement rentrer toutes les valeurs initiales dans les possibilités maximales de 255 points sur l'axe des X et de 175 points sur l'axe des Y.

- ligne 800 et suivantes, réalisation de la ligne

graphique proprement dite, à l'aide de deux fonctions prédéfinies qui travaillent sur les valeurs directement entrées au clavier en début de travail par l'utilisateur pour leur donner leur valeur graphique en fonction des unités d'échelle et par rapport à une origine différente (B = bas, G = gauche). La ligne brisée est doublement plottée pour lui donner plus d'épaisseur par rapport aux graduations. Une seconde ligne est pour cela tracée exactement au-dessous de la première (ligne 860).

EXERCICE 3.4

Ajouter au programme ci-dessus, à la fin, une brève routine qui permette d'effectuer la surimposition de plusieurs graphiques de façon à arriver au résultat représenté à la figure III-22.

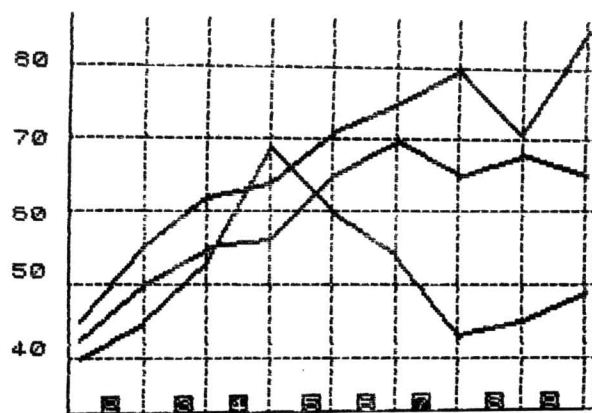


Figure III-22

LES REGRESSIONS SIMPLES

Les problèmes de régression sont une très intéressante application des possibilités du ZX-Spectrum, tant dans la puissance et la rapidité de calcul que dans la finesse de définition graphique. Il s'agit là d'une technique très utile qui se situe au point de rencontre

du calcul des courbes tel qu'il a été exposé en début de chapitre et des représentations graphiques qui ont fait l'objet des développements qui précèdent.

Le principe de la régression

Le problème posé est le suivant : lorsque l'on se trouve en présence d'une série importante de données, statistiques principalement, la projection graphique des différents points qui représentent chacun une observa-

tion permet souvent de mettre en évidence une tendance générale qui suggère que le phénomène observé peut être une courbe explicative correspondant à l'un des grands types de courbes étudiés précédemment. Une courbe peut être définie par le calcul de telle sorte que, passant entre les différents points qui constituent la série complète, la somme des distances de ces points à la courbe théorique ainsi définie soit la plus petite possible. Pour obtenir la courbe dite de **régression**, il va être fait usage d'une

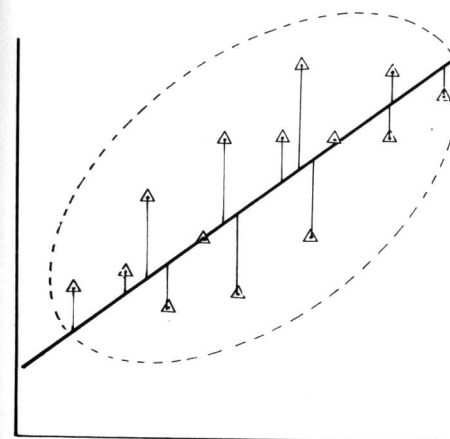


Figure III-23

méthode très simple appelée "**méthode des moindres carrés**".

L'opération menée est représentée dans la figure III-23. Un certain nombre de points ont été projetés graphiquement sur un couple d'axes orthonormés (perpendiculaires). Le calcul aboutira à la définition d'une courbe exponentielle négative dont le pointillé apparaît, épousant de très près la forme générale du nuage. Il s'agit dans ce cas particulier d'une exponentielle négative dont les coefficients sont présentés en vidéo inversée. Cette courbe, qui est fictive, pourrait par exemple représenter la chute des temps d'accès aux mémoires mortes sur les 20 dernières années. Les temps d'accès sont présentés sur l'axe vertical et les différentes années sur l'axe horizontal à raison d'une graduation pour 2 années. Plus on va et plus les temps d'accès se raccourcissent en proportion avec une réduction qui est en gros de 10% par an. La valeur absolue

du coefficient de corrélation représenté en haut du graphique (.97) **est très proche de l'unité**, ce qui prouve que la courbe théorique d'ajustement rend parfaitement compte du phénomène.

La méthode des moindres carrés

La méthode de la régression linéaire utilisant la méthode des moindres carrés est illustrée dans la figure III-24. On va calculer la moyenne des points sur chacun des deux axes pour obtenir une sorte de point pivot et par ce point on va faire passer une droite.

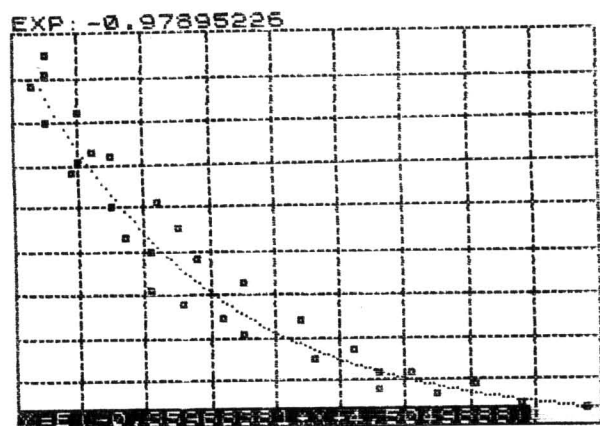


Figure III-24

L'angle que formera cette droite avec l'axe des X va être calculé de façon à minimiser la somme des carrés des distances figurées en traits fins. Il s'agit dans ce cas d'une régression linéaire, c'est-à-dire qu'un ensemble de points est ramené à une ligne droite.

Comment est-il donc possible de ramener à d'autres types de courbes ? Il faut pour cela jouer sur les valeurs que l'on introduit en effectuant l'ensemble des calculs, non seulement pour la valeur telle qu'elle est, **mais aussi pour son logarithme**. Pour comprendre ce mécanisme, imaginons une disposition de points qui rappelle manifestement une courbe exponentielle. Conformément à ce qui a été dit plus haut sur les échelles logarithmiques, si ces mêmes points avaient été entrés en prenant

en compte chaque fois, non la valeur Y de départ, mais le logarithme de cette valeur, la totalité des points se disposerait alors selon une ligne droite.

Inversement, si l'on avait affaire à un ensemble de points qui se disposent selon une courbe de type logarithmique, si ces différents points avaient été introduits en prenant chaque fois pour abscisse, non la valeur X, mais le logarithme de cette valeur, ces points se retrouveraient disposés selon une ligne.

Le programme de régression (figure III-25)

Techniquement, il va donc se produire quatre calculs en parallèle, utilisant respectivement les valeurs de X et Y simples (régression linéaire), les valeurs de X simples et les logarithmes de Y (régression exponentielle), les valeurs de Y et les logarithmes des valeurs X (régression logarithmique), les logarithmes des Y et les logarithmes des X (régression puissance, selon une courbe de puissance).

Lorsque l'on commandera les calculs de régression en introduisant une valeur 0 à la place d'un X (ce qui signifiera la fin de l'introduction des couples X, Y), le programme va calculer quatre ajustements par les moindres carrés et **se déterminera en faveur de l'ajustement qui présentera la corrélation la plus élevée**. Ceci dit, il existe toujours dans ce genre de problème des valeurs aberrantes qui risquent de fausser les résultats. On s'en rend très vite compte car la courbe de régression qui est alors dessinée par le programme ne passe manifestement pas à travers le nuage des points. Il suffit alors de modifier l'échelle des X pour obtenir une régression pertinente.

Les différentes parties du programme sont les suivantes :

- jusqu'à la ligne 1000, initialisation et calcul du cadrage en fonction des valeurs extrêmes selon la technique déjà rencontrée précédemment pour les graphiques simples ;

- ensuite la partie qui calcule les paramètres, qui va de la ligne 1000 à 3900 : de la ligne 1000 à 2800 sont définis pour chacune des quatre catégories les sommes et produits partiels portant sur les différentes valeurs successivement entrées et leurs logarithmes. Une fois que tous les couples sont entrés, les calculs de valeurs moyennes sont effectués par les lignes 3000 à 3900.

```

10 REM COURBES DE REGRESSION
20 REM -----
100 REM Initialisations
110 LET N=1: LET Z=0
120 LET SX=Z: LET SLX=Z
130 LET SY=Z: LET SLY=Z
140 LET XX=0: LET LXLX=0
150 LET XLY=0: LET LXY=0
160 LET YY=0: LET LYLY=0
170 LET XY=0: LET LXLY=0
500 PRINT "QUELLE EST LA PLUS G
RANDE VALEUR (ARRONDIS PAR EXCES
) SUR L'AXE DES 'X'?"
510 INPUT D
520 PRINT "SUR L'AXE DES 'Y'?"
530 INPUT E: CLS
600 PLOT 0,0: DRAW 0,167: PLOT
0,0: DRAW 255,0
700 LET D$=STR$ D: LET E$=STR$
E: LET K=10+(LEN D$-1): LET M=10
+(LEN E$-1)
705 LET D=254/D: LET E=167/E
710 FOR I=1 TO VAL D$(1)
720 FOR J=0 TO 167 STEP 1.3
730 PLOT I*K*D,J
740 NEXT J: NEXT I
750 FOR J=1 TO VAL E$(1)
760 FOR I=0 TO 252 STEP 1.4
770 PLOT I,J*M*E: NEXT I
780 NEXT J
790 REM
1000 REM Introduction des couple
s X,Y
1100 PRINT AT 0,0;"Couple no: ";N
1200 INPUT X: IF X=0 THEN GO TO
3000
1250 LET C=X*D
1300 INPUT Y: LET L=Y*E
1400 PLOT C-1,L+1: PLOT C-1,L: P
LOT C-1,L-1: PLOT C,L+1: PLOT C,
L-1: PLOT C+1,L+1: PLOT C+1,L: P
LOT C+1,L-1
2100 LET LX=LN X: LET LY=LN Y
2200 LET SX=SX+X: LET SLX=SLX+LX
2300 LET SY=SY+Y: LET SLY=SLY+LY
2400 LET XX=XX+X*X: LET LXLX=LXL
X+LX*LX
2500 LET XY=XY+X*Y: LET XLY=XLY+
X*LY: LET LXY=LXY+LX*Y: LET LXLY
=LXLY+LX*LY
2700 LET YY=YY+Y*Y: LET LYLY=LYL
Y+LY*LY
2800 LET N=N+1: GO TO 1100
3000 REM
3100 LET N=N-1
3200 LET SX=SX/N: LET SLX=SLX/N
3300 LET SY=SY/N: LET SLY=SLY/N
3320 LET XX=XX-N*SX*SX
3330 LET LXLX=LXLX-N*SLX*SLX
3400 LET XY=XY-N*SX*SY
3420 LET XLY=XLY-N*SX*SLY
3430 LET LXY=LXY-N*SLX*SLY
3440 LET LXLY=LXLY-N*SLX*SLY
3500 LET YY=YY-N*SY*SY
3520 LET LYLY=LYLY-N*SLY*SLY
3600 REM
4000 REM Correlation
4100 DIM C(4): DIM C$(4,3)
4200 LET C(1)=XY/SQR (XX*YY)
4250 LET C$(1)="LIN"
4300 LET C(2)=XLY/SQR (XX*LYLY)

```

```

4350 LET C$(2)="EXP"
4400 LET C(3)=LXY/SQR (LXLX*YY)
4450 LET C$(3)="LOG"
4500 LET C(4)=LXLY/SQR (LXLX*LYL
Y)
4550 LET C$(4)="PUI"
5000 LET H1=C(1): LET H2=1
5100 FOR G=2 TO 4: IF ABS C(G)>A
BS H1 THEN LET H1=C(G): LET H2=G
5200 NEXT G
5250 PRINT AT 0,0;C$(H2);";";C(H
2)
5300 GO SUB (6000+200*H2)
5900 REM
6000 REM Graphe de la regression
6100 REM
6199 REM LIN
6200 LET A=XY/XX: LET B=SY-A*SX
6250 LET P=A*E/D
6260 PRINT INVERSE 1;AT 21,0;"Y=
";A;" * X + ";B
6300 FOR I=0 TO 255 STEP 2: PLOT
I,P*I+B*E: NEXT I: STOP
6399 REM EXP
6400 LET A=XLY/XX: LET B=SLY-A*S
X
6420 PRINT INVERSE 1;AT 21,0;"Y=
E(";A;" * X + ";B;")"
6450 DEF FN W(I)=EXP (A/D*I+B)*E
6500 FOR I=0 TO 255 STEP 2: PLOT
I, FN W(I): NEXT I: STOP
6599 REM LOG
6600 LET A=LXY/LXLX: LET B=SY-A*
SLX
6640 PRINT INVERSE 1;AT 21,0;"Y=
";A;" * LN(X + ";B;")"
6650 FOR I=1 TO 255 STEP 2: PLOT
I,A*E*LN (I/D+B): NEXT I: STOP
6799 REM PUI
6800 LET A=LXLY/LXLX: LET B=SLY-
A*SLX
6850 PRINT INVERSE 1;AT 21,0;"Y=
E(";A;" * LN(X + ";B;")"
6900 FOR I=1 TO 255 STEP 2: PLOT
I,E*EXP (A*LN (I/D+B)): NEXT I:
STOP

```

Figure III-25

- les lignes 4000 à 5900 calculent les quatre coefficients de corrélation correspondant aux quatre types d'ajustement fondamentaux, puis détectent quel est le coefficient de corrélation le plus élevé en valeur absolue pour renvoyer à l'une des quatre routines graphiques qui dessinera la courbe d'ajustement en pointillé :

- ce sont les lignes 6000 à 6900. Chacune de ces sous-routines va d'abord calculer quel est le coefficient A de la courbe linéaire, quel est le coefficient B de valeur initiale pour X = 0, et, munie de ces paramètres, redresser, si besoin est (aucun redressement n'intervient

pour le cas linéaire pur) ; la formule de chacune des courbes définies est simple et apparaît en vidéo inversée dans le bas du cadrage.

En revanche le PLOTage est relativement plus difficile à organiser, car il faut tenir compte des transformations d'échelles X et Y et passer de X,Y à I, fonction de I pour le tracé.

Un exemple de régression

Prenons un exemple pour illustrer ceci : c'est la croissance des euro-devises de 1958 à 1977 telle qu'elle résulte des rapports annuels de la Banque des règlements internationaux (source : Gérard Békerman, *Les Euro-dollars*, QUE SAIS-JE ? P.U.F.). Cette série est typique d'une croissance exponentielle et nous allons voir de quelle façon le programme la traite.

1958 (année 0)	1,5	1968 (année 10)	30
1959 (année 1)	2,5	1969 (année 11)	45
1960 (année 2)	3,5	1970 (année 12)	57
1961 (année 3)	4	1971 (année 13)	71
1962 (année 4)	5,3	1972 (année 14)	105
1963 (année 5)	7	1973 (année 15)	155
1964 (année 6)	9	1974 (année 16)	210
1965 (année 7)	11,5	1975 (année 17)	250
1966 (année 8)	14,5	1976 (année 18)	305
1967 (année 9)	21	1977 (année 19)	350

Les chiffres sont exprimés en milliards de dollars.

Les différents couples de points introduits se disposent selon une ligne très caractéristique d'un développement exponentiel : ceci est sur la *figure III-26*. Une fois tous les couples entrés, on commande l'exécution en faisant X = 0 (si dans la série il existe véritablement une valeur nulle, pour éviter l'exécution du programme, on choisira une valeur très petite, par exemple 0,0001). Aussitôt les calculs s'exécutent et la courbe choisie est une exponentielle avec un très fort coefficient de corrélation, 0,99 et même 0,998, et la courbe recherchée a la formule $Y = E(0.292 * X + 0.5)$. Le résultat définitif est à la *figure III-27*.

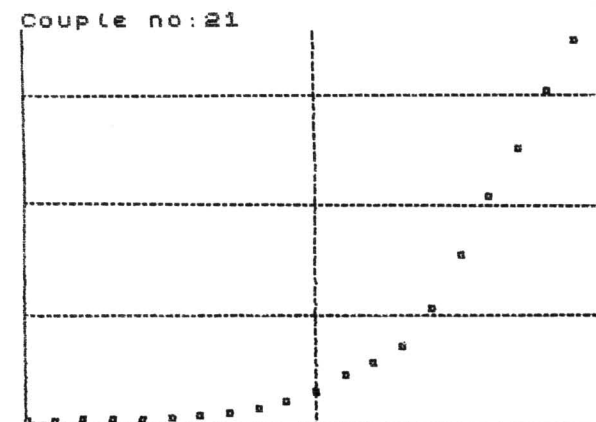


Figure III-26

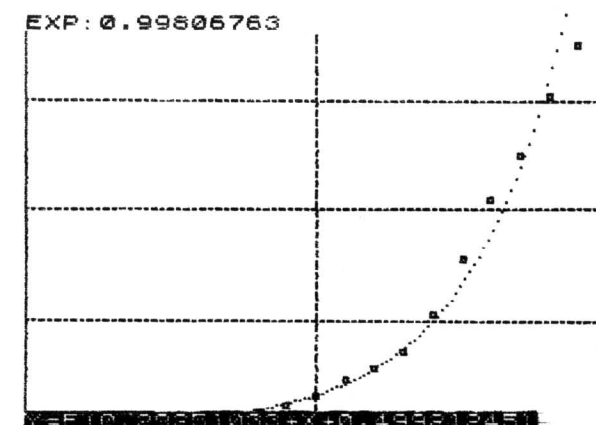


Figure III-27

LES REPRESENTATIONS DE COURBES EN TROIS DIMENSIONS

Ce chapitre consacré à l'exploitation des possibilités du ZX-Spectrum en matière de haute définition graphique serait incomplet sans une initiation à une application hautement spectaculaire : la représentation de

courbes en trois dimensions.

Les techniques de représentation de courbes en fonction de plusieurs variables en trois dimensions sont très complexes et ce n'est pas le lieu de les discuter ici. Nous allons seulement introduire le lecteur à une technique simple, **celle de la représentation en trois dimensions d'une fonction d'une seule variable**, c'est-à-dire de la forme engendrée dans l'espace par une courbe, telle qu'elle apparaîtrait normalement en deux dimensions sur le papier, lorsqu'on la fait pivoter autour de l'axe des Y de 360 degrés. C'est lorsque l'axe des Y est en même temps un axe de symétrie pour l'ensemble de la courbe que le résultat visuel est le plus spectaculaire.

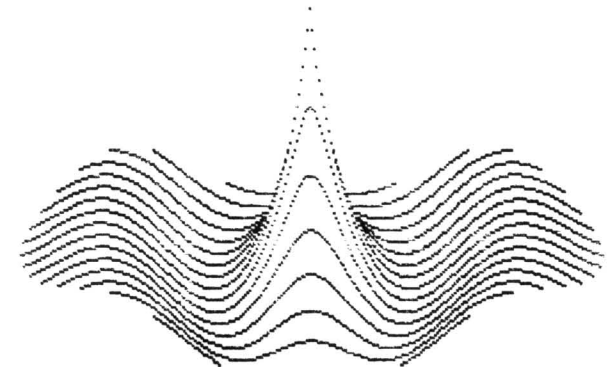
Deux de ces courbes représentant le traitement en trois dimensions de sinusoides simples sont présentées aux figures III-28 et III-29. En haut des figures sont représentées les formules qui les ont générées. Nous allons progressivement exposer au lecteur la méthode suivie en montrant les divers stades d'élaboration de la figure III-27.

Une méthode simple de représentation 3 D

Nous allons faire comme si la forme en trois dimensions était coupée perpendiculairement à son axe de symétrie vertical par un plan horizontal. L'intersection du plan et de cette forme est un cercle. Ce cercle, représenté en perspective, donne une ellipse. La figure III-30 illustre ce premier stade de la construction en montrant quel est le principe de dessin d'une forme elliptique. Examinons la ligne 1100. Si le coefficient .25 qui précède le **SQR** était supprimé, c'est en fait un véritable cercle qui serait inscrit sur l'écran, un cercle d'un rayon de 120 points. La formule du cercle étant $R^2 = X^2 + Y^2$, on définit bien pour chaque X la valeur de Y en soustrayant le carré de X du carré du rayon. Pour représenter, non pas un cercle qui ne tiendrait d'ailleurs pas dans l'écran, mais une ellipse, nous allons, chaque fois qu'une valeur de Y a été calculée, l'"écraser" en l'affectant d'un coefficient inférieur à 1 : .25, un quart dans cet exemple. Si le coefficient d'écrasement était moindre, l'effet de perspective de l'ellipse serait moindre également. S'il était supérieur, on s'apercevrait que dans la suite des calculs la courbe en trois dimensions ne serait plus aussi lisible : il faut donc trouver un optimum, qui se situe aux environs du quart au tiers.

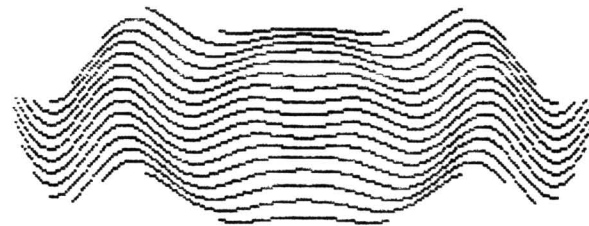
$$100 * \cos (X/15) / (ABS X/10 + 1)$$

Figure III-28



$$(X/7 - 1) * \sin (X/10)$$

Figure III-29



```
1000 FOR X=-120 TO 120
1100 LET Y=.25*SQR (120*120-X*X)
1200 PLOT X+120,Y+50
1300 PLOT X+120,50-Y
1400 NEXT X
```

Figure III-30



La phase suivante est illustrée par la figure III-31 accompagnée du programme qui la génère.

```

1000 FOR X=-120 TO 120
1100 LET Y=.25*SQR (120*120-X*X)
1200 IF X/2-INT (X/2)=0 THEN PLO
T X+120,Y+50: PLOT X+120,50-Y
1300 LET Y1=5*INT (Y/5)
1400 FOR Y=Y1 TO -Y1 STEP -5
1500 PLOT X+120,Y+50
1600 NEXT Y
1700 NEXT X

```



Figure III-31

Cette figure n'est autre que l'ellipse précédente, mais représentée en pointillés ; on voit surtout une série de lignes horizontales qui s'inscrivent dans cette ellipse et qui figurent les lignes d'intersection du plan de cette ellipse avec une succession de plans verticaux espacés d'une même distance. Chacune de ces différentes lignes représente l'axe des X d'une courbe qui va être calculée ultérieurement et qui représentera l'intersection de la forme recherchée avec chacun des plans verticaux figure III-32.

La figure III-33 illustre bien le stade ultérieur : c'est déjà la courbe représentée comme si l'ellipse était vue exactement par le travers, une fois que l'on a introduit la formule de la fonction qu'on souhaite représenter. Chacune des courbes qui se trouvent composer la figure globale n'est autre que l'une des lignes de la figures III-31 mais dont la valeur en Y, au lieu d'être égale à 0, correspond à la valeur de la fonction, à ce point précis d'éloignement du centre.

