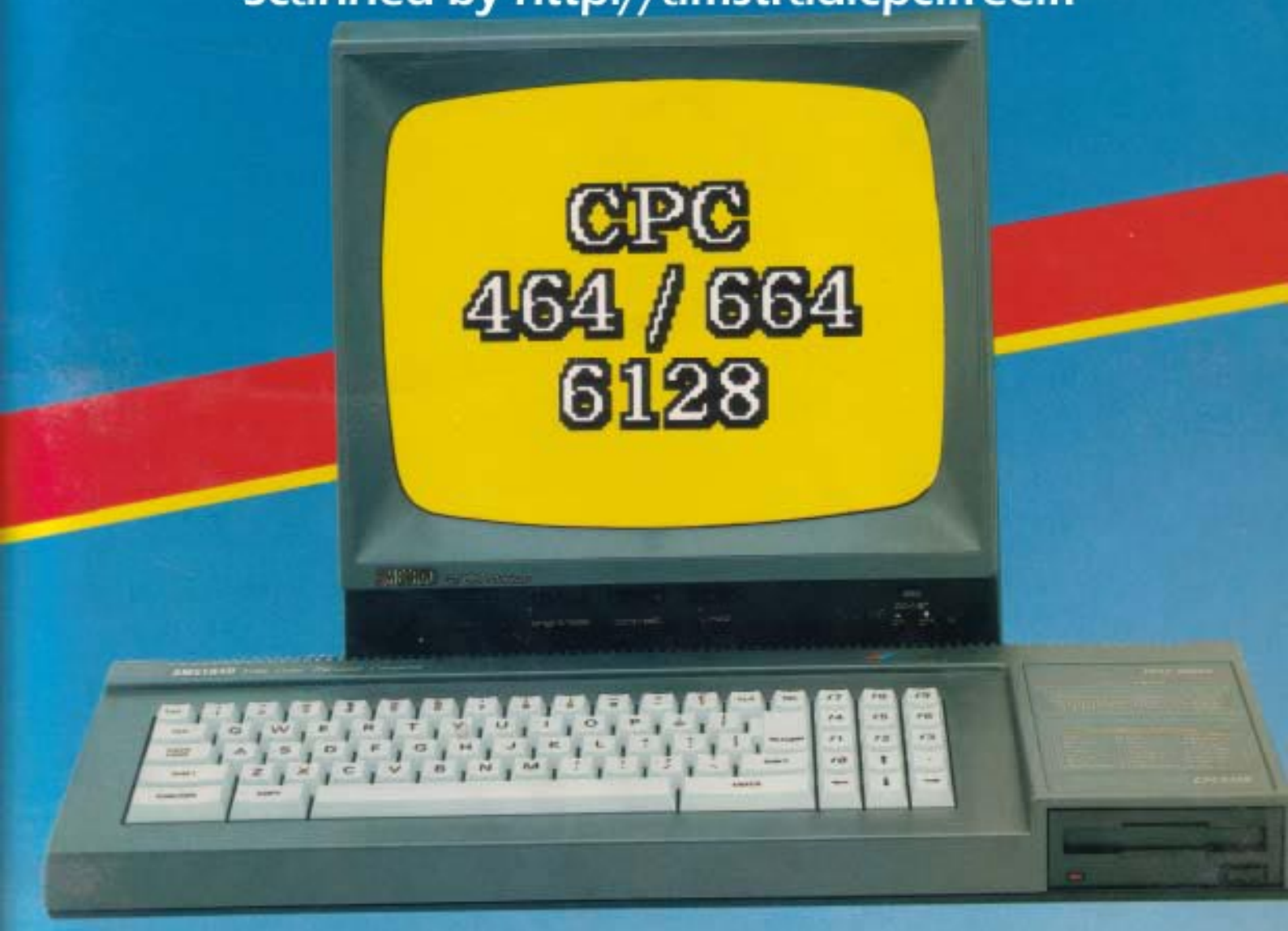
# Comment exploiter toutes les ressources et augmenter les performances de votre

AMSTRAD






ÉDITIONS WEKA

# 5/0

# Table des matières

## 5/10.4 Graphicomanies

# 5/1

# Généralités

Les AMSTRAD CPC possèdent trois modes d'affichage qui seront utilisés en fonction du type de l'application.

MODE 0 : 25 lignes de 20 caractères, 16 couleurs, définition 160 × 200 pixels.

MODE 1 : 25 lignes de 40 caractères, 4 couleurs, définition 320 × 200 pixels.

MODE 2 : 25 lignes de 80 caractères, 2 couleurs, définition 640 × 200 pixels.

Aucune distinction n'est faite entre l'écran texte et l'écran graphique. Il existe même une instruction (TAG) qui permet d'afficher un texte alphanumérique à une position graphique (au pixel près) sur l'écran.

L'écran est divisé en 400 points verticaux et 640 points horizontaux respectivement repérés de 0 à 399 et de 0 à 639.

En MODE 0, comme la résolution est de 160 × 200, un point élémentaire occupe (640/160 = ) 4 pixels horizontaux et (400/200 = ) 2 pixels verticaux.

En MODE 1, comme la résolution est de 320 × 200, un point élémentaire occupe (640/320 = ) 2 pixels horizontaux et (400/200 = ) 2 pixels verticaux.

En MODE 2, comme la résolution est de 640 × 200, un point élémentaire occupe (640/640 = ) 1 pixel horizontal et (400/200 = ) 2 pixels verticaux.

A la mise sous tension, le MODE d'affichage est le MODE 1.

# 5/2

# Tracé de points en BASIC

En BASIC, nous disposons de trois instructions pour accéder à un point élémentaire :

PLOT, PLOTR et GRAPHICS PEN.

**PLOT <Abcisse>, <Ordonnée>[, <encre>[, <mode d'encre>]]**

— <Abcisse> est compris entre 0 et 639,

— <Ordonnée> est compris entre 0 et 399.

Cet ordre permet d'afficher un point élémentaire aux coordonnées absolues spécifiées.

**PLOTR <Déplacement en X>, <Déplacement en y>**
**[,<encre>[, <mode d'encre>]]**

Cet ordre permet d'afficher un point élémentaire aux coordonnées relatives (par rapport au point courant) spécifiées.

Si X, Y est la position courante du curseur graphique, un point sera affiché en X + <Déplacement en X>, Y + <Déplacement en Y>.

**GRAPHICS PEN [<encre>][, <Mode d'affichage du fond>]**

Cet ordre n'existe pas sur CPC 464.

Si vous possédez un CPC 664 ou 6128, GRAPHICS PEN vous permettra de choisir une couleur de stylo (<encre>) pour dessiner, et la nature du fond de l'écran (<Mode d'affichage du fond> = 0 pour un fond opaque, et = 1 pour un fond transparent.)

# 5/3

# Déplacement du curseur graphique en BASIC

En BASIC, nous disposons des instructions MOVE, MOVER, ORIGIN, XPOS et YPOS.

**MOVE <abscisse>, <ordonnée>[, <encre>[ <mode d'encre>]]**

— <abscisse> est compris entre 0 et 639,

— <ordonnée> est compris entre 0 et 399.

Cet ordre permet de déplacer le curseur graphique aux coordonnées absolues spécifiées sans afficher le point élémentaire pointé.

Le paramètre <encre> permet, s'il est présent, de changer la couleur du stylo graphique.

Le paramètre <mode d'encre> permet, s'il est présent, de fixer le mode d'affichage graphique :

0 pour le mode normal,

1 pour le mode XOR,

2 pour le mode AND,

3 pour le mode OR.

**MOVER <déplacement en X>, <déplacement en Y>**
**[,<encre>[<mode d'encre>]]**

Cet ordre permet de déplacer le curseur graphique aux coordonnées relatives (par rapport au point courant) spécifiées sans afficher le point élémentaire pointé.

Le paramètre <encre> permet, s'il est présent, de changer la couleur du stylo graphique.

Le paramètre <mode d'encre> permet, s'il est présent, de fixer le mode d'affichage graphique :

0 pour le mode normal,

1 pour le mode XOR,

2 pour le mode AND,

3 pour le mode OR.

**ORIGIN <abscisse>, <ordonnée>[, <gauche>, <droite>, <haut>, <bas>]**

Cet ordre fixe les coordonnées du point qui sera pris comme origine de l'écran graphique. Par défaut, l'origine est en 0, 0.

Quand les paramètres <gauche>, <droite>, <haut>, <bas> sont précisés, ils définissent les dimensions de la fenêtre graphique.

**XPOS** indique l'abscisse courante du curseur graphique (entre 0 et 639).

**YPOS** indique l'ordonnée courante du curseur graphique (entre 0 et 399).

# 5/4

# Tracé de droites en BASIC

En BASIC, nous disposons de quatre instructions relatives au tracé de droites : DRAW, DRAWR, GRAPHICS PEN et MASK.

**DRAW <Abscisse>, <Ordonnee>[, <encre>[, <mode d'encre>]]**

— <Abscisse> est compris entre 0 et 639,

— <Ordonnée> est compris entre 0 et 399.

Cet ordre permet de tracer un trait de la position courante du curseur graphique à la position absolue spécifiée.

**DRAWR <Deplacement en X>, <Deplacement en Y>**
**[, <encre>[, <mode d'encre>]]**

Cet ordre permet de tracer un trait de la position courante du curseur graphique à la position relative (par rapport au point courant) spécifiée.

Si la position courante du curseur graphique est X,Y, le trait prendra fin en X + <Deplacement en X>, Y + <Deplacement en Y>.

**GRAPHICS PEN [<encre>][, <Mode d'affichage du fond>]**

Cet ordre n'existe pas sur CPC 464.

Si vous possédez un CPC 664 ou 6128, GRAPHICS PEN vous permettra de choisir une couleur de stylo (<encre>) pour dessiner, et la nature du fond de l'écran (<Mode d'affichage du fond> = 0 pour un fond opaque, et = 1 pour un fond transparent.)

**MASK [<entier>][, <premier point tracé>]**

— <entier> est compris entre 0 et 255,

— <premier point tracé> vaut 0 ou 1.

Cet ordre permet de définir le mode de tracé des lignes fabriquées à partir des ordres DRAW ou DRAWR.

Le paramètre <entier> est un nombre sur 8 bits où chaque bit représente un pixel. Si un bit est à 1, le pixel correspondant est allumé. Le pixel est éteint si le bit est à zéro.

Si le paramètre <premier point trace> vaut 0, le premier point n'apparaîtra pas. Le premier point apparaîtra si ce paramètre vaut 1.

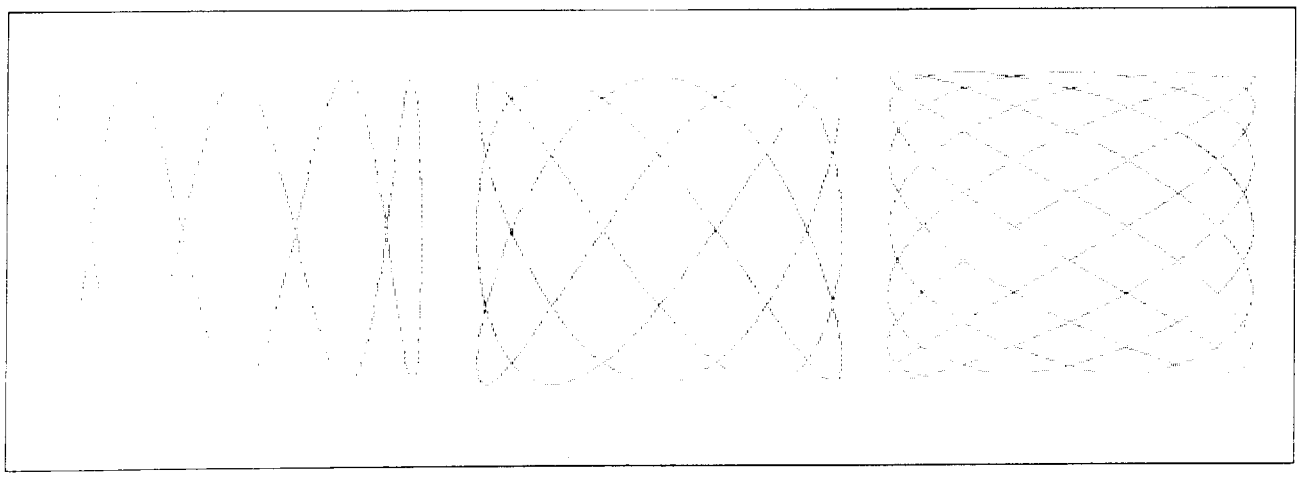
Pour illustrer

- le tracé de droites avec l'ordre DRAW,
- le positionnement du curseur avec l'ordre MOVE ou PLOT.

Nous vous proposons une série de programmes de tracé de courbes de type Y(t), X(t).

a) Courbes de Lissajous

```
10 CLS
20 FOR F=0.2 TO 4 STEP 0.2
30    PLOT 490,200
40    A=0
50    A=A+0.1
60    DRAW 300+190*COS(A*F), 200+190*SIN(A)
70    IF INKEY$ = "" THEN 50
80    CLS
90 NEXT
```



# 5/5

# Test de la couleur d'un point en BASIC

Deux commandes BASIC permettent de connaître la couleur d'un point de l'écran. Il s'agit de TEST et TESTR.

### TEST (<Abscisse>, <Ordonnée>)

— <Abscisse> est compris entre 0 et 639,

— <Ordonnée> est compris entre 0 et 399.

Déplace le curseur aux coordonnées absolues spécifiées, et renvoie la couleur du point élémentaire situé à cet endroit.

### TESTR (<Déplacement en X>, <Déplacement en Y>)

Déplace le curseur aux coordonnées relatives (par rapport aux coordonnées courantes) spécifiées, et renvoie la couleur du point élémentaire situé à cet endroit.

# 5/6

# Tracé de points et de droites en assembleur

L'équivalent des ordres BASIC MOVE, MOVER, PLOT, PLOTR, DRAW et DRAWR existe dans les ROM du FIRMWARE. Reportez-vous à la partie 4 chap. 2.7 pour avoir plus de détails à ce sujet.

Pour utiliser les programmes élémentaires situés dans le FIRMWARE (couramment appelés routines), il suffit de charger les registres demandés en entrée, et d'appeler le programme élémentaire.

Par exemple, pour appeler la routine « MOVE ABSOLUTE » située en #BBC0, il faudra faire :

*En assembleur :*

```
LD      DE, 100  ;Abscisse X = 100
LD      HL, 120  ;Ordonnee Y = 120
CALL    #BBC0    ;Appel de MOVE ABSOLUTE
```

*En BASIC :*

```
1000 FOR I=0 TO 9
1010  READ A:POKE &9000+I,A
1020 NEXT I
1030 DATA &11, 100, 0, &21, 120, 0, &CD, &C0, &BB, &C9
1040 CALL &9000
```

ou encore PLOT 100, 120

Les routines équivalentes aux PLOT, MOVE et DRAW BASIC sont les suivantes :

MOVE absolu :

**CALL #BBC0**

avec, en entrée, DE qui contient l'abscisse absolue (entre 0 et 639), HL qui contient l'ordonnée absolue (entre 0 et 399).

MOVE relatif :

**CALL #BBC3**

avec, en entrée, DE qui contient le déplacement signé en abscisse, HL qui contient le déplacement signé en ordonnée.

PLOT absolu :

**CALL #BBEA**

avec, en entrée, DE qui contient l'abscisse absolue (entre 0 et 639), HL qui contient l'ordonnée absolue (entre 0 et 399).

PLOT relatif :

**CALL #BBED**

avec, en entrée, DE qui contient le déplacement signé en abscisse, HL qui contient le déplacement signé en ordonnée.

LINE absolu :

**CALL #BBF6**

avec, en entrée, DE qui contient l'abscisse absolue du point final, HL qui contient l'ordonnée absolue du point final.

LINE relatif :

**CALL #BBF9**

avec, en entrée, DE qui contient le déplacement signé en X du point final, HL qui contient le déplacement signé en Y du point final.

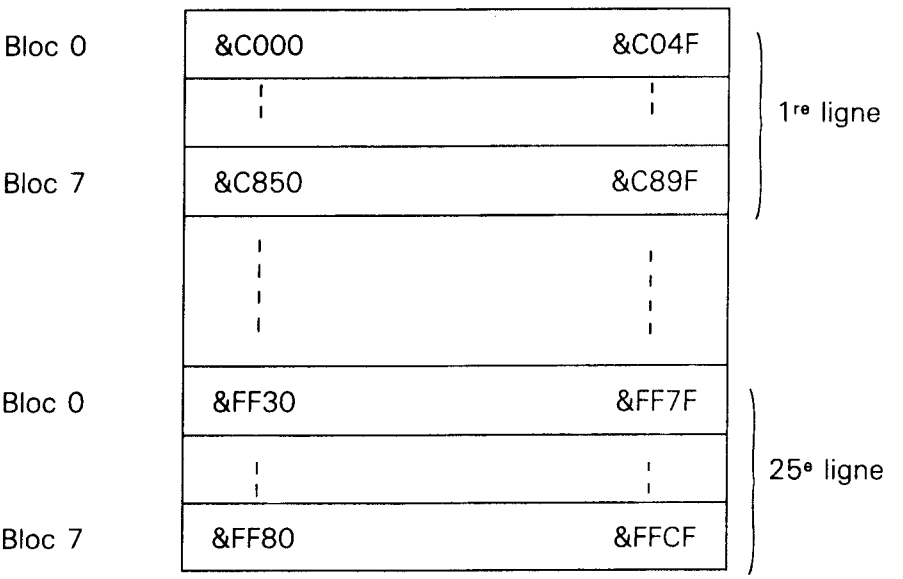
# 5/7

# La mémoire d'écran

Une autre manière d'utiliser l'écran graphique est d'écrire dans la mémoire d'écran.

L'écran peut être considéré comme une mémoire RAM implantée entre les adresses &C000 et &FFFF, et donc, directement adressable par le micro-processeur. Quel que soit le mode d'affichage, la mémoire d'écran est divisée en huit blocs. Une bloc représente une ligne élémentaire (d'épaisseur un point élémentaire, et de largeur 80 octets).

Les blocs sont répartis comme suit :




Le bloc i (i compris entre 0 et 7) représente la ième ligne élémentaire de chacune des 25 lignes de l'écran, et fait donc 80 × 25 = 2000 octets. Pour des raisons de commodité de manipulation, un bloc a été défini sur $2^{12}$ octets (2048 octets), et les 48 octets supplémentaires de chaque ligne sont inutilisés.

Comme nous l'avons vu plus haut, la dimension en pixels d'un point élémentaire dépend du mode de résolution.

*Dans le MODE 0*, un octet comprend 2 points élémentaires de 4 pixels de large chacun.

Les octets sont codés comme suit :

| Bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Stylo | 8 | 8 | 2 | 2 | 4 | 4 | 1 | 1 |
| Point élémentaire | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

Ce tableau est à interpréter de la façon suivante :

— Si, par exemple, le bit 2 est à 1, le point élémentaire 0 sera allumé avec la couleur de stylo 2.

— De même, pour avoir le point 1 allumé avec la couleur 6, il faudra positionner à 1 les bits 3 et 5.

*En MODE 1*, un octet comprend 4 points élémentaires de 2 pixels de large chacun.

Les octets sont codés comme suit :

| Bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Stylo | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 |
| Point élémentaire | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 |

Enfin, *en MODE 2*, un octet comprend 8 points élémentaires d'1 pixel de large chacun.

Les octets sont codés comme suit :

| Bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Stylo | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Point élémentaire | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

*En BASIC*, on pourra afficher un ou plusieurs points sur l'écran par l'instruction **POKE <Adresse écran>, <Valeur sur 8 bits>.**

*En assembleur*, il faudra faire :

```
LD    A, <Valeur>
LD    (Adresse ecran),A
```

# 5/8

# Caractères graphiques et signes spéciaux

Ils occupent les positions ASCII 123 à 255 et sont affichables au moyen de leur code ASCII.

Par exemple, le code de valeur ASCII 254 sera affichable en BASIC par PRINT CHR$(254).

En assembleur, on utilisera la routine du FIRMWARE GRA WR CHAR située en #BBFC de la manière suivante :

*En assembleur :*

```
LD      A, 254      ;Code du caractere a afficher
CALL    #BBFC       ;Affichage
```

*En BASIC :*

```
1000 FOR I=0 TO 5
1010 READ A:POKE &9000+I,A
1020 NEXT I
1030 DATA &3E, 254, &CD, &FC, &BB, &C9
1040 CALL &9000
```

Reportez-vous à l'annexe 3 Partie 11 où les caractères graphiques AMSTRAD sont donnés.

# 5/9

# Sprites : définition de caractères par l'utilisateur

Des caractères de la dimension d'un caractère normal (8 × 8 pixels) sont définissables par l'utilisateur, en BASIC ou en ASSEMBLEUR.

### 1) En BASIC

... Les ordres SYMBOL et SYMBOL AFTER permettent d'y arriver.

**SYMBOL AFTER [<entier>]**

où <entier> est compris entre 0 et 255.

Cet ordre fixe le nombre de caractères redéfinissables à 255-<entier>. Par défaut, si aucun argument n'est précisé, seize caractères redéfinissables seront permis. Ils seront accessibles par **CHR$(240)** à **CHR$(255)**.

*Remarque :*

Pour des raisons internes, la commande MEMORY interdit l'utilisation de SYMBOL AFTER (l'erreur « improper argument » sera générée si vous tentez d'utiliser SYMBOL AFTER après MEMORY).

**SYMBOL <Numéro du caractère>, <8 Données sur 8 bits>**

Le numéro du caractère sera compris entre 0 et 255 – n, où n est l'argument de la commande SYMBOL AFTER.

Le caractère redéfini de numéro i sera accessible par

**CHR$(255 – n + i)**

où n est l'argument de la commande SYMBOL AFTER.

A titre d'exemple, voici un petit programme écrit en BASIC qui vous permettra de définir vos propres caractères :

```
1000 REM Redefinition de caracteres
1010 CLS:DIM T(8, 8)
1020 FOR I=1 TO 8
1030   PRINT « ..... »
1040 NEXT I
1050 X=1 : Y=1
1060 LOCATE X,Y
1070 A$=INKEY$:IF A$="" " THEN 1070
1080 CA=ASC(A$)
1090 IF A$="P" THEN T(Y, X)=1:PRINT "*"
1100 IF A$="V" THEN T(Y, X)=0:PRINT"."
1110 IF CA=243 AND X<8 THEN X=X+1
1120 IF CA=242 AND X>1 THEN X=X-1
1130 IF CA=240 AND Y>1 THEN Y=Y-1
1140 IF CA=241 AND Y<8 THEN Y=Y+1
1150 IF CA<> 13 THEN 1060 'Boucle de saisie
1160 LOCATE 1,15 : PRINT "Donnees de la commande SYMBOL :
":PRINT
1170 FOR I=1 TO 8
1180   CA=0
1190   FOR J=1 TO 8
1200     CA=CA+(2^(8-J))*T(I, J)
1210   NEXT J
1220   PRINT CA;
1230 NEXT I
```

Les lignes 1010 à 1040 affichent la grille de redéfinition du caractère.

Les lignes 1050 à 1150 permettent de saisir le caractère par les commandes suivantes :

— les touches-flèches permettent de se déplacer dans la grille,

— la touche P permet d'allumer un pixel,

— la touche V permet d'éteindre un pixel.

Les lignes 1160 à 1230 affichent les données correspondant au caractère saisi qu'il faudra entrer dans la commande SYMBOL.

### 2) En ASSEMBLEUR

... Vous pouvez redéfinir des caractères grâce aux routines du FIRMWARE **TXT SET MATRIX** et **TXT SET M TABLE.**

**TXT SET M TABLE** est l'équivalent de SYMBOL AFTER.

Le registre pair DE doit contenir le premier caractère de la table (entre 0 et 255).

Le registre pair HL doit contenir l'adresse de départ de la table de redéfinition des caractères.

L'appel à TXT SET M TABLE se fait de la manière suivante :

```
LD    DE,<Numero du caractere>
LD    HL,<Adresse de la table>
CALL  #BBAB
```

**TXT SET MATRIX** est l'équivalent de SYMBOL.

Le registre A doit contenir le numéro du caractère à redéfinir. Le registre pair HL doit contenir l'adresse de la matrice où se trouvent les octets de redéfinition.

L'appel à TXT SET MATRIX se fait de la manière suivante :

```
LD    A,<Numero du caractere>
LD    HL,<Adresse de la matrice de redefinition>
CALL  #BBA8
```

L'adresse entrée dans HL doit pointer sur l'adresse mémoire du premier octet de redéfinition du caractère entré dans le registre A.

# 5/9.1

# Définition de caractères multiples

Grâce à ce programme, vous pourrez définir des caractères graphiques dans une grille de 6 × 10 caractères, soit 48 × 80 pixels.

