5/9.2

Définition et animation de sprites

Si vous êtes amateur de jeux d'arcades, vous savez très certainement ce que sont les sprites. Pour les néophites, les sprites (ou lutins) sont des objets graphiques qui se déplacent sur l'écran sans l'effacer. Dans ce paragraphe, nous allons :

– étudier un générateur de sprites en **MODE 1** ;
– étudier une RSX (¦SPRITE) qui permettra aux programmeurs Basic de déplacer très simplement jusqu'à 8 sprites en même temps sur un écran en **MODE 1** ;
– utiliser la RSX ¦ SPRITE pour effectuer une démonstration du déplacement d'un sprite en **MODE 1**.

UN PEU DE THÉORIE

Comme vous le savez (voir Partie 5 Chapitre 7), l'écran peut être considéré comme un bloc mémoire dans lequel chaque octet représente deux, quatre ou huit points élémentaires (MODES 0, 1 et 2). En ce qui nous concerne (MODE 1), chaque octet représente quatre points élémentaires ainsi codés :

Bit	7	6	5	4	3	2	1	0
Stylo	2	2	2	2	1	1	1	1
Point	0	1	2	3	0	1	2	3

Interprétation du tableau :

– chaque point élémentaire peut avoir quatre couleurs ;
– lorsque tous les bits correspondant à un point élémentaire valent 0, le point a la couleur du fond de l'écran **(PEN 0)** ;

– pour allumer le point élémentaire le plus à gauche avec la couleur 1, il faut stocker la valeur **&H08** (bit 3 à 1) dans la mémoire concernée ;
– pour allumer le point le plus à gauche avec la couleur 2, il faut stocker la valeur **&H80** (bit 7 à 1) dans la mémoire concernée ;
– pour allumer le point le plus à gauche avec la couleur 3, il faut stocker la valeur **&H88** (bits 7 et 3 à 1) dans la mémoire concernée ;
– le codage des 3 autres points élémentaires suit la même démarche logique :

Point 1 :		Point 2 :		Point 3 :	
Couleur	**Valeur**	**Couleur**	**Valeur**	**Couleur**	**Valeur**
0	&H00	0	&H00	0	&H00
1	&H04	1	&H02	1	&H01
2	&H40	2	&H20	2	&H10
3	&H44	3	&H22	3	&H11

La RSX que nous allons étudier permet de manipuler des sprites de 8 points élémentaires sur 8, ce qui représente 16 octets : 2 octets pour représenter une ligne de 8 points élémentaires, et 8 lignes élémentaires.

Les quatre couleurs possibles pourront être modifiées à volonté à l'aide d'instructions Basic INK (voir Partie 4).

LE GÉNÉRATEUR DE SPRITES

Avant de parler de la RSX !SPRITE, nous allons étudier un générateur de sprites qui vous permettra de créer vos propres sprites de 8 points élémentaires sur 8, et comportant jusqu'à quatre couleurs.

Comment utiliser le programme

Saisissez et exécuter le programme Basic suivant :

```
1000 '=======================
1010 ' Definition de sprites
1020 '=======================
1030 '
1040 '- - - - - - - - - - - - - - - - - -
1050 ' Tableaux de donnees du programme
1060 '- - - - - - - - - - - - - - - - - -
1070 '
1080 DIM t(8,8) 'Tableau du sprite
1090 DIM t2(4,4) 'Valeur des couleurs
1100 t2(1,1)=0:t2(1,2)=0:t2(1,3)=0:t2(1,4)=0
1110 t2(2,1)=8:t2(2,2)=4:t2(2,3)=2:t2(2,4)=1
1120 t2(3,1)=128:t2(3,2)=64:t2(3,3)=32:t2(3,4)=16
1130 t2(4,1)=136:t2(4,2)=68:t2(4,3)=34:t2(4,4)=17
1140 '
1150 '- - - - - - - - - -
1160 ' Trace de la grille
1170 '- - - - - - - - - -
1180 '
1190 MODE 1
1200 GRAPHICS PEN 1
1210 FOR i=1 TO 9
1220   MOVE i*20,100
1230   DRAW i*20,260
1240   MOVE 20,80+i*20
1250   DRAW 180,80+i*20
1260 NEXT i
1270 '
1280 '- - - - - - - - - - - -
1290 ' Ecran de presentation
1300 '- - - - - - - - - - - -
1310 '
1320 LOCATE 20,1
1330 PRINT"Couleurs possibles"
1340 FOR i=0 TO 3
1350   LOCATE 23,i+5
1360   PAPER i
1370   PRINT" "
1380 NEXT i
1390 PAPER 0
1400 LOCATE 15,12
1410 PRINT "Donnees correspondantes"
1420 LOCATE 15,15
1430 PRINT"00 00 00 00 00 00 00 00"
1440 LOCATE 15,16
1450 PRINT"00 00 00 00 00 00 00 00"
1460 '
1470 LOCATE 1,22
1480 PRINT"   Deplact = Fleches, Couleur = Copy"
1490 PRINT"   Pixel   = Enter,   Fin      = Q"
```

```
1500 '
1510 '- - - - - - - - - - - - - - - - - -
1520 ' Boucle principale de definition
1530 '- - - - - - - - - - - - - - - - - -
1540 '
1550 PAPER 0
1560 LOCATE 22,5
1570 PRINT">"
1580 cc=1 'Couleur courante=1
1590 l=1:c=1:sl=1:sc=1:GOSUB 1710
1600 a$=INKEY$:IF a$="" THEN 1600
1610 a=ASC(a$)
1620 sc=c:sl=l 'Sauvegarde ligne et colonne
1630 IF a=243 THEN c=c+1:GOSUB 1710 'Vers la droite
1640 IF a=242 THEN c=c-1:GOSUB 1710 'Vers la gauche
1650 IF a=240 THEN l=l-1:GOSUB 1710 'Vers le haut
1660 IF a=241 THEN l=l+1:GOSUB 1710 'Vers le bas
1670 IF a=224 THEN GOSUB 1950 'Changement de couleur
1680 IF a=13 THEN GOSUB 2080 'Affichage d'un pixel
1690 IF UPPER$(a$)<>"Q" THEN 1600
1700 END
1710 '
1720 '- - - - - - - - - - - - - - - - - -
1730 ' Affichage de la case courante
1740 '- - - - - - - - - - - - - - - - - -
1750 '
1760 IF l=9 THEN l=8
1770 IF l=0 THEN l=1
1780 IF c=9 THEN c=8
1790 IF c=0 THEN c=1
1800 MASK 255
1810 MOVE 20+(sc-1)*20,260-(sl-1)*20
1820 DRAW 40+(sc-1)*20,260-(sl-1)*20
1830 DRAW 40+(sc-1)*20,240-(sl-1)*20
1840 DRAW 20+(sc-1)*20,240-(sl-1)*20
1850 DRAW 20+(sc-1)*20,260-(sl-1)*20
1860 MASK 85
1870 MOVE 20+(c-1)*20,260-(l-1)*20
1880 DRAW 40+(c-1)*20,260-(l-1)*20
1890 DRAW 40+(c-1)*20,240-(l-1)*20
1900 DRAW 20+(c-1)*20,240-(l-1)*20
1910 DRAW 20+(c-1)*20,260-(l-1)*20
1920 MASK 255
1930 '
1940 RETURN
1950 '
1960 '- - - - - - - - - - - - - - -
1970 ' Changement de couleur
1980 '- - - - - - - - - - - - - - -
1990 '
```

```
2000 sc=cc 'Sauvegarde couleur courante
2010 cc=cc+1
2020 IF cc=5 THEN cc=1
2030 LOCATE 22,sc+4
2040 PRINT " "
2050 LOCATE 22,cc+4
2060 PRINT ">"
2070 RETURN

2080 '
2090 '- - - - - - - - - - - -
2100 ' Affichage d'un pixel
2110 '- - - - - - - - - - - -
2120 '
2130 GRAPHICS PEN cc-1
2140 MOVE 20+(c-1)*20,260-(l-1)*20
2150 DRAW 40+(c-1)*20,260-(l-1)*20
2160 DRAW 40+(c-1)*20,240-(l-1)*20
2170 DRAW 20+(c-1)*20,240-(l-1)*20
2180 DRAW 20+(c-1)*20,260-(l-1)*20
2190 MOVE 25+(c-1)*20,255-(l-1)*20
2200 FILL cc-1
2210 GRAPHICS PEN 1
2220 MASK 85
2230 MOVE 20+(c-1)*20,260-(l-1)*20
2240 DRAW 40+(c-1)*20,260-(l-1)*20
2250 DRAW 40+(c-1)*20,240-(l-1)*20
2260 DRAW 20+(c-1)*20,240-(l-1)*20
2270 DRAW 20+(c-1)*20,260-(l-1)*20
2280 MOVE 25+(c-1)*20,255-(l-1)*20
2290 MASK 255
2300 t(l,c)=cc-1
2310 '
2320 IF c<5 THEN s=1 ELSE s=2
2330 octet=(l-1)*2+s
2340 IF octet <9 THEN LOCATE 12+octet*3,15 ELSE LOCATE octet
*3-12,16
2350 tot=0
2360 IF c<5 THEN FOR z=1 TO 4:tot=tot+t2(t(l,z)+1,z):NEXT z

2370 IF c>4 THEN FOR z=5 TO 8:tot=tot+t2(t(l,z)+1,z-4):NEXT
z
2380 PRINT HEX$(tot,2)
2390 RETURN
```