La figure III-34 qui représente le stade suivant, correspond à la même courbe, une fois réintroduite l'impression de perspective. Ce dessin est un dessin provisoire qui présente un certain nombre de défauts qu'il va falloir corriger. D'abord, lorsque le PLOTage rencontre une limite intérieure ou supérieure de l'écran, ceci

Figure III-32

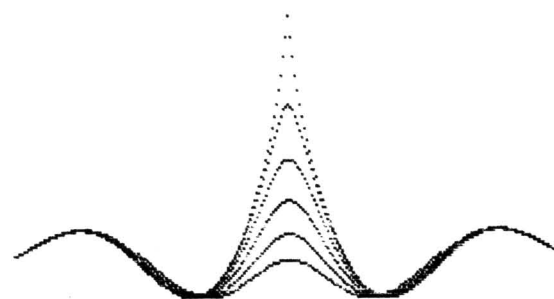
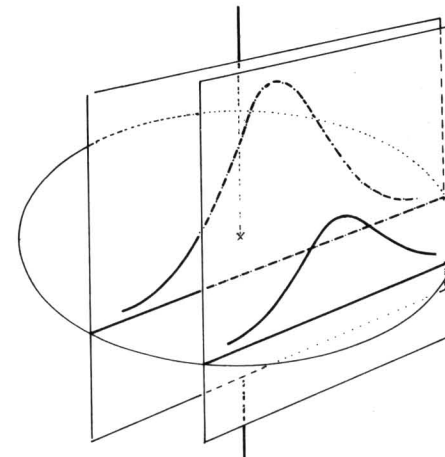
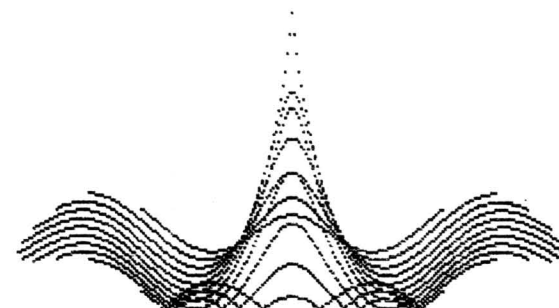


Figure III-33

$$100 * \cos (X/15) / (ABS X/10 + 1)$$

Figure III-34



provoque un rebroussement qui nuit à la clarté du dessin : il va falloir le supprimer. Deuxièmement, il se produit un effet de **transparence**, en ce sens que les courbes de la partie arrière sont reproduites intégralement. Pour obtenir un effet matériel de perspective, il va falloir supprimer la partie de ces courbes qui est sensée être dissimulée par la partie antérieure.

Deux routines très simples réaliseront ces correctifs. Nous allons voir comment elles sont agencées dans le programme 3D, très court, *figure III-35*.

```

1000 REM COURBES 3*D
1010 REM -----
1100 REM
1500 INPUT "Fonction?";F#
1600 DEF FN F(X)=VAL F#
1690 REM
1700 REM Parametres
1701 REM
1702 PRINT INVERSE 1;"Commencer
sur valeurs moyennes"
1800 PRINT "HAUTEUR ? -(0<H<.6)"
: INPUT H
1900 PRINT "LARGEUR ? -(0<L<1)":
: INPUT L
1910 PRINT "INCREMENT ? -(0<E<10)
: INPUT E
1920 PRINT "BASE ? -(25<B<75)": I
: INPUT B
1930 PRINT "PERSPECTIVE ? -(0<T<
1)": INPUT T
1990 CLS
2000 LET L=167*L
2010 FOR X=-L TO L
2050 LET P=0
2100 LET Y=H*SQR (L*L-X*X)
2300 LET Y1=E*INT (Y/E)
2400 FOR Y=Y1 TO -Y1 STEP -E
2450 LET M=SQR (X*X+Y*Y)
2500 LET Z=B-T*Y+FN F(M)
2550 IF Z<0 OR Z>175 THEN GO TO
2800
2650 IF Z<=P THEN GO TO 2800
2660 LET P=Z
2700 PLOT X+L,Z
2800 NEXT Y: NEXT X
3900 STOP
4000 REM Essais
4100 PLOT 0,25: DRAW 255,0
4150 PLOT 127,0: DRAW 0,100
4200 INPUT "Fonction ? ";F#
4250 DEF FN F(X)=VAL F#
4300 FOR X=-127 TO 127
4350 PLOT X+127,FN F(X)+25
4400 NEXT X

```

Figure III-35

Le programme 3 D

La première partie du programme, 1500-1600, correspond à l'introduction de la courbe recherchée qui est introduite sous la forme d'une chaîne de caractères dont la valeur numérique est récupérée en tant que fonction avec la très précieuse fonction **DEF FN** (définition de fonction).

Suit une partie importante, interactive, entre l'utilisateur et l'ordinateur, en vue de définir les paramètres d'impression. Cette partie n'est pas fonctionnellement importante mais donne de la vie et de la souplesse au programme. En effet l'utilisateur pourra, en modifiant tel ou tel paramètre, beaucoup mieux comprendre le fonctionnement du programme en appréciant la traduction graphique de cette modification.

Hauteur et largeur déterminent les deux dimensions de l'ellipse. La **base** définit l'emplacement de l'ellipse sur l'écran : une courbe faisant nettement saillie vers le haut aura besoin d'une base faible, une courbe faisant saillie vers le bas aura besoin d'une base plus haute. La **perspective** commandera la distance relative des différents plans de coupe verticaux. Si ces plans sont rapprochés la perspective est faible, si ces plans sont éloignés la perspective est forte et l'on aura un effet de plongé plus marqué (lignes 1700 à 1990).

Avant d'expliciter le coeur du programme, expliquons l'utilité de la routine d'essai commençant à la ligne 4000. Cette routine, qui n'est pas nécessaire au déroulement du programme, introduit juste une sorte de banc d'essai en deux dimensions qui permet de voir si une courbe a des chances de produire un graphe intéressant et de modifier les paramètres en conséquence.

Voici le détail de la routine principale, qui va de 2000 à 2800. Une première boucle, 2010 à 2800, commande le balayage de l'écran de gauche à droite, c'est-à-dire que le graphique 3D va se construire en dessinant les courbes représentatives, non pas une par une entièrement, mais toutes point par point. La base de chaque courbe est calculée selon la même procédure que le programme précédent; ce qui compte est l'apparition à la ligne 2450 d'une nouvelle valeur M qui est la valeur de l'éloignement du centre de l'ellipse. Il suffit alors de prendre le résultat de la fonction introduite pour cette valeur M. La ligne 2550 commande le passage à la valeur X suivante si le dessin qui en résulte est en dehors des limites haute et basse de l'écran. Les lignes

2650 et 2660 ont pour objet d'empêcher l'impression d'un point si son ordonnée est inférieure à celle du point précédent. Cela signifie qu'il est en principe caché par la forme 3D : c'est la ligne 2660 qui réajuste chaque fois cette hauteur.

Sur le schéma III-36 sont figurées les grandeurs représentant chacune des variables.

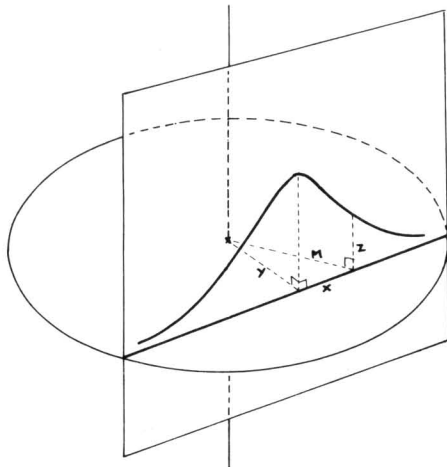


Figure III-36

EXERCICE 3.5

Apporter au programme figure III-30 une légère modification avec la suppression d'une ligne pour aboutir au dessin suivant que l'on peut appeler "le trou" (un conseil : c'est la variable Y1 qu'il faut supprimer), figure III-37.

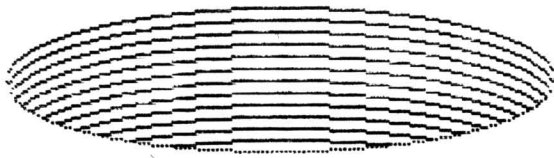


Figure III-37

CHAPITRE IV LA COULEUR ET LE SON

La possibilité de gérer un écran couleur et de produire des sons sur toute la gamme audible sont deux spécifications majeures du ZX-Spectrum. Les impératifs de la politique commerciale font que la couleur et le son sont même souvent présentés comme les qualités principales de cet ordinateur. Le programmeur ne raisonne pas tout-à-fait ainsi : pour lui, la haute définition graphique et la rapidité des calculs, ainsi que la richesse du jeu d'instructions Basic constituent l'essentiel d'un ordinateur, et il considère que la couleur et le son apportent seulement des "plus" agréables à l'oeil et à l'oreille.

Cependant, ces caractéristiques, accessoires dans une grande majorité d'applications et même parfois négatives, puisqu'elles contribuent à mobiliser une place mémoire importante et à faire baisser la définition de l'écran, deviennent évidemment primordiales dans le genre particulier des logiciels de divertissement : jeux éducatifs, jeux de pure distraction. Aussi, les illustrations qui interviendront dans ce chapitre seront-elles exclusivement de cette nature.

A côté de plusieurs petits programmes illustratifs qui pourront, le cas échéant, être réintroduits dans des programmes plus importants, nous développerons de manière approfondie deux logiciels plus longs qui utilisent, le premier, l'ensemble de la gestion des couleurs ("Gibraltar", un jeu dans lequel un sous-marin doit

passer de l'Atlantique à la Méditerranée en évitant les destroyers qui patrouillent dans le détroit) et le second, les problèmes relatifs à la production de sons musicaux (orgue à mémoire sur sept octaves).

LA COULEUR SUR LE ZX-SPECTRUM

Le chapitre sur la couleur dans le manuel de base est particulièrement explicite ; il n'y a donc pas à insister longuement sur le fonctionnement des différentes commandes relatives à la couleur. Nous mettrons simplement l'accent sur plusieurs règles.

Les grandes règles de l'écriture couleur

- *Premièrement* : le ZX-Spectrum traite comme des couleurs, non seulement les couleurs proprement dites (couleurs du fond, **PAPER**, couleurs de l'encre, **INK**,) mais aussi la brillance (**BRIGHT**) et le clignotement (**FLASH**). Les deux couleurs, la brillance et le clignotement constituent autant d'*attributs* qui affectent obligatoirement chaque position de l'écran. Il y a huit couleurs, en comptant le noir et le blanc comme des couleurs, ce qui fait $64 = 8 \times 8$ combinaisons fond/encre. Il n'y a que deux possibilités de brillance, nulle (0) ou effective (1), et deux possibilités de clignotement, nulle (0) ou effective (1). Il en résulte $64 \times 2 \times 2 = 256$ possibilités d'attributs.

Sachant que les possibilités numériques d'un octet, en fonction des bits qui le composent, vont de 0 à 255, il suffit donc d'un octet pour rendre compte de l'ensemble des attributs relatifs à une position ; sachant également que la haute définition du ZX-Spectrum permet l'adressage de $256 \times 192 = 49152$ points, il faudrait, pour les seuls attributs, mobiliser une partie de mémoire très importante si l'attribution des caractéristiques devait se faire point par point. La solution retenue a donc consisté à prendre comme unité d'attribution, non le point, mais l'ensemble de $8 \times 8 = 64$ points d'une position d'impression de caractère.

Cette solution, originale à plus d'un titre, présente des inconvénients mais aussi de nombreux avantages : elle interdit l'impression de plus de deux couleurs sur une même position de caractère, ce qui exclut tout graphisme réellement raffiné ; en revanche, le gain de

temps est considérable lorsque l'on cherche à produire un effet rapide sur la totalité de l'écran, ce dont les programmeurs avisés sauront parfaitement tirer parti.

- *Deuxièmement*, la règle à retenir est la suivante : toutes les commandes de couleur, au sens large, ainsi que les deux commandes **INVERSE** et **OVER** qui ne commandent par les attributs, mais l'impression proprement dite des caractères, peuvent avoir un effet temporaire ou permanent ;

- *permanente* (c'est-à-dire jusqu'à la spécification inverse), lorsqu'elles sont introduites dans un programme en qualité d'instructions principales ;

- *temporaire* lorsqu'elles sont introduites à la suite d'une instruction d'impression ; elles affectent alors uniquement celle-ci et sont annulées une fois que celle-ci a été exécutée.

- *Troisièmement* : ces instructions sont très puissantes et peuvent être paramétrées, non seulement par des valeurs entières de 0 à 9, 0 à 1, 0-1, selon le cas, mais aussi par des expressions ou des variables ; elles sont donc pleinement programmables, ce qui en fait toute l'utilité.

- *Quatrièmement* : elles sont, d'une certaine manière, *intervertibles* par l'intermédiaire du mot Basic **CHR\$** ; en effet, en vertu d'une correspondance à bien connaître, **CHR\$**, suivi d'un nombre entier compris entre 16 et 21, équivaut respectivement aux ordres **INK**, **PAPER**, **FLASH**, **BRIGHT**, **INVERSE** et **OVER**.

Comme cette commande peut être affectée d'une expression, il en résulte qu'une même instruction, selon la valeur de la variable, peut aboutir à l'équivalent de commandes d'attributs différentes.

Voici, en particulier, un court programme (figure IV-1) assez peu parlant, mais très édifiant. Considérons la ligne 50 dont l'écriture est très hermétique.

```
10 RANDOMIZE
15 PRINT AT 1,1;"a"
20 LET a=INT (RND*4)+8
30 LET b=INT (RND*6)+16
40 LET c=(INT (RND*8))*(b<=17)
+ (INT (RND*2))*(b>17)
50 PRINT CHR$ a;CHR$ b+CHR$ c;
CHR$ (b*4+a+c);
55 PAUSE 100
60 GO TO 20
```

Figure IV-1

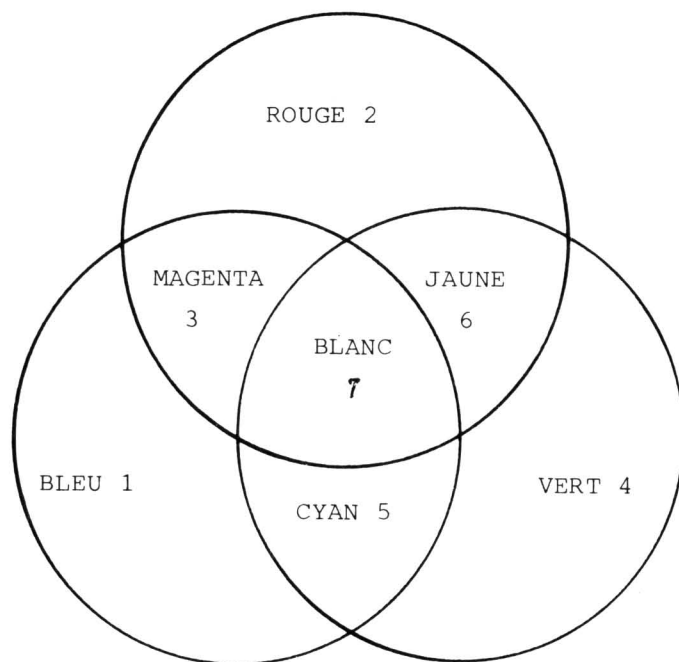


Figure IV-2

```

10 REM MELANGE DES COULEURS
15 REM -----
20 FOR I=0 TO 21
30 FOR J=0 TO 31
40 LET BLEU=(I>1 AND I<15 AND
J>7 AND J<23)
50 LET ROUGE=2*(I>5 AND I<20 AND
ND J>2 AND J<14)
60 LET VERT=4*(I>11 AND I<18 AND
ND J>10 AND J<30)
70 LET COULEUR=BLEU+VERT+ROUGE
80 PRINT PAPER COULEUR;AT I,J;
90 NEXT J: NEXT I

```

Figure IV-3

Cette simple ligne, basée sur l'utilisation de la commande **CHR\$**, couvre l'écran de choses invraisemblables en jouant sur la position d'écriture, **CHR\$ a**, sur les caractéristiques de l'impression **CHR\$ b + CHR\$ c**, enfin sur la nature du caractère à imprimer, quatrième **CHR\$**. Les variables aléatoires, a, b, c, sont calculées de telle sorte qu'elles entrent naturellement dans la zone des valeurs entières admissibles pour chaque **CHR\$**. Une **PAUSE**, ligne 55, a été introduite pour que l'utilisateur voie, caractère par caractère, ce qui se passe : il constatera des retours en arrière, des recouvrements, des effacements, des sauts, des changements de couleur, etc.

La puissance de ces instructions dites "de contrôle", pour la gestion des couleurs, est réelle, mais il est recommandé de ne les utiliser que dans des cas précis car, systématiquement utilisées, elles rendent les listes de programmes très difficilement compréhensibles.

Les huit couleurs de base du ZX-Spectrum

Le ZX-Spectrum possède huit couleurs, y compris le noir (qui est l'"absence de couleur") et le blanc qui est la superposition de toutes les couleurs. Les autres sont : bleu, rouge, pourpre, vert, bleu pâle et jaune. Les valeurs respectives de ces différentes couleurs, au sens courant, sont de 1 à 6. L'attribution de ces valeurs aux différentes couleurs n'a rien d'arbitraire : elles sont, au contraire, liées entre elles par des correspondances basées sur le principe de combinaison des trois couleurs de base qui sont le bleu, le vert et le rouge. Il en résulte qu'il y a trois couleurs de base, plus trois couleurs résultant de la combinaison des trois couleurs de base, plus une seule couleur résultant de la superposition des trois couleurs de base (figure IV-2).

Le petit programme de la figure IV-3 illustre immédiatement ce principe de superposition, en inscrivant trois rectangles qui se recoupent partiellement de façon à déterminer ainsi 9 zones d'écran : du pourtour, qui est noir (absence de couleur) à la zone où ils se superposent tous les trois (7, totalité des couleurs), ceci en utilisant les fonctions logiques qui effectuent les additions des valeurs des couleurs concernées. La maîtrise des substitutions contrôlées par programme de couleurs entre elles peut être très utile dans certaines applications de jeux où les changements de couleur peuvent correspondre à des états voulus. Il importe donc de se familiariser avec ce type de permutation.

48K :
65368

```

2000 DATA 170,85,170,85,170,85,1
70,85
2010 FOR I=0 TO 7: READ R: POKE
(32600+I),R: NEXT I: STOP
4000 REM COULEURS DU SPECTRE
4010 DATA 1,5,5,4,4,6,6,2,2,3
4015 FOR C=0 TO 4: READ I,J
4020 PRINT PAPER I; " "
4030 PRINT PAPER I; INK J; " "
4040 NEXT C
8000 STOP
9000 INK 0: PAPER 7: BORDER 7: C
LS

```

Figure IV-4

Les 36 couleurs du ZX-Spectrum

En fait, l'utilisateur n'est pas limité aux huit couleurs de base de l'ordinateur. En recourant à un artifice très simple, il est en mesure de disposer d'un nombre de couleurs beaucoup plus élevé, ceci par mélange. Il suffit pour cela de définir, dans la zone réservée aux caractères directement définis par l'utilisateur, un ou plusieurs caractères représentant des trames plus ou moins denses. Ce qui, sur une vidéo noir et blanc, serait représenté par un grisé, devient alors un mélange de couleurs, l'oeil ayant l'illusion d'une couleur intermédiaire entre celle du papier et celle de l'encre. Certes, la couleur obtenue ne paraîtra pas aussi homogène, mais l'effet de fondue peut être satisfaisant lorsque les deux couleurs de départ ne sont pas très contrastées : le rouge et le jaune, le vert et le bleu, etc.

Le petit programme de la *figure IV-4* utilise cette intéressante propriété. Après avoir défini un caractère de trame uniforme (ligne 2000), puis reclassé les couleurs de base du ZX-Spectrum dans un ordre qui rappelle davantage la succession des couleurs de l'arc-en-ciel (ligne 4010), il commande ensuite l'impression alternative d'une ligne de couleur franche et d'une ligne de cette même couleur mixée avec la couleur qui la suit. L'effet global sur l'écran est celui d'une succession beaucoup plus nuancée des couleurs.

L'introduction de la couleur dans les programmes

Quel que soit l'attrait que puisse exercer la couleur sur le programmeur, il faut avoir à l'esprit qu'un mauvais programme très coloré ne vaudra jamais un bon programme en noir et blanc.

Avec un ordinateur comme le ZX-Spectrum et les caractéristiques très spécifiques de ses commandes couleur, il est vivement recommandé d'élaborer tous ces programmes en deux temps :

- les commencer en noir et blanc pour vérifier qu'ils tournent bien, organiser l'enchaînement des routines, améliorer la rapidité de celles-ci, etc;

- ce n'est que dans un deuxième temps que l'on se précupera d'introduire la couleur en révisant l'ensemble des instructions d'impression. Autant l'introduction des attributs de couleur dans un programme achevé est facile, autant la mise au point d'un programme dans lequel figurent déjà quelques instructions couleur peut être pénible sur le ZX-Spectrum : la modification des attributs de la couleur vient constamment interférer avec les listes.

Il sera toujours recommandé de procéder de cette manière lorsque la couleur a seulement une finalité décorative.

Il en va différemment lorsque la couleur a une importance structurelle dans un programme, c'est-à-dire que des décisions sont prises dans le programme, en fonction de celle-ci. C'est ce qui se passe lorsque la fonction attribut, **ATTR**, teste la valeur des attributs d'une position d'affichage pour voir si l'on n'a pas affiché à une région interdite, par exemple dans un jeu interactif. Souvent d'ailleurs, il sera possible de tourner la difficulté en utilisant la fonction **SCREEN\$** à la place de la fonction **ATTR** lorsque, à des attributs différents, correspondent aussi des caractères imprimés différents.

Nous allons ainsi développer deux états progressifs d'un programme interactif assez vivant que l'on appelle le "programme du serpent ou de la chenille".

L'utilisateur déplace un serpent sur l'écran en se servant des différents curseurs, vers le haut, vers le bas, vers la gauche, vers la droite, de façon à absorber des proies qui apparaissent aléatoirement sur l'écran. A chaque proie, qui a une valeur comprise entre 1 et 9, il allonge d'autant de segments que la valeur de la proie absorbée, ce qui rend chaque fois son maniement plus difficile car le serpent n'a pas le droit de repasser sur lui-même.

La première version du jeu, qui n'est qu'en noir et blanc, a pour objet de démontrer le mécanisme de ce jeu très simple. Par rapport à cette première version, la deuxième, qui comporte la couleur, permet ainsi de

```

1 REM SERPENT NOIR ET BLANC
100 LET C=20: LET L=10
110 LET S$="": LET P=0
120 FOR A=10 TO 20
130 LET S$=S$+"10"+STR$ A
140 PRINT AT 10,A;"A"
150 NEXT A
160 GO SUB 500
199 IF INKEY$="" THEN GO TO 199
200 LET I$=INKEY$
205 IF I$>"4" AND I$<"9" THEN L
    ET M$=I$
210 LET L=L+(M$="6")-(M$="7"):
    LET C=C+(M$="8")-(M$="5")
220 IF L=22 OR C=32 OR C=-1 OR
    L=-1 THEN STOP
230 LET N$=SCREEN$(L,C)
240 IF N$>"1" AND N$<"9" THEN
    LET P=P+VAL N$: GO SUB 500
300 PRINT AT L,C;"A"
310 LET L$=STR$ L: IF L<10 THEN
    LET L$=" "+L$
320 LET C$=STR$ C: IF C<10 THEN
    LET C$=" "+C$
330 LET L1=VAL S$(1 TO 2): LET
    C1=VAL S$(3 TO 4)
335 IF P>0 THEN GO TO 350
340 PRINT AT L1,C1;" "
350 LET S$=S$+L$+C$
355 IF P=0 THEN LET S$=S$(5 TO
)
360 LET P=P-(P>0)
370 GO TO 200
500 PRINT AT (RND*21)+1,(RND*31
)+1;INT (RND*9)+1
510 RETURN

```

Figure IV-5

mesurer l'agrément, mais aussi la complication supplémentaire apportée par l'introduction de celle-ci.

