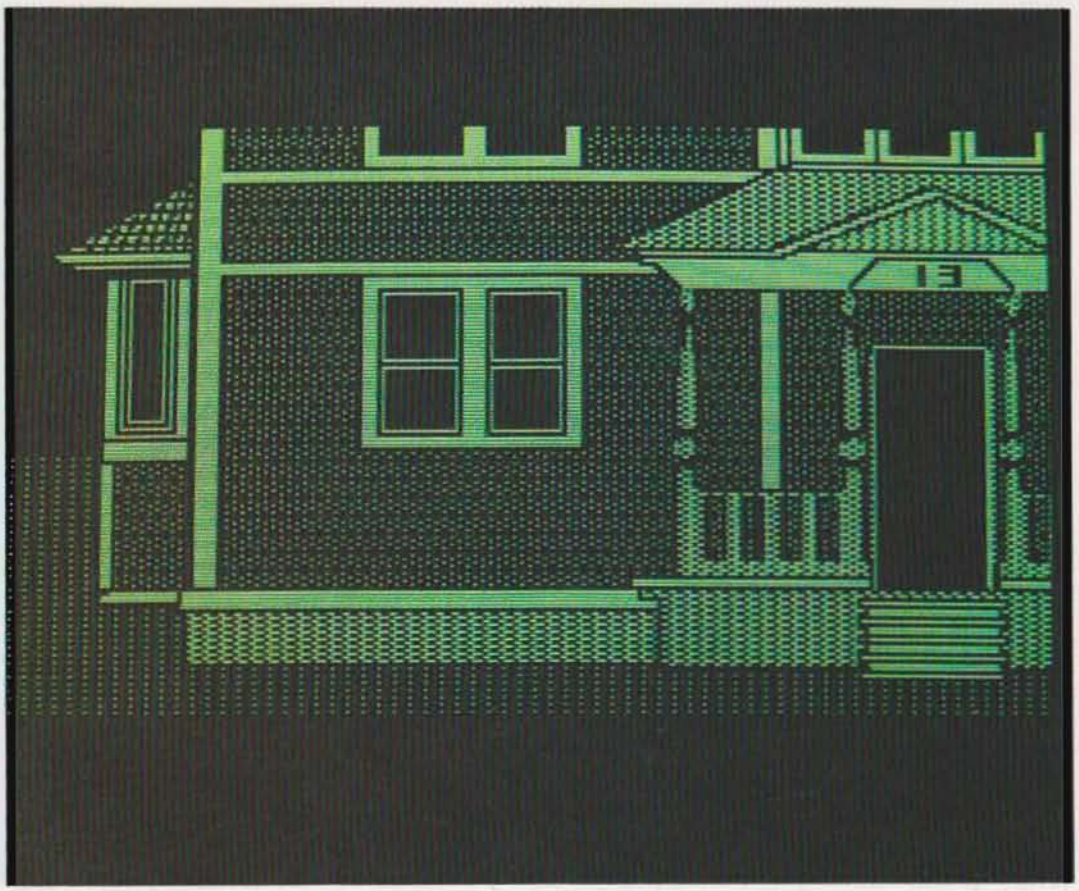
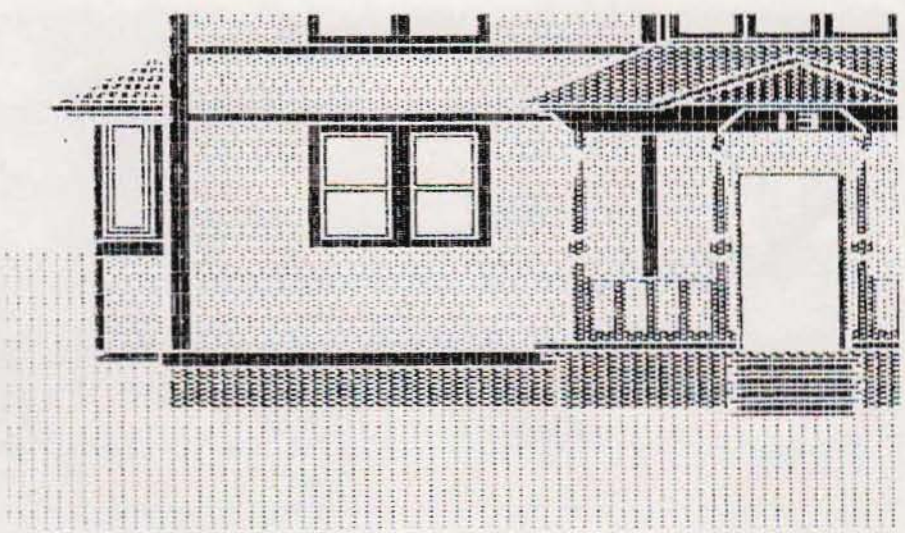
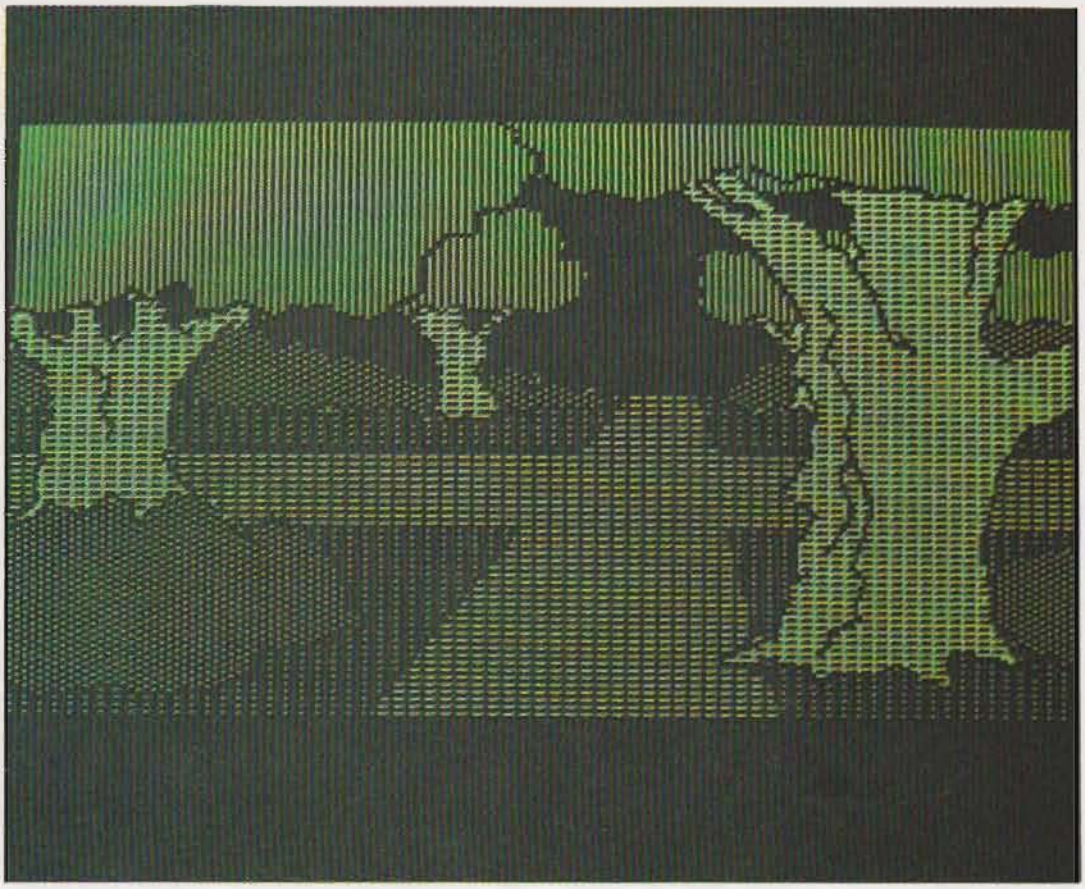
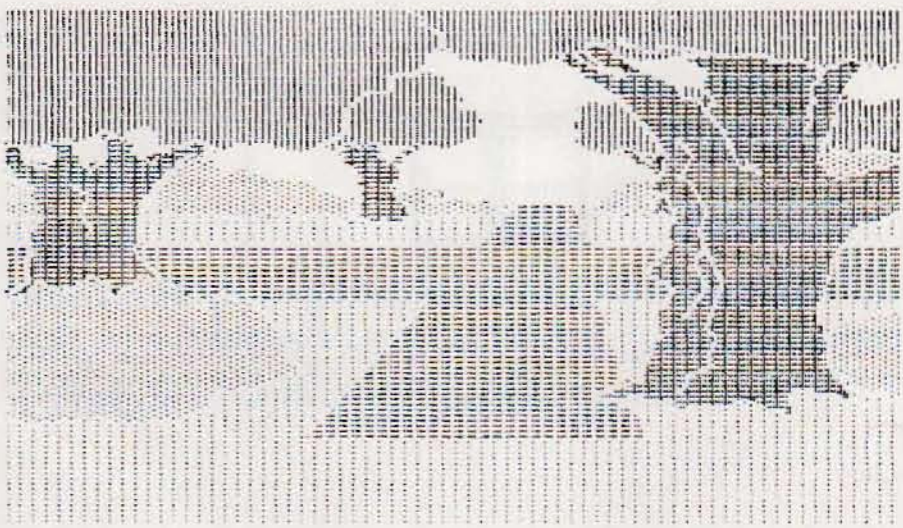
SORTIE SUR IMPRIMANTE D'UNE IMAGE GRAPHIQUE (1)





SORTIE SUR IMPRIMANTE D'UNE IMAGE GRAPHIQUE (2)

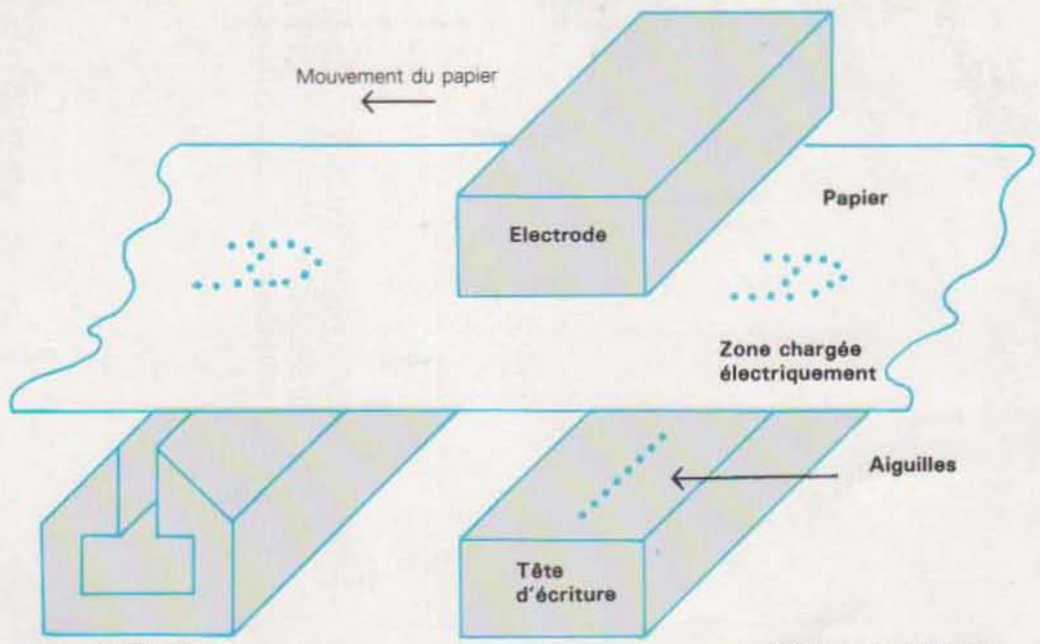




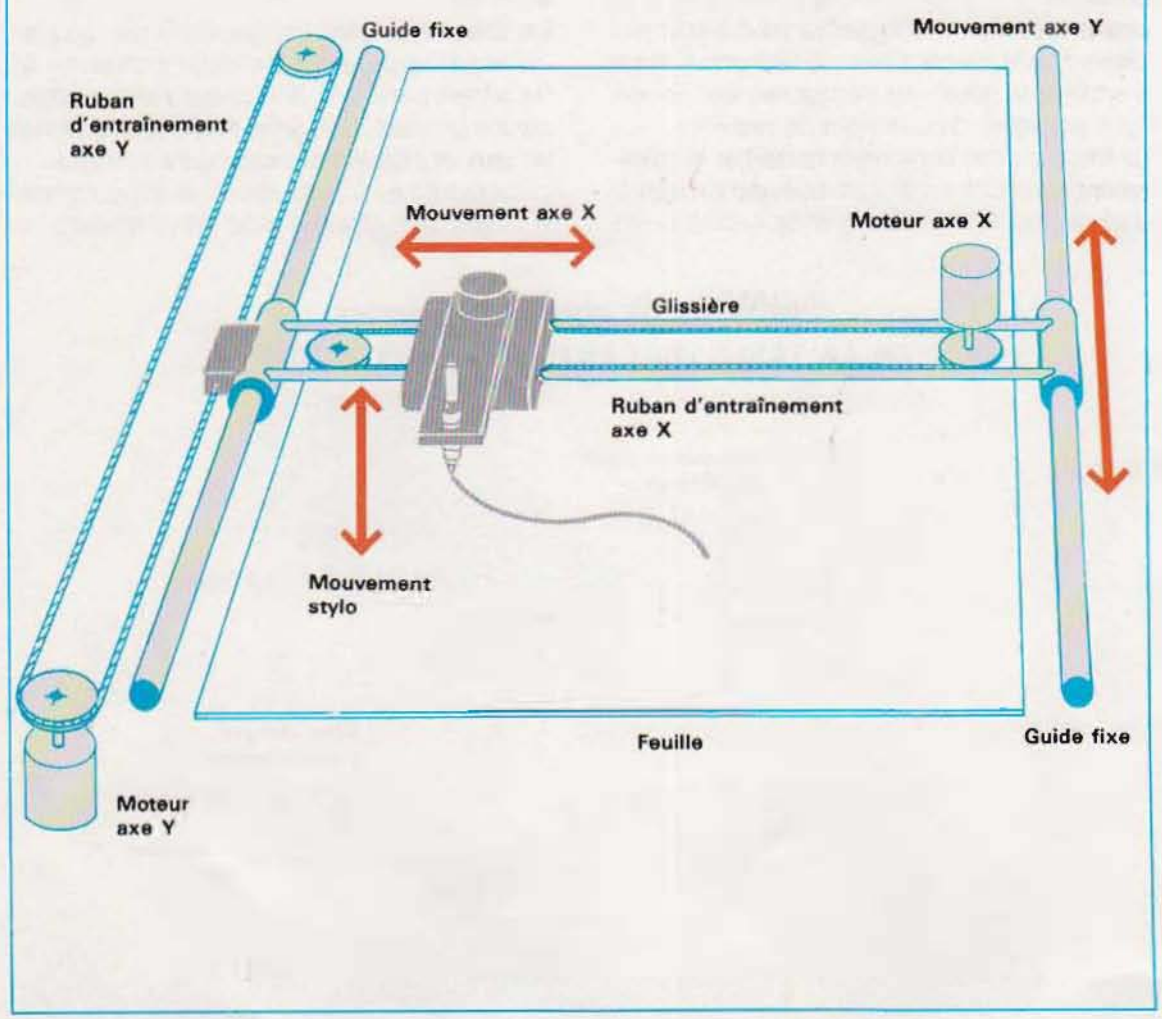
- traceurs avec photostyle
- traceurs électrostatiques

Dans les premiers, l'élément qui trace le dessin est un stylo très semblable à celui employé par les dessinateurs ; dans les électrostatiques, également appelés **rasters,** l'écriture est réalisée en chargeant électriquement les points du papier qui devront être colorés et en déposant un applicateur sur le papier. Ce procédé se rapproche de l'impression électrostatique (schéma ci-dessous). Ce type de périphérique présente deux inconvénients : le recours à un papier spécial et l'impossibilité de dessiner plusieurs fois le même point ; il faut donc, avant d'envoyer le dessin au traceur, le décomposer ligne par ligne (d'où le nom de raster).
Le traceur avec photostyle constitue le matériel le plus courant, car il associe une excellente qualité graphique à une grande facilité d'emploi. Le style se déplace dans toutes les directions et repasse sans difficulté sur des zones déjà partiellement dessinées. Cette caractéristique laisse une grande liberté de manœuvre à l'usager qui pilote son logiciel comme il l'entend, sans entrave imposée par la structure physique de la machine.
Les traceurs se classent également selon le type de mouvement effectué. Les plus répandus sont les traceurs à plat et ceux à tambour. D'autres, comme les traceurs à rouleau ou avec mouvement de papier, n'en sont que des variantes.
Le traceur à plat est constitué par un plan qui reçoit le papier et par deux « chemins de fer » fixes parallèles, le long desquels se déplace une glissière. Sur cette glissière, on a monté un chariot porte-stylo (voir figure p. 1402).
Les mouvements (X,Y) nécessaires pour tracer le dessin sont obtenus avec deux moteurs : le

PRINCIPE DE FONCTIONNEMENT DE LA TABLE TRAÇANTE ELECTROSTATIQUE



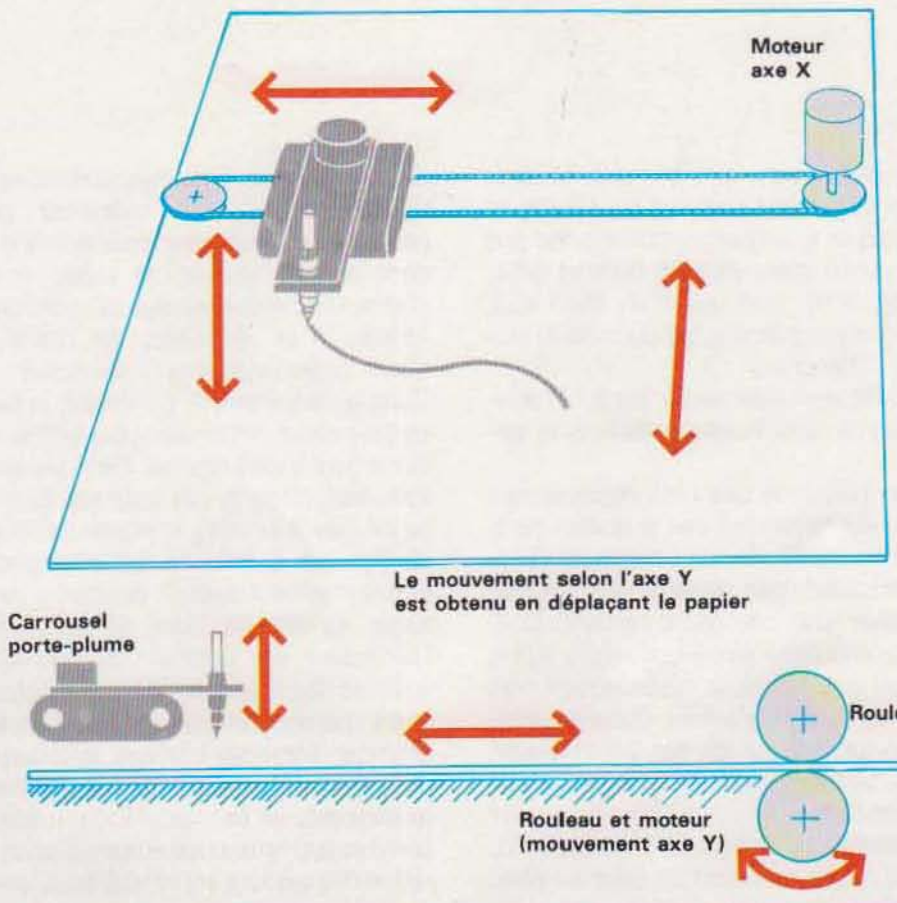
SCHEMA DU TRACEUR A PLAT

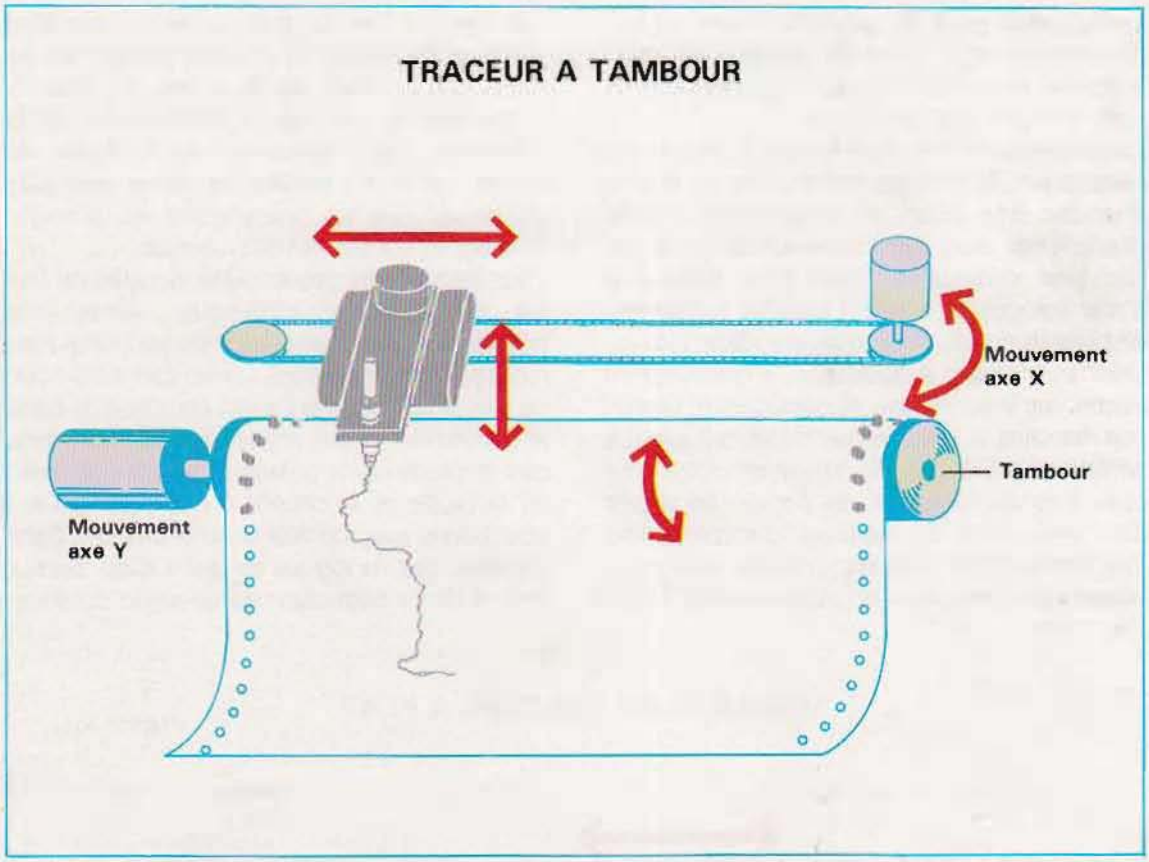


premier déplace la glissière (axe Y), le second déplace le chariot le long de la glissière (axe X). A ces mouvements, s'ajoute le déplacement perpendiculaire de la plume, obtenu grâce à un électro-aimant monté sur le chariot.
La mécanique même de ce périphérique montre à quel point son emploi est extrêmement simple. Deux instructions seulement sont nécessaires, l'une pour déplacer le stylo à un point donné X,Y, et l'autre pour préciser si ce mouvement doit être fait avec le stylo levé ou baissé. Dans le premier cas, on obtiendra un simple positionnement du style (sans traçage de signe) ; dans le second cas (stylo baissé),