Une grille de 8 points sur 8 apparaît sur l'écran. C'est dans cette grille que vous définirez vos sprites.

La partie haute de l'écran laisse apparaître les quatres couleurs disponibles. Une flèche vers la droite (>) indique quelle est la couleur courante, c'est-à-dire la couleur de tracé. Pour changer la couleur courante, appuyez autant de fois que nécessaire sur la touche <Copy>.

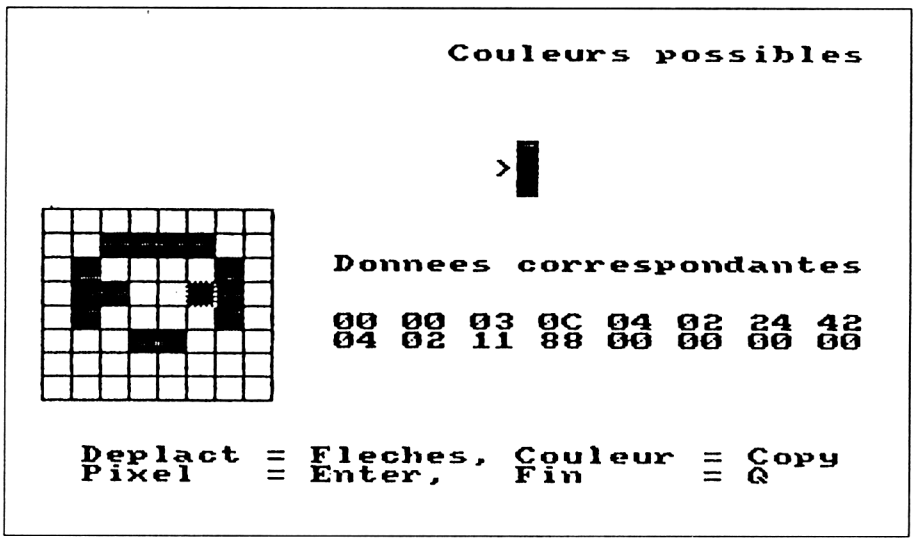
La partie centrale de l'écran affiche les données hexadécimales correspondant au sprite en cours de création. Lorsque vous avez défini un sprite, notez ces données sur un bout de papier. Elles seront par la suite intégrées au programme d'animation...

Enfin, la partie basse de l'écran résume les diverses actions possibles.

Pour définir un sprite, respectez les étapes suivantes :

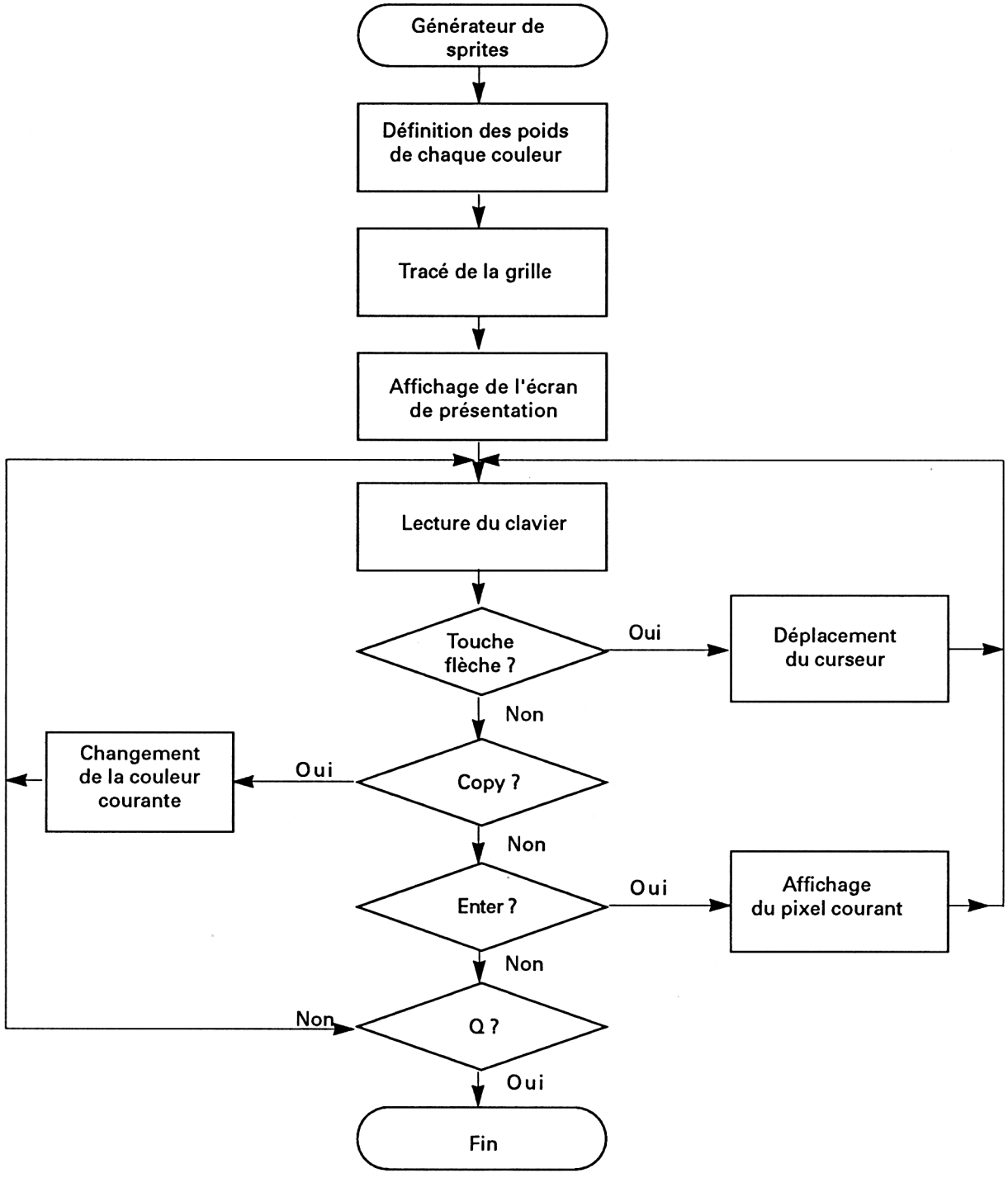
– positionnez le « curseur » sur la case de votre choix (le curseur est symbolisé par une case discontinue) à l'aide des touches flèches ;
– choisissez éventuellement la couleur de la case courante à l'aide de la touche <Copy> du clavier ;
– appuyez sur la touche <Enter> du clavier pour colorier une case courante ;
– recommencez les opérations qui viennent d'être décrites jusqu'à ce que le sprite soit entièrement défini ;
– notez alors les données hexadécimales correspondantes.