Ce programme utilise les concepts manipulés par le précédent, et, en particulier les mêmes commandes.

```
1000 '=======================================
1010 ' Redefinition de caracteres multiples
1020 '=======================================
1030 '
1040 '----------------
1050 'Initialisation
1060 '----------------
1070 MODE 1
1080 DIM T(48,80)
1090 X=1:Y=1 'Position de depart
1100 '-----------------------------
1110 'Gestion du curseur graphique
1120 '-----------------------------
1130 LOCATE INT((X-1)/2)+1,INT((Y-1)/2)+1
1140 L$=INKEY$:IF L$="" THEN 1140
1150 L=ASC(L$)
1160 IF UPPER$(L$)="P" THEN T(Y,X)=1:CP=1:GOSUB 2000:PRINT CHR$(CA) 'Plein
1170 IF L$="V" THEN T(Y,X)=0:CP=0:GOSUB 2000:PRINT CHR$(CA) 'Vide
1180 IF L=243 AND X<80 THEN X=X+1
1190 IF L=242 AND X>1 THEN X=X-1
1200 IF L=241 AND Y<48 THEN Y=Y+1
1210 IF L=240 AND Y>1 THEN Y=Y-1
1220 IF L<>243 AND L<>242 AND L<>241 AND L<>240 AND L<>80 AND L<>86 AND L<>112 A
ND L<>118 AND L<>13 THEN PRINT CHR$(7) 'Action refus
ee
1230 IF L<>13 THEN 1130 'Boucle principale
1240 '-----------------------
1250 'Affichage des resultats
1260 '-----------------------
1270 CLS
1280 PRINT"1) Sur ecran"
1290 PRINT"2) Sur imprimante"
1300 PRINT:INPUT "Votre choix";R
1310 IF R<>1 AND R<>2 THEN 1300
1320 IF R=1 THEN C=1 ELSE C=8
1330 MODE 2
```

```
1340 FOR I=1 TO 6
1350   FOR J=1 TO 10
1360     PRINT#C:PRINT#C,"Ligne"I"Colonne"J":";
1370     FOR K=1 TO 8
1380       A=0
1390       FOR L=1 TO 8
1400         A=A+(2^(8-L))*T(K+(I-1)*8,L+(J-1)*8)
1410       NEXT L
1420       PRINT#C,A;
1430     NEXT K
1440   NEXT J
1450 NEXT I
1460 '
2000 '=========================
2010 'Zone des sous-programmes
2020 '=========================
2030 '
2040 '----------------------------------------
2050 'Carre plein ou Carre vide (CP=1 OU CP=0)
2060 '----------------------------------------
2070 S=0
2080 IF INT(X/2)<>X/2 AND INT (Y/2)<>Y/2 THEN GOSUB 2140
2090 IF INT(X/2)=X/2 AND INT (Y/2)<>Y/2 THEN GOSUB 2220
2100 IF INT(X/2)<>X/2 AND INT (Y/2)=Y/2 THEN GOSUB 2300
2110 IF INT(X/2)=X/2 AND INT (Y/2)=Y/2 THEN GOSUB 2380
2120 CA=128+S 'Caractere a afficher
2130 RETURN
2140 '----------------------------------------
2150 'Caractere en haut et a gauche
2160 '----------------------------------------
2170 IF CP=1 THEN S=S+1
2180 IF T(Y,X+1)=1 THEN S=S+2
2190 IF T(Y+1,X)=1 THEN S=S+4
2200 IF T(Y+1,X+1)=1 THEN S=S+8
2210 RETURN
2220 '----------------------------------------
2230 'Caractere en haut et a droite
2240 '----------------------------------------
2250 IF CP=1 THEN S=S+2
2260 IF T(Y,X-1)=1 THEN S=S+1
2270 IF T(Y+1,X-1)=1 THEN S=S+4
2280 IF T(Y+1,X)=1 THEN S=S+8
2290 RETURN
2300 '----------------------------------------
2310 'Caractere en bas et a gauche
2320 '----------------------------------------
2330 IF CP=1 THEN S=S+4
2340 IF T(Y-1,X)=1 THEN S=S+1
2350 IF T(Y-1,X+1)=1 THEN S=S+2
2360 IF T(Y,X+1)=1 THEN S=S+8
2370 RETURN
2380 '----------------------------------------
2390 'Caractere en bas et a droite
2400 '----------------------------------------
2410 IF CP=1 THEN S=S+8
2420 IF T(Y-1,X-1)=1 THEN S=S+1
2430 IF T(Y-1,X)=1 THEN S=S+2
2440 IF T(Y,X-1)=1 THEN S=S+4
2450 RETURN
```

Lignes 1000 à 1090 : Initialisation

Lignes 1100 à 1230 : Gestion du curseur graphique
(affichage ou effacement d'un pixel sur l'écran)

Lignes 1240 à 1460 : Affichage des données à programmer par SYMBOL sur écran ou imprimante

Lignes 2000 à 2450 : Zone des sous-programmes.
Calcul du motif à afficher en fonction de la position du point courant et du contexte précédent.

# 5/10

# Logiciels

Dans cette partie, nous allons étudier et donner des programmes utilitaires traitant de problèmes spécifiques au graphisme.

## 5/10.1

## Programme de dessin

Ce programme permet de créer d'une façon tout à fait conventionnelle des dessins en MODE 1 en utilisant la technique du **TÉLÉCRAN.**

Après avoir choisi les couleurs utilisées dans le dessin, les touches-flèches permettent de déplacer le curseur. Pour dessiner avec la couleur n, il suffit d'appuyer simultanément sur SHIFT et n. Pour obtenir la fonction « gomme », il faut appuyer simultanément sur SHIFT et 0 (car 0 est la couleur de fond).

Quand le dessin est terminé, appuyez sur la touche « ENTER » et répondez « O » (Oui) à la question « **Sauvegarde magnétique ?** ». Si vous possédez un 464 sans lecteur de disquettes, armez-vous de patience car l'écran occupe 16 KO (Kilo Octets) et prend (16/2 + 1) 9 blocs. Vous pouvez accélérer les choses en précisant « SPEED WRITE 1 » avant de lancer l'exécution du programme.

Une fois la sauvegarde effectuée, vous pouvez poursuivre votre travail

en reprenant les choses au point où elles en étaient en répondant « O » (Oui) à la question « Poursuite ? ».

Le listing du programme BASIC est le suivant :

```
1000 REM **********************
1001 REM Trace point par point
1002 REM **********************
1010 '
1020 'Prog. ASM Sauvegarde et affichage ecran
1030 '
1040 FOR I=0 TO &17:READ A:POKE &3000+I,A:NEXT
1050 DATA &21,0,&C0,&11,0,&40,1,&FF,&3F,&ED,&B0,&C9
1060 DATA &21,0,&40,&11,0,&C0,1,&FF,&3F,&ED,&B0,&C9
1070 '
1080 'Initialisation
1090 '
1100 STYLO=0 'Stylo leve
1110 BORDER 0:INK 0,0:INK 1,10:X=0:Y=0
1120 MODE 1
1130 INPUT"Affichage monochrome (O/N)";R$:R$=UPPER$(R$)
1140 IF R$<>"O" AND R$<>"N" THEN 1120
1150 IF R$="O" THEN NBCOUL=1 ELSE NBCOUL=3
1160 PRINT
1170 FOR I=0 TO NBCOUL
1180    PRINT"INK";I;": (0 A 26) ";:INPUT A:INK I,A
1190 NEXT I
1200 PRINT :INPUT"Chargement d'un ecran (O/N) ";R$:R$=UPPER$(R$)
1210 IF R$<>"O" AND R$<>"N" THEN 1200
1220 IF R$="O" THEN PRINT:INPUT"Nom de l'ecran ";ECR$:LOAD ECR$,&C000:GOTO 1240
1230 CLS
1240 PLOT X,Y,2
1250 '
1260 'Boucle principale
```

```
1270 '
1280 A$=INKEY$:IF A$="" THEN 1280 'Attente action
1290 A=ASC(A$)
1300 IF A=55 THEN PLOT X,Y,STYLO:Y=Y+2:X=X-2:GOTO 1240 'En haut a gauche
1310 IF A=57 THEN PLOT X,Y,STYLO:Y=Y+2:X=X+2:GOTO 1240 'En haut a droite
1320 IF A=49 THEN PLOT X,Y,STYLO:Y=Y-2:X=X-2:GOTO 1240 'En bas a gauche
1330 IF A=51 THEN PLOT X,Y,STYLO:Y=Y-2:X=X+2:GOTO 1240 'En bas a droite
1340 IF A=56 THEN PLOT X,Y,STYLO:Y=Y+2:GOTO 1240 'Vers le haut
1350 IF A=50 THEN PLOT X,Y,STYLO:Y=Y-2:GOTO 1240 'Vers le bas
1360 IF A=52 THEN PLOT X,Y,STYLO:X=X-2:GOTO 1240 'Vers la gauche
1370 IF A=54 THEN PLOT X,Y,STYLO:X=X+2:GOTO 1240 'Vers la droite
1380 IF A=95 OR (A>32 AND A<33+NBCOUL) THEN 1420 'Changement de stylo
1390 IF A=13 THEN 1470 'Fin de trace
1400 GOTO 1280 'Boucle de trace
1410 '-----------------------------------------------
1420 REM Changement de couleur stylo
1430 '
1440 IF A=95 THEN STYLO=0 ELSE STYLO=A-32
1450 GOTO 1240
1460 '-----------------------------------------------
1470 REM Fin de trace
1480 '
1490 PLOT X,Y,0:CALL &3000 'Sauvegarde ecran
1500 CLS
1510 LOCATE 1,10:INPUT"Sauvegarde magnetique (O/N) ";R$:R$=UPPER$(R$)
1520 IF R$<>"O" AND R$<>"N" THEN 1510
1530 IF R$="N" THEN CALL &300C:GOTO 1240 'Restitution ecran
1540 LOCATE 1,12:INPUT"Nom de l'ecran ";N$
1550 SAVE N$,B,&4000,&3FFF
1560 LOCATE 1,14:INPUT"Poursuite (O/N) ";R$:R$=UPPER$(R$)
1570 IF R$="O" THEN CALL &300C:GOTO 1240 'Restitution ecran
1580 END
```

Lignes 1010 à 1060 : Chargement des sous-programmes ASSEMBLEUR.

Lignes 1070 à 1230 : Initialisation du programme.

Lignes 1240 : Affichage du curseur graphique.

Lignes 1280 : Lecture du clavier.

Lignes 1290 à 1400 : Action en fonction de la touche pressée.

Ligne 1440 : Changement de la couleur du tracé.

Lignes 1470 à 1580 : Fin de tracé.

Ligne 1550 : avec sauvegarde magnétique.

Ligne 1530 à 1570 : et/ou retour au tracé.

Le programme BASIC défini ci-dessus utilise deux sous-programmes écrits en ASSEMBLEUR. Ces sous-programmes permettent :

1°) de sauvegarder l'écran en mémoire RAM
(transfert du bloc #C000 à #FFFF en #4000),

2°) de restituer la sauvegarde RAM sur l'écran
(transfert du bloc #4000 à #7FFF en #C000).

Ces deux sous-programmes utilisent une mnémonique qui a fait la renommée du Z80 : il s'agit de « LDR ». Son utilisation est très simple : avant l'appel à LDIR, il faut charger les registres HL, DE et BC avec les valeurs suivantes :

HL = Adresse de la première mémoire à déplacer,

DE = Adresse de la première mémoire où va s'effectuer le déplacement,

BC = Longueur du bloc déplacé.

Ce qui donne lieu aux programmes suivants :

```
1                       ORG  3000H
2                       LOAD 3000H
3               ;----------------------
4               ;Sauvegarde de l'ecran
5               ;----------------------
6  3000 2100C0          LD   HL,0C000H      ;@ Depart
7  3003 110040          LD   DE,4000H       ;@ Sauvegarde
8  3006 01FF3F          LD   BC,3FFFH       ;Longueur
9  3009 EDB0            LDIR                ;Transfert
```

```
10 300B C9                    RET
11                  ;----------------------
12                  ;Restitution de l'ecran
13                  ;----------------------
14 300C 210040                LD   HL,4000H          ;@ Depart
15 300F 1100C0                LD   DE,0C000H         ;@ Ecran
16 3012 01FF3F                LD   BC,3FFFH          ;Longueur
17 3015 EDB0                  LDIR                   ;Transfert
18 3017 C9                    RET
19                            END
```

# 5/10.2

# Utilitaires de manipulation de dessins

Lors de la création d'un écran graphique, vous pouvez être confronté au problème suivant : *un objet doit être représenté plusieurs fois sur la même image. Selon la complexité de l'objet, sa duplication peut prendre de quelques minutes à quelques heures.*

Pour éviter cette tâche rébarbative, nous vous proposons trois utilitaires de copie d'objets :

— le premier reproduit un bloc rectangulaire que l'on a défini par ses extrémités bas-gauche et haut-droite ;

— les deuxième et troisième permettent d'obtenir un effet de miroir respectivement par rapport à un axe vertical et par rapport à un axe horizontal.

## 5/10.2.1

## Reproduction de blocs graphiques

Après avoir entré les noms d'écran (avant et après duplication) et les couleurs du dessin, un curseur graphique apparaît en bas et à gauche de l'écran. Dirigez-le avec les touches du bloc numérique pour l'amener en bas et à gauche du rectangle délimitant le bloc à recopier. Appuyez ensuite sur la touche « ENTER ». Déplacez à nouveau le curseur pour l'amener en haut et à droite du même rectangle. Appuyez sur la touche « ENTER ». Un rectangle apparaît. Si l'encadrement est correct (Ques-

tion « OK ? », Réponse « O » (Oui)) déplacez le curseur pour l'amener en bas et à gauche de l'endroit où le bloc doit être dupliqué. Appuyez sur « ENTER ». La copie s'effectue instantanément. A ce niveau, vous pouvez appuyer sur :

« R » pour revenir à l'écran précédent,

« S » pour obtenir une sauvegarde magnétique,

une autre touche pour poursuivre la duplication.

Le programme BASIC de duplication est le suivant :

```
1000 REM Recopie d'un objet sur l'ecran
1010 REM ================================================================
1020 REM 1) Determiner le coin INF GAUCHE de l'objet a deplacer
1030 REM 2) Determiner le coin SUP DROIT  de l'objet a deplacer
1040 REM 3) L'objet est encadre
1050 REM 4) Repondre "O" si l'encadrement est correct,"N" sinon
1060 REM 5) Determiner le coin superieur gauche de la nouvelle position
1070 REM 6) L'objet apparait a sa nouvelle position
1080 REM ================================================================
1090 INK 0,0:BORDER 0:INK 1,10:INK 3,10,6:PEN 1:MODE 1:C=1 'Initialisation
1100 '
1110 'Prog. ASM Sauvegarde et affichage ecran
1120 '
1130 FOR I=0 TO &17:READ A:POKE &9000+I,A:NEXT
1140 DATA &21,0,&C0,&11,0,&40,1,&FF,&3F,&ED,&B0,&C9
1150 DATA &21,0,&40,&11,0,&C0,1,&FF,&3F,&ED,&B0,&C9
1160 '
1170 'Prog ASM Interface entre MBG et ABG
1180 '
1190 FOR I=0 TO 28:READ A:POKE &9018+I,A:NEXT
1200 DATA 1,&64,0,&11,&64,0,&26,&A,&2E,&A,&3E,0,&CD,&35,&90,&C9
1210 DATA 1,&64,0,&11,&64,0,&21,0,&80,&CD,&A2,&90,&C9
1220 '
1230 'Programme MBG (Memorisation de blocs graphiques)
1240 '
1250 FOR I=0 TO 108:READ A:POKE &9035+I,A:NEXT
```

```
1260 DATA &18,&B,&73,&20,&58,&20,&61,&75,&20,&64,&65,&70,&61,&ED,&43,&37,&30,&ED
,&53,&39,&30,&32,&3D,&30,&7C,&32,&3B,&30,&7D,&32,&3C

,&30,&CD,&3,&B9,&21,&0,&80,&11,&94,&2,&3A,&3D,&30,&47,&B7,&28,&4,&ED,&5A,&10,&FC
,&3A,&3B,&30,&77,&23,&22,&3E,&30,&ED,&5B

1270 DATA &37,&30,&2A,&39,&30,&CD,&1D,&BC,&22,&40,&30,&ED,&5B,&3E,&30,&3A,&3B,&3
0,&47,&7E,&12,&23,&13,&10,&FA,&2A,&40,&30,&CD,&26,&B

C,&22,&40,&30,&3A,&3C,&30,&3D,&32,&3C,&30,&20,&E4,&3E,&CF,&12,&C9

1280 '

1290 'Programme ABG (Affichage de blocs graphiques)

1300 '

1310 FOR I=0 TO 72:READ A:POKE &90A2+I,A:NEXT

1320 DATA&18,&A,&41,&42,&47,&D,&A,&3B,&D,&A,&4F,&52,&ED,&43,&A4,&30,&ED,&53,&A6,
&30,&7E,&32,&AA,&30,&23,&22,&A8,&30,&CD,&3,&B9,&ED,&

5B,&A4,&30,&2A,&A6,&30,&CD,&1D,&BC,& 22,&AC,&30,&ED,&5B,&A8,&30,&3A,&AA,&30,&47,
&1A,&77,&23,&13,&10,&FA,&2A,&AC,&30,&CD,&26

1330 DATA &BC,&22,&AC,&30,&1A,&FE,&CF,&20,&E8,&C9

1340 '

1350 PRINT"         COPIE DE BLOCS GRAPHIQUES":PRINT:PRINT

1360 LOCATE 1,6:INPUT"Affichage monochrome (O/N)";R$:R$=UPPER$(R$)

1370 IF R$<>"O" AND R$<>"N" THEN 1360

1371 IF R$="O" THEN NBCOUL=1 ELSE NBCOUL=3

1380 PRINT

1390 FOR I=0 TO NBCOUL

1400    PRINT"INK";I;": (0 A 26) ";:INPUT A:INK I,A

1410 NEXT I

1420 PRINT:PRINT

1421 INPUT"Nom de l'ecran a charger ";N$

1430 INPUT"Nom de l'ecran a sauver  ";N2$

1440 PRINT:PRINT"Une fois toutes les modifications faites, appuyez sur 'S' pour
obtenir une sau- -vegarde de l'ecran sur K7 ou disqu

ette"

1450 PRINT"Appuyez sur 'R' pour retourner a l'ecran precedent et sur une autre t
ouche pour continuer le deplacement des blocs."

1460 PRINT:PEN 3:PRINT"          Appuyez sur une touche"

1470 a$=INKEY$:IF a$="" THEN 1470

1480 LOAD N$,&C000

1490 CALL &9000 'Sauvegarde ecran charge
```

```
1500 B=TEST(X,Y) 'Memo. du pt ou se trouve le curseur
1510 PLOT X,Y,3
1520 '
1530 'Boucle principale
1540 '
1550 A$=INKEY$:IF A$="" THEN 1550 'Attente action
1560 A=ASC(A$)
1570 IF A=55 THEN PLOT X,Y,B:Y=Y+2:X=X-2:GOTO 1500 'En haut a gauche
1580 IF A=57 THEN PLOT X,Y,B:Y=Y+2:X=X+2:GOTO 1500 'En haut a droite
1590 IF A=49 THEN PLOT X,Y,B:Y=Y-2:X=X-2:GOTO 1500 'En bas a gauche
1600 IF A=51 THEN PLOT X,Y,B:Y=Y-2:X=X+2:GOTO 1500 'En bas a droite
1610 IF A=56 THEN PLOT X,Y,B:Y=Y+2:GOTO 1500 'Vers le haut
1620 IF A=50 THEN PLOT X,Y,B:Y=Y-2:GOTO 1500 'Vers le bas
1630 IF A=52 THEN PLOT X,Y,B:X=X-2:GOTO 1500 'Vers la gauche
1640 IF A=54 THEN PLOT X,Y,B:X=X+2:GOTO 1500 'Vers la droite
1650 IF A=13 THEN 1670 'Appui sur ENTER
1660 GOTO 1510
1670 REM Appui sur ENTER
1680 ON C GOTO 1690,1710,1800
1690 'Memo coin superieur gauche
1700 P1=X:P2=Y:C=2:PLOT X,Y,3:GOTO 1500
1710 'Memo coin inferieur droit et trace rectangle
1720 P3=X:P4=Y
1730 PLOT P1,P2:DRAW X,P2:DRAW X,Y:DRAW P1,Y:DRAW P1,P2
1740 IF X>P1 THEN SX=X ELSE SX=P1 'SX=SUP(X,P1)
1750 IF Y>P2 THEN SY=Y ELSE SY=P2 'SY=SUP(Y,P2)
1760 LOCATE ABS(SX/15+2),ABS(26-SY/15):INPUT "OK (O/N) ";R$:R$=UPPER$(R$)
1770 IF R$<>"O" AND R$<>"N" THEN 1760
1780 IF R$="N" THEN CALL &900C:C=1:X=0:Y=0:GOTO 1550
1790 C=3:GOTO 1550
1800 'Tranfert a la position indiquee
1810 'Memo bloc graphique
1820 CALL &900C 'Restitution ecran initial
```

```
1830 A=P1/2:GOSUB 1970:POKE &9019,B:POKE &901A,C
1840 A=P4/2:GOSUB 1970:POKE &901C,B:POKE &901D,C
1850 POKE &901F,INT(ABS((P3-P1)/8))+1:POKE &9021,ABS((P4-P2)/2)+1
1860 CALL &9018 'MBG
1870 'Transfert
1880 A=X/2:GOSUB 1970:POKE &9029,B:POKE &902A,C
1890 A=Y/2:GOSUB 1970:POKE &902C,B:POKE &902D,C
1900 CALL &9028 'ABG
1910 C=1:X=0:Y=0 'Repositionnement curseur
1920 A$=INKEY$:IF A$="" THEN 1920
1930 A$=UPPER$(A$) 'Conversion majuscule
1940 IF A$="R" THEN CALL &900C:GOTO 1500 'Retour a l'ecran precedent
1950 IF A$="S" THEN SAVE N2$,B,&C000,&3FFF
1960 GOTO 1490 'Nouvelle intervention sur le dessin
1970 'Extraction MSB/LSB d'un nombre 16 bits
1980 C=INT(A/256):B=A-C*256
1990 RETURN
```

Lignes 1110 à 1330 : Chargement des sous-programmes ASSEMBLEUR.

Lignes 1350 à 1470 : Initialisation du programme.

Ligne 1480 : Chargement de l'image.

Ligne 1490 : Sauvegarde de l'image pour permettre l'option « R » par la suite.

Lignes 1550 à 1660 : Gestion du curseur.

Lignes 1670 à 1790 : Mémorisation des positions du curseur suite à l'appui sur la touche « ENTER ».

Lignes 1820 à 1900 : Duplication.

Lignes 1910 à 1960 : Attente d'une action au clavier (R, S ou autre).

Le programme BASIC défini ci-dessus utilise des programmes écrits en **ASSEMBLEUR.**

Les sous-programmes de mémorisation et de restitution d'écrans décrits précédemment sont repris ici. De plus, pour permettre les sauvegarde et restitution de portions d'écrans de tailles plus modestes définies par l'utilisateur, deux sous-programmes supplémentaires sont nécessaires : nous les appelerons **MBG** (Mémorisation de Blocs Graphiques) et **ABG** (Affichage de Blocs Graphiques).

**MBG :**

Avant de décrire la structure du sous-programme MBG, il est nécessaire de faire un rappel sur la structure de l'écran.

L'écran peut être considéré comme une mémoire RAM implantée entre les adresses #C000 et #FFFF.

Reportez-vous à la partie 5/7 pour avoir le détail du codage de l'écran au niveau pixel.

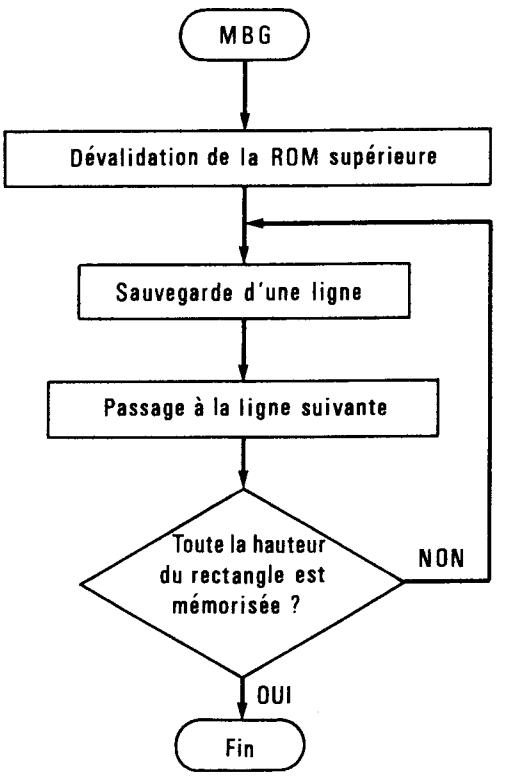
Le problème suivant se pose :

*Soit un rectangle de dimensions quelconques dessiné sur l'écran. Comment mémoriser le dessin qui s'y trouve inscrit ?*

Vu la complexité structurale de la mémoire d'écran, les constructeurs des CPC ont eu le tact d'agrémenter le FIRMWARE de la routine « DOTPOS » qui, lorsqu'on lui fournit des coordonnées d'écran absolues X et Y, est capable de donner l'adresse de l'octet correspondant en mémoire RAM.