s'effectue le tracé du segment reliant la position actuelle et le point de coordonnées X,Y. Avec ces deux instructions, on parvient à réaliser n'importe quel graphique.
Dans certains types de traceurs à plat, on ne dispose pas du mouvement de glissière le long d'un des deux axes ; le déplacement correspondant est alors obtenu en déplaçant le papier (voir ci-dessous). Mais il ne s'agit que d'une variante de matériel puisque l'utilisateur emploie le dispositif de manière identique.
Dans le **traceur à tambour,** le mouvement selon l'axe Y est obtenu en déplaçant le papier, comme dans la variante des traceurs à plat. La différence réside dans le moyen employé pour obtenir le déplacement du papier. Le papier doit avoir des perforations d'entraînement (comme le papier pour imprimantes) et le mouvement est obtenu par un rouleau denté. Naturellement, le rouleau doit pouvoir tourner dans les deux directions, de manière à effectuer les déplacements dans les deux sens de l'axe Y.
L'avantage de ce type de traceur vient de la dimension (sans limitation) de la feuille de papier ; mais il s'agit de machines bien plus coûteuses que les précédentes et rarement employées sur les petits systèmes.
Tous les traceurs peuvent être équipés de stylos de couleurs différentes. Deux cas possibles : dans le premier tous les stylos sont montés en même temps sur un carrousel et on ne baisse que celui qui a été sélectionné. Dans le second, les divers stylos sont déposés dans des emplacements prévus à cet effet au bord de la feuille et le chariot prend celui qui est sélectionné avec un mécanisme à déclic. Dans les deux cas, le logiciel de gestion du traceur doit prévoir l'instruction de sélection du stylo.

VARIANTE DU TRACEUR A PLAT





Il faut prendre garde au fait que le mot « stylo » est également employé pour indiquer le « stylo logique », auquel ne correspond pas de stylo physique mais un type de trait différent. Il s'agit donc, non pas d'un stylo réel, mais d'un sous-programme qui contrôle le tracé de lignes différentes.

Deux autres facteurs intervenant dans l'évaluation d'un traceur sont la **résolution** et la **répétitivité.**

La résolution indique le plus petit déplacement possible, qui est déterminé par la qualité de la mécanique, en particulier des moteurs. D'un type très particulier, ces derniers sont en mesure de réaliser une très petite rotation (pas) pour chaque impulsion électrique reçue : plus la rotation est petite, plus le déplacement correspondant du stylo sera précis. Comme ordre de grandeur, un traceur de qualité moyenne permet des déplacements de quelques dixièmes de millimètre.

Le second paramètre, qui a trait à la répétition, indique avec quelle précision on peut se positionner plusieurs fois sur un même point. En d'autres termes, à un couple de valeurs numériques envoyées par l'ordinateur, ne correspond pas toujours la même position physique de la plume. Dans ce cas aussi, l'erreur a une origine mécanique et elle est normalement inférieure à la résolution. En réalité, elle est moins déterminante qu'il n'y paraît.

Dans les applications générales, le résultat est certainement nettement plus précis que n'importe quel travail manuel. Dans les applications spéciales, comme par exemple la préparation de circuits imprimés, le dessin est d'abord tracé à grande échelle, la réduction photographique permettant dans un deuxième temps d'éliminer les défauts. Dans ce cas, le recours à l'ordinateur est vraiment indispensable car il autorise des essais à différentes échelles, sans autre intervention que la mise en route du programme. Dans les traceurs spéciaux à **laser,** un rayon lumineux remplace le photo-style. Naturellement, on fait appel à du matériel photosensible (comme celui employé pour la préparation des circuits imprimés) avec une structure mécanique étudiée pour obtenir une préci-

sion très poussée (pour certains traceurs, on parvient à 1/10 000 " avec un stylo à lumière **laser**).

Le moniteur monochrome. La technologie employée dans la fabrication des écrans dépend du type d'application auquel ils sont destinés. D'où diversité et différence de coût.
L'organe essentiel d'un poste de télévision est le tube à rayons cathodiques (CRT pour Cathode Ray Tube). C'est un convertisseur transformant un signal électrique en image sur un écran luminescent.
Le fonctionnement d'un CRT repose sur les propriétés de certaines substances (comme par exemple les sels de phosphore) qui émettent de la lumière visible quand elles reçoivent un faisceau d'électrons.
Page suivante, nous avons représenté 2 types de tubes :

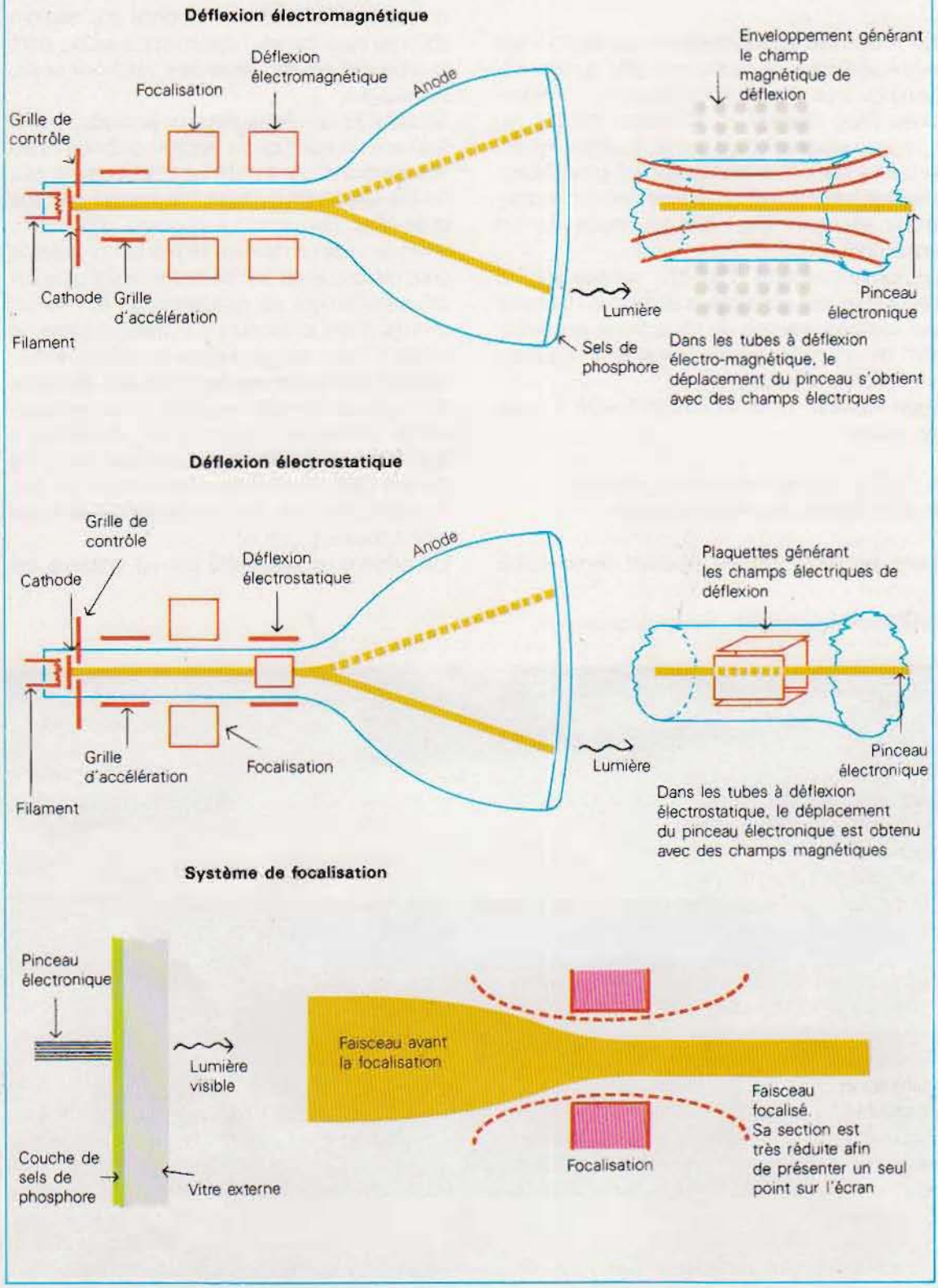
- CRT à déflexion électromagnétique
- CRT à déflexion électrostatique

Dans les deux cas, un filament semblable à celui des ampoules ordinaires est parcouru par un courant électrique qui le rend incandescent. A proximité (parfois il s'agit d'un petit cube coaxial au filament), on interpose une matière spéciale sous forme d'une mince couche, dont la propriété est de libérer des électrons si elle est excitée.
Sous l'effet du réchauffement provoqué par le filament, le composant appelé cathode émet des électrons, qui sont mis en mouvement vers l'autre extrémité du tube (dit anode) par une grille d'accélération. La quantité d'électrons envoyés vers l'anode est réglée par la grille de contrôle qui, avec son potentiel électrique, fait office d'élément de régulation du flux tel un robinet. Sans le secours d'autres éléments, le nuage d'électrons produit par la cathode à destination de l'anode exciterait les sels de phosphore de manière désordonnée, d'où une luminosité diffuse sur l'écran. C'est pourquoi il a fallu focaliser les électrons avec une électrode dans le CRT, de manière à n'atteindre les sels de phosphore que sur une surface très faible (voir schéma p. 1406).
L'ensemble est complété par un système qui

Une imprimante-table traçante couleur.

Marka

SCHEMA D'UN TUBE CATHODIQUE



déplace le faisceau focalisé d'électrons (dit pinceau). Ce résultat s'obtient avec un champ soit magnétique (déflexion magnétique) soit électrique (déflexion électrostatique).
Les champs (magnétique ou électrique) sont perpendiculaires, de manière à réaliser des déplacements de pinceau cartésiens (axes X,Y) balayant toute la surface de l'écran.
Les composants électriques qui règlent l'émission, l'accélération et la focalisation du faisceau d'électrons sont gérés par des dispositifs indépendants de l'ordinateur. Les éléments de déflexion sont, de leur côté, commandés par l'ordinateur : grâce à des interfaces spécifiques, les données sont converties en signaux électriques qui règlent la position du pinceau sur l'écran.
La génération d'une image s'obtient en déplaçant le pinceau électronique selon les instructions envoyées par l'ordinateur sous la forme de signaux de positionnement. Comme pour les traceurs, l'ordinateur doit préciser si un déplacement doit être reproduit avec un trait visualisé ou non. L'émission de la lumière, en un point de l'écran, dure tant que le pinceau électronique excite ce point. Si on doit visualiser deux points sur une même ligne horizontale, à l'instant où le pinceau atteint le second point, les phosphores du premier ne sont plus excités.
Le tracé d'un segment se réalisant par l'excitation successive des points le composant, il faut éviter que, lors de la visualisation des derniers points, les premiers aient perdu leur luminosité.
Une technique simple consiste à exploiter la persistance de l'image, c'est-à-dire la propriété qu'ont les sels de phosphore de continuer à émettre de la lumière même après l'excitation. Mais cette persistance n'excède pas un certain temps. On a alors recours à la technique du rafraîchissement qui consiste à redessiner l'image plusieurs fois par seconde. En général, on ajuste la persistance à la fréquence du rafraîchissement pour obtenir des images de qualité acceptable, en éliminant dans la mesure du possible les phénomènes de scintillement optique. Pour tenir compte de la persistance de l'image sur la rétine, on choisit en général une fréquence égale à 30 Hz : une image visualisée est balayée 30 fois par seconde.
La conservation de l'image avec la technique du rafraîchissement immobilise une zone de mémoire consacrée à l'écran. Elle permet de mémoriser l'état de chaque point. Le système d'exploitation prélève les données de cette mémoire et active en conséquence les divers points de l'écran, en répétant la lecture à 30 Hz. Inconvénient de cette technique : une zone mémoire importante est retirée aux programmes d'application. Son atout : des possibilités importantes qui permettent à l'utilisateur de modifier la présentation graphique des données en intervenant directement sur la mémoire.
Le rafraîchissement reste interdit à certaines applications quand, par exemple, la résolution demandée exige un espace mémoire dédié trop grand. Dans ce cas, on adopte un tube particulier, dit CRT à mémoire, qui a la propriété de conserver l'image pendant un délai remarquablement long. Mais il a aussi son handicap ; il n'autorise ni l'effacement ni la modification d'une partie de l'écran.
Avec le rafraîchissement, il suffit de modifier le contenu de la mémoire : ce qui est présenté à l'écran disparaît et peut être recomposé avec les éventuelles modifications. Dans les tubes à mémoire l'image est fixée en exploitant la persistance : elle ne peut être mise à jour qu'après effacement complet.
Une dernière classification des moniteurs, reposant sur la technique de formation de l'image, propose deux types de rafraîchissement :

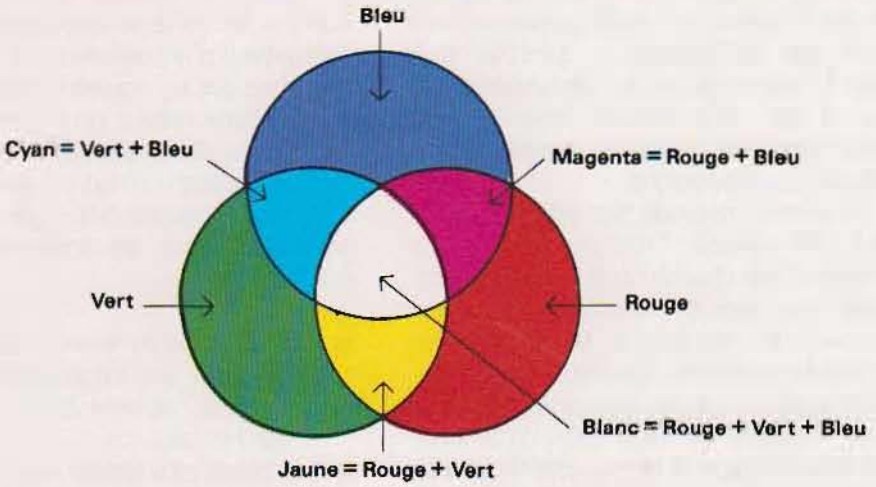
- par vecteurs, normalement indiquée par le terme **stroke**
- par points, appelée **raster** (mode récurrent)

Il existe des variantes de ces méthodes mais peu utilisées sinon abandonnées. C'est le cas de la méthode **starbust**, qui a été délaissée pour sa mauvaise résolution, au profit de techniques certes moins économes en mémoire, mais plus évoluées.
L'augmentation de la capacité de mémoire ne constitue désormais plus un obstacle en termes de coût. On n'est limité que par la capacité d'adressage de l'UC. La méthode de stroke (vecteurs) consiste à guider le pinceau électronique dans n'importe quelle zone de l'écran sans ordre de priorité particulier. Chaque caractère ou élément de dessin est obtenu en déplaçant le pinceau selon une série de mouvements élémentaires.

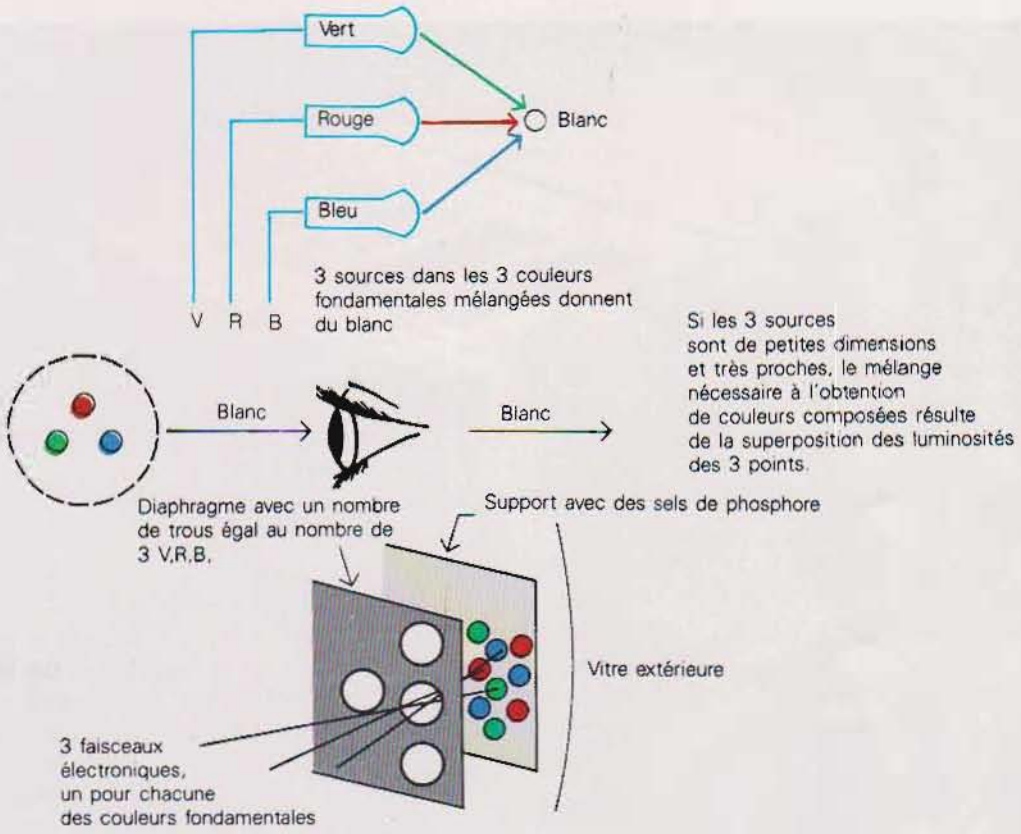
Dans le mode raster (récurrent), l'écran est continuellement balayé par le pinceau électronique, formant une série de lignes imaginaires très rapprochées et presque horizontales. Pour obtenir un point sur l'écran, on envoie un signal, synchronisé avec la position du pinceau, qui inhibe ou active la visualisation.
Avec cette méthode, chaque élément de figure est reproduit en activant de manière sélective une matrice de points, comme dans les imprimantes à aiguilles. La qualité graphique dépend directement du nombre de points adressables sur l'écran. Les valeurs usuelles sont, pour la matrice de caractères, 5 × 7 ou 7 × 9 points, et pour l'écran tout entier de quelques dizaines de milliers à plusieurs millions de points. Normalement, on indique la résolution graphique d'un écran en précisant le nombre de points horizontaux et verticaux.
Exemple : un écran 280 × 190 visualise 280 points consécutifs à l'horizontale et 190 à la verticale, soit un total de 53 000 points pour l'écran.
Une telle définition est peu adaptée aux applications graphiques où les valeurs courantes avoisinent les 1000 × 1000 points. Cette limite est largement dépassée dans certaines applications. Le nombre de points écran est fonction de la granulosité des sels de phosphore de l'écran et surtout de la capacité mémoire.
Pour avoir un ordre de grandeur, prenons un écran 1000 × 1000. Puisque l'état de chaque point (actif/non actif) peut être mémorisé dans un bit, il faut au moins 10^6 bits. Pour une machine à 16 bits, chaque emplacement de mémoire peut justement contenir 16 bits. La taille mémoire dédiée à l'écran sera de $10^6/16 =$ 62 500 octets, l'équivalent de toute la zone adressable pour une UC de 16 bits employée pour des programmes d'applications. Pour une machine à mots de 8 bits, les emplacements mémoire nécessaires sont de 125 000, soit environ le double de la capacité d'adressage. Ces difficultés d'adressage limitent donc la programmation.
Dans leur aspect graphique, les micro-ordinateurs et ordinateurs personnels appartiennent à deux catégories : ceux dont la mémorisation de l'écran passe par une occupation partielle de la mémoire (zone utilisateur) ; ceux qui disposent de ressources séparées. L'avantage des premières est d'offrir un accès facile, à l'utilisateur, des données graphiques. La mémoire est adressable par l'interprète ou le compilateur, entièrement gérée par le programme. La zone mémoire disponible est diminuée de l'espace dédié à la partie graphique. C'est pour cela qu'on ne prévoit généralement pas de valeurs haute résolution.
Dans la seconde catégorie, l'image est conservée dans une zone mémoire n'appartenant pas à l'aire utilisateur, qui garde ainsi toute la partie adressable. Mais on ne peut lire et écrire des données graphiques qu'en recourant à des sous-programmes spécifiques du système, avec des instructions de haut niveau prévues par la machine. En pratique, cela se traduit par une moindre souplesse dans certaines applications, voire par une plus grande lenteur d'exécution des programmes.