Exemple de sprite :



Le programme en détail

La logique du programme obéit à l'ordinogramme suivant :



La ligne 1080 définit le tableau dans lequel seront stockées les données (allumé/éteint) des points du sprite :

DIM t(8,8) 'Tableau du sprite

La ligne 1090 définit le tableau qui contient le poids de chaque point élémentaire en fonction de sa couleur :

DIM t2(4,4) 'Valeur des couleurs

Chaque octet contient la définition de quatre points élémentaires successifs. Les quatres points sont ainsi codés :

	Points			
Couleurs	1	2	3	4
0	0	0	0	0
1	8	4	2	1
2	128	64	32	16
3	136	68	34	17

Les données de ce tableau s'interprètent ainsi :

– lorsque le point élémentaire 1 est allumé avec la couleur 0 (couleur de fond), l'octet de définition est inchangé ;
– lorsque le point élémentaire 1 est allumé avec la couleur 1, l'octet de définition doit être incrémenté de 8 ;
– lorsque le point élémentaire 1 est allumé avec la couleur 2, l'octet de définition doit être incrémenté de 128 ;
– lorsque le point élémentaire 1 est allumé avec la couleur 3, l'octet de définition doit être incrémenté de 136 ;
– de ce qui précède, vous pouvez déduire vous même le codage des points élémentaires 2 à 4.

Les données de ce tableau sont stockées dans t2 entre les lignes 1100 et 1130 :

```
1100 t2(1,1)=0:t2(1,2)=0:t2(1,3)=0:t2(1,4)=0
...
1130 t2(4,1)=136:t2(4,2)=68:t2(4,3)=34:t2(4,4)=17
```

Le tracé de la grille est effectué entre les lignes 1190 et 1260 :

```
1190 MODE 1
1200 GRAPHICS PEN 1
1210 FOR i=1 TO 9
```

```
1220    MOVE i*20,100
1230    DRAW i*20,260
1240    MOVE 20,80+i*20
1250    DRAW 180,80+i*20
1260 NEXT i
```

La ligne 1200 spécifie que la grille doit être tracée avec la couleur PEN 1.

Les diverses données qui accompagnent la grille sont affichées sur l'écran entre les lignes 1320 et 1480 :

– les quatre couleurs possibles sont affichées en haut et à gauche de l'écran à l'aide d'une boucle FOR NEXT dans laquelle la couleur du papier *(paper)* est modifiée :

```
1320 LOCATE 20,1
1330 PRINT "Couleurs possibles"
1340 FOR i=0 TO 3
1350    LOCATE 23,i+5
1360    PAPER i
1370    PRINT " "
1380 NEXT i
```

– les données hexadécimales correspondant au sprite sont initialisées à 0 et affichées à l'aide d'instructions PRINT :

```
1400 LOCATE 15,12
1410 PRINT  "Données correspondantes"
1420 LOCATE 15,15
1430 PRINT  "00 00 00 00 00 00 00 00"
1440 LOCATE 15,16
1450 PRINT  "00 00 00 00 00 00 00 00"
```

– enfin, le rappel des commandes est affiché en bas de l'écran à l'aide d'instructions PRINT :

```
1470 LOCATE 1,22
1480 PRINT"    Deplact = Fleches, Couleur = Copy"
1490 PRINT"    Pixel   = Enter,   Fin     = Q"
```

Le programme se poursuit par la boucle principale de définition dans laquelle les diverses touches pressées sur le clavier sont interprétées.

Après diverses initialisations, le sous-programme situé en ligne 1710 est appelé. Ce sous-programme affiche la case courante dans la grille :

```
1590 1=1:c=1:sl=1:sc=1:GOSUB 1710
```

La case courante affichée, le programme se met en attente d'une action au clavier :

```
1600 a$=INKEY$:IF a$="" THEN 1600
```

Lorsqu'une touche est pressée, son code ASCII est stocké dans la variable a :

```
1610 a=ASC(a$)
```

Les ligne et colonne courantes sont sauvegardées dans les variables sc et sl :

```
1620 sc=c:sl= l 'Sauvegarde ligne et colonne
```

Lorsque la touche pressée est une touche flèche, la variable l ou c est modifiée en conséquence :

```
1630 IF a=243 THEN c=c+1:GOSUB 1710 'Vers la droite
1640 IF a=242 THEN c=c-1:GOSUB 1710 'Vers la gauche
1650 IF a=240 THEN 1=1-1:GOSUB 1710 'Vers le haut
1660 IF a=241 THEN 1=1+1:GOSUB 1710 'Vers le bas
```

Lorsque la touche <Copy> est pressée, le sous-programme de changement de couleur est activé :

```
1670 IF a=224 THEN GOSUB 1950 'Changement de la couleur
```

Lorsque la touche <Enter> est pressée, le sous-programme d'affichage d'un pixel est activé :

```
1680 IF a=13 THEN GOSUB 2080 'Affichage d'un pixel
```

Enfin, lorsque la touche "Q" est pressée, le programme prend fin. Dans tous les autres cas, la touche est ignorée et le programme se met en attente d'une autre action au clavier :

```
1690 IF UPPER$(a$)<>"Q" THEN 1600
```

Le listing se termine par les trois sous-programmes dont nous venons de parler :

Les lignes 1710 à 1940 affichent la case courante dans la grille.
Les paramètres nécessaires en entrée sont les nouvelles ligne (l) et colonne (c) de la case courante.

