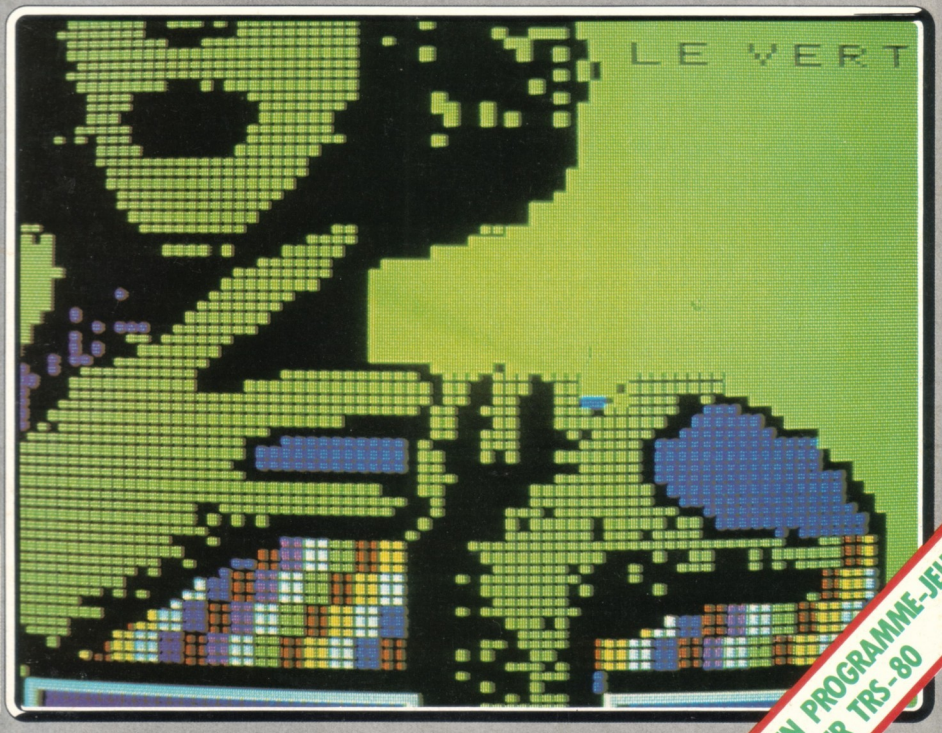
BASIC PLUS

ENCYCLOPEDIE D'INITIATION A L'INFORMATIQUE ET AUX ORDINATEURS

70



FICHE: UN PROGRAMME-JEU POUR TRS-80

M 6268-70-13FF 90 FB - 4,40 FS

EDITIONS DU HENNIN

BASIC PLUS

ENCYCLOPEDIE D'INITIATION A L'INFORMATIQUE ET AUX ORDINATEURS

Parution hebdomadaire publiée par
les Editions du Hennin
20-22, rue de Clichy 75009 Paris.
Directeur général : Charles-Jean Pradelle.
Directeur délégué : Italo Milani.
Avec la collaboration de **Bull** et de
Thomson Micro-Informatique Grand Public.

Direction éditoriale : Nathalie Perrin.
Secrétariat général de la rédaction: Servane Wattel.
Adaptation et révision technique
dirigées par Mustafa Zaoui.

© Editions du Hennin S.A.R.L. 1984.
© Armando Curcio - Editore - Rome.
Titre original : Basic, Enciclopedia dell'Informatica dei Mini e Personal Computer.
Réalisé par le Département Grands Ouvrages d'Armando Curcio Editore, sous la direction technique de Sante Senni et la direction éditoriale de Gabriella Costarelli.

Dépôt légal : 2e trimestre 1985, ISSN : en cours.
Composition : Unipag, Paris. Imprimé en Belgique par l'ASAR, département offset, Bruxelles.
Photo de couverture : Essai graphique sur une norme Télétel.
Réalisation Etienne Bonnet.

Directeur de la publication : Maurice Brébart.

Administration
France : Éditions du Hennin, 20-22, rue de Clichy 75009 Paris. Tél. : (1) 280.64.65.
Belgique : Femmes d'Aujourd'hui S.A., avenue Frans Van Kalken, 9 B-1070 Bruxelles, Tél. : (02) 523.20.60.

Distribution
France : NMPP. Belgique : AMP S.A. Suisse : NAVILLE S.A.

Vente au numéro
Les fascicules peuvent être obtenus chez les marchands de journaux ou, à défaut, chez l'Éditeur, au prix en vigueur au moment de la commande. Pour toute commande par lettre, joindre le règlement (chèque bancaire ou postal) majoré de 10 % représentant la participation aux frais de port.

Vente par souscription
France : 72 fascicules + 6 reliures mobiles pour un prix global de 1.300,00 FF dont 107,00 FF de frais de participation au conditionnement et de port (pour la France métropolitaine).
Payable par prélèvement automatique en 20 mensualités de 65,00 FF.
Adressez votre courrier à :
AZ Distribution , 20-22, rue de Clichy, 75009 Paris.
N.B. : Avec votre commande, n'envoyez pas d'argent (ni chèque, ni mandat). Toute souscription concerne l'intégralité de l'ouvrage et commence à partir du n° 1.

Belgique : adressez-vous à Basic Plus, avenue Frans Van Kalken, 9. B-1070 Bruxelles.

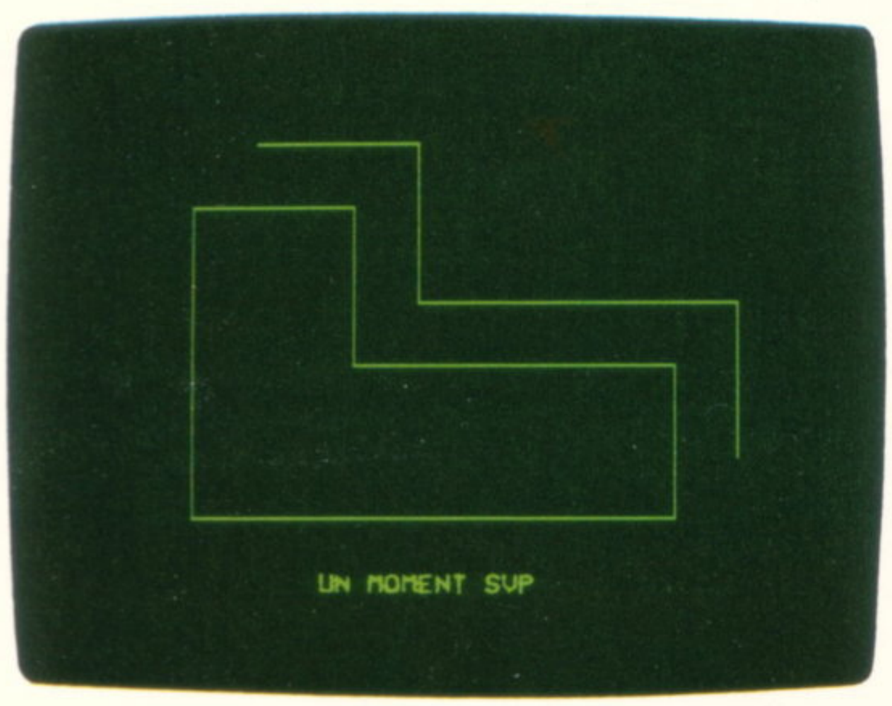
BASIC PLUS, première encyclopédie d'initiation à l'informatique et aux ordinateurs, se compose de 72 fascicules de 24 pages (+ 4 pages de couverture), destinés à être assemblés en 6 volumes de 288 pages à l'aide de reliures mobiles spécialement conçues à cet effet par l'Editeur et qui sont à la disposition de nos lecteurs chez les marchands de journaux.

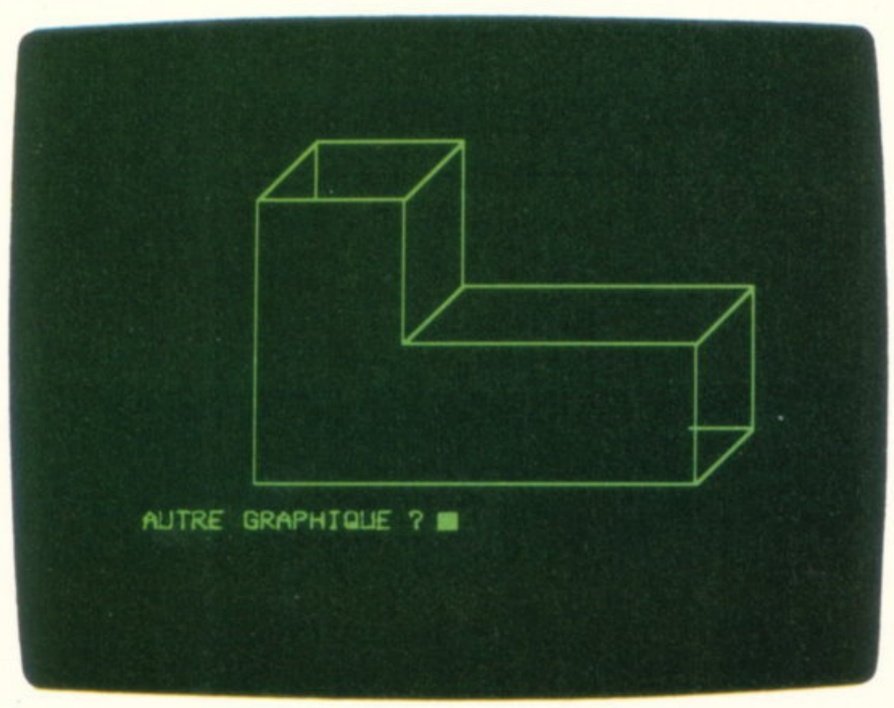
AVIS AUX LECTEURS

BASIC PLUS paraît le mercredi. En achetant vos fascicules chaque semaine chez le même marchand de journaux, vous serez mieux servis et vous nous permettrez ainsi d'assurer avec plus de précision la distribution.

```
5525 IF X=1 THEN XØ=XA
     :YØ=INT((YA+YB)/2):GOTO 554Ø
553Ø XØ=INT((XA+XB)/2):YØ=INT(M*XØ+N)
554Ø XØ=XQ:YØ=YQ
5545 S(1,1)=Ø:S(2,1)=Ø:T(1,1)=Ø
     :T(2,1)=Ø
555Ø CF=FL(1):KE=1:FL(1)=Ø:GOSUB 3Ø1Ø
558Ø IF FL(J)=CN THEN XA=XQ:YA=YQ
     :GOTO 56ØØ
559Ø XB=XQ:YB=YØ
56ØØ FL(1)=CF
5605 IF L<=L1 THEN SW=Ø:RETURN

561Ø GOTO 552Ø

5999 :
6ØØØ REM ============
6ØØ1 REM * LIGNES DE *
6ØØ2 REM * FONCTION  *
6ØØ3 REM ============

6ØØ4 :
6Ø1Ø FOR I=1 TO N1
6Ø2Ø IF FL(I)=CN THEN 6Ø4Ø
6Ø3Ø HPLOT XP(I),YP(I) TO XT(I),YT(I)
6Ø4Ø NEXT I
6Ø5Ø RETURN

1ØØØØ :
1ØØØ1 REM ==========
1ØØØ2 REM * ERREUR *
1ØØØ3 REM ==========

1ØØØ4 :
1ØØØ5 TEXT:HOME
1ØØ1Ø ER=PEEK(222)
1ØØ2Ø IF ER=53 THEN VTAB 1Ø:PRINT
       "Coordonnées fausses "
     :GOTO 1ØØ5Ø
1ØØ3Ø VTAB 1Ø:PRINT "Données fausses "
1ØØ5Ø VTAB 22:HTAB 3Ø:PRINT ">>";
1ØØ6Ø GET Q$:RUN
```

Exemple de fonctionnement du programme présenté.
A gauche, le programme a analysé les points de la figure de base et a présenté, après translation, ceux qui sont visibles. A droite, le dessin a été complété. Remarquer les deux segments erronés.





L'animation d'images par ordinateur

L'animation d'images graphiques constitue une activité bien particulière dans l'informatique graphique. Les techniques d'animation sont déjà largement utilisées dans les programmes de jeux vidéo. Il existe de très nombreux exemples dans lesquels l'animation d'une figure à l'écran peut simplifier considérablement la résolution d'un problème. C'est le cas quand on désire connaître l'effet d'une force sur une structure solide, ou observer le mouvement d'un organe mécanique.