Partant de cette constatation, notre tâche est grandement simplifiée. Connaissant la largeur du rectangle, et vu qu'un octet contient quatre pixels en MODE 1, il sera très simple de sauver une ligne élémentaire. Sachant quelle est la hauteur du rectangle et munis de la routine DOTPOS, le problème devient enfantin. Cependant, précisons un autre petit détail : il ne faut pas oublier de dévalider la ROM supérieure qui occupe également les octets #C000 à #FFFF.

La logique de MBG suit donc l'organigramme suivant :



Voici le listing ASSEMBLEUR correspondant occupant les lignes 1250 à 1270 dans le programme BASIC de recopie d'objets.

```
1                ;
2                ;Memorisateur de blocs graphiques
3                ;
4                ;Entree: BC=Pos X au depart
5                ;        DE=Pos Y au depart
6                ;        H =Largeur en octets
7                ;        L =Hauteur en pixels
8                ;        A =No de bloc a memoriser
9                ;Sortie: Memorisation du bloc
10               ;Pt d'entree: MBG
11               ;
12                            ORG  9C6FH
13                            LOAD 9C6FH
14               MBG:         EQU  $              ;Point d'entree
15 9C6F 180B                  JR   MBGDP          ;Debut du programme
16               ;
17               ;Variables locales
18               ;
19               PX:          DS   2              ;Pos Abs X au depart
20               PY:          DS   2              ;Pos Abs Y au depart
21               LA:          DS   1              ;Largeur d'image(octets)
22               HA:          DS   1              ;Hauteur d'image(pixels)
23               NB:          DS   1              ;No de bloc a memoriser
24               AS:          DS   2              ;@ Sauvegarde bloc
25               SAD:         DS   2              ;Sauvegarde @ mem ecran
26               APS:         EQU  7000H          ;@ 1ere Sauvegarde
27               DOTPOS:      EQU  0BC1DH
28               NTLINE:      EQU  0BC26H
29               ;
```

```
30                      ;Initialisation
31                      ;
32                      MBGDP:      EQU  $
33 9C7C ED43719C                    LD   (PX),BC        ;Pos X au depart
34 9C80 ED53739C                    LD   (PY),DE        ;Pos Y au depart
35 9C84 32779C                      LD   (NB),A         ;No de bloc
36 9C87 7C                          LD   A,H
37 9C88 32759C                      LD   (LA),A         ;Largeur
38 9C8B 7D                          LD   A,L
39 9C8C 32769C                      LD   (HA),A         ;Hauteur
40 9C8F CD03B9                      CALL 0B903H         ;UROM DIS
41                      ;
42 9C92 210070                      LD   HL,APS         ;1ere sauvegarde
43 9C95 119402                      LD   DE,294H        ;Espace entre 2 sauveg.
44 9C98 3A779C                      LD   A,(NB)
45 9C9B 47                          LD   B,A
46 9C9C B7                          OR   A
47 9C9D 2804                        JR   Z,MBGOO
48                      MBGO:       EQU  $              ;Cacul @ Sauvegarde
49 9C9F ED5A                        ADC  HL,DE
50 9CA1 10FC                        DJNZ MBGO
51                      MBGOO:      EQU  $
52 9CA3 3A759C                      LD   A,(LA)
53 9CA6 77                          LD   (HL),A
54 9CA7 23                          INC  HL             ;Largeur en octets
55 9CA8 22789C                      LD   (AS),HL        ;@ Sauvegarde
56                      ;
57 9CAB ED5B719C                    LD   DE,(PX)
58 9CAF 2A739C                      LD   HL,(PY)
59 9CB2 CD1DBC                      CALL DOTPOS         ;OUT HL=@ Mem ecran
60 9CB5 227A9C                      LD   (SAD),HL       ;Sauv mem ecran
```

```
61 9CB8 ED5E789C          LD   DE,(AS)

62              ;
63              MBG1:     EQU  $
64 9CBC 3A759C            LD   A,(LA)
65 9CBF 47                LD   B,A
66              MBG2:     EQU  $
67 9CC0 7E                LD   A,(HL)          ;Mem ecran
68 9CC1 12                LD   (DE),A          ;Mem sauvegarde
69 9CC2 23                INC  HL
70 9CC3 13                INC  DE
71 9CC4 10FA              DJNZ MBG2
72 9CC6 2A7A9C            LD   HL,(SAD)
73 9CC9 CD26BC            CALL NTLINE
74 9CCC 227A9C            LD   (SAD),HL
75 9CCF 3A769C            LD   A,(HA)          ;Hauteur
76 9CD2 3D                DEC  A
77 9CD3 32769C            LD   (HA),A
78 9CD6 20E4              JR   NZ,MBG1
79 9CD8 3ECF              LD   A,0CFH
80 9CDA 12                LD   (DE),A          ;Terminateur
81 9CDB C9                RET
82                        END                  ;Zi eind arf arf arf...
```

**ABG :**

Puisque nous savons sauvegarder en mémoire un objet situé sur l'écran, nous pouvons penser qu'il est simple d'afficher un objet sauvegardé en RAM à une position quelconque sur l'écran. La démarche est effectivement simple : elle consiste à faire l'opposé de la démarche précédente, à savoir :

1°) Lecture d'une ligne élémentaire en mémoire,

2°) Affichage de la ligne sur l'écran,

3°) Passage à la ligne suivante,

4°) Si toute la hauteur du rectangle n'est pas parcourue, retour en 1.

D'où le listing du sous-programme ABG dont les données hexadécimales occupent les lignes 1310 à 1330 dans le programme BASIC de recopie d'objet.

```
1                ;
2                ; Afficheur de blocs graphiques
3                ;
4                ; Entree: BC=Pos X depart
5                ;         DE=Pos Y depart
6                ;         HL=@ RAM depart
7                ; Sortie: Affichage du bloc
8                ; Pt d'entree: ABG
9                ;
10                         ORG  9CDCH
11                         LOAD 9CDCH
12               ABG:      EQU  $              ;Point d'entree
13 9CDC 180A               JR   ABGDP          ;Debut programme
14               ;
15               ;Variables locales
16               ;
17               PX:       DS   2              ;Pos abs X au deaprt
18               PY:       DS   2              ;Pos abs Y au depart
19               MI:       DS   2              ;@ de depart RAM image
20               LA:       DS   2              ;Largeur image
21               SAD:      DS   2              ;Sauveg @ mem ecran
22               DOTPOS:   EQU  0BC1DH
23               NTLINE:   EQU  0BC26H
24               ;
25               ;Initialisation
26               ;
27               ABGDP:    EQU  $
28 9CE8 ED43DE9C           LD   (PX),BC        ;Pos X depart
29 9CEC ED53E09C           LD   (PY),DE        ;Pos Y depart
```

```
30 9CF0 7E                  LD    A,(HL)
31 9CF1 32E49C              LD    (LA),A          ;Largeur image
32 9CF4 23                  INC   HL
33 9CF5 22E29C              LD    (MI),HL         ;@ Mem image
34 9CF8 CD03B9              CALL  0B903H          ;UROM DIS
35                ;
36 9CFB ED5BDE9C            LD    DE,(PX)
37 9CFF 2AE09C              LD    HL,(PY)
38 9D02 CD1DBC              CALL  DOTPOS          ;OUT HL=@ Mem ecran
39 9D05 22E69C              LD    (SAD),HL        ;Sauveg @ mem ecran
40 9D08 ED5BE29C            LD    DE,(MI)
41              ABGO:       EQU   $
42 9D0C 3AE49C              LD    A,(LA)
43 9D0F 47                  LD    B,A             ;Largeur image
44              ABG1:       EQU   $
45 9D10 1A                  LD    A,(DE)
46 9D11 77                  LD    (HL),A
47 9D12 23                  INC   HL
48 9D13 13                  INC   DE
49 9D14 10FA                DJNZ  ABG1            ;Affichage 1 ligne elem.
50 9D16 2AE69C              LD    HL,(SAD)
51 9D19 CD26BC              CALL  NTLINE          ;Next line
52 9D1C 22E69C              LD    (SAD),HL
53 9D1F 1A                  LD    A,(DE)
54 9D20 FECF                CP    0CFH            ;Fin?
55 9D22 20E8                JR    NZ,ABGO         ;Non
56 9D24 C9                  RET                   ;Zi einde.
57                          END
```

*Remarque :*

Pour faciliter l'interfaçage entre le programme BASIC de recopie d'objets et MBG/ABG, un sous-programme interface a été écrit pour MBG et un autre pour ABG.

Le premier donne les informations nécessaires à MBG dans les registres A, BC, DE et HL :

BC est l'abcisse absolue du rectangle source en haut et à gauche.
DE est l'ordonnée absolue du rectangle source en haut et à gauche.

H est la largeur en octets (1 octet = 4 pixels) du rectangle.
L est la hauteur en pixels du rectangle.
A donne le numéro de bloc à sauvegarder (0 si un seul bloc est sauvegardé à la fois. C'est le cas ici.)

Le deuxième donne les informations nécessaires à ABG dans les registres BC, DE et HL :

BC est l'abscisse absolue du rectangle à recopier en haut et à gauche.
DE est l'ordonnée absolue du rectangle à recopier en haut et à gauche.
HL est l'adresse en RAM où se trouve le pavé à recopier (largeur et hauteur sont contenues dans le bloc de mémoire rempli par MBG : il est donc inutile de les préciser ici).

```
1                       ORG  3018H
2                       LOAD 3018H
3               MBG:    EQU  3035H          ;Memo bloc graph
4               ABG:    EQU  30A2H          ;Affich bloc graph
5               ;------------------
6               ;Interface avec MBG
7               ;------------------
8 3018 016400           LD   BC,100         ;X en haut a gauche
9 301B 116400           LD   DE,100         ;Y en haut a gauche
10 301E 260A            LD   H,10           ;Largeur en octets
11 3020 2E0A            LD   L,10           ;Hauteur en pixels
12 3022 3E00            LD   A,0            ;Numero de bloc
13 3024 CD3530          CALL MBG
14 3027 C9              RET
15              ;------------------
16              ;Interface avec ABG
17              ;------------------
18 3028 016400          LD   BC,100         ;X en haut a gauche
19 302B 116400          LD   DE,100         ;Y en haut a gauche
20 302E 2100B0          LD   HL,0B000H      ;@ RAM bloc
21 3031 CDA230          CALL ABG
22 3034 C9              RET
```

# 5/10.2.2

# Miroir par rapport à un axe vertical

*Supposons qu'un dessin soit inscrit dans un rectangle.*

**Quelle démarche utiliser pour reproduire le contenu du rectangle réfléchi dans un miroir vertical ?**

Pour chaque ligne élémentaire haute d'un pixel , il suffit de prendre les pixels successifs et de les inverser (le plus à gauche va à l'extrême droite et le plus à droite va à l'extrême gauche). Pour simplifier l'approche du problème, nous vous présentons ici un programme de symétrisation écrit en BASIC.

Le listing du programme est le suivant :

```
1000 REM Symetrie d'un objet sur l'ecran %0y
1010 REM ============================================================
1020 REM 1) Determiner le coin INF GAUCHE de l'objet a deplacer
1030 REM 2) Determiner le coin SUP DROIT  de l'objet a deplacer
1040 REM 3) L'objet est encadre
1050 REM 4) Repondre "O" si l'encadrement est correct,"N" sinon
1060 REM 5) Determiner le coin INFERIEUR GAUCHE de la nouvelle position
1070 REM 6) L'objet symetrique %0y apparait a sa nouvelle position
1080 REM ============================================================
1090 INK 0,0:BORDER 0:INK 1,10:INK 3,10,6:PEN 1:MODE 1:C=1 'Initialisation
1100 '
1110 'Prog. ASM Sauvegarde et affichage ecran
1120 '
1130 FOR I=0 TO &17:READ A:POKE &9000+I,A:NEXT
1140 DATA &21,0,&C0,&11,0,&40,1,&FF,&3F,&ED,&B0,&C9
1150 DATA &21,0,&40,&11,0,&C0,1,&FF,&3F,&ED,&B0,&C9
1160 '
1170 PRINT"      SYMETRIE DE BLOCS GRAPHIQUES":PRINT:PRINT
```

```
1180 LOCATE 1,6:INPUT"Affichage monochrome (O/N)";R$:R$=UPPER$(R$)
1190 IF R$<>"O" AND R$<>"N" THEN 1180
1200 IF R$="O" THEN NBCOUL=1 ELSE NBCOUL=3
1210 PRINT
1220 FOR I=0 TO NBCOUL
1230    PRINT"INK";I;": (0 A 26) ";:INPUT A:INK I,A
1240 NEXT I
1250 PRINT:PRINT
1260 INPUT"Nom de l'ecran a charger ";N$
1270 INPUT"Nom de l'ecran a sauver  ";N2$
1280 PRINT:PRINT"Une fois toutes les modifications faites, appuyez sur 'S' pour
obtenir une sau- -vegarde de l'ecran sur K7 ou disqu
ette"
1290 PRINT"Appuyez sur 'R' pour retourner a l'ecran precedent et sur une autre t
ouche pour continuer la symetrisation."
1300 PRINT:PEN 3:PRINT"          Appuyez sur une touche"
1310 a$=INKEY$:IF a$="" THEN 1310
1320 LOAD N$,&C000
1330 CALL &9000 'Sauvegarde ecran charge
1340 B=TEST(X,Y) 'Memo. du pt ou se trouve le curseur
1350 PLOT X,Y,3
1360 '
1370 'Boucle principale
1380 '
1390 A$=INKEY$:IF A$="" THEN 1390 'Attente action
1400 A=ASC(A$)
1410 IF A=55 THEN PLOT X,Y,B:Y=Y+2:X=X-2:GOTO 1340 'En haut a gauche
1420 IF A=57 THEN PLOT X,Y,B:Y=Y+2:X=X+2:GOTO 1340 'En haut a droite
1430 IF A=49 THEN PLOT X,Y,B:Y=Y-2:X=X-2:GOTO 1340 'En bas a gauche
1440 IF A=51 THEN PLOT X,Y,B:Y=Y-2:X=X+2:GOTO 1340 'En bas a droite
1450 IF A=56 THEN PLOT X,Y,B:Y=Y+2:GOTO 1340 'Vers le haut
1460 IF A=50 THEN PLOT X,Y,B:Y=Y-2:GOTO 1340 'Vers le bas
1470 IF A=52 THEN PLOT X,Y,B:X=X-2:GOTO 1340 'Vers la gauche
```

```
1480 IF A=54 THEN PLOT X,Y,B:X=X+2:GOTO 1340 'Vers la droite
1490 IF A=13 THEN 1510 'Appui sur ENTER
1500 GOTO 1350
1510 REM Appui sur ENTER
1520 ON C GOTO 1530,1550,1640
1530 'Memo coin superieur gauche
1540 P1=X:P2=Y:C=2:PLOT X,Y,3:GOTO 1340
1550 'Memo coin inferieur droit et trace rectangle
1560 P3=X:P4=Y
1570 PLOT P1;P2:DRAW X,P2:DRAW X,Y:DRAW P1,Y:DRAW P1,P2
1580 IF X>P1 THEN SX=X ELSE SX=P1 'SX=SUP(X,P1)
1590 IF Y>P2 THEN SY=Y ELSE SY=P2 'SY=SUP(Y,P2)
1600 LOCATE ABS(SX/15+2),ABS(26-SY/15):INPUT "OK (O/N) ";R$:R$=UPPER$(R$)
1610 IF R$<>"O" AND R$<>"N" THEN 1600
1620 IF R$="N" THEN CALL &900C:C=1:X=0:Y=0:GOTO 1390
1630 C=3:GOTO 1390
1640 'Symetrie %0y
1650 CALL &900C 'Effacement cadre
1660 XF=X+P3-P1:C=0 'Initialisation
1670 FOR I=P2 TO P4 STEP 2
1680   B=0:C=C+2
1690   FOR J=P1 TO P3 STEP 2
1700     B=B+2:A=TEST(J,I):PLOT XF-B,Y+C,A
1710   NEXT J
1720 NEXT I
1730 '
1740 C=1:X=0:Y=0 'Repositionnement curseur
1750 A$=INKEY$:IF A$="" THEN 1750
1760 A$=UPPER$(A$) 'Conversion majuscule
1770 IF A$="R" THEN CALL &900C:GOTO 1340  Retour a l'ecran precedent
1780 IF A$="S" THEN SAVE N2$,B,&C000,&3FFF
1790 GOTO 1330 'Nouvelle intervention sur le dessin
```

Lignes 1130 à 1150 : Chargement des sous-programmes assembleur.

Lignes 1170 à 1310 : Initialisation du programme.

Lignes 1320 : Chargement d'une image.

Lignes 1330 : Sauvegarde pour permettre l'option « R » par la suite.

Lignes 1390 à 1500 : Gestion du curseur.

Lignes 1510 à 1590 : Mémorisation des positions du curseur suite à un appui sur la touche « ENTER ».

Lignes 1640 à 1720 : Fonction miroir vertical.

Lignes 1740 à 1790 : Attente d'une action au clavier (R, S ou autre).

Les sous-programmes ASSEMBLEUR utilisés sont les sous-programmes de mémorisation et de restitution de mémoire d'écran décrits au chap. 10.1 de cette partie.

# 5/10.2.3

# Miroir par rapport à un axe horizontal

Le problème est le même que celui exposé au chapitre 10.2.2 de cette partie mais le miroir est vertical.

*Supposons qu'un dessin soit inscrit dans un rectangle.*

**Quelle démarche utiliser pour reproduire le contenu du rectangle réfléchi dans un miroir horizontal ?**

Appelons « rectangle source » le rectangle qui va être réfléchi et « rectangle image » le rectangle réfléchi. Si le rectangle source comporte n lignes élémentaires (en hauteur), il suffit de le parcourir et de faire :

Ligne n – i du rectangle image = Ligne i du rectangle source pour i variant entre 1 et n – 1.

Comme précédemment, pour simplifier l'approche du problème, nous vous proposons un programme écrit en BASIC.

Les sous-programmes écrits en ASSEMBLEUR sont les mêmes que ceux du programme précédent.

Le listing du programme est le suivant :

```
1000 REM Symetrie d'un objet sur l'ecran
1010 REM ================================================================
1020 REM 1) Determiner le coin INF GAUCHE de l'objet a deplacer
1030 REM 2) Déterminer le coin SUP DROIT  de l'objet a deplacer
1040 REM 3) L'objet est encadre
1050 REM 4) Repondre "O" si l'encadrement est correct,"N" sinon
1060 REM 5) Determiner le coin SUPERIEUR GAUCHE de la nouvelle position
1070 REM 6) L'objet symetrique %0y apparait a sa nouvelle position
1080 REM ================================================================
1090 INK 0,0:BORDER 0:INK 1,10:INK 3,10,6:PEN 1:MODE 1:C=1 'Initialisation
1100 '
1110 'Prog. ASM Sauvegarde et affichage ecran
```

```
1120 '
1130 FOR I=0 TO &17:READ A:POKE &9000+I,A:NEXT
1140 DATA &21,0,&C0,&11,0,&40,1,&FF,&3F,&ED,&B0,&C9
1150 DATA &21,0,&40,&11,0,&C0,1,&FF,&3F,&ED,&B0,&C9
1160 '
1170 PRINT"      SYMETRIE DE BLOCS GRAPHIQUES":PRINT:PRINT
1180 LOCATE 1,6:INPUT"Affichage monochrome (O/N)";R$:R$=UPPER$(R$)
1190 IF R$<>"O" AND R$<>"N" THEN 1180
1200 IF R$="O" THEN NBCOUL=1 ELSE NBCOUL=3
1210 PRINT
1220 FOR I=0 TO NBCOUL
1230     PRINT"INK";I;": (0 A 26) ";:INPUT A:INK I,A
1240 NEXT I
1250 PRINT:PRINT
1260 INPUT"Nom de l'ecran a charger ";N$
1270 INPUT"Nom de l'ecran a sauver  ";N2$
1280 PRINT:PRINT"Une fois toutes les modifications faites, appuyez sur 'S' pour
obtenir une sau- -vegarde de l'ecran sur K7 ou disqu
e"
1290 PRINT"Appuyez sur 'R' pour retourner a l'ecran precedent et sur une autre t
ouche pour continuer la symetrisation."
1300 PRINT:PEN 3:PRINT"          Appuyez sur une touche"
1310 a$=INKEY$:IF a$="" THEN 1310
1320 LOAD N$,&C000
1330 CALL &9000 'Sauvegarde ecran charge
1340 B=TEST(X,Y) 'Memo. du pt ou se trouve le curseur
1350 PLOT X,Y,3
1360 '
1370 'Boucle principale
1380 '
1390 A$=INKEY$:IF A$="" THEN 1390 'Attente action
1400 A=ASC(A$)
1410 IF A=55 THEN PLOT X,Y,B:Y=Y+2:X=X-2:GOTO 1340 'En haut a gauche
1420 IF A=57 THEN PLOT X,Y,B:Y=Y+2:X=X+2:GOTO 1340 'En haut a droite
```

```
1480 IF A=54 THEN PLOT X,Y,B:X=X+2:GOTO 1340 'Vers la droite
1490 IF A=13 THEN 1510 'Appui sur ENTER
1500 GOTO 1350
1510 REM Appui sur ENTER
1520 ON C GOTO 1530,1550,1640
1530 'Memo coin superieur gauche
1540 P1=X:P2=Y:C=2:PLOT X,Y,3:GOTO 1340
1550 'Memo coin inferieur droit et trace rectangle
1560 P3=X:P4=Y
1570 PLOT P1;P2:DRAW X,P2:DRAW X,Y:DRAW P1,Y:DRAW P1,P2
1580 IF X>P1 THEN SX=X ELSE SX=P1 'SX=SUP(X,P1)
1590 IF Y>P2 THEN SY=Y ELSE SY=P2 'SY=SUP(Y,P2)
1600 LOCATE ABS(SX/15+2),ABS(26-SY/15):INPUT "OK (O/N) ";R$:R$=UPPER$(R$)
1610 IF R$<>"O" AND R$<>"N" THEN 1600
1620 IF R$="N" THEN CALL &900C:C=1:X=0:Y=0:GOTO 1390
1630 C=3:GOTO 1390
1640 'Symetrie %0y
1650 CALL &900C 'Effacement cadre
1660 XF=X+P3-P1:C=0 'Initialisation
1670 FOR I=P2 TO P4 STEP 2
1680   B=0:C=C+2
1690   FOR J=P1 TO P3 STEP 2
1700     B=B+2:A=TEST(J,I):PLOT XF-B,Y+C,A
1710   NEXT J
1720 NEXT I
1730 '
1740 C=1:X=0:Y=0 'Repositionnement curseur
1750 A$=INKEY$:IF A$="" THEN 1750
1760 A$=UPPER$(A$) 'Conversion majuscule
1770 IF A$="R" THEN CALL &900C:GOTO 1340  Retour a l'ecran precedent
1780 IF A$="S" THEN SAVE N2$,B,&C000,&3FFF
1790 GOTO 1330 'Nouvelle intervention sur le dessin
```

Lignes 1130 à 1150 : Chargement des sous-programmes assembleur.

Lignes 1170 à 1310 : Initialisation du programme.

Lignes 1320 : Chargement d'une image.

Lignes 1330 : Sauvegarde pour permettre l'option « R » par la suite.

Lignes 1390 à 1500 : Gestion du curseur.

Lignes 1510 à 1590 : Mémorisation des positions du curseur suite à un appui sur la touche « ENTER ».

Lignes 1640 à 1720 : Fonction miroir vertical.

Lignes 1740 à 1790 : Attente d'une action au clavier (R, S ou autre).

Les sous-programmes ASSEMBLEUR utilisés sont les sous-programmes de mémorisation et de restitution de mémoire d'écran décrits au chap. 10.1 de cette partie.

# 5/10.2.2

# Miroir par rapport à un axe vertical

*Supposons qu'un dessin soit inscrit dans un rectangle.*

**Quelle démarche utiliser pour reproduire le contenu du rectangle réfléchi dans un miroir vertical ?**

Pour chaque ligne élémentaire haute d'un pixel , il suffit de prendre les pixels successifs et de les inverser (le plus à gauche va à l'extrême droite et le plus à droite va à l'extrême gauche). Pour simplifier l'approche du problème, nous vous présentons ici un programme de symétrisation écrit en BASIC.