Si ces données sont incorrectes (par exemple suite à l'appui sur la touche flèche vers la droite alors que le curseur se trouve dans la dernière colonne de la grille), elles sont révisées :

```
1760 IF l=9 THEN l=8
1770 IF l=0 THEN l=1
1780 IF c=9 THEN c=8
1790 IF c=0 THEN c=1
```

La case courante est repérée dans la grille par une ligne discontinue. L'ancienne case courante (de coordonnées sc,sl) est affichée avec une ligne continue :

```
1800 MASK 255
1810 MOVE 20+(sc-1)*20,260-(sl-1)*20
1820 DRAW 40+(sc-1)*20,260-(sl-1)*20
1830 DRAW 40+(sc-1)*20,240-(sl-1)*20
1840 DRAW 20+(sc-1)*20,240-(sl-1)*20
1850 DRAW 20+(sc-1)*20,260-(sl-1)*20
```

et la nouvelle case courante est affichée avec une ligne discontinue :

```
1800 MASK 85
1810 MOVE 20+(c-1)*20,260-(l-1)*20
1820 DRAW 40+(c-1)*20,260-(l-1)*20
1830 DRAW 40+(c-1)*20,240-(l-1)*20
1840 DRAW 20+(c-1)*20,240-(l-1)*20
1850 DRAW 20+(c-1)*20,260-(l-1)*20
```

Les lignes 1950 à 2070 permettent de changer la couleur courante.

La couleur courante est sauvegardée puis incrémentée :

```
2000 sc=cc 'Sauvegarde couleur courante
2010 cc=cc+1
```

Lorsque la nouvelle couleur est égale à 5 (le nombre de couleurs disponibles étant égal à 4), la variable **cc** est initialisée à 1 pour pointer sur la première couleur disponible :

```
2020 IF cc=5 THEN cc=1
```

Le signe **">"** pointant sur l'ancienne couleur courante est effacé et le nouveau signe est affiché :

```
2030 LOCATE 22,sc+4
2040 PRINT " "
2050 LOCATE 22,cc+4
2060 PRINT ">"
```

Les lignes 2080 à 2390

– affichent un point élémentaire dans la grille lorsque la touche <Enter> est pressée,
– modifient et affichent les valeurs hexadécimales correspondantes.

Le remplissage de la case sélectionnée se fait à l'aide de l'instruction FILL :

```
2130 GRAPHICS PEN cc-1
```

```
2140 MOVE 20+(sc-1)*20,260(sl-1)*20
...
2180 DRAW 20+(sc-1)*20,260-(sl-1)*20
2190 FILL cc-1
2200 GRAPHICS PEN 1
2220 MASK 85
2230 MOVE 20+(sc-1)*20,260-(sl-1)*20
...
2280 DRAW 20+(sc-1)*20,260-(sl-1)*20
2290 MASK 255
```

La couleur de la case affichée est mémorisée dans le tableau t :

```
2300 t(1,c)=cc-1
```

Le curseur est positionnné sur la donnée hexadécimale à modifier :

```
2320 IF c<5 THEN s=1 ELSE s=2
2330 octet=(l-1)*2+s
2340 IF octet <9 THEN LOCATE 12+octet*3,15 ELSE
LOCATE octet*3-12,16
```

La nouvelle valeur hexadécimale est calculée à partir des tableaux t et t2 :

```
2350 tot=0
2360 IF c<5 THEN FOR z=1 TO 4:tot=tot+t2(t(l,z)+1,z):NEXT z
2370 IF c>4 THEN FOR z=5 TO 8:tot=tot+t2(y(l,z)+1,z-4):NEXT z
```

et affichée sur deux digits :

```
2380 PRINT HEX$(tot,2)
```

LA RSX D'AFFICHAGE ¦SPRITE

Comment afficher un objet sur l'écran et le déplacer sans effacer le décor ? Eh bien tout simplement en mémorisant le décor qui se trouve sous le sprite et en le réaffichant lors du prochain déplacement. Ce raisonnement simpliste est la base de la RSX ¦SPRITE.

Lors du premier affichage du sprite sur l'écran, aucun décor n'a encore été mémorisé. Il est donc inutile d'afficher la zone de mémoire correspondant au décor. Par contre, il faut sauvegarder le décor qui se trouve sous le sprite avant de l'afficher. Lors des futurs déplacements de sprite, il faudra :

- restituer le décor sauvegardé ;
- mémoriser le nouveau décor ;
- afficher le sprite à la nouvelle position.