Les objets animés

Nous avons vu que la visualisation d'une image graphique n'est pas une opération élémentaire. Elle passe par le tracé d'un grand nombre de segments qui serviront à la reconstitution de l'image. Le déplacement de l'objet correspondra à une succession d'effacements et de nouveaux tracés dans une autre position

Un **objet animé** (on dit « **sprite** », lutin, en anglais), est, à l'inverse, une image graphique qui est gérée comme telle par le programme, en choisissant une fois pour toutes sa forme et ses différentes positions à l'écran.

L'objet animé, qui est limité en fonction des dimensions de l'écran vidéo, possède trois caractéristiques principales :

— il est complètement programmable par l'utilisateur ;
— il se déplace dans les quatre directions du plan ;
— il mémorise d'éventuelles collisions avec d'autres objets animés ou d'autres figures présentes à l'écran.

Chacune de ces caractéristiques peut être mise en œuvre à l'aide de sous-programmes spéciaux, assurant la simulation d'objets animés même dans les systèmes ne les prévoyant pas. Les autres machines possèdent, quant à elles, des instructions spéciales de déplacement, de contrôle de collisions, etc. Elles sont donc dotées de certaines implémentations qui ne se trouvent pas dans le Basic Standard.

La gestion des objets animés est généralement commandée par des composants matériels ad hoc. Pour les machines plus banales il faut écrire des programmes qui vont simuler ces composants. Les fonctions exécutables sont toutefois plus limitées et la vitesse moins élevée, tout au moins en Basic interprété.

Pilotage des objets animés par logiciel

L'emploi des objets animés est étroitement lié à la production de dessins animés. Certaines des difficultés, rencontrées dans le pilotage par programme, sont justement dues à la nature de cette application.

Un objet animé peut être créé comme une table de figures et déplacé par variation de l'origine. Cette technique, bien connue, consiste à reproduire la figure à déplacer (l'objet animé) comme une série de déplacements (paramétrables) par rapport à une origine.

Si les coordonnées du point choisi comme référence sont modifiées, la figure se trouve déplacée (il faut naturellement prévoir son effacement).

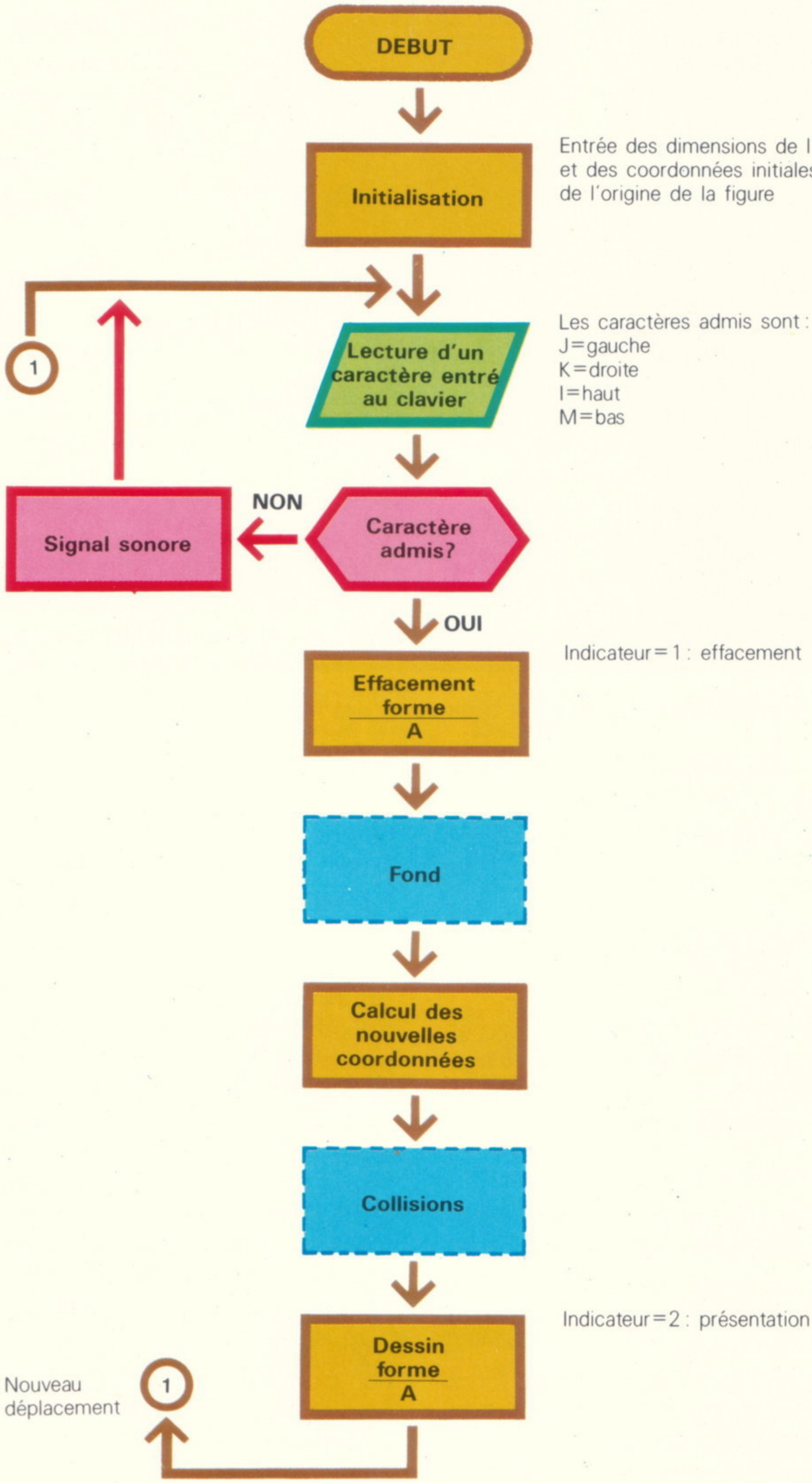
Page ci-contre, se trouve l'organigramme d'un programme qui déplace une figure à l'écran en suivant les commandes entrées au clavier. Les différentes phases de ce programme apparaissent immédiatement. Le sous-programme marqué A mérite, toutefois, une observation : il exécute aussi bien l'effacement que la visualisation. Il est déclenché par l'introduction d'un indicateur spécifiant laquelle de ces deux fonctions est désirée.

L'organigramme comporte également deux blocs (Fond et Collisions) dont l'emploi est spécifique à cette application.

L'animation d'images graphiques. L'animation par ordinateur est une technique très proche de celle employée au cinéma. La scène contient des images fixes (le fond) par rapport auxquelles se déplace l'objet de l'animation.

Au cinéma, la technique consiste à dessiner uniquement le fond, plus autant de tables, sur support transparent, que de mouvements désirés. Sur chacune de ces tables, la figure à déplacer est reproduite dans une position légèrement modifiée par rapport à la précédente. En posant ces différentes tables sur le fond et en photographiant toutes les scènes intermédiaires obtenues, on obtient un fond, qui, à la projection, donnera une réelle impression de mouvement.

DEPLACEMENT D'UNE FORME



P.A. Simon/Grazia Neri-Photolake

Un moment de la conception assistée par ordinateur des image du film « Tron ».

En réalité le problème n'est pas aussi simple qu'on pourrait le croire.
Noter que la figure qui se déplace est gérée de deux façons :

1 / Lors du déplacement par rapport au fond, la figure reste inchangée ; il faut donc faire varier uniquement sa position.
2 / La figure se déplace par rapport au fond et subit des modifications. Dans ce cas, il faut ajouter au déplacement un nouveau tracé de la figure, partiel ou total, à chaque position.

Le deuxième cas, très facile à résoudre avec une feuille de papier et un crayon, pose quelques difficultés au niveau de l'ordinateur. La première grande difficulté, c'est de rétablir le fond après un mouvement.
Un point donné de l'écran appartient au fond, du moins tant que sa zone n'est pas touchée par la figure déplacée. De fait, il est actif (visible) ou éteint en fonction de ce que l'on doit montrer en fond.

Quand le même point est caché par la figure déplacée, il subit les impératifs de tracé de celle-ci et doit être rétabli à sa position initiale après un déplacement le dégageant.
Par exemple, si un petit bonhomme qui se promène passe devant un arbre, l'ordinateur devra, de pas en pas, remplacer une partie de l'arbre par la silhouette du bonhomme, puis rétablir l'image d'origine de l'arbre dès qu'il sera passé. Si, de plus, la figure est susceptible de subir des modifications, il ne s'agit alors plus seulement de déplacer un graphique, mais aussi de prévoir ses variations.
Dans les machines assurant l'animation, le fond est traité par le système : la figure mobile constitue l'objet animé et peut être déplacée par rapport au fond.
La sensation obtenue est celle de mouvement d'une figure statique. Pour obtenir une animation plus réaliste, il faut alors également modifier la forme de la figure, par exemple en dessinant quelques détails en différents points.
Avec la technique des objets animés, deux possibilités sont offertes : modifier la forme de

l'objet animé avant déplacement ou créer autant d'objets animés qu'il y a de positions à simuler et les présenter les uns à la suite des autres.
Dans ce cas, chacune des figures représente un objet, et l'animation consiste à les activer successivement. Ce système est alors tout à fait analogue à l'animation traditionnelle.
Les mêmes problèmes se posent dans la gestion des objets animés avec des programmes écrits par l'utilisateur. Il est nécessaire d'y prévoir un sous-programme de reconstruction du fond et un autre pour le contrôle des collisions. Avec les machines dédiées à ce type d'applications, cette dernière fonction est automatiquement prise en charge.

Gestion du fond. Il existe différentes méthodes de gestion du fond d'un dessin animé. Leur degré de complexité dépend principalement de deux paramètres :

- La vitesse d'exécution
- L'occupation de la mémoire

Pour donner une impression de déplacement continu, l'animation d'un dessin demande une vitesse d'exécution élevée.
Si le programme n'est pas suffisamment rapide, l'œil réussit à percevoir les différentes phases du déplacement (effacement, reconstruction du fond, nouvelle présentation), ce qui n'est pas agréable ; de plus, il faut effectuer des déplacements relativement importants pour ne pas obtenir un mouvement trop lent, ce qui serait inacceptable dans les jeux vidéo par exemple.
La méthode et les exemples exposés ci-après ne servent qu'à illustrer ces aspects de la gestion du fond. Pour rester suffisamment généraux, ils n'exploitent pas les particularités matérielles de certaines machines, qui pourraient permettre de réduire les problèmes de lenteur d'exécution et d'espace mémoire. Il est en tout cas pratique d'utiliser la version interprétée des programmes dans la phase d'écriture et de contrôle, puis de recourir à la version compilée pour l'application, afin d'obtenir des vitesses d'exécution acceptables.
Une autre solution consisterait à écrire certains sous-programmes en Assembleur.

Le fond peut être considéré comme un dessin faisant appel à l'écran tout entier. La gestion la plus simple, lors d'une animation, serait d'enregistrer le dessin dans une zone mémoire spéciale puis de le rappeler au moment de sa reconstruction.
Voici les fonctions impliquées dans le déplacement d'un objet animé :

1 / Transfert du fond de la mémoire de soutien à l'écran (visualisation)
2 / Présentation de l'objet dans sa position initiale
3 / Présentation du fond (entraînant l'effacement de l'objet)
4 / Présentation de l'objet dans la position suivante

Le mouvement de l'objet est obtenu en activant, de façon itérative et rapide, ces quatre phases, jusqu'à reproduire le déplacement tout entier au moyen d'une série de petits mouvements donnant une impression de continuité.
L'effet visuel de cette méthode n'est toutefois pas des meilleurs. En effet, la faible vitesse d'exécution, liée à la quasi-totalité des interpréteurs Basic, rend perceptible l'enchaînement des opérations.
Il n'y a pas de solution logicielle généralisable, car une bonne animation nécessite obligatoirement l'emploi de certains éléments matériels particuliers.