Les moniteurs couleur. Le principe de fonctionnement et les techniques de formation et de persistance de l'image sont les mêmes que celles qui sont employées dans les monochromes. En revanche, la structure physique et les composants nécessaires à leur fonctionnement en sont très différents.
La formation d'une image couleur s'obtient de deux manières. On emploie généralement la méthode additive (la même que pour les téléviseurs couleur) alors qu'on retrouve la technique soustractive principalement en photographie et dans les écrans graphiques pour les grands systèmes.
On compte les couleurs primaires et les couleurs composées. On appelle primaires les couleurs qui, combinées de diverses manières, permettent d'en obtenir d'autres.
Dans le graphique ci-contre, on a décrit la technique de formation de certaines couleurs avec la méthode additive, en prenant comme couleurs primaires le rouge, le vert et le bleu. Les couleurs secondaires (blanc, jaune, cyan, magenta) sont obtenues par additions des couleurs fondamentales ; les autres sont générées en variant les proportions ajoutées : par exemple, le jaune est obtenu en mélangeant le vert et le rouge en parts égales, alors que l'orange s'obtient en augmentant la part de rouge.
Dans le second graphique, on a représenté le principe de formation des images couleur. Un point apparaît blanc (ou d'une autre couleur composée) s'il est le point de convergence de

COMPOSITION DES COULEURS AVEC LA TECHNIQUE ADDITIVE



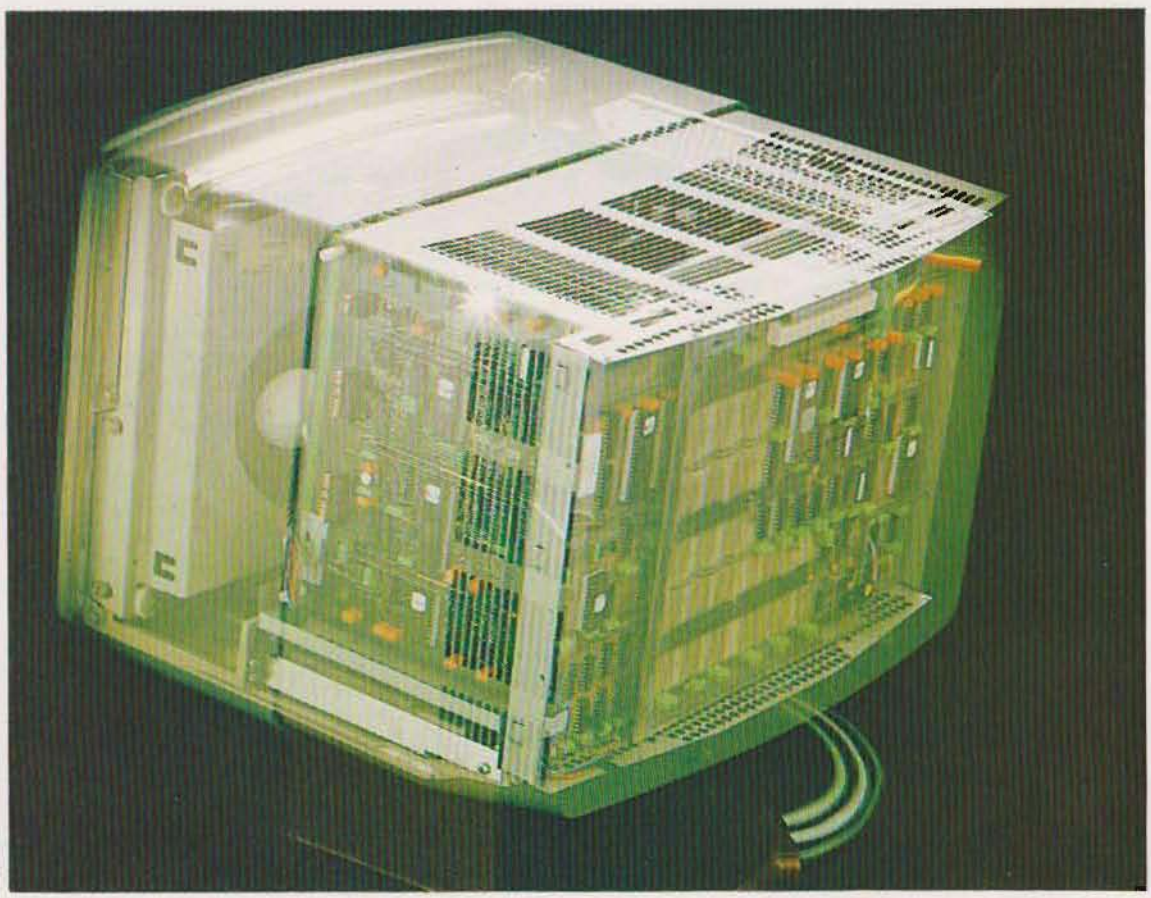
FORMATION DES COULEURS DANS UN TELEVISEUR A TROIS FAISCEAUX



faisceaux lumineux de trois écrans différents, chacun avec une émission lumineuse dans l'une des trois couleurs fondamentales. De même, on peut obtenir un mélange en excitant des particules de phosphore, dans les trois couleurs fondamentales, de dimensions très réduites et très rapprochées : ainsi les trois faisceaux lumineux produits seront perçus comme un faisceau unique.
Cette deuxième méthode est plus employée dans les CRT couleur. Trois dispositifs génèrent et contrôlent chacun un faisceau qui active un des trois types de phosphore ; en réglant le flux d'électrons sur chacun des phosphores on obtient la génération de couleurs composées. Un second type ne contient qu'un seul genre de dispositif de génération du pinceau électronique (on l'appelle couramment dans les CRT « canon électronique ») alors que l'écran conserve la structure à trois phosphores.
La sélection des couleurs s'opère en réglant la pénétration du faisceau dans les phosphores. Dans les projets prévoyant la visualisation d'images en couleur, il faut souvent tenir compte de certains effets physiologiques, qui permettent d'obtenir de bons résultats en économisant sur les couleurs tout en ne réservant qu'une zone réduite de mémoire, pour les informations nécessaires à la gestion de l'écran. Les principales précautions sont dictées par les caractéristiques suivantes (semblables à ce qui se produit pour les émissions télévisées en couleur) :

- Les zones de dimensions plus grandes, correspondant aux informations chromatiques principales, doivent utiliser les 3 couleurs fondamentales.
- Les zones plus petites n'utilisent que 2 couleurs.
- Les zones de dimensions minimes peuvent ne pas être colorées dans la mesure où,

Anatomie d'un moniteur couleur, avec circuiterie.



Ericsson

au-delà d'une certaine dimension, l'œil ne perçoit plus les couleurs mais seulement la brillance.

L'espace mémoire nécessaire dans une télévision couleur est nettement plus grand que dans un modèle monochrome. Dans ce dernier, un seul bit indique l'état d'un point de l'écran, son état physique ne prenant que deux valeurs (allumé ou éteint).
Dans les écrans couleur, pour chacun des points il faut également mémoriser la composition chromatique, c'est-à-dire ses couleurs fondamentales. Les couleurs composables ne sont que théoriquement limitées par la taille de la mémoire. Dans les systèmes de qualité moyenne, on adopte entre 4 et 8 couleurs.
Pour mémoriser 4 combinaisons de couleurs fondamentales, on a besoin de 2 bits par point (avec 2 bits on peut écrire les valeurs binaires 00, 01, 10, 11 qui correspondent aux décimales 1, 2, 3, 4) et on obtient donc un doublement de la mémoire nécessaire par rapport au cas monochrome.
Dans le système à 8 couleurs, il faut 3 bits, et la zone de mémoire occupée triple (sur un écran de 1000 × 1000, on occupe donc environ 180 Koctets de 16 bits).
Pour mettre en mémoire une quantité d'informations aussi grande, il faut utiliser diverses pages de mémoire, chacune d'une grandeur (en nombre de bits) égale au nombre de points de l'écran : le nombre de pages doit être égal au nombre de points de l'écran : le nombre de pages doit être égal au nombre de bits réservés aux combinaisons chromatiques à réaliser. Ainsi, pour une représentation en quadrichromie, on a besoin de 2 pages ; pour 8 couleurs, 3 pages. Ces pages sont considérées, par le système d'exploitation, comme des matrices dont la composition génère, pour chaque point de l'écran, le mélange de couleurs fondamentales.
Dans d'autres types de machines, la zone de mémoire destinée au graphique est déjà dimensionnée pour héberger les indicateurs de couleur et elle n'est pas divisée en pages. Cette différence de structure ne se manifeste pas pour l'utilisateur s'il emploie les instructions de haut niveau prévues dans le langage machine. Par contre, elle est à l'origine de complications dans l'adressage de la mémoire s'il travaille en Assembleur ou en DMA.

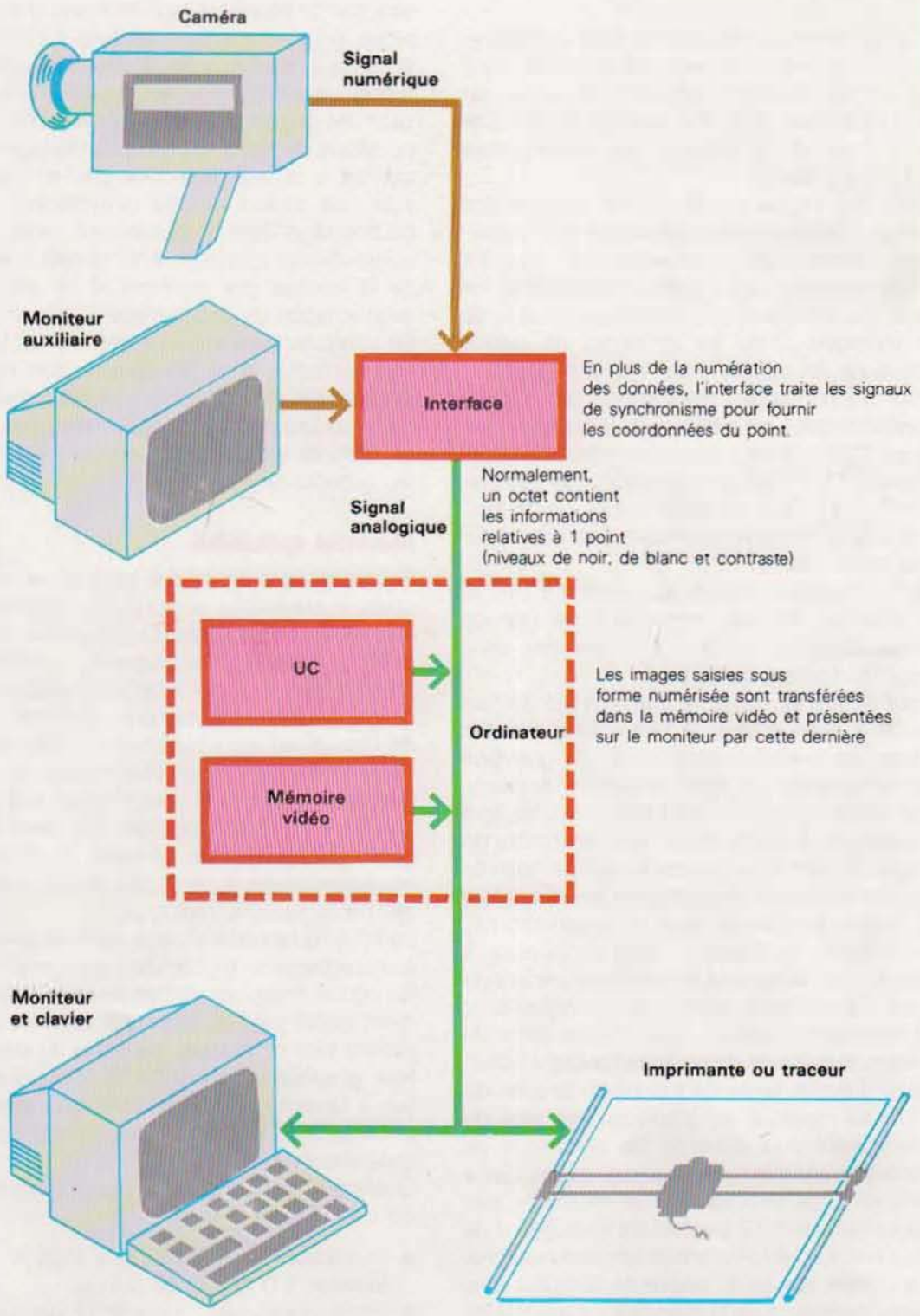
Un autre problème vient de la transposition du graphique sur le papier. Normalement, les traceurs ou les imprimantes graphiques ne disposent pas de plus de 2 ou 3 couleurs. Une technique, pour obtenir des résultats acceptables, consiste à reproduire les diverses tonalités au moyen d'impressions successives, chacune dans une couleur avec un symbole prenant plus ou moins de place selon le pourcentage de la couleur à obtenir. Par exemple, en utilisant pour une couleur donnée le symbole . , son pourcentage dans le dessin sera faible, alors qu'en utilisant un symbole qui remplit davantage la matrice (par exemple X) on aura une augmentation de pourcentage.
En intercalant, de manière très serrée, les divers symboles dans les couleurs, on obtient l'impression d'un dessin comportant davantage de couleurs que celle, réellement employées (la méthode est la même que pour les différentes nuances de noir et de blanc).

Matériel spécialisé

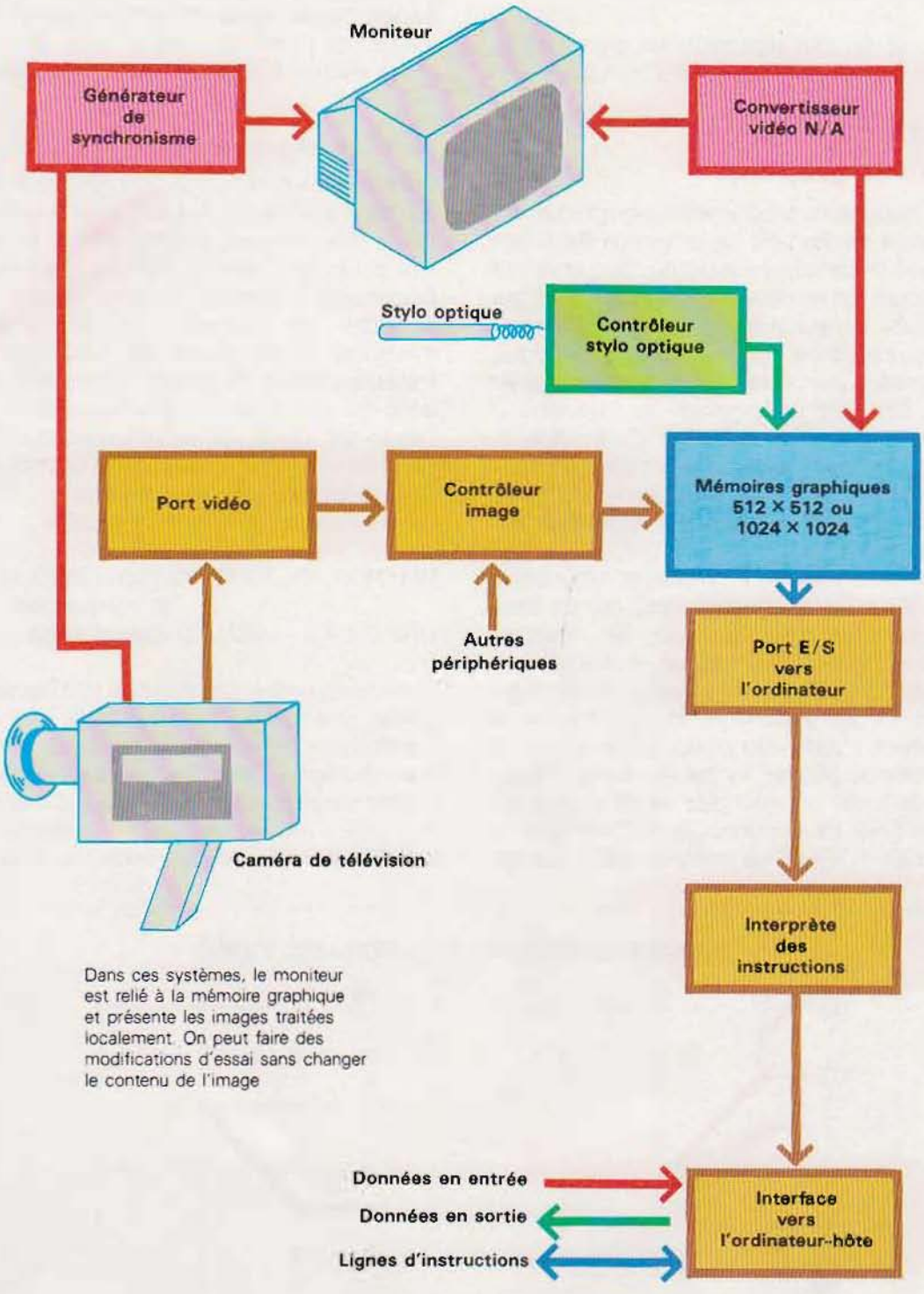
Etant donnée la variété des applications graphiques, il existe des composants matériels ou des systèmes complets dédiés. Ainsi, pour la saisie d'images relativement complexes (photos, dessins), des interfaces spéciales relient la machine à la caméra. La numérisation de l'image est alors automatique. Elle permet l'acquisition de figures très complexes dans des délais très courts. Les champs d'application de cette technique sont très diversifiés : graphisme publicitaire, analyse de données météorologiques à partir des photos transmises par le satellite, radiologie...
Le schéma représenté page suivante se réfère à une application d'ordinateur personnel. Dans les petites machines, le traitement est entièrement réalisé par l'UC alors que, dans les applications plus complexes, on utilise un **contrôleur graphique** (voir p. 1413). Celui-ci effectue, à lui seul, une grande partie du traitement et utilise l'ordinateur-hôte uniquement comme superviseur.
Quelques caractéristiques de ces systèmes sont énumérées ci-dessous :

- Résolution : de 512 × 512 à 1024 × 1024 (environ 1 000 000 de points).
- Pages graphiques : jusqu'à 12 (les valeurs usuelles sont 6/8).

ACQUISITION D'IMAGES COMPLEXES



CONTROLEUR D'ECRAN GRAPHIQUE



Dans ces systèmes, le moniteur est relié à la mémoire graphique et présente les images traitées localement. On peut faire des modifications d'essai sans changer le contenu de l'image

- Couleurs : de 64 000 à 16 000 000 (nuances quasiment illimitées).
- Vitesse de transfert des données : environ 1 000 000 pixels par seconde.

Le coût de ces équipements est, grosso modo, 20 fois plus élevé que celui d'un ordinateur.

Instruction Basic pour le graphique

Les instructions spécifiques aux graphiques dépendent étroitement de la version Basic employée. A partir du moment où, dans la version standard, on ne prévoit pas d'instructions graphiques, il n'existe aucun point de référence ; chaque machine utilise ses propres symboles. La gestion d'un écran en mode graphique est très différente de la gestion de caractères et elle comporte des difficultés d'adressage de mémoire, sans parler de la complexité découlant de la couleur. On peut néanmoins définir des méthodologies générales applicables à la majorité des cas.
Dans toute application graphique, il faut disposer d'une série d'instructions, ou de sous-programmes pour accomplir des fonctions élémentaires ; un graphique, quel qu'il soit, s'obtiendra par une combinaison de ces fonctions. La plus importante consiste à tracer un segment à partir des points extrêmes, ce qui permet de générer toutes les autres figures. Par la suite, les exemples seront étudiés sur deux types de machines : le 2010 de Siprel et le M20 d'Olivetti. La première utilise des instructions communes à l'Apple et à ses compatibles et la seconde des instructions spécifiques.
En comparant les deux formes, on mettra en évidence les analogies et les méthodes qui permettent de passer de l'une à l'autre. Pour les autres micros, il suffit de consulter le manuel.