Comment utiliser la RSX

La RSX est bien entendu écrite en Assembleur. En voici le listing :

```
1                               ORG  9000H
2                               LOAD 9000H
3                       ;------------------------------
4                       ; RSX SPRITE
5                       ; Format : |SPRITE,N,X,Y
6                       ; Entree : N=Numero du sprite
7                       ;          X=Abscisse du sprite
8                       ;          Y=Ordonnee du sprite
9                       ; Sortie : Affichage du sprite
10                      ;------------------------------
11                      ;
12                      ;
13                      ;--------------------------------
14                      ; Declaration des constantes
15                      ; et des variables du programme
16                      ;--------------------------------
17                      ;
18                      LOGEXT:     EQU  0BCD1H         ;KL LOG EXT
19                      DOTPOS:     EQU  0BC1DH         ;SCR DOT POS
20                      NEXTLINE:   EQU  0BC26H         ;SCR NEXT LINE
21                      SAUVDEC:    EQU  8100H          ;@ sauv decor
22                      PREMAFF:    EQU  8200H          ;Indic 1er aff
23                      NUMERO:     DS   1              ;Numero du sprite
24                      ABS:        DS   2              ;Abscisse du sprite
25                      ORD:        DS   2              ;Ordonne du sprite
26                      ADR:        DS   2              ;Adresse ecran
27                      ESPRIT:     DS   2              ;Adresse sprite
28                      ANCXY:      DS   32             ;Adresse anc coord
29                      BUF:        DS   4              ;Zone RAM pour LOG EXT
30 902D 3290            PTRTAB:     DW   TABLE          ;Pointeur TABLE
31 902F C34390                      JP   SPRITE         ;Affichage du sprite
```

```
32 9032 53505249 TABLE:      DB    "SPRIT"
32 9036 54
33 9037 C5                   DB    "E"+80H
34 9038 00                   DB    0                 ;Fin de table
35              ;
36              ;---------------------------------
37              ; Definition de la RSX
38              ;---------------------------------
39              ;
40              DEFRSX:      EQU   $                 ;Point d'entree
41 9039 012D90               LD    BC,PTRTAB         ;Ptr table definition
42 903C 212990               LD    HL,BUF            ;Buffer pour LOG EXT
43 903F CDD1BC               CALL  LOGEXT            ;Definition de la RSX
44 9042 C9                   RET
45              ;
46              ;---------------------------------
47              ; Traitement de SPRITE
48              ;---------------------------------
49              ;
50              SPRITE:      EQU   $                 ;Point d'entree
51 9043 DD6601               LD    H,(IX+1)
52 9046 DD6E00               LD    L,(IX+0)
53 9049 220390               LD    (ORD),HL
54 904C DD6603               LD    H,(IX+3)
55 904F DD6E02               LD    L,(IX+2)
56 9052 220190               LD    (ABS),HL          ;Sauv abscisse
57 9055 DD7E04               LD    A,(IX+4)
58 9058 320090               LD    (NUMERO),A        ;Sauv No Sprite
59              ;
```

```
60                ;- - - - - - - - - - -
61                ; Test 1er affichage
62                ;- - - - - - - - - - -
63                ;
64 905B 210082              LD   HL,PREMAFF
65 905E 3A0090              LD   A,(NUMERO)
66 9061 3D                  DEC  A
67 9062 1600                LD   D,0
68 9064 5F                  LD   E,A
69 9065 19                  ADD  HL,DE
70 9066 7E                  LD   A,(HL)          ;Indic 1er aff
71 9067 B7                  OR   A
72 9068 CAC190              JP   Z,SAUVEDE       ;Sauvegarde decor
73                ;
74                ;- - - - - - - - - - - -
75                ; Affichage ancien decor
76                ;- - - - - - - - - - - -
77                ;
78                ; Calcul adresse ecran
79                ;
80 906B 210990              LD   HL,ANCXY
81 906E 3A0090              LD   A,(NUMERO)
82 9071 3D                  DEC  A
83 9072 87                  ADD  A,A
84 9073 87                  ADD  A,A
85 9074 1600                LD   D,0
86 9076 5F                  LD   E,A
87 9077 19                  ADD  HL,DE           ;Ordonnee
88 9078 5E                  LD   E,(HL)
89 9079 23                  INC  HL
90 907A 56                  LD   D,(HL)
```

```
 91 907B 23                 INC   HL
 92 907C 4E                 LD    C,(HL)          ;Abscisse
 93 907D 23                 INC   HL
 94 907E 46                 LD    B,(HL)
 95 907F 60                 LD    H,B
 96 9080 69                 LD    L,C
 97 9081 CD1DBC             CALL  DOTPOS
 98 9084 220590             LD    (ADR),HL        ;@ ecran
 99              ;
100              ; Calcul adresse decor
101              ;
102 9087 210081             LD    HL,SAUVDEC      ;@ 1er decor
103 908A 111000             LD    DE,16
104 908D 3A0090             LD    A,(NUMERO)
105              BIS:       EQU   $
106 9090 3D                 DEC   A
107 9091 2803               JR    Z,FININC        ;Fin incrementation
108 9093 19                 ADD   HL,DE
109 9094 18FA               JR    BIS             ;Boucle incrementation
110              FININC:    EQU   $
111 9096 220790             LD    (ESPRIT),HL     ;@ decor
112              ;
113              ; Affichage
114              ;
115 9099 ED5B0790           LD    DE,(ESPRIT)
116 909D 2A0590             LD    HL,(ADR)
117 90A0 3E08               LD    A,8
118 90A2 47                 LD    B,A
119              BISDEC1:
LD A,(D
```

```
120 90A3 1A                 LD    A,(DE)
121 90A4 77                 LD    (HL),A
122 90A5 CD26BC             CALL  NEXTLINE          ;SCR NEXT LINE
123 90A8 13                 INC   DE
124 90A9 13                 INC   DE
125 90AA 10F7               DJNZ  BISDEC1
126
127 90AC ED5B0790           LD    DE,(ESPRIT)
128 90B0 13                 INC   DE
129 90B1 2A0590             LD    HL,(ADR)
130 90B4 23                 INC   HL
131 90B5 3E08               LD    A,8
132 90B7 47                 LD    B,A
133             BISDEC2:
LD A,(D
134 90B8 1A                 LD    A,(DE)
135 90B9 77                 LD    (HL),A
136 90BA CD26BC             CALL  NEXTLINE          ;SCR NEXT LINE
137 90BD 13                 INC   DE
138 90BE 13                 INC   DE
139 90BF 10F7               DJNZ  BISDEC2
140             ;
141             ;- - - - - - - - - - -
142             ; Sauvegarde du decor
143             ;- - - - - - - - - - -
144             ;
145             SAUVEDE:    EQU   $
146             ;
147             ; Sauvegarde des coordonnees
148             ;
149 90C1 210990             LD    HL,ANCXY
150  90C4  340090            ID    A, (NUMERO)
```

```
151 90C7 3D                 DEC   A
152 90C8 87                 ADD   A,A
153 90C9 87                 ADD   A,A
154 90CA 1600               LD    D,0
155 90CC 5F                 LD    E,A
156 90CD 19                 ADD   HL,DE
157 90CE ED5B0190           LD    DE,(ABS)
158 90D2 73                 LD    (HL),E
159 90D3 23                 INC   HL
160 90D4 72                 LD    (HL),D
161 90D5 ED5B0390           LD    DE,(ORD)
162 90D9 23                 INC   HL
163 90DA 73                 LD    (HL),E
164 90DB 23                 INC   HL
165 90DC 72                 LD    (HL),D
166             ;
167             ; Calcul de l'adresse ecran
168             ;
169 90DD ED5B0190           LD    DE,(ABS)
170 90E1 2A0390             LD    HL,(ORD)
171 90E4 CD1DBC             CALL  DOTPOS          ;SCR DOT POSITION
172 90E7 220590             LD    (ADR),HL        ;Sauvegarde adresse
173             ;
174             ; Sauvegarde
175             ;
176 90EA 210081             LD    HL,SAUVDEC      ;@ 1er decor
177 90ED 111000             LD    DE,16
178 90F0 3A0090             LD    A,(NUMERO)
179             BIS2:       EQU   $
```

```
 180 90F3 3D                    DEC   A
 181 90F4 2803                  JR    Z,FININC2          ;Fin incrementation
 182 90F6 19                    ADD   HL,DE
 183 90F7 18FA                  JR    BIS2               ;Boucle incrementatior
 184              FININC2:      EQU   $
 185 90F9 220790                LD    (ESPRIT),HL        ;@ decor
 186              ;
 187 90FC ED5B0790              LD    DE,(ESPRIT)
 188 9100 2A0590                LD    HL,(ADR)
 189 9103 3E08                  LD    A,8
 190 9105 47                    LD    B,A
 191              BISSAV1:
LD A,(H
 192 9106 7E                    LD    A,(HL)
 193 9107 12                    LD    (DE),A
 194 9108 CD26BC                CALL  NEXTLINE           ;SCR NEXT LINE
 195 910B 13                    INC   DE
 196 910C 13                    INC   DE
 197 910D 10F7                  DJNZ  BISSAV1
 198              ;
 199 910F ED5B0790              LD    DE,(ESPRIT)
 200 9113 13                    INC   DE
 201 9114 2A0590                LD    HL,(ADR)
 202 9117 23                    INC   HL
 203 9118 3E08                  LD    A,8
 204 911A 47                    LD    B,A
 205              BISSAV2:
LD A,(H
 206 911B 7E                    LD    A,(HL)
 207 911C 12                    LD    (DE),A
 208 911D CD26BC                CALL  NEXTLINE           ;SCR NEXT LINE
 209 9120 13                    INC   DE
```

```
212                 ;
213                 ;- - - - - - - - - - -
214                 ; Affichage du sprite
215                 ;- - - - - - - - - - -
216                 ;
217 9124 210080             LD    HL,8000H          ;@ 1er sprite
218 9127 111000             LD    DE,16
219 912A 3A0090             LD    A,(NUMERO)
220                 BIS3:    EQU   $
221 912D 3D                 DEC   A
222 912E 2803               JR    Z,FININC3         ;Fin incrementation
223 9130 19                 ADD   HL,DE
224 9131 18FA               JR    BIS3              ;Boucle incrementati
225                 FININC3: EQU   $
226 9133 220790             LD    (ESPRIT),HL
227                 ;
228                 ; Affichage
229                 ;
230 9136 ED5B0790           LD    DE,(ESPRIT)
231 913A 2A0590             LD    HL,(ADR)
232 913D 3E08               LD    A,8
233 913F 47                 LD    B,A
234                 BISCOL1:
LD A,(D
235 9140 1A                 LD    A,(DE)
236 9141 77                 LD    (HL),A
237 9142 CD26BC             CALL  NEXTLINE          ;SCR NEXT LINE
238 9145 13                 INC   DE
239 9146 13                 INC   DE
```

```
240 9147 10F7                   DJNZ BISCOL1
241                   ;
242 9149 ED5B0790               LD   DE,(ESPRIT)
243 914D 13                     INC  DE
244 914E 2A0590                 LD   HL,(ADR)
245 9151 23                     INC  HL
246 9152 3E08                   LD   A,8
247 9154 47                     LD   B,A
248                   BISCOL2:
LD A,(D
249 9155 1A                     LD   A,(DE)
250 9156 77                     LD   (HL),A
251 9157 CD26BC                 CALL NEXTLINE          ;SCR NEXT LINE
252 915A 13                     INC  DE
253 915B 13                     INC  DE
254 915C 10F7                   DJNZ BISCOL2
255                   ;
256                   ;
257                   FIN:      EQU  $                ;Fin du programme
258 915E C9                     RET
259                   ;
260                   ;---------------------------------
261                   ; Zone des sous-programmes
262                   ;---------------------------------
263                   ;
264                             END
```

```
ABS         9001 ADR        9005 ANCXY      9009 BUF        9029
BIS         9090 BISDEC1    90A3 BISDEC2    90B8 BIS2       90F3
BISSAV1     9106 BISSAV2    911B BIS3       912D BISCOL1    9140
BISCOL2     9155 DOTPOS     BC1D DEFRSX     9039 ESPRIT     9007
FININC      9096 FININC2    90F9 FININC3    9133 FIN        915E
LOGEXT      BCD1 NEXTLINE   BC26 NUMERO     9000 ORD        9003
PREMAFF     8200 PTRTAB     902D SAUVDEC    8100 SPRITE     9043
SAUVEDE     90C1 TABLE      9032
```