Le listing du programme est le suivant :

```
1000 REM Symetrie d'un objet sur l'ecran %0y
1010 REM ============================================================
1020 REM 1) Determiner le coin INF GAUCHE de l'objet a deplacer
1030 REM 2) Determiner le coin SUP DROIT  de l'objet a deplacer
1040 REM 3) L'objet est encadre
1050 REM 4) Repondre "O" si l'encadrement est correct,"N" sinon
1060 REM 5) Determiner le coin INFERIEUR GAUCHE de la nouvelle position
1070 REM 6) L'objet symetrique %0y apparait a sa nouvelle position
1080 REM ============================================================
1090 INK 0,0:BORDER 0:INK 1,10:INK 3,10,6:PEN 1:MODE 1:C=1 'Initialisation
1100 '
1110 'Prog. ASM Sauvegarde et affichage ecran
1120 '
1130 FOR I=0 TO &17:READ A:POKE &9000+I,A:NEXT
1140 DATA &21,0,&C0,&11,0,&40,1,&FF,&3F,&ED,&B0,&C9
1150 DATA &21,0,&40,&11,0,&C0,1,&FF,&3F,&ED,&B0,&C9
1160 '
1170 PRINT"      SYMETRIE DE BLOCS GRAPHIQUES":PRINT:PRINT
```

```
1180 LOCATE 1,6:INPUT"Affichage monochrome (O/N)";R$:R$=UPPER$(R$)
1190 IF R$<>"O" AND R$<>"N" THEN 1180
1200 IF R$="O" THEN NBCOUL=1 ELSE NBCOUL=3
1210 PRINT
1220 FOR I=0 TO NBCOUL
1230    PRINT"INK";I;": (0 A 26) ";:INPUT A:INK I,A
1240 NEXT I
1250 PRINT:PRINT
1260 INPUT"Nom de l'ecran a charger ";N$
1270 INPUT"Nom de l'ecran a sauver  ";N2$
1280 PRINT:PRINT"Une fois toutes les modifications faites, appuyez sur 'S' pour
obtenir une sau- -vegarde de l'ecran sur K7 ou disqu
ette"
1290 PRINT"Appuyez sur 'R' pour retourner a l'ecran precedent et sur une autre t
ouche pour continuer la symetrisation."
1300 PRINT:PEN 3:PRINT"          Appuyez sur une touche"
1310 a$=INKEY$:IF a$="" THEN 1310
1320 LOAD N$,&C000
1330 CALL &9000 'Sauvegarde ecran charge
1340 B=TEST(X,Y) 'Memo. du pt ou se trouve le curseur
1350 PLOT X,Y,3
1360 '
1370 'Boucle principale
1380 '
1390 A$=INKEY$:IF A$="" THEN 1390 'Attente action
1400 A=ASC(A$)
1410 IF A=55 THEN PLOT X,Y,B:Y=Y+2:X=X-2:GOTO 1340 'En haut a gauche
1420 IF A=57 THEN PLOT X,Y,B:Y=Y+2:X=X+2:GOTO 1340 'En haut a droite
1430 IF A=49 THEN PLOT X,Y,B:Y=Y-2:X=X-2:GOTO 1340 'En bas a gauche
1440 IF A=51 THEN PLOT X,Y,B:Y=Y-2:X=X+2:GOTO 1340 'En bas a droite
1450 IF A=56 THEN PLOT X,Y,B:Y=Y+2:GOTO 1340 'Vers le haut
1460 IF A=50 THEN PLOT X,Y,B:Y=Y-2:GOTO 1340 'Vers le bas
1470 IF A=52 THEN PLOT X,Y,B:X=X-2:GOTO 1340 'Vers la gauche
```

```
1430 IF A=49 THEN PLOT X,Y,B:Y=Y-2:X=X-2:GOTO 1340 'En bas a gauche
1440 IF A=51 THEN PLOT X,Y,B:Y=Y-2:X=X+2:GOTO 1340 'En bas a droite
1450 IF A=56 THEN PLOT X,Y,B:Y=Y+2:GOTO 1340 'Vers le haut
1460 IF A=50 THEN PLOT X,Y,B:Y=Y-2:GOTO 1340 'Vers le bas
1470 IF A=52 THEN PLOT X,Y,B:X=X-2:GOTO 1340 'Vers la gauche
1480 IF A=54 THEN PLOT X,Y,B:X=X+2:GOTO 1340 'Vers la droite
1490 IF A=13 THEN 1510 'Appui sur ENTER
1500 GOTO 1350
1510 REM Appui sur ENTER
1520 ON C GOTO 1530,1550,1640
1530 'Memo coin superieur gauche
1540 P1=X:P2=Y:C=2:PLOT X,Y,3:GOTO 1340
1550 'Memo coin inferieur droit et trace rectangle
1560 P3=X:P4=Y
1570 PLOT P1,P2:DRAW X,P2:DRAW X,Y:DRAW P1,Y:DRAW P1,P2
1580 IF X>P1 THEN SX=X ELSE SX=P1 'SX=SUP(X,P1)
1590 IF Y>P2 THEN SY=Y ELSE SY=P2 'SY=SUP(Y,P2)
1600 LOCATE SX/15+2,26-SY/15:INPUT "OK (O/N) ";R$:R$=UPPER$(R$)
1610 IF R$<>"O" AND R$<>"N" THEN 1600
1620 IF R$="N" THEN CALL &900C:C=1:X=0:Y=0:GOTO 1390
1630 C=3:GOTO 1390
1640 'Symetrie %0x
1650 CALL &900C 'Effacement cadre
1660 C=0 'Initialisation
1670 FOR I=P2 TO P4 STEP 2
1680   B=0:C=C+2
1690   FOR J=P1 TO P3 STEP 2
1700     B=B+2:A=TEST(J,I):PLOT X+B,Y-C,A
1710   NEXT J
1720 NEXT I
1730 '
1740 C=1:X=0:Y=0 'Repositionnement curseur
```

```
1750 A$=INKEY$:IF A$="" THEN 1750
1760 A$=UPPER$(A$) 'Conversion majuscule
1770 IF A$="R" THEN CALL &900C:GOTO 1340 'Retour a l'ecran precedent
1780 IF A$="S" THEN SAVE N2$,B,&C000,&3FFF
1790 GOTO 1330 'Nouvelle intervention sur le dessin
```

Lignes 1130 à 1150 : Chargement des sous-programmes assembleur.

Lignes 1170 à 1310 : Initialisation du programme.

Ligne 1320 : Chargement d'une image.

Ligne 1330 : Sauvegarde pour permettre l'option « R » par la suite.

Lignes 1390 à 1500 : Gestion du curseur.

Lignes 1510 à 1590 : Mémorisation des positions du curseur suite à un appui sur la touche « ENTER ».

Lignes 1640 à 1720 : Fonction miroir vertical.

Lignes 1740 à 1790 : Attente d'une action au clavier (R, S ou autre).

**Remarque sur l'utilisation du programme :**

Le troisième appui sur la touche « ENTER » doit se faire lorsque le curseur est positionné en haut et à gauche du rectangle image (et non pas en bas et à gauche comme dans le programme précédent).

# 5/10.3

# Utilitaires de compactage

Devant le peu de mémoire RAM disponible sur les CPC 464, 664 ou même 6128, il nous semble judicieux d'étudier le fonctionnement de routines simples de compactage graphique.

Ces routines vous permettront :

— de stocker plusieurs images en mémoire RAM sans avoir à faire de nombreux accès disquette ou cassette ;

— d'afficher rapidement les images compactées.

Sachant qu'un écran occupe 16 kilo-octets de mémoire RAM, il sera possible de stocker 2 ou 3 écrans sur CPC 464 et 664 et 5 ou 6 sur CPC 6128. Si vous voulez (par exemple) créer des jeux d'aventures, il vous faudra faire des accès disque pour charger chaque nouvel écran, à moins que vous n'utilisiez un décompacteur/afficheur.

Nous étudierons trois types de compactage/décompactage/affichage. Chacun s'applique à des cas très particuliers pour être efficace, et le lecteur pourra créer de nouvelles routines de compactage en partant des propos développés ici.

# 5/10.3.1

# Compactage filiforme

Nous allons étudier :

1) Le principe de compactage d'objets filiformes.

2) Le principe de décompactage et d'affichage de fichiers filiformes.

## I. Le compacteur

Les dessins monochromes définis par leurs contours (dessins du genre « fil de fer ») pourront être énormément compactés (jusqu'à un facteur 20 !) par le programme qui va suivre.

Le procédé de compactage est très simple. Il consiste en :

— la définition d'un point quelconque appartenant au dessin,

— le codage de la direction dans laquelle il faut se déplacer pour atteindre le point suivant.

Le programme de compactage filiforme défini est écrit en BASIC et occupe les lignes 1200 à 1450.

Entrez le nom de l'écran à compacter. Cet écran aura été créé par le programme de tracé défini au chapitre 10.1 de la partie 5.

Déplacez ensuite le curseur graphique (grâce aux touches-flèches) jusqu'à rencontrer un point allumé sur le dessin.

Appuyez sur la touche « ENTER ». Le programme trace alors d'une autre couleur (PEN 2) le contour de la forme jusqu'à aboutir à une discontinuité. Arrivé à ce point, le tracé s'arrête. Il faut alors déplacer le curseur pour « sauter » la discontinuité. Dès que le curseur se trouve sur un autre point (voisin) du dessin, appuyez sur la touche « ENTER », et ainsi de suite jusqu'à ce que tout l'objet ait changé de couleur.

Appuyez alors deux fois sur la touche « ESC ». Le programme indique la place occupée par le fichier compacté et propose une sauvegarde magnétique ou un retour au compactage (appui sur « ENTER »).

Le programme de compactage est le suivant :

```
1000 REM ********************
1010 REM Codage de formes
1020 REM ********************
1030 '
1040 'Initialisation
1050 '
1060 'Prog. ASM Sauvegarde et affichage ecran
1070 FOR I=0 TO &17:READ A:POKE &2F00+I,A:NEXT
1080 DATA &21,0,&C0,&11,0,&40,1,&FF,&3F,&ED,&B0,&C9
1090 DATA &21,0,&40,&11,0,&C0,1,&FF,&3F,&ED,&B0,&C9
1100 '
1110 ON BREAK GOSUB 1670 'Sortie du programme
1120 INK 0,0:INK 1,10:INK 3,6,25:BORDER 0:MODE 1
1130 INPUT "Nom de l'ecran a coder ";N$:LOAD N$,&C000
1140 W=10:AG=&3000 'Adresse graphique
1150 GOTO 1490 'Positionnement en debut de Forme
1160 AG=&3000:V1=INT(X/256):V2=X-V1*256:V3=INT(Y)/256:V4=Y-V3*256:POKE AG,V1:POK
E AG+1,V2:POKE AG+2,V3:POKE AG+3,V4:AG=&3003 'Entete
1170 '
1180 ' Codage de la forme en memoire
1190 '
1200 mx=x:my=y
1210 PLOT X,Y,2
1220 W=9 'Init du calcul
1230 IF TEST(X+2,Y)=1 THEN U=X+2:V=Y:D=0:GOSUB 1430
1240 IF TEST(X+2,Y+2)=1 THEN U=X+2:V=Y+2:D=1:GOSUB 1430
1250 IF TEST(X,Y+2)=1 THEN U=X:V=Y+2:D=2:GOSUB 1430
1260 IF TEST(X-2,Y+2)=1 THEN U=X-2:V=Y+2:D=3:GOSUB 1430
1270 IF TEST(X-2,Y)=1 THEN U=X-2:V=Y:D=4:GOSUB 1430
1280 IF TEST(X-2,Y-2)=1 THEN U=X-2:V=Y-2:D=5:GOSUB 1430
1290 IF TEST(X,Y-2)=1 THEN U=X:V=Y-2:D=6:GOSUB 1430
1300 IF TEST(X+2,Y-2)=1 THEN U=X+2:V=Y-2:D=7:GOSUB 1430
1310 IF W=9 THEN 1490
```

```
1330 IF G=1 THEN G=0 ELSE G=1
1340 IF G=1 THEN OCT=16*wD ELSE OCT=OCT+wD:AG=AG+1:POKE AG,OCT
1350 IF W=0 THEN PLOT x,y,2:GOTO 1370 'Saut de plume
1360 GOTO 1210
1370 IF G=1 THEN OCT=OCT OR &80 ELSE OCT=OCT OR 8
1380 IF g=1 THEN ag=ag+1
1390 POKE AG,OCT:GOTO 1490
1400 '
1410 'Calcul optimal du prochain point
1420 '
1430 W1=-(TEST(U+2,V)=1)-(TEST(U+2,V+2)=1)-(TEST(U,V+2)=1)-(TEST(U-2,V+2)=1)-(TE
ST(U-2,V)=1)-(TEST(U-2,V-2)=1)-(TEST(U,V-2)=1)-(TEST
(U+2,V-2)=1)
1440 IF W1<W THEN W=W1:WX=U:WY=V:WD=D
1450 RETURN
1460 '
1470 'Positionnement du curseur
1480 '
1490 WX=0:WY=0
1500 IF PT1=1 THEN AG=AG+1:POKE AG,&88 'Point unique
1510 cou=TEST(x,y):PLOT x,y,3
1520 A$=INKEY$:IF A$="" THEN 1520
1530 A=ASC(A$)
1540 PLOT X,Y,COU
1550 IF A=240 THEN Y=Y+2:WY=WY+2
1560 IF A=241 THEN Y=Y-2:WY=WY-2
1570 IF A=242 THEN X=X-2:WX=WX-2
1580 IF A=243 THEN X=X+2:WX=WX+2
1590 IF a<>13 THEN 1510
1600 IF WX =0 THEN AX=WX/2 ELSE AX=(-WX/2) OR &80
1610 IF WY =0 THEN AY=WY/2 ELSE AY=(-WY/2) OR &80
1620 AG=AG+1:POKE AG,AX:AG=AG+1:POKE AG,AY:g=0:ag=ag+1
1630 PT1=1:GOTO 1200
```

```
1640 '
1650 'Sortie du programme
1660 '
1670 CALL &2F00 'Sauvegarde ecran
1680 CLS:PRINT"La forme occupe la memoire situee"
1690 PRINT"entre &3000 et ";HEX$(AG+1);"."
1700 PRINT:INPUT"Nom de la sauvegarde (ou ENTER) ";R$
1710 IF LEN(R$)=0 THEN CALL &2F0C:RETURN 'Pas de sauvegarde
1720 POKE AG+1,&FF:SAVE R$,B,&3000,AG+2-&3000 'Sauvegarde
1730 END
```

Lignes 1070 à 1090 : Chargement des sous-programmes Assembleur

Lignes 1110 à 1160: Initialisation du programme

Lignes 1200 à 1390: Codage de la forme

Lignes 1430 à 1450: Calcul optimal du prochain point

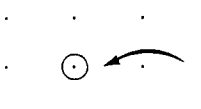
Lignes 1490 à 1630: Positionnement du curseur graphique sur le prochain départ

Ligne 1710 : Retour au compactage si appui sur ENTER

Ligne 1720 : ou sortie avec sauvegarde du fichier compacté

Une technique intéressante employée dans ce programme est *la recherche du prochain point à allumer en créant le moins possible de discontinuités*. Cette technique est basée sur le principe suivant :

Tout point de l'écran peut être entouré de huit façons différentes :



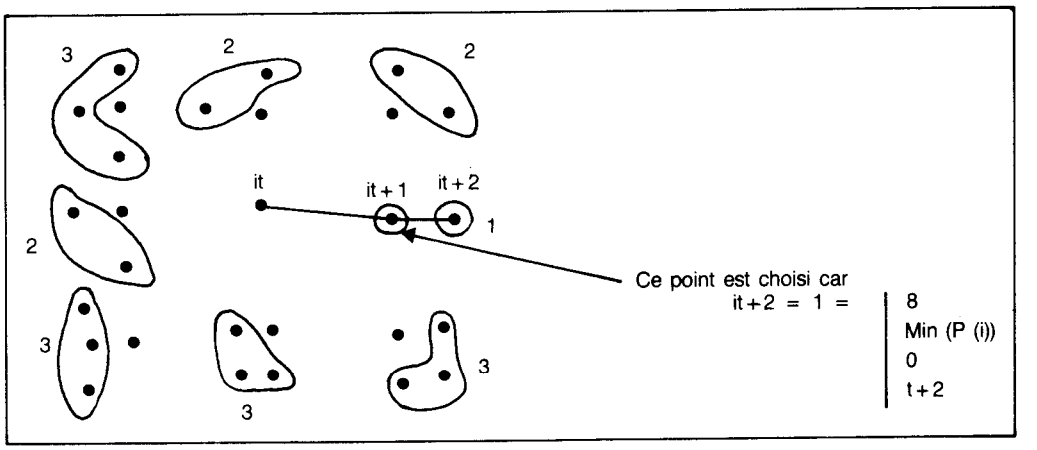
un point quelconque de l'écran

Appelons « t » le contexte actuel, « t+1 » le contexte après un déplacement, et « t+n » le contexte après n déplacements.

Si, pour chacun de ces huit points, nous calculons le nombre de points immédiatement contigus P(i) (et donc le nombre de déplacements possibles en t+2), il apparaît que :

$$\underset{i=0}{\overset{8}{\text{Min}}}\ (P(i))$$

donnera le point i ayant le moins de chance de provoquer une discontinuité (Calcul de Min (P(i)) effectué ligne 1440, et calcul de P(i) effectué ligne 1430).

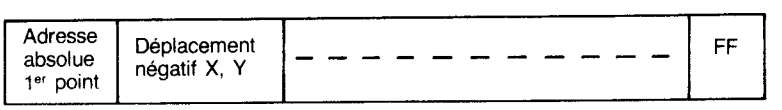


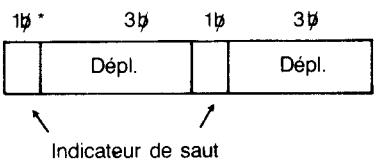

*Remarque importante :*

La forme devra OBLIGATOIREMENT être dessinée en PEN 1. Référez-vous aux lignes 1230 à 1300 du listing qui explique le pourquoi de la chose : le calcul du prochain point est effectué en comparant le résultat de la fonction TEST (qui donne la couleur d'un point) à 1.

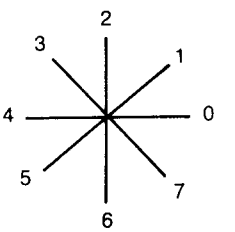
Le programme défini ci-dessus utilise des sous-programmes écrits en langage d'assemblage pour l'affichage des dessins filiformes.

L'utilisation du langage d'assemblage est quasi obligatoire pour réduire le temps d'affichage qui, malgré tout, n'est pas négligeable (une seconde pour 2 000 points). Le principe est simple. Le fichier compacté généré par le programme précédent possède la structure suivante :

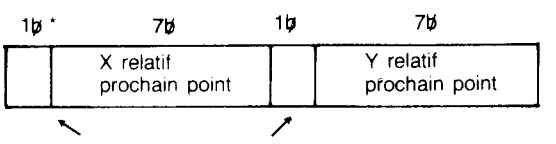



Avec la convention de codage de déplacement suivante :



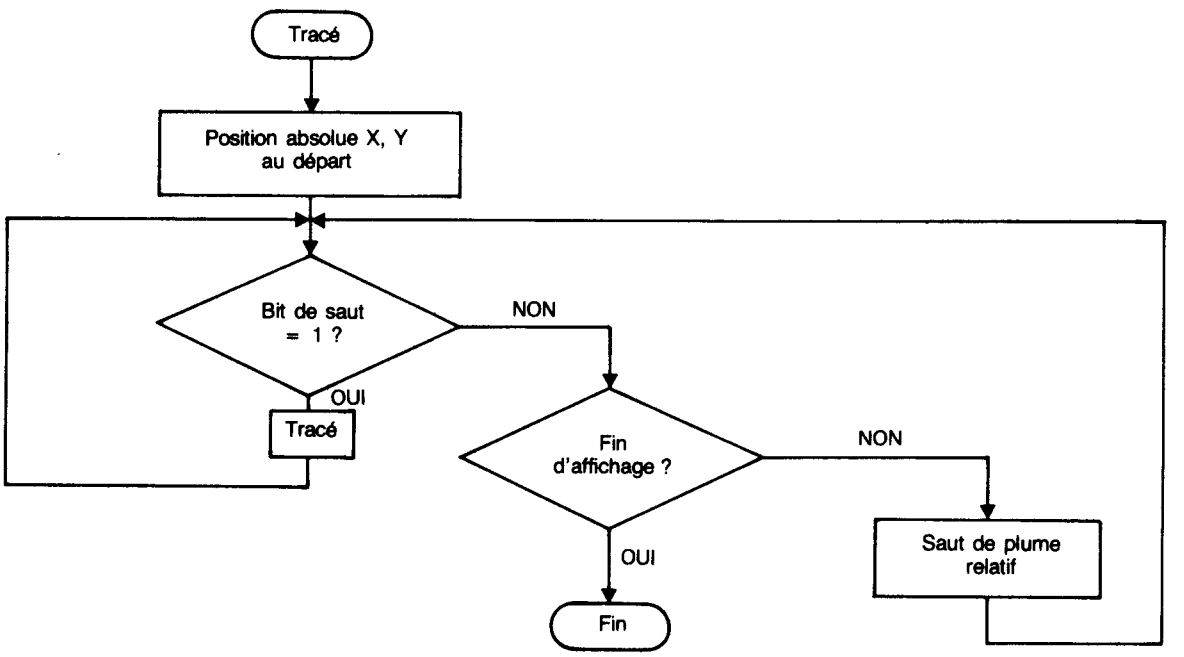

Si un des bits indicateur de saut est à un, un déplacement relatif sur 7 bits suit :




0 = vers la droite ou vers le haut
1 = vers la gauche ou vers le bas

* Remarque : ø est l'abréviation de bit

Partant de la structure de ce fichier, voyons comment écrire le programme de tracé.