Le programme "serpent" en noir et blanc

Au point de vue fonctionnel, le serpent est représenté par une chaîne de caractères qui se forme à la ligne 130, chaîne qui comprend quatre fois plus d'octets que le serpent ne contient de segments (10) car il faut, bien sûr, pour imprimer chaque segment à sa place, une indication relative à la ligne et une indication relative à la colonne. Chaque fois que le serpent reçoit une commande de direction, sur la base de la procédure interactive des lignes 200, 205 et 210, la chaîne représentative du serpent va se tronquer de ces quatre premiers éléments (ligne 355) et, au contraire, s'augmenter à l'autre extrémité des caractéristiques ligne-colonne de la nouvelle position d'impression (ligne 350); la ligne 340 a pour effet d'effacer sur l'écran le dernier segment du serpent à chaque mouvement.

A chaque déplacement, on teste le contenu de la case que doit occuper la tête du serpent (lignes 230-240, aux fins de vérifier qu'il ne s'agit pas d'une proie. Lorsque c'est le cas, le nombre de segments (P) s'augmente d'autant et l'effacement du segment de queue est "inhibé" jusqu'à ce que le serpent se soit allongé de la quantité équivalente. Les lignes 310 et 320 ont pour fonction d'introduire dans la chaîne, en deux chiffres, les numéros de ligne ou de colonne inférieurs à 10 pour ne pas perturber l'ordre des segments. Enfin, la routine 500 est celle qui imprime d'une manière aléatoire l'apparition des proies (voir figure VI-6).

Sous cette forme très simple, le jeu n'est pas programmé pour se bloquer lorsque le serpent revient sur lui-même. Il serait pourtant très facile de développer la ligne 240 en introduisant un test de plus dirigeant cette fois-ci, s'il est positif, vers la fin du jeu (test de la forme `IF N$ = "A", THEN STOP`). Nous avons préféré réserver ce cas à l'illustration de l'introduction d'une commande spécifique à la couleur, dans la version développée de ce jeu que nous présentons maintenant.

Le programme de "chenille" en couleur

Le programme précédent a été développé de plusieurs manières pour le rendre plus attractif. D'abord, les différents segments de la chenille ont un dessin prédéfini par l'utilisateur pour faire apparaître des petites pattes : c'est l'objet des routines 900 et suivantes dont le mécanisme est désormais connu du lecteur.

Le calcul d'un score a été introduit : la ligne 620 se contente, pour cela, de calculer la longueur du serpent.

Des petites routines musicales, très simples, sont introduites dans deux cas : lorsque la chenille a avalé une proie (ligne 510) ou lorsqu'elle heurte un rebord (lignes 630 et suivantes).

Enfin, on a introduit, à la ligne 220, un test pour vérifier que le serpent ne repasse pas sur lui-même (ce test est basé sur l'utilisation de la fonction `ATTR` qui implique, évidemment, que les attributs du serpent soient différents des attributs du reste de l'écran). Pour ce faire, il faudra donner des attributs différents à la feuille, aux diverses proies et à la chenille elle-même. Ici, la feuille est verte (`PAPER 4`). Les attributs du


```

1 REM CHENILLE COULEUR
2 REM -----
10 GO SUB 800
15 GO SUB 900
20 BORDER 5: PAPER 4
100 LET C=20: LET L=10
110 LET S$="": LET P=0
120 FOR A=10 TO 20
130 LET S$=S$+"10"+STR$ A
140 PRINT BRIGHT 1; PAPER 2; IN
K 0; AT 10, A; "0"
150 NEXT A
160 GO SUB 500
199 IF INKEY$="" THEN GO TO 199
200 LET I$=INKEY$
205 IF I$="4" AND I$<"9" THEN L
ET M$=I$
210 LET L=L+(M$="6")-(M$="7"):
LET C=C+(M$="8")-(M$="5")
220 IF ATTR(L,C)=80 OR L=22 OR
C=32 OR C=-1 OR L=-1 THEN GO TO
500
230 LET N$=SCREEN$(L,C)
240 IF N$>"1" AND N$<="9" THEN
LET P=P+VAL N$: GO SUB 500
300 PRINT BRIGHT 1; PAPER 2; IN
K 0; AT L,C; "0"
310 LET L$=STR$ L: IF L<10 THEN
LET L$=" "+L$
320 LET C$=STR$ C: IF C<10 THEN
LET C$=" "+C$
330 LET L1=VAL S$( TO 2): LET
C1=VAL S$(3 TO 4)
335 IF P>0 THEN GO TO 350
340 PRINT BRIGHT 0; AT L1,C1; " "
350 LET S$=S$+L$+C$
355 IF P=0 THEN LET S$=S$(5 TO
)
360 LET P=P-(P>0)
370 GO TO 200
499 REM
500 REM NOMBRE
501 REM
510 BEEP .1,2: BEEP .1,2: BEEP
.1,2: BEEP .3,15: BEEP .15,6
520 PRINT BRIGHT 1; FLASH 1; IN
K 7; AT (RND*20)+1, (RND*30)+1; INT
(RND*9)+1
530 RETURN
599 REM
600 REM MAUVAISE DIRECTION
601 REM
605 LET L=L-(L=22)+(L=-1)
610 LET C=C-(C=32)+(C=-1)
620 PRINT AT L,C; FLASH 1; " "; A
T 0,0; "POINTS="; LEN S$/4-10
630 FOR B=10 TO 40 STEP 10
640 BEEP .1,B
650 NEXT B
660 PAUSE 4E4
670 GO TO 200
799 REM
800 REM DESSIN DE LA FEUILLE
801 REM
805 FOR I=1 TO 704
810 PRINT PAPER 4; " ";
820 NEXT I: RETURN
899 REM

```

```

→ 900 REM DEF.SEGMENT CHENILLE
901 REM
905 DATA 36,126,195,90,90,195,1
26,36
910 FOR I=0 TO 7
920 READ C: POKE USA'"A"+I,C
930 NEXT I
940 RETURN
9000 BORDER 7: INK 0: PAPER 7: C
LS

```

Figure IV-6

serpent seront différents de l'entourage par sa couleur (PAPER 2), rouge, et par sa brillance. L'ensemble de ces attributs est égal à 80, soit : valeur de BRIGHT multipliée par 64 + valeur de PAPER multipliée par 8. Rappelons, à ce propos, que la valeur globale de l'attribut est égale à la somme des différents éléments de brillance, clignotement, papier et encre, avec les pondérations suivantes : FLASH = 128 * (0 ou 1), BRIGHT = 64 * (0 ou 1), PAPER = 8 * (0 à 7), INK = 1 * (0 à 7). (voir figure IV-6).

EXERCICE 1 : *Ecrire un logiciel de jeu très court qui remplisse l'écran d'astérisques imprimés d'une couleur différente à chaque ligne, puis fasse apparaître un petit enzyme glouton (que le petit caractère "a" de la touche "2" permet de bien figurer), que l'on déplace à l'aide des quatre curseurs et qui avale le plus d'astérisques possible, mais qui prene, chaque fois, la couleur de la ligne sur laquelle il évolue, par mimétisme.*

Le programme "Gibraltar"

Le programme Gibraltar est nettement plus important. On peut même considérer que, du point de vue de la longueur, il s'agit d'un programme-limite pour un jeu. En effet, au delà, on commence à enregistrer des problèmes de lenteur et il faut avoir recours, pour accélérer le rythme, à des routines en langage machine dont la technique ne sera exposée que plus tard.

L'organisation du jeu

Ce jeu a été spécialement conçu pour intégrer le plus grand nombre de fonctions et de commandes du ZX-Spectrum, ceci afin que son étude soit la plus profitable possible. Du point de vue de la construction, il repose uniquement sur des techniques éprouvées qui peuvent servir de base à de nombreuses réalisations.

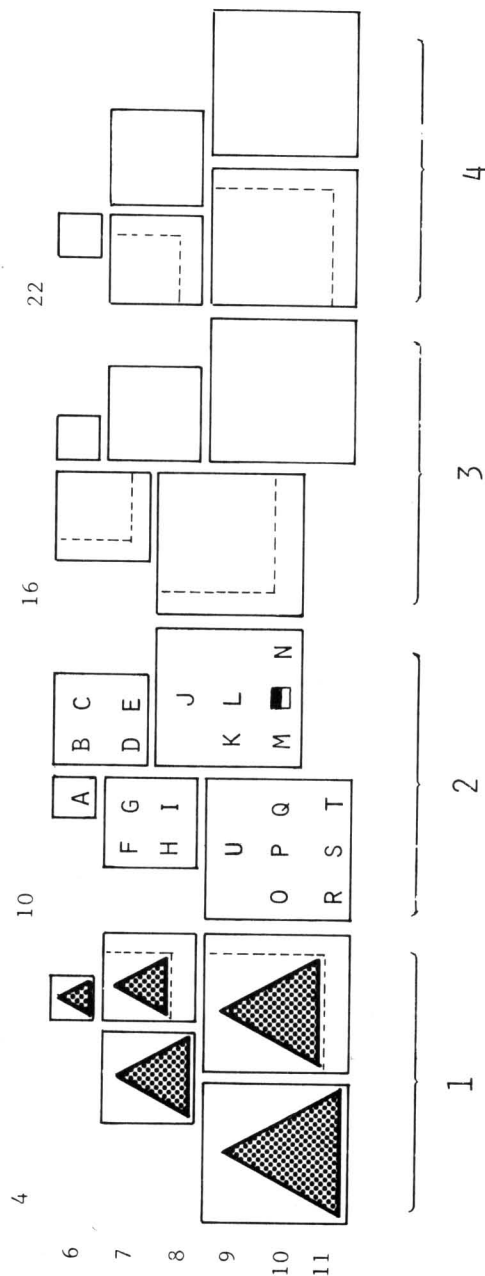


Figure IV-7

Le but du jeu est le suivant : un sous-marin cherche à passer de l'Atlantique à la Méditerranée, en traversant les eaux dangereuses du détroit de Gibraltar ; il lui faut éviter de se trouver à l'aplomb des bâtiments de guerre ennemis qui ne cessent de croiser en va-et-vient à travers le détroit. Il doit contourner également les mines suspendues entre deux eaux ; il ne peut, non plus, affleurer la surface car il est repéré immédiatement : dans chacun de ces cas, il explose. Le sous-marin peut être guidé avec les quatre curseurs, en haut, en bas, en avant et en arrière. Pour corser le tout, quand il croit avoir passé le détroit, il peut se trouver nez-à-nez avec un filot qui lui bloque le passage et l'expose à se trouver à l'aplomb d'un bâtiment.

Les mouvements du sous-marin (lignes 1190 et suivantes) sont commandés par la procédure interactive classique, joignant fonction logique et commande **INKEY\$**.

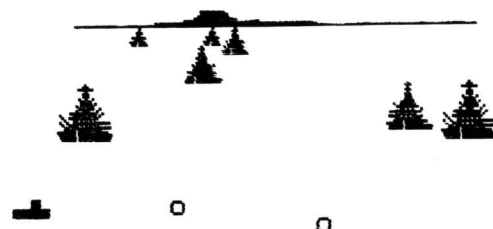
Le mouvement des unités de surface est plus complexe ; on a cherché à le reproduire avec le plus de réalisme possible. La totalité des caractères graphiques à définir par le joueur a été utilisée pour dessiner cinq silhouettes de croiseurs vus de face : ainsi, les croiseurs peuvent donner l'impression d'approcher à toute vitesse avec l'effet véridique de la perspective.

La largeur du détroit est divisée en quatre chenaux et l'apparition des bâtiments dans le lointain, à l'extrémité de chaque chenal, se fait de façon aléatoire. Une fois engagé, le bateau occupe la série des positions en quinconce qui le font progresser vers l'avant, en donnant l'impression de louvoyer. Pour chacune des 20 positions représentées sur la figure IV-7, l'ordinateur n'affiche rien si cette position n'est pas occupée par un bateau, ou affiche le bateau du format correspondant si la position est occupée.

La structure du programme (voir figure IV-8)

Les silhouettes s'inscrivent dans des carrés de largeur variant selon le format : format n° 1, une case ; format n° 2, une case et demie ; format n° 3, deux cases ; format n° 4 : deux cases et demie ; format n° 5, trois cases. Il y a donc 5 routines d'impression (2, 3, 4, 5, 6) * 1000. Les différentes positions de bateaux sont incluses dans deux tableaux, ligne et colonne (lignes 8000 et suivantes).

La progression des bateaux, sur chaque chenal, est commandée par une chaîne particulière, d'où 4 chaînes (lignes 1000 et suivantes). Chaque chaîne est une



```

10 REM GIBRALTAR
11 REM -----
300 RESTORE 9000
350 GO SUB 9000
450 RESTORE 8000
500 GO SUB 8000
600 BRIGHT 0: INK 0: PAPER 7: B
ORDER 7: CLS
700 GO SUB 7000
800 LET S=0: LET K=10
900 LET X=15: LET Y=2
990 REM
1000 REM MOUVEMENT DES BATEAUX
1001 REM
1010 FOR I=1 TO 4
1020 LET C$=B$(I)
1030 LET B$(I)=STR$(INT (RAND*1.
5))+B$(I,1 TO 4)
1040 FOR J=1 TO 5
1050 IF B$(I,J)=C$(J) THEN GO TO
1100
1055 PAPER 5
1060 GO SUB (1000+500*(B$(I,J)="
1")+1000*J)
1100 NEXT J
1180 REM
1190 REM MOUVEMENT SOUS-MARIN
1191 REM
1200 LET X=X+(INKEY$="6")-(INKEY
$="7"): LET Y=Y+(INKEY$="8")-(IN
KEY$="5")
1250 IF SCREEN$(X,Y+3)="#" THEN
LET Y=Y-1: GO TO 1200
1310 PRINT PAPER 1;AT X,Y-2;" ";
INVERSE 1;" "; INVERSE 0;" ";
PRINT PAPER 1;AT X-1,Y-2;" ";
PRINT PAPER 1;AT X+1,Y-2;" "
1320 LET S=S+(INKEY$="8")
1325 LET H=(Y<15)*INT ((Y+1)/5)+
(Y>15)*INT ((Y-3)/5)
1326 IF H=0 OR H=5 THEN GO TO 13
30
1327 IF B$(H,5)="1" THEN GO TO 1
500
1330 IF ATTR (X,Y+3)=208 THEN GO
TO 1500
1339 REM
1340 REM PASSAGE SAIN ET SAUF
1341 REM
1350 IF Y>=30 THEN BEEP .2,20: B
EEP .2,10: BEEP .2,20: PRINT PAP
ER 1;AT X,Y-2;" ": GO TO 900
1390 GO TO 1800
1490 REM
1500 REM EXPLOSION
1501 REM
1510 FOR T=-3 TO 1

```

```

1515 LET P=ABS (T+1)
1520 BRIGHT 1: PAPER P
1530 PRINT AT X-1,Y-1;" "
1535 PRINT AT X,Y-1;" "
1540 PRINT AT X+1,Y-1;" "
1550 BEEP .1,1
1560 NEXT T
1570 PRINT FLASH 1;AT 0,0;"SCORE
=";S
1580 STOP : GO TO 600
1790 REM
1800 REM FILET
1801 REM
1810 LET K=K-1
1820 IF K<>0 THEN GO TO 1900
1825 LET R=INT (RAND*2)
1830 FOR F=13 TO 20
1840 PRINT PAPER 1;AT F,30;CHR$
(35+93*R)
1850 NEXT F
1860 LET K=10
1900 NEXT I
1950 GO TO 1000
1980 REM
1990 REM IMPRESSION BATEAUX
1991 REM
1995 REM Format no 1
1996 REM
2000 PRINT AT L(I,J),C(I,J);" "
2100 RETURN
2500 PRINT AT L(I,J),C(I,J);" "
2600 RETURN
2900 REM
2950 REM Format no 2
2960 REM
3000 PRINT AT L(I,J),C(I,J);" "
3010 PRINT AT L(I,J)+1,C(I,J);" "
3020 RETURN
3500 PRINT AT L(I,J),C(I,J);" "
3510 PRINT AT L(I,J)+1,C(I,J);" "
3520 RETURN
3900 REM
3950 REM Format no 3
3960 REM
4000 PRINT AT L(I,J),C(I,J);" "
4010 PRINT AT L(I,J)+1,C(I,J);" "
4020 RETURN
4500 PRINT AT L(I,J),C(I,J);" "
4510 PRINT AT L(I,J)+1,C(I,J);" "
4520 RETURN
4900 REM
4950 REM Format no 4
4960 REM
5000 PRINT AT L(I,J),C(I,J);" "
5010 PRINT AT L(I,J)+1,C(I,J)-1;
" "
5020 PRINT AT L(I,J)+2,C(I,J)-1;
" "
5030 RETURN
5500 PRINT AT L(I,J),C(I,J);" "
5510 PRINT AT L(I,J)+1,C(I,J)-1;
" "
5520 PRINT AT L(I,J)+2,C(I,J)-1;
" "; INVERSE 1;" "; INVERSE 0;" "
5530 RETURN

```

```

5900 REM
5950 REM Format no 5
5960 REM
6000 PRINT AT L(I,J),C(I,J);" "
6010 PRINT AT L(I,J)+1,C(I,J)-1;
6020 PRINT AT L(I,J)+2,C(I,J)-1;
6030 RETURN
6500 PRINT AT L(I,J),C(I,J);"1"
6510 PRINT AT L(I,J)+1,C(I,J)-1;
6520 PRINT AT L(I,J)+2,C(I,J)-1;
6530 RETURN
6900 REM
7000 REM DESSIN PAYSAGE
7001 REM
7010 FOR X=0 TO 21
7020 LET P=6*(X<=5)+5*(X>5 AND X
<=12)+(X>12 AND X<=20)
7030 PRINT PAPER P;AT X,0;"
7040 NEXT X
7045 INK 0
7046 REM
7047 REM DESSIN MONTAGNE
7048 REM
7050 FOR X=1 TO 7
7060 LET Y=EXP X
7070 PLOT 100,135-X: DRAW 5+Y*.1
7075 PLOT 100,135-X: DRAW -(Y*.0
5+5),0
7080 NEXT X
7090 REM
7100 REM DISPOSITION MINES
7105 REM
7110 FOR G=1 TO 2
7120 LET RL=INT (RAND*8)
7130 LET RC=INT (RAND*23)
7140 PRINT BRIGHT 1; FLASH 1; PA
PER 2; AT RL+13,RC+5;"0"
7150 NEXT G
7200 RETURN
7900 REM
8000 REM EMPLACEMENTS BATEAUX
8001 REM
8100 REM Lignes
8101 REM
8300 DIM L(4,5): DIM C(4,5)
8350 DATA 6,7,7,9,9,6,6,7,8,9,6,
6,7,8,9,6,7,7,9,9
8400 FOR I=1 TO 4: FOR J=1 TO 5
8420 READ L(I,J)
8430 NEXT J: NEXT I
8435 REM
8440 REM Colonnes
8450 REM
8490 DATA 8,8,6,8,5,12,13,11,14,
11,19,17,19,17,20,23,22,24,23,25
8500 FOR I=1 TO 4: FOR J=1 TO 5
8510 READ C(I,J)
8520 NEXT J: NEXT I
8530 DIM B$(4,5)
8540 FOR I=1 TO 4
8550 LET B$(I)=""00000"
8560 NEXT I
8590 RETURN
8990 REM
9000 REM DEFINITION BATEAUX
9001 REM

```

```

9100 DATA 8,8,28,24,126,60,238,1
11
9110 DATA 2,7,2,7,7,15,47,31,0,0
0,0,0,0,192,128
9120 DATA 95,55,247,119,0,0,0,0,
192,192,224,240
9130 DATA 0,0,0,0,1,3,1,3,3,7,3,
7
9140 DATA 0,128,0,128,128,192,12
0,192
9150 DATA 23,15,31,127,31,119,24
7,247
9160 DATA 232,240,224,252,240,25
0,255,255
9170 DATA 64,224,64,224,64,224,2
04,240
9180 DATA 3,19,15,3,127,15,125,1
5
9190 DATA 240,248,252,81,254,248
95,254
9200 DATA 30,254,126,126,0,0,0,0
9210 DATA 192,0,240,240,0,0,0,0
9220 DATA 0,0,0,0,7,65,39,17
9230 DATA 82,191,127,156,255,255
255,170
9240 DATA 128,0,32,64,144,160,19
0,128
9250 DATA 73,39,27,31,255,127,12
7,63
9260 DATA 255,171,255,127,127,12
7,127
9270 DATA 254,192,254,224,252,25
0,254,254
9280 DATA 8,28,127,8,28,28,73,62
9290 DATA 8,28,127,8,28,28,73,62
9300 FOR I=0 TO 167: READ D
9310 POKE (32600+I),D
9320 NEXT I
9330 RETURN
9900 BRIGHT 0: INK 0: PAPER 7: B
ORDER 7: CLS

```

48K :
55368

Figure IV-8

succession de "0" (absence de bateau) et de "1" (présence d'un bateau). A chaque cycle du jeu, la chaîne se décale d'une unité et la première case, celle de l'entrée des bateaux dans un chenal, se charge d'un 1 ou d'un 0 aléatoirement. A ce propos, il est possible d'intervenir sur la difficulté du jeu en modifiant le coefficient de la fonction aléatoire de la ligne 1030. Il a été choisi de 1.5 ici, mais on peut rendre le jeu plus difficile en passant à 2 et plus, plus facile en se rapprochant de la valeur 1.

Pour augmenter la vitesse, on ne passe à la routine d'impression, dans une position, que quand l'état de cette position a changé. Si elle n'a pas changé parce que, précédemment occupée par un bateau, elle se trouve occupée par un bateau qui suivait immédiatement, on saute directement à l'analyse de la position suivante. Quand la situation d'un chenal est entièrement examinée, l'ordinateur passe à l'étude de la chaîne suivante.

La structure générale du jeu est divisible en trois. Les sous-programmes utilitaires de préparation qui ne sont destinés à être lus qu'une fois sont relégués en fin de programme : coloriage des éléments de paysage, etc. (lignes 7000 et suivantes). Les sous-programmes, qui ne sont lus qu'une fois sur cinq, c'est-à-dire l'impression des bateaux, sont placés en milieu de programme (lignes 1990 et suivantes). Enfin, le coeur du jeu est placé en tout début de programme (lignes 1000 à 1900). Il est de règle, en effet, dans une perspective de rapidité, de toujours placer en début les instructions les plus fréquemment lues.

Globalement, le cycle fondamental du jeu comprend donc les opérations suivantes : avancement d'une position de l'ensemble des bateaux d'un chenal, plus possibilité de 1 mouvement pour le sous-marin.

EXERCICE 2 : Comment modifier la ligne 1330 pour que le sous-marin explose également lorsqu'il touche le fond ou qu'il franchit la surface ?

LE SON SUR LE ZX-SPECTRUM

Le maniement de la commande BEEP, suivie des deux arguments de durée et de hauteur de note, ne présente en soi aucune difficulté et le jeu de Gibraltar, ainsi que celui de la chenille, ont déjà mis en oeuvre cette commande d'une manière simple. Mais il est possible de tirer beaucoup plus de cette commande, en entrant davantage dans la compréhension de son mécanisme.