La version Assembleur est intéressante pour comprendre le fonctionnement des sprites, mais ce sont surtout les données hexadécimales correspondantes qui seront utilisées en programmation Basic.

Pour utiliser cette RSX , procédez comme suit :

– Définissez la RSX en exécutant le programme situé en **&H9039** :

```
CALL &H9039
```

– Avant d'afficher pour la première fois le sprite **N** (où N est compris entre 1 et 8) sur l'écran, initialisez la mémoire **&H8200+N-1** à zéro par un **POKE**. Par exemple pour le sprite 1 :

```
POKE &H8200,0
```

puis affichez le sprite en spécifiant les coordonnées d'affichage. Par exemple pour afficher le sprite 1 aux coordonnées **x=100, y=50** :

```
|SPRITE,1,100,50
```

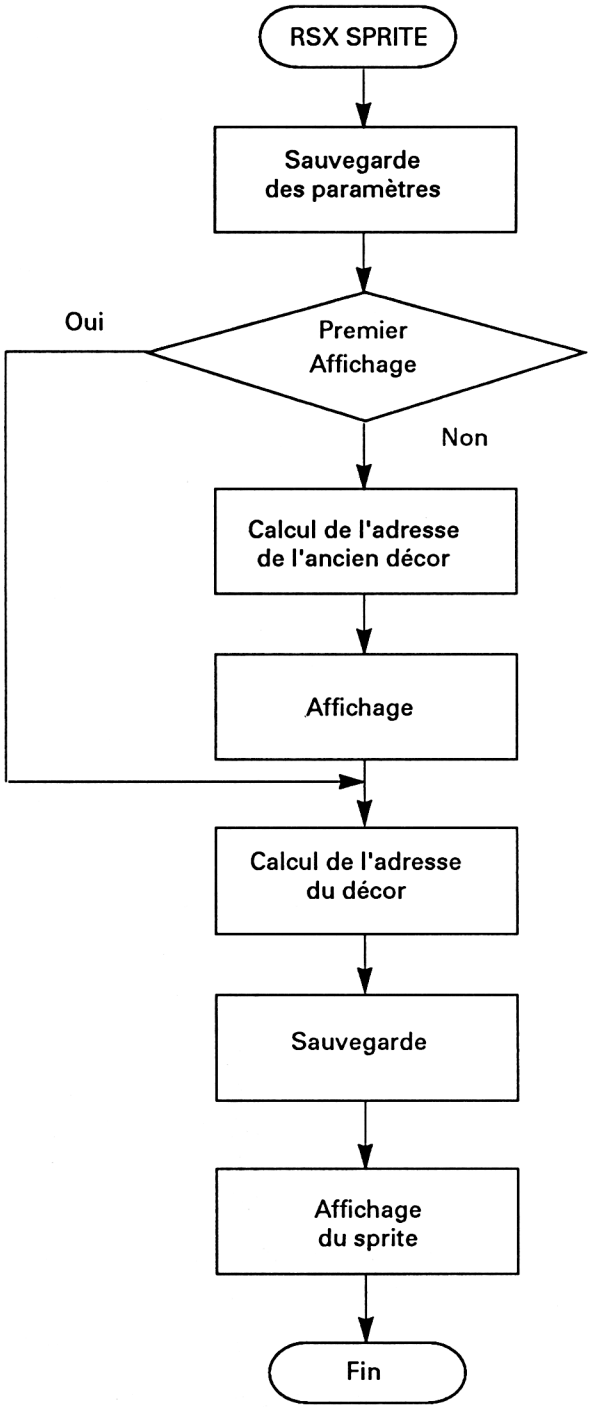
– Lorsque le sprite **N** (où N est compris entre 1 et 8) a été affiché une fois, initialisez la mémoire **&H8200+N-1** à un par un **POKE**. Par exemple pour le sprite 1 :

```
POKE &H8200,0
```

puis affichez le sprite comme dans l'exemple ci-dessus.

Le programme en détail

La logique de la RSX apparaît dans l'ordinogramme suivant :



La RSX est définie à l'aide du petit programme situé entre les lignes 40 et 44 :

```
DEFRSX:   EQU     $             ;Point d'entrée
          LD      BC,PTRTAB     ;Ptr table définition
          LD      HL,BUF        ;Buffer pour LOG EXT
          CALL    LOGEXT        ;Définition de la RSX
```

Lorsque le mot **SPRITE** sera reconnu par le Basic, le programme situé à l'étiquette **SPRITE** sera exécuté. En effet, le tableau **PTRTAB** pointe sur l'étiquette **SPRITE** et contient le mot clé **SPRITE** :

```
PTRTAB:   DW      TABLE         ;Pointeur TABLE
          JP      SPRITE        ;Affichage du sprite
TABLE:    DB      "SPRIT"
          DB      "E"+80H
          DB      0             ;Fin de table
```

Le programme situé à l'étiquette **SPRITE** débute par la mémorisation des données qui lui sont passées :

```
SPRITE:   EQU     $             ;Point d'entrée
          LD      H, (IX+1)
          LD      L, (IX+0)
          LD      (ORD) ,HL
          LD      H, (IX+3)
          LD      L, (IX+2)
          LD      (ABS) ,HL     ;Sauv abscisse
          LD      A, (IX+4)
          LD      (NUMERO) ,A
```