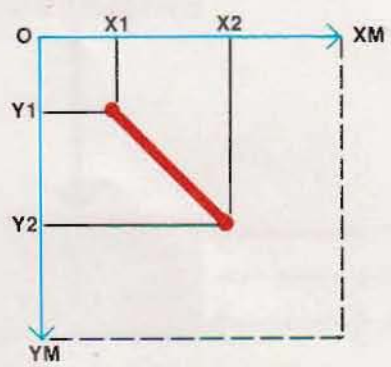
Tracé de segments

L'écran vidéo constitue le périphérique le plus commun dans les graphiques : c'est pour cette raison que la plupart des instructions se réfèrent à l'écran. Dans les autres périphériques (imprimante graphique, traceur...) et sauf cas particulier (en général sur de grosses machines) on ne trouve pas facilement de sous-programme de gestion. L'utilisateur doit donc les écrire lui-même. Chaque point de l'écran est répété par un couple de coordonnées. Pour dessiner un segment, il faut préciser les coordonnées des deux extrémités.
Les instructions de base sont du type :

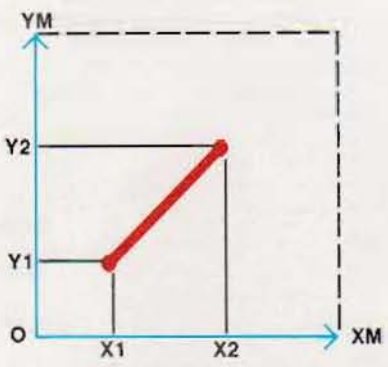
HPLOT X1, Y1, TO X2, Y2 (Siprel 2010, Apple et compatibles)
LINE (X1, Y1) — (X2, Y2) (Olivetti M20)

Comme on peut le constater, la syntaxe est la même, seul l'aspect formel change.
Les plus adaptées pour le système de coordonnées de l'écran : la différence réside essentiellement dans l'orientation de l'axe y. Il faut garder cela à l'esprit quand on mentionne les coordonnées des points concernés. Dans le

DIMENSIONNEMENT DES ECRANS VIDEO



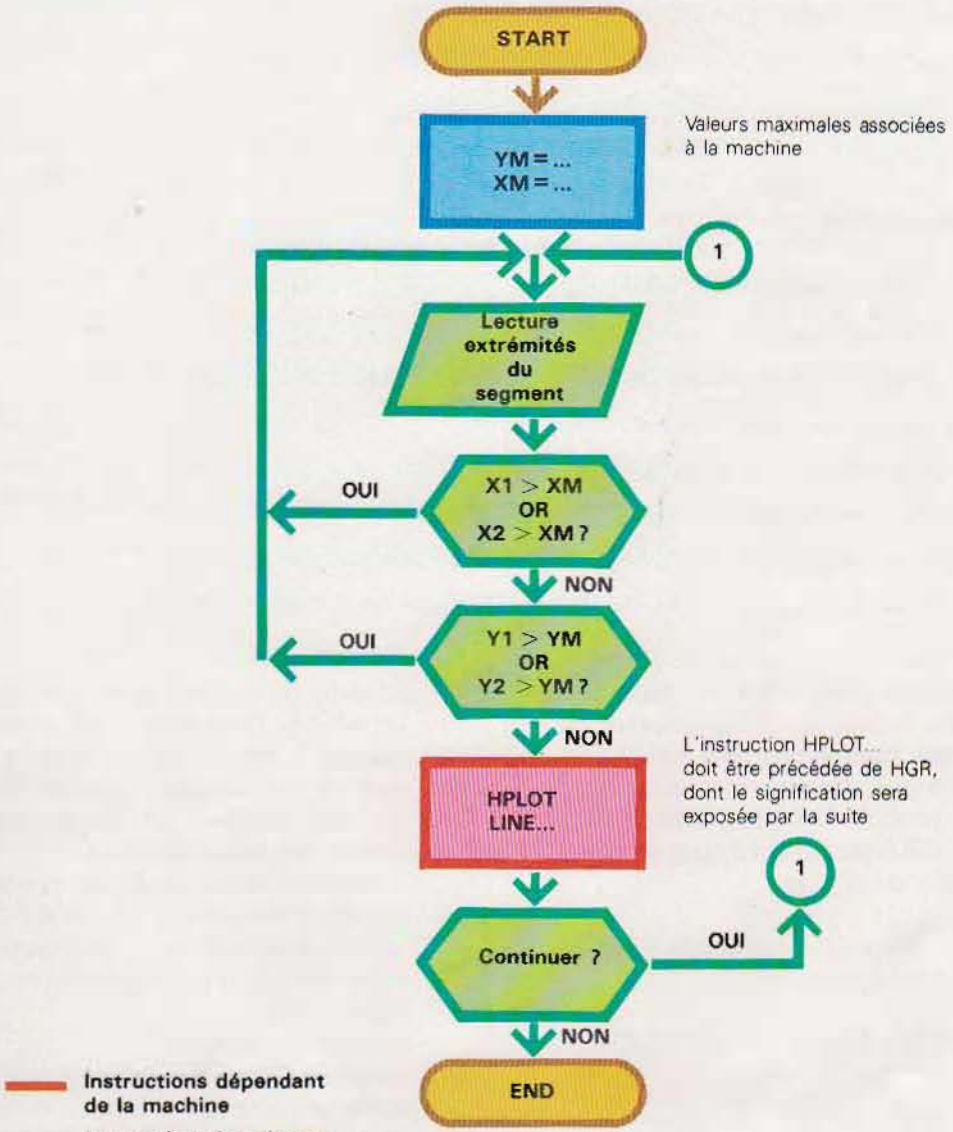
HPLOT X1,Y1 TO X2,Y2



LINE (X1,Y1) – (X2,Y2)

Siprel 2010, l'origine des axes est en haut à gauche, dans la M20, elle se trouve en bas à gauche. L'axe x a toujours la même orientation. Dans la première machine, en augmentant la valeur de y, l'extrémité initiale du segment se déplace vers le bas, dans la seconde vers le haut. La figure ci-dessous représente l'organigramme d'un programme de dessin d'un segment à partir des points définis par l'utilisateur coordonnées. La seule particularité de ces programmes réside dans le contrôle des valeurs introduites, qui a pour but de vérifier que les coordonnées des extrémités ne débordent pas de l'écran.

ORGANIGRAMME DU PROGRAMME DE CONSTRUCTION D'UN SEGMENT



TRACE D'UN SEGMENT

Version Siprel 2010

```
10 HOME
20 XM=279:YM=191
30 PRINT "coordonnées initiales: ";
40 INPUT X1, Y1
50 PRINT "coordonnées finales  : ";
60 INPUT X2, Y2
70 IF X1>XM OR X2>XM OR Y1>YM OR Y2>YM THEN GOTO 10
80 REM ****** dessin ******
90 HGR HCOLOR=3
100 HPLOT X1, Y1 TO X2, Y2
110 PRINT "Poursuivre (O/N): ";
120 INPUT A$
130 IF A$="O" THEN GOTO 10
140 END
```

Version Olivetti M20

```
10 CLS
20 XM=511:YM=255
30 PRINT "coordonnées initiales: ";
40 INPUT X1, Y1
50 PRINT "coordonnées finales  : ";
60 INPUT X2, Y2
70 IF X1>XM OR X2>XM OR Y1>YM OR Y2>YM THEN GOTO 10
80 REM ****** dessin ******
90 REM ce dessin est exécuté sur M20
100 LINE (X1, Y1) - (X2, Y2)
110 PRINT "Poursuivre (O/N): ";
120 INPUT A$
130 IF A$="O" THEN GOTO 10
140 END
```

La différence principale entre les deux programmes (voir listings p. 1392) provient de l'instruction qui trace le segment sur l'écran, (ligne 100). De plus, dans la première machine, il faut mettre en route le mode graphique avec l'instruction HGR (ligne 90) dont nous expliquerons la signification plus loin.

La forme complète de l'instruction LINE de l'Olivetti M20 est la suivante :

LINE % N STEP (X1, Y1) — STEP (X2, Y2), C, B, F, option

où :

LINE : Instruction générale pour tracer un segment entre les points de coordonnées (X1, Y1) et (X2, Y2)

%N : Valeur qui indique une des fenêtres (division de l'écran). Si on précise ce paramètre, l'instruction LINE n'opère que sur la fenêtre correspondante.

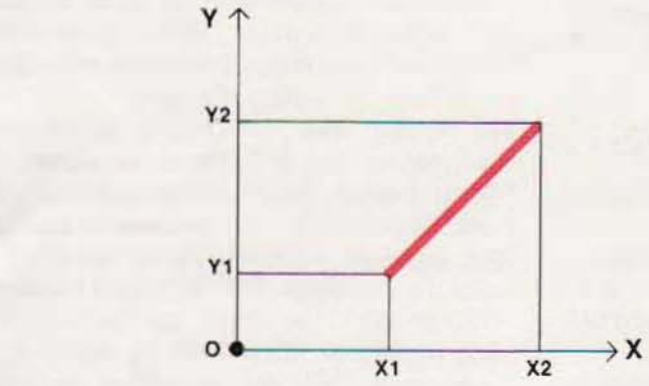
STEP : Mot clé pour indiquer l'usage de coordonnées relatives. En utilisant cette option, les coordonnées X1, Y1 deviennent relatives au dernier positionnement effectué. X2, Y2 sont les nouvelles valeurs initiales. Le schéma page suivante indique le fonctionnement de l'option STEP.

C : Couleur du segment.

B : Paramètre optionnel grâce auquel on trace un rectangle à la place du segment ayant pour extrémités les points X1, Y1, et X2, Y2 ; le rectangle a ce segment pour diagonale.

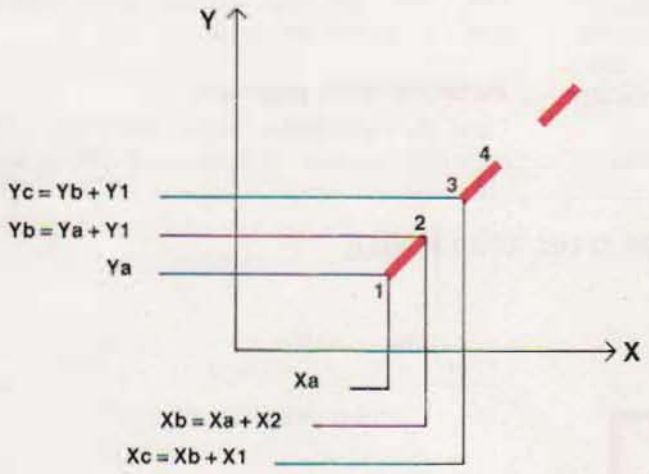
F : Paramètre qui permet de remplir, avec la couleur précisée en C, la surface du rectangle tracé avec B.

EFFET DE L'OPTION STEP



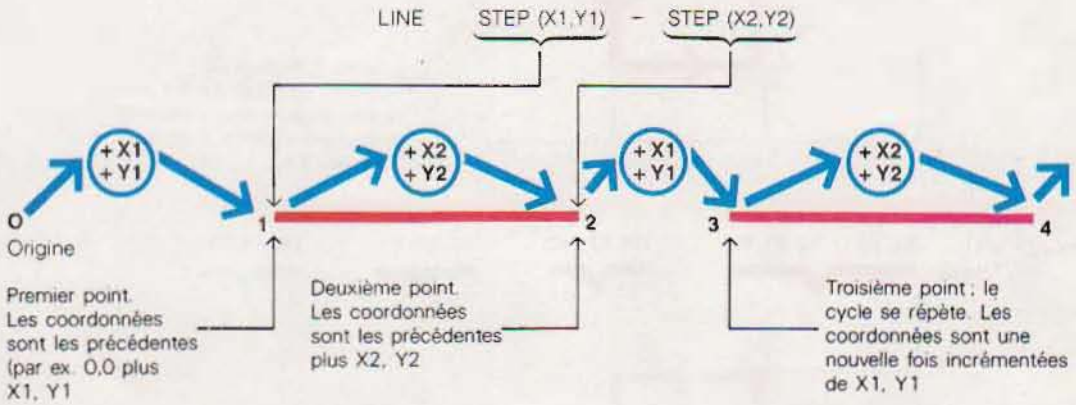
LINE (X1,Y1) – (X2,Y2)

Tracer le segment compris entre les points de coordonnées X1, Y1 et X2, Y2



LINE STEP (X1,Y1) – STEP(X2,Y2)

Les coordonnées des points sont obtenues en considérant X1, Y1 et X2, Y2 comme des valeurs relatives. Par exemple, une fois que le segment 1-2 a été tracé, la valeur X1, Y1 est ajoutée aux coordonnées finales et le segment 3-4 est tracé.



F : Paramètre permettant de remplir, avec la couleur indiquée par C, la surface du rectangle tracé d'après le paramètre B.

L'option prend une des valeurs AND, XOR, OR, NOT, PSET et PRESET et n'est utilisable qu'avec la couleur.

L'option B libère le programmeur de l'obligation d'écrire le sous-programme de tracé de rectangles comme une suite d'instructions de tracé de segments. Avec d'autres machines, on obtient le même résultat en utilisant la possibilité d'adressage multiple de l'instruction HPLOT. La formule de base :

HPLOT X1, Y1, TO X2, Y2

peut être développée pour obtenir des positionnements successifs en répétant l'option TO pour chacun des nouveaux couples de coordonnées.

Par exemple la formule :

HPLOT X1, Y1, TO X2, Y2 TO X3, Y3 TO X4, Y4

s'utilise pour le tracé d'un rectangle ayant pour sommets les points de coordonnées X1, Y1, etc. Son effet est donc semblable à celui de l'instruction précédente avec l'option B. La figure ci-dessous compare ces deux types d'instruction.

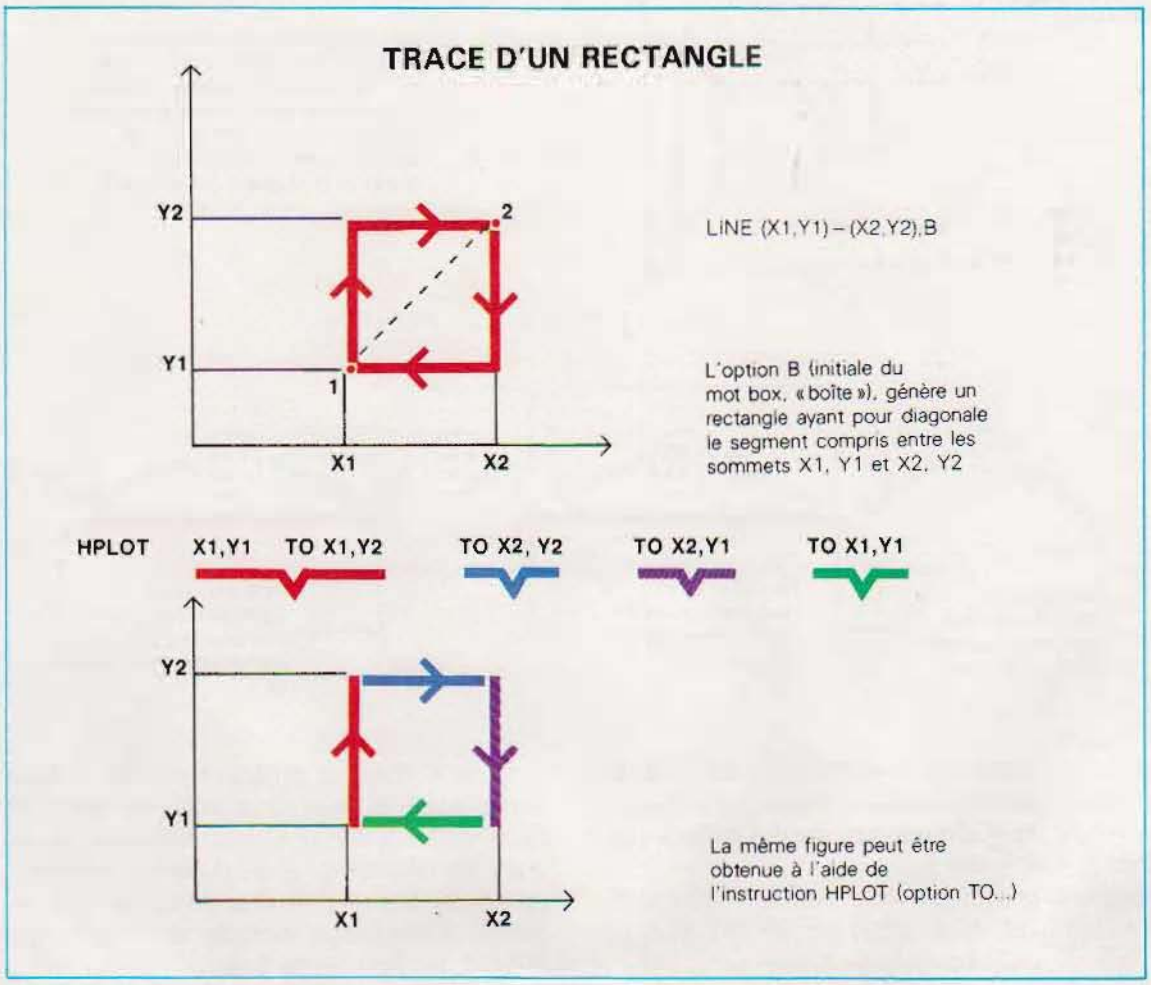
Avec l'instruction LINE, associée à l'option B, les côtés du rectangle obtenu sont toujours parallèles au bords de l'écran (ou de la fenêtre). En revanche, HPLOT n'est pas limité : toute figure générique peut être tracée, quel que soit le nombre de segments.

Les instructions de tracé des segments autorisent le dessin de toutes formes de polygones. L'organigramme ci-contre illustre un programme de dessin de carrés et de rectangles positionnés n'importe où sur l'écran ; il prévoit même la modification des dimensions de la figure en donnant un facteur d'échelle.

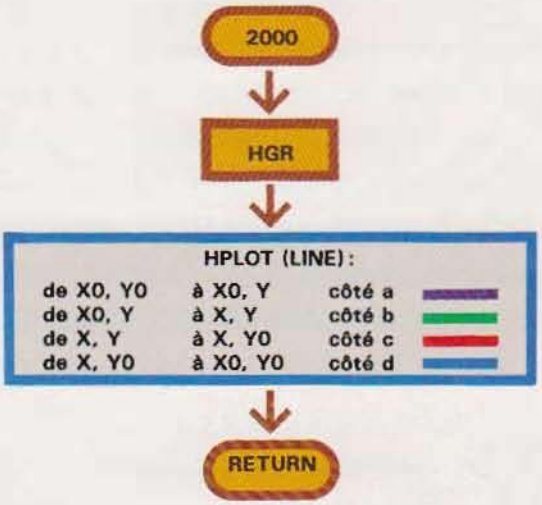
Les paramétrages sont obtenus en affectant, aux coordonnées de la figure, les valeurs X1, Y0 du sommet de positionnement et en calculant les déplacements nécessaires pour obtenir les côtés. Cette logique est appliquée au tracé d'un rectangle dans les figures des pages 1420 et 1421. Les listings pp. 1421 et 1422 sont donnés en deux versions Siprel 2010 (ordinateurs Apple et compatibles) et Olivetti M20 pour bien faire ressortir les différences entre les jeux d'instructions.

Rotation d'un segment

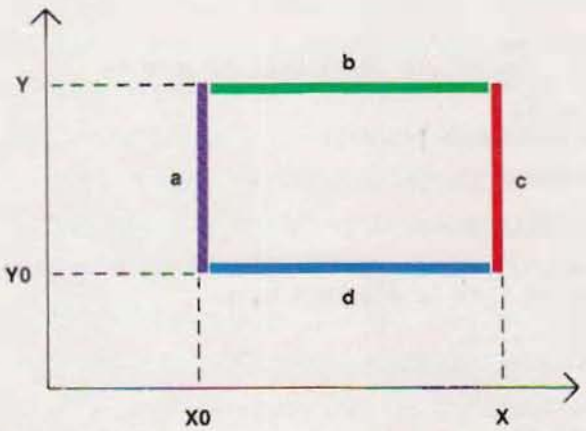
Dans les applications graphiques, il faut généralement disposer de sous-programmes assu-



TRACE DE LA FIGURE



Les coordonnées X, Y sont calculées dans le sous-programme 1000.
Les côtés de la figure étant parallèles aux axes, aucun autre point n'est nécessaire.