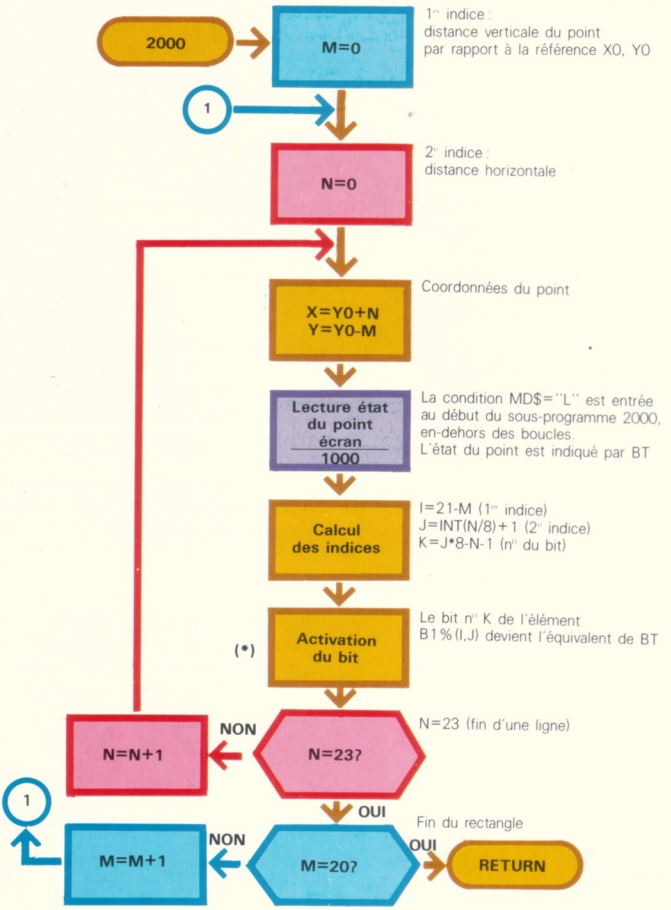
Visualisation d'un objet animé. Voyons maintenant de près comment on peut résoudre le problème d'une machine. Nous voulons visualiser à l'écran un petit rectangle de 24 (base) sur 21 (hauteur) points écran *, position au point X0, Y0 sur une image graphique constituant le fond.
Chacun des déplacements de l'objet (rectangle) devra être accompagné de la reconstruction de la partie du fond précédemment masquée.
Ce processus comporte les trois phases successives suivantes :

---

*Ces valeurs ont été choisies pour conserver les dimensions de l'objet, utilisées par les systèmes prévoyant la gestion matérielle des objets animés.

Phase 1. Avant le positionnement de l'objet, la zone de l'écran concernée (rectangle ayant les mêmes dimensions que l'objet) est sauvegardée dans une zone mémoire (tableau bidimensionnel). Cette fonction consiste à calculer, à partir des coordonnées à l'écran de chaque point à sauvegarder, l'adresse correspondante dans la mémoire vidéo.

MEMORISATION DE LA FENETRE CONTENANT L'OBJET



*Ce bloc d'instructions est identique à celui utilisé dans le sous-programme 1000 pour activer un point écran. Cette méthode concerne les machines n'admettant pas d'employer les opérateurs logiques dans la création de masquages à l'intérieur de l'octet. Quand cette possibilité existe (par exemple sous CP/M), il suffit d'utiliser l'opérateur logique approprié.

Le contenu de cette mémoire est lu et transféré dans une mémoire tampon. Dans l'organigramme ci-contre, le tampon utilisé est B1%(21,3) (ce dimensionnement sera expliqué plus loin).

Phase 2. L'objet est introduit à partir du point X0, Y0. Le contenu de la mémoire vidéo correspondante est remplacé par le contenu du tableau S%(21,3) qui représente, sous une forme spéciale, l'objet.

Phase 3. Le déplacement de l'objet est obtenu en rétablissant le fond, c'est-à-dire en transférant B1%(21,3) dans la zone mémoire vidéo occupée par l'objet et en reprenant le cycle après avoir modifié les valeurs de X0,Y0.
Les phases 2 et 3 peuvent être décrites dans le même organigramme. Le sous-programme correspondant, qui est paramétré, présente, à partir des coordonnées X0,Y0, l'objet S%(21,3) ou le fond B1%(21,3).

Le transfert de données, à partir et à destination de la mémoire vidéo, nécessite le calcul de la position de mémoire correspondant aux coordonnées en points écran et du bit qu'elle contient. Sur ce point, nous avons repris le sous-programme 100 déjà présenté en l'adaptant.
Ce sous-programme demande, en entrée, les coordonnées X et Y du point écran et la valeur affectée à MD$ (MD$="L" pour la lecture de l'état allumé/éteint, MD$="S" pour l'écriture). Il fournit, en sortie, BT=1 si le point est allumé et BT=0 s'il est éteint.
Il est illustré, page suivante, par le tableau B1%(21,3). Chacun des éléments se composant de 8 bits, il peut contenir l'état d'autant de points écran. Les 24 points horizontaux sont donc mémorisés dans 3 cellules mémoire seulement. La même logique est utilisée dans certaines machines prévoyant la gestion des objets animés.
L'organigramme de reconstruction du fond ou de présentation de l'objet en fonction de l'indicateur F (F=1 fond, F=2 objet) est en page 1665. Les fonctions exécutées sont tout à fait analogues à celles du sous-programme 2000 avec, pour seule différence, l'utilisation de deux tableaux différents selon la valeur de F (F=1 pour B1%, F=2 pour S%).

Page 1666 enfin, c'est l'organigramme d'un programme dans lequel les sous-programmes 5000 et 6000 servent uniquement à préparer les données (voir listing en pages 1667 à 1669).

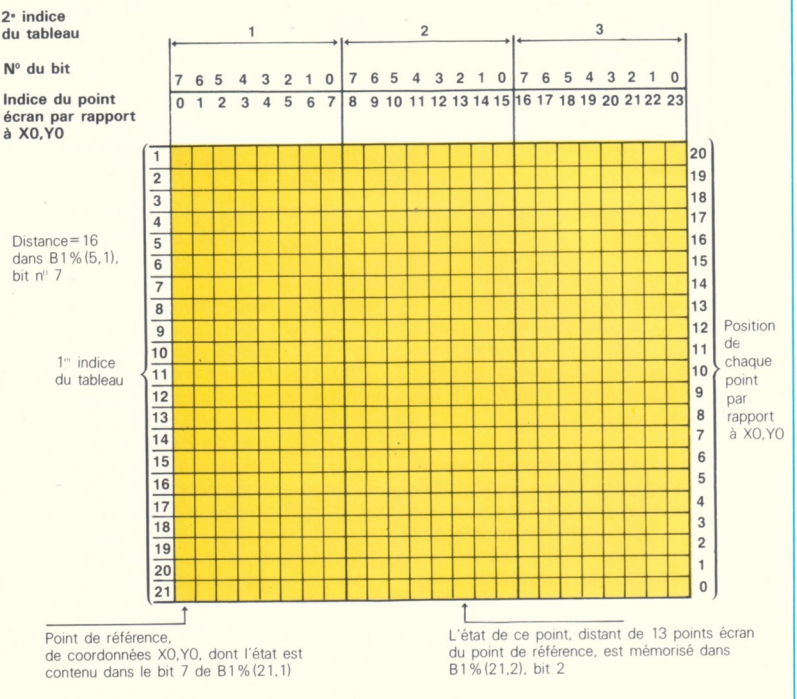
Problèmes d'animation. Dans le programme de la page 1666, l'objet est assimilé à un rectangle qui, converti en image binaire, est mémorisé dans le tableau S%. Cette méthode présente deux inconvénients :

1 / Elle ne permet pas de définir des contours.
2 / Elle ne prévoit pas d'animations à l'intérieur de l'objet.

Le premier oblige à remplir toute la zone disponible pour l'objet, faute de quoi il se créerait une zone toujours éteinte. Celle-ci correspondrait à la différence entre la zone du rectangle effectivement occupée par le dessin (objet) et la totalité de la zone à la disposition de l'objet. Ce défaut peut être éliminé à l'aide d'un sous-programme de présentation plus évolué. Il ne s'agit plus simplement de présenter le rectangle formant l'objet, mais de construire l'intersection entre ses points et ceux de l'écran. Autrement dit, si un point de l'objet est actif, il doit être visualisé à l'écran (il couvre le fond) ; s'il est inactif (éteint), la mémoire vidéo doit rester intacte, comme couverte par le fond. Cela équivaut à dessiner l'objet sur un rectangle transparent.
Cette fonction peut être obtenue très simplement à l'aide d'un IF lors du transfert en mémoire vidéo de l'image de l'objet. Si le point concerné n'est pas actif, il n'y aura pas de transfert, de manière à garder le fond inchangé (voir schéma en page 1671).
Le deuxième défaut (pas d'animation dans l'objet) est nettement plus difficile à traiter. En fait, même avec des machines dédiées, l'animation reste une fonction difficile. Il s'agit, en effet, de modifier la forme de l'objet au fur et à mesure que le rectangle qui le contient se déplace.
On crée ce mouvement de deux façons :

1 / En mémorisant certaines positions intermédiaires.
2 / En schématisant les déplacements à l'aide de règles mathématiques.

ADRESSAGE DE LA MEMOIRE DE L'OBJET



Le balayage commençant par la dernière rangée (élément de 1er indice=21), les points écran de la rangée 21 peuvent être divisés en trois groupes:

— le premier, (2e indice=1) est contenu dans les bits 0 à 7 de l'élément B1%(21,1)
— le deuxième est contenu dans les bits 0 à 7 de B1%(21,2)
— le troisième est contenu dans B1%(21,3)

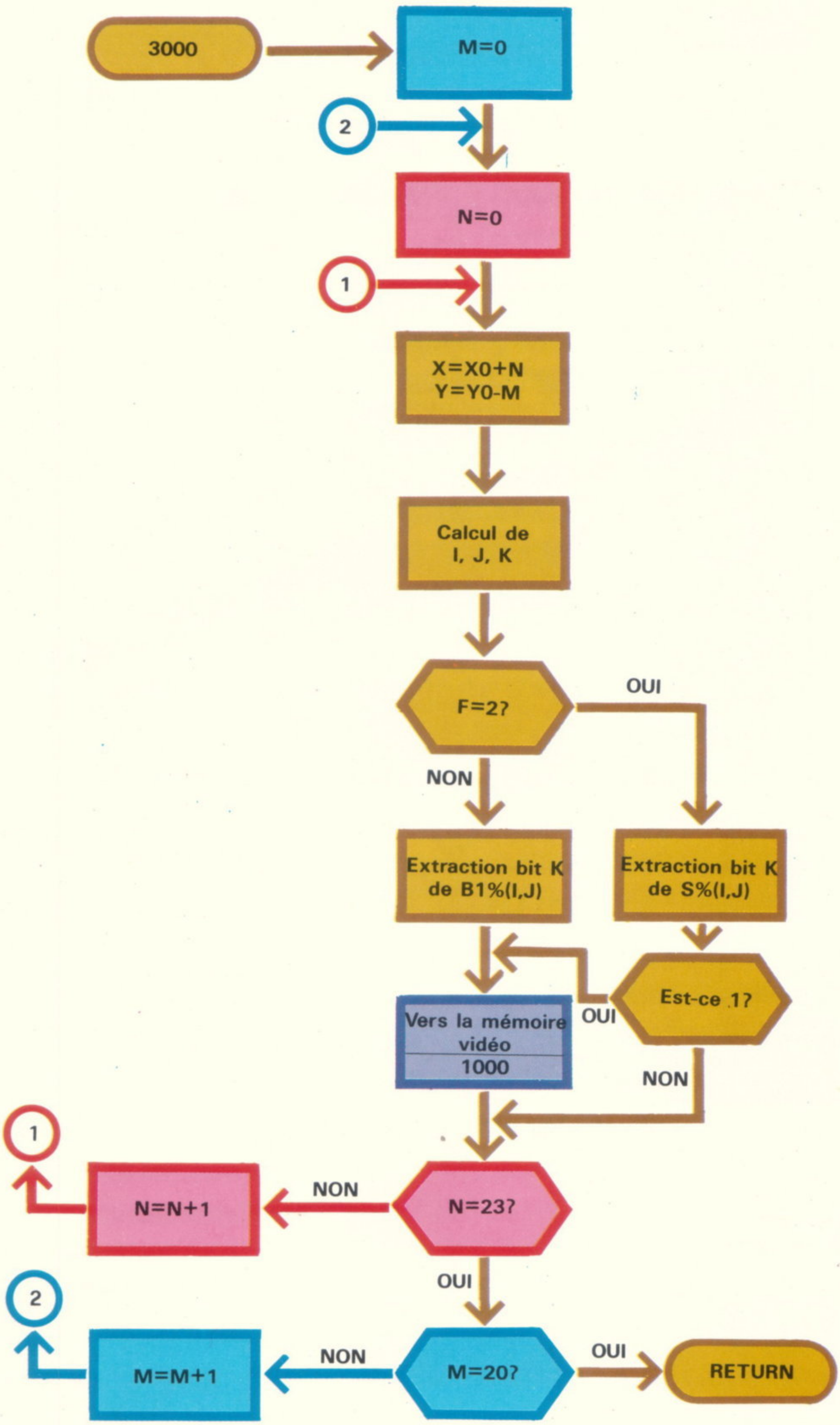
La même procédure permet d'obtenir l'adresse (indices et bits) de tout autre point. Par exemple, le point N=12, M=13 est contenu dans l'élément B1%(8,2) bit 3, selon la règle:

1er indice=21-M
2e indice =INT(N/8)+1
3e du bit=Second indice*8-N-1

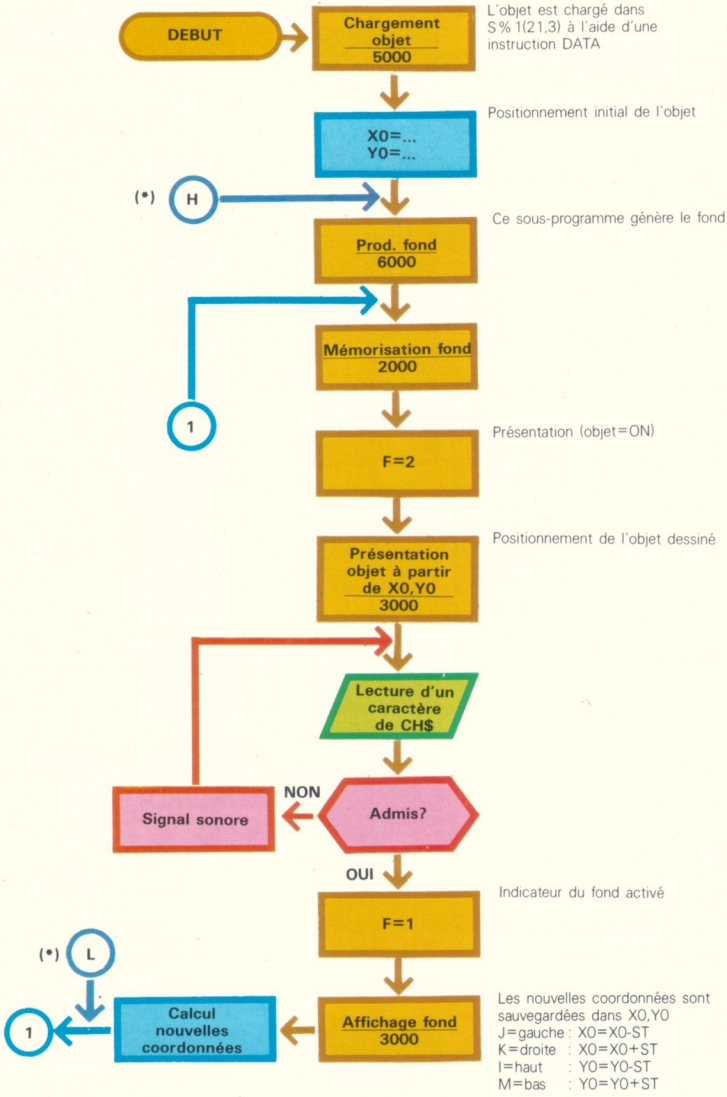
Noter que le point de référence (point X0,Y0) est distant de lui-même des quantités N=0, M=0. Par conséquent:

1er indice=21-0=21
2e indice =1
n° du bit =7

PRESENTATION DU FOND OU DE L'OBJET



EXEMPLE DE GESTION DES OBJETS ANIMES



(*) : Il sera fait référence aux connecteurs H et L plus loin.

EXEMPLE DE GESTION DES OBJETS ANIMES PAR LE LOGICIEL

```
10 REM ----------------------
20 REM * PROGRAMME PRINCIPAL *
30 REM ----------------------
40 DIM S%(21,3),BI%(21,3)
50 ST=5
60 GOSUB 5000
   :REM Chargement de l'objet animé
70 X0=10:Y0=180
80 GOSUB 6000:REM Dessin du fond
90 GOSUB 2000
   :REM Lecture de la fenêtre

100 F=2
110 GOSUB 3000
    :REM Ecriture de l'objet animé
120 GET CH$
130 IF CH$<>"I" AND CH$<>"J" AND
     CH$<>"M" AND CH$<>"K"
     AND CH$<>"CHR$(27) THEN 120
140 F=1
150 GOSUB 3000
    :REM Ecriture de la fenêtre
160 IF CH$="I" THEN Y0=Y0-ST
170 IF CH$="M" THEN Y0=Y0+ST
180 IF CH$="J" THEN X0=X0-ST
190 IF CH$="K" THEN X0=X0+ST
200 IF CH$=CHR$(27) THEN END

210 GOTO 90

1000 REM ----------------------
1010 REM * CALCUL DES ADRESSES *
1020 REM ----------------------
1030 GY=INT(Y/8)
1040 IF Y<64 THEN MR=8232+(GY-8)*128
     :GOTO 1070
1050 IF Y<128 THEN MR=8232+(GY-8)*128
     :GOTO 1070
1060 MR=8272+(GY-16)*128
1070 A=Y-GY*8
1080 MY=MR+A*1024
1090 GX=INT(X/7)

1100 MX=MY+GX
1110 BIT=X-7*GX
1120 NN=PEEK(MX)
1130 BY=NN
1140 FOR KK=0 TO BIT
1150 MS=NN/2
1160 NN=INT(MS)
1170 NEXT
1180 IF MS<>NN THEN BT=1:GOTO 1200
     :REM Le point écran est allumé
1190 BT=0
     :REM Le point écran est éteint

1200 RETURN

2000 REM ------------------------
2010 REM * LECTURE DE LA FENETRE *
2020 REM ------------------------
2030 MD$="L"
2040 FOR M=0 TO 20
2050 FOR N=0 TO 23
2060 X=X0+N:Y=Y0-M
```

```
2070 GOSUB 1000
2080 I=21-M:REM Premier indice
2090 J=INT(N/8)+1:REM Deuxième indice

2100 K=J*8-N-1
     :REM Numéro du point écran
2110 B1%(I,J)=B1%(I,J)+BT*2^K
2120 NEXT N
2130 NEXT M
2140 RETURN

3000 REM ---------------------------
3010 REM * ECRITURE DE LA FENETRE *
3020 REM ---------------------------
3030 MD$="S"
3040 FOR M=0 TO 20
3050 FOR N=0 TO 23
3060 X=X0+N:Y=Y0-M
3070 I=21-M:REM cf. 2080
3080 J=INT(N/8)+1:REM cf. 2090
3090 K=J*8-N-1:REM cf. 2100

3100 IF F=2 THEN N0=S%(I,J):GOTO 3120
3110 N0=B1%(I,J)
3120 FOR N1=0 TO K
3130 N2=N0/2
3140 N0=INT(N2)
3150 NEXT N1
3160 IF N2=N0 THEN BB=0:GOTO 3180
3170 BB=1
3180 GOSUB 1000
3190 IF BT=BB THEN 3220

3200 IF BT>BB THEN POKE MX,BY-2^BIT
     :GOTO 3220
3210 POKE MX,BY+2^BIT
3220 NEXT N
3230 NEXT M
3240 RETURN

5000 REM ---------------------------
5010 REM * CHARGEMENT OBJET ANIME *
5020 REM ---------------------------
5030 FOR I=1 TO 21
5040 FOR J=1 TO 3
5050 READ S%(I,J)
5060 NEXT J,I
5070 DATA 0,0,0
5080 DATA 0,0,0
5090 DATA 0,0,0

5100 DATA 0,0,0
5110 DATA 0,0,0
5120 DATA 0,0,0
5130 DATA 0,0,0
5140 DATA 28,0,248
5150 DATA 62,1,252
5160 DATA 103,3,254
5170 DATA 199,7,158
5180 DATA 143,15,28
5190 DATA 30,15,28

5200 DATA 60,30,60
5210 DATA 60,30,108
5220 DATA 31,252,248
5230 DATA 15,248,144
```

```
5240 DATA 7,240,48
5250 DATA 0,0,0
5260 DATA 0,0,0
5270 DATA 0,0,0
5280 RETURN

6000 REM ----------------------
6010 REM * CHARGEMENT DU FOND *
6020 REM ----------------------
6030 HOME
6040 INPUT "Nom du fond ";NS$

6050 HGR:POKE -16302,0:HCOLOR=3
6060 PRINT CHR$(4);"BLOAD";NS$
     ;"A$2000"
6070 RETURN
```

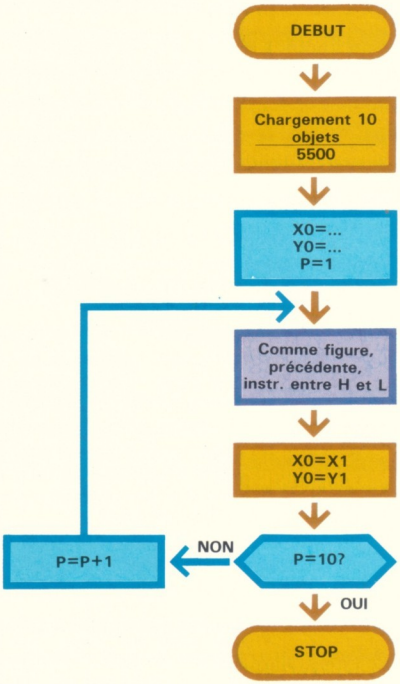
La seconde solution est la plus pratique, tant en raison de sa vitesse d'exécution que de la moindre occupation de l'espace mémoire. Elle n'est toutefois pas toujours applicable, les déplacements d'une figure étant difficiles à schématiser avec des formules mathématiques.
Quant à la première solution, elle ne nécessite qu'un compteur et une zone de mémoire contenant toutes les positions intermédiaires que l'on veut utiliser. Par exemple, si l'on range 100 positions intermédiaires dans T% (100,21,3), il suffit de les transférer (les unes après les autres, avec le compteur) dans S% (21,3) pour obtenir un mouvement également à l'intérieur de l'objet.
Une autre possibilité, enfin, consiste à ne dessiner que les positions initiales et finale, et éventuellement quelques intermédiaires si les deux précédentes sont très différentes. On laissera à l'ordinateur le soin de calculer tous les passages intermédiaires nécessaires à l'animation. Cette technique, qui présente des aspects extrêmement complexes, est encore à l'étude mais elle sera de plus en plus employée pour les productions cinématographiques.
On trouvera, à la page suivante, l'organigramme donné en page 1666, modifié de façon à permettre également l'animation à l'intérieur de l'objet. Cette animation s'obtient en préparant une dizaine de formes qui seront successivement présentées. Le déplacement de l'objet étant commandé par touches, l'animation est ralentie par la phase d'entrée. Pour obtenir un résultat plus réaliste, on définit préalablement un parcours qui sera exécuté automatiquement. La méthode la plus simple, pour décrire ce parcours consiste à définir un tableau (à 10 valeurs) dans lequel seront introduits (à l'aide d'une instruction DATA) les codes correspondant aux touches de déplacement. L'instruction de lecture sera alors activée par l'instruction CH$ = SP (P), SP étant un tableau contenant les codes de placement.
Une boucle dans laquelle P va de 1 à 10 simulera les pressions consécutives de 10 touches de déplacement et génèrera automatiquement le parcours défini dans l'instruction DATA.
La gestion des collisions est un élément indispensable pour ce type d'applications. Elle consiste à chercher si, lors de son déplacement, un objet animé en rencontre un autre ou une zone particulière du fond.
Ce contrôle s'effectue par lecture de la zone mémoire occupée par l'objet : il y a collision si elle est active.

Gestion matérielle des objets animés

Beaucoup d'ordinateurs personnels, certes plus proches des jeux vidéo que des applications professionnelles, sont équipés de composants matériels dédiés à la gestion des objets animés.
Ces composants peuvent mémoriser, déplacer et contrôler simultanément plusieurs objets. Il ne reste, à l'utilisateur, qu'à produire ces objets et le fond, puis à donner les instructions de mouvement, sans avoir à s'occuper ni d'effacement ni de reconstruction.
Dans ce type de machine, chaque objet animé se compose d'un ensemble de points divisés en rangées et colonnes ; on peut constituer la forme désirée en allumant certains d'entre eux.

Mémorisation de l'objet animé. Pour chaque objet il est prévu plusieurs centaines de points écran. Par exemple, la valeur utilisée par le Commodore 64 est de 504 points divisés en 21 rangées de 24 colonnes. L'occupation

GESTION DES OBJETS AVEC ANIMATION INTERNE



Les objets préparés (10) sont des formes différentes de la même figure. Présentées les unes après les autres, elles permettent d'animer l'intérieur du rectangle. Les 10 formes sont mémorisées dans T(10,21,3) à l'aide d'une instruction DATA

P est le compteur réglant la succession des 10 formes de l'objet

La seule différence se situe au niveau du sous-programme 3000, qui doit concerner le fond, alors qu'un autre (3700) gère l'objet.