Les bruitages

Il faut, dans un premier temps, apprendre à l'utiliser de façon judicieuse dans des boucles pour obtenir des effets de bruitage moins ternes et linéaires que la succession plate de l'instruction répétée plusieurs fois. Il est intéressant également de jouer sur le pas d'incrémentation, qu'il ne soit pas constant, mais dépende d'une variable. Ces petites techniques permettent la production de sons qui ne sont certes pas d'un réalisme poussé, étant donné les faibles possibilités du ZX-Spectrum (qui ne peut produire qu'un son à la fois et pas d'harmonique), mais qui peuvent agrémenter très valablement un programme de jeu, surtout lorsque la prise d'écouteur

```
5 REM RIANEMENT IRONIQUE
6 REM -----
10 FOR K=0 TO 1000
15 LET C=COS K+.5
20 BEEP 0.1,-10*C+1: BEEP 0.1,
-20*C+1: BEEP 0.1,-30*C+1
30 NEXT K
```

```
5 REM SIRENE DE POLICE
6 REM -----
10 FOR K=1 TO 1000
20 BEEP 0.05,5: BEEP 0.05,1: B
EEP 0.05,-60: BEEP 0.05,1
30 NEXT K
```

```
5 REM BRUIT DE CHUTE
6 REM -----
10 LET E=.5
20 FOR B=30 TO 10 STEP -E
30 BEEP E/100,B
40 LET E=E+.1
50 NEXT B
60 STOP
```

```
5 REM HELICOPTERE
6 REM -----
10 FOR K=0 TO 1000 STEP .1
20 LET C=COS K
30 BEEP 0.01,(-20+5*C)
40 NEXT K
```

```
5 REM BRUIT DE TRAIN
6 REM -----
10 FOR K=0 TO 1000
20 BEEP 0.05,-2: BEEP 0.05,-2
40 BEEP 0.5,-10: BEEP 0.2,-60:
BEEP 0.2,60
50 NEXT K
```

Figure IV-9

est reliée à la prise micro de l'enregistreur à cassette pour obtenir une petite amplification. Les petits programmes de la figure IV-9 montrent quelques-unes des techniques utilisables pour cela.

On notera en particulier, dans l'exemple n° 1 et l'exemple n° 4, l'utilisation d'une fonction circulaire (cosinus) pour obtenir des effets de battements intéressants. Il faut simplement éviter que les valeurs correspondant à la hauteur de la note introduite ne dépassent pas les limites autorisées, c'est-à-dire -60, où l'on n'obtient qu'une vibration, et +70, où le bruit est à peine audible parce qu'il est trop aigu.

Le ZX-Spectrum et l'apprentissage des lois de l'acoustique musicale

Il est certain que le BEEP du ZX-Spectrum manque un peu d'amplitude. Mais sa précision est bonne et les notes sonnent assez juste, ce qui fait qu'il est fort possible de le transformer en instrument d'étude théorique élémentaire de la musique, pour la construction de jeux musicaux.

Pour comprendre tout le parti que l'on peut en tirer, il faut savoir qu'à l'organisation artistique des notes, qui constitue la musique, correspond une étude physique de ces notes qui est : l'acoustique musicale.

Si l'ordinateur est capable de reconstituer des notes très justes sur la simple indication d'un argument de hauteur, c'est que toutes les notes peuvent être réduites à un phénomène sonore mesurable et qu'elles sont liées entre elles par une loi elle-même précise.

Une note est un son continu d'une certaine fréquence. Le phénomène qui est à l'origine du son est une vibration : le ZX-Spectrum ouvre et ferme un circuit, un certain nombre de fois par seconde ; l'ensemble ouverture-fermeture dure un certain temps qui constitue la "période".

Le nombre de ces périodes par seconde est appelé la "fréquence". Ainsi l'ouverture et la fermeture, 261 fois par seconde, du circuit du haut-parleur produira la note *do*. Cette note, qui est la note de référence, la base de la gamme ordinaire, reçoit naturellement, pour ceci, la valeur 0 et BEEP,1,0 signifie donc : faire sonner *do* pendant une seconde.

Si cette fréquence de base est multipliée par deux, on obtient un son de 512 de fréquence qui est le *do* de la gamme immédiatement supérieure. Mais pour atteindre ce *do*, il aura fallu passer par une succession de notes qui sont *do* dièse (ou *ré* bémol), *ré*, *ré* dièse (ou *mi* bémol), *mi*, *fa*, *fa* dièse (ou *sol* bémol), *sol*, *sol* dièse (ou *la* bémol), *la*, *la* dièse (ou *si* bémol), c'est-à-dire que d'un *do* à l'autre, il va exister 12 intervalles.

La musique moderne, basée sur les grandes innovations de Jean-Sébastien Bach, a habitué l'oreille à considérer que ces 12 notes sont séparées par le même intervalle, c'est-à-dire qu'il suffit de multiplier toujours la fréquence d'une note par une certaine valeur pour obtenir la note immédiatement supérieure. Si l'on veut obtenir un doublement sur 12 notes, il est clair que ce

multiplicateur, qui est la raison de la progression géométrique représentée par la suite des fréquences, sera la racine douzième de 2, soit 1.059463094. Dans cette suite de notes, il y a des sons privilégiés qui forment entre eux des rapports harmoniques plus beaux à l'oreille : ce sont les notes de base, les 7 notes de la gamme.

Voilà le mystère du deuxième argument de la fonction BEEP : il s'agit simplement du nombre de fois qu'il a fallu multiplier la fréquence de la note de base par le multiplicateur pour obtenir le son cherché. Les arguments négatifs correspondent, non plus à des multiplications, mais à des divisions. La succession des notes est ainsi réduite à une simple fonction mathématique, ce qui sera de la plus grande utilité :

- soit pour représenter les notes sur un graphique (portée) ;
- soit pour, et c'est ce que nous ferons dans une deuxième application, aller directement au devant du système en lui épargnant d'avoir à faire la conversion de ce chiffre simple pour retrouver la valeur complexe de la fréquence. Cette méthode implique un court passage au langage machine, selon une technique facile qui trouvera, ultérieurement, son explication.

L'écriture musicale à l'écran

La première application intéressante de cette existence d'une relation stricte entre les différentes notes est celle qui consiste à faire apparaître sur l'écran les notes d'une musique, en même temps qu'elles sont BEEPées. Ces notes viendront se placer exactement à la hauteur correspondante sur la portée : l'effet général est amusant et assez joli, même si le temps pris par le dessin des notes, spécialement les notes crochétées (doubles croches notamment), vient hacher un peu le déroulement du son. Pour cette raison, la démonstration n'est vraiment concluante qu'avec des musiques relativement martelées, comme des marches ou toute forme de musique un tant soit peu scandée. Ici, c'est la "Chevauchée des Walkyries" de Richard Wagner qui a été retenue (voir figure IV-10 de la page ci-après).

EXERCICE 3 : Pour apprécier la différence de rapidité, et donc la continuité dans les notes, le lecteur peut s'exercer à recomposer le programme en supprimant tout ce qui n'est pas production musicale. Il lui est suggéré



```

5 REM CHEVAUCHEE WALKYRIES
6 REM -----
10 LET C=-12
20 FOR I=1 TO 14 STEP 3
30 PLOT 0,160-I: DRAW 255,0
40 NEXT I
50 LET K=-1
100 FOR I=1 TO 22
105 LET K=K+1: IF K=5 THEN GO 5
UB 400
110 READ A,B: GO SUB 300: BEEP
A,B+C: NEXT I
115 CLS
120 LET C=C+2: RESTORE 1000: GO
TO 20
310 CIRCLE I*11,154+B-(B>=3),1
320 PLOT I*11+2,154+B-(B>=3)-1:
DRAW 0,8
330 IF A>=1 THEN RETURN
340 DRAW 4,-1,-PI/4: IF A>=.5 T
HEN RETURN
350 PLOT I*11+2,159+B-(B>=3): D
RAW 4,-1,-PI/4: RETURN
400 LET K=0: PLOT I*11,147: DRA
W 0,12: LET I=I+1: RETURN
1010 DATA .5,-1,.25,-6,.5,-1,1,3
,1,-1
1020 DATA .5,3,.25,-1,.5,3,1,6,1
,3
1030 DATA .5,6,.25,3,.5,6,1,10,-
,2
1040 DATA .5,3,.25,-2,.5,3,1.5,6

```

Figure IV-10

de placer la suite des notes dans une boucle qui rejoue l'air en incrémentant, chaque fois, la valeur de deux points pour obtenir l'air sur différents tons.

Un programme d'orgue électronique à mémoire

Pour ceux qui ne savent pas jouer d'un instrument, mais qui aiment exercer leur oreille, il est intéressant de disposer d'un programme (figure IV-11) qui permette de composer une musique par tâtonnement, de mémoriser les séquences qu'on a réussi à jouer correctement et de faire, ensuite, imprimer la succession des notes avec leur durée et leur valeur pour pouvoir réintroduire l'ensemble de la musique ainsi constituée dans un programme musical différent. Ce type de programme, très utile, présente un certain nombre de difficultés qui n'ont pu être surmontées que par le recours partiel au

```

10 REM ORGUE A MEMOIRE
11 REM (7 octaves)
12 REM *****
20 REM
100 REM MENU
110 REM
120 CLS
130 PRINT INVERSE 1;"VOULEZ-VOU
S:"; PRINT
140 PRINT "-initialiser le prog
ramme ?(9)";
150 PRINT "-essayer une suite d
e notes ?(1)";
160 PRINT "-memoriser la suite
?(2)";
170 PRINT "-executer la suite ?
(3)";
180 PRINT "-voir les valeur des
notes ?(4)";
190 INPUT A: GO TO A*1000
900 PAUSE 4E4
990 REM
000 REM ESSAIS
1001>REM
1010 LET G=2: LET M=0
1015 CLS: PRINT INVERSE 1;"Pour
arreter les essais presser "
SHIFT/ BREAK
1020 PAUSE 4E4
1070 IF INKEY$="" THEN GO TO 107
0
1080 IF INKEY$="6" THEN LET G=G-
(G>0): GO TO 1095
1090 IF INKEY$="7" THEN LET G=G+
(G<6)
1095 IF INKEY$="6" OR INKEY$="7"
OR INKEY$="" THEN GO TO 1095
1100 LET B=CODE INKEY$-64+256*G
1110 LET F=1
1200 POKE 32600,D(B,2): POKE 326
01,D(B,1)
1250 POKE 32602,H(B,2): POKE 326
03,H(B,1)
1300 LET K=USR 32604
1310 IF INKEY$<>"" THEN LET F=F+
1: GO TO 1300
1320 IF M=1 THEN GO SUB 2200
1350 GO TO 1070
1900 REM
2000 REM MEMORISATION
2010 REM
2025 PRINT "Pour arreter: BREAK"
2100 CLS: INPUT FLASH 1;"Memori
sation/numero depart ? ";J
2120 PRINT INVERSE 1;AT 0,0;"MEM
ORISATION EN COURS": PRINT
2130 LET M=1: GO TO 1020
2200 LET N(J,4)=H(B,1): LET N(J,
3)=H(B,2)
2250 LET T=(D(B,1)*256+D(B,2))*F
2260 LET N(J,2)=INT (T/256)
2270 LET N(J,1)=INT ((T/256-INT
(T/256))*256)
2280 PRINT J;" "; LET J=J+1
2300 RETURN
9999 REM
0000 REM EXECUTION SUITE
0001 REM
0050 INPUT INVERSE 1;"Note de de
part ?";C
0060 INPUT INVERSE 1;"Note d'arr
ivee ?";D

```

```

0100 FOR K=0 TO 0
0200 FOR L=1 TO 4
0310 POKE (32599+L),N(K,L)
0320 NEXT L
0330 LET U=USR 32604
0400 NEXT K
0900 GO TO 100
0990 REM
40000 REM VALEURS DE LA SUITE
4010 REM
4020 INPUT INVERSE 1;"Note de de
part ?";C
4030 INPUT INVERSE 1;"Note d'arr
ivee ?";Q
4040 CLS
0800 PRINT INVERSE 1;"VALEUR DES
NOTES:"
1100 FOR V=C TO Q
1120 LET H=N(V,3)+256*N(V,4)
1130 LET H=437500/(H+30.125)
1140 LET D=N(V,1)+256*N(V,2)
1150 LET D=(INT ((D/H)*100))/100
1160 LET H=INT ((12*LN (H/256))/L
(V,2)+.15)
1170 PRINT "no.:";V;" duree=";D;"
hauteur=";H
4180 NEXT V
4190 STOP : GO TO 100
4500 REM
50000 REM TABLE FREQUENCES
5005 CLS
5010 DIM D(175,2): DIM H(175,2)
5020 LET A$="QASEDRFGYHUJIKLP"
5030 FOR G=0 TO 6
5040 LET D=53.5*(2^G)
5100 FOR I=1 TO 16
5110 LET D=D*1.059463094
5120 LET H=437500/D-30.125
5130 LET X=CODE A$(I)-64+25*G
5140 LET D1=D/10
5210 LET D(X,1)=INT (D1/256)
5220 LET H(X,1)=INT (H/256)
5230 LET D(X,2)=INT ((D1/256-INT
(D1/256))*256)
5240 LET H(X,2)=INT ((H/256-INT
(H/256))*256)
5300 NEXT I: NEXT G
5400 DIM N(100,4)
5400 REM
5500 REM ROUTINE LANGAGE MACHINE
5501 REM
5510 DATA 0,0,0,0,237,91,88,127,
42,90,127,205,181,03,201
5520 FOR I=0 TO 14: READ A
5530 POKE (32600+I),A: NEXT I
5540 GO TO 100

```

Figure IV-11

langage machine, mais sous une forme très simple qui n'a rien pour effrayer le lecteur ; celui-ci pourra toujours revenir sur ce programme après s'être initié au langage machine, au chapitre prévu à cet effet.

Ces difficultés tiennent à une grande raison : la fonction **BEEP** ne s'interrompt pas lorsque l'on presse la touche **BREAK**. Il faut donc programmer la longueur d'un son avant de l'exécuter ; il n'est pas possible d'interrompre son exécution pour lui donner la longueur voulue, en temps réel, celle-ci doit être fixée *a priori*. Aussi faut-il fractionner le son en une suite de sons beaucoup plus courts.

Pour atténuer l'effet de discontinuité qui en résulte, il faut attaquer directement la routine ROM qui déclenche le son du petit haut-parleur en lui proposant, chaque fois, des valeurs adaptées. Il est donc nécessaire de faire correspondre aux touches du clavier que l'on a décidé de sonoriser une longue table comprenant des valeurs pré-mémorisées (période fondamentale et fréquence de chaque son) et envoyer celles-ci à deux registres du microprocesseur, les registres DE et HL qui contiennent, le second, la fréquence, le premier, la durée.

On pourrait penser que le registre DE, qui contient la durée, est chargé avec une valeur absolue simple : par exemple, 10 pour une seconde, 1 pour un dixième de seconde, etc. En fait, il n'en est rien et il est nécessaire de tenir compte du fait que, plus la fréquence d'un son est élevée, et plus il est exécuté rapidement. Aussi, le registre DE doit-il être chargé avec la valeur $F \times T$, c'est-à-dire : le temps souhaité multiplié par la fréquence.

Quant à la valeur qui va dans le registre paire HL, elle est le résultat de l'expression : 437500 divisé par la fréquence - 30.125.

Comme ces valeurs sont destinées à être lues par une routine en langage machine, on doit, pour des raisons que nous expliquerons ultérieurement dans le chapitre consacré à ces problèmes, coder ces valeurs sur deux octets. Cela signifie que chaque pression de touche sonorisée va extraire quatre valeurs différentes dans un tableau. Si l'on veut sonoriser, par exemple, les touches 1 à 8, il faut donc un tableau de (8×4) .

La sonorisation des touches du clavier

Pour faire ressembler le plus possible le ZX-Spectrum à un piano (!), nous avons résolu d'utiliser les deux

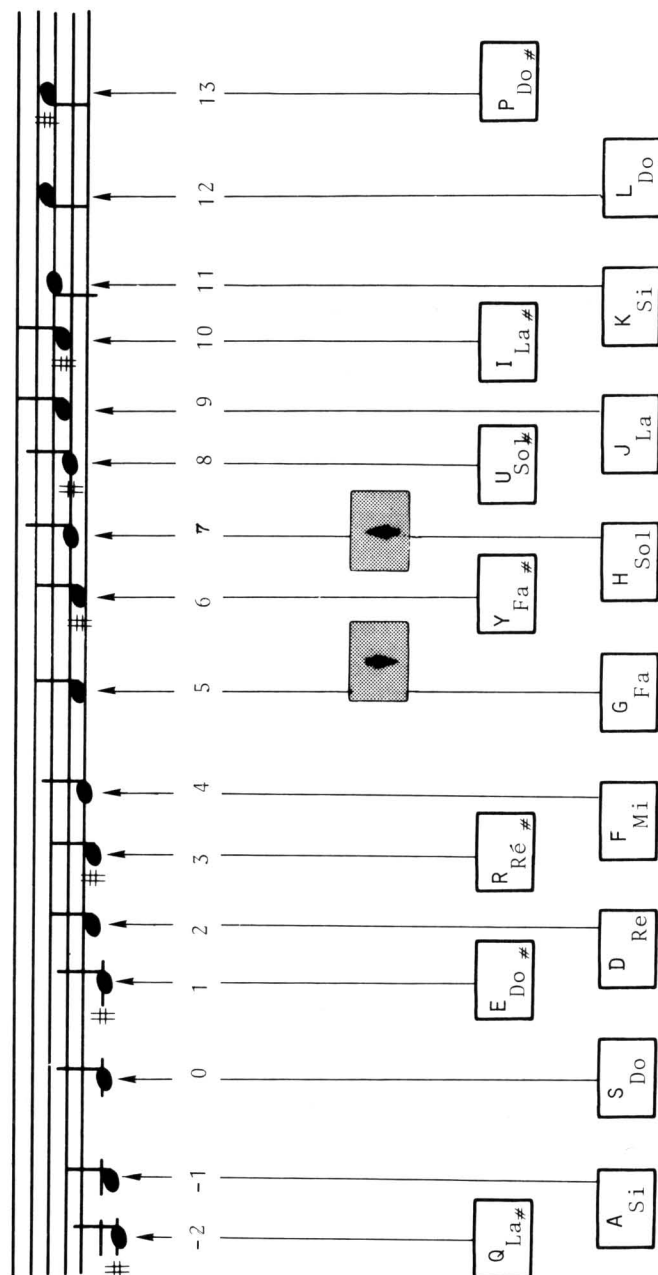


Figure IV-12

rangées médianes, selon la correspondance représentée dans la *figure IV-12* ci-contre. Les demi-tons figurent donc sur la rangée supérieure. Le clavier couvre ainsi plus qu'un octave, en fait 16 notes au lieu de 12, ce qui permet de déborder un peu au-dessus et au-dessous pour composer des musiques simples. Si l'on veut changer de clé, c'est-à-dire augmenter la base, comme le programme permet de disposer de 7 octaves, il suffit de faire jouer les curseurs vers le haut ou vers le bas (curseurs "6" et "7").

La mise en oeuvre du programme est simple : un menu propose différentes actions, et le programme est prévu pour revenir au menu dès qu'il y a rupture de séquence. Lorsque cette rupture se produit d'elle-même, pour une raison ou pour une autre (par exemple, lorsque l'on a pressé sur une touche non sonorisée), on peut toujours revenir à ce menu en faisant `GOTO 100`.

Tel quel, le programme permet d'entrer un air d'une centaine de notes. Il est possible d'obtenir davantage de place en réduisant le nombre des gammes possibles. Il faut, pour cela, jouer sur la longueur du tableau initialisé en ligne 9010, qui est prévu ici de 175 valeurs de notes, et simultanément sur le nombre des gammes prévues (il y en a ici 7) à la boucle 9030 (G comme gamme).

La composition d'un air est réellement rapide. On commence par essayer la séquence : si cette séquence est correcte, on la rejoue en mémorisation, on la fait exécuter, cette fois-ci d'affilée, sans le moindre chevrottement de notes et, si le résultat est satisfaisant, on demande l'affichage des valeurs de notes. Il est naturellement possible de corriger un air en demandant la mémorisation à partir de la note qui n'est pas satisfaisante.

La structure du programme

Sa compréhension, dans le détail, requiert la lecture préalable des deux derniers chapitres de cet ouvrage, à cause du chargement direct dans la mémoire par les instructions **POKE** et à cause de la routine en langage machine.

En voici, toutefois, les grandes lignes : la procédure d'initialisation, qui commence à 9000, affecte aux 16 touches retenues par la sonorisation une valeur de note dans une gamme ; puis, pour constituer les 7 gammes, on ouvre un tableau. Selon la gamme où on se trouvera, la touche réagira donc différemment. Ainsi,

lorsqu'une des 16 touches sera pressée, le code de cette touche sera calculé et on cherchera les valeurs correspondant à ce code dans celle des 7 sous-parties du tableau qui est concernée par la gamme en cours.

Les 350 valeurs de durée et de hauteur sont chargées, au moment de l'initialisation, par le calcul que nous avons exposé plus haut, c'est-à-dire que chaque note est multipliée par une certaine valeur pour obtenir la note au-dessus.

Simultanément, au moment de l'initialisation, s'ouvre un tableau de 100 valeurs numériques qui permettra de conserver les valeurs de notes créées au moment de la mémorisation.

La routine de langage machine est une routine très simple de 14 codes qui consiste à lire des contenus de mémoire (4 emplacements de mémoire) et à les envoyer à la routine de mémoire morte qui exécute l'ordre **BEEP**. Cette routine s'appelle par l'instruction **LET K** ou **LET U = USR 32604**. Il en résulte, ce qui peut surprendre le lecteur à ce niveau, qu'il a en face de lui un programme qui ne fait, à aucun moment, appel à l'instruction **BEEP**.

L'appel de la routine en langage machine est effectué par chaque pression de touche à l'issue du sous-programme central qui est celui qui commence à la ligne 1070. Ce sous-programme teste d'abord si l'on demande un changement de clé vers le haut ou vers le bas, puis quel est le code de la touche pressée ; il calcule l'emplacement des valeurs de notes correspondantes dans le grand tableau ; il place ces valeurs dans deux octets qui font partie de la routine en langage machine et envoie directement à celle-ci pour la lecture de ces notes. Si la mémorisation est demandée, ces notes sont également entrées sous forme codée dans le second tableau qui est celui des notes mémorisées.

EXERCICE 4 : Construire un petit programme éducatif coloré qui fasse correspondre, aux touches "1" à "8", les huit valeurs de la gamme la plus simple, do, ré, mi, fa, sol, la, si, do, et qui affiche sur l'écran, en clair, chaque note pressée en même temps que celle-ci est jouée, le nom de la note apparaissant à sa place relative sur une trame de huit lignes de couleurs différentes.

CHAPITRE V

LA MEMOIRE DU ZX SPECTRUM

Le lecteur sait qu'il existe deux modèles de ZX-Spectrum : un ZX-Spectrum 16 K et un ZX-Spectrum 48 K. Les deux appareils comprennent en fait, respectivement, 64 K et 32 K. Dans un cas comme dans l'autre, une partie importante de la mémoire, équivalente à 16 K (16000 octets environ), est composée d'octets qui ne peuvent être modifiés par l'utilisateur et sont appelés, de ce fait, **ROM** (Read Only Memory). Rien n'est moins mort que cette mémoire qui peut, au contraire, être exploitée de façon très utile et qui, de toute façon, fonctionne continuellement de deux manières : d'abord en interprétant le langage Basic pour le transformer en codes fondamentaux compréhensibles pour le microprocesseur, ensuite en exploitant les calculs et en lisant des tables. Nous verrons ultérieurement de quelle manière il est possible d'accéder aux routines les plus intéressantes de cette ROM, pour les incorporer dans un programme personnel.

Restent donc respectivement 16 K et 48 K pour la mémoire composée d'octets modifiables que l'utilisateur peut charger avec les valeurs qu'il désire. C'est la mémoire **RAM** (Random Access Memory) ou "mémoire vive". Cette fois-ci, c'est le vocabulaire anglo-saxon qui ne paraît pas tout à fait adapté car, en programmation, l'accès aux octets libres est rien moins qu'aléatoire, mais défini d'une façon volontaire. Il faut cependant avoir à l'esprit que la gestion de cette mémoire vive n'est pas entièrement le fait de l'utilisateur, mais que le

système y effectue constamment des allocations pour stocker un certain nombre de données de travail, ou protéger certains emplacements dans lesquels vont figurer des résultats provisoires (la pile) ou des variables définies par l'utilisateur (zone des variables). Ce sont des variables spécialisées qui délimitent les zones pour éviter toute forme de recouvrement. Dans la suite de cet exposé, c'est plus précisément la version de base 16K qui sera examinée, mais le lecteur avisé comprendra vite qu'il suffit d'ajouter le cas échéant les 32768 octets des 32K supplémentaires pour transposer aisément toutes les démonstrations à la version 48K.