TRACE PARAMETRE DE CARRES ET DE RECTANGLES

Version Siprel 2010, Apple et compatibles

```
  5  MX=279 : MY=191
 10   TEXT : REM sélection de la page de texte
 15   HOME : REM l'écran est vidé
 20   PRINT  "Facteur de forme";
 30   INPUT  N
 40   PRINT  "coordonnées de référence";
 50   INPUT  X0,Y0
 60   PRINT  "Facteur d'échelle";
 70   INPUT  S
 80   GOSUB  1000
 90   IF KE=0 THEN GOTO 100
 95   PRINT "Erreur": FOR P=1 TO 2000: NEXT P: GOTO 10
100   GOSUB 2000
105   VTAB 22: REM Le curseur est positionné à la 22ème ligne
110   PRINT "Voulez-vous poursuivre ?(O/N)";
```

```
120  INPUT A$
130  IF A$="0" THEN GOTO 10
140  END
1000  REM ***** controles *****
1010  KE=0
1020  X=X0+N*S : Y=Y0+S
1030  IF Y>MY OR Y<0 OR X>MX OR X<0 THEN KE=1
1040  RETURN
2000  REM ***** Dessin *****
2010  HGR : HCOLOR= 3
2020  HPLOT X0,Y0 TO X0,Y
2030  HPLOT X0,Y TO X,Y
2040  HPLOT X,Y TO X,Y0
2050  HPLOT X,Y0 TO X0,Y0
2060  REM L'instruction suivante aurait suffi :
2070  REM HPLOT X0,Y0 TO X0,Y TO X,Y TO X,Y0 TO X0,Y0
2080  RETURN
```

Version Olivetti M20

```
5  MX=511 : MY=255
10  REM Non obligatoire
15  CLS : REM L'écran est vidé
20  PRINT  "Facteur de forme"
30  INPUT  N
40  PRINT  "Coordonnées de référence"
50  INPUT  X0,Y0
60  PRINT  "Facteur d'échelle"
70  INPUT  S
80  GOSUB  1000
90  IF KE=0 THEN GOTO 100
95  PRINT  "Erreur" : FOR P=1 TO 2000: NEXT P: GOTO 10
100  GOSUB 2000
105  REM Non obligatoire
110  PRINT "Tu veux continuer ?(O/N)";
120  INPUT A$
130  IF A$="0" THEN GOTO 10
140  END
1000  REM ***** controles *****
1010  KE=0
1020  X=X0+N*S : Y=Y0+S
1030  IF Y>MY OR Y<0 OR X>MX OR X<0 THEN KE=1
1040  RETURN
2000  REM ***** Dessin *****
2010  REM Non obligatoire
2020  LINE (X0,Y0) - (X0,Y)
2030  LINE (X0,Y) - (X,Y)
2040  LINE (X,Y) - (X,Y0)
2050  LINE (X,Y0) - (X0,Y0)
2060  REM L'instruction suivante aurait suffi:
2070  REM LINE (X0,Y0) - (X,Y),1,B
2080  RETURN
```

méthode utilisée dans les organigrammes précédents pour le changement d'échelle.Prenons l'exemple de la figure de la page 1420. Si les coordonnées X0, Y0 sont modifiées, mais non les paramètres N et S, la figure sera alors tracée à partir du point défini par le nouveau couple de coordonnées X0, Y0 (sommet 1)...
De cette façon, la figure est déplacée dans une autre zone de l'écran, mais il faut noter le tracé précédent n'est pas effacé.

Pour l'éliminer, il faut vider tout l'écran ou repasser sur cette première figure avec une « gomme ». La première solution n'est pas toujours utilisable, car d'autres figures présentes — et devant rester affichées — pourraient être affectées. La seconde solution, qui consiste à n'effacer que la figure déplacée, est bien souvent la seule acceptable.
En mode graphique, on utilise généralement plusieurs couleurs, y compris la couleur du

fond (background), indiquée par le code numérique 0. On a recours à cette couleur comme à une gomme, en la faisant repasser sur les lignes d'une figure. Le code 0, indiquant la couleur de fond noire (le dessin est tracé en blanc sur noir à l'écran), est quasiment valable pour toutes les machines, avec toutefois quelques exceptions.
On sélectionne une couleur avec :

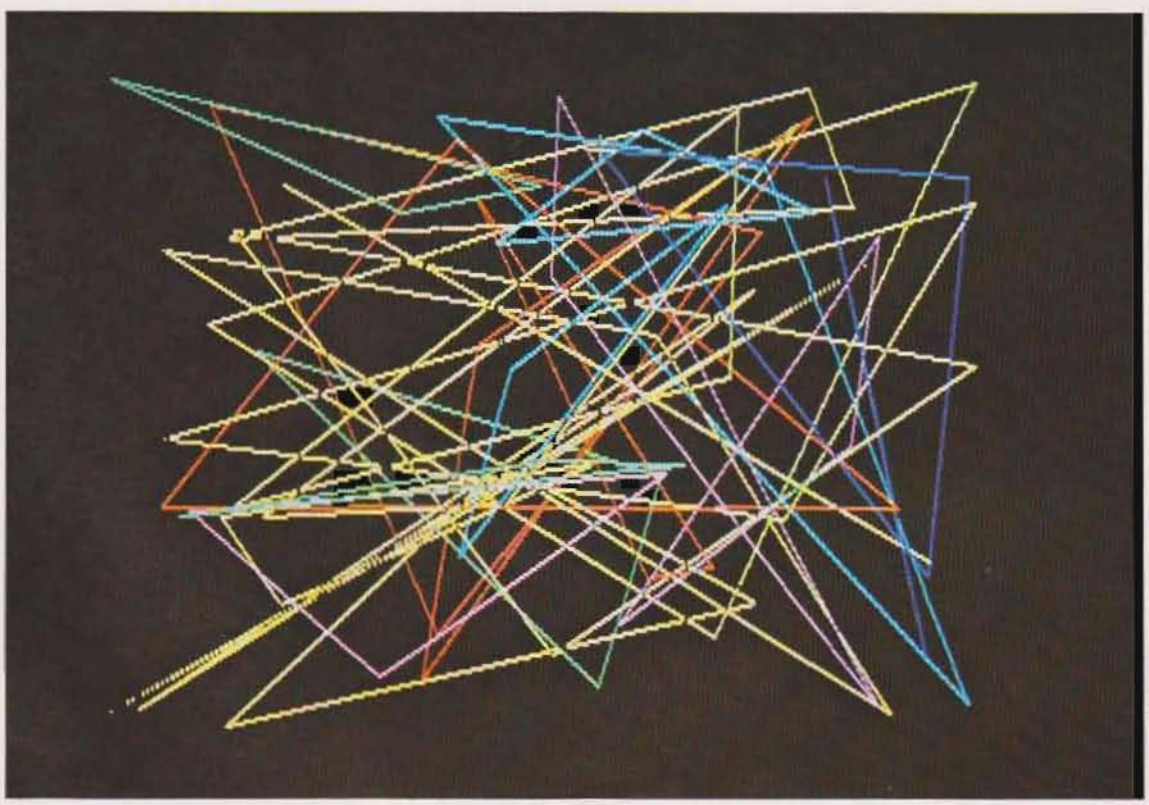
COLOR = n

n étant son code. Dans certains cas, le symbole = doit être omis.
Pour réactiver l'affichage après avoir utilisé la couleur du fond, il suffit d'indiquer la nouvelle couleur à l'aide de l'instruction COLOR.
Avec le Siprel 2010, par exemple, seul le code 7 est utilisable, l'écran étant monochrome ; ce micro-ordinateur n'offre donc que deux possibilités : COLOR = 0 (non visible) et COLOR = 7 (visible)
Page 1424, on trouvera l'organigramme d'un programme illustrant la technique de déplacement d'un rectangle.
Le listing correspondant n'est pas indiqué ; en effet il est pratiquement identique à celui du programme de tracé paramétré de carrés et de rectangles.
Contrairement à la translation, la rotation présente quelques difficultés : il faut recourir à la trigonométrie pour obtenir les paramètres (coordonnées).
L'organigramme de la page 1425 sert à effectuer la rotation d'un segment.
Le bloc 1000 regroupe les instructions d'entrée des données initiales :

— longueur du segment
— angle initial par rapport à l'axe X
— position initiale (X0, Y0) à du centre de rotation.

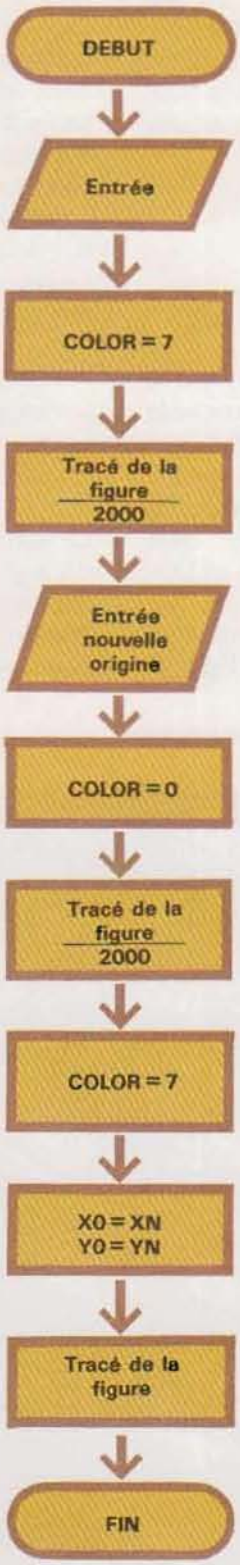
Le bloc 1500 peut ne pas provoquer l'affichage du segment dans sa position initiale et sauter directement au bloc 4000.
A la fin de la présentation, si l'opérateur désire poursuivre, le programme revient aux instruc-

Tracé de segments sur un écran vidéo couleur.



Centro THC/Marka

ORGANIGRAMME DU PROGRAMME DE DEPLACEMENT D'UNE FIGURE



Entrée :
des coordonnées initiales (XO, YO)
du facteur de forme N
du facteur d'échelle S

Sélection de la couleur

Le sous-programme 2000 est identique au sous-programme de tracé paramétré de carrés et de rectangles

Les coordonnées du nouveau point d'origine (après le déplacement) sont affectées à deux variables purement pratiques XN et YN

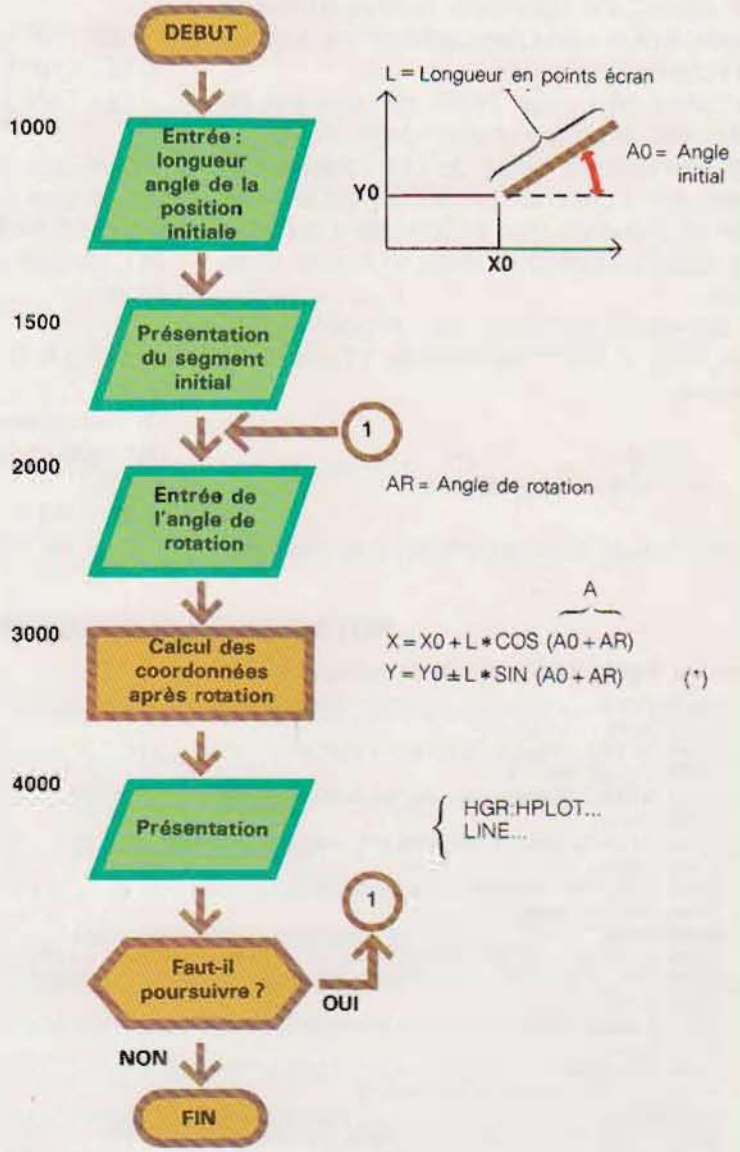
Sélection de la couleur du fond (effacement)

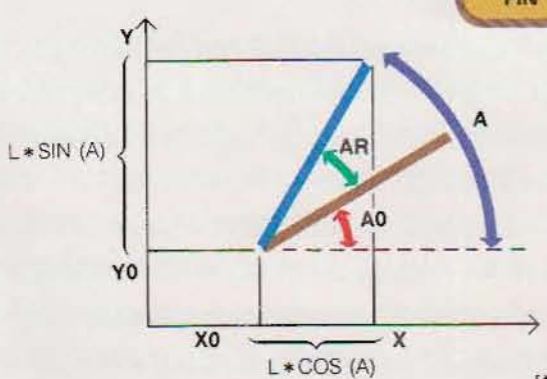
L'effacement s'obtient en repassant sur la même figure avec COLOR = 0

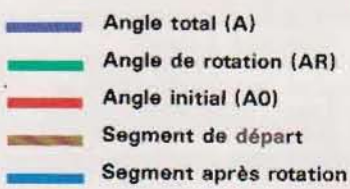
Réactivation de l'affichage

Nouvelle position initiale

ORGANIGRAMME DE ROTATION D'UN SEGMENT







[*) Le signe (+ ou –) dépend de l'orientation de l'axe y. C'est donc un paramètre caractéristique de la machine.

tions du bloc 2000 (point 1) et le cycle d'entrée de l'angle de rotation, de calcul avec de nouvelles coordonnées et d'affichage est réactivé.
Par rapport à la translation, la seule différence réside dans le calcul des coordonnées du point de rotation.
Ce calcul (voir page 1425) ne pose pas de difficultés de programmation particulières.
Les formules de calcul de ces coordonnées (indiquées à côté du bloc 3000) sont applicables telles quelles, mais il faut veiller à exprimer les angles en radians, comme en Fortran et en Basic.
La conversion en radians (voir programmes pages 1426 et 1427) est obtenue à l'aide de la formule :

$$\text{Angle en radians} = \text{Angle en degrés} * 3.14/180$$

Cette formule n'est qu'indicative et n'est pas optimale du point de vue de la programmation. Elle peut être rendue plus « professionnelle » avec les fonctions :

```
DEF FNR (AG) = AG* 3.14/180
DEF FNX (L,A) = X0 + L*COS (A)
DEF FNY (L,A) = Y0 + L*SIN (A)
```

(AG = angle en degrés, A : angle en radians, L = longueur du segment).
L'angle A (radians) s'obtient à partir de l'angle AG (degrés) au moyen de la formule toute simple :

```
A = FN R (AG)
```

De même, les nouvelles coordonnées résultent des instructions :

```
X = FN X (L,A)
Y = FN Y (L,A)
```

ROTATION D'UN SEGMENT

Version Siprel 2010, Apple et compatibles

```
1000 TEXT
1100 HOME
1200 PRINT "Position initiale";
1300 INPUT X0,Y0
1400 PRINT "Longueur du segment";
1500 INPUT L
1600 PRINT "Angle de départ (en degrés)";
1700 INPUT A0
1800 HGR: HCOLOR=7
1900 GOSUB 3000
2000 HOME
2100 VTAB22: PRINT "Angle de rotation (en degrés)";
2200 INPUT A1
2300 HCOLOR=0
2400 GOSUB 4000: REM Effacement
2500 AR=A1
2600 HCOLOR=7
2700 GOSUB 3000: REM Dessin
2750 HOME
2800 VTAB 22: PRINT "Tu veux continuer (O/N)";
2850 INPUT A$
2900 IF A$="O" THEN GOTO 2000
2950 END
3000 REM Calcul
3100 A0=(A0*3.14)/180
3200 AR=(AR*3.14)/180
3300 X=X0+L*COS(A0+AR)
3400 Y=Y0-L*SIN(A0+AR)
4000 REM Tracé du segment
4100 HPLOT X0,Y0 TO X,Y
4200 RETURN
```

Version Olivetti M20

```
1000 CLEAR
1100 REM Non obligatoire
1200 PRINT "Position initiale";
1300 INPUT X0,Y0
1400 PRINT "Longueur du segment";
1500 INPUT L
1600 PRINT "Angle de départ (en degrés)";
1700 INPUT A0
1800 CLS: COLOR 1
1900 GOSUB 3000
2000 CURSOR (1,15),1
2100 PRINT "Angle de rotation (en degrés)";
2200 INPUT A1
2210 CURSOR (1,15),1
2220 PRINT "                                        ": REM 40 espaces
2300 COLOR 0
2400 GOSUB 4000: REM Effacement
2500 AR=A1
2600 COLOR 1
2700 GOSUB 3000: REM Dessin
2750 REM Non obligatoire
2770 CURSOR (1,15),1
2800 PRINT "Tu veux continuer ?(O/N)";
2850 INPUT A$
2900 IF A$="0" THEN GOTO 2000
2950 END
3000 REM Calcul
3100 A0=(A0*3.14)/180
3200 AR=(AR*3.14)/180
3300 X=X0+L*COS(A0+AR)
3400 Y=Y0+L*SIN(A0+AR)
4000 REM Tracé du segment
4100 LINE (X0,Y0) - (X,Y)
4200 RETURN
```

Tracé d'une circonférence

Certaines machines disposent d'instructions pour le dessin de figures complexes sans passer par des sous-programmes. Nous avons déjà vu un exemple avec l'instruction LINE complétée de l'option B qui, avec l'Olivetti M20, permet d'obtenir le tracé d'un rectangle. Un cas plus complexe est celui de l'instruction de tracé d'une circonférence. Sa syntaxe et les options prévues dépendent du type de machine et de la version du Basic utilisée.
Dans le système M20, l'instruction complète est :

CIRCLE %N, (X,Y), R, C, A, V,

avec :

%N : Numéro de la fenêtre (s'il n'est pas indiqué, le système prend, par défaut, la dernière fenêtre sélectionnée)
X,Y : Coordonnées du centre
R : Rayon
C : Couleur

Le paramètre A est une valeur numérique exprimant le facteur de forme. L'emploi de l'option A permet de compenser les différentes densités des points d'écran lors du passage de l'axe X à l'axe Y. Dans certains cas, elle aide au tracé des figures circulaires allongées du type ellipse.
Dans la machine considérée, la valeur de A pour obtenir un cercle est 0,807. En l'augmentant ou en la diminuant, on obtient une déformation sur l'un ou l'autre des axes. Ce paramètre est optionnel et sa valeur par défaut est la valeur standard.
La valeur V est un autre paramètre optionnel utilisé avec la couleur ; il a la même signification que dans l'instruction LINE.
Tous les paramètres cités étant optionnels — à l'exception des coordonnées du centre et du rayon — la forme la plus simple de cette instruction est :

CIRCLE (X,Y), R

Certaines machines ont recours à deux autres paramètres optionnels pour ne tracer qu'un arc de cercle, en indiquant son angle d'ouverture.

Dans certains cas, il faut spécifier l'angle du début et l'angle de fin de l'arc (en radians). Par exemple, l'instruction :

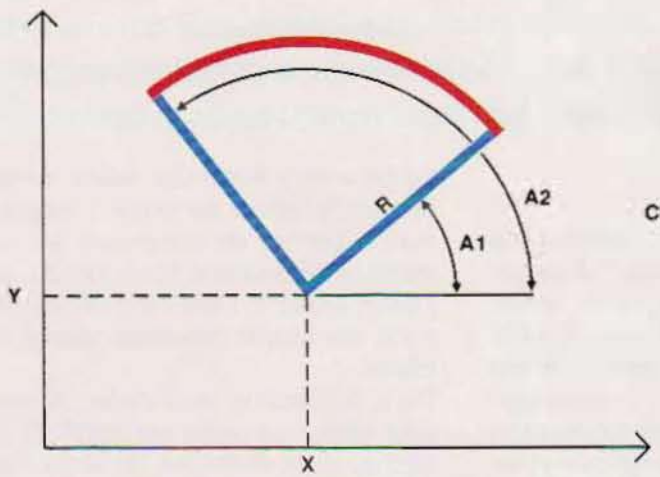
CIRCLE (X,Y), R,A1,A2

trace un arc de cercle (ayant son centre au point X,Y et de rayon R) compris entre un rayon incliné de l'angle A2 et un angle incliné de A2 (exemple ci-dessous).
Quand la machine ne reconnaît pas l'instruction CIRCLE, le système appelle un sous-programme, écrit par l'utilisateur à chaque visualisation d'un cercle.
Abstraction faite du facteur de forme, on a besoin de trois paramètres pour tracer un cercle : les coordonnées du centre (XO,YO) et le rayon (R). Le paramètrage s'obtient en calculant les coordonnées de chaque point de la circonférence en fonction des valeurs de XO, YO et R.
La figure 1 (page 1429) indique la méthode de calcul d'un point quelconque P de la circonférence, dont les coordonnées (A étant l'angle) sont données par les formules :

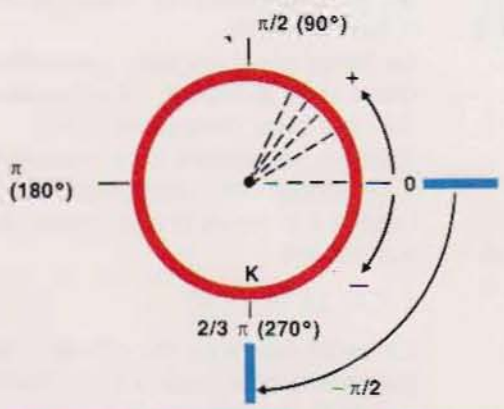
X = XO + R * COS (A)
X = YO + R * SIN (A)

Une fois données les valeurs des coordonnées du centre (XO, YO) et du rayon, on obtient, grâce aux formules précédentes et en variant l'angle A de 0 à 2 π (360 degrés), les coordonnées de tous les points du cercle.
En partant du point P1, l'angle aura pour valeur : A = 0.