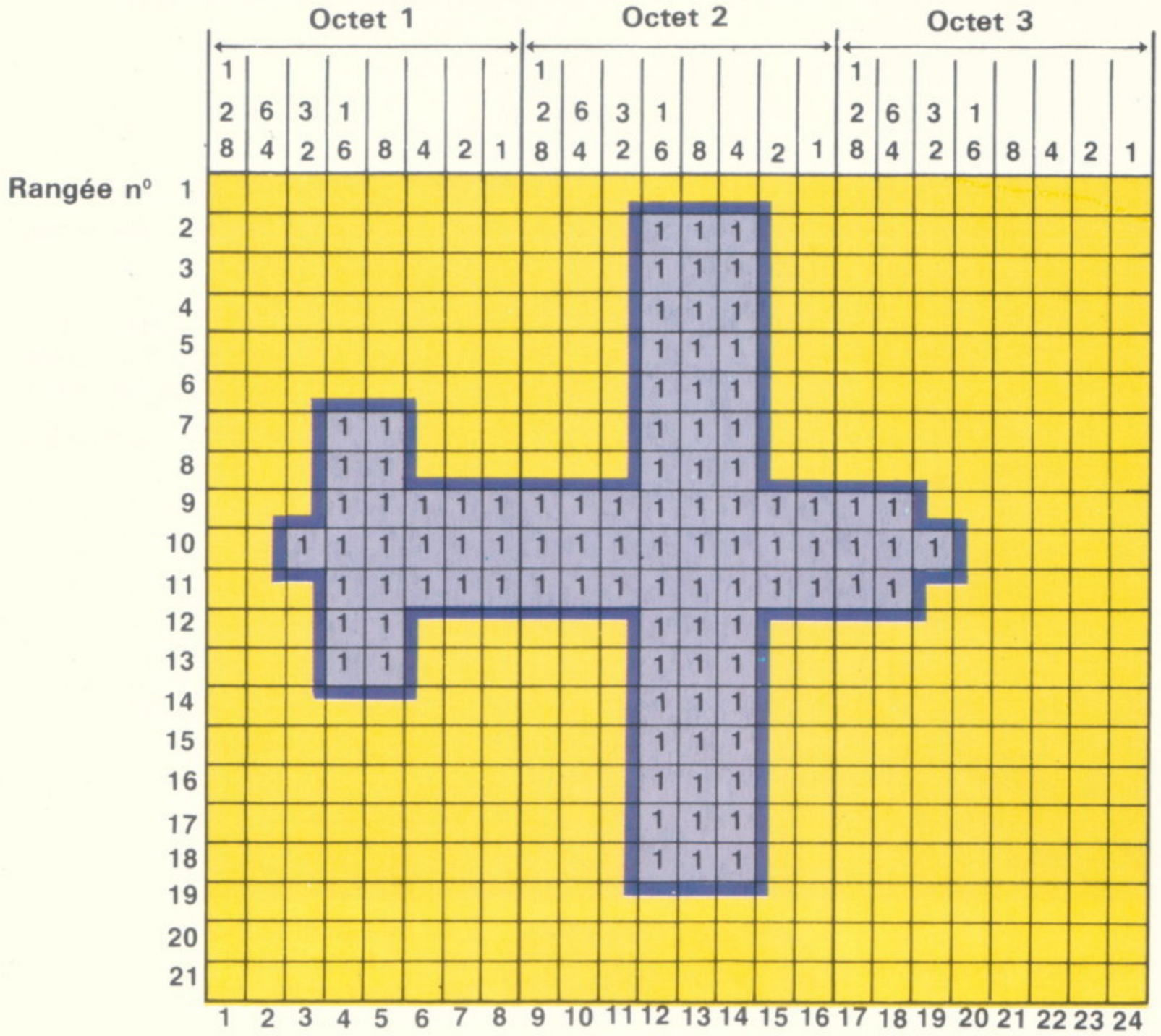
Le sous-programme 3700 est identique au 3000, à ceci près :
— une boucle initiale transfère T%(P,21,3) dans S%(21,3)
— le test de F et le bloc d'extraction du bit K de B1%(I,J) sont éliminés ; le sous-programme n'est plus paramétré et n'a donc pas à gérer le fond

en mémoire est donc de 504 bits soit 63 octets. La construction d'un objet consiste à mémoriser, dans 63 cellules de mémoire contiguës, l'état allumé/éteint de chacun des 504 bits. Pour allumer un point, il faut écrire, dans l'octet correspondant, une valeur décimale particulière qui active le bit sélectionné en le mettant à 1. Par exemple, pour activer le point se trouvant au croisement entre la première rangée et la première colonne, il faut écrire la valeur 128, tandis que pour la colonne 4, cette valeur sera 16 (ne pas oublier que les valeurs décimales correspondant à la fonction du bit sont 1, 2, 4, 8, 16, 32, 64, 128). Les valeurs pouvant être additionnées en écrivant le chiffre décimal 7, on active simultanément les bits se trouvant aux positions 1, 2 et 3.

Dans le schéma ci-contre, la 1ère rangée ne contient aucun point actif : les points qui lui sont associés sont donc 0, 0, 0 (octets 1, 2 et 3 respectivement). A la 2e rangée, sont actifs les points 12, 13, 14 qui appartiennent à l'octet 2 et valent 16+8+4=28 ; elle est donc représentée par le groupe 0, 28, 0.

De même, pour le calcul des valeurs de la rangée 10, on a :

SCHEMA DE MEMORISATION D'UN OBJET



3 octets×8 bits=24 colonnes

Rangée n°	Valeurs		
	Octet 1	Octet 2	Octet 3
1	0	0	0
2	0	28	0
3	0	28	0
4	0	28	0
5	0	28	0
6	0	28	0
7	24	28	0
8	24	28	0
9	31	255	192
10	63	255	224
11	31	255	192
12	24	28	0
13	24	28	0
14	0	28	0
15	0	28	0
16	0	28	0
17	0	28	0
18	0	28	0
19	0	0	0
20	0	0	0
21	0	0	0

Points actifs de l'octet 1 : 3, 4, 5, 6, 7, 8 = =32+16+8+4+2+1=63
Points actifs de l'octet 2 : tous, soit =255
Points actifs de l'octet 3 : 17, 18, 19 = =128+64+32=224

La rangée 10 est donc représentée par le groupe de valeurs 63, 255, 224.
L'archivage de la table ainsi constituée est réalisé à l'aide de l'instruction POKE. En supposant que la mémorisation commence à la position 896 (valeur prise uniquement à titre d'exemple), il faudra 3 instructions pour mémoriser la première rangée (une instruction par octet) :

```
POKE 896,0      1er octet
POKE 897,0      2e octet
POKE 898,0      3e octet
```

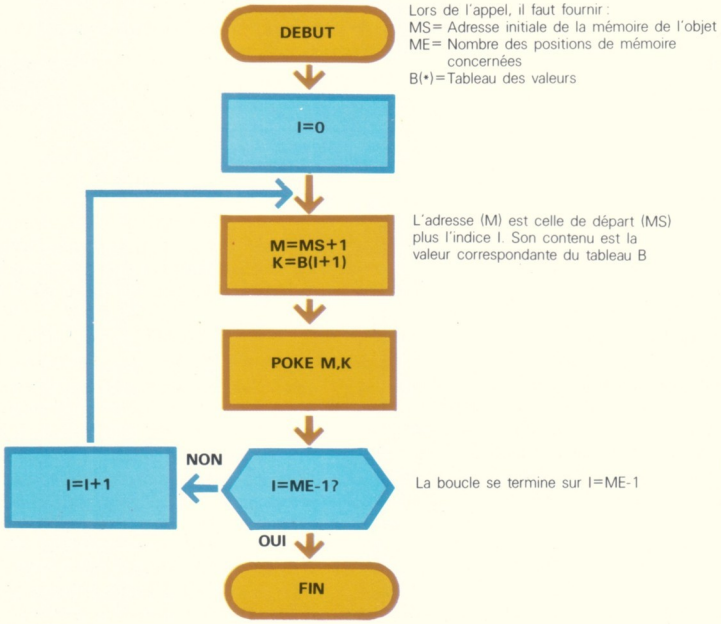
tandis que la 2e rangée sera entrée à l'aide de :

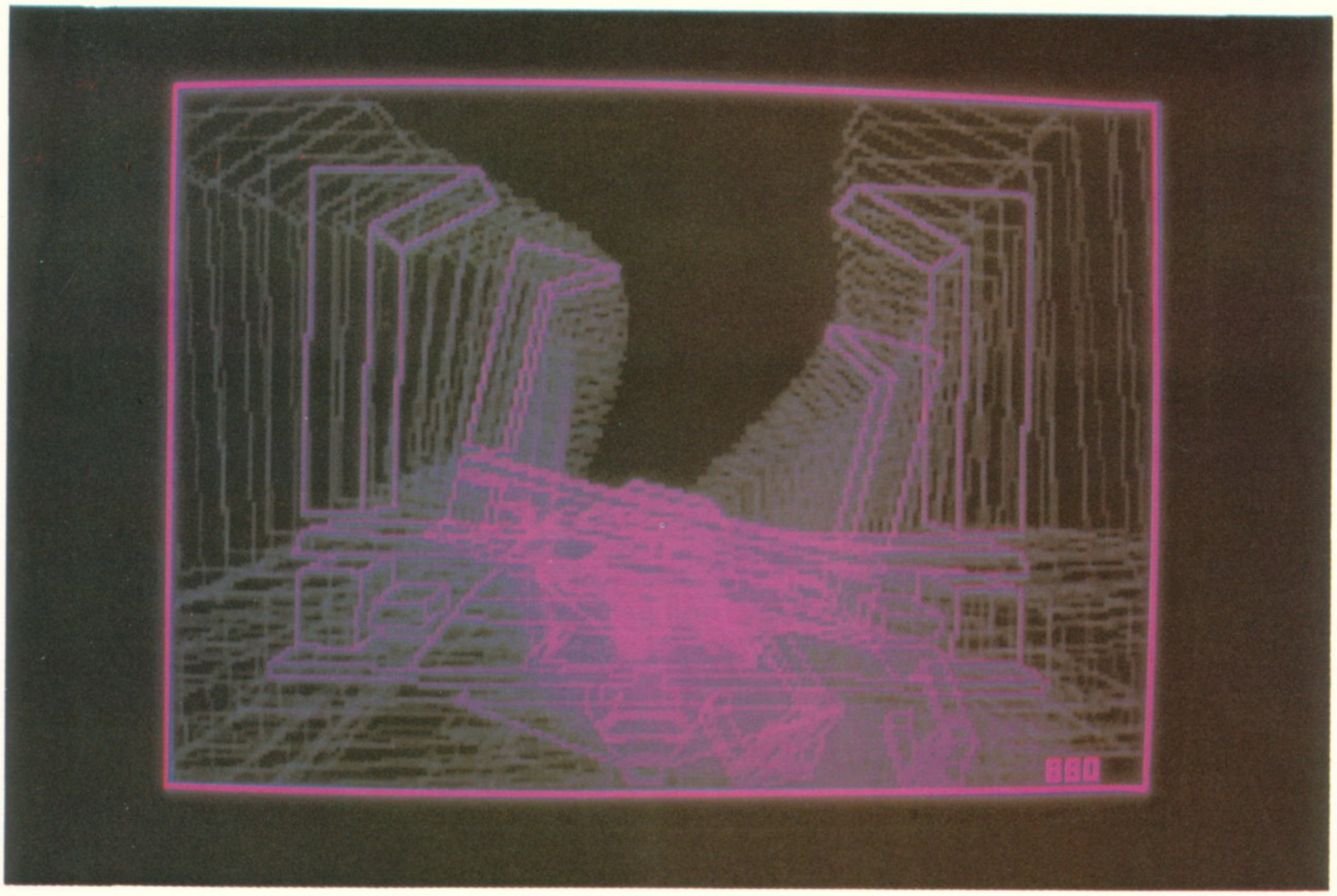
```
POKE 899,0
POKE 900,28
POKE 901,0
```

Il n'est évidemment pas nécessaire d'écrire autant d'instructions qu'il existe de positions de mémoire à activer. La meilleure solution consiste à utiliser une boucle depuis la position de départ jusqu'à la position finale.
L'organigramme ci-dessous représente un sous-programme de mémorisation de la table de la page précédente.
Les variables utilisées sont :

- B (63) = Contient les 3 octets de chaque rangée (3 octets × 21 rangées = 63 octets)
- MS = Position de mémoire de départ
- ME = Nombre de positions de mémoire à activer. Pour l'ensemble de la table, il faut entrer ME=63 (64 si l'on tient compte de l'octet nul qui achève la description).