PEEK, POKE ET LE TRAVAIL DIRECT AVEC LES OCTETS DE MEMOIRE

Bien connaître les lignes générales de l'organisation mémoire du ZX-Spectrum est donc utile à un double point de vue, dans la perspective d'une programmation avancée. Elle permet, premièrement, de modifier certains octets de mémoire dont on connaît l'emplacement, pour arriver à un résultat qui serait beaucoup plus long à obtenir par le Basic. Elle permet ensuite, lorsque certaines techniques sont maîtrisées, d'intervenir directement sur la répartition des différentes zones de la mémoire.

Pour pouvoir lire ou modifier tel ou tel octet de mémoire, il est nécessaire de disposer de fonctions spécialisées pour cela : ce sont les commandes **PEEK** et **POKE** qui n'ont été que parcimonieusement utilisées jusqu'à présent, mais qui seront désormais systématiquement mises en oeuvre.

PEEK est la commande qui permet de lire le contenu d'un octet de mémoire, **POKE** est la commande qui permet de modifier ce contenu. Le contenu de l'octet à modifier est une valeur décimale comprise entre 0 et 255. Ainsi, lorsqu'on effectue, au moment de la mise sous tension normale, le petit programme de la *figure V-1*, on obtient le contenu des différents octets servant à coder le caractère A. En effet, l'octet 32000 marque le début de la zone réservée aux graphiques utilisateur qui est normalement chargée avec le caractère "A" quand l'utilisateur ne l'a pas modifiée. Si on avait le programme (*figure V-2*), on arriverait au même résultat. En effet, on a demandé à l'ordinateur de lire, dans la ROM, la partie de la table générale des caractères qui y est contenue correspondant au caractère "A".

Faisons maintenant **POKE 32600, 255** ; puis commandons au clavier l'impression du caractère graphique de la touche "A", et nous verrons apparaître un A surmonté d'une barre : l'ensemble des bits qui composent le premier octet ont été mis à "1".

```
10 FOR I=32600 TO 32607
20 PRINT PEEK I;" ";
30 NEXT I
```

```
0 60 66 66 126 66 66 0
```

Figure V-1

```
10 FOR I=15880 TO 15887
20 PRINT PEEK I;" ";
30 NEXT I
```

```
0 60 66 66 126 66 66 0
```

Figure V-2

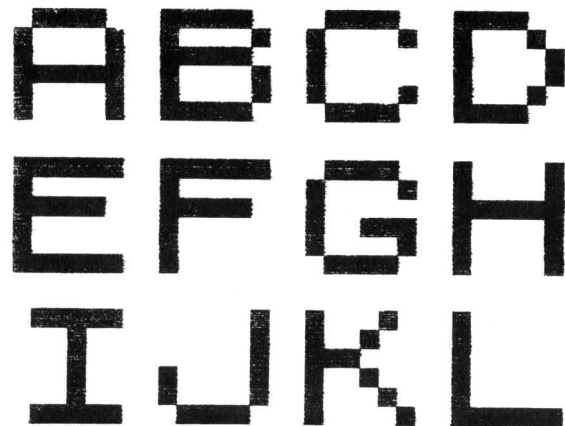
Une question se pose : puisque l'on ne peut **PEEK**er ou **POKE**er que des valeurs comprises entre 0 et 255, comment peut-on représenter un nombre beaucoup plus important, par exemple une des adresses de l'ordinateur de 0 à 64000 ? Il est alors nécessaire de coder cette valeur d'adresse sur deux octets, le second octet ayant un poids 256 fois supérieur au premier.

Exemple : désire-t-on savoir à quel endroit commence la zone affectée aux variables dans un programme, il faut alors imprimer :

```
PRINT PEEK 23627 + 256 * PEEK 23628
```

La commande **PEEK**, étant destinée à la lecture d'un octet de mémoire, ne présente aucun danger pour le déroulement d'un programme, même si elle fournit un renseignement erroné ou inutile. On peut par exemple, grâce à elle, construire à peu de frais un simulateur de nombres aléatoires en lisant, les uns après les autres, les octets de la ROM à partir de n'importe quel endroit.

La commande **POKE**, en revanche, doit être utilisée avec davantage de précautions, dans la mesure où elle modifie le contenu de la mémoire ; outre qu'elle est inopérante sur l'ensemble des octets de mémoire morte, si elle modifie abusivement l'une des variables dont le système se sert pour sa gestion, le fonctionnement de l'ordinateur sera perturbé d'une manière qui n'est pas toujours contrôlable.



```

10 LET D=15880
20 FOR L=0 TO 16 STEP 8: FOR C
=0 TO 3
30 FOR I=0 TO 6
40 LET K=PEEK (D+I)
45 FOR J=1 TO 8
50 LET K=INT K
60 LET K=K/2
70 IF (K-INT K)=0 THEN GO TO 8
2
75 PRINT AT L-1+I,C*8+8-J;"■"
80 NEXT J
90 NEXT I
100 LET D=D+8
110 NEXT C
120 NEXT L

```

Figure V-3

Ajoutons, pour terminer, que ces deux commandes sont très puissantes et acceptent de lire des expressions. Il faut, dans ce cas, les mettre entre parenthèses. Le petit programme de la *figure V-3* se sert ainsi d'une expression simple pour indiquer les commandes **PEEK**, afin d'obtenir l'impression sur toute la largeur de l'écran des 12 premiers caractères de l'alphabet. Voici son fonctionnement : l'ordinateur va chercher, dans la partie de la ROM qui comprend la table des caractères, les octets représentatifs de chacune des huit lignes qui composent chaque caractère et lit cet octet en effectuant autant de divisions par deux qu'il y a de bits dans cet octet, c'est-à-dire huit. Lorsque le reste de la division est égal à .5, c'est-à-dire que le bit est égal à 1, l'ordinateur commande sur l'écran, à une position appropriée, l'impression d'un pavé noir. On dispose ainsi, agrandi 64 fois (8 x 8), du codage des caractères de A à N.

```

!"#$%&'()*+,-./0
123456789:;<=>?@
ABCDEFGHIJKLMNPO
QRSTUVWXYZ[\]^_`
abcdefghijklmnopqrstuvwxyz{|}~©

```

Figure V-4

EXERCICE 1 : En effectuant une lecture de la table des caractères dans la mémoire morte par l'utilisation de la commande **PEEK** et en utilisant non plus la commande **PRINT**, mais la commande **PLOT**, effectuer l'affichage simultané de l'ensemble des caractères disponibles sur le ZX-Spectrum agrandi de 4 fois (2x2), de manière à obtenir l'affichage suivant : *figure V-4*.

LES GRANDES LIGNES D'ORGANISATION DE LA MEMOIRE VIVE

Dans la version de base du ZX-Spectrum, la ROM occupe les adresses 0 à 16383. Il en résulte que la première adresse de mémoire vive accessible est 16384. Le tableau reproduit à la *figure V-5*, montre de quelle manière cette mémoire accessible est répartie en diverses zones, de l'adresse 16384 à l'adresse 32768.

Ce sont les variables-systèmes, (VS), qui sont situées dans la zone comprise entre les octets 23552 et 23733, qui déterminent la position respective de chaque grande zone de travail. L'ensemble de ces variables-système est reproduit dans le tableau du chapitre 25 du manuel de base. Malheureusement, ce tableau exhaustif présente l'inconvénient de ne pas renseigner sur la hiérarchie des différentes variables les plus intéressantes à connaître pour l'utilisateur. Aussi, dans l'exposé qui suit, nous attarderons-nous seulement sur les principales d'entre elles.

Une première remarque s'impose : contrairement à ce qui se passait sur le ZX-81, la zone réservée au fichier d'affichage est stable dans la mémoire. Elle est même placée, contrairement au ZX-81, avant même la zone réservée aux variables-système. Ceux qui possèdent un ZX-81 se souviennent que la gestion de l'affichage de cet ordinateur se fait d'une façon très peu conventionnelle, puisque celui-ci se déplace constamment dans

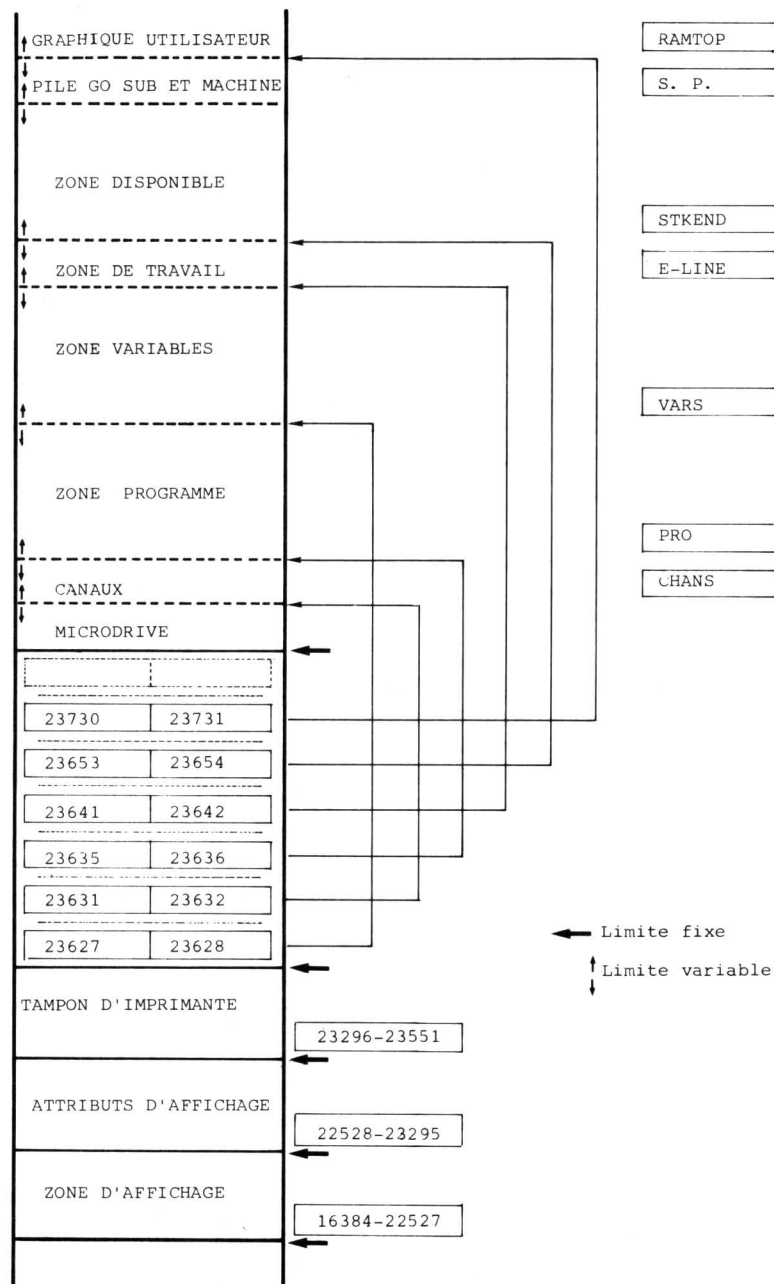


Figure V-5

la mémoire et n'a pas la même structure, selon que le module de mémoire supplémentaire a été enfiché ou non.

On notera que sur le ZX-Spectrum les trois premières zones mémoire, le fichier d'affichage, l'emplacement des attributs de chaque pavé d'affichage et le tampon d'imprimante occupent à eux seuls près de la moitié de la mémoire disponible, soit près de 7500 octets.

Tous les octets qui composent cette partie de la RAM peuvent être POKés, mais il est impossible d'augmenter ou de réduire le nombre des octets considérés. En effet, la gestion des couleurs, sur le Spectrum, relève d'une méthode originale. Contrairement à d'autres ordinateurs où il est possible d'augmenter la mémoire disponible pour des variables en diminuant, par exemple, la définition de l'écran, on ne peut, par logiciel, réserver de manière durable une partie de mémoire d'affichage de façon, par exemple, à y sauvegarder des valeurs ou des données. En effet, il suffira d'un **CLS** ou d'une réédition de liste pour effacer l'ensemble de ces données.

Une précision s'impose en ce qui concerne les variables-système. Il ne s'agit pas de variables-utilisateur. Les variables-utilisateur sont placées dans une zone de la mémoire qui leur est réservée et l'on peut les appeler par des instructions du type **PRINT A**, par exemple. On ne peut pas rappeler ainsi la variable-système qui désigne l'emplacement du début de cette zone des variables, à savoir la variable système **VARS**, avec une instruction du type **PRINT VARS**.

Les zones de la mémoire qui sont intéressantes à étudier et travailler sont l'*affichage*, la zone où sont situés les paramètres de chaque octet affiché, paramètres appelés *attributs*, la zone du *tampon d'imprimante*, la zone du *programme Basic*, la zone des *variables* et la *pile*.

La zone d'affichage

Le fichier d'affichage comprend 24 lignes de 32 caractères, avec des restrictions pour les deux dernières lignes qui servent, en principe, à l'affichage de comptes rendus de fonctionnement. Les octets qui constituent ce fichier d'affichage vont de 16384 à 22527. Ils sont donc au nombre de 6144, 8 octets par pavé, puisqu'il y a $24 \times 32 = 770$ pavés. Le pavé est composé de $8 \times 8 = 64$ pixels (*picture elements*), l'octet d'affichage représente donc une ligne de "picture elements".

Pour apprendre à bien connaître l'affichage, un exercice instructif consiste à remplir l'écran avec des caractères quelconques, sauvegarder l'écran et le recharger. On verra, cette opération ayant d'ailleurs duré un certain temps car le fichier d'affichage est très important, que la reconstitution de l'écran ne se fait pas du tout d'un seul tenant, en commençant par le haut, le bas, la droite ou la gauche. Les octets ne se suivent donc pas régulièrement dans la mémoire.

La structure générale du fichier d'affichage est, en effet, la suivante : il est fondamentalement divisé en trois parties de 8 lignes, les lignes 0 à 7, les lignes 8 à 15 et les lignes 16 à 24. Dans chacune de ces trois sous-zones, le balayage s'effectue, non pas ligne après ligne, mais par huitième de ligne, autrement dit : au 6ème balayage, ce ne sont pas les 6 premières lignes qui sont entièrement reconstituées, mais chacune des huit lignes est reconstituée aux trois quarts.

Ainsi, pour tracer une ligne horizontale tout au sommet de l'écran, il suffit de charger les 32 premiers octets de l'affichage, de 16384 à 16384 + 31, avec la valeur 255, ce qui s'effectuera à l'aide du petit programme : (figure V-6). Et, de la même manière, pour tracer une ligne verticale tout à gauche de l'écran, il suffira d'entrer : figure V-7. L'ordinateur doit sauter 32 caractères à chaque fois.

```
10 FOR I=16384 TO 16384+31
20 POKE I,255
30 NEXT I
```

Figure V-6

```
2 10 FOR I=16384 TO 22400 STEP 3
20 POKE I,1
30 NEXT I
```

Figure V-7

EXERCICE 2 : Ecrire un programme qui couvre la totalité de l'écran avec la lettre A.

En conclusion, le fichier d'affichage est composé de 6144 octets et, pour un caractère déterminé de l'écran, chacun des 8 octets qui se suivent pour composer la matrice de ce caractère est séparé du suivant par 255 octets.

La zone des attributs

Les attributs, ainsi que nous l'avons déjà vu, servent à définir de quelle manière chaque position de caractère sera imprimée, c'est-à-dire quel sera son fond (**PAPER**), la couleur d'impression (**INK**), la brillance (**BRIGHT**) et le clignotement (**FLASH**).

La zone de RAM occupée par ces attributs est contenue entre les octets 22528 et 23295, avec un octet par position de caractère, c'est-à-dire $24 \times 32 = 768$ positions. On remarquera, et c'est une des grandes caractéristiques du ZX-Spectrum, que la couleur, en particulier, ne peut donc être attribuée d'une manière différenciée aux différents points qui composent une matrice de caractères. Pour chaque position de caractère, c'est une seule et même combinaison d'attributs qui s'impose parmi 256 possibilités.

Le petit programme de la figure V-8 explique bien la définition des attributs pour chaque case. Dans chaque position d'attribut est POKEée une valeur choisie aléatoirement. Le résultat, sur l'écran, est une débauche de couleurs, avec des différences de brillance et de flash.

```
10 FOR I=22528 TO 23295
20 LET A=RND*256
30 POKE I,INT A
40 NEXT I
```

Figure V-8

Le système de gestion des couleurs par le ZX-Spectrum présente pas mal d'inconvénients, puisque la couleur ne peut être définie point par point. Néanmoins, ce système offre un avantage de rapidité incontestable lorsque l'on cherche certains effets colorés très rapides. Le programme de la figure V-9, qui peut être aménagé à volonté par l'utilisateur, exploite ces possibilités. Il forme sur l'écran un carré divisé en 8 régions où les points se reflètent par symétrie, pour former de beaux motifs qui évoluent constamment. La ligne 10 définit une chaîne où sont inscrites les principales couleurs, dans un ordre qui rappelle le plus possible celui de l'arc-en-ciel. Mais il est possible de disposer les couleurs d'une façon tout à fait différente, en particulier de modifier la ligne 70 pour y introduire une fonction aléatoire de la forme **RND * 127** ou *** 255**. Les motifs s'animeront alors, car l'effet de flash sera introduit dans la variable aléatoire.

ligne 230, l'intervention de la valeur 1792 qui représente $6144 / 3 - 256$.

EXERCICE 3 : *Proposons maintenant au lecteur d'organiser un défilement vers le bas en modifiant, d'une façon judicieuse, le signe des octets à charger et la variable d'initialisation D. Pour éviter les problèmes de fin d'écran, on se contentera des deux premiers tiers de l'écran, soit 16 lignes.*

Le tampon d'imprimante

Les 256 octets qui suivent, dans la RAM, la zone où sont contenus les attributs, ont pour fonction de stocker les lignes incomplètes qui doivent être imprimées sur l'imprimante. Ce tampon d'imprimante est nécessaire, car chaque fois qu'une commande d'impression contient des virgules ou des points-virgules, ce qui signifie que les résultats suivants devront être imprimés sur la même ligne, la ligne incomplète ne peut être imprimée immédiatement, l'imprimante ne pouvant imprimer qu'une ligne complète à la fois, à cause du mécanisme d'entraînement du papier.

On notera que le tampon d'imprimante est huit fois plus important sur le ZX-Spectrum qu'il l'était sur le ZX-81, puisqu'il est tenu compte de la définition graphique point par point, c'est-à-dire que, pour chaque caractère, il y a huit octets. On s'explique, de cette manière, pourquoi l'imprimante ne peut transmettre de renseignements concernant la couleur. Aussi ne peut-elle imprimer, en mode normal, que ce qui est relatif à la définition du caractère, c'est-à-dire noir sur blanc.

Le tampon d'imprimante est un emplacement qui présente, pour le programmeur, un très gros intérêt dont il n'est pas suffisamment fait mention dans le manuel de base du ZX-Spectrum. Premièrement, cet emplacement de 256 octets est une région bien protégée dans laquelle il est, par conséquent, possible d'implanter une routine en langage machine. Certes, cette utilisation est relativement incompatible avec l'utilisation de l'imprimante elle-même, mais bien des programmations, en particulier des programmations de jeux, n'ont pas besoin tellement d'imprimante.

Le deuxième intérêt, à notre avis, celui-ci très souvent ignoré, est que la connaissance des octets du tampon d'imprimante permet de porter efficacement remède à l'une des infériorités de l'impression, par rapport à

```

5 LET A=1
6 LET S=0
10 FOR I=0 TO 255
20 IF S=32 THEN LET A=-A: LET
S=0
30 POKE (23297+I), 136*(A>0)+34
*(A<0)
35 LET S=S+1
40 NEXT I
50 LPRINT
60 GO TO 10

```

Figure V-11

l'écran. L'imprimante ne peut en effet pas rendre compte des couleurs. En chargeant judicieusement les octets du tampon d'imprimante, il est possible de faire correspondre, sur le papier, à chaque couleur de l'écran, un grisé de densité équivalente, soutenu pour les couleurs foncées, et léger pour les couleurs claires. Pour ce faire, il suffit de tester quel est l'attribut de la case d'écran que l'on souhaite reproduire et de faire correspondre, à la couleur de l'encre, un grisé défini par certaines valeurs comprises entre 1 et 255. Pour illustrer cette technique, nous proposons un programme (figure V-11) qui a pour effet de charger séquentiellement les 256 octets du tampon d'imprimante avec des valeurs alternées. Grâce à ce programme, soit en faisant jouer le pas d'incrément des octets, soit en modifiant les valeurs à POKER, on a pu obtenir plusieurs variétés de grisé qui, une fois juxtaposées, permettent une belle différenciation des sorties d'écran lorsque l'on souhaite en conserver la trace sur imprimante (figure V-12).

La zone programme

Contrairement aux zones précédentes qui ont une adresse de début fixe, le début de la zone programme est conditionné par l'addition de périphériques, essentiellement des microdisquettes. En l'absence de micro-disquette, le programme débute à l'octet 23755. Sinon, il est possible d'interroger la variable-système située aux pas 23635-23636 pour connaître le début du programme.

Pour pouvoir travailler efficacement sur les octets de la zone programme, il faut avoir auparavant bien assimilé la manière dont les lignes de programme sont codées dans le Sinclair. Le codage de ces lignes, qui est indiqué au chapitre 24 du manuel de base, n'est pas fait d'une manière suffisamment concrète, et il est beaucoup plus simple, pour se rendre compte de la


```

9000>REM RENUMEROTAGE
9001 REM -----
9005 INPUT "ligne de depart";D
9010 INPUT "nouveau pas?";P
9015 LET Y=23755
9020 POKE Y,INT (D/256)
9030 POKE (Y+1),INT ((D/256-INT
(D/256))*256)
9040 LET D=D+P
9050 LET Y=Y+PEEK (Y+2)+256*PEEK
(Y+3)+4
9060 IF PEEK (Y)=35 AND PEEK (Y+
1)=40 THEN GO TO 9080
9070 GO TO 9020
9080 LIST

```

Figure V-15

Nous allons développer ci-dessous trois exemples qui mettent à profit ces différents renseignements de numéro de ligne, de longueur de ligne, et nous donnerons également quelques précisions sur la façon dont les nombres sont codés dans le système. Ces trois exemples illustrent chacun une manière simple d'intervenir sur les octets de la zone programme.

Le renumérotage des octets de la zone programme

Le programme de la figure V-15, intitulé "renumérotage", peut être placé à la suite d'un programme en cours d'élaboration pour modifier la numérotation entre plusieurs lignes, de façon à insérer de nouvelles lignes ou, au contraire, ramasser une routine qui occupe trop de place pour les numéros de ligne. Cette routine renumérote l'ensemble du programme à partir du début, jusqu'à ce qu'elle rencontre la ligne 9000 ; elle s'arrête alors pour ne pas perturber l'exécution en cours. Il est possible, grâce à cette routine, d'obtenir ainsi de belles lignes de programme espacées de 10 en 10 ou de 100 en 100.