INSTRUCTION CIRCLE AVEC INDICATION DES ANGLES

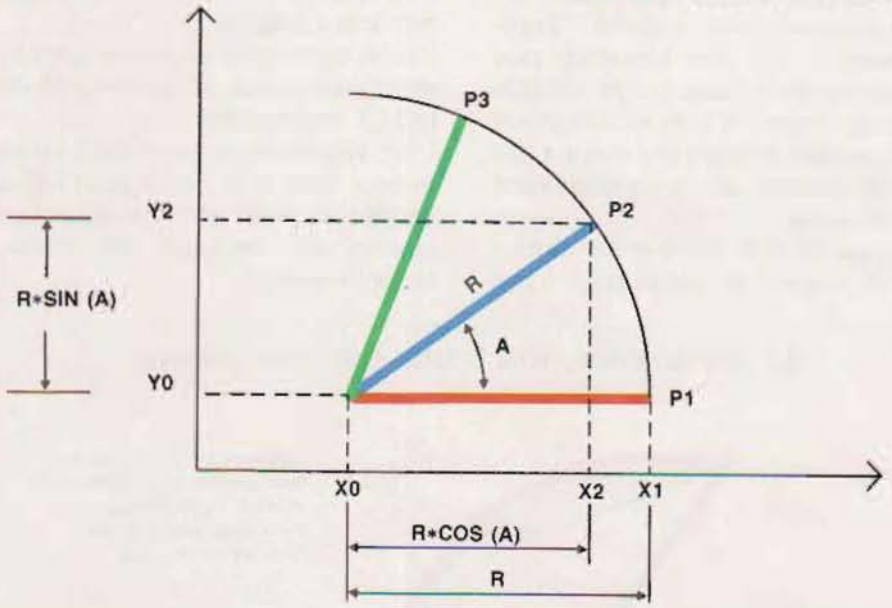


CIRCLE (X,Y), R, A1, A2

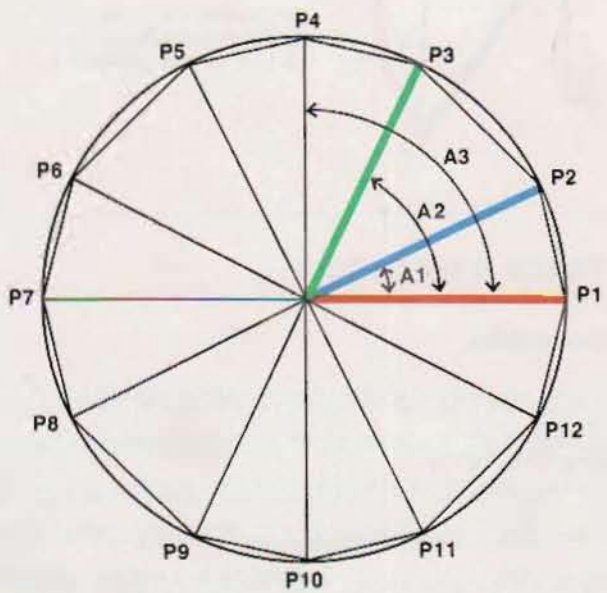


Les angles (A1 et A2) sont exprimés en radians et rapportés à la circonférence goniométrique. L'angle 0 est à l'extrême droite et les valeurs augmentent en tournant ver la gauche.
Avec certaines machines, il est possible d'utiliser des angles négatifs.
Le point K, par exemple, peut être indiqué par + 2/3 π (rotation dans le sens inverse des aiguilles d'une montre, sens positif) ou par π /2 (rotation dans le sens des aiguilles d'une montre, sens négatif)

1/ CALCUL DES COORDONNEES D'UN POINT SUR UN CERCLE



2/ TRACE D'UN CERCLE



A1 = Angle de départ

A2 = A1 + incrément

A3 = A2 + incrément

.....

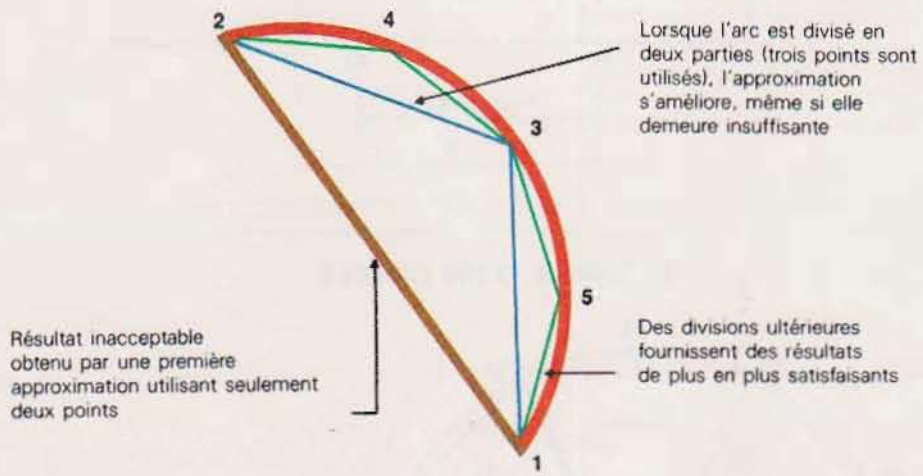
Les coordonnées [COS (0) = 1, SIN (0) = 0] auront pour valeur :

X = X0 + R
Y = Y0

On peut ensuite augmenter l'angle A d'un nombre fixe (increment), calculer les coordonnées à l'aide des formules indiquées, et ainsi de suite. En fin de calcul (figure 2, ci-dessus) le résultat sera un ensemble de points appartenant au cercle de rayon R et de centre X,Y. En les unissant deux à deux (segments), on peut obtenir une bonne approximation de la circonférence à l'aide du polygone inscrit.

La trop grande distance entre les points tracés sur la figure fait ressortir la disparité par rapport à un vrai cercle. Dans la réalité, l'increment de l'angle A doit être beaucoup plus petit ; la distance entre deux points calculés l'un après l'autre diminuera alors et le segment de droite les unissant donnera une meilleur approximation de l'arc de cercle correspondant (figure 3, ci-dessous).
Les listings (page 1430 et 1431) présentent un programme de tracé d'un cercle pour Siprel 2010 (Apple et compatible) et Olivetti M20, bien que la seconde machine dispose d'une instruction spéciale.
Il suffit de modifier deux instructions pour passer d'une version à l'autre (GHR disparaît et HPLOT devient LINE...).
L'organigramme en page 1432 est une variante pour tracé d'un cercle plein (on élimine les égalités X1 = X2 et Y1 = Y2 et le cercle est obtenu en dessinant les rayons de la circonférence).

3/ APPROXIMATION D'UNE CIRCONFERENCE



TRACE D'UN CERCLE

Version Siprel 2010, Apple et compatibles

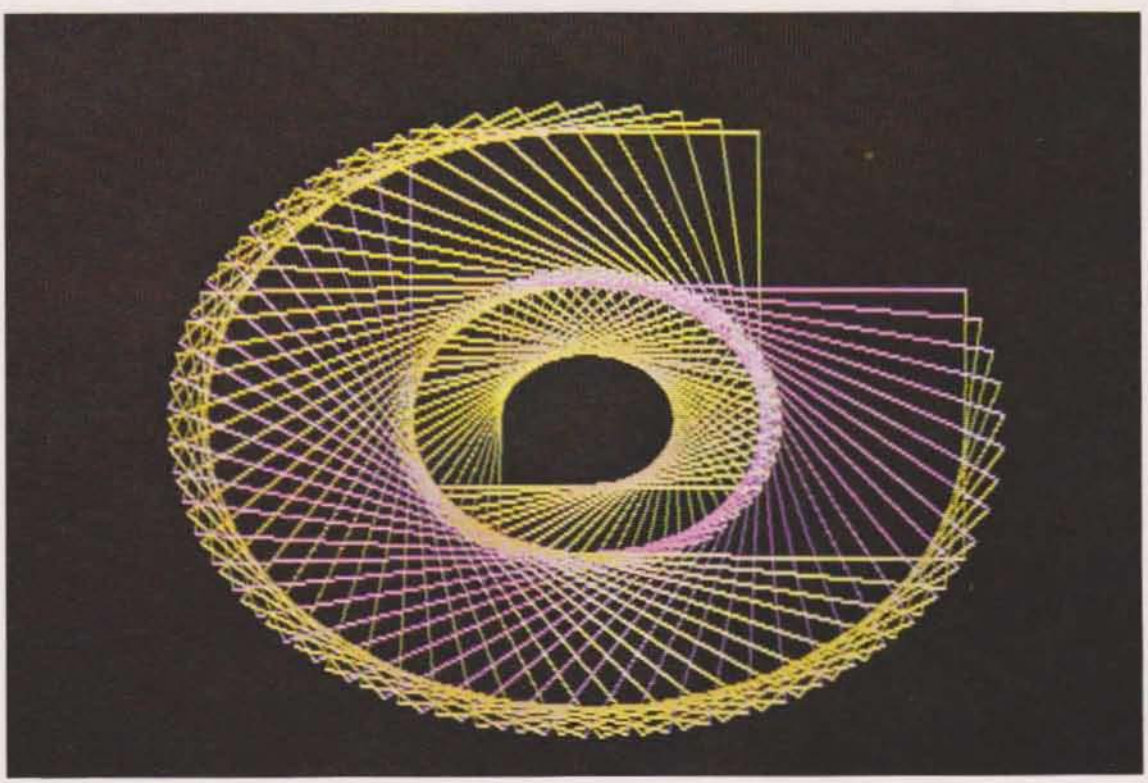
```
10 REM    Cercle
20 TEXT
30 HOME
40 PRINT "Coordonnées du centre (X0,Y0)";
50 INPUT X0,Y0
60 PRINT "Longueur du rayon";
70 INPUT R
80 PRINT "Pas d'incrément";
90 INPUT D
100 HGR: COLOR=7
110 GOSUB 1000
120 VTAB 22
130 PRINT "Continue ?(O/N)"
140 INPUT C$
150 IF C$="0" THEN GOTO 10
160 END
1000 REM Dessin
1010 X1=X0+R
1020 Y1=Y0
1030 FOR A=0 TO 6.28 STEP D
1040 X2=X0+R*COS(A)
```

```
1050 Y2=Y0+R*SIN(A)
1060 HPLOT X1,Y1 TO X2,Y2
1070 X1=X2: Y1=Y2
1080 NEXT A
1090 RETURN
```

Version Olivetti M20

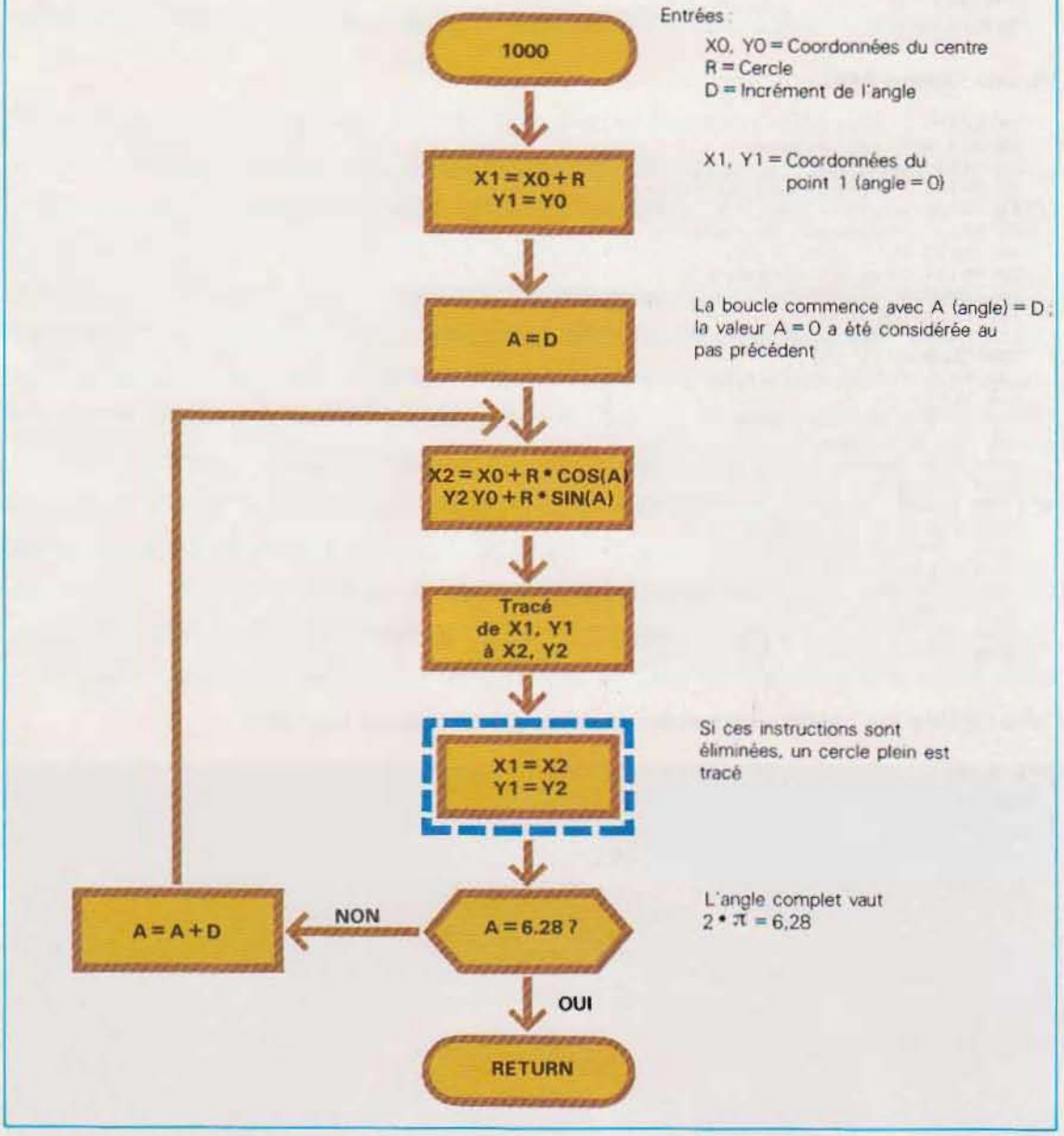
```
 10 REM      Cercle
 20 REM Non obligatoire
 30 CLS: COLOR 1
 40 PRINT "Coordonnées du centre (X0,Y0)";
 50 INPUT X0,Y0
 60 PRINT "Longueur du rayon";
 70 INPUT R
 80 PRINT "Pas d'incrément";
 90 INPUT D
100 CLS: COLOR 3
110 GOSUB 1000
120 CURSOR (1,15),1
130 PRINT "Continue (O/N)";
140 INPUT C$
150 IF C$="0" THEN GOTO 10
160 COLOR 1: END
1000 REM Dessin
1010 X1=X0+R
1020 Y1=Y0
1030 FOR A=0 TO 6.28 STEP D
1040 X2=X0+R*COS(A)
1050 Y2=Y0+R*SIN(A)
1060 LINE (X1,Y1) - (X2,Y2)
1070 XI=X2: Y1=Y2
1080 NEXT A
1090 RETURN
```

Cette figure a été obtenue en traçant un rectangle en position tournante.



Centro THC/Marka

ORGANIGRAMME DU SOUS-PROGRAMME CERCLE



Visualisation du graphique d'une fonction

Une fonction mathématique monodrome, ou uniforme, est une loi analytique qui fait correspondre, à chaque valeur de la variable indépendante X, une valeur de la variable dépendante Y.

Quand il existe plus d'une variable indépendante, la définition reste la même, mais pour plusieurs variables notées X1, X2, etc... Inversement, on parlera de fonction à plusieurs valeurs (ou polydrome) quand la valeur de la variable dépendante n'est pas unique pour des valeurs de variables indépendantes.

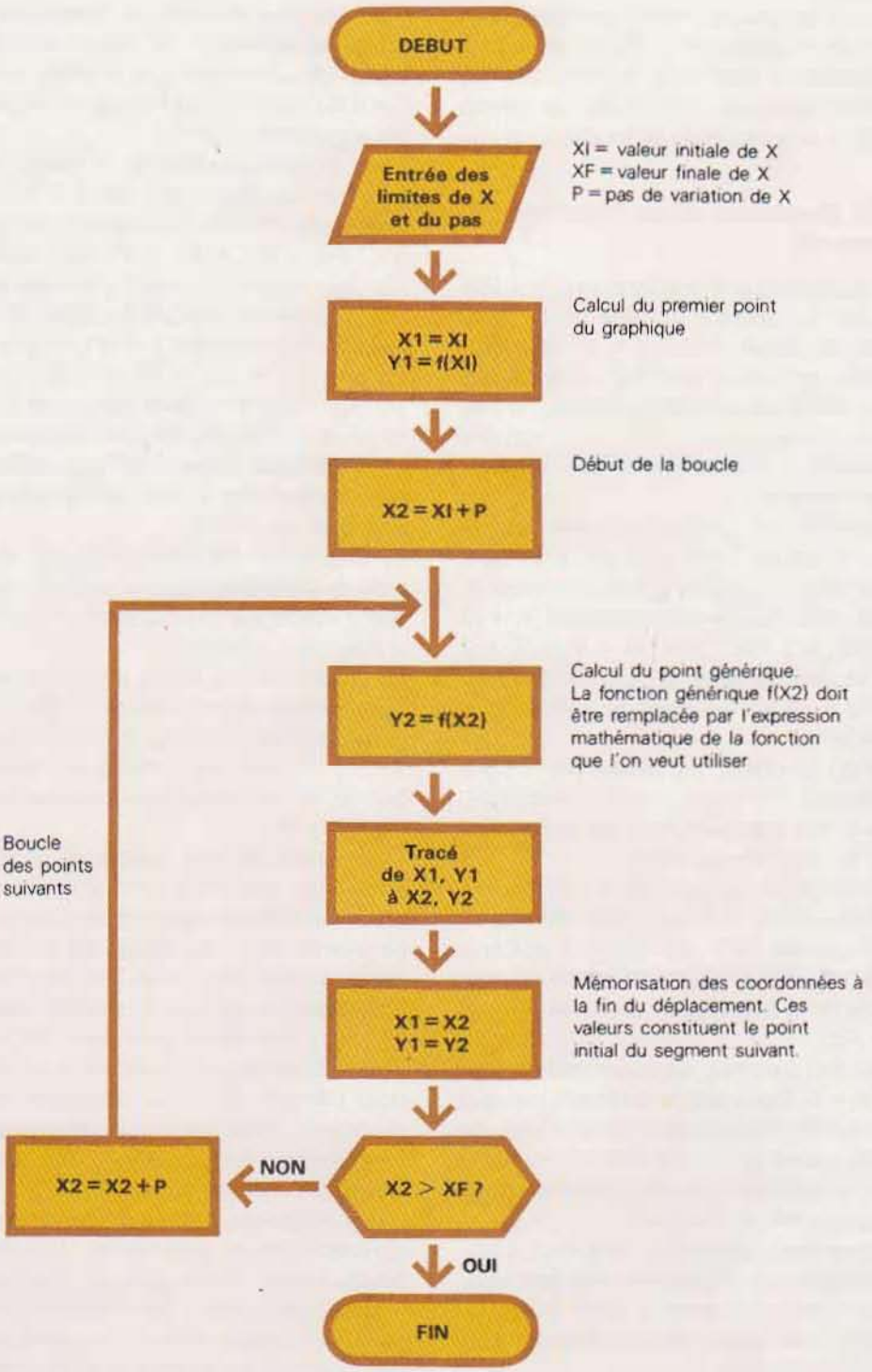
La représentation la plus courante d'une fonction est : Y = F (X) qui signifie que Y est fonction de la variable indépendante X. Etant donnée une fonction monodrome, la façon la plus simple de connaître les valeurs de la

variable dépendante consiste à construire un tableau avec les valeurs arbitraires de X (variable indépendante) et les valeurs correspondantes de Y, calculées à l'aide des calculs sur la fonction.
Toutefois, ce tableau ne fournit pas une vision immédiate de la relation X/Y. Aussi recourt-on à la représentation graphique, en reportant les valeurs du tableau sur une feuille. Le dessin obtenu est le graphique de la fonction.