EXEMPLE DE MEMORISATION D'UN OBJET





P.A. Simon/Grazia Neri-Phototake

Une image animée réalisée pour le film « Tron ».

La table de difinition est rangée dans n'importe quelle zone de la mémoire vive (RAM) et pas nécessairement à proximité immédiate des définitions d'autres objets.
En effet, un registre pointeur permettra de retrouver chacune d'entre elles. Dans le Commodore 64, qui gère jusqu'à 8 objets différents, les pointeurs des définitions des objets occupent les positions 2040 à 2047.
Dans les applications, il est toutefois utile de réserver, à la mémorisation des tables de description des objets, une zone unique divisée en autant de blocs de 64 octets qu'il y a d'objets à gérer.
En supposant que la zone mémoire réservée aux objets commence à la position 832, l'objet défini précédemment (et chargé à la position 896) se trouvera dans le bloc 2. Après son activation, il faudra donc écrire la valeur 14 (12 + 2) à la position réservée à l'adressage (2040).
Le système peut ainsi retrouver sa position en effectuant la multiplication 14×64= 896.

Visualisation de l'objet animé. Une fois mémorisée la table, on doit activer la logique de visualisation de l'objet. La gestion des objets (présentation, déplacement, etc.) se fait généralement en activant ou en désactivant des positions de mémoire spéciales et préprogrammées.
Le système étant capable d'accepter plusieurs objets il faut lui indiquer sur lequel on désire travailler. Cette fonction est généralement activée en mettant à 1 un bit d'une position mémoire particulière. Par exemple, en écrivant 3 (décimal, c'est-à-dire 11 binaire) à cette adresse, on déclare actifs les objets n° 0 et n° 1.
Dans le Commodore 64, le registre de contrôle de la visualisation des objets se trouve à l'adresse 53269. Si le bit n° 0 de ce registre est mis à 1, la visualisation de l'objet n° 0 est activée. Il en sera de même pour tous les autres objets.
Toutefois il n'y aura visualisation que si le point de l'écran où se trouve l'objet est spécifié. Pour cela, chaque objet est contrôlé par deux registres dans lesquels il est nécessaire d'entrer les coordonnées (en points écran) du point désiré. Par exemple, la position de l'objet 0 est contrôlée par les registres 53248 et 53249 .
Dans le premier, c'est la valeur de la coordon-

née X qui devra être chargée (POKE), alors que dans le second c'est celle de la coordonnée Y. La visualisation de l'objet 0 (mémorisé dans le bloc 2) au point 100, 150 s'obtiendra par :

```
30 POKE 2040,2
40 POKE 53248,100
50 POKE 53249,150
60 POKE 53269,1
```

Déplacement de l'objet. Pour déplacer l'objet visualisé, il faut, à l'aide d'une boucle, faire varier les coordonnées de sa posotion contenues dans les deux registres de contrôle.
Dans le cas considéré, le déplacement horizontal de l'objet 0, s'obtiendra par :

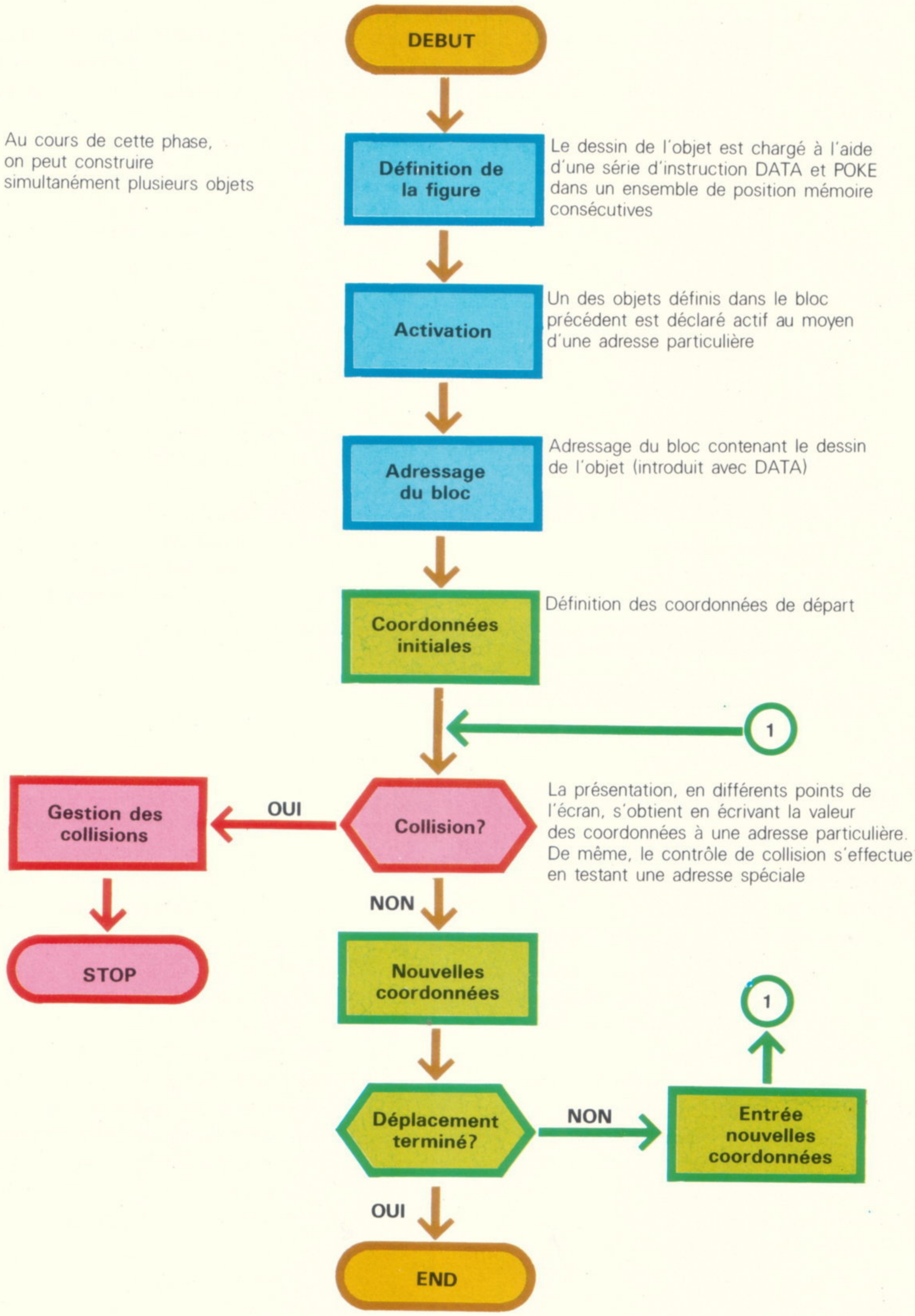
```
100 POKE 53249,100
110 FOR I = 50 TO 150
120 POKE 53248,I
130 NEXT I
```

La ligne 100 fixe la valeur de Y, tandis que la boucle attribue les valeurs 50 à 150 à la coordonnée X.
Il est souvent nécessaire d'associer un contrôle de collisions au déplacement d'un objet. Là encore, il existe des positions mémoire spéciales dont le contenu indique si une collision s'est produite.
Dans le Commodore 64, la collision entre deux objets est contrôlée par le registre 53278 et celle objet-fond par le registre 53279 (voir organigramme ci-contre).

Utilisation de la mémoire pour la gestion des objets animés. La zone mémoire du Commodore 64 chargée de la gestion des objets est schématisée en page 1676 ; elle commence à la position 53248.
Les 16 premières cellules sont les registres de positionnement des 8 objets que la machine peut gérer et dans lesquelles seront chargées les coordonnées de leur positionnement.
Vient ensuite un registre (53264) qui stockera les bits les plus significatifs des coordonnées X (1 bit par objet). Les 4 octets suivants ne concernent pas les objets animés. Le registre d'activation/désactivation est à l'adresse 53269 (adresse relative 21). En mettant à 1 un bit de ce registre, on active l'objet correspondant.
Aux adresses 53271 et 53277 se trouvent deux registres qui permettent de doubler les dimensions de l'objet, le premier le long de l'axe X et le second le long de l'axe Y, en mettant à 1 le bit correspondant (0 pour l'objet 0, 1 pour l'objet I, etc).
Le registre 53275 établit la priorité d'un objet par rapport au fond. Par exemple, si le bit n° 3 de ce registre a la valeur 0, l'objet 3 a la priorité sur le fond (c'est-à-dire qu'il couvrira le fond). S'il vaut 1, ce sera l'inverse.
Les positions 53278 et 53279 contiennent les indicateurs de collision objet-objet et objet-fond. Si l'objet 2 entre en collision avec l'objet 3, les bits 2 et 3 du registre 53278 seront mis à 1 ; il en sera de même en cas de contact avec le fond.
Pour finir, les registres 53287 à 53294 permettront de choisir la couleur des objets en chargeant les codes numériques correspondants (0 = blanc, 1 = noir, etc).
Les autres registres, assez nombreux, sont utilisés pour exécuter des fonctions particulières sur lesquelles nous ne nous arrêterons pas.
En revanche, voyons plus en détail la question de la mémorisation des objets.
Nous avons dit que la description d'un objet se compose d'une suite de 63 + 1 octets consécutifs (le dernier contenant toujours la valeur 0) mémorisés (instruction POKE) à partir d'une adresse de départ.
L'adresse de cette position doit ensuite être mémorisée dans le registre pointeur correspondant à l'objet en question.
En fait, ce qui y est réellement mémorisé ce n'est pas une adresse mémoire, mais un numéro d'ordre d'un bloc de mémoire de 64 octets. En multipliant cette valeur par 64, le système obtient l'adresse réelle à laquelle commence la description de l'objet.
Cette logique est détaillée page 1677.

Exemple d'application. Pages 1678 et 1680, on trouvera l'organigramme d'un jeu utilisant 4 objets animés multicolores.
Le jeu consiste à déplacer horizontalement un parachutiste afin d'éviter les collisions avec une série d'objets qui apparaissent de façon aléatoire. Le jeu est gagné si le parachutiste réussit à toucher le sol.
Pour compliquer sa descente et faire perdre son contrôle au joueur, un vent latéral déplacera le parachute de gauche à droite.