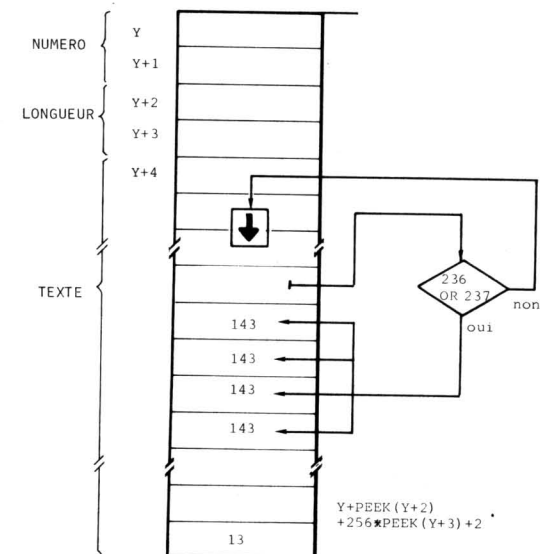
Ce programme de renumérotage est souvent proposé en langage machine, mais nous ne voyons pas, a priori, l'intérêt de cette solution, car il s'agit d'une procédure que l'on fait intervenir peu souvent dans le cours de l'élaboration d'un programme, et l'on accepte, dans ce cas, de patienter les dix secondes nécessaires à une exécution par le Basic.

Le programme se positionne sur l'octet de début de zone programme (ligne 9015) et décompose le numéro de la ligne de départ qui a été demandé en deux octets qui sont POKés dans les deux premiers octets de la zone

programme. Puis le pas se trouve ajouté à ce numéro par la ligne 9040. La ligne 9050 utilise les deux octets qui indiquent la longueur du numéro de ligne pour sauter tout de suite à l'octet occupé par le début de la ligne suivante. La ligne 9060 teste s'il s'agit là du numéro de ligne qui doit commander l'arrêt de l'exécution (9000) ; si ce n'est pas le cas, la procédure de stockage des deux premiers octets de la ligne suivante est de nouveau appelée : c'est la ligne 9070. En fin d'exécution, le listage de l'ensemble du programme est demandé : ligne 9080.

EXERCICE 4 : *Proposons au lecteur une petite complication qui rendra le programme plus souple. La procédure de renumérotage ci-dessus doit être réécrite partiellement pour affecter une limite supérieure et une limite inférieure au renumérotage. Ceci est intéressant pour se donner de la marge sur une partie seulement d'un programme.*

Cette procédure de renumérotage, très utile, présente cependant un inconvénient : les numéros de transfert qui accompagnent les instructions de branchement, comme GOSUB et GOTO, ne sont pas modifiés. Le petit programme (figure V-17) est une addition utile au programme précédent. Sa fonction est d'imprimer de gros pavés noirs à la suite des GOTO et des GOSUB du programme renuméroté, pour attirer l'attention de l'utilisateur sur les



```

0000>LET Y=23755
0010 FOR I=Y+4 TO Y+PEEK (Y+2)+2
0020 *PEEK (Y+3)+2
0030 IF PEEK (I)=236 OR PEEK (I)
0037 THEN GO SUB 9250
0038 LET Y=Y+PEEK (Y+2)+256*PEEK
(Y+3)+4
0040 IF PEEK Y=35 AND PEEK (Y+1)
=40 THEN LIST
0045 GO TO 9210
0050 POKE (I+1),143: POKE (I+2),
143: PRINT (I+3),143: POKE (I+4)
143
0060 RETURN

```

Figure V-17

```

0000>REM CHANGEMENT DE VARIABLE
0001 REM -----
0004 INPUT "LETTRE A CHANGER ?";
B$
0006 INPUT "POUR QUELLE LETTRE?"
C$
0008
0008 LET Y=23755
0010 FOR I=Y+4 TO Y+PEEK (Y+2)+2
0020 *PEEK (Y+3)+2
0030 IF PEEK (I)=CODE A$ THEN GO
SUB 9250
0038 NEXT I
0038 LET Y=Y+PEEK (Y+2)+256*PEEK
(Y+3)+4
0040 IF PEEK Y=35 AND PEEK (Y+1)
=40 THEN LIST
0045 GO TO 9210
0050 POKE I,CODE B$
0050 RETURN

```

Figure V-18

modifications à opérer dans les branchements pour introduire les nouveaux numéros de ligne (*figure V-16*). Cette petite addition présente un intérêt supplémentaire, celui de nous introduire à la manière dont le ZX code les nombres dans la mémoire programme (à ne pas confondre avec le codage dans la zone variable, c'est-à-dire le codage sur six octets, un octet de nom de variable, un octet d'exposant et quatre octets de mantisse).

L'utilisation de PEEK et POKE pour modifier automatiquement le nom des variables

Dès qu'un programme devient un peu complexe, il est intéressant de pouvoir modifier le nom des variables pour grouper celle-ci (A et B en X,Y, etc.).

La structure du programme qui effectue ce traitement (changement de variable - *figure V-18*) est fondamentalement la même que celle du programme précédent pour la

recherche. Dès que le code de la lettre a été rencontré, il lui est substitué par voie de routine (9250) le code de la nouvelle lettre. Il s'agit même d'une application relativement spectaculaire puisqu'on a un peu l'impression que le programme s'est réécrit tout seul.

LES GRANDES LIGNES DE LA ROM DU SPECTRUM

La ROM du ZX-Spectrum est située dans les 16 premiers K-octets de la mémoire. Elle est constituée essentiellement par un système d'exploitation, un interpréteur Basic et une table de 96 caractères. Le programme moniteur est, en fait, un énorme programme en langage machine et on pourrait penser qu'il est prématuré de s'y intéresser tant qu'une initiation systématique au langage machine n'a pas été menée. C'est une erreur : la ROM présente plusieurs intérêts pour le débutant en langage machine.

D'abord, lorsque l'on connaît l'emplacement de certaines routines très utiles, il suffit d'apprendre la technique simple qui consiste à les appeler pour disposer de vastes pans de programmes en langage machine, très efficacement rédigés. Ensuite, comme c'est essentiellement la ROM d'un ordinateur qui fait son originalité, il est exclu de prétendre bien connaître son Spectrum si l'on ne sait pas comment le programme moniteur est structuré dans ses grandes lignes.

Le système d'exploitation du Spectrum

Cette partie se compose d'abord des **routines d'initialisation** qui occupent les pas de 0 à 7 et 4555 à 4769. Cette routine a pour principal effet de voir la quantité de mémoire disponible pour l'exécution des opérations suivantes. Les routines suivantes consistent en **routines d'édition**, c'est-à-dire tout ce qui est fonction d'édition et vérification de syntaxe. Lorsqu'une ligne de programme a été incorrectement entrée, on sait qu'on aboutit à un compte rendu d'erreur. C'est cette partie de la ROM qui assure cette fonction ; elle va de 3884 à 5550 et comprend donc une partie de la routine d'initialisation précédente.

Une sous-routine très intéressante de cet ensemble est constituée par la **sous-routine de décodage du clavier** qui lit quelle est la dernière touche qui a été pressée ; elle va de 4264 à 4380.

Les **routines d'impression**, qui font également partie du système d'exploitation, très utiles pour l'utilisateur - il en sera fait usage plus loin dans notre programme de démonstration -, sont contenues entre les pas 2548 et 3404.

L'interpréteur Basic

Cette partie essentielle de la ROM a pour fonction de traduire les lignes Basic en autant de routines en langage machine compréhensible par le microprocesseur. Les parties les plus importantes de l'interpréteur Basic sont les **routines de contrôle** qui font passer l'interpréteur Basic d'une ligne de programme à la suivante : elles sont situées entre les pas 6935 et 7168. L'analyse des paramètres qui accompagnent les fonctions et commandes du Basic est confiée à une routine située entre les pas 7176 et 7389.

Suit, jusqu'au pas 9466, la masse des routines correspondant aux **commandes**. A chaque ordre Basic (50) correspond ainsi un sous-programme en langage machine qui en est la traduction. Ce sous-programme fonctionne en relation avec la **table des commandes** qui est située entre les pas 6728 et 6934.

Les pas 9467 à 10955 sont occupés par **l'évaluateur d'expression** et les routines de manipulation des variables.

Suivent les **routines de calcul**, placées entre 12187 et 14445. C'est dans cette partie que sont effectuées les opérations arithmétiques, les calculs de fonctions trigonométriques, etc.

La troisième grande partie du moniteur est constituée par la **table des caractères** qui commence à 15616.

L'utilisation des routines du programme moniteur

Le lecteur qui serait intéressé par le détail de la ROM du ZX-Spectrum est invité à se reporter au tome II de cet ouvrage qui approfondit cette question. Dans le cadre de ce paragraphe, nous allons utiliser un exemple simple pour montrer combien certaines routines de la ROM, très efficaces puisqu'elles appartiennent au système, peuvent être récupérées avec facilité (*figure V-19*).

```

5 REM APPEL DE ROUTINE MEM
6 REM -----
10 FOR I=1 TO 20
20 PRINT "*****"
*****
30 NEXT I
40 INPUT "COMBIEN DE LIGNES ?"
LIGNES
50 POKE 32601,LIGNES
60 LET K=USR 32600: STOP
100 DATA 6,10,205,68,14,201
110 FOR I=0 TO 5
120 READ A: POKE (32600+I),A
130 NEXT I

```

48K :
65368

Figure V-19

Au pas 3652 de la ROM commence la routine qui efface un certain nombre de lignes, à compter du bas de l'écran. Le sous-programme ci-dessus contient, à la ligne 100, une très courte routine de 6 valeurs décimales qui seront POKées dans la partie de la mémoire réservée aux graphiques utilisateur, soit 32600 et suivants. Concentrons-nous sur les codes 3, 4 et 5.

Le code 205 est le code Z 80 d'appel de routine. Il est suivi d'une adresse codée sur deux octets inversés, qui est précisément l'adresse de la routine d'effacement de ligne. Ces trois octets sont précédés par le code de chargement d'un nombre dans un registre, suivi du nombre à charger. Enfin, la série des codes est terminée par le code 201 qui veut dire : retour au Basic.

La mise en oeuvre de cette très brève routine est effectuée par les lignes Basic de début de programme dont les trois premières se chargent de remplir l'écran avec des astérisques. Il est ensuite demandé combien de lignes on souhaite effacer et le nombre que l'on rentre à ce moment-là est précisément POKé au pas 32601, c'est-à-dire dans le deuxième octet de la routine machine. L'effacement du bas de l'écran du nombre de lignes que l'on a commandé est absolument immédiat. On perçoit tout l'intérêt de ce genre de procédure pour un certain nombre de programmes où l'on souhaite le maintien de titres explicatifs en haut de l'écran, par exemple.

UN PROGRAMME DE VISUALISATION DE LA PLACE MEMOIRE DISPONIBLE

Ce chapitre sur l'organisation générale de la mémoire du ZX-Spectrum, sur l'utilisation appropriée des fonctions PEEK et POKE pour connaître ou modifier les

éléments de celle-ci, serait incomplet sans un programme mettant l'utilisateur en mesure de connaître instantanément et à tout moment, pendant le travail sur une programmation longue faisant intervenir beaucoup de variables et de calculs, quelles sont les grandes masses de mémoire occupées, ceci pour voir quelle est la place qui reste encore disponible, mais aussi pour faire éventuellement quelques transactions en toute connaissance de cause entre un programme plus long, mais comprenant moins de variables, ou un programme un peu plus court comprenant plus de variables. L'expérience apprenant vite que le choix se fait souvent sur cette base. Souvent, en effet, le codage d'une variable numérique en variable alphanumérique dans de grands tableaux permet un gain de place important au prix de quelques lignes de programme supplémentaires. En revanche, inversement, l'utilisation de quelques variables intermédiaires de travail permet parfois de raccourcir les pas de programme dont l'écriture est complexe et qui reviennent souvent.

Un tout premier programme (*figure V-20*), facile à rédiger comme à incorporer, donne déjà de bons renseignements et présente l'avantage d'être d'un accès permanent puisque, une fois testé l'emplacement des grandes zones de la mémoire, le résultat est transcrit à l'intérieur d'une REM sous une forme aisément visualisable. Si la routine 9000 est appelée après un programme placé entre 10 et 9000, les 23 caractères qui suivent la première ligne de remarques seront modifiés comme suit : ces 23 caractères représenteront proportionnellement l'ensemble de la mémoire comprise entre 32600, qui est la zone réservée aux graphiques-utilisateur, et 23760, qui est le début de la zone programme. La zone proportionnellement occupée par le programme sera remplie de P, celle occupée par les variables sera remplie de V, celle occupée par la zone de travail de l'ordinateur sera remplie de T, et la zone entièrement disponible sera figurée par des blancs. La zone de travail de l'ordinateur va du début de la zone d'édition à la fin de la pile du calculateur. La représentation n'est pas très fine puisque chaque caractère correspond à une zone de 385 octets, ce qui a pour résultat, en particulier, de faire rarement apparaître la zone de travail qui ne comprend qu'assez peu d'octets. Les limites, successivement testées, sont déterminées par le contenu des variables-système VARS et RAM TOP.

Un deuxième programme (*figure V-21*), plus long à l'exécution, utilise l'ensemble de l'écran et la couleur pour proposer une représentation plus fine de la

```

1 REM MARQUEUR DE PLACE MEMO.
2 REM -----
0000 FOR I=23760 TO 32599 STEP 1
0010 IF I<(PEEK 23627+256*PEEK 2
0020 THEN LET A=80: GO TO 9050
0030 IF I<(PEEK 23641+256*PEEK 2
0040 THEN LET A=86: GO TO 9050
0050 IF I<(PEEK 23653+256*PEEK 2
0060 THEN LET A=84: GO TO 9050
0070 LET A=32
0080 POKE (23760+(I-23760)/385),
0090 A
0100 NEXT I: LIST

```

Figure V-20

```

1 REM MARQUEUR DE PLACE MEMO.
2 REM -----
0000 FOR I=23760 TO 32599 STEP 1
0010 IF I<(PEEK 23627+256*PEEK 2
0020 THEN LET A=80: LET P=1: GO
0030 TO 9050
0040 IF I<(PEEK 23641+256*PEEK 2
0050 THEN LET A=86: LET P=5: GO
0060 TO 9050
0070 IF I<(PEEK 23653+256*PEEK 2
0080 THEN LET A=84: LET P=2: GO
0090 TO 9050
0100 LET A=32: LET P=4
0110 PRINT PAPER P; INK 9; CHR# A
0120 NEXT I

```

Figure V-21

répartition mémoire. Les 704 cases des 22 premières lignes sont modifiées comme suit : elles apparaissent en bleu foncé pour les octets occupés par le programme, en bleu clair pour ceux occupés par la zone variable, en rouge pour la zone de travail et en vert pour la zone encore disponible.

En fait, la zone entièrement disponible, qui commence à la fin de la pile du calculateur, ne va pas exactement jusqu'en haut de la mémoire. Il faut en soustraire un certain nombre d'octets qui représentent la pile du microprocesseur Z 80 dans son travail, ainsi que la pile des adresses de retour de transfert des instructions GOSUB (3 octets par transfert). Malheureusement, il n'existe pas, dans le jeu des variables-système du Spectrum, une variable constamment réactualisée pour indiquer quel est le début de la pile du Z 80. Il est nécessaire, pour cela, de passer par l'intermédiaire d'un petit programme machine que le lecteur introduira en nous faisant entièrement crédit pour l'instant, puisque c'est le chapitre suivant qui doit développer ces problèmes.


```

1 REM MARQUEUR DE PLACE MEMO.
2 REM -----
9000 FOR I=23760 TO 32599 STEP 1
9050 IF I<(PEEK 23627+256*PEEK 2
9060 THEN LET A=80: LET P=1: GO
9070 TO 9070
9080 IF I<(PEEK 23641+256*PEEK 2
9090 THEN LET A=86: LET P=5: GO
9100 TO 9070
9110 IF I<(PEEK 23653+256*PEEK 2
9120 THEN LET A=84: LET P=2: GO
9130 TO 9070
9140 LET K=USR 32602
9150 LET A=84: LET P=2
9160 IF I<(PEEK 32600+256*PEEK 3
9170 THEN LET A=32: LET P=4: GO
9180 TO 9070
9190 LET A=84: LET P=2
9200 PRINT PAPER P; INK 9;CHR$ A
9210 NEXT I
9220 DATA 237,115,88,127,201
9230 FOR I=0 TO 4: READ A: POKE
(32602+I),A: NEXT I

```

Figure V-22

Ce court programme (figure V-22), qui consiste à charger les cellules mémoire 32600 et 32601 avec la valeur du premier octet utilisé par la pile, est introduit par l'instruction **DATA 9500** et chargé par l'instruction 9510. Pour initialiser l'ensemble du programme marqueur, il faut donc faire, d'abord, **GOTO 9500**. La routine machine sera appelée par la ligne 9040 selon une technique que nous expliciterons plus tard, et la nouvelle limite calculée par la ligne 9050.

Après avoir introduit différents programmes d'essai et appelé la routine de marquage, on tirera rapidement l'enseignement que les octets occupés par la pile des instructions **GOSUB** et la pile Z 80 ne sont jamais très nombreux ; même dans les programmations les plus compliquées, ils dépassent rarement 100 octets. Il en résulte que cette routine, bien que plus précise que la précédente, n'apporte pas de renseignement complémentaire très utile ; en revanche, le lecteur qui se sera donné la peine de l'introduire et qui aura compris, à la lecture du chapitre suivant, les grandes lignes du travail en langage machine, n'en connaîtra que mieux le fonctionnement du système Spectrum.

EXERCICE 5 : Chaque fois qu'un sous-programme a été appelé et que le branchement de retour a été effectué, la pile des instructions **GOSUB** n'a plus besoin de conserver l'adresse de retour : aussi, est-elle écrasée. L'exercice

proposé au lecteur est le suivant : pour pallier cette difficulté, emboîter un certain nombre de sous-programmes entre eux pour augmenter sensiblement la longueur de la pile des instructions **GOSUB** et placer la routine de marquage au dernier niveau de cet emboîtement. La succession des adresses de retour ne sera pas alors écrasée et le résultat obtenu à l'écran sera, cette fois-ci, conforme à la réalité.

CHAPITRE VI

INITIATION AU LANGAGE MACHINE

BASIC ET LANGAGE MACHINE

Le Basic est un langage informatique évolué, c'est-à-dire qu'il utilise des mots (les fonctions et commandes) qui sont directement compréhensibles par l'utilisateur (tout au moins l'utilisateur anglo-saxon), car ils décrivent exactement ce qu'ils font avec des termes empruntés au langage courant. De même, sa syntaxe a été conçue pour se rapprocher le plus possible des modes de raisonnement humain : déclaration de variables, transferts, itérations, etc. Il utilise, enfin, la numération qui nous est la plus naturelle, c'est-à-dire la numération décimale.

Comme le microprocesseur qui se trouve au centre du système ne comprend que l'information binaire, c'est-à-dire celle qui correspond à une succession de 0 ou de 1, il en résulte que, pour pouvoir effectuer les opérations qui seront commandées par le programme Basic, il est nécessaire que le programme moniteur (ROM) se charge de traduire les termes de Basic et de décomposer les opérations complexes en opérations binaires simples.

On peut dire qu'un ordinateur passe presque 99 % de son temps à préparer une opération et 1 % seulement à exécuter cette opération. Il est clair qu'en court-circuitant cette phase de préparation, on aboutit à un déroulement de l'exécution plusieurs dizaines de fois plus rapide et c'est là, la grande indication du langage machine, la recherche de la rapidité.

Le ZX-Spectrum est un appareil intrinsèquement près de quatre fois plus rapide que le ZX-81 qui était obligé d'interrompre périodiquement l'exécution de ses programmes pour rafraîchir son affichage (en fait, un quart du temps du déroulement était consacré à l'opération et trois quarts à l'affichage). On pourrait penser que, de ce fait, la recherche de la vitesse sur le Spectrum est moins cruciale. Mais comme il s'agit, par ailleurs, d'un appareil aux possibilités graphiques poussées et que la disposition de la couleur et du son le rendent particulièrement indiqué aux applications de jeux qui ne sont jamais suffisamment rapides, l'apprentissage du langage machine est aussi payant sur cet ordinateur que sur son prédécesseur et, sans doute, plus facile.

L'ETUDE DU LANGAGE MACHINE

L'étude du langage machine est réservée aux programmeurs qui ont acquis une certaine expérience en matière de programmation en Basic. En effet, le langage machine présente trois difficultés : il est, d'une part, plus abstrait et rigide que le Basic ; abstrait, car il repose sur un codage, et rigide, car la disposition des différents codes et leur emplacement précis sont aussi importants que les codes eux-mêmes ; ensuite, comme il est fait pour attaquer directement le microprocesseur qui n'effectue que les opérations fondamentales, l'objet de la programmation doit être décomposé d'une façon beaucoup plus poussée.

Enfin, troisième difficulté, tandis que les erreurs de syntaxe du Basic peuvent être signalées par le programme moniteur du Spectrum, les erreurs de syntaxe du programmeur en code machine peuvent provoquer l'effacement complet de la mémoire en bloquant complètement le système, ce qui aboutit parfois, lorsqu'on n'a pas pris la précaution de sauvegarder provisoirement à chaque étape, à la perte complète du travail effectué.

Heureusement, l'étude du travail en programmation machine peut être très progressive. Il est possible de commencer par l'écriture de routines très courtes, directement appelées par le programme Basic. D'ailleurs, les plus belles programmations sont celles qui savent faire une association performante entre le Basic pour l'organisation générale de la programmation et des routines en code machine pour l'accélération de telle ou telle opération particulière.

C'est cet objectif modeste que se propose le présent chapitre : initier le lecteur à la rédaction de routines en langage machine simples mais efficaces. Le lecteur auquel ce chapitre, s'il y parvient, aurait donné le désir d'aller plus loin, se reportera avec profit au tome II de ce présent ouvrage, entièrement consacré à la programmation machine sur le ZX-Spectrum, et à l'ouvrage plus général d'Alain Pinaud, dans la même collection, " *Programmer en assembleur* ", qui présente une approche théorique complète des possibilités du microprocesseur Z80, indépendamment de tout système particulier. Nous nous contenterons, ici, d'une brève présentation des caractéristiques de ce très puissant microprocesseur.

LA STRUCTURE DU MICROPROCESSEUR Z80A

Lorsqu'un programme est exécuté, chaque opération de ce programme ne correspond pas à la modification directe du contenu d'une mémoire par le microprocesseur. La procédure est différente : le contenu de l'octet à modifier est recopié à l'intérieur du microprocesseur, plus précisément stocké dans l'une de ses mémoires internes que l'on appelle un **registre** ; puis, ce registre est modifié selon les instructions qui ont été entrées par l'utilisateur dans la mémoire programme, par le biais du programme moniteur et, ensuite, le contenu du registre est recopié dans l'octet de mémoire considéré ou dans un autre.

Travailler en langage machine, qu'il s'agisse d'une programmation simple ou compliquée, c'est donc essentiellement connaître les principaux registres du microprocesseur Z80 et savoir comment ils se modifient sous l'action des différentes instructions qu'on lui envoie. Fort heureusement, pour le programmeur débutant, il est possible de réaliser de très beaux logiciels en langage machine en ne connaissant que le tiers de ces registres et le vingtième ou le dixième des instructions que comprend le microprocesseur. Avec un petit bagage de huit registres et d'une cinquantaine d'instructions, on entre déjà dans le domaine de la programmation hautement satisfaisante.