La variable située en &H8200+NUMERO est ensuite examinée pour déterminer si le sprite est affiché pour la première fois :

```
          LD      HL,PREMAFF
          LD      A, (NUMERO)
          DEC     A
          LD      D,0
          LD      E,A
          ADD     HL,DE
          LD      A, (HL)       ;Indic 1er aff
          OR      A
```

Lorsque le sprite est affiché pour la première fois, la phase d'affichage du décor est sautée :

```
          JP      Z,SAUVDE      ;Sauvegarde decor
```

A partir du second affichage du sprite, le décor mémorisé à l'ancienne position du sprite est réaffiché. Les coordonnées de ce décor se

trouvent dans la zone mémoire qui commence en ANCXY :

– ordonnée en ANCXY + (NUMERO)*4
– abscisse en ANCXY + (NUMERO)*4 + 2

Ces coordonnées sont extraites de la mémoire et stockées dans les registres DE et HL :

```
LD      HL,ANCXY
LD      A,(NUMERO)
DEC     A
ADD     A,A
ADD     A,A       → (NUMERO)*4
LD      D,0
LD      E,A
ADD     HL,DE     → ANCXY + (NUMERO)*4
LD      E,(HL)
INC     HL
LD      D,(HL)    → DE = (ANCXY + (NUMERO)*4)
INC     HL
LD      C,(HL)
INC     HL
LD      B,(HL)    → BC = (ANCXY + (NUMERO)*4 + 2)
LD      H,B
LD      L,C
```

La routine SCR DOT POSITION du Firmware est ensuite appelée pour connaître l'adresse de l'octet qui correspond à ces coordonnées :

```
CALL    DOTPOS
```

Cette routine renvoie l'adresse recherchée dans le registre HL. Le programme stocke cette adresse dans la variable ADR :

```
LD      (ADR),HL
```

L'adresse mémoire du décor est calculée par une simple addition : l'adresse du premier décor débute en SAUVDEC, et chaque décor occupe 16 octets. L'adresse qui nous intéresse est donc calculée de la manière suivante :

adresse = SAUVDEC + (NUMERO)*16

Comme la multiplication ne fait pas partie des opérations de base du Z80, nous avons eu recours à une boucle pour calculer (NUMERO)*16 :

```
        LD HL,SAUVDEC       ;Adresse du premier décor
        LD DE,16
        LD A,(NUMERO)
BIS:    EQU     $
```

```
            DEC     A
            JR      Z,FININC        ;Fin de l'incrémentation
            ADD     HL,DE
            JR      BIS             ; Boucle d'incrémentation
FININC:     EQU     $
            LD      (ESPRIT) ,HL    ;Adresse du décor
```

L'affichage des 16 octets du décor se fait à l'aide de deux boucles. La première affiche les 8 octets correspondant à la partie gauche du décor :

```
            LD      DE, (ESPRIT)
            LD      HL, (ADR)
            LD      A,8
            LD      B,A
BISDEC1 :
            LD      A, (DE)         → octet lu en mémoire
            LD      (HL) ,A         → affiché sur l'écran
            CALL    NEXTLINE        → passage à la ligne
                                      suivante sur l'écran
            INC     DE              → ... et en mémoire
            INC     DE
            DJNZ    BISDEC1         → boucle d'affichage
```

Remarquez :

– la macro **NEXTLINE** du Firmware qui permet de connaître l'adresse de l'octet qui représente les 4 points situés en-dessous des 4 points courants ;

– l'utilisation pratique de l'instruction **DJNZ** qui exécute la boucle 8 fois, jusqu'à ce que le registre B soit nul.

La seconde boucle affiche les 8 octets correspondant à la partie droite du décor. Elle utilise la même logique que la précédente. Nous n'y reviendrons pas.

Le nouveau décor est sauvegardé selon le procédé inverse. Les données sont lues sur l'écran à partir des coordonnées **ABS** et **ORD**, et sauvegardées en mémoire (lignes 140 à 211). Nous n'y reviendrons pas.

Le programme se termine par l'affichage du sprite aux coordonnées **ABS,ORD**. Cet affichage est très similaire à l'affichage du décor. Il occupe les lignes 212 à 254.

Le programme se termine par une instruction **RET** qui redonne le contrôle au Basic.

DÉMONSTRATION DE L'ANIMATION D'UN SPRITE

Le court programme présenté dans ce paragraphe montre comment utiliser la RSX SPRITE dans un programme Basic. Son but est uniquement démonstratif. Aucune action n'est donc demandée à la personne qui l'utilise, si ce n'est l'appui sur une touche quelquonque du clavier qui met fin à son exécution.