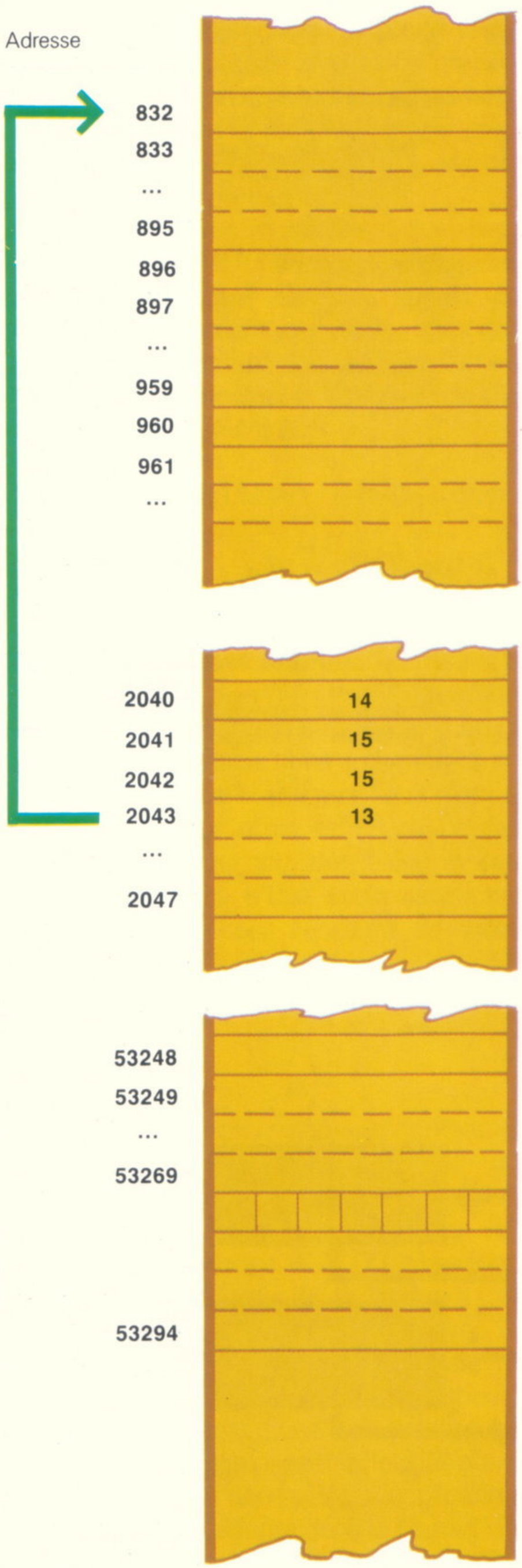
SCHEMA LOGIQUE DE GESTION DES OBJETS ANIMES



ORGANISATION DES REGISTRES DES OBJETS DANS LE COMMODORE 64

Bit n°		7	6	5	4	3	2	1	0	
Base = 53248		DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
	0									OBJET 0 X
	1									OBJET 0 Y
	2									OBJET 1 X
	3									OBJET 1 Y
	4									OBJET 2 X
	5									OBJET 2 Y
	6									OBJET 3 X
	7									OBJET 3 Y
	8									OBJET 4 X
	9									OBJET 4 Y
	10									OBJET 5 X
	11									OBJET 5 Y
	12									OBJET 6 X
	13									OBJET 6 Y
	14									OBJET 7 X
	15									OBJET 7 Y
53264	16									Bits de plus fort poids
	17									de la valeur X
	18									TABLE
	19									PHOTOSTYLE X
	20									PHOTOSTYLE Y
53269	21									Validation objet
	22									
53271	23									Augmentation Y de l'objet
	24									Mémoire vidéo de l'objet
	25									Demandes d'interruption
	26									MASQUES interruptions
53275	27									Priorité fond de l'objet
	28									Sélection objet multicolore
53277	29									Augmentation X de l'objet
53278	30									COLLISION Objet-Objet
53279	31									COLLISION Objet-Fond
	32			INFORMATIONS COULEUR						Marge
	33									Fond 0
	34									Fond 1
	35									Fond 2
	36									Fond 3
	37						Objets multicolores			OMC 0
	38									OMC 1
53287	39									Couleur objet 0
	40									Couleur objet 1
	41									Couleur objet 2
	42									Couleur objet 3
	43									Couleur objet 4
	44									Couleur objet 5
	45									Couleur objet 6
53294	46									Couleur objet 7

UTILISATION DE LA MEMOIRE DANS LA GESTION DES OBJETS ANIMES



Début DATA objet n° 3 (bloc 13)

Début DATA objet n° 0 (bloc 14)

Début DATA objets n° 1 et 2 (bloc 15)

Table d'adressage des définitions des objets

L'adresse 2040 contient la valeur 14 ;
l'objet n° 0 est donc défini dans le bloc 14

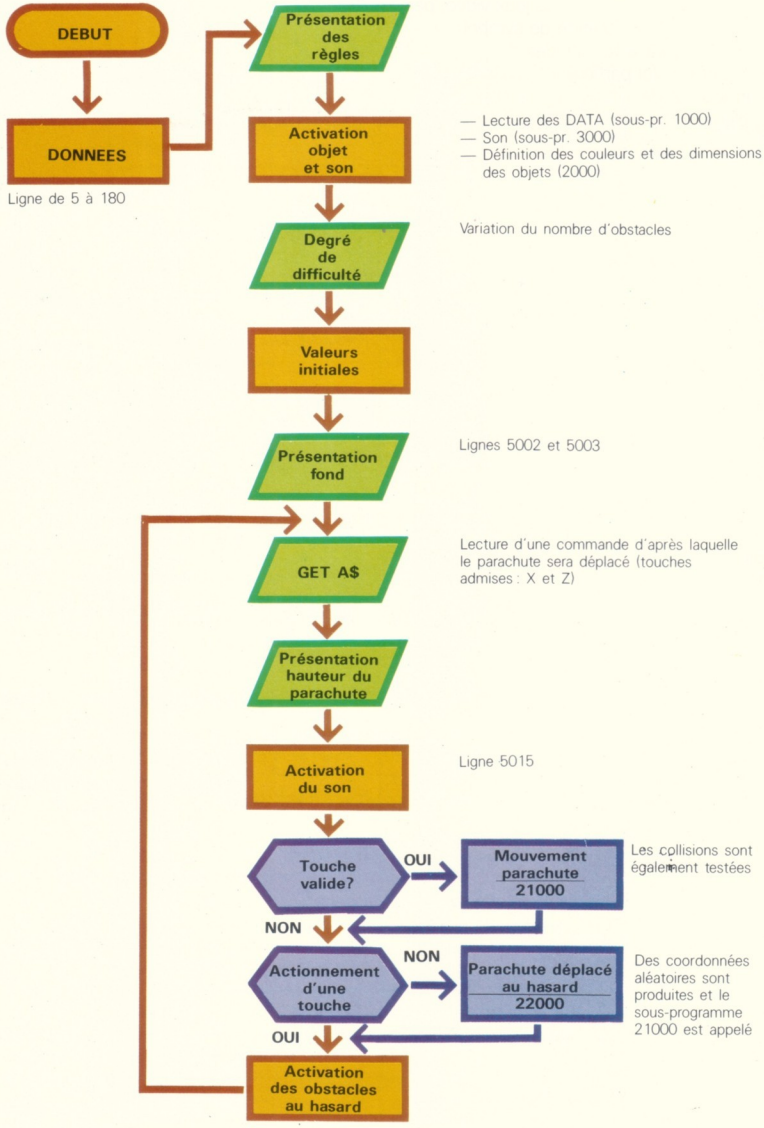
Les objets 1 et 2 ont la même description (même bloc de données). Ils sont donc identiques

Début de la table de gestion

Indicateur d'activation des objets

Fin de la table de gestion

ORGANIGRAMME DU PROGRAMME PARACHUTES



Gestion évoluée des objets animés

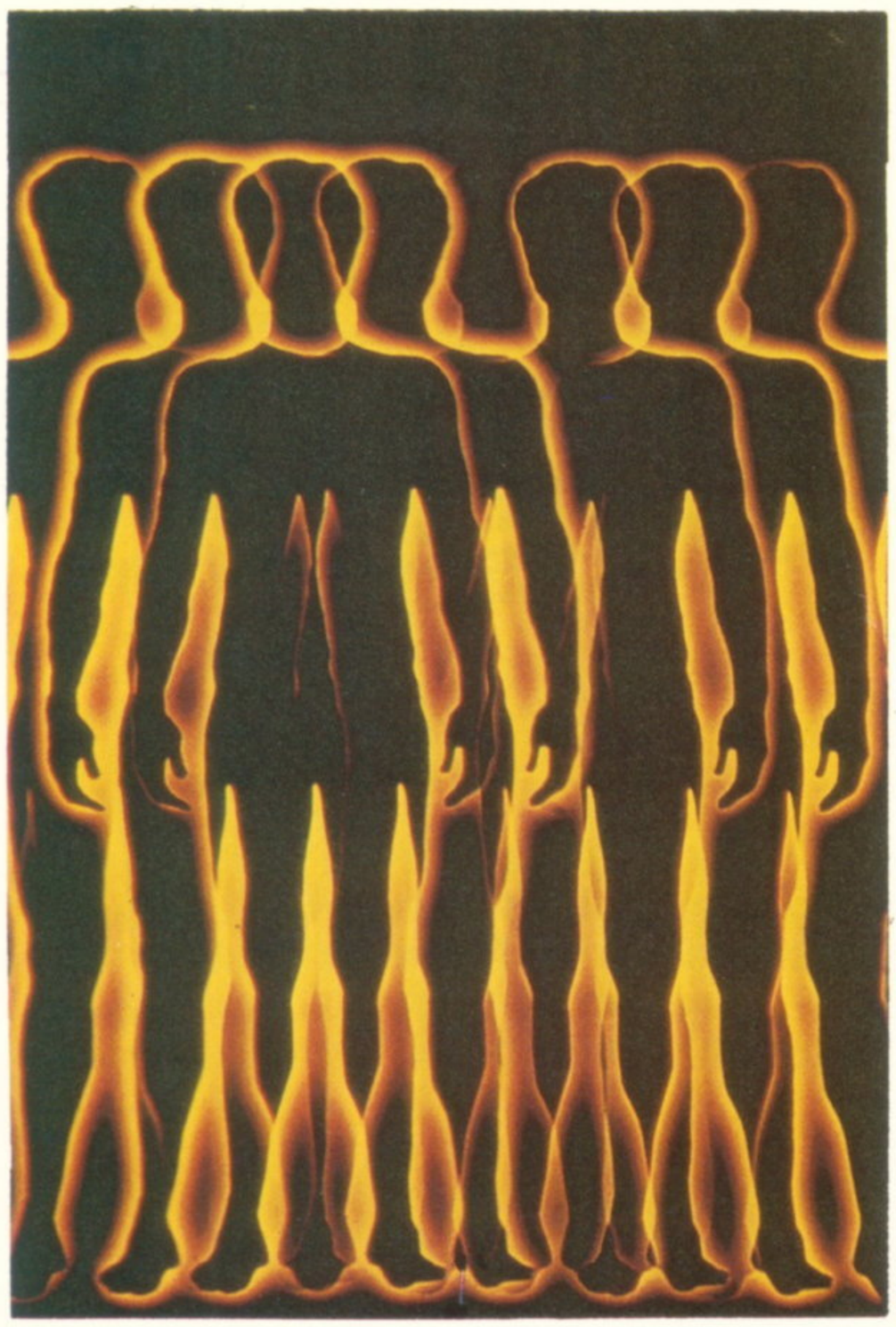
Les objets animés se retrouvent dans d'autres applications graphiques que les jeux vidéo, par exemple dans la préparation de symboles élémentaires servant à former des dessins complexes. Cet emploi particulier nécessite toutefois un logiciel de gestion évolué permettant d'appeler les fonctions nécessaires directement en Basic, sans s'occuper de questions aussi complexes que l'adressage de la mémoire ou que l'affectation des tables.
Les machines les plus modernes, notamment dans la catégorie des ordinateurs personnels, acceptent des versions de Basic avancé qui prévoient non seulement des instructions graphiques, mais également une gestion des objets animés. Les principales fonctions implémentées sont les suivantes :

— Définition des objets animés comme variables
— Présentation
— Contrôle des collisions

Définition des objets animés. La méthode reste la même que celle qui vient d'être décrite. Elle consiste à réserver une zone de mémoire dans laquelle l'image binaire de l'objet est transférée. Mais, contrairement aux formes précédentes, chaque objet est, dans ce cas, défini comme une variable de chaîne et sera donc chargé à l'aide d'une seule instruction DATA, sans passer par l'écriture de tables d'adressage.
Voici une forme très employée de cette instruction :

```
SPRITE$(N)=B$
```

Elle définit l'objet N comme égal au contenu de la variable de chaîne B$, qui est à son tour chargée au moyen d'une instruction DATA.
Cette instruction, comme les suivantes, est propre au système Philips VG8000, mais fonctionne également sur tous les ordinateurs utilisant le Basic Standard MSX. Cette forme de Basic est très proche du Basic 80 sous CP/M, tout en étant plus riche en instructions spécifiques. Le Basic MSX est d'une diffusion récente mais, du fait de sa remarquable facilité d'emploi (notamment dans les applications graphiques) ; il a été adopté dans la plupart des machines basées sur le microprocesseur Z80.



Marka

Effet de superposition d'images produit par l'ordinateur.