d'où le programme d'assemblage suivant :

```
1                ;
2                ; Afficheur de formes 4 bits
3                ;
4                ; Entree: HL=à de la forme
5                ; Sortie: Tts registres ecrases
6                ; Pt d'entree: AF0
7                ;
8                          ORG  9000H
9                          LOAD 9000H
10               ;
11               FORM:     DS   2              ;à Forme
12               RT:       DS   1
13               TSAUT:    EQU  $
```

```
14 9003 D090                DW   DIR0
15 9005 D890                DW   DIR1
16 9007 E090                DW   DIR2
17 9009 E890                DW   DIR3
18 900B F090                DW   DIR4
19 900D F890                DW   DIR5
20 900F 0091                DW   DIR6
21 9011 0891                DW   DIR7
22                  ;
23                  AFO:    EQU  $                ;Point d'entree
24 9013 220090              LD   (FORM),HL        ;Sauvegarde à forme
25 9016 FD2A0090            LD   IY,(FORM)
26 901A FD5600              LD   D,(IY+0)
27 901D FD5E01              LD   E,(IY+1)
28 9020 FD6602              LD   H,(IY+2)
29 9023 FD6E03              LD   L,(IY+3)
30 9026 CDEABB              CALL OBBEAH           ;Plot absolu
31                  ;
32 9029 DD2A0090            LD   IX,(FORM)
33 902D DD23                INC  IX
34 902F DD23                INC  IX
35 9031 DD23                INC  IX
36 9033 DD23                INC  IX               ;IX=à 1er octet forme
37                  ;
38                  AFOOO:  EQU  $
39 9035 DD7E00              LD   A,(IX+0)
40                  AFOO:   EQU  $
41 9038 FE88                CP   88H
42 903A 2804                JR   Z,AFO1
43 903C E670                AND  70H
44 903E CB2F                SRA  A
45 9040 CB2F                SRA  A
46 9042 CB2F                SRA  A
```

```
47 9044 CB2F                SRA  A
48 9046 87                  ADD  A,A
49 9047 87                  ADD  A,A
50 9048 87                  ADD  A,A              ;Quartet fort X 8
51 9049 2A0390              LD   HL,(TSAUT)
52 904C 0600                LD   B,0
53 904E 4F                  LD   C,A
54 904F 09                  ADD  HL,BC
55 9050 3E00                LD   A,0
56 9052 320290              LD   (RT),A           ;Memo retour
57 9055 E9                  JP   (HL)             ;Trait quartet fort
58              AF02:       EQU  $
59 9056 DD7E00              LD   A,(IX+0)
60 9059 CB7F                BIT  7,A
61 905B 2020                JR   NZ,AF01          ;Saut de plume
62 905D E607                AND  7
63 905F 87                  ADD  A,A
64 9060 87                  ADD  A,A
65 9061 87                  ADD  A,A              ;Quartet faible X 8
66 9062 2A0390              LD   HL,(TSAUT)
67 9065 0600                LD   B,0
68 9067 4F                  LD   C,A
69 9068 09                  ADD  HL,BC
70 9069 3E01                LD   A,1
71 906B 320290              LD   (RT),A           ;Memo retour
72 906E E9                  JP   (HL)             ;Trait quartet faible
73              ;
74              AF03:       EQU  $
75 906F DD7E00              LD   A,(IX+0)
76 9072 CB5F                BIT  3,A
77 9074 2007                JR   NZ,AF01          ;Saut de plume
78 9076 DD23                INC  IX
```

```
 79 9079 DD7E00             LD   A,(IX+0)
 80 907B 18BB               JR   AF00
 81                ;
 82           AF01:         EQU  $
 83 907D DD23               INC  IX
 84 907F DD7E00             LD   A,(IX+0)
 85 9082 FEFF               CP   0FFH
 86 9084 C8                 RET  Z
 87 9085 CB7F               BIT  7,A
 88 9087 200B               JR   NZ,AF011          ;X negatif
 89 9089 6F                 LD   L,A
 90 908A 2600               LD   H,0
 91 908C 37                 SCF
 92 908D 3F                 CCF
 93 908E ED6A               ADC  HL,HL
 94 9090 54                 LD   D,H
 95 9091 5D                 LD   E,L               ;Sauvegarde
 96 9092 1812               JR   AF012
 97           AF011:        EQU  $
 98 9094 E67F               AND  7FH
 99 9096 6F                 LD   L,A
100 9097 2600               LD   H,0
101 9099 37                 SCF
102 909A 3F                 CCF
103 909B ED6A               ADC  HL,HL
104 909D 54                 LD   D,H
105 909E 5D                 LD   E,L
106 909F 210000             LD   HL,0
107 90A2 ED52               SBC  HL,DE
108 90A4 54                 LD   D,H
109 90A5 5D                 LD   E,L               ;Sauvegarde depl. X rel
110
```

```
111                 AF012:      EQU  $
112 90A6 DD7E01                 LD   A,(IX+1)
113 90A9 CB7F                   BIT  7,A
114 90AB 2009                   JR   NZ,AF013          ;Y negatif
115 90AD 6F                     LD   L,A
116 90AE 2600                   LD   H,0
117 90B0 37                     SCF
118 90B1 3F                     CCF
119 90B2 ED6A                   ADC  HL,HL
120 90B4 1810                   JR   AF014
121                 AF013:      EQU  $
122 90B6 E67F                   AND  7FH
123 90B8 6F                     LD   L,A
124 90B9 2600                   LD   H,0
125 90BB 37                     SCF
126 90BC 3F                     CCF
127 90BD ED6A                   ADC  HL,HL
128 90BF 44                     LD   B,H
129 90C0 4D                     LD   C,L
130 90C1 210000                 LD   HL,0
131 90C4 ED42                   SBC  HL,BC
132                 AF014:      EQU  $
133 90C6 CDEDBB                 CALL 0BBEDH            ;Plot relatif
134 90C9 DD23                   INC  IX
135 90CB DD23                   INC  IX
136 90CD C33590                 JP   AF000             ;Passage a la suite
137                 ;
138                 DIR0:       EQU  $
139 90D0 110200                 LD   DE,2
140 90D3 210000                 LD   HL,0
141 90D6 1836                   JR   DIR8
142                 ;
```

```
143                     DIR1:       EQU  $
144 90D8 110200                     LD   DE,2
145 90DB 210000                     LD   HL,2
146 90DE 182E                       JR   DIR8
147                     ;
148                     DIR2:       EQU  $
149 90E0 110000                     LD   DE,0
150 90E3 210200                     LD   HL,2
151 90E6 1826                       JR   DIR8
152                     ;
153                     DIR3:       EQU  $
154 90E8 11FEFF                     LD   DE,0FFFEH
155 90EB 210200                     LD   HL,2
156 90EE 181E                       JR   DIR8
157                     ;
158                     DIR4:       EQU  $
159 90F0 11FEFF                     LD   DE,0FFFEH
160 90F3 210000                     LD   HL,0
161 90F6 1816                       JR   DIR8
162                     ;
163                     DIR5:       EQU  $
164 90F8 11FEFF                     LD   DE,0FFFEH
165 90FB 21FEFF                     LD   HL,0FFFEH
166 90FE 180E                       JR   DIR8
167                     ;
168                     DIR6:       EQU  $
169 9100 110000                     LD   DE,0
170 9103 21FEFF                     LD   HL,0FFFEH
171 9106 1806                       JR   DIR8
172                     ;
173                     DIR7:       EQU  $
174 9108 110200                     LD   DE,2
```

```
175 910B 21FEFF              LD   HL,0FFFEH
176              ;
177              DIR8:       EQU  $
178 910E CDEDBB              CALL 0BBEDH
179 9111 3A0290              LD   A,(RT)
180 9114 CB47                BIT  0,A
181 9116 CA5690              JP   Z,AF02
182 9119 C36F90              JP   AF03
183                          END
```

*Remarque :*

Ce programme utilise la routine du FIRMWARE « PLOT RELATIVE ». Reportez-vous en au chapitre 2.7 de la partie 4 pour avoir plus de détails.

## II. Le Décompacteur/Afficheur

Ce programme écrit en BASIC intègre le sous-programme précédent et permet d'afficher simplement une forme à l'écran.

Pour l'utiliser, entrez le nom de la forme à afficher et son implantation en mémoire.

Cette implantation sera toujours &3000 pour une utilisation standard du programme de codage. Pour pouvoir afficher plusieurs formes, il faudra leur donner des adresses d'implantation différentes. Dans ce cas, modifiez le programme de codage lignes 1160, 1690 et 1720 en conséquence. Après avoir chargé la ou les formes, leur dessin apparaît à l'écran.

Le programme d'affichage est le suivant :

```
1000 REM ******************************
1010 REM Affichage de formes codees
1020 REM ******************************
1030 '
1040 'Initialisation
1050 '
1060 MEMORY &3000:MODE 1:INK 0,0:INK 1,10:BORDER 0
1070 PRINT"Ce programme permet d'afficher une ou"
1080 PRINT"plusieurs formes a la sortie du"
1090 PRINT"programme 'Compacteur filiforme'
```

```
1100 '
1110 'Chargement du S/P ASM afficheur de formes
1120 '
1130 FOR I=&9003 TO &911B:READ A:POKE I,A:NEXT I
1140 DATA &DD,&90,&D8,&90,&E0,&90,&E8,&90,&F0,&90,&F8,&90,&0,&91,&8,&91
1150 DATA &22,&0,&90,&FD,&2A,&0,&90,&FD,&56,&0,&FD,&5E,&1,&FD,&66,&2,&FD,&6E,&3,
&CD,&EA,& BE,&DD,&2A,&0,&90,&DD,&23,&DD,&23,&DD,&23,
&DD,&23,&DD,&7E,&0,&FE,&88,&28,&41,&E6,&70,&CB,&2F,&CB,&2F,&CB,&2F,&CB,&2F,&87,&
87,&87,&2A,&3,&90,&6,&0,&4F,&9,&3E,&1,&32,&2,&90,&E9,&DD,&7E,&0,&CB,&5F,
&20,&7,&DD,&23,&DD,&7E,&0,&18,&BB,&DD,&23,&DD,&7E,&0,&FE,&FF,&C8,&CB,&7F,&20,&B,
&AF,&26,&0,&37,&3F,&ED,&6A,&54,&5D,&18
1170 DATA &12,&E6,&7F,&6F,&26,&0,&37,&3F,&ED,&6A,&54,&5D,&21,&0,&0,&ED,&52,&54,&
5D,&DD,&7E,&1,&CB,&7F,&20,&9,&6F,&26,&0,&37,&3F,&ED,
&6A,&18,&10,&E6,&7F,&6F,&26,&0,&37,&3F,&ED,&6A,&44,&4D,&21,&0,&0,&ED,&42,&CD,&ED
,&5B,&DD,&23,&DD,&23,&C3,&35,&90,&11,&2
1190 DATA &0,&21,&0,&0,&18,&36,&11,&2,&0,&21,&2,&0,&18,&2E,&11,&0,&0,&21,&2,&0,&
18,&26,&11,&FE,&FF,&21,&2,&0,&18,&1E,&11,&FE,&FF,&21
,&0,&0,&18,&16,&11,&FE,&FF,&21,&FE,&FF,&18,&E,&11,&0,&0,&21,&FF,&FF,&18,&6,&11,&
2,&0,&21,&FE,&FF,&CD,&ED,&BB,&3A,&2,&90
1190 DATA &CB,&47,&CA,&56,&90,&C3,&6F,&90
1200 FOR I=0 TO 6:READ A:POKE &9200+I,A:NEXT 'Commande de l'afficheur
1210 DATA &21,0,&30,&CD,&13,&90,&C9
1220 '
1230 'Saisie des formes a afficher
1240 '
1250 NF=NF+1 'Numero de forme
1260 LOCATE 1,10:INPUT"Nom de la forme";N$
1270 'INPUT"Implantation memoire";IM(NF)
1280 LOAD N$,IM(NF) 'Chargement forme
1290 'INPUT"Position sur l'ecran: X=";X
1300 'INPUT"                      Y=";Y
1310 A1=INT(X/256):A2=X-A1*256:A3=INT(Y/256):A4=Y-A3*256
1320 POKE IM(NF)+1,A2:POKE IM(NF)+2,A1:POKE IM(NF),A1:POKE IM(NF)+2,A3
1330 PRINT:INPUT"Une autre forme (O/N) ";R$:R$=UPPER$(R$)
```

```
1340 IF R$<>"O" AND R$<>"N" THEN 1330
1350 IF R$="O" THEN 1250
1360 '
1370 'Affichage de la (des) forme(s)
1380 '
1390 CLS:NF=1 'Affichage forme 1
1400 WHILE IM(NF)<>0
1410   MSB=INT(IM(NF)/256):LSB=IM(NF)-MSB*256 'à sur 8 bits
1420   POKE &9201,LSB:POKE &9202,MSB 'Interface afficheur
1430   CALL &9200 'Affichage
1440   NF=NF+1
1450 WEND
1460 END
```

Lignes 1060 à 1090 : Présentation

Lignes 1130 à 1190 : Chargement du sous-programme afficheur

Lignes 1200 à 1210 : Chargement de l'interface BASIC/ASSEMBLEUR

Lignes 1250 à 1350 : Saisie des formes à afficher

Lignes 1390 à 1460 : Affichage des formes.

Les sous-programmes écrits en langage d'assemblage utilisés sont :

— le programme d'affichage défini dans le compacteur filiforme,

— un programme d'interfaçage avec le BASIC qui consiste à donner dans HL la première adresse de la forme à afficher. Cette adresse est décomposée en poids fort et poids faible Ligne 1300.

# 5/10.3.2

# Compacteurs monochromes en mode 1

Nous allons étudier :

— Le compactage d'objets monochromes selon deux principes :

- le tassement d'octets,
- la recherche de répétition.

— Le décompactage et l'affichage d'objets compactés selon les deux principes.

## I. Les compacteurs

### TASSEMENT D'OCTETS

Nous avons vu plus haut (reportez-vous à la description du sous-programme ASSEMBLEUR MBG, chapitre 10.2 de la partie 5) quelle était la structure de l'écran en MODE 1.

*Si un objet affiché sur l'écran possède seulement deux couleurs :*

La couleur de fond et une couleur de motif, il apparaît que l'on peut réduire son occupation en mémoire.

*Le principe adopté est le suivant :*

Quatre bits sur les huit utilisés pour définir un groupe de 4 pixels sont essentiels pour définir totalement le dessin monochrome.

*Remarque :*

Dans la suite, nous parlerons de ce compacteur en le désignant par le nom « type 1 ».

Si PEN 1 est la couleur du dessin, les 4 bits seront ceux de poids faible,

Si PEN 2 est la couleur du dessin, les 4 bits seront ceux de poids fort,

Si PEN 3 est la couleur du dessin, les 4 bits seront au choix, ceux de poids faible ou ceux de poids fort.

Cette distinction des couleurs apparaît lignes 1450 à 1480 :

Ligne 1450 : Extraction d'un octet du dessin,

Ligne 1460 : PEN 1,

Ligne 1470 : PEN 2,

Ligne 1480 : PEN 3.

## REPETITION DE MOTIFS

Un deuxième type de compactage intéressant et facilement mis en œuvre consiste à compter le nombre d'octets (groupe de 4 pixels en MODE 1) identiques sur une ligne élémentaire, et à coder cette répétition :

Nombre de répétitions, Octet, Terminateur.

*Remarque :*

Dans la suite, nous parlerons de ce compacteur en le désignant par le nom « type 2 ».

L'écriture du sous-programme BASIC correspondant à ce compactage est immédiate.

Pour chaque ligne du bloc à compacter, une recherche de répétition au niveau octet est effectuée puis codée :

Lignes 1610 à 1630 : Recherche de répétition

Ligne 1640 : Mémorisation

Ligne 1660 : Terminateur

Entrez le nom de l'image à compacter, puis délimitez les coins supérieur gauche et inférieur droit de la portion d'image à compacter en vous servant des touches-flèches (la validation des deux coins se fait par la touche « ENTER »).

Un certain temps, proportionnel aux dimensions du dessin à compacter et à la lenteur du BASIC (!) est nécessaire pour le calcul des fichiers compactés.

Le programme affiche alors la place occupée par le compactage de type 1 et 2.

Le programme de compactage des types 1 et 2 est le suivant :

```
1000 REM Compacteur graphique monochrome simple
1010 '*****************************************************
1020 REM Chargement de l'image d'ecran concernee
1030 '
1040 INK 0,0:INK 1,10:BORDER 0:MODE 1
1050 PRINT"Entrez le nom de l'image a traiter,"
```

```
1010 PRINT"puis servez-vous des touches-fleches"
1070 PRINT"pour delimiter les coins superieur"
1080 PRINT"gauche et inferieur droit de l'image"
1090 PRINT:INPUT "Nom de l'image a traiter ";N$
1100 LOAD N$
1110 '*************************************************************
1120 'Chargement de la routine ASM DOTPOS
1130 '
1140 FOR I=&9006 TO &9013
1150   READ A:POKE I,A
1160 NEXT I
1170 DATA &ED,&5B,&0,&90,&2A,&2,&90,&CD,&1D,&BC,&22,&4,&90,&C9
1180 '*************************************************************
1190 REM Positionnement des coins superieur gauche et inferieur droit
1200 '
1210 X=0: Y=0 'Initialisation du curseur graphique
1220 COU=TEST(X,Y):PLOT X,Y,3
1230 A$=INKEY$:IF A$="" THEN 1230 'Saisie deplacement
1240 A=ASC(A$):PLOT X,Y,COU 'Affichage curseur
1250 IF A=240 THEN Y=Y+2 'Vers le haut
1260 IF A=241 THEN Y=Y-2 'Vers le bas
1270 IF A=242 THEN X=X-2 'Vers la gauche
1280 IF A=243 THEN X=X+2 'Vers la droite
1290 IF A<>13 THEN 1220 'Boucle de saisie
1300 '
1310 P=P+1 'Nombre de saisies
1320 IF P=1 THEN X1=X/2:Y1=Y/2:GOTO 1220 'Saisie coin inferieur droit
1330 IF P=2 THEN X2=X/2:Y2=Y/2 'Fin de saisie
1340 CALL &9006,INT((X2-X1)/4) 'Nombre d'octets horizontalement
1350 '*************************************************************
1360 '
1370 '
```

```
1380 AG=&9005 'Adresse memoire du debut des donnees compactees
1390 FOR Y=Y2 TO Y1
1400   XH=INT(X1/256):XL=X1-XH*256:YH=INT(Y/256):YL=Y-YH*256
1410   POKE &9000,XL:POKE &9001,XH:POKE &9002,YL:POKE &9003,YH
1420   CALL &9006 'DOTPOS
1430   AD=PEEK(&9004)+PEEK(&9005)*256 'Adresse ecran debut de ligne
1440   FOR X=X1 TO X2 STEP 4
1450     A=PEEK(AD+(X-X1)/4) 'Octet graphique
1460     IF A<16 THEN AF=AF+A*(16^(1-PA))
1470     IF A>15 AND INT(A/16)=A/16 THEN AF=AF+(A/16)*(16^(1-PA))
1480     IF A>15 AND INT(A/16)<>A/16 THEN AF=AF+((A AND &F0)/16)*(16^(1-PA))
1490     IF PA=1 THEN PA=0 ELSE PA=1
1500     IF PA=0 THEN AG=AG+1:POKE AG,AF:AF=0 '1 Octet compacte mis en memoire
1510   NEXT X
1520 NEXT Y
1530 IF PA=1 THEN AG=AG+1:POKE AG,AF
1540 AG=AG+1:POKE AG,&FF:AG=AG+1:POKE AG,&AA 'Terminateur
1550 '*****************************************************
1560 ' Compactage 2
1570 '
1580 P1=&9006:P2=&9006 'Pointeurs sur fichiers compactes
1590 A=PEEK(P1):I=1:P1=P1+1:B=PEEK(P1)
1600 IF A=&FF THEN IF B=&AA THEN 1660
1610 WHILE B=A
1620   I=I+1:P1=P1+1:B=PEEK(P1)
1630 WEND
1640 POKE P2,I:POKE P2+1,A:P2=P2+2
1650 GOTO 1590 'Boucle de compactage
1660 POKE P2,A:POKE P2+1,B:P2=P2+2 'Terminateur
1670 '*****************************************************
2000 ' Informations sur les longueurs des fichiers compactes
2010 'et eventuelle sauvegarde magnetique
```

```
2020 '
2030 CLS:PRINT"Fichier compacte 1) de &8000 a ";HEX$(AG)
2040 FOR I=&8000 TO &8005
2050   POKE I+&1000,PEEK(I)
2060 NEXT I
2070 POKE &9004,PEEK(&9004)+1
2080 PRINT:PRINT"Fichier compacte 2) de &9000 a ";HEX$(P2)
2090 PRINT:PRINT"Sauvegarde magnetique (O/N) ?"
2100 A$=INKEY$:IF A$="" THEN 2100
2110 A$=UPPER$(A$)
2120 IF A$<>"O" AND A$<>"N" THEN SOUND 1,100,30:GOTO 2100
2130 IF A$="N" THEN 2230 'Fin du programme
2140 PRINT:PRINT"1)Type 1, 2)Type 2 ou 3)Les deux types"
2150 A$=INKEY$:IF A$="" THEN 2150
2160 IF A$<>"1" AND A$<>"2" AND A$<>"3" THEN SOUND 1,100,30:GOTO 2150
2170 PRINT
2180 IF A$="1" THEN INPUT"Nom du fichier type 1: ";N1$
2190 IF A$="2" THEN INPUT"Nom du fichier type 2: ";N2$
2200 IF A$="3" THEN INPUT"Nom du fichier type 1: ";N1$:INPUT"Nom du fichier type
 2: ";N2$
2210 IF N1$<>"" THEN SAVE N1$,B,&8000,AG+1-&8000
2220 IF N2$<>"" THEN SAVE N2$,B,&9000,P2+1-&9000
2230 END
```

Lignes 1040 à 1100 : Initialisation et présentation

Ligne 1100 : Chargement de l'image à traiter

Lignes 1140 à 1170 : Interface Assembleur avec la routine « DOTPOS » (Reportez-vous au chap. 10.2 de la partie 5 pour avoir plus de détails.)

Lignes 1190 à 1240 : Définition des coins supérieur gauche et inférieur droit du bloc à compacter.

Lignes 1400 à 1430 : Interfaçage avec DOTPOS

Lignes 1440 à 1510 : Compactage

Ligne 1530 : Mémorisation

Ligne 1540 : Terminateur (&FFAA)

Lignes 1580 à 1660 : Compacteur de type 2

Lignes 2000 à 2230 : Sauvegarde d'un des compactages

La routine du FIRMWARE « DOTPOS » est utilisée dans ce programme. Reportez-vous au chapitre 2.7 de la partie 4 pour avoir plus de détails à ce sujet.

## II. Les décompacteurs-afficheurs

Le programme de compactage « type 1 et 2 » ne serait pas complet sans ce petit programme d'affichage de fichiers compactés de type 1 et 2.

Entrez le nom du fichier compacté par le programme précédent, le type de compactage (1 ou 2) et la position graphique sur l'écran en bas et à gauche du bloc à afficher (0<X<329 et 0<Y<200). Le fichier est chargé et l'affichage est immédiat.

Le programme d'affichage est le suivant :

```
1000 'Affichage de fichiers compactes de type 1 ou 2

1010 '

1020 'Decompacteur de type 1

1030 '

1040 FOR I=&A00F TO &A0DF:READ A:POKE I,A:NEXT

1050 DATA &E5,&21,&0,&A0,&6,&F,&3E,&0,&77,&23,&10,&FC,&E1,&11,&4,&A0,&1,&6,&0,&E
D,&B0,&22,&0,&A0,&FD,&21,&3,&A0,&DD,&2A,&0,&A0,&3A,&

9,&A0,&3D,&28,&D,&3D,&28,&5,&21,&A9,& A0,&18,&8,&21,&93,&A0,&18,&3,&21,&81,&A0,&
22,&B,&A0,&ED,&5B,&4,&A0,&2A,&6,&A0

1060 DATA & CD,&1D,&BC,&3A,&8,&A0,&47,&DD,&7E,&0,&FE,&FF,&20,&8,&DD,&7E,&1,&FE,&
AA,&20,&1,& C9,&DD,&7E,&0,&E6,&F0,&CB,&F,&CB,&F,&CB,

&F,&CB,&F,&57,&DD,&7E,&0,&E6,&F,&5F,&DD,&E5,&DD,&2A,&B,&A0,&DD,&E9,&DD,&E1,&72,&
23,&5,&CC,&CB,&A0,&73,&23,&DD,&23,&5

1070 DATA &CC,&CB,&A0,&18,&C3,&7A,&CB,&7,&CB,&7,&CB,&7,&CB,&7,&57,&7B,&CB,&7,&CB
,&7,&CB,&7,& CB,&7,&5F,&18,&D8,&7A,&FD,&77,&0,&CB,&7

,&CB,&7,&CB,&7,&CB,&7,&FD,&B6,&0,&57,&7B,&FD,&77,&0,&CB,&7,&CB,&7,&CB,&7,&CB,&7,
&FD,&B6,&0,&5F,&18,&B6,&D5,&2A,&6,&A0

1080 DATA &23,&22,&6,&A0,&ED,&5B,&4,&A0,&CD,&1D,&BC,&3A,&8,&A0,&47,&D1,&C9

1090 '------------------------------ ------------------------------------

1100 'Decompacteur de type 2

1110 '
```

```
1120 FOR I=&A108 TO &A1DF: READ A:POKE I,A:NEXT

1130 DATA &11,&0,&A1,&1,&6,&0,&ED,&B0,&22,&9,&A1,&FD,&21,&8,&A1,&DD,&2A,&9,&A1,&
2A,&2,&A1,&ED,&5B,&0,&A1,&CD,&1D,&BC,&3A,&4,&A1,&47,

&DD,&7E,&0,&FE,&FF,&20,&8,&DD,&7E,&1,&FE,&AA,&20,&1,&C9,&DD,&4E,&0,&DD,&7E,&1,&E
6,&F0,&CB,&F,&CB,&F,&CB,&F,&CB,&F

1140 DATA &57,&DD,&7E,&1,&E6,&F,&5F,&3A,&5,&A1,&3D,&28,&39,&3D,&28,&22,&7A,&FD,&
77,&0,&CB,&7,&CB,&7,&CB,&7,&CB,&7,&FD,&B6,&0,&57,&7B

,&FD,&77,&0,&CB,&7,&CB,&7,&CB,&7,&CB,&7,& FD,&B6,&0,&5F,&19,&14,&7A,&CB,&7,&CB,&
7,&CB,&7,&CB,&7,&57,&7B,&CB,&7,&CB,&7,&CB

1150 DATA &7,&CB,&7,&5F,&73,&23,&5,&28,&E,&73,&23,&5,&28,&27,&D,&20,&F3,&DD,&23,
&DD,&23,& 10,&B9,&C5,&D5,&2A,&2,&A1,&23,&22,&2,&A1,&

ED,&5B,&0,&A1,&CD,&1D,&BC,&22,&6,&A1,&D1,&C1,&2A,&6,&A1,&3A,&4,&A1,&47,&19,&D4,&
CE,&D5,&3A,&2,&A1,&23,&22,&2,&A1,&ED

1160 DATA &5B,&0,&A1,&CD,&1D,&BC,&22,&6,&A1,&D1,&C1,&2A,&6,&A1,&3A,&4,&A1,&47,&1
9,&D9

1170 '------------------------------------------------------------------------

1180 'Interface BASIC/ASM avec les S/P decompacteur 1 et 2

1190 '

1200 FOR I=&A200 TO &A206:READ A:POKE I,A:NEXT

1210 DATA &21,&0,&30,&CD,&F,&90,&C9

1220 '------------------------------------------------------------------------

1230 MEMORY &4000:MODE 1:INK 0,0:INK 1,10:BORDER 0:PEN 2:PRINT"   AFFICHEUR DE F
ICHIERS COMPACTES":PEN 1

1240 LOCATE 1,10

1250 INPUT"Nom du fichier ";F$

1260 INPUT"Type de compactage (1 ou 2) ";R

1270 IF R<>1 AND R<>2 THEN SOUND 1,100,30:GOTO 1260

1280 INPUT"Position d'affichage: X=";X

1290 INPUT"                      Y=";Y

1300 '

1310 LOAD F$ 'Chargement du fichier compacte

     IF R=1 THEN AD=&8000 ELSE AD=&9000 'Adresse d'implantation du fichier

     INT(Y/256):A2=X-A1*256:A3=INT(Y/256):A4=Y-A3*256

     CALL AD,A2:POKE AD+1,A1:POKE AD+2,A4:POKE AD+3,A3 'Coordonnees X,Y

     ... 2   'Couleur d'encre

     ... :POKE &A204,&F:POKE &A205,&A0 'Interface DECOMI
```

```
1370 IF P=2 THEN POKE &A202,&90:POKE &A204,&B:POKE &A205,&A1 'Interface DECOM2
1380 POKE &0004,11
1390 CLS:CALL &A200 'Affichage
1400 END
```

Lignes 1040 à 1160 : Chargement des décompacteurs

Lignes 1200 à 1210 : Programme d'interfaçage entre le BASIC et les décompacteurs

Lignes 1250 à 1300 : Entrée du type de compactage et chargement du fichier compacté

Lignes 1320 à 1380 : Interfaçage ASSEMBLEUR

Ligne 1390 : Affichage

Deux sous-programmes sont écrits en ASSEMBLEUR pour minimiser leur temps d'exécution. Il s'agit des programmes de décompactage/affichage.