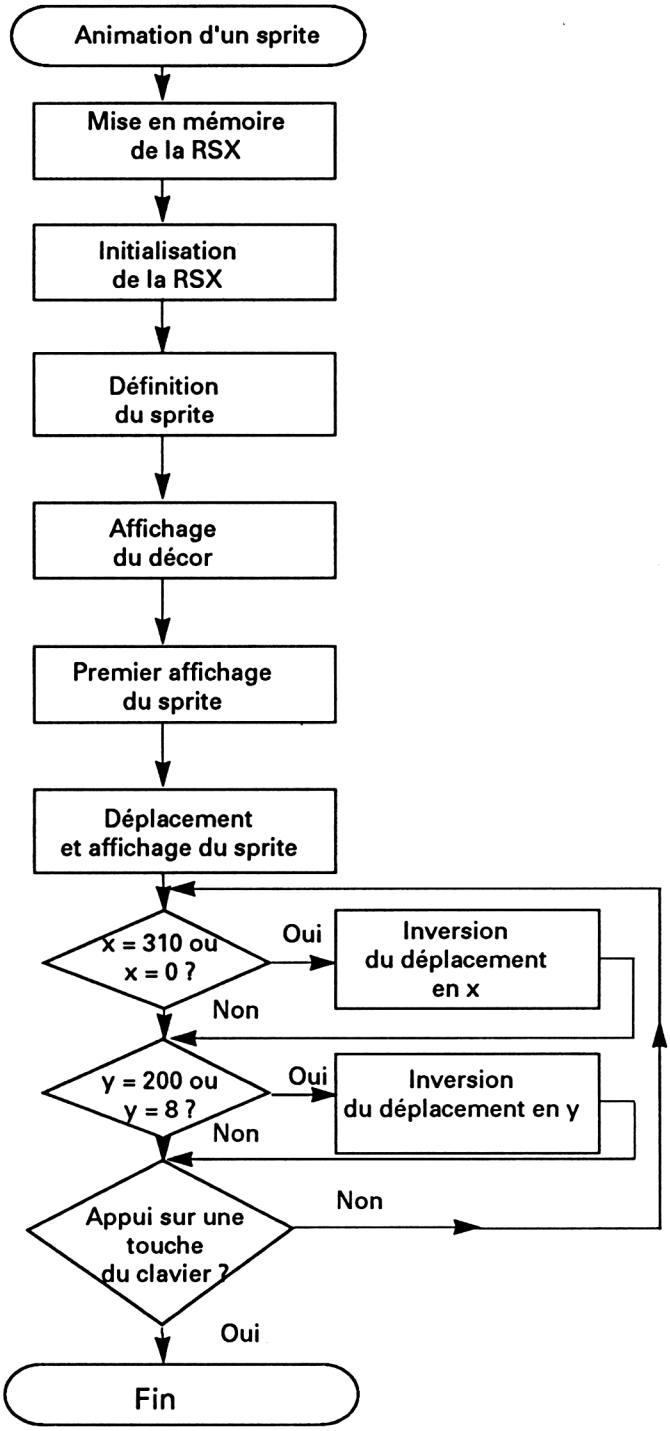
Son listing est le suivant :

```
1000 '=============================================
1010 ' Demonstration de l'animation d'un sprite
1020 '=============================================
1030 '
1040 FOR i=&9000 TO &915E
1050   READ a$
1060   a$="&"+a$
1070   a=VAL(a$)
1080   POKE i,a
1090 NEXT i
1100 '
1110 '---------------------------
1120 ' Initialisation de la RSX
1130 '---------------------------
1140 '
1150 CALL &9039
1160 '
1170 '-------------------------
1180 ' Definition du sprite
1190 '-------------------------
1200 '
1210 FOR i=&8000 TO &8007
1220   IF (i/2)=INT(i/2) THEN POKE i,&FF ELSE POKE i
1230 NEXT i
1240 '
1250 FOR i=&8008 TO &800F
1260   IF (i/2)=INT(i/2) THEN POKE i,0 ELSE POKE i,&I
1270 NEXT i
1280 '-----------
1290 ' Animation
1300 '-----------
1310 '
1320 MODE 1
1330 FOR i=1 TO 1000
1340   PRINT"-";
1350 NEXT i
1360 POKE &8200,0:|SPRITE,1,80,100
1370 POKE &8200,1
```

```
1380 x=80
1390 y=100
1400 dx=2
1410 dy=2
1420 a$=INKEY$
1430 IF (x=310) OR (x=0) THEN dx=-dx
1440 IF (y=200) OR (y=8) THEN dy=-dy
1450 x=x+dx
1460 y=y+dy
1470 |SPRITE,1,x,y
1480 IF a$="" THEN 1420
1490 MODE 2
1500 END
1510 '
1520 '-----------------------------------------------
1530 ' Programme assembleur d'affichage de sprites
1540 '-----------------------------------------------
1550 '
1560 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1570 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1580 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,32,90,C3
1590 DATA 43,90,53,50,52,49,54,C5,0,1,2D,90,21,29,90,CD
1600 DATA D1,BC,C9,DD,66,1,DD,6E,0,22,3,90,DD,66,3,DD
1610 DATA 6E,2,22,1,90,DD,7E,4,32,0,90,21,0,82,3A,0
1620 DATA 90,3D,16,0,5F,19,7E,B7,CA,C1,90,21,9,90,3A,0
1630 DATA 90,3D,87,87,16,0,5F,19,5E,23,56,23,4E,23,46,60
1640 DATA 69,CD,1D,BC,22,5,90,21,0,81,11,10,0,3A,0,90
1650 DATA 3D,28,3,19,18,FA,22,7,90,ED,5B,7,90,2A,5,90
1660 DATA 3E,8,47,1A,77,CD,26,BC,13,13,10,F7,ED,5B,7,90
1670 DATA 13,2A,5,90,23,3E,8,47,1A,77,CD,26,BC,13,13,10
1680 DATA F7,21,9,90,3A,0,90,3D,87,87,16,0,5F,19,ED,5B
1690 DATA 1,90,73,23,72,ED,5B,3,90,23,73,23,72,ED,5B,1
1700 DATA 90,2A,3,90,CD,1D,BC,22,5,90,21,0,81,11,10,0
1710 DATA 3A,0,90,3D,28,3,19,18,FA,22,7,90,ED,5B,7,90
1720 DATA 2A,5,90,3E,8,47,7E,12,CD,26,BC,13,13,10,F7,ED
1730 DATA 5B,7,90,13,2A,5,90,23,3E,8,47,7E,12,CD,26,BC
1740 DATA 13,13,10,F7,21,0,80,11,10,0,3A,0,90,3D,28,3
1750 DATA 19,18,FA,22,7,90,ED,5B,7,90,2A,5,90,3E,8,47
1760 DATA 1A,77,CD,26,BC,13,13,10,F7,ED,5B,7,90,13,2A,5
1770 DATA 90,23,3E,8,47,1A,77,CD,26,BC,13,13,10,F7,C9,0
```

Le programme en détail

La logique du programme apparaît dans l'ordinogramme suivant :



Le programme débute par la mise en mémoire de la RSX ¦SPRITE à l'aide d'une boucle **FOR NEXT** :

```
1040 FOR i=&9000 TO &915E
1050        READ a$
1060        a$="&"+a$
1070        a=VAL(a$)
1080        POKE i ,a
1090 NEXT i
```

La RSX est ensuite initialisée :

```
1150 CALL &9039
```

Le programme se poursuit par la définition du sprite en mémoire à partir de l'adresse **&8000** :

```
1210 FOR i=&8000 TO &8007
1220        IF (i/2)=INT(i/2) THEN POKE i,&FF ELSE POKE i,0
1230 NEXT i
1240 '
1250 FOR i=&8008 TO &800F
1260        IF (i/2)=INT(i/2) THEN POKE i,0 ELSE POKE i,&FF
1270 NEXT i
```

Pour montrer que les sprites se déplacent sur l'écran sans effacer le décor affiché, une boucle **FOR NEXT** remplit l'écran **MODE 1** de tirets :

```
1320 MODE 1
1330 FOR i=1 TO 1000
1340        PRINT "-";
1350        NEXT i
```

Avant d'afficher le sprite pour la première fois, la mémoire d'adresse **&H8200** est initialisée à **0** :

```
1360 POKE &8200,0: SPRITE,1,80,100
```

La mémoire d'adresse **&H8200** est alors initialisée à **1** pour tout le reste du programme :

```
1370 POKE &H8200,1
```

Les dernières instructions du programme réalisent l'animation du sprite qui semble rebondir sur les coins de l'écran.

La position de départ du sprite est 80,100 :

```
1380 x=80
1390 y=100
```

Les déplacements selon les deux axes sont fixés à 2 pixels :

```
1400 dx=2          → déplacement en x
1410 dy=2          → déplacement en y
```

La fonction **INKEY$** scrute le clavier. Lorsqu'une touche est pressée, le programme prend fin :

```
1420 a$=INKEY$
...
1480 IF a$=" " THEN 1420
1490 MODE 2
1500 END
```

La logique de déplacement du sprite est ultra simple :

– si le sprite se trouve sur l'extrémité droite ou gauche de l'écran, le déplacement en **X** est inversé :

```
1430 IF (x=310) OR (x=0) THEN dx=-dx
```

– si le sprite se trouve sur l'extrémité haute ou basse de l'écran, le déplacement en **Y** est inversé :

```
1440 IF (y=200) OR (y=0) THEN dy=-dy
```

– les coordonnées du sprite sont incrémentées de **dx** et **dy** :

```
1450 x=x+dx
1460 y=y+dy
```

Le sprite est enfin affiché à l'aide d'une instruction ¦**SPRITE** :

```
1470 ¦SPRITE,1,x,y
```

Les dernières lignes du programme (1520 à 1770) contiennent les données hexadécimales de la RSX ¦SPRITE.

Afin d'éviter toute erreur dans la saisie des DATA, voici les données de checksum correspondantes :

```
0 0 86 94 C4 25 A4 7E 57 EE DE FB A1 ED 71 F9 AA B7 24 14 93 7B
```