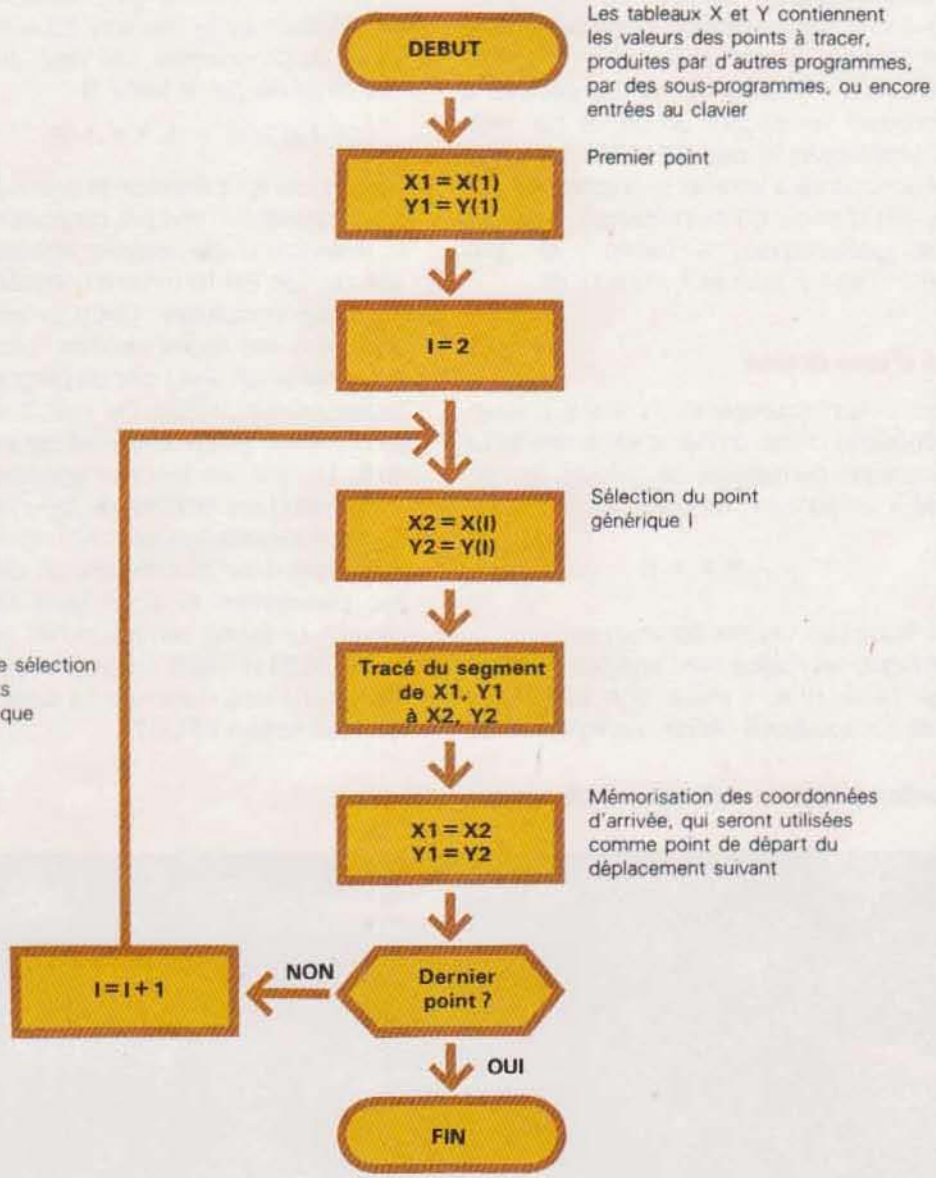
Tracé du graphique d'une fonction par segments

Le tracé du graphique d'une fonction peut être effectué par l'ordinateur à l'aide de la seule instruction graphique de tracé d'un segment.
La méthode consiste à produire un ensemble de valeurs de la variable indépendante, à calculer pour chacune d'elles celle de la variable dépendante et à unir les points ainsi obtenus (voir page suivante).
Le premier bloc est l'entrée au clavier des valeurs initiale (XI) et finale (XF) de la variable indépendante ainsi que du pas de variation P.
Le premier point aura les coordonnées X = XI et Y = f(XI), le suivant X = XI + P et Y + f (XI + P), et ainsi de suite, en augmentant toujours X de P et en calculant la valeur de Y correspondante.
Le dessin du graphique est donné par la ligne brisée unissant les points ainsi déterminés, c'est-à-dire une approximation de la fonction Y ≐ F(X) à l'aide de segments.
La « qualité » de la représentation est liée à la proximité des points avec un pas P très réduit.
La valeur optimale de P est difficile à établir a priori : elle dépend en effet, du type de courbe, c'est-à-dire de la forme particulière de la fonction Y = F(X).
Soit la fonction suivante, décrivant une droite : Y = 3 * X + 5. Deux points suffisent, puisque par deux points il ne passe qu'une droite. En unissant les coordonnées des deux points de la droite par un segment, on est certain de représenter exactement la fonction.
Si la fonction avait une forme complexe, il faudrait la découper en petites portions dont chacune pourra être approchée à l'aide d'un segment, selon une logique proche de celle d'un tracé de circonférence.
Les coordonnées X, Y de chaque point d'une circonférence sont liées par une relation du type $X^2 + Y^2 = R^2$ laquelle peut être ramenée à $Y = R^2 - X^2$, qui est une forme Y = F(X).
Pour déterminer les valeurs de X et de Y, nous avons employé les formules trigonométriques, mais ces deux aspects sont parfaitement identiques et découlent de deux méthodes différentes de résolution d'un triangle rectangle (à l'aide du théorème de Pythagore ou des formules trigonométriques).
Dans l'organigramme de la page 1434, les coordonnées du premier point (X1, Y1) et celles du second (X2, Y2), obtenu en ajoutant P sur l'axe X (X2 = X1 + P), sont calculées.
Au pas suivant, le point d'arrivée précédent (X2, Y2) devient le point de départ (il faut donc modifier les valeurs X1 et Y1 à l'aide de l'instruction X1 = X2 et Y1 = Y2) ; la nouvelle position finale (nouvelles valeurs de X2, Y2) est toujours obtenue en ajoutant la quantité P à X1 et en calculant (à partir de l'expression algébrique représentant la fonction) la valeur correspondante de Y (Y2).
Ce processus doit être répété sur tout l'intervalle de tracé graphique, c'est-à-dire jusqu'à ce que X2 atteigne (ou dépasse) XF (valeur finale entrée par l'utilisateur).
Ce processus est activé à l'aide d'une boucle dans laquelle X varie du pas P de XI à XF.
Pour simplifier le listing, le premier point X1 = XI, Y1 = f(XI) est calculé en dehors de la boucle, qui a comme premier argument la valeur XI + P.
Cette méthode n'est valable qu'en théorie. En effet, pour obtenir un bon programme de tracés, il faut tenir compte des différents facteurs.
Le premier est la **typologie de la fonction**. A priori, il n'est pas dit qu'une fonction puisse être exprimée par une loi mathématique. Dans ce cas, il faut établir une table des valeurs de X et Y. Ces valeurs seront alors stockées dans deux tableaux (de dimension égale au nombre de points maximum) et le programme devra simplement les prélever et les utiliser (organigramme page 1435).
Cette méthode n'est pas fondamentalement différente de la précédente. Simplement, la boucle ne se répète plus en incrémentant X. En effet, les valeurs du tableau ne varient pas forcément régulièrement (il pourrait, par exemple, contenir les valeurs 2 et 5 mais non les valeurs 3 et 4).

GRAPHIQUE D'UNE FONCTION ; ORGANIGRAMME DE PRINCIPE



TRACE DE COURBES A PARTIR DES VALEURS D'UN TABLEAU



C'est l'indice I, désignant les éléments du tableau, qui augmente. De cette façon, si I = 2, les coordonnées du 2e élément sont extraites, alors que quand I = 3, ce sont les coordonnées du 3e élément qui le sont, et ainsi de suite.
Autre différence par rapport à l'organigramme précédent : l'absence du calcul Y2 = f(X2), remplacé par un simple transfert de variable, X2 = X(I), Y2 = Y(I), X(I) et Y(I) étant les éléments du tableau se trouvant à la position I.

Il faut également prendre en considération d'autres facteurs comme les ordres de grandeur et les erreurs d'interpolation.
Pour obtenir, dans tous les cas possibles, des valeurs représentables à l'écran (ou sur d'autres terminaux), il faut tracer non pas les valeurs réelles, mais des valeurs qui leur sont proportionnelles selon un facteur d'échelle.
Tous les points ne sont pas exploitables dans le tracé d'un graphique à partir d'une fonction.

On suppose généralement, qu'entre deux points, le tracé est rectiligne. S'ils sont situés à une certaine distance, cette interpolation peut aboutir à une représentation inexacte et sans rapport avec la forme réelle de la fonction. L'approximation résultante est comparable à celle obtenue en traçant un cercle par segments (mais dans le cas du cercle, la forme exacte est connue a priori et le graphique n'engendre pas d'erreur d'interprétation). Dans les courbes quelconques, la forme n'est pas connue et l'erreur peut être importante.

Tracé d'une droite

La fonction la plus simple est la droite. L'équation générique d'une droite, c'est-à-dire la loi mathématique permettant de calculer les valeurs de Y en fonction de celles de X, est :

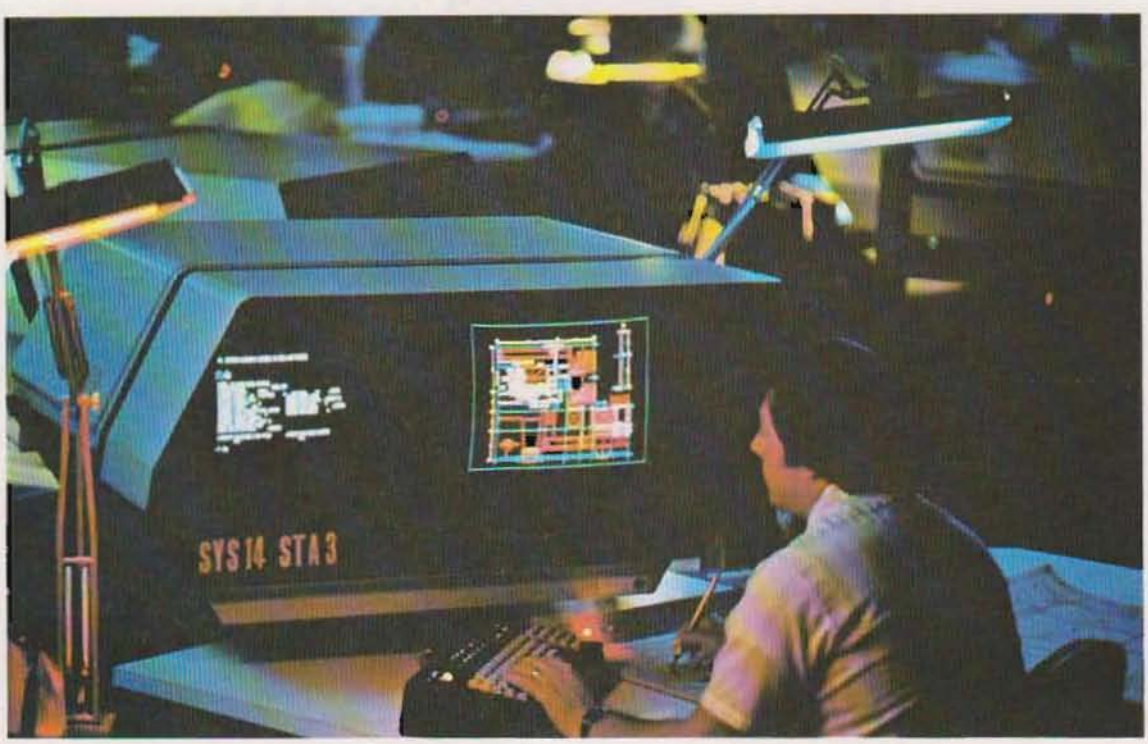
$$Y = A * X + B$$

A et B étant des valeurs constantes.
Le graphique est réalisé avec le programme de la page 1434, mais il existe une forme plus élégante qui consiste à définir une fonction utilisateur équivalente à la droite. Le calcul des valeurs de Y se fera alors simplement par appel de la fonction pour les différentes valeurs de X. La définition de la fonction est à introduire en début de programme. Elle peut, par exemple, être indiquée par la lettre R :

```
DEF FN R(X) = A * X + B
```

Dans le cas spécifique de la droite, la simplicité de la formule ne rend pas particulièrement utile la définition d'une fonction utilisateur. En revanche, elle est fortement conseillée pour des fonctions complexes. Cette procédure offre, alors, des avantages certains, car la fonction est écrite en un seul point du programme et est facilement modifiable. De plus, tous les sous-programmes graphiques sont paramétrés puisqu'ils traitent une fonction générique FN R(X) qui les rend exploitables de façon généralisée. L'organigramme (pages 1437 et 1438) trace une droite dans un intervalle de valeurs XI, XF (les paramètres A et B sont introduits au clavier). Le listing correspondant se trouve en pages 1438 et 1439. Le programme est portable sur d'autres machines : il suffit de remplacer l'instruction HPLOT.

Application professionnelle d'un ordinateur graphique.



J. Zaner/Marka-Int'l Stock Photo

```
      < 5 OR YB > 153 OR YB < 5
        THEN 180
170   HPLOT XA,YA TO XB,YB
180   X1 = X2
    : Y1 = Y2
190   NEXT
200   HOME
210   VTAB 23
220   INPUT "Je continue ? (O/N)"
230   IF S$ = "O" THEN RUN
240   END
997   REM  ---------------
998   REM  Tracé des axes
999   REM  ---------------
1000  HGR
    : HCOLOR= 3
1010  HPLOT O - 15,N - 5 TO E + 14,N
      - 5 TO E + 14,5 + 3 TO O - 15,5
      + 3 TO O - 15,N - 5
1020  HPLOT O,Y0 TO E,Y0
    : HPLOT X0,S TO X0,N
1030  N2 = N
    : FOR N1 = 2 TO 0 STEP - 1
    : HPLOT X0 - N1,N2 TO X0 + N1,N2
    : N2 = N2 - 1
    : NEXT N1
1040  N2 = E
    : FOR N1 = 2 TO 0 STEP - 1
    : HPLOT N2,Y0 - N1 TO N2,Y0 + N1
    : N2 = N2 + 1
    : NEXT N1
1050  HPLOT E + 6,Y0 - 2 TO E + 11,Y0
        + 3
    : HPLOT E + 11,Y0 - 2 TO E + 6,Y0
        + 3
1060  HPLOT X0 - 10,N - 2 TO X0 - 7,N
        + 1
    : HPLOT X0 - 4,N - 2 TO X0 - 10,N
        +3
1070  IF FX = 1 THEN 1140
1080  FOR G = X0 TO O STEP - FX
1090  HPLOT G,Y0 + 1 TO G,Y0 - 1
1100  NEXT G
1110  FOR G = X0 TO E STEP FX
1120  HPLOT G,Y0 + 1 TO G,Y0 - 1
1130  NEXT G
1140  IF FY = 1 THEN 1210
1150  FOR G = Y0 TO N STEP - FY
1160  HPLOT X0 + 1,G TO X0 -1,G
1170  NEXT G
1180  FOR G = Y0 TO S STEP FY
1190  HPLOT X0 + 1,G TO X0 - 1,G
1200  NEXT G
1210  RETURN
```

A la différence des précédents organigrammes, celui-ci contient deux paramètres d'échelle FX et FY et calcule le pas P.
Les facteurs d'échelle (FX, FY) sont nécessaires pour ramener les valeurs de X et Y aux dimensions de l'écran.
Prenons un intervalle sur l'axe de 500, par exemple, XI = 0, XF = 500), avec un écran de 265 points horizontaux, il faut alors utiliser un facteur d'échelle de 0,5.
La valeur 500 est traitée comme s'il s'agissait de 500 × 0,5 = 250 qui tient dans les 265 points disponibles. Le facteur d'échelle sur l'axe X se calcule à l'aide de l'expression :

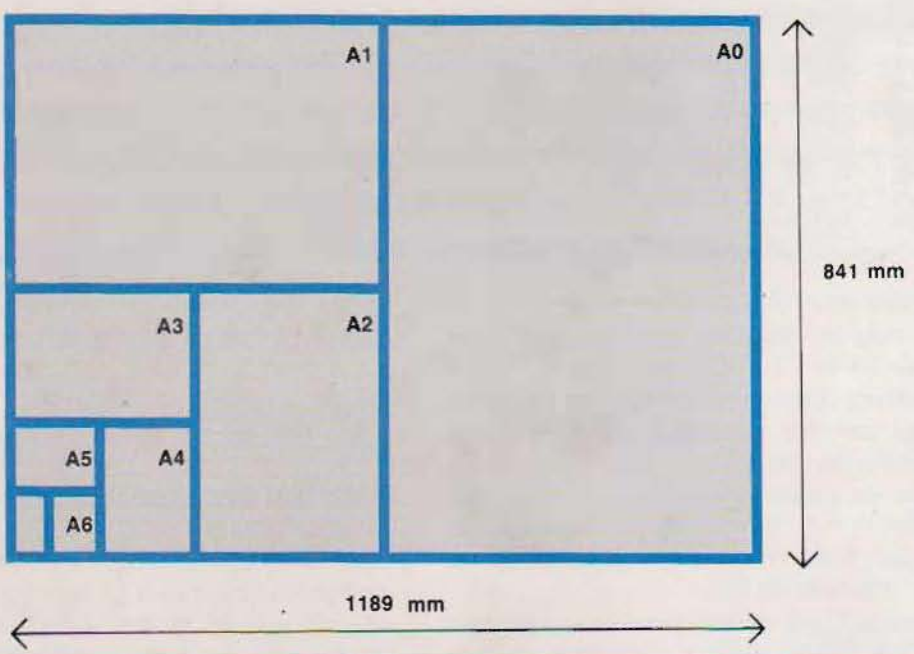
$$FX = \frac{XF - XI}{\text{Nombre des points d'écran disponibles}}$$

Remarquer que l'expression analogue est valable pour l'axe Y.
Il faut noter que la valeur du quotient est le plus souvent un nombre à plusieurs décimales, ce qui fait que le facteur d'échelle calculé ne permet pas une lecture rapide du graphique.

Il convient donc d'utiliser la fraction entière la plus proche : la réduction de la portion d'écran utilisée permettra une interprétation du graphique plus aisée.
Dans l'exemple précédent, le facteur d'échelle, calculé au moyen de l'expression donnée, serait 265/500 = 0,53 ; pour pouvoir interpréter le graphique, il faudrait multiplier les valeurs par 1,886 (1/0,53) à chaque lecture.
En arrondissant à 2, on obtient une réduction de l'image utile (les points utilisés ne sont plus 265, mais 250), qui a l'avantage de permettre d'effectuer mentalement les calculs d'échelle et donc d'avoir une lecture immédiate du graphique.
Le choix d'une échelle de représentation n'est pas, comme on pourrait le croire, arbitraire. Il existe des normes très précises réglementant les applications techniques.
Dans l'utilisation d'un traceur ou d'une tablette graphique, par exemple, il faut que le plan de travail soit suffisamment grand pour recevoir des feuilles au format commercial. Les formats standard définis par l'UNI sont désignés par un sigle composé de la lettre A suivie d'un chiffre (par ex. A0, A1, A2...). Le format le plus utilisé (dimensions standard d'une feuille de papier pour dactylographie) est le format A4. Les différents formats s'obtiennent en divisant successivement en deux le format de base A0, qui est défini par une surface de 1m^2.
Les dimensions correspondant aux différents formats sont indiquées dans le tableau ci-dessous.

FORMATS UNINORMALISES

Désignation	Dimensions en mm
A0	841 × 1 189
A1	594 × 841
A2	420 × 594
A3	297 × 420
A4	210 × 297
A5	148 × 210
A6	105 × 148



La partie initiale du programme (lignes 10 à 40) définit les paramètres de l'écran. Les valeurs N = 5, S = 153, E = 265, O = 15 délimitent la zone utile (en points écran), alors que les valeurs X0 = 140 et Y0 = 80 (Siprel 2010) présentent l'origine de référence des coordonnées. La nomenclature (N, S, E, O) renvoie aux points cardinaux et repère la « fenêtre » employée pour la représentation graphique. Les instructions 140, 150 et 160 calculent les valeurs XA, YA et XB, YB en points écran.
Comparativement à l'organigramme, les formules diffèrent de X0, Y0 : en effet, dans le cas général (organigramme), les coordonnées de chaque point de la courbe se rapportent à l'origine 0 de coordonnés X0 = 0 et Y0 = 0. Dans le listing, en revanche, on a fixé l'origine à 140, 80 pour obtenir une représentation centrée sur l'écran.
Notons que la coordonnée Y est la différence entre l'origine et la valeur résultant de l'équation de la droite. Cela est dû à l'orientation de l'axe Y qui commence en haut à gauche. Sur d'autres machines (le M20 d'Olivetti) on peut avoir besoin du signe +.
Le pas de visualisation est calculé à la ligne 110 alors que la boucle s'effectue entre la ligne 120 et 190. La ligne 160 contrôle que les valeurs des coordonnées (en points écran) ne dépassent pas les valeurs maximales (ligne 20). Pour abréger, en fin de boucle (ligne 190) on a omis l'indice (NEXT au lieu de NEXT 1).
De même, pour relancer le programme, on a adopté une syntaxe particulière qui n'est pas autorisée sur toutes les machines : il est alors nécessaire de compléter la ligne 190 et d'écrire la 230 sous la forme :

```
IF S$ = "S" THEN GOTO 50
```

Pour « porter » le programme sur d'autres machines, il faut en outre remplacer les instructions suivantes :

HOME : Effacement de l'écran et positionnement du curseur en haut à gauche (instruction 200).
VTABn : Tabulation verticale avec arrêt sur la ligne n (instruction 210).
HPLOT : Tracé d'un segment entre les deux points dont les coordonnées sont contenues dans l'instruction (instruction 1010).
HGR : Mise en route du mode graphique (instruction 1000).