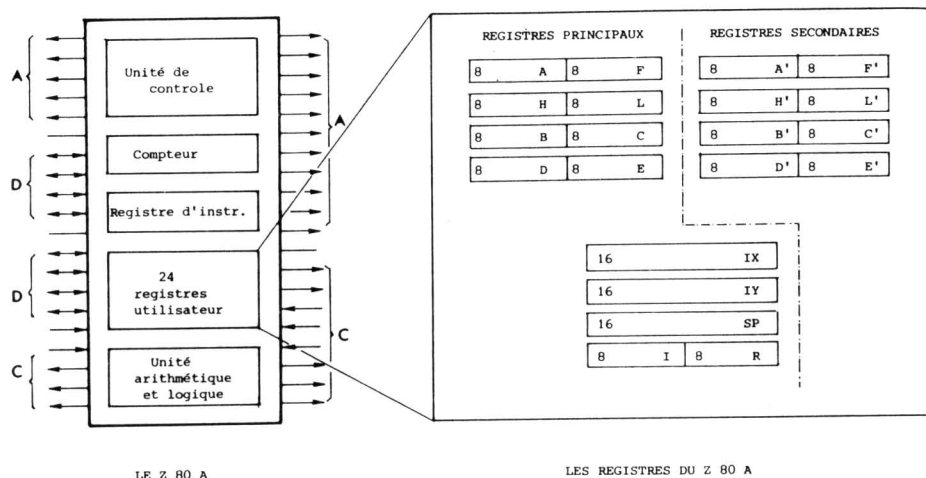


Figure VI-1

Les registres du Z80A

La figure VI-1 donne déjà une bonne idée de la structure interne du Z80. On y reconnaît ses cinq parties essentielles : une **unité de contrôle**, un **compteur de programme**, un **registre d'instructions**, une **unité arithmétique et logique** et **24 registres-utilisateur**. Le nom et la disposition de ces registres ne sont pas absolument rationnels, dans la mesure où le Z80A dérive d'autres microprocesseurs. Cependant, la présence d'un I dans les registres IX et IY signifie qu'ils peuvent être indexés ; A est l'initiale d'accumulateur, F, celle de Flag (drapeau). Enfin, H et L signifient High et Low, etc. Nous ne nous intéresserons pas, ici, aux registres secondaires dont l'utilité principale consiste à permettre des échanges avec les registres principaux.

En ce qui concerne les registres principaux, on doit distinguer deux familles : les registres simples qui peuvent être couplés en registre paire et d'autres registres IX, IY, SP, qui ne peuvent pas être désappariés

La règle la plus importante est la suivante : **les registres simples peuvent être utilisés simples ou appariés, mais seul un registre double peut être ajouté ou soustrait à un registre double**. Il est clair qu'un registre double ne peut être ajouté ou soustrait à un registre simple, car on ne peut pas faire entrer 16 bits dans 8.

Effectuons rapidement le passage en revue de chaque registre pour mettre en avant ses caractéristiques. Les

opérations simples peuvent être menées sur l'ensemble de ces registres, mais une certaine spécialisation intervient entre eux pour des opérations particulières.

Le **registre A**, accumulateur, est le plus utilisé de tous les registres ; il est à la base d'un nombre important d'instructions, instructions de comparaison notamment.

Les 8 bits qui composent le **registre F** sont autant de drapeaux ou indicateurs. Selon les résultats des opérations qui ont été menées sur d'autres registres, chacun des bits de ce registre est modifié d'une certaine manière. On distingue ainsi le bit de signe, le bit de 0, le bit de retenue, etc., et chacun de ces bits peut être testé pour en tirer les conséquences.

Les **registres B et C** peuvent être utilisés aussi comme compteurs dans un certain nombre d'instructions. Ainsi, certaines instructions décrémentent automatiquement le registre B, sans qu'il soit nécessaire de le préciser (DJNZ, etc.). C'est le contenu de ces deux registres couplés qui est retourné au Basic lorsqu'on fait **PRINT USR N**.

Les **registres D et E**, d'une part, **H et L**, d'autre part, sont des registres qui n'ont pas de spécialisation particulière, mais qui possèdent un appareil d'instructions très riche. Les registres H et L, en particulier, sont très utilisés pour contenir et charger des adresses mémoire, directement ou indirectement, etc. : ce sont les registres de rangement par excellence.

Les **registres IX et IY**, seulement utilisables comme registres paire, ont l'avantage de pouvoir être indexés : ils servent aux opérations itératives sur les tableaux.

Le **registre SP**, Stack Pointer, gère la pile, la pile étant un emplacement de la mémoire dans lequel il est possible de ranger provisoirement le contenu de tel ou tel registre, pour éviter qu'il soit oblitéré par d'autres opérations. On range dans la pile par l'instruction **PUSH** registre, et on rappelle par l'instruction **POKE** registre. Dans le programme que nous allons développer ultérieurement, il sera fait appel à ces instructions. La pile fonctionne selon le principe LIFO, *Last In First Out*, autrement dit "le dernier entré, le premier sorti" ; il faut aller rechercher dans la pile les registres dans l'ordre strictement inverse de celui dans lequel on les a empilés, faute de quoi on perturbe les données en provoquant des échanges involontaires.

Les différentes familles d'instructions du Z80A

Les codes du Z80A sont des nombres qui vont de 0 à 255. Les instructions qui consistent à charger, incrémenter ou échanger des registres, tiennent, la plupart, sur un seul octet, c'est-à-dire qu'elles n'ont qu'un seul code. Mais de nombreuses autres instructions travaillent sur des nombres ou sur des adresses de mémoire. Elles ont alors besoin de 1, 2 ou 3 octets supplémentaires, selon le cas, et sont donc composées de plusieurs codes.

La liste des instructions est reproduite à la fin du manuel de base du Sinclair, dans l'ordre numérique croissant, ce qui ne renseigne guère sur la façon dont elles sont groupées ; heureusement pour le programmeur, elles sont accompagnées d'une courte dénomination appelée "*mnémonique*".

Les codes, étant de simples valeurs numériques, ne sont pas très parlants, alors que les mnémoniques qui leur correspondent renseignent directement sur l'opération menée. **LD B,A** signifie charger le registre B avec l'accumulateur ; c'est, évidemment, plus évocateur que 71, représentation décimale du code de cette instruction. Lorsqu'on programme exclusivement avec l'aide de ces mnémoniques, on effectue une programmation en *langage d'assemblage*. Certains programmes, et il en existe d'excellents pour Spectrum, s'appellent "assembleurs" et leur fonction est précisément de traduire les mnémoniques afin que les codes correspondants puissent être chargés directement en mémoire à leur emplacement.

Ces logiciels évitent au programmeur d'aller chaque fois chercher le code de l'instruction qu'il programme et effectuent automatiquement le décompte des pas dans toutes les instructions de transfert, sans risque d'erreur. Ils sont, évidemment, complexes et exigent une grande quantité de mémoire.

Les mnémoniques renseignent mieux que les codes sur les différentes familles d'instructions.

La première famille est constituée par les codes de *chargement* : **LD** (LOAD) chargement de la mémoire dans un registre, chargement d'un registre dans la mémoire, chargement d'un registre dans un registre. Ainsi, la valeur 71 signifie charger l'accumulateur dans le registre B.

La deuxième famille comprend les *instructions arithmétiques* : additions, soustractions, incréments, décréments.

décrémentations. Ainsi, **INC D** signifie augmenter d'une unité le contenu du registre D.

La famille suivante est celle des *instructions logiques* : ce sont les instructions **AND** et **OR**, mais transposées au microprocesseur Z80, c'est-à-dire impliquant des comparaisons de registres bit à bit. A partir de cette famille, également l'instruction **XOR** ("ou" exclusif) et celle de la comparaison avec une valeur **CPN**, etc.

La quatrième famille est composée des instructions de *branchement*, branchement conditionnel, branchement inconditionnel : **JR**, saut relatif, **JP**, saut simple, **CALL**, **RESET** (semblables à **GOTO** et **GOSUB**), **RET** (semblable à **RETURN**), etc. Ces instructions commandent des ruptures de séquence.

Suivent d'autres familles d'instructions représentées chacune par un ensemble de codes moins nombreux que les familles précédentes : les instructions de *pile*, les instructions de *rotation*, les instructions de *bit* (pour ne travailler que sur un bit particulier), les instructions *entrée-sortie*, les instructions *d'interruption*, etc.

Les problèmes de conversion décimale-hexadécimale

Le lecteur que la programmation directe du Z80A intéresse va se trouver confronté à une difficulté lorsqu'il cherchera à s'inspirer de routines en langage machine publiées dans la presse spécialisée. En effet, ces routines sont, le plus souvent, reproduites sous la forme hexadécimale, c'est-à-dire fondées sur la numération en base 16, qui présente de nombreux avantages de transcription, car la base 16 est dérivée du binaire, ce que n'est pas la base décimale (16 est une puissance de 2, 10 n'est pas une puissance de 2).

Or, le ZX Spectrum lit les codes des routines machine qui y sont implantées sous *la forme décimale*. Il est donc nécessaire de posséder une certaine maîtrise des conversions mutuelles du décimal à l'hexadécimal et de l'hexadécimal au décimal.

Nous proposons ci-après une routine de chaque type qu'il est possible d'incorporer sous une forme ou sous une autre, dans tout programme de chargement ou de lecture de code machine.


```

10 REM CONVERSION DECI.=>HEXA.
20 REM -----
100 INPUT "Decimal?";D
110 IF D<256 THEN LET L=2
120 IF D>=256 THEN LET L=4
130 DIM H$(L): LET A=D
140 FOR I=L TO 1 STEP -1
150 LET P=D/16: LET D=(P-INT P)
160 IF P<=9 THEN LET H$(I)=STR$
170 IF P>=10 THEN LET H$(I)=CHR$
180 LET D=INT (D/16)
190 NEXT I
200 PRINT "Decimal=";A,"Hexadecimal=";H$
210 GO TO 100

```

Figure VI-2

```

10 REM CONVERSION HEXA.=>DECI.
20 REM -----
100 INPUT "Hexa=";H$
110 LET L=LEN H$
120 LET S=0
140 FOR I=L TO 1 STEP -1
150 IF CODE H$(I)=48 THEN GO TO
160 IF CODE H$(I)<=57 THEN LET
S=S+(CODE H$(I)-48)*16^(L-I)
170 IF CODE H$(I)>=65 THEN LET
S=S+(CODE H$(I)-55)*16^(L-I)
180 NEXT I
190 PRINT "Hexa=";H$,"Decimal="
;S
200 GO TO 100

```

Figure VI-3

Le principe de la **routine de conversion décimal-hexadécimal** (figure VI-2) est simple : le nombre décimal entré est successivement divisé par 16, 16 fois 16, 16 fois 16 fois 16, et le reste de la division chaque fois transformé de chiffre décimal en chiffre hexadécimal. Rappelons que les chiffres de la numération hexadécimale sont : 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.

La routine de **conversion hexadécimal-décimal** (figure VI-3) met en oeuvre le principe inverse : la chaîne qui représente le nombre hexadécimal est successivement découpée et le caractère isolé est transformé en sa valeur décimale, de 0 à 16, qui sera multipliée par 1, 16, 16 * 16 ou 16 * 16 * 16, selon l'emplacement du caractère hexadécimal. Pour plus de précisions concernant la numération hexadécimale, nous renvoyons le lecteur au tome II de cet ouvrage.

DEVELOPPEMENT D'UN PROGRAMME EN LANGAGE MACHINE

Nous allons, dans le cadre de cette section, illustrer, point par point, la construction d'une routine en langage machine, très utile. Cette routine d'application universelle consiste à remplir à toute vitesse un rectangle d'écran, une fois définies sa hauteur, sa largeur, la ligne et la colonne du coin supérieur gauche, sa couleur et sa brillance. C'est une routine courte et très efficace qui fait appel aux instructions les plus simples du Z80 : les instructions de chargement, les instructions de pile et les instructions de décrémentation-transfert. Elle démontrera au lecteur que la connaissance d'un faible nombre d'instructions du Z80 permet, pourtant, d'aboutir à des programmations extrêmement rapides. Cette rapidité sera démontrée, tant dans le programme principal que dans les exercices qui l'accompagnent avec le dessin instantané de quelques drapeaux bien connus, français, américain, belge, suisse et canadien.

Mais, afin que cette initiation soit le plus progressive possible, nous avons imaginé de proposer d'abord un petit programme de calcul simple qui ne fasse pas intervenir les problèmes d'affichage.

Premier programme : le calcul de la somme d'une série

Servons-nous du langage machine pour calculer immédiatement la somme d'une série de N nombres entiers jusqu'à 255 au maximum.

Pour se rendre mieux compte du gain considérable de rapidité, nous rédigerons d'abord un court programme en Basic effectuant précisément cette opération : une boucle incrémente la variable S représentative de la somme des N premiers nombres (figure VI-4).

```

1 REM ADDITION N PREMIERS NO.
2 REM -----
10 INPUT A
20 LET S=0
30 FOR I=1 TO A
40 LET S=S+I
50 NEXT I
60 PRINT S
70 STOP

```

Figure VI-4

```

1 REM ADDITION N PREMIERS NO.
2 REM -----
10 INPUT N
20 POKE 23681,N
30 LET A=PEEK (129+256*92)
40 LET B=A
50 LET HL=0+256*0
60 LET D=0
70 LET E=B
80 LET HL=HL+D+E
90 LET B=B-1
100 IF B<>0 THEN GO TO 70
110 LET B=INT (HL/256)
120 LET C=(HL/256-INT (HL/256))
*256
130 LET K=256*B+C
140 PRINT K
150 STOP

```

Figure VI-5

```

7F58 3A815C LD A,(23681)
7F5B 47 LD B,A
7F5C 210000 LD HL,0
7F5F 1600 LD D,0
7F61 BOUC:
7F61 58 LD E,B
7F62 19 ADD HL,DE
7F63 10FC DJNZ BOUC
7F65 44 LD B,H
7F66 4D LD C,L
7F67 C9 RET

```

Figure VI-6

Ensuite, nous allons simuler, dans un deuxième programme (*figure VI-5*), lui-même en Basic, le même calcul décomposé, en utilisant des variables qui portent les mêmes noms que les registres principaux du Z80, registre simple ou registre paire. Ainsi, si l'exécution de ce programme en Basic donne satisfaction, il suffira d'effectuer la transposition en langage machine. Cette transposition en langage machine, ou plus exactement en assembleur (*figure VI-6*), liste les adresses des différents codes, les codes et leurs mnémoniques. Le programme se lit de la manière suivante :

- charger le registre A avec le contenu de l'octet de mémoire 23681 (il s'agit d'un emplacement non utilisé par les variables système et donc, ce ce fait, accessible à l'utilisateur pour un stockage provisoire de valeur) ;
- charger le registre B avec la valeur A ;
- initialiser le registre paire HL avec la valeur 0 ;
- initialiser le registre D avec la valeur 0 ;
- puis ouvrir une boucle (on voit apparaître le symbole BOUC qui n'est pas lui-même un opérateur, mais qui est

là uniquement pour la lisibilité du programme (son apparition, dans la suite des mnémoniques, ne provoque pas l'incrémentation des emplacements mémoire) ;

- charger le registre E avec la valeur du registre B ;
- ajouter la valeur du registre paire DE à la valeur du registre paire HL ;
- apparaît une instruction complexe, au mécanisme très efficace, mais difficile à comprendre en première analyse : l'instruction DJNZ, qui signifie "décrémenter le registre B et, tant qu'il n'a pas atteint la valeur 0, revenir en arrière d'une certaine quantité pour retrouver le début de la boucle de référence" ; le nombre qui suit cette instruction est FC qui signifie 252, c'est-à-dire 256 - 4 ; l'explication est la suivante : lorsque le nombre qui suit le code de cette instruction est supérieur à 128, cela signifie que le branchement se fait en arrière, inversement, le branchement se fait vers l'avant quand le nombre est inférieur à 128 ; il s'agit d'un branchement relatif ; le nombre de pas du saut est décompté à partir du pas qui suit immédiatement l'instruction ;

- charger le registre D avec H ;
- charger le registre L avec C ;
- retour au Basic.

Il a été nécessaire de recharger B et C avec les valeurs de travail du registre HL car, dans le système du Sinclair, le retour du langage machine se fait par l'intermédiaire du registre BC.

Maintenant que nous avons rédigé ce programme machine qui semble se tenir, il va être nécessaire de l'implanter dans un programme Basic plus large qui puisse appeler la routine ainsi conçue. Pour ce faire, nous allons placer la série des codes avec leur valeur décimale dans une instruction **DATA** (*figure 7*, ligne 9000), suivie de 16 codes qui sont la traduction décimale de

```

1 REM ADDITION N PREMIERS NO.
2 REM -----
10 INPUT N
20 POKE 23681,N
30 LET K=USR 32500
40 PRINT K
50 STOP
9000 DATA 58,129,92,71,33,0,0,22
,0,88,25,16,252,68,77,201
9010 FOR I=1 TO 20
9015 READ A
9020 POKE (32599+I),A
9030 NEXT I

```

Figure VI-7

l'hexadécimal du programme précédent. Cette instruction **DATA** sera suivie d'une boucle de lecture qui **POKE**ra successivement les différents codes dans une zone mémoire très adaptée à cet usage, étant donné la protection dont elle bénéficie contre tout effacement accidentel : ce sont les octets en principe réservés à la définition de caractère et qui commencent à l'adresse 32600.

Le programme Basic d'appel est simple : le nombre supérieur de la série, dont on demande la sommation, est entré par la ligne 10 et stocké à l'octet 23681. Puis, on appelle la routine de langage machine avec l'instruction :

LET K =USR 32600

et l'on demande l'impression du résultat de cette routine. En choisissant N assez grand, par exemple 255, le lecteur se rendra aisément compte de la différence de rapidité entre le programme Basic et le programme machine.

Deuxième programme : coloriage de l'écran par grandes masses

Après cet exemple purement explicatif, développons un second programme plus utile pour la programmation avancée et qui utilise toutes les possibilités couleur du Spectrum.

Le lecteur qui aura mis en oeuvre le programme Gibraltar du chapitre IV, se sera rendu compte de la lenteur des routines destinées à planter le décor du jeu en coloriant l'écran par grandes masses. Le Basic a recours à des boucles qui sont toujours longues à lire pour l'ordinateur qui les interprète. Maintenant, nous allons proposer une routine qui effectue ce travail instantanément, dès que les paramètres essentiels lui auront été communiqués, à savoir la hauteur, la largeur, la couleur et l'emplacement précis des rectangles souhaités. Rappelons qu'un rectangle est une figure très générale qui peut être réduite à une ligne en hauteur ou en largeur et occuper n'importe quelle portion verticale ou horizontale d'écran.

La liste en assembleur de ce sous-programme très efficace est représentée sur la *figure VI-8*. Il est assemblé à partir de l'adresse, désormais familière, 32600. Sa structure est très simple puisqu'elle consiste à faire un usage intensif de la routine d'impression du système lui-même qui figure dans la ROM et qui est appelée avec l'instruction **RST 10h**, soit **RST 16d**. Cette routine im-

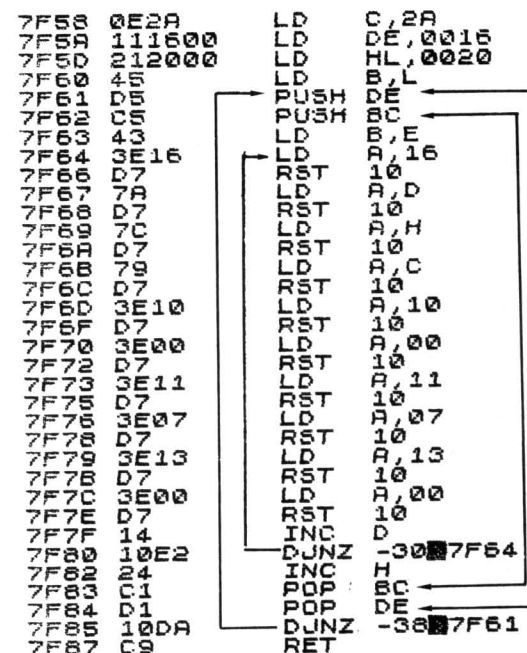


Figure VI-8

prime à l'emplacement d'écran défini par le contenu du registre D (indice de colonne) et le contenu du registre H (indice de ligne), un pavé dont les différents attributs seront déterminés par des chargements successifs du registre A. Dans le cas figuré, l'encre est égale à 0, le papier égal à 1, la brillance égale à 0, et l'on demande d'imprimer sur la totalité de l'écran.

A l'initialisation, le contenu du registre est le suivant : le registre C contient le type du caractère que l'on souhaite afficher, le registre E : la hauteur, le registre L : la largeur, le registre D : la ligne de départ, le registre H : la colonne de départ. On fait usage des instructions de *pile* pour la raison que c'est la même instruction **DJNZ**, très utile par la décrémentation qu'elle organise, qui sert à décrémenter et la ligne et la colonne. Il faut donc sauver provisoirement chaque valeur intermédiaire, ligne ou colonne.

```

1 REM COLORIAGE PAR MASSES
2 REM -----
3 REM
4 DATA 14,42,17,10,0,33,32,0,
59 213,197,67,62,22,215,122,215,
124 215,121,215,62,16,215,62,1,2
165 62,17,215,62,5,215,62,19,215,
62,1,215,20,16,226,36,193,209,16
,218,201
5 LET A=32600
6 FOR I=0 TO 47: READ C: POKE
(A+I),C: NEXT I
7 STOP
8 REM ESSAIS
9 REM
10 INPUT "caractere ?";A$
20 LET B=CODE A$
30 POKE 32601,B
40 INPUT "hauteur ?";H
45 POKE 32603,H
50 INPUT "ligne de depart ?";L
55 POKE 32604,L
60 INPUT "largeur ?";E
65 POKE 32606,E
70 INPUT "colonne de depart ?"
;K
75 POKE 32607,K
80 INPUT "encre ?";F
85 POKE 32625,F
90 INPUT "papier ?";P
95 POKE 32631,P
100 INPUT "brillance ?";G
105 POKE 32637,G
110 PRINT AT 10,10;"*"
120 LET K=USR 32600
190 GO TO 10
300 REM
1000 REM EXEM: DRAPEAU FRANCAIS
1001 REM
1100 LET CA=32601
1110 LET HA=32603
1120 LET LI=32604
1130 LET LA=32606
1140 LET CO=32607
1150 LET EN=32625
1160 LET PA=32631
1170 LET BR=32637
1200 REM DEBUT
1220 POKE CA,32
1230 POKE HA,20
1250 POKE LA,10
1280 POKE BR,0
1310 PRINT PAPER 1;AT 0,0;" "
1320 POKE PA,1
1330 POKE LI,0
1340 POKE CO,0
1360 LET K=USR 32600
1390 PRINT PAPER 2;AT 0,20;" "
1400 POKE PA,2
1410 POKE CO,20
1430 LET K=USR 32600
1500 STOP

```

Figure VI-9

Détail du programme "coloriage par masse"

Dans le programme coloriage par masses (figure VI-9), le chargement de la routine se fait selon la procédure classique par une instruction **DATA**, et le chargement dans les octets successifs à partir du pas 32600 (lignes 1 à 9).

Puis figure une routine destinée aux essais, où l'utilisateur peut charger successivement les octets sensibles comprenant les principaux paramètres avec les valeurs qu'il souhaite pour réaliser, au coup par coup, les coloriages par masses qu'il désire.

La ligne 110 mérite une explication : elle consiste à imprimer un astérisque dans un emplacement indifférent de l'écran, avant que soit lancée la routine CM. Cette opération a pour objet d'initialiser la routine d'impression de la mémoire morte ; il nous aurait été possible de faire précéder la routine en code machine d'une procédure semblable d'initialisation, mais elle nous aurait emmené trop loin dans les secrets du système ZX.

Une fois que l'utilisateur aura compris le mécanisme de la routine, il peut sortir de la boucle en faisant un **STOP** et commander l'impression du drapeau français en faisant **GOTO 1000**.

Le début de ce programme va qualifier, en tant que variables, les principaux octets à modifier selon les normes suivantes : CA = caractère, HA = hauteur, LI = ligne de départ, LA = largeur, CO = colonne de départ, EN = encre, PA = papier, BR = brillance. Et c'est seulement après cette initialisation qu'il sera possible de procéder au coloriage par masses. L'impression de chacun des deux grands rectangles de couleur, bleu et rouge, est confiée à la routine 32600 ; auparavant toutefois la première case de chaque rectangle de couleur est imprimée par une instruction spécialisée, ici 1310 et 1390 (pour effectuer l'initialisation de la routine machine d'une façon qui convienne au graphisme général).