*Décompactage de type 1 :*

Ce décompacteur occupe les lignes 1040 à 1080 dans le listing BASIC. Le principe de décompactage est simple. Il consiste à isoler les quartets de poids fort puis faible de chaque octet et à les afficher en tant qu'octets, jusqu'à la rencontre du code terminateur &FFAA qui marque la fin du fichier graphique compacté.

La routine du FIRMWARE « DOTPOS » est utilisée pour calculer l'adresse de la ligne élémentaire suivante. Reportez-vous au chap. 2.7 de la partie 4 pour avoir plus de détails.

```
1               ;
2               ; Afficheur de fichiers graphiques
3               ; de type 1
4               ;
5               ; Entree : HL=à Fichier
6               ; Pt d'entree : TYPE1
7               ; Sortie : Tts registres ecrases
8               ;
9                       ORG  9AAFH
10                      LOAD 9AAFH
11              ;
12              ;Recherche nb de zones
```

```
13                ;
14                DATA:       EQU   $
15                PTRAD:      DS    2          ;à courante ecran
16 9AB1 00        GEN1:       DB    0          ;Usage general
17                GEN2:       DS    1          ;Usage general
18                SX:         DS    2          ;Sauvegarde en X
19                SY:         DS    2          ;Sauvegarde en Y
20                LA:         DS    1          ;Larg. en oct. du dessin
21                CE:         DS    1          ;Couleur d'encre
22                LC:         DS    1          ;Oct. largeur courante
23                TR:         DS    2          ;à de trait=F(couleur)
24                DEFI:       DS    2          ;Debut fichier graph.
25                ;
26                DOTPOS:     EQU   0BC1DH     ;Dot Position
27                ;
28                ;Initialisation
29                ;
30                TYPE1:      EQU   $
31 9ABE E5                    PUSH  HL         ;Sauv à data
32 9ABF 21AF9A                LD    HL,DATA    ;Debut de RAZ
33 9AC2 060F                  LD    B,15       ;Lgr
34 9AC4 3E00                  LD    A,0
35                RAZ:        EQU   $
36 9AC6 77                    LD    (HL),A
37 9AC7 23                    INC   HL
38 9AC8 10FC                  DJNZ  RAZ
39 9ACA E1                    POP   HL         ;Restitution à data
40                ;
41                ;HL=à Source
42 9ACB 11B39A                LD    DE,SX      ;Destination
43 9ACE 010600                LD    BC,6       ;Longueur du transfert
44 9AD1 EDB0                  LDIR             ;Transfert
```

```
45 9AD3 22AF9A              LD   (PTRAD),HL        ;Pointeur debut DATA
46 9AD6 FD21B29A            LD   IY,GEN2
47 9ADA DD2AAF9A            LD   IX,(PTRAD)
48              ;
49              ;Initialisation couleur d'encre
50              ;
51 9ADE 3AB89A              LD   A,(CE)           ;Couleur d'encre
52 9AE1 3D                  DEC  A
53 9AE2 280D                JR   Z,COU1           ;Couleur 1
54 9AE4 3D                  DEC  A
55 9AE5 2805                JR   Z,COU2           ;Couleur 2
56              COU3:       EQU  $                ;Couleur 3
57 9AE7 21589B              LD   HL,TR3           ;à de traitement 3
58 9AEA 1808                JR   COUFIN
59              COU2:       EQU  $
60 9AEC 21429B              LD   HL,TR2           ;à de traitement 2
61 9AEF 1803                JR   COUFIN
62              COU1:       EQU  $                ;Couleur 1
63 9AF1 21309B              LD   HL,TR1           ;à de traitement 1
64              COUFIN:     EQU  $
65 9AF4 22BA9A              LD   (TR),HL          ;à de trait. choisie
66              ;
67              ;
68              AFO:        EQU  $                ;Debut affichage
69              ;Initialisation
70 9AF7 ED5BB39A            LD   DE,(SX)
71 9AFB 2AB59A              LD   HL,(SY)
72 9AFE CD1DBC              CALL DOTPOS           ;Position Ecran
73 9B01 3AB79A              LD   A,(LA)
74 9B04 47                  LD   B,A              ;B=Nbre Octets largeur
75              ;
76              ;Decompactage
```

```
77                ;
78                AF0:      EQU   $
79 9B05 DD7E00             LD    A,(IX+0)
80 9B08 FEFF               CP    0FFH
81 9B0A 2008               JR    NZ,AF00
82 9B0C DD7E01             LD    A,(IX+1)
83 9B0F FEAA               CP    0AAH
84 9B11 2001               JR    NZ,AF00
85 9B13 C9                 RET                ;Terminateur rencontre
86                AF00:     EQU   $
87 9B14 DD7E00             LD    A,(IX+0)
88 9B17 E6F0               AND   0F0H
89 9B19 CB0F               RRC   A
90 9B1B CB0F               RRC   A
91 9B1D CB0F               RRC   A
92 9B1F CB0F               RRC   A
93 9B21 57                 LD    D,A          ;Quartet fort
94 9B22 DD7E00             LD    A,(IX+0)
95 9B25 E60F               AND   0FH
96 9B27 5F                 LD    E,A          ;Quartet faible
97 9B28 DDE5               PUSH  IX
98 9B2A DD2ABA9A           LD    IX,(TR)
99 9B2E DDE9               JP    (IX)         ;Traitement d'un octet
100               ;
101               TR1:      EQU   $            ;Affichage
102 9B30 DDE1              POP   IX
103 9B32 72                LD    (HL),D
104 9B33 23                INC   HL
105 9B34 05                DEC   B
106 9B35 CC7A9B            CALL  Z,LISUI      ;Ligne suivante
107 9B38 73                LD    (HL),E
108 9B39 23                INC   HL
```

```
109 9B3A DD23                   INC   IX
110 9B3C 05                     DEC   B
111 9B3D CC7A9B                 CALL  Z,LISUI           ;Ligne suivante
112 9B40 18C3                   JR    AFO               ;Boucle d'affichage
113                 ;
114                 TR2:        EQU   $                 ;Affichage couleur 2
115 9B42 7A                     LD    A,D
116 9B43 CB07                   RLC   A
117 9B45 CB07                   RLC   A
118 9B47 CB07                   RLC   A
119 9B49 CB07                   RLC   A
120 9B4B 57                     LD    D,A
121 9B4C 7B                     LD    A,E
122 9B4D CB07                   RLC   A
123 9B4F CB07                   RLC   A
124 9B51 CB07                   RLC   A
125 9B53 CB07                   RLC   A
126 9B55 5F                     LD    E,A
127 9B56 18D8                   JR    TR1               ;Affichage
128                 TR3:        EQU   $                 ;Affichage couleur 3
129 9B58 7A                     LD    A,D
130 9B59 FD7700                 LD    (IY+0),A
131 9B5C CB07                   RLC   A
132 9B5E CB07                   RLC   A
133 9B60 CB07                   RLC   A
134 9B62 CB07                   RLC   A
135 9B64 FDB600                 OR    (IY+0)
136 9B67 57                     LD    D,A
137 9B68 7B                     LD    A,E
138 9B69 FD7700                 LD    (IY+0),A
139 9B6C CB07                   RLC   A
140 9B6E CB07                   RLC   A
```

```
141 9B70 CB07                RLC  A
142 9B72 CB07                RLC  A
143 9B74 FDB600              OR   (IY+0)
144 9B77 5F                  LD   E,A
145 9B78 18B6                JR   TR1              ;Affichage
146              ;
147              LISUI:      EQU  $                ;Affich. ligne suiv.
148 9B7A D5                  PUSH DE
149 9B7B 2AB59A              LD   HL,(SY)
150 9B7E 23                  INC  HL
151 9B7F 22B59A              LD   (SY),HL
152 9B82 ED5BB39A            LD   DE,(SX)
153 9B86 CD1DBC              CALL DOTPOS           ;HL=à mem ecran
154 9B89 3AB79A              LD   A,(LA)
155 9B8C 47                  LD   B,A
156 9B8D D1                  POP  DE
157 9B8E C9                  RET                   ;Ligne suivante
158                          END
```

*Décompacteur de type 2 :*

Il occupe les lignes 1120 à 1160 du listing BASIC.

Le principe est différent du précédent. Il consiste en l'isolement de deux octets : le deuxième représente le motif à répéter, le premier le nombre de répétitions à effectuer. Le deuxième octet est affiché n fois et l'opération se répète jusqu'à la rencontre du terminateur &FFAA.

Comme dans le sous-programme précédent, la routine du FIRMWARE « DOTPOS » est utilisée pour calculer l'adresse de la ligne élémentaire suivante lors de l'affichage.

```
1                ;
2                ;Afficheur de fichiers graphiques
3                ; de type 2 compactes par COMPAC2
4                ;
5                ; Entree : HL=à Fichier
6                ; Pt d'entree : TYPE2
```

```
 7                ; Sortie : Ts registres ecrases
 8                ;
 9                          ORG  91FCH
10                          LOAD 91FCH
11                ;
12                XABS:     EQU  9000H
13                YABS:     EQU  9002H
14                LAOC:     EQU  9004H
15                COEN:     EQU  9005H
16                DEDA:     EQU  9006H
17                DOTPOS:   EQU  OBC1DH
18                ;
19                DATA:     EQU  $              ;Debut des DS
20                SX:       DS   2
21                SY:       DS   2
22                LA:       DS   1
23                CE:       DS   1
24                POS:      DS   2
25                GEN1:     DS   1
26                PTRAD:    DS   2
27                ;
28                ;
29                ;Initialisation
30                ;
31                ;HL=à Source
32                TYPE2:    EQU  $
33 9207 11FC91             LD   DE,SX          ;Destination
34 920A 010600             LD   BC,6           ;Longueur
35 920D EDB0               LDIR                ;Transfert
36 920F 220592             LD   (PTRAD),HL
37 9212 FD210492           LD   IY,GEN1
38 9216 DD2A0592           LD   IX,(PTRAD)
```

```
39                      ;
40 921A 2AFE91          LD    HL,(SY)
41 921D ED5BFC91        LD    DE,(SX)
42 9221 CD1DBC          CALL  DOTPOS
43                      ;
44                      ;Decompactage
45                      ;
46 9224 3A0092          LD    A,(LA)
47 9227 47              LD    B,A
48                      AF0:  EQU   $
49 9228 DD7E00          LD    A,(IX+0)
50 922B FEFF            CP    0FFH
51 922D 2008            JR    NZ,AF00
52 922F DD7E01          LD    A,(IX+1)
53 9232 FEAA            CP    0AAH
54 9234 2001            JR    NZ,AF00
55 9236 C9              RET                     ;Terminateur rencontre
56                      ;
57                      AF00: EQU   $
58 9237 DD4E00          LD    C,(IX+0)          ;Nb de repetitions
59 923A DD7E01          LD    A,(IX+1)
60 923D E6F0            AND   0F0H
61 923F CB0F            RRC   A
62 9241 CB0F            RRC   A
63 9243 CB0F            RRC   A
64 9245 CB0F            RRC   A
65 9247 57              LD    D,A               ;Octet gauche isole
66 9248 DD7E01          LD    A,(IX+1)
67 924B E60F            AND   0FH
68 924D 5F              LD    E,A               ;Octet droit isole
69                      ;
70 924E 3A0192          LD    A,(CE)
```

```
 71 9251 3D                     DEC   A
 72 9252 2839                   JR    Z,AF3             ;Encre demandee=1
 73 9254 3D                     DEC   A
 74 9255 2822                   JR    Z,INK2            ;Encre demandee=2
 75              INK3:          EQU   $
 76 9257 7A                     LD    A,D
 77 9258 FD7700                 LD    (IY+0),A
 78 925B CB07                   RLC   A
 79 925D CB07                   RLC   A
 80 925F CB07                   RLC   A
 81 9261 CB07                   RLC   A
 82 9263 FDB600                 OR    (IY+0)
 83 9266 57                     LD    D,A
 84 9267 7B                     LD    A,E
 85 9268 FD7700                 LD    (IY+0),A
 86 926B CB07                   RLC   A
 87 926D CB07                   RLC   A
 88 926F CB07                   RLC   A
 89 9271 CB07                   RLC   A
 90 9273 FDB600                 OR    (IY+0)
 91 9276 5F                     LD    E,A
 92 9277 1814                   JR    AF3               ;Suite
 93              ;
 94              INK2:          EQU   $
 95 9279 7A                     LD    A,D
 96 927A CB07                   RLC   A
 97 927C CB07                   RLC   A
 98 927E CB07                   RLC   A
 99 9280 CB07                   RLC   A
100 9282 57                     LD    D,A
101 9283 7B                     LD    A,E
102 9284 CB07                   RLC   A
```

```
103 9286 CB07                   RLC   A
104 9288 CB07                   RLC   A
105 928A CB07                   RLC   A
106 928C 5F                     LD    E,A
107                 ;
108                 ;Affichage
109                 ;
110                 AF3:        EQU   $
111 928D 72                     LD    (HL),D
112 928E 23                     INC   HL
113 928F 05                     DEC   B
114 9290 280E                   JR    Z,AF1
115                 AF31:       EQU   $
116 9292 73                     LD    (HL),E
117 9293 23                     INC   HL
118 9294 05                     DEC   B
119 9295 2827                   JR    Z,AF2
120                 AF32:       EQU   $
121 9297 0D                     DEC   C
122 9298 20F3                   JR    NZ,AF3
123 929A DD23                   INC   IX
124 929C DD23                   INC   IX
125 929E 1888                   JR    AF0
126                 AF1:        EQU   $
127 92A0 C5                     PUSH  BC
128 92A1 D5                     PUSH  DE
129 92A2 2AFE91                 LD    HL,(SY)
130 92A5 23                     INC   HL
131 92A6 22FE91                 LD    (SY),HL
132 92A9 ED5BFC91               LD    DE,(SX)
133 92AD CD1DBC                 CALL  DOTPOS
134 92B0 220292                 LD    (POS),HL
```

```
135 92B3 D1                     POP   DE
136 92B4 C1                     POP   BC
137 92B5 2A0292                 LD    HL,(POS)
138 92B8 3A0092                 LD    A,(LA)
139 92BB 47                     LD    B,A
140 92BC 18D4                   JR    AF31
141              AF2:           EQU   $
142 92BE C5                     PUSH  BC
143 92BF D5                     PUSH  DE
144 92C0 2AFE91                 LD    HL,(SY)
145 92C3 23                     INC   HL
146 92C4 22FE91                 LD    (SY),HL
147 92C7 ED5BFC91               LD    DE,(SX)
148 92CB CD1DBC                 CALL  DOTPOS)
149 92CE 220292                 LD    (POS),HL
150 92D1 D1                     POP   DE
151 92D2 C1                     POP   BC
152 92D3 2A0292                 LD    HL,(POS)
153 92D6 3A0092                 LD    A,(LA)
154 92D9 47                     LD    B,A
155 92DA 18BB                   JR    AF32
156                             END
```

# 5/10.4

# Graphicomanies

Ce chapitre est une galerie ouverte à tous, petits et grands pourront y proposer les plus beaux fleurons de leur logithèque personnelle. Les critères qui feront que votre œuvre sera retenue et publiée sont simples. Beau sera le dessin, court sera le programme. Ces programmes pourront être écrits en tous langages tournant sur CPC.

## 5/10.4.1

## Jeux de points

L'idée directrice de ces quelques petits programmes est de faire correspondre à tout point de l'écran un nombre z fonction plus ou moins compliquée des coordonnées x et y du point écran et dont la valeur absolue autorisera le tracé si elle est divisible par 2 et ne l'autorisera pas si elle n'est pas multiple de 2.

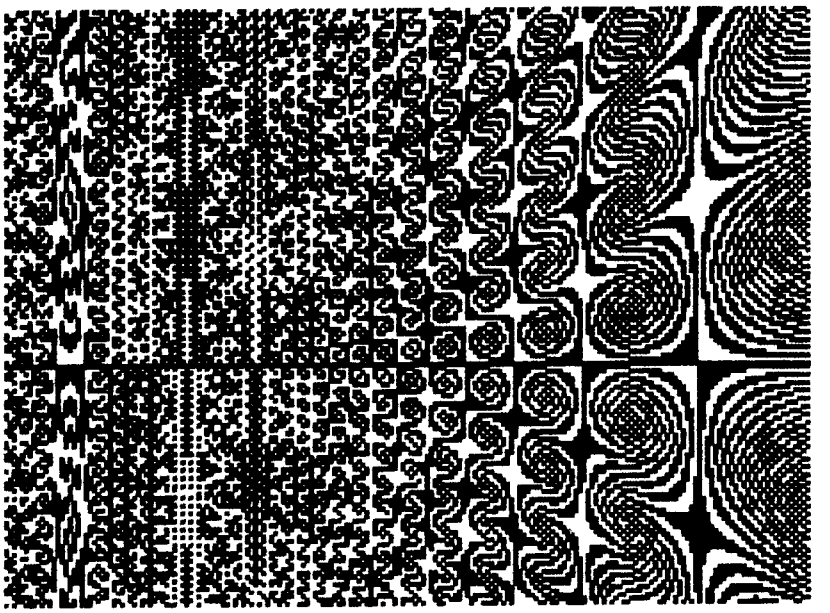
Ce critère apparaît en ligne 100 dans chacun des programmes.

Trois fonctions différentes vous sont proposées ici, et comme vous pouvez le constater, chacune de ces fonctions conduit à un graphisme différent et pour le moins original. Les valeurs de « a » et « b » sont en fait les points de départ du balayage d'une portion d'un plan fictif. Le nombre « cote » est en fait la longueur du côté d'un carré de ce plan et dont le coin inférieur gauche se situe en [a, b].

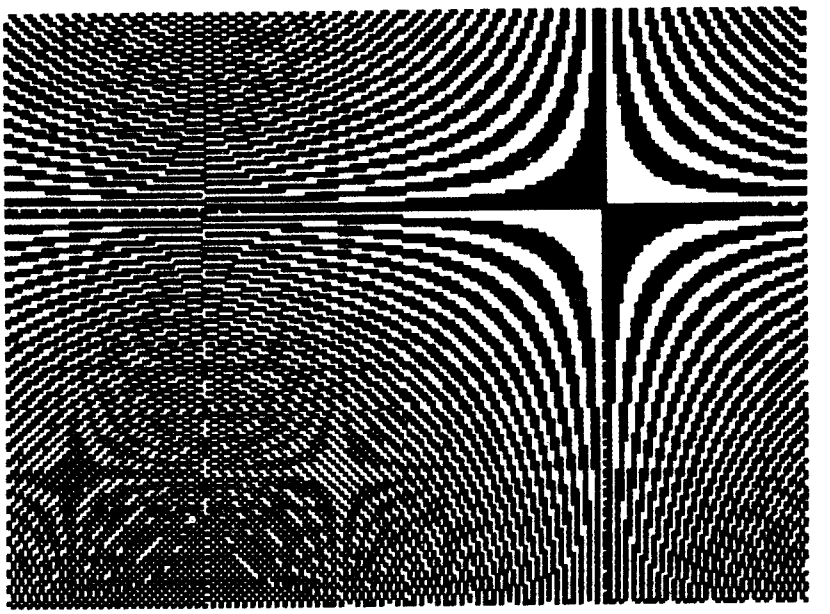
A chacun des points de ce carré, on fait correspondre un point de votre écran, celui-là bien réel. Les coordonnées du plan-écran étant ici « i » et « j ». Le balayage de l'écran se fait aux lignes 40 et 50, la transformation en 60 et 70 et le calcul de la fonction « z » en 80.