Visualisation. La forme la plus simple de cette instruction est

```
PUT SPRITE P, (X, Y), C, N
```

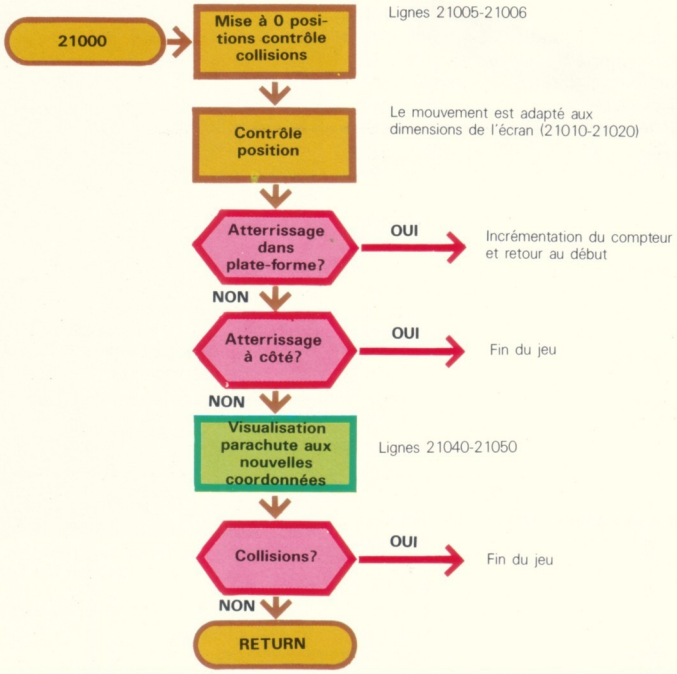
dans laquelle :

P : Indique la priorité de l'objet (pour éviter d'éventuels conflits en cas de présentation de deux objets au même point de l'écran)
X, Y : Sont les coordonnées
C : Est le nombre qui désigne la couleur choisie, de 0 à 15
N : Est le n° d'identification de l'objet, défini dans l'instruction SPRITE$(N)

La syntaxe présentée est, là aussi, propre au système Philips, également utilisable avec les ordinateurs acceptant le Basic MSX. Mais cette instruction n'établit pas les dimensions de l'objet, qui doivent être préalablement définies au moyen de l'instruction SCREEN,

```
SCREEN A, B, C, D
```

SOUS-PROGRAMME DE DEPLACEMENT ET DE CONTROLE



CARACTERES PARTICULIERS UTILISES DANS LE PROGRAMME PARACHUTE VERSION CMB 64

CHR $	Fonction	CHR $	Fonction
147	Vider l'écran	144	Ecrire en noir
18	ON inversé	31	Ecrire en bleu
146	OFF inversé	5	Ecrire en blanc
159	Ecrire en bleu clair	17	Déplacer le curseur vers le bas
28	Ecrire en rouge		

Positions de mémoires

650,128 — Répétition automatique

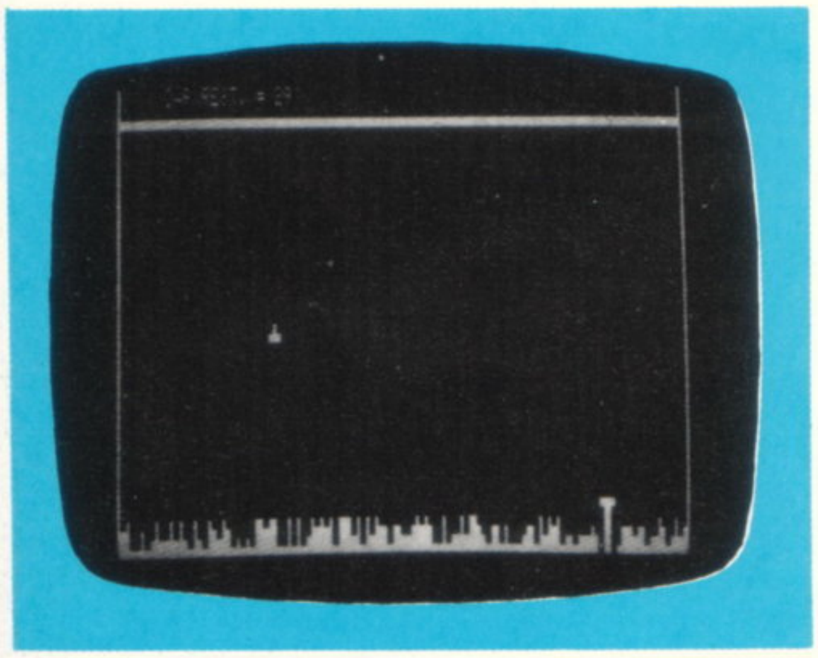
53248+28,8 — Définition objet multicolore

53248+42, 53248+37, 53248+38 — Couleurs de l'objet

53248+29,1 — Doubler la dimension de l'objet n° 0 (décimal 1) sur l'axe X

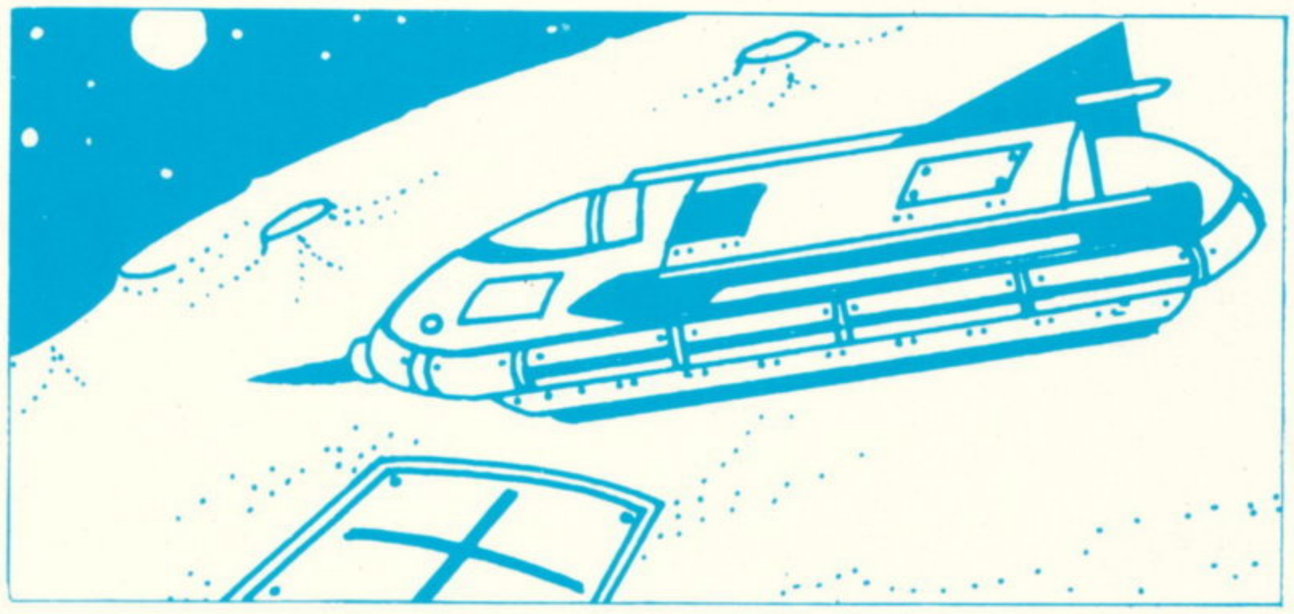
53278, 53279 — Collisions

Instruction d'activation de l'objet N.''n'' :
POKE S+21, PEEK (S+21) AND ''n''



ATTERRISSAGE

Un jeu pour TRS-80

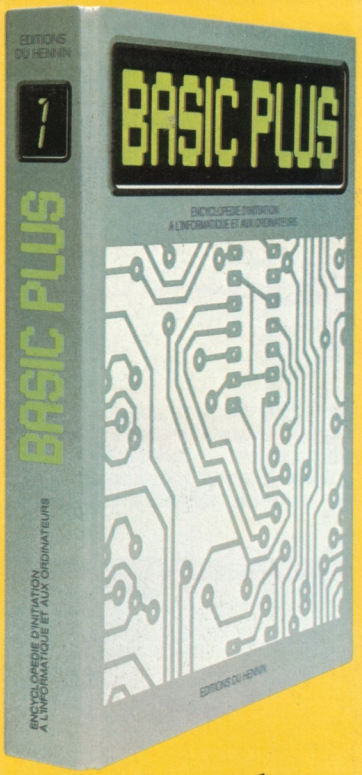


Le but de ce jeu est de réussir à poser votre vaisseau spatial sur l'aire d'atterrissage prévue à cet effet. Utilisez les touches de déplacement du curseur gauche et droite pour modifier votre trajectoire et la touche de déplacement du curseur vers le bas pour freiner votre descente. Attention ! Vous disposez d'une quantité limitée de carburant qui dépend du niveau dans lequel vous évoluez. Le jeu se termine lorsque vous réussissez à atterrir.

```
10 REM ** ATTERRISSAGE **
20 REM DE CHRIS PALMER
30 REM INITIALISATION DES VARIABLES
40 CLS:INPUT"NIVEAU 1-5";N
50 IF N>5 OR N<1 THEN GOTO 40
60 CA=N*140
70 GOSUB 1000
80 T=-1:TL=-1:TR=-1:PX=32:PY=5
90 GP=1:GC=1.003
100 REM BOUCLE PRINCIPALE
110 GOSUB 200
120 GOSUB 300
130 GOSUB 900
140 GOSUB 400
150  IF CF=1 THEN GOTO 600
160 IF SL=1 THEN GOTO 700
190 GOTO 100
200 REM INITIALISATION ECRAN
210 A$=INKEY$:IF A$="" THEN RETURN
220 IF ASC(A$)=10 THEN T=T*-1
230 IF ASC(A$)=9 THEN TR=TR*-1
240 IF ASC(A$)=8 THEN TL=TL*-1
250 RETURN
300 REM CALCUL DES VALEURS
305 OX=PX:OY=PY
310  IF TR=1 AND CA>0 THEN UX=UX-.2:CA=CA-
1.5
320 IF TL=1 AND CA>0 THEN UX=UX+.2:CA=CA-1
.5
330 IF T=1 AND CA>0 THEN UV=-.5:CA=CA-3
340 IF T<>1 THEN UV=.7
350 PY=PY+UV:PX=PX+UX
355 IF CA<0 THEN CF=1
360 RETURN
400 REM PLOT VAISSEAU
405 IF PX>126 OR PX<1 THEN CF=1:RETURN
406 IF PY<4 OR PY>43 THEN CF=1:RETURN
407 IF INT(PX)=X*2 AND INT(PY)>40 THEN SL=
1:RETURN
410 RESET(OX,OY):RESET(OX+1,OY):RESET(OX+2
,OY):RESET(OX+1,OY-1)
420 SET(PX,PY):SET(PX+1,PY):SET(PX+2,PY):S
ET(PX+1,PY-1)
430 RETURN
600 REM ACCIDENT
610 CLS:PRINT:PRINT"   PERDU VOUS VOUS ETE
S ECRASE "
620 PRINT:PRINT"  RETOURNEZ A L'ECOLE DE P
ILOTAGE"
630 END
700 REM ATTERRISSAGE REUSSI
710 CLS:PRINT:PRINT"        BRAVO ! ! ATTE
RRISSAGE REUSSI"
720 PRINT:PRINT" VOUS ETIEZ EN NIVEAU";N;"
 ET AVEZ UTILISE"
730 PRINT"  ";CA;" LITRES DE CARBURANT."
740 PRINT:PRINT"     SCORE";(6-N)*CA
750 END
900 REM MISE A JOUR CARBURANT
910 PRINT @ 16,"      ";:PRINT @ 16,CA;
940 PRINT @ 30," ";:PRINT @ 35," ";:PRINT
@ 40," ";
950 IF T=1 THEN PRINT @ 35,CHR$(92);
960 IF TL=1 THEN PRINT @ 30,">";
970 IF TR=1 THEN PRINT @ 40,"<";
980 RETURN
1000 REM INITIALISATION AIRE
1010 CLS:FOR I=0 TO 127
1020 Y=RND(4):SET(127-I,3)
1025 SET(0,ABS(I*.370079)):SET(127,ABS(47-
I*.370079))
1030 FOR D=1 TO Y
1040 SET(I,48-D)
1050 NEXT D
1060 NEXT I
1070 X=RND(50)+5
1080 PRINT @ X+960,CHR$(170);CHR$(149);
1090 PRINT @ X+896,CHR$(171);CHR$(151);
1100 PRINT @ 5,"CAR REST. =";
1110  RETURN
```

Tiré de «Jeux en BASIC sur TRS-80»
de Chris Palmer, Editions SYBEX,
Réf. 302, Ft 16 × 22, 96 p., 49 F.

SYBEX 6-8, impasse du Curé
75881 PARIS CEDEX 18
Tél. : 203.95.95



PROTEGEZ VOS FASCICULES

ET CONSTITUEZ VOUS-MÊME VOTRE ENCYCLOPEDIE D'INITIATION A L'INFORMATIQUE ET AUX ORDINATEURS.

45 FF . 290 FB . 14,50 FS
chaque reliure mobile contient
12 fascicules

NOS RELIURES MOBILES SONT EN VENTE CHEZ TOUS LES MARCHANDS DE JOURNAUX