EXERCICE 1 : le lecteur est invité à allonger ce programme d'un certain nombre de sous-programmes destinés à imprimer, avec la même rapidité, n'importe quel drapeau de son choix, compte tenu du fait que la routine en question n'est pas adaptée aux coloriages en diagonale (mais la grande majorité des drapeaux du monde est composée de coloriages rectangulaires). Il trouvera,

en corrigé, des propositions d'exécution pour les drapeaux suivants : américain, belge, suisse et canadien (figure VI.10).

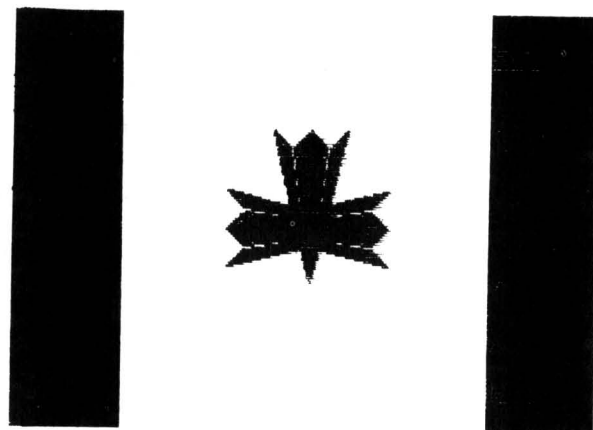


Figure VI-10

ANNEXE

CORRIGES DES EXERCICES

EXERCICE 2-1

```

4180 IF INKEY$="S" OR INKEY$="s"
  THEN GO TO 9000

9000>REM SUPPRESSION
9100 LET I=E+C
9150 CLS : PRINT FLASH 1;"A SUPP
RIMER:" : PRINT T$(I)
9200 FOR K=I TO 100
9250 LET T$(K)=T$(K+1): LET P$(K
)=P$(K+1)
9350 NEXT K
9400>LET P$(101)=Z$: LET T$(101)
="RIEN"
9500 GO TO 2000
  
```

EXERCICE 2-2

Il faut modifier le programme en début et en fin, de la façon suivante. Pour ce qui est de l'impression du nombre maximum de lignes d'écran, 22, il suffit d'installer un nouveau compteur L (ligne 906). Pour ce qui est de la transformation en titre, il faut faire tester le premier caractère de chaque nouveau paragraphe. S'il est égal à "t", on renvoie à une routine de transformation en titre (ligne 5000). Cette routine est simple à comprendre : elle calcule la longueur du paragraphe à transformer en titre et le divise éventuellement, s'il dépasse une ligne, en deux éléments, la deuxième ligne étant plus brève que la première, cette division s'effectuant sur la base de la prise en compte d'un blanc. Enfin, la ligne 5000 effectue un centrage pur et simple. L'objet de la ligne 5040 est de transformer le paragraphe de minuscules en majuscules.


```

700>LET L=0
800 REM NOUVEAU PARAGRAPHE
801 REM
810 READ N$
830 LET N=1
840 IF N$(1)="t" THEN GO TO 500
0
890 REM
900 REM NOUVELLE LIGNE
901 REM
905 LET D=0: LET A=0
906 LET L=L+1: IF L>22 THEN GO
TO 4500
910 LET P$=N$(N TO )
990 REM
1000 REM SITUATION FIN DE LIGNE

```

```

4500>COPY : CLS : LET L=0: GO TO
900
5000 REM TITRE
5001 REM
5010 LET V=LEN N$(3 TO )
5020 FOR W=3 TO 2+V
5040 IF CODE N$(W)>=97 THEN LET
N$(W)=CHR$(CODE N$(W)-32)
5050 NEXT W
5055 IF V<=30 THEN PRINT TAB (32
-V)/2;N$(3 TO ): PRINT : GO TO 8
00
5060 FOR W=V/2 TO V
5070 IF N$(W+2)=" " THEN GO TO 5
100
5080 NEXT W
5100 PRINT TAB (32-W)/2;N$(3 TO
W+1): PRINT TAB (30-V+W)/2;N$(W+
2 TO )
5200 PRINT : GO TO 800

```

EXERCICE 2-3

Réponse : voici une proposition de programme qui comporte les trois fonctions. Son incorporation, dans le programme principal, nécessitera d'examiner avec soin l'actualisation des variables d'entrée et de sortie, notamment FE, FC.

```

4500 INPUT INVERSE 1:"ENREGISTRE
MENT A SUPPRIMER ?";P0
4600 LET P1=FN Y(P0)
4620 LET P2=FN Y(P0+1)-1
4630 LET LE=P2-P1
4640 FOR J=P1 TO FC-LE
4650 LET C$(J)=C$(J+LE)
4660 NEXT J
4700 FOR I=P0+1 TO FE
4720 LET P1=FN Y(I)
4730 LET P1=P1-LE
4740 LET E$(I)=CHR$(FN X(P1)+CHR
$(FN Z(P1)
4750 NEXT I
4760 LET E$(I)=" "
4800 REM
4810 FOR H=2 TO N
4820 FOR J=15 TO 1 STEP -1
4840 IF M$(H,J)=CHR$(P0) THEN GO
SUB 4900
4850 IF M$(H,J)>CHR$(P0) THEN LET
M$(H,J)=CHR$(P0-1)
4855 IF M$(H,J)=CHR$(0) THEN NEXT
H
4860 NEXT J
4870 NEXT H
4880 GO TO 100
4900 FOR G=J TO 1 STEP -1
4920 LET M$(H,J)=M$(H,J-1)
4930 IF M$(H,J)=CHR$(0) THEN RETU
RN
4940 NEXT G: RETURN
4980 GO TO 100
4990 REM

```

*
* *
*

EXERCICE 3-2

```

430>LET U=(I/2-INT (I/2))*2
450 GO SUB 500
470 NEXT I
480 GO TO 1500
490 REM
500 REM Grisage
510 LET L=X(I+1)-X(I)
520 FOR K=110*(U=0)+150*(U=1) T
0 150*(U=0)+110*(U=1) STEP I*(1-
2*U)
525 LET Q=(150-K)*(U=0)+(K-110)
*(U=1): LET L1=L*(Q)=L)+Q*(Q/L)
530 PLOT X(I),K: DRAW L1,L1*(1-
2*U)
540 NEXT K
550 FOR X=X(I) TO X(I+1) STEP I
555 LET L=X(I+1)-X: LET L1=L*(L
<40)+40*(L)>=40)
560 PLOT X,110*(U=0)+150*(U=1):
DRAW L1,L1*(1-2*U)
570 NEXT X: RETURN

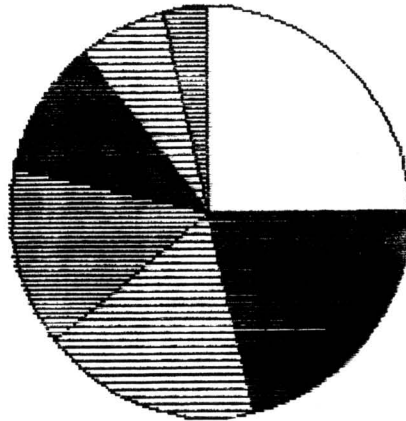
```

EXERCICE 3-3

```

3.57 %
7.14 %
10.71 %
14.28 %
17.85 %
21.42 %
25 %
28.57 %
32.14 %
35.71 %
39.28 %
42.85 %
46.42 %
50 %
53.57 %
57.14 %
60.71 %
64.28 %
67.85 %
71.42 %
75 %
78.57 %
82.14 %
85.71 %
89.28 %
92.85 %
96.42 %
100 %

```



```

4150>LET P=(T(K,1)-1)/3
4155 LET P=(P-INT P)*3+1
4160 LET P=P*SGN SIN Z
4165 PRINT P

```

L'astuce a consisté à diviser le pas par le moyen d'une division modulo N pour retenir le reste de la division. Ici, le diviseur retenu a été 3. Il faut, évidemment, choisir un diviseur plus petit que le nombre des données, faute de quoi le reste, retourné par la division modulo N, serait strictement équivalent au numéro d'ordre des secteurs.

La ligne 4155 ajoute 1 au reste pour éviter le cas où le pas est égal à 0, cas où le programme ne progresse plus.

EXERCICE 3-4

```

900>REM Surimposition 2eme grap
905 REM
910 INPUT "Nouveau graphique ?
(O/N) ";R$
920 IF R$="N" THEN STOP
930 FOR I=1 TO N
940 INPUT ("Y pour etape: ";I);
? ";T(I,2)
950 NEXT I
960 GO TO 800

```

Cette routine est très simple, à la condition de ne surimposer qu'un graphique qui entre parfaitement dans les données extrêmes définies pour les graphiques précédents. Il y a donc juste à substituer les nouvelles valeurs aux premières dans le tableau d'entrée, puis commander directement la routine 800 de traçage de la ligne. Il n'y a pas à revenir sur les paramètres déjà calculés. Voici un exemple de court programme permettant l'introduction de nouvelles données.

EXERCICE 3-5

```

1000 FOR X=-120 TO 120
1100 LET Y=.25*50R (120*120-X*X)
1200 IF X/2-INT (X/2)=0 THEN PLO
T X+120,Y+50: PLOT X+120,50-Y
1400 FOR Y=Y TO -Y STEP -5
1500 PLOT X+120,Y+50
1600 NEXT Y
1700 NEXT X

```

*
* *
*

EXERCICE 4-1

```

100 FOR I=0 TO 20
110 FOR J=1 TO 31
120 PRINT INK I/3;AT I,J;"*"
130 NEXT J
140 NEXT I
200 LET L=10: LET C=15
205 LET S=0
210 GO TO 340
290 IF INKEY$="" THEN GO TO 290
300 LET C=C+(INKEY$="8")-(INKEY$="5")
310 IF C<0 OR C>31 THEN GO TO 500
320 IF L<0 OR L>20 THEN GO TO 500
340 PRINT INK L/3; INVERSE 1;AT L,C;"@"
345 LET S=S+1
350 GO TO 290
500 BEEP 1,10: PRINT AT 0,0;S

```

EXERCICE 4-2

Il faut développer la ligne 1330 en testant l'attribut de la case immédiatement supérieure au dessin du sous-marin (surface) ou immédiatement inférieure à ce dessin (fond). La mer profonde, dans laquelle se trouve le sous-marin, étant bleu foncé, son attribut sera différent des attributs de la surface, bleu clair, et du fond, noir.

Le lecteur constatera que l'introduction de ces deux tests supplémentaires ralentit sensiblement le jeu : en effet, le test de la fonction ATTR est extrêmement complexe pour le système. L'attribut du fond est égal à 0 et l'attribut de la surface est égal à 40.

```

1330>IF ATTR (X,Y+3)=208 OR ATTR
(X-2,Y)=40 OR ATTR (X+2,Y)=0 TH
EN GO TO 1500

```

EXERCICE 4-3

```

10 DATA .5,-1,.25,-6,.5,-1,1,3
11 DATA .5,3,.25,-1,.5,3,1,6,1
12 DATA .5,6,.25,3,.5,6,1,10,.
13 DATA .5,3,.25,-2,.5,3,1,5,6
14 LET c=-12
100 FOR i=0 TO 18
110 READ a,b: BEEP a,b+c: NEXT i
120 LET c=c+2: RESTORE 10: GO TO 100

```

EXERCICE 4-4

```

1 REM LE SOLFEGE DE LUCIE
2 REM -----
10 DIM N$(8,3)
20 DIM N(8)
100 DATA "Do",0,"Re",2,"Mi",4,"
Fa",5,"Sol",7,"La",9,"Si",11,"Do"
110 FOR I=1 TO 8
120 READ N$(I),N(I)
130 NEXT I
140 FOR I=1 TO 8
150 PRINT PAPER (8-I);"
160 NEXT I
170 LET C=0
180 PAUSE 4E4
200 IF INKEY$="" THEN GO TO 200
210 LET Y=VAL INKEY$
220 BEEP 0.5,N(Y)
230 PRINT PAPER (Y-1); INK 9;AT
8-Y,C;N$(Y)
250 LET C=C+2+(Y=5)

```

*
* *

EXERCICE 5-1

```

10 LET D=15624
15 FOR L=0 TO 16: FOR C=1 TO 1
6 20 FOR I=0 TO 7
30 LET K=PEEK (D+I)
40 FOR J=1 TO 8
45 LET K=INT K
50 LET K=K/2
60 IF (K-INT K)=0 THEN GO TO 6
0 70 LET X=C*16-J*2: LET Y=175-I
  *2-L*16
71 PLOT X,Y: PLOT X-1,Y: PLOT
X,Y-1: PLOT X-1,Y-1
80 NEXT J
90 NEXT I
100 LET D=D+8
110 NEXT C: NEXT L

```

EXERCICE 5-2

```

5 DATA 0,60,66,66,126,66,66,0
7 FOR K=0 TO 2
9 RESTORE 5
10 FOR I=0 TO 7
20 READ A
30 FOR J=16384+2048*K+256*I TO
16384+2048*K+256*(I+1)
40 POKE J,A
50 NEXT J
60 NEXT I
70 NEXT K

```

EXERCICE 5-3

```

5 REM DEFILEMENT VERS LE BAS
6 REM -----
10 FOR I=0 TO 15
20 PRINT "#####"
30 NEXT I
100 LET D=20448
110 FOR L=0 TO 7
120 LET P=D
130 FOR K=1 TO 7
140 FOR J=0 TO 31
150 POKE (P+J),PEEK (P+J-256) ;
160 NEXT J
170 LET P=P-256
180 NEXT K
190 FOR J=0 TO 31
195 POKE (J+P),(L*7)*PEEK (P+J+
1760)+(L*7)*PEEK (P+J-32)
200 NEXT J
210 LET D=D-32
220 NEXT L
230 LET D=D-1792
240 GO TO 110
400 STOP
500 DATA 1,3,7,15,31,63,127,255
510 FOR I=0 TO 7
515 READ A
520 POKE (32600+I),A
530 NEXT I

```

EXERCICE 5-4

La réponse est très simple : le programme va quand même se positionner sur l'octet de début de mémoire programme et lire, à partir de là, l'ensemble des numéros de ligne (9015-9020) jusqu'à ce qu'il rencontre le numéro de ligne à partir duquel on lui a demandé d'exécuter le renumérotage : c'est la ligne 9030. A partir de là, le programme est strictement identique au programme précédent, à ceci près que l'utilisateur a été prié de donner également un numéro de ligne N à laquelle l'ordinateur doit s'arrêter : ce nombre N est également codé sur deux octets (ligne 9160).

```

9000>REM RENUMEROTAGE
9001 REM -----
9005 INPUT "A PARTIR DE QUELLE L
IGNE ?":N
9006 INPUT "JUSQU'A QUELLE LIGNE
?":M
9010 INPUT "nouveau pas?":P
9015 LET Y=23755
9020 LET A=INT (M/256): LET B=(M
/256-INT (M/256))*256
9030 IF 256*PEEK Y+PEEK (Y+1)=N
THEN GO TO 9100
9050 LET Y=Y+PEEK (Y+2)+256*PEEK
(Y+3)+4
9060 GO TO 9030
9100 LET D=N
9120 POKE Y,INT (D/256)
9130 POKE (Y+1),INT ((D/256-INT
(D/256))*256)
9140 LET D=D+P
9150 LET Y=Y+PEEK (Y+2)+256*PEEK
(Y+3)+4
9160 IF PEEK Y=A AND PEEK (Y+1)=
B THEN GO TO 9180
9170 GO TO 9120
9180 LIST

```

EXERCICE 5-5

```

1 REM MARQUEUR DE PLACE MEMO.
2 REM -----
100 GO SUB 200
110 STOP
200 REM
210 GO SUB 300
220 RETURN
300 REM
310 GO SUB 400
320 RETURN
400 REM
410 GO SUB 500
420 RETURN

```

```

→ 500 REM
510 GO SUB 600
520 RETURN
600 CLS : GO SUB 9000
610 RETURN
1000 DIM A(1000)
9000 FOR I=23760 TO 32599 STEP 1
9010 IF I<(PEEK 23627+256*PEEK 2
3628) THEN LET A=80: LET P=1: GO
TO 9070
9020 IF I<(PEEK 23641+256*PEEK 2
3642) THEN LET A=86: LET P=5: GO
TO 9070
9030 IF I<(PEEK 23653+256*PEEK 2
3654) THEN LET A=84: LET P=2: GO
TO 9070
9040 LET K=USR 32602
9050 IF I<(PEEK 32600+256*PEEK 3
2601) THEN LET A=32: LET P=4: GO
TO 9070
9060 LET A=84: LET P=2
9070 PRINT PAPER P; INK 9;CHR$ A
: NEXT I
9400 RETURN
9500 DATA 237,115,88,127,201
9510 FOR I=0 TO 4: READ A: POKE
(32602+I),A: NEXT I

```

*
* *
*

EXERCICE 6-1

```

1600 REM DRAPEAU AMERICAIN
1601 REM -----
1605 POKE CO,0: POKE HA,1: POKE
LA,20: POKE PA,2
1610 FOR I=0 TO 12 STEP 2
1620 POKE LI,I
1625 PRINT PAPER 2;AT I,0;" "
1630 LET K=USR 32600
1640 NEXT I
1650 POKE CA,42: POKE LI,0: POKE
PA,1: POKE EN,7: POKE BR,1: POK
E HA,7: POKE LA,7
1660 PRINT INK 7; PAPER 1; BRIGH
T 1;AT 0,0;"*"
1670 LET K=USR 32600
1690 STOP
1700 REM DRAPEAU BELGE
1710 REM -----
1730 POKE LA,10: POKE HA,20: POK
E LI,0: POKE CA,32: POKE BR,0
1740 LET A$="062"
1750 FOR I=0 TO 2
1760 POKE PA,VAL A$(I+1): POKE C
O,I+10
1770 PRINT PAPER VAL A$(I+1);AT
0,I+10;" "
1780 LET K=USR 32600
1790 NEXT I
1795 STOP
1800 REM DRAPEAU SUISSE
1810 REM -----
1820 POKE HA,20: POKE LA,32: POK
E PA,2: POKE LI,0: POKE CO,0: PO
KE CA,32
1830 PRINT PAPER 2;AT 0,0;" "
1840 LET K=USR 32600
1850 POKE PA,7: POKE LI,4: POKE
CO,14: POKE HA,12: POKE LA,4
1860 PRINT PAPER 7;AT 4,14;" "
1870 LET K=USR 32600
1875 POKE LI,8: POKE CO,10: POKE
HA,4: POKE LA,12
1880 PRINT PAPER 7;AT 8,10;" "
1885 LET K=USR 32600
1890 STOP
1900 REM DRAPEAU CANADIEN
1910 REM -----
1920 POKE CA,32: POKE PA,2: POKE
LI,0: POKE CO,0: POKE HA,22: PO
KE LA,6: POKE BR,0
1930 PRINT PAPER 2;AT 0,0;" "
1940 LET K=USR 32600
1950 POKE CO,26
1960 PRINT PAPER 2;AT 0,26;" "
1970 LET K=USR 32600
2000 BORDER 0: INK 2
2100 FOR J=1 TO 4
2110 DEF FN Y(I)=8*(J<=2)-8*(J>2
)-I*((J/2-INT (J/2))=0)+I*((J/2-
INT (J/2))<>0)
2120 FOR I=1 TO 8
2130 LET X=56+I
2140 PLOT 100,84
2150 DRAW X,FN Y(I)
2155 PLOT 156,84
2156 DRAW -X,FN Y(I)
2157 PLOT 130,61
2158 DRAW FN Y(I),X
2160 NEXT I
2170 NEXT J

```


I N D E X

ASCII, 11, 24, 32
 ATTR, 91, 93, 119, 120
 Acoustique, 104
 Affectation de Procruste, 34
 Assembleur, 141
 Attributs, 86

BEEP, 104
 BIN, 14
 BREAK, 109
 BRIGHT, 86, 95
 Bruitages, 102

CHR\$, 87, 89
 CIRCLE, 60
 Caracteres predefinis, 17
 Changement variables, 130
 Correlation, 73
 Courbe exponentielle, 46
 Courbe logarithmique, 46
 Courbes 3 dimensions, 77, 78
 Courseurs, 20

DATA, READ, RESTORE, 9, 28, 149
 DEF FN, FN, 10, 83
 DIM, 34
 DJNZ, 149
 DRAW, 53, 56, 61
 Decalage-insertion, 42
 Decoupage de chaine, 32
 Defilements d'ecran, 122

Echelle, 87

FLASH, 86, 95
 Fichiers en tableau, 35
 Fichier a chaine unique, 35

Graphique 2 dimensions, 57
 Graphique circulaire, 60, 63
 Graphique-bande, 52
 Graphique-barre, 49, 50, 51
 Grises, 54

Hexadecimal, 145

INK, 86, 87
 INKEY\$, 17, 30, 97
 INVERSE, 87
 Insertion de tirets, 27
 Interpreteur Basic, 132

Justification a droite, 25

LEN, 69
 Langage machine, 139
 Ligne brisee, 86

Matrice graphique, 17
 Microprocesseur, 141
 Mnemonique, 144
 Moindres carres, 71, 72
 Mots-cles, 42, 43

OVER, 15, 87

PAPER, 86, 93
 PEEK, 114, 115
 PLOT, 14, 117
 POKE, 17, 111, 114
 PUSH, 143
 perspective, 83
 Pile de GOSUB, 135
 Pixel, 119
 Pointeurs, 43
 Principe LIFO, 143
 procedure dichotomique, 36
 prog.ADDITION SERIE, 148
 prog.CAMEMBERT, 62, 64
 prog.CHENILLE, 94
 prog.COLORIAGE MASSES, 152
 prog.COUBRES 3*0, 82
 prog.DEFILEMENT D'ECRAN, 123
 prog.FICHER, 40
 prog.GIBRALTAR, 98
 prog.GRAPHIQUE 2*0, 57
 prog.GRAPHIQUE BANDE, 53, 55
 prog.GRAPHIQUE SIMPLE, 68
 prog.INDEX, 21
 prog.ORGUE ELECTRONIQUE, 107
 prog.REGRESSIONS, 74
 prog.TRAITEMENT DE TEXTE, 28
 prog.VISU.MEMOIRE, 135, 136
 prog.WAKYRIES, 106
 Puissances, 48

RAM, 113
 ROM, 113, 131
 Recherche sequentielle, 36
 Reference d'index, 20
 Registres, 141, 143
 Regressions, 70
 Renumerotage, 128

SCREEN\$, 91
 STR\$, 69

Tampon d'imprimante, 124
 Tri, 37

USR, 112, 150

VAL, 10
 Variable-systeme, 117

Zone d'affichage, 119
 Zone programme, 125, 128

Achevé d'imprimer en juin 1984
sur les presses de l'imprimerie Laballery et C^{ie}
58500 Clamecy
Dépôt légal : juin 1984

N° d'impression : 404090
N° d'édition : 86595-78-3
ISBN : 2-86595-078-6



EDITIONS DU P.S.I.

90,00 FF

la pratique du ZX Spectrum

Destiné à compléter la documentation de base du ZX Spectrum, cet ouvrage exploite toutes les possibilités du Sinclair dans le domaine de la programmation avancée. L'auteur y expose d'une manière soigneusement graduée : la définition de caractères et les fonctions de chaîne (fichiers, alphabets, traitement de texte, etc.); la haute définition graphique et la puissance de calcul (régression, graphiques en trois dimensions, etc.); la couleur et le son (différents jeux); les techniques d'accès direct à la mémoire et à l'utilisation du langage machine. Les possesseurs de ZX81 trouveront dans cet ouvrage plusieurs programmes transposables pouvant être réintroduits tels quels sur leur appareil.

couverture A GRAPH

EDITIONS DU P.S.I.

BP 86

France - 77402 Lagny s/marne cedex

ISBN 2-86595-078-6

imprimé en France