Graph 1

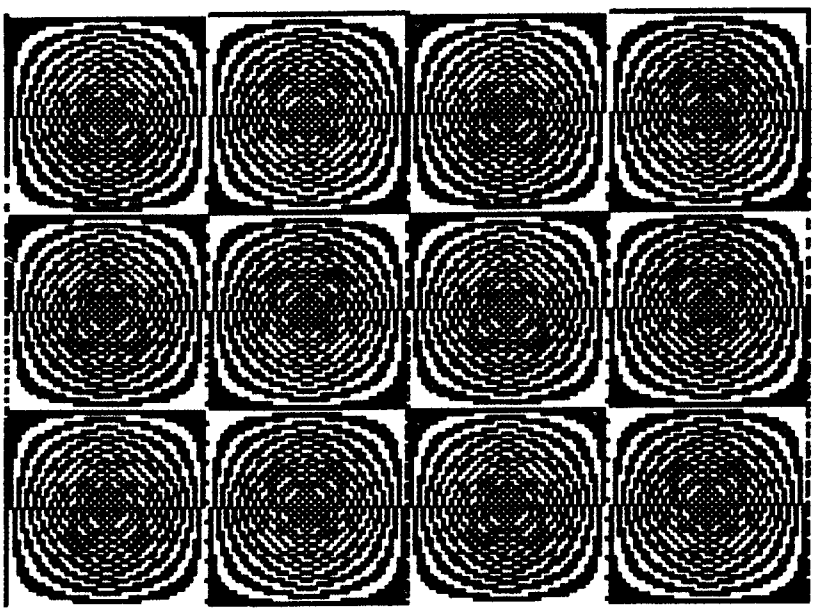
```
1 REM GRAPH1
5 MODE 2:CLS
10 INPUT"a,b";a,b
20 INPUT"cote";cote
30 CLS:LOCATE 1,1
40 FOR i=0 TO 300
50 FOR j=0 TO 200
60 x=a+cote*i/100
70 y=b+cote*j/100
80 z=(1.1)^x*y
90 c=INT(z)
100 IF c MOD 2=0 THEN PLOT i,j
110 NEXT j,i
```



Variations sur Graph 2
diverses valeurs de a, b et côté

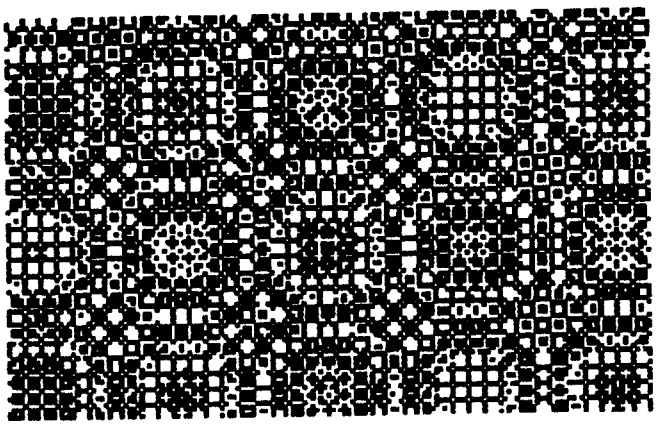


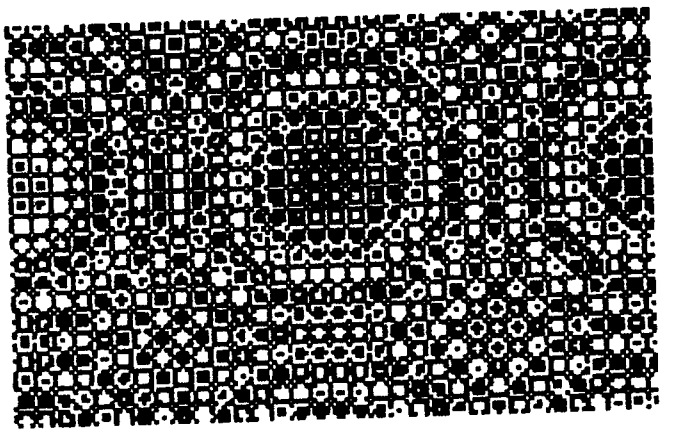
```
1 REM GRAPH2
5 MODE 2:CLS
10 INPUT"a,b";a,b
20 INPUT"cot(";cote
30 CLS:LOCATE 1,1
40 FOR i=0 TO 300
50 FOR j=0 TO 200
60 x=a+cote*i/100
70 y=b+cote*j/100
80 z=x*y
90 c=INT(z)
100 IF c MOD 2= 0 THEN PLOT i,j
110 NEXT j,i
```

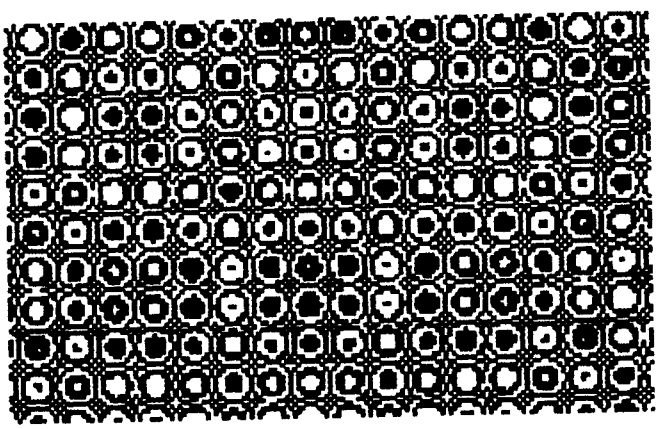


Graph 3

```
1 REM GRAPH3
5 MODE 2:CLS
10 INPUT"a,b";a,b
20 INPUT"cote";cote
30 CLS:LOCATE 1,1
40 FOR i=0 TO 300
50 FOR j=0 TO 200
60 x=a+cote*i/100
70 y=b+cote*j/100
80 z=(x^2+y^2)
90 c=INT(z)
100 IF c MOD 2= 0 THEN PLOT i,j
110 NEXT j,i
```

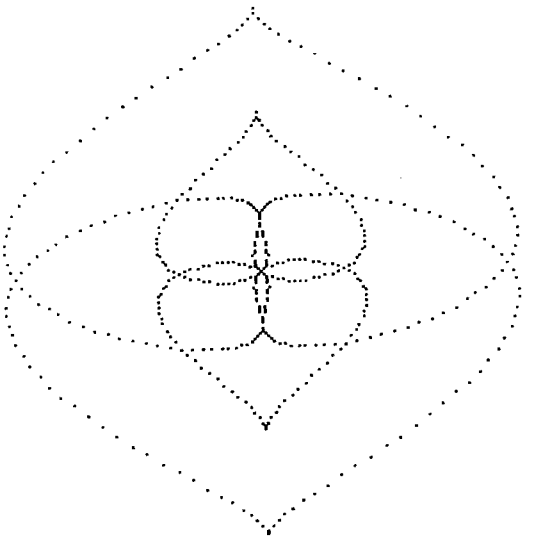




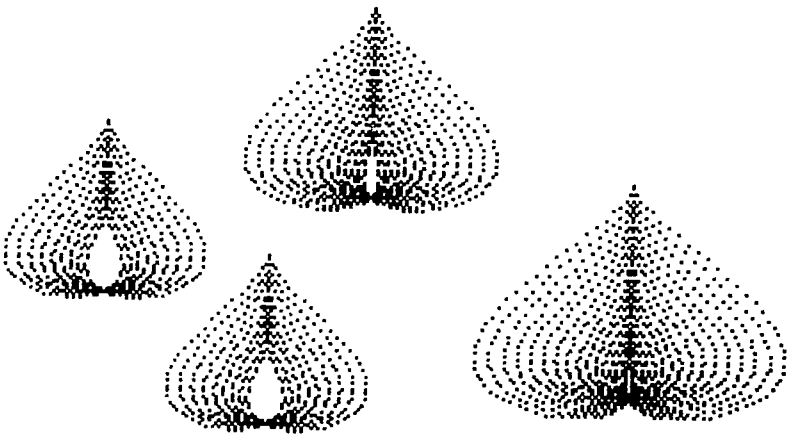


## Entrelacs

```
1 REM ENTRELACS
10 CLS:MODE 2
20 H=90
30 FOR i=PI TO 5*PI+0.1 STEP PI/30
40 a=COS(i/2):b=SIN(i/2)
50 x=320+H*a*a:y=200+H*b*0.65:x1=320-H*a*a:y1=y:x2=x:y2=200-H*b*0.65
60 x3=x1:y3=y2:PLOT x,y:PLOT x1,y1:PLOT x2,y2:PLOT x3,y3:H=H-2:NEXT i
```

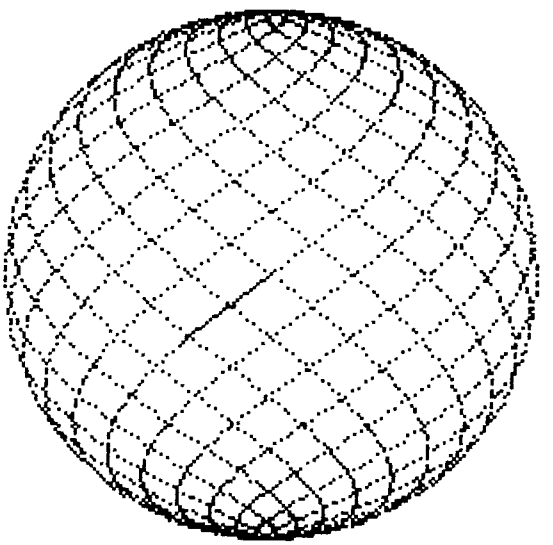


```
10 REM Bagdad
20 MODE 2:CLS
30 r=90:s=100:t=80:u=80:GOSUB 60:r=85:s=95:t=75:u=75:GOSUB 60:r=80:s=90:t=70:u=7
0:GOSUB 60:r=75:s=85:t=65:u=65:GOSUB 60
40 r=70:s=80:t=60:u=60:GOSUB 60
50 r=65:s=75:t=60:u=60:GOSUB 60:r=60:s=70:t=60:u=60:GOSUB 60:r=60:s=65:t=60:u=60
:GOSUB 60:r=60:s=60:t=60:u=60:GOSUB 60:END
60 FOR i=PI TO 3*PI  STEP PI/30:a=COS(i/2)*COS(i/2):b=SIN(i/2)*0.65
70 x=160+r*a:y=100+r*b:v=160-r*a:w=y:PLOT x,y:PLOT v,w:r=r-2:c=240+s*a:d=160+s*b
:e=240-s*a:f=d:PLOT c,d:PLOT e,f:s=s-2
80 g=80+t*a:h=130+t*b:j=80-t*a:l=h:PLOT g,h:PLOT j,l:t=t-2:m=50+g:n=40+h:o=50+j:
p=40+l:PLOT m,n:PLOT o,p:u=u-2
90 NEXT i:RETURN
```



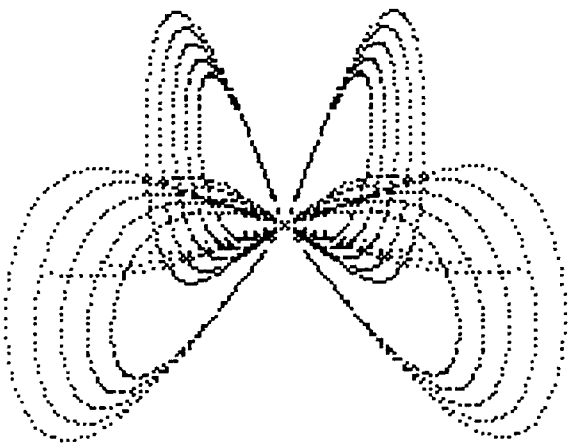
## Balle

```
1 REM Balle sur une ligne !...
10 MODE 1:CLS:FOR P=0 TO 31.55 STEP.01:X
=320+160*COS(2*P)*SIN(2.6*P):Y=200+160*S
IN(2*P):PLOT X,Y,3:NEXT
```



## Butterfly

```
10 REM Butterfly
20 MODE 1:CLS
30 r=90:GOSUB 40:r=85:GOSUB 40:r=80:GOSUB 40:r=75:GOSUB 40:r=70:GOSUB 40:END
40 FOR i=PI TO 2*PI-0.3 STEP 0.03
50 x=160+r*COS(i/2)*3:y=100-r*SIN(2*i)
60 x1=160-r*COS(i/2)*3:y1=y:x2=160+r*COS(i/2)*1.5:y2=100+r*SIN(2*i)
70 x3=160-r*COS(i/2)*1.5:y3=y2:PLOT x,y:PLOT x1,y1:PLOT x2,y2:PLOT x3,y3
80 r=r-1:NEXT i:RETURN
```

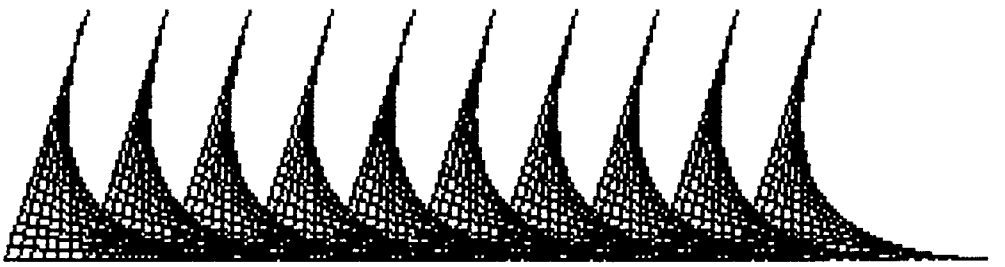


# 5/10.4.2

# Jeux de lignes

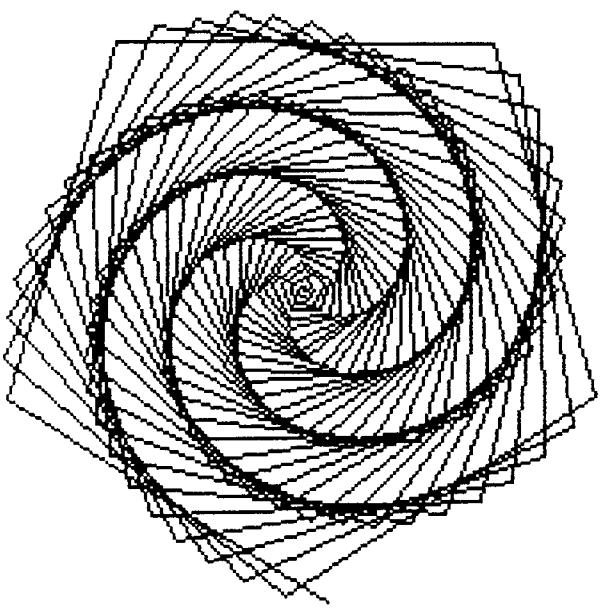
## La digue de la Rochelle

```
10 REM La digue de La Fochelle
20 MODE 2:CLS
30 x0=10:y0=150:r=160
40 a=2*PI/5*i:b=2*PI/5:x=x0+r*COS(a-b):y=y0-r*SIN(a-b):PLOT x0+r,y0:DRAW x0,y0:D
RAW x,y
50 FOR j=1 TO 10:FOR i=1 TO 20
60 x1=x0+r-(r/20)*i*COS(a):y1=y0:x2=x0+(r/20)*i*COS(a-b):y2=y0-(r/20)*i*SIN(a-b)
:PLOT x1,y1:DRAW x2,y2:NEXT i
70 x0=x0+r*COS(a-b):NEXT j
```



## Colimagone

```
1 REM Colimagone
10 CLS:MODE 1:PLOT 320,200:FOR X=0 TO 18
00 STEP 5:A=A+.55:C=COS(X)*A+320:L=SIN(X
)*A+200:DRAW C,L:NEXT
```



# 5/10.4.3

# Les espaces inconnus

### Notions mathématiques de base

Quelques notions sont nécessaires pour découvrir ces nouveaux êtres mathématiques errant dans des espaces immatériels.

Que les forts en maths, pour qui ces notions sont triviales, nous excusent, il faut bien que tout le monde comprenne...

Tout commence avec la question :

L'équation $x^2 = -1$ possède-t-elle une solution ?

A la réponse NON, partant du principe qu'on ne peut extraire la racine carrée d'un nombre négatif, on peut opposer la réponse OUI, pourquoi pas ?

Cette réponse affirmative, que nous vous demandons simplement d'admettre soulève le voile sur l'immensité étrange et passionnante des nombres complexes.

Admettons donc qu'il existe un nombre que nous noterons **i** et dont le carré est égal à $-1$.

Ce nombre d'un type nouveau appartient à l'ensemble des nombres complexes. L'ensemble des réels n'est qu'une partie infime de ce dernier.

Tout nombre complexe est repéré par son affixe. L'affixe **z** de tout nombre complexe est composée de 2 parties.
L'une appelée *partie réelle* sera notée **x** si vous le voulez bien alors que l'autre, nommée *partie imaginaire*, sera notée **y**.

Ces trois nombres sont alors reliés par la relation :

$$z = x + iy$$

Cela posé, on peut définir sur cet ensemble toutes les opérations classiques connues sur l'ensemble des réels, à savoir :

— *L'addition*

$z_1 = x_1 + iy_1$ et $z_2 = x_2 + iy_2$ entraînera

$$z_1 + z_2 = (x_1 + x_2) + i\,(y_1 + y_2)$$

La somme de deux nombres complexes est un nombre complexe dont la partie réelle est la somme des parties réelles et la partie imaginaire, la somme de deux parties imaginaires.

— De même, *la soustraction* de deux nombres complexes s'écrira

$$z_1 = x_1 + iy_1 \text{ et } z_2 = x_2 + iy_2$$

$$z_1 - z_2 = (x_1 - x_2) + i(y_1 - y_2)$$

— La *multiplication* de deux nombres complexes est un peu plus compliquée, en effet :

$$Z = X + iY = z_1 z_2 = (x_1 + iy_1)(x_2 + iy_2) = x_1 x_2 + ix_1 y_2 + ix_2 y_1 + i^2 y_1 y_2$$

souvenons-nous d'avoir décidé que $i^2 = -1$ alors :

$$Z = X + iY = (x_1 x_2 - y_1 y_2) + i(x_1 y_2 + x_2 y_1)$$ étonnant non ?

Les notations x et y ne sont pas sans rappeler les notations des coordonnées d'un point dans un **plan** muni d'un système d'axes Ox,Oy. Ce choix n'est en effet pas innocent puisqu'il va nous permettre de représenter un nombre complexe par un point dans un plan bien réel celui-ci.

Il suffira de représenter les parties imaginaires et réelles, respectivement par des points sur les axes d'ordonnée Oy et d'abscisse Ox.

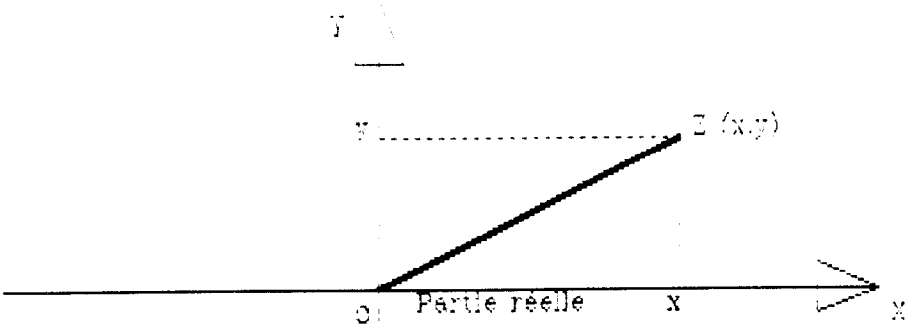



figure 1

Tout nombre complexe **z** sera alors représenté par un point **Z** de coordonnées **x** et **y**.

On peut cette fois définir le module d'un nombre complexe comme la longueur du vecteur **OZ**. Notons ce module **|z|**.

$$|z| = \sqrt{(x^2 + y^2)}$$

d'après Pythagore appliqué au triangle rectangle OZx.

Ces quelques notions sont pour l'instant suffisantes pour aborder ces espaces inconnus. N'hésitons plus et entrons alors dans le monde étonnant des nombres complexes.

### Passage du plan complexe à l'écran de l'Amstrad

Le principe des deux programmes qui suivent est très simple. Considérons une portion du plan complexe délimitée par le rectangle formé par les points $Z_{11}$, $Z_{22}$, $Z_{12}$, et $Z_{21}$.

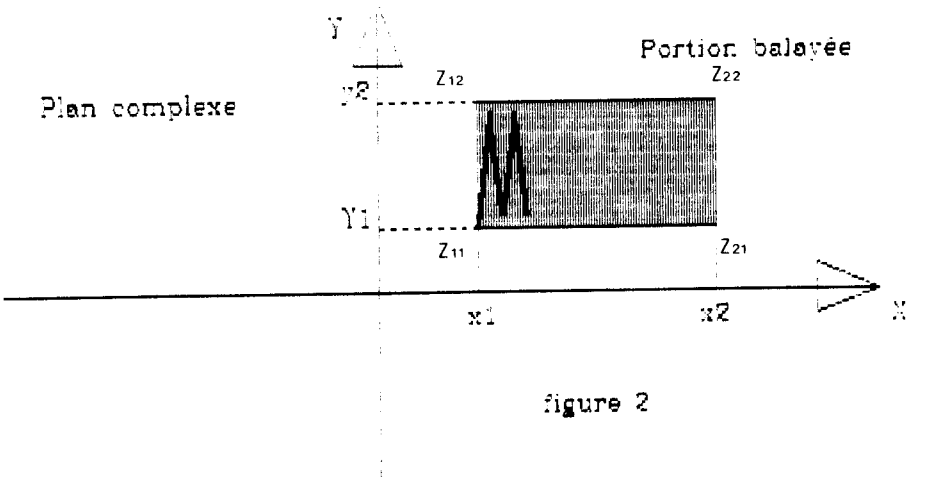



figure 2

Balayons systématiquement cette portion de bas en haut et de gauche à droite après avoir défini un accroissement **dx** sur l'axe des réels et **dy** sur l'axe imaginaire.

Faisons correspondre à chaque point ainsi balayé, un point de l'écran du CPC en écrivant que :

$$dx = (x_2 - x_1)/640 \text{ et } dy = (y_2 - y_1)/400$$

Le point courant dans le plan complexe correspondra alors au point-écran **XP, YP**.

Ces principes de balayage et de correspondance entre plan complexe et écran étant définis, nous pouvons alors faire subir à tout point du plan complexe toute sorte de torture et obtenir des graphismes assez merveilleux.

### Coucher de soleil imaginaire

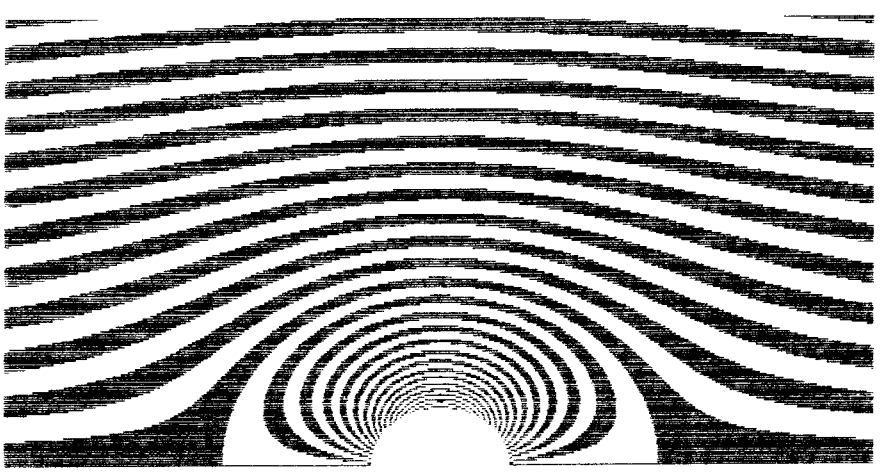
Décidons que pour tout point de la portion de plan complexe définie plus haut, nous calculions le module (ici **xx** en ligne 120) et que si ce module est inférieur à 0,1 nous passons au point suivant.

Dans le cas contraire, nous formons le nombre $f = y - y/xx$.

Si la partie entière de **f** multipliée par 10 est paire, nous traçons le point-écran correspondant, sinon nous passons au point suivant (ligne 150).

```
10 CLG
20 MODE 2
30 XP=0:YP=0
40 x1=-2:x2=2
50 y1=0:y2=(y1+(x2-x1)*400/640)
60 dx=(x2-x1)/640
70 dy=(y2-y1)/400
80 FOR x=x1 TO x2 STEP dx
90 XP=XP+1
100 FOR y=y1 TO y2 STEP dy
110 YP=YP+1
120 xx=x*x+y*y
130 IF xx<0.1 THEN 160
140 f=y-y/xx
150 IF (INT(f*10)MOD 2=0) THEN PLOT XP,YP
160 NEXT y
165 YP=0
170 NEXT x
```

Ce traitement barbare, dissociant les nombres complexes très particuliers obéissant à la règle, des autres, pauvres êtres informes ne méritant pas de figurer sur notre écran, va générer le coucher du soleil le plus bizarre que vous ayez jamais vu...



Coucher de soleil imaginaire.

On peut cette fois définir le module d'un nombre complexe comme la longueur du vecteur **OZ**. Notons ce module **|z|**.

$$|z| = \sqrt{(x^2 + y^2)}$$

d'après Pythagore appliqué au triangle rectangle OZx.

Ces quelques notions sont pour l'instant suffisantes pour aborder ces espaces inconnus. N'hésitons plus et entrons alors dans le monde étonnant des nombres complexes.

### Passage du plan complexe à l'écran de l'Amstrad

Le principe des deux programmes qui suivent est très simple. Considérons une portion du plan complexe délimitée par le rectangle formé par les points $Z_{11}$, $Z_{22}$, $Z_{12}$, et $Z_{21}$.

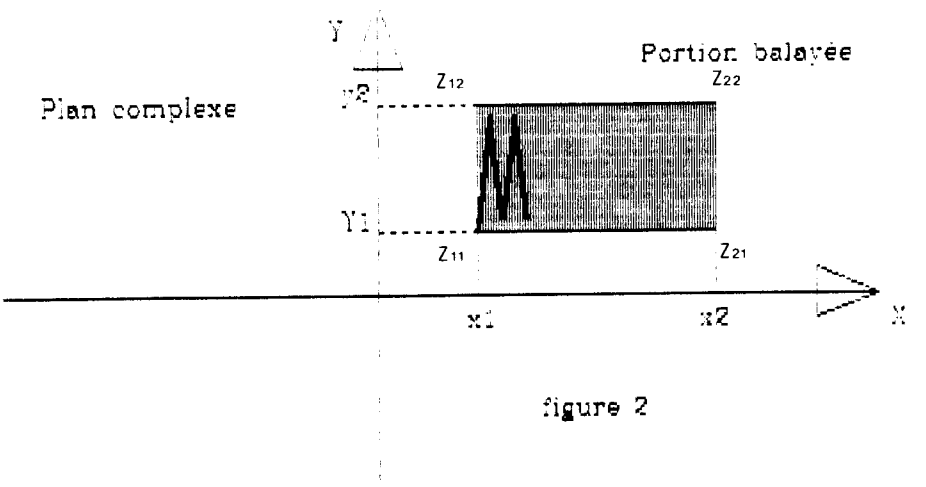



figure 2

Balayons systématiquement cette portion de bas en haut et de gauche à droite après avoir défini un accroissement **dx** sur l'axe des réels et **dy** sur l'axe imaginaire.

Faisons correspondre à chaque point ainsi balayé, un point de l'écran du CPC en écrivant que :

$$dx = (x_2 - x_1)/640 \text{ et } dy = (y_2 - y_1)/400$$

Le point courant dans le plan complexe correspondra alors au point-écran **XP, YP**.

Ces principes de balayage et de correspondance entre plan complexe et écran étant définis, nous pouvons alors faire subir à tout point du plan complexe toute sorte de torture et obtenir des graphismes assez merveilleux.

## Coucher de soleil imaginaire

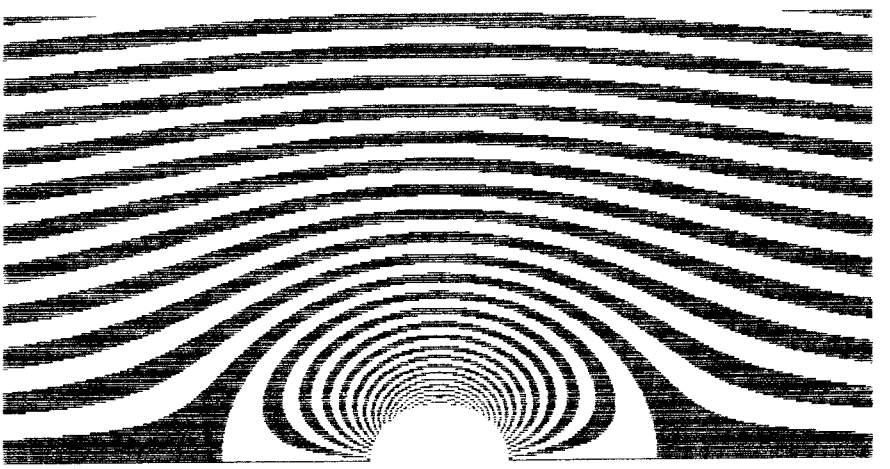
Décidons que pour tout point de la portion de plan complexe définie plus haut, nous calculions le module (ici **xx** en ligne 120) et que si ce module est inférieur à 0,1 nous passons au point suivant.

Dans le cas contraire, nous formons le nombre **f = y − y/xx**.

Si la partie entière de **f** multipliée par 10 est paire, nous traçons le point-écran correspondant, sinon nous passons au point suivant (ligne 150).

```
10 CLS
20 MODE 2
30 XP=0:YP=0
40 x1=-2:x2=2
50 y1=0:y2=(y1+(x2-x1)*400/640)
60 dx=(x2-x1)/640
70 dy=(y2-y1)/400
80 FOR x=x1 TO x2 STEP dx
90 XP=XP+1
100 FOR y=y1 TO y2 STEP dy
110 YP=YP+1
120 xx=x*x+y*y
130 IF xx<0.1 THEN 160
140 f=y-y/xx
150 IF (INT(f*10)MOD 2=0) THEN PLOT XP,YP
160 NEXT y
165 YP=0
170 NEXT x
```

Ce traitement barbare, dissociant les nombres complexes très particuliers obéissant à la règle, des autres, pauvres êtres informes ne méritant pas de figurer sur notre écran, va générer le coucher du soleil le plus bizarre que vous ayez jamais vu...



Coucher de soleil imaginaire.

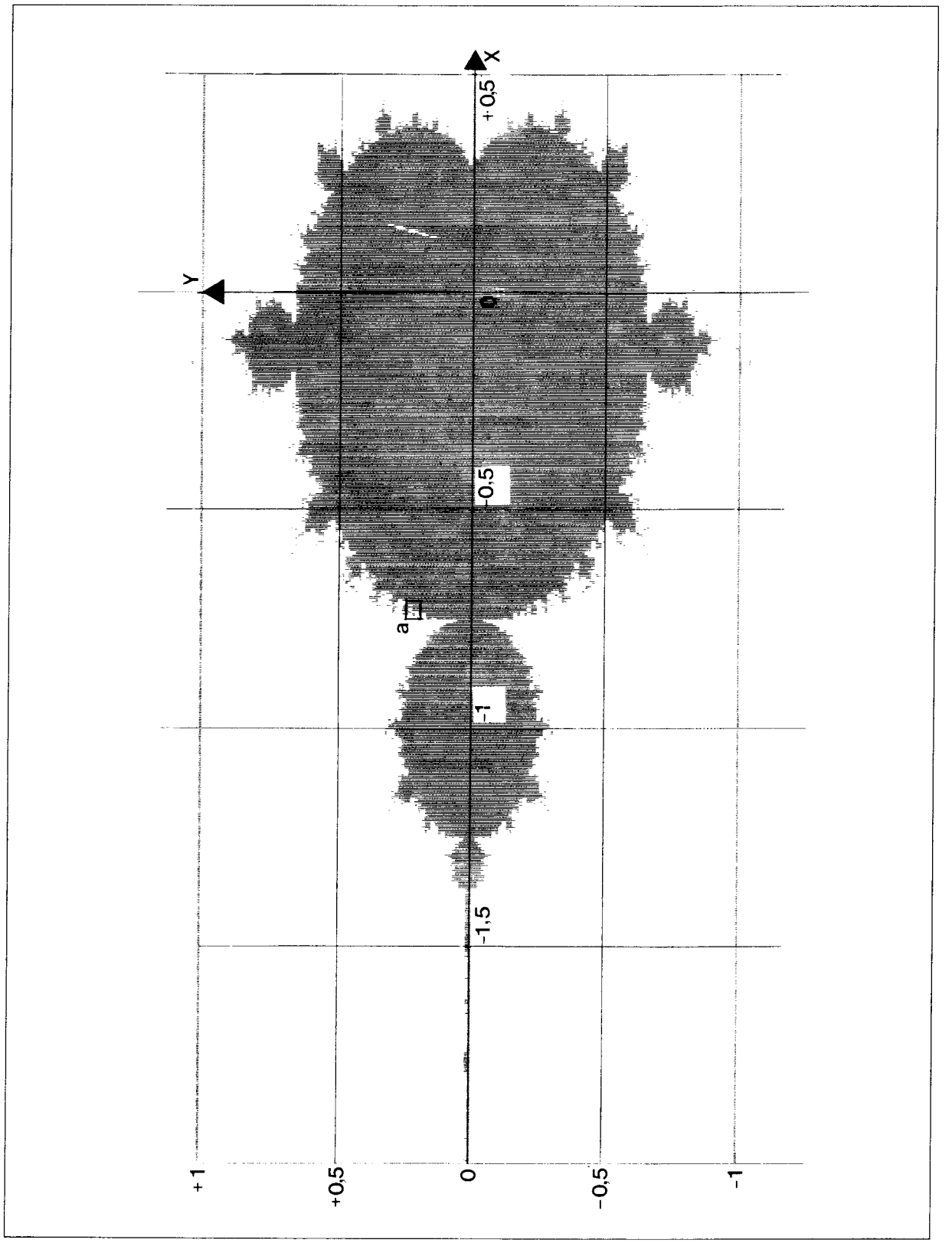
## Variations sur Mandelbrot

Benoît Mandelbrot, mathématicien français qui s'est illustré au centre de recherches T. Watson d'IBM, par ses travaux sur la géométrie fractale, ouvre pour nous la porte d'une immensité curieuse et magnifique : l'ensemble de Mandelbrot. Cet ensemble est constitué des nombres complexes **c** qui, quel que soit le nombre d'itérations appliquées à l'algorithme $z = z^2 + c$ restent finis.

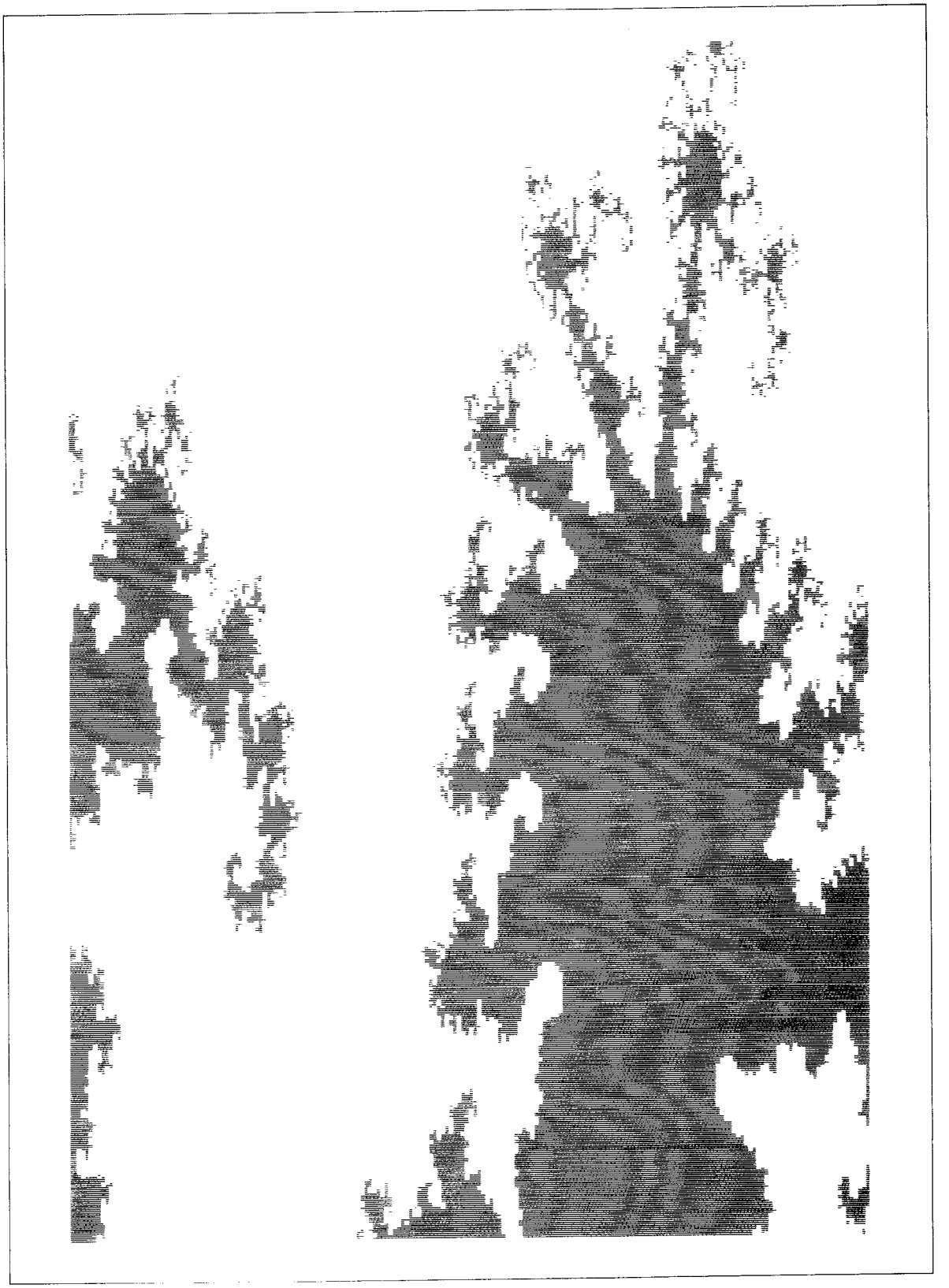
```
10 REM (c) WEKA (jpc 1988)
15 INK 1,13:INK 0,0:BORDER 0
20 CLS
30 MODE 2
40 XP=0:YP=0
50 INPUT"x1,x2";x1,x2
60 INPUT"y1,y2";y1,y2
65 INPUT"Nombre d'iterations N";N
66 CLS
70 dx=(x2-x1)/640
80 dy=(y2-y1)/400
90 FOR x=x1 TO x2 STEP dx
100 XP=XP+1
110 FOR y=y1 TO y2 STEP dy
120 x0=0:y0=0
130 YP=YP+1
140 FOR i=1 TO N
150 xx=x0*x0-y0*y0+x
160 y0=2*x0*y0+y:x0=xx
170 yy=(x0*x0+y0*y0)
180 IF yy>4 THEN 210
190 NEXT i
200 PLOT XP,YP
210 NEXT y
220 YP=0
230 NEXT x
```

Choisissons parmi ceux-ci, ceux dont le module reste inférieur à 2. Et représentons-les sur notre écran en suivant le principe défini plus haut.

Ces nombres sont alors représentés sur la figure suivante. Cet ensemble a été dessiné avec seulement 50 itérations : plus ce nombre sera grand, plus les détails seront finis (dans la limite de définition de votre écran) mais aussi plus le temps de calcul sera long, passant de quelques heures à quelques... jours.




L'ensemble de Mandelbrot dans le plan complexe (50 itérations).



Détail de l'ensemble de Mandelbrot dans le plan complexe $x_1 = -0,745 - x_2 = -0,72 - y_1 = 0,216 - y_2 = 0,25$.

Chaque zone est en principe intéressante à explorer mais la frontière de l'ensemble est plus riche lorsque le nombre d'itérations est faible.

La figure de détail en page 9 correspond à un agrandissement de la zone notée **[a]** sur la figure précédente. On pourrait agrandir encore cette zone et y découvrir de nouvelles formes. Il n'y a aucune limite à cela pas même celle de la dimension atomique. Seul votre Amstrad imposera ses limites de précision et de temps de calcul à votre vertigineuse soif d'entrer dans l'infiniment petit.

### Les algorithmes de Barry Martin

D'autres mathématiciens comme Barry Martin se sont inspirés de la démarche de B. Mandelbrot. A savoir, faire des itérations successives sur une même fonction. Mais cette fois, cette fonction s'appliquait sur des nombres réels et sur une seule valeur initiale. On ne balaye pas un plan comme précédemment, mais on part d'un point unique pour en calculer une série d'autres.

Dans les deux programmes qui suivent, ce principe est appliqué sur deux fonctions différentes.

Le premier programme « Hopalong » utilise les fonctions suivantes :

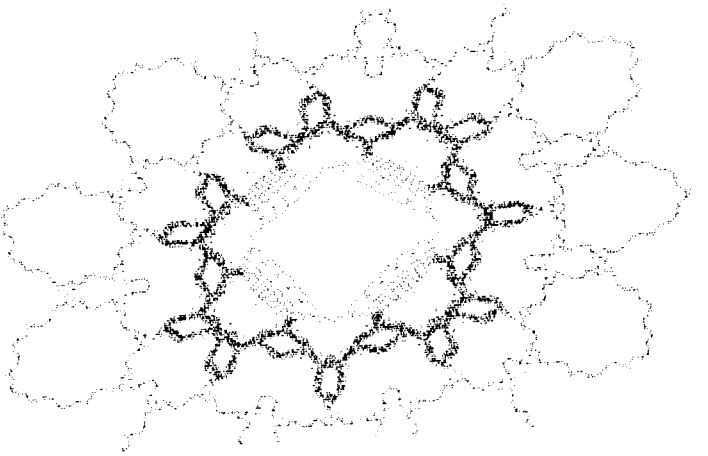
$$x = y - SGN(x) * \sqrt{|(b*x - c)|}$$
$$y = a - x$$

La fonction SGN(x) (signe de x) prend la valeur $-1$ si **x** est négatif et 1 si **x** est positif. L'étonnant pour ce type de programme est que le graphisme obtenu varie énormément suivant les valeurs des paramètres **a**, **b** et **c**.

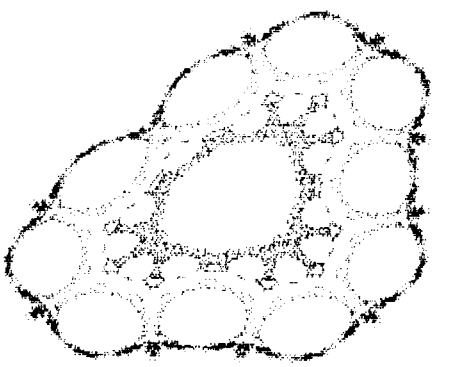
Le nombre d'itérations influe sur le niveau de détail du graphisme (voir figures page 9). Là encore, vous devrez attendre quelques heures avant de voir votre écran se couvrir de ces merveilles bizarres.

```
10 REM Algorithme   HOPALONG
20 MODE 2:CLS
30 INPUT"a,b,c";a,b,c
31 INPUT"N",N
32 INPUT"Echelle",E
35 INPUT"coordonnees du centre x,y";cx,cy
50 LOCATE 1,1:x=0:y=0
60 FOR i=1 TO N
70 PLOT x*E+cx,y*E+cy
80 xx=y-SGN(x)*(ABS(b*x-c))^0.5
90 yy=a-x
100 x=xx:y=yy
110 NEXT i
```

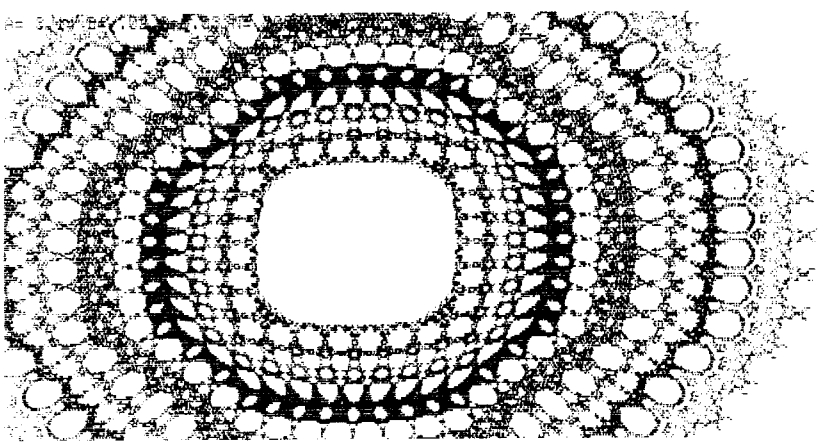
Diversités d'Hopalong :



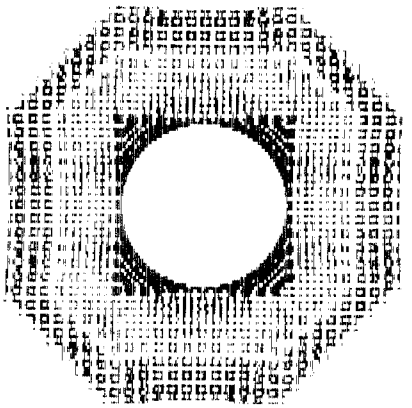
A = 3,14 - B = 0,3 - C = 0,3 - N = 10 000



A = 3,14 - B = – 1 - C = 0,03 - N = 10 000

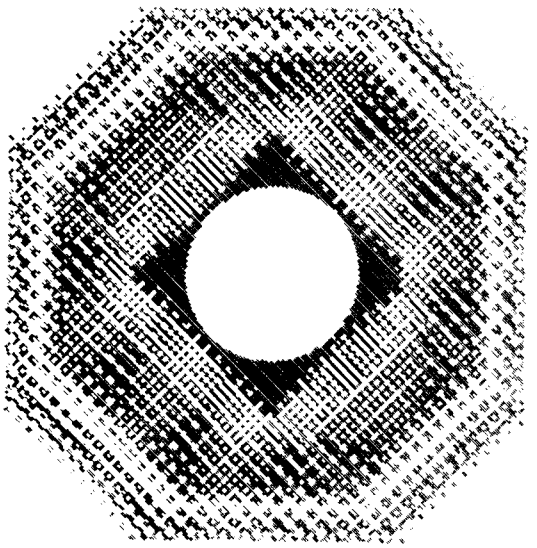


A = 3,14 - B = 0,08 - C = 0,03 - N = 60 000

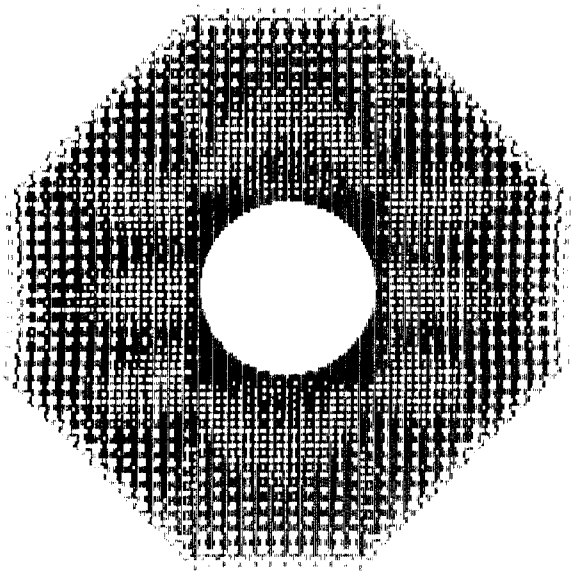


Evolution du graphisme en fonction du nombre d'itérations N

a = − 200 - b = 0,1 - c = − 80 - N = 120 135



Hopalong :
a = − 200 - b = 0,1 - c = − 80 - N = 423 424



Hopalong :
a = − 200 - b = 0,1 - c = − 80 - N = 1 359 620

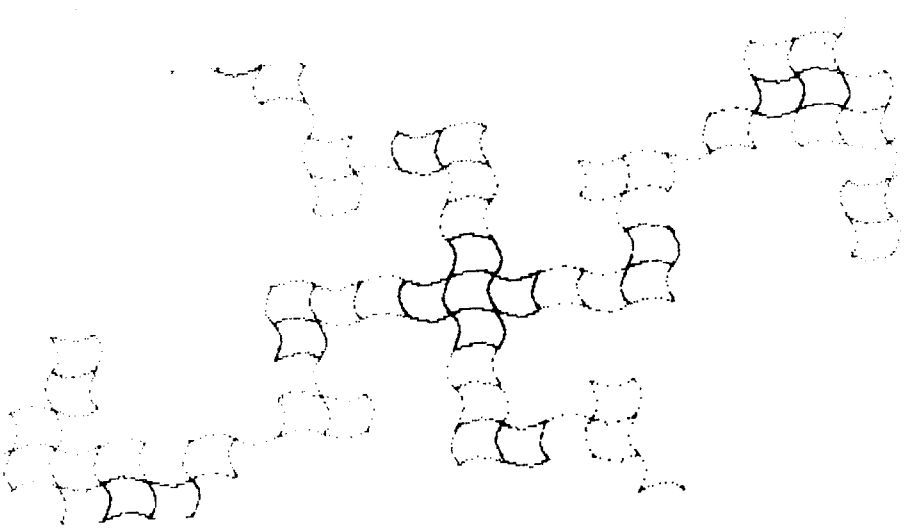
Le dernier programme proposé est composé comme le précédent ; mais utilise les fonctions :

$$x = y - SIN\ (x)$$
$$y = a - x$$

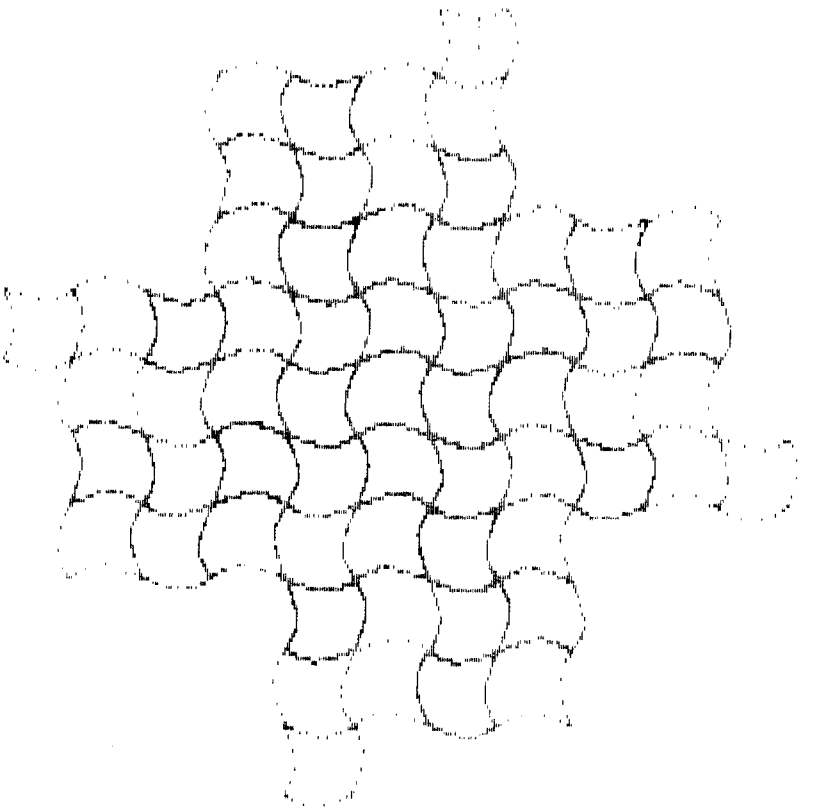
Un seul paramètre entre en jeu ici **a**. Les meilleurs résultats seront obtenus pour **a** très voisin de $\pi$.

Vous pourrez choisir sur ce dernier programme, l'échelle de votre dessin ainsi que le point de celui-ci à afficher au centre de votre écran.

```
10 REM Algorithme de Barry MARTIN
20 MODE 2:CLS
30 INPUT"a";a
40 INPUT"N",N
50 INPUT"Echelle",E
60 INPUT"coordonnees du centre x,y";cx,cy
70 CLS:PRINT"Algorithme de Barry MARTIN"
80 LOCATE 1,1:x=0:y=0
90 FOR i=1 TO N
100 PLOT x*E+cx,y*E+cy
110 xx=y-SIN(x)
120 yy=a-x
130 x=xx:y=yy
140 NEXT i
```



Algorithme de Barry Martin : a = 3,12 N = 100 000



Algorithme de Barry Martin : a = 3,09 N = 300 000