Le programme présenté n'est pas le plus rationnel pour le tracé d'une droite. Le développement par points est, en général, inutile ; donc l'ensemble des instructions comprises entre 100 et 190 peuvent être remplacées par une seule : le tracé d'un segment entre les points de coordonnées XI, YI et XF, YF.
Dans cet exemple, on a adopté une boucle pour élargir le programme. On obtient ainsi le graphique de n'importe quelle fonction différente de la droite.

Visualisation des axes cartésiens

La présentation graphique d'une fonction suppose l'affichage de ses axes cartésiens.
Les paramètres à fournir en entrée, indiqués dans l'organigramme, ont la même signification que prédédemment. Le sous-programme trace les axes comme deux segments perpendiculaires se croisant au point X0, Y0 pris comme origine (voir page 1444). Il dessine les flèches indiquant l'orientation des axes aux extrémités et visualise enfin des divisions sur chacun des axes.
Le symbole « flèche » s'obtient en traçant des segments parallèles de longueur décroissante (instructions 1030-1040 du listing) comme indiqué dans la figure de la page 1 444. On pourrait aussi recourir à un triangle, mais le résultat serait moins bon esthétiquement. Quoi qu'il en soit, cette deuxième solution est également présentée.
En appelant N la valeur positive maximale de la coordonnée Y, les instructions se réduisent au tracé de 3 segments :

	de :	à :
Segment 1	X0 - 1, N	X0 + 1, N
Segment 2	X0 + 1, N	X0, N - 3
Segment 3	X0, N - 3	X0 -1, N

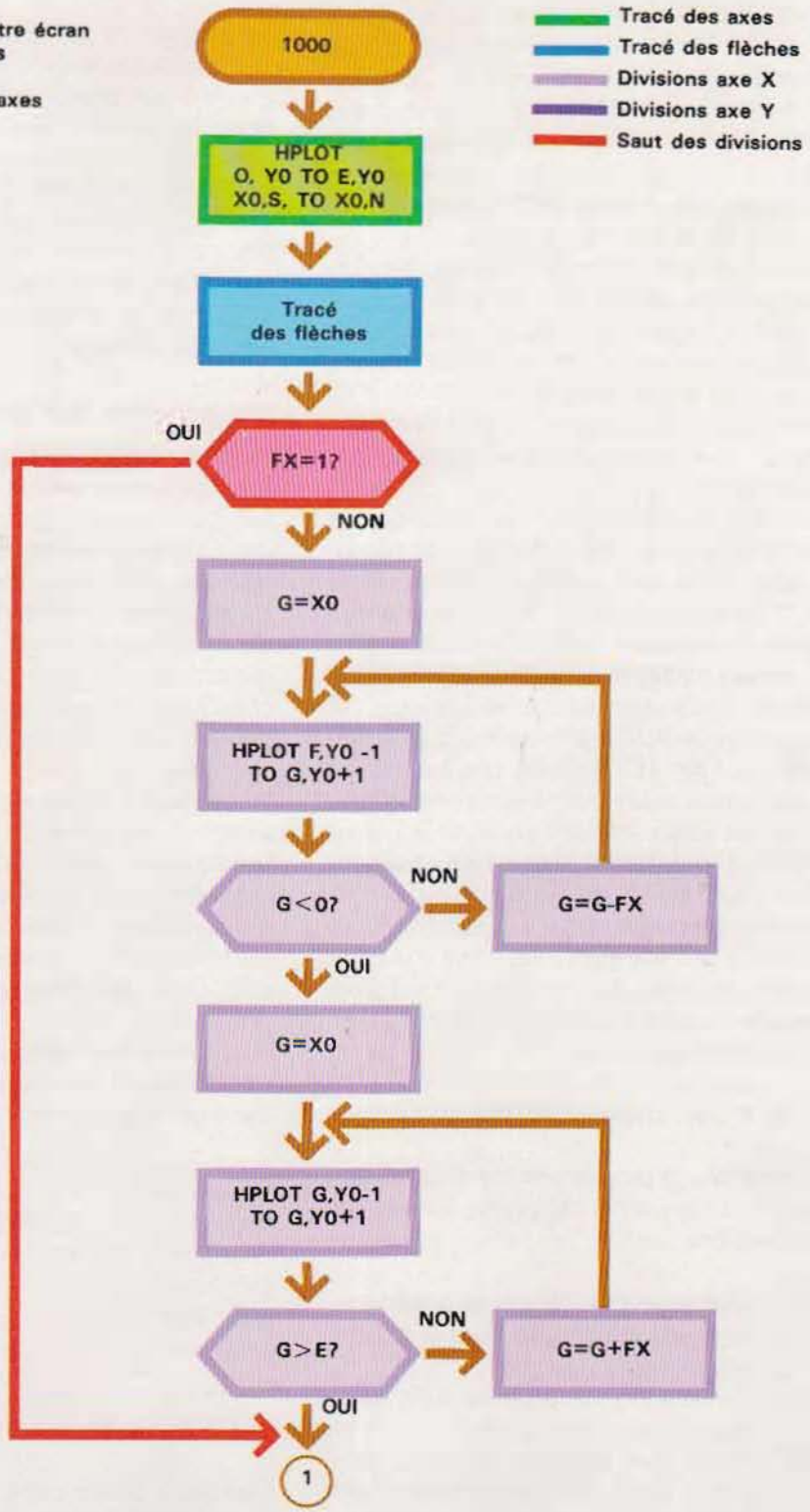
Elles sont introduites par une seule ligne de programme :

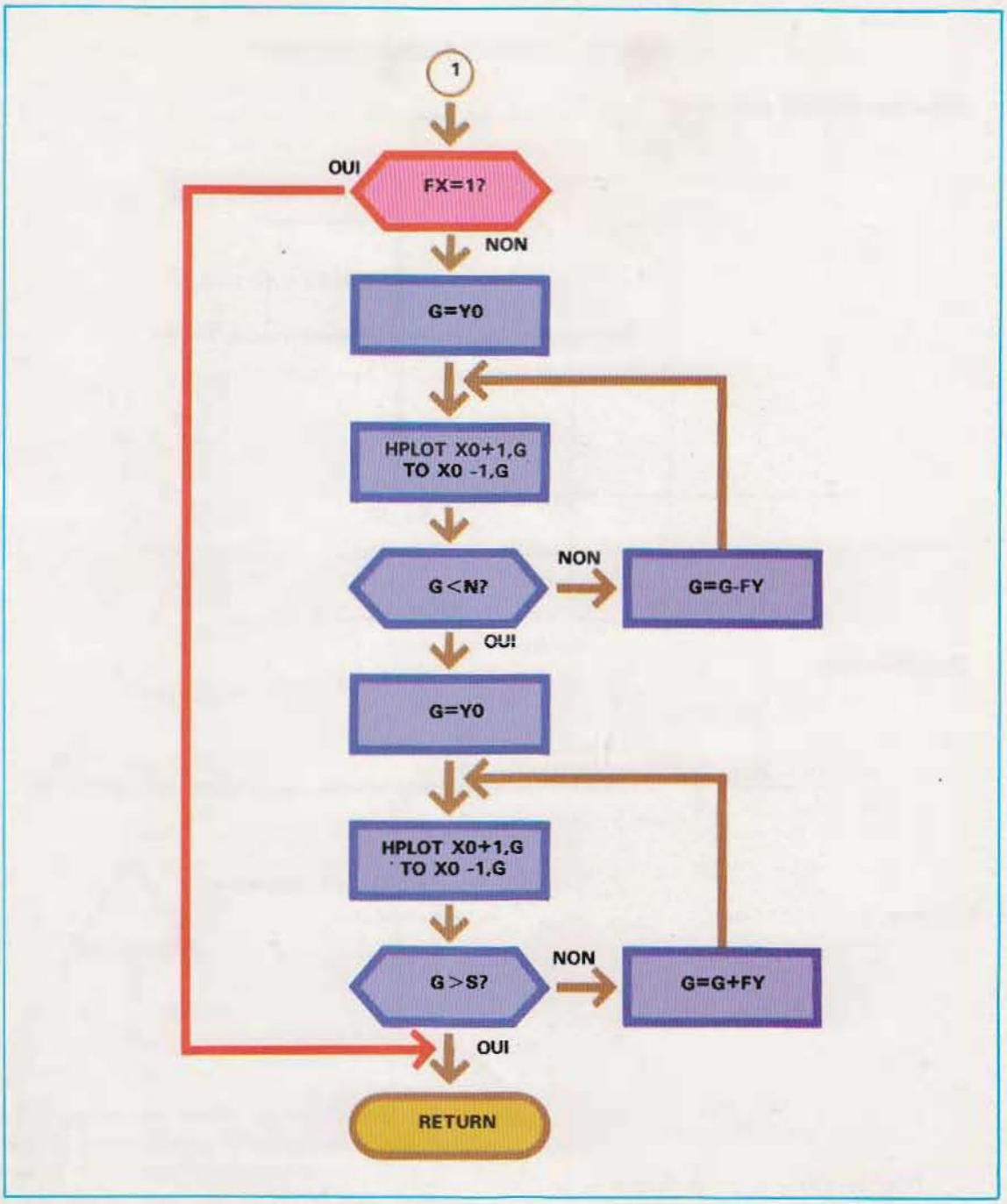
```
HPLOT X0 -1, N TO X0 +1, N TO X0, N -3
TO X0 -1, N
```

Pour les 3 autres extrémités des axes (S,E O) les instructions sont analogues.

ORGANIGRAMME DU TRACE DES AXES

Entrées :
N,S,E,O Limites fenêtre écran
Y0,Y0 Coordonnées de l'origine
FX,FY Echelle des axes





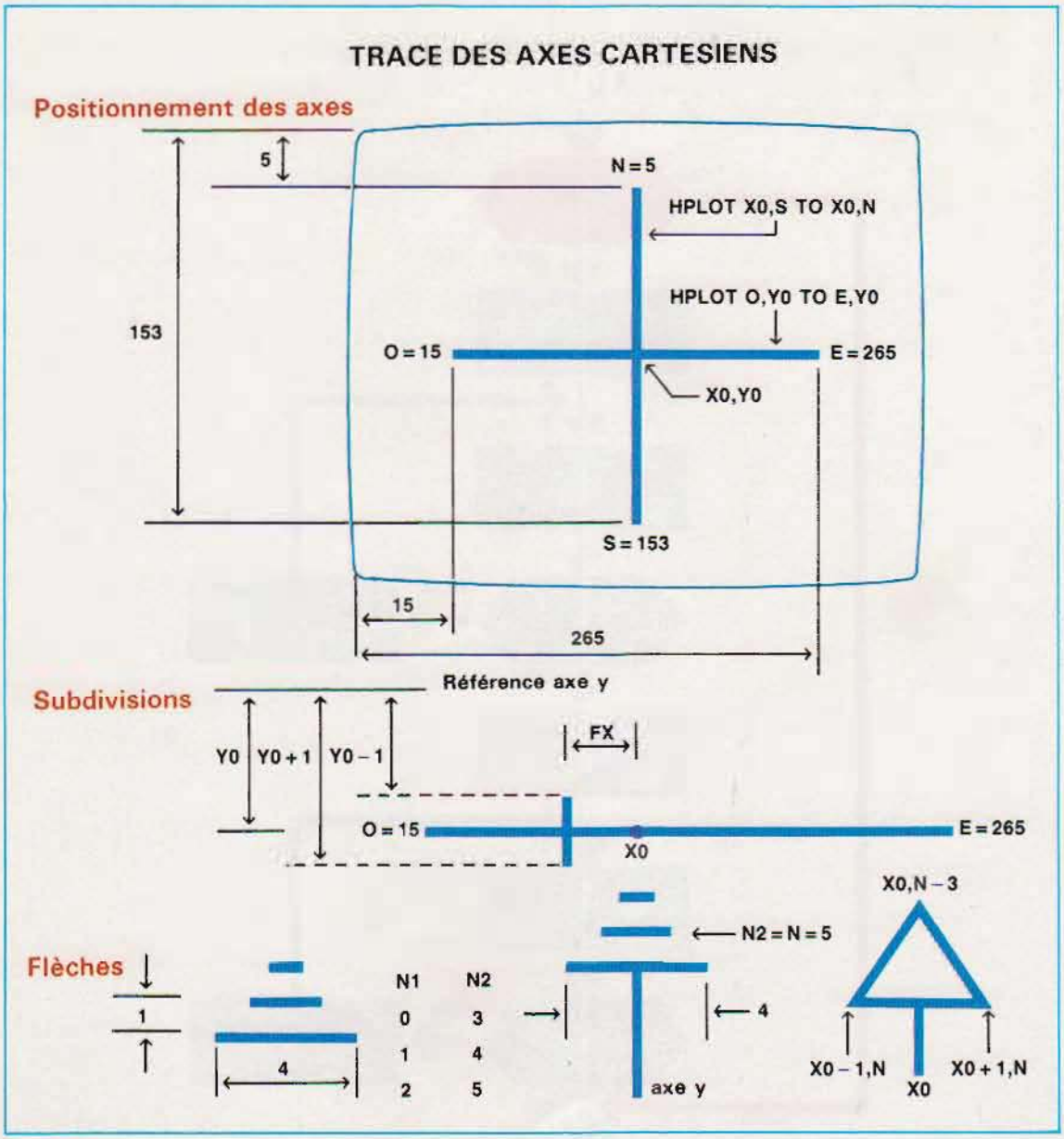
Tracé d'une droite à l'aide de deux points

Dans beaucoup d'applications graphiques, il est utile de disposer d'un sous-programme capable de déduire l'équation d'une droite passant par deux points dont on a les coordonnées. L'équation générale d'une droite s'écrivant : Y = A * X + B, pour trouver celle de la droite passant par deux points X1, Y1 et X2, Y2, on applique les formules :

$$A = \frac{Y2 - Y1}{X2 - X1}$$

$$B = \frac{(X1*X2)-(Y2*X1)}{X2-X1}$$

Soit les points X1 = 2, Y1 = 2,5 et X2 = 5, Y2 = 4, on aura :



$$A=\frac{4-2.5}{5-2}=\frac{1.5}{3}=0.5$$

$$B=\frac{(2.5*5)-(4*2)}{5-2}=\frac{12.5-8}{3}=\frac{4.5}{3}=1.5$$

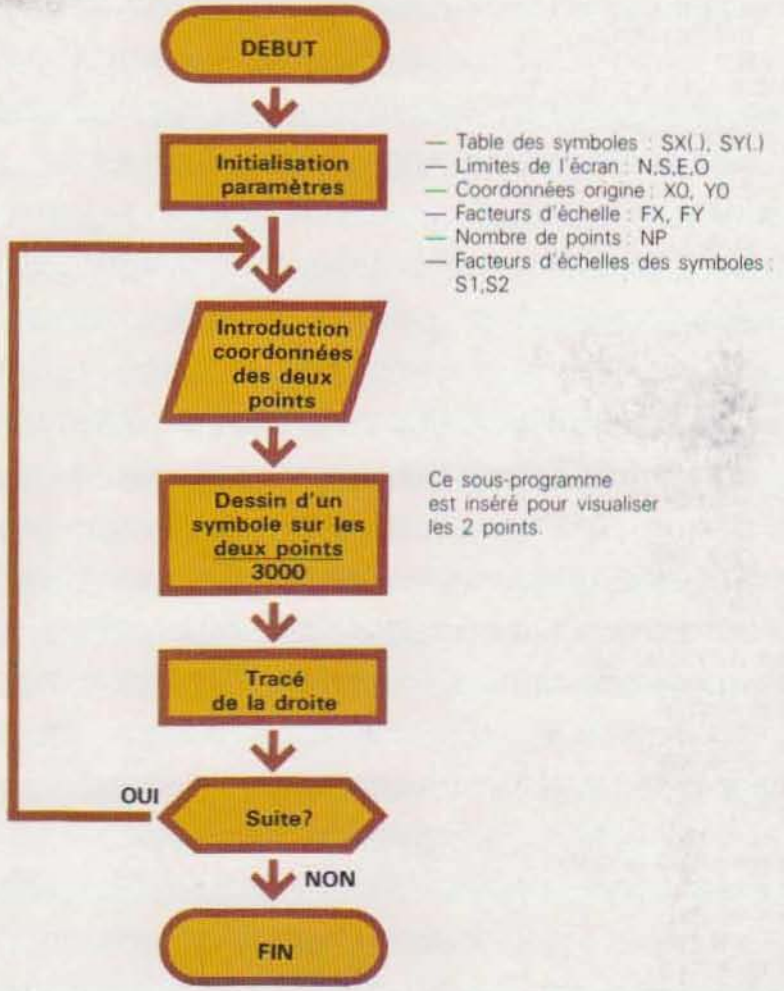
La droite aura donc pour équation :

$$Y=0.5*X+1.5$$

Les formules présentées servent à écrire un sous-programme de visualisation d'une droite passant par deux points dont les coordonnées seront introduites au clavier par exemple. Le programme demande les coordonnées du premier et du deuxième points (lignes 80 à 110) et restitue deux symboles (ici deux carrés, sous-programme 3000). Il calcule ensuite les valeurs de A et de B (lignes 120 à 130) et trace la droite (boucle : lignes 170 à 240).

Cette solution n'est indiquée qu'à titre d'illustration d'une méthode que nous utiliserons plus loin. Pour obtenir le même résultat, l'instruction du tracé d'un segment reliant les deux points donnés (HPLOT X1, Y1 TO X2, Y2) suffisait mais on n'aurait pas disposé des coefficients

DETERMINATION DE L'EQUATION ET TRACE D'UNE DROITE PASSANT PAR DEUX POINTS



A et B. La droite aurait été présentée comme un segment délimité par les points X1, Y1 et X2, Y2, et sans que le programme en connaisse la forme analytique (l'équation Y = A * X + B).

Le listing, qui se trouve en page 1446 et suivantes, montre comment se servir d'une table de déplacements pour générer un symbole sur n'importe quel point de l'écran. Les valeurs des coordonnées déterminant les côtés du symbole sont mémorisées aux lignes 20 et 30 (ici un carré). Pour le dessiner en un point donné, il faut ajouter ces valeurs à celles du point en question.

Le sous-programme 3000 fait apparaître, en outre, les coefficients S1 et S2 (égaux à 2 pour le carré, ligne 50) qui permettent de modifier la dimension de chaque segment. Les trois premiers côtés du symbole sont dessinés par la boucle 3050-3090 et le dernier côté par la ligne 3100.

Pour transporter ce programme sur d'autres machines que les Siprel, Apple et compatibles, il faut modifier les instructions VTAB, HPLOT,

TRACE D'UNE DROITE PASSANT PAR DEUX POINTS

Version Apple et compatibles

```
10 HOME
20 FOR J=1 TO 4
   : READ SX(J),SY(J)
   : NEXT J
30 DATA -1,1,1,1,-1,-1,-1
40 N=5
   : S=153
   : E=265
   : O=15
50 X0=140
   : Y0=80
   : FX=5
   : FY=5
   : NP=10
   : S1=2
   : S2=2
60 GOSUB 1000
70 DEF FN R(X)=A*X+B
80 VTAB 23
   : INPUT "COORDONNEES DU PREMIER POINT: " ; X1,Y1
90 XS=X1*FX+X0
   : YS=Y0-Y1*FY
   : GOSUB 3000
100 HOME
110 VTAB 23
   : INPUT "COORDONNEES DU DEUXIEME POINT: " ; X2,Y2
120 A=(Y2-Y1)/(X2-X1)
130 B=((Y1*X2)-(Y2*X1))/(X2-X1)
140 XS=X2*FX+X0
   : YS=Y0-Y2*FY
   : GOSUB 3000
150 P=(X2-X1)/NP
160 X=X1
   : Y=FN R(X)
170 FOR XX=X1+P TO X2 STEP P
180 YY=FN R(XX)
190 XA=X0+FX*X
   : YA=Y0-FY*Y
200 XB=X0+FX*XX
   : YB=Y0-FY*YY
210 IF XA>265 OR XA<15 OR XB>265 OR XB<15 OR YA>153 OR YA<5 OR YB>153 OR YB<5 THEN 230
220 HPLOT XA,YA TO XB,YB
230 X=XX
   : Y=YY
240 NEXT
250 HOME
260 VTAB 23
270 INPUT "POURSUIVRE? (O/N) " ; S$
280 IF S$="O" THEN RUN
290 END
997 REM ----------------
998 REM TRACE DES AXES
999 REM ----------------
1000 HGR
   : HCOLOR=3
1010 HPLOT 0-15,N-5 TO E+14,N-5 TO E+14,S+3 TO 0-15,S+3 TO 0-15,N-5
1020 HPLOT 0,Y0 TO E,Y0
   : HPLOT X0,S TO X0,N
1030 N2=N
   : FOR N1=2 TO 0 STEP -1
```

```
    : HPLOT X0-N1,N2 TO X0+N1,N2
    : N2=N2-1
    : NEXT N1
1040 N2=E
    : FOR N1=2 TO 0 STEP -1
    : HPLOT N2,Y0-N1 TO N2,Y0+N1
    : N2=N2+1
    : NEXT N1
1050 HPLOT E+6,Y0-2 TO E+11,Y0+3
    : HPLOT E+11,Y0-2 TO E+6,Y0+3
1060 HPLOT X0-10,N-2 TO X0-7,N+1
    : HPLOT X0-4,N-2 TO X0-10,N+3
1070 IF FX=1 THEN 1140
1080 FOR G=X0 TO 0 STEP -FX
1090 HPLOT G,Y0+1 TO G,Y0-1
1100 NEXT G
1110 FOR G=X0 TO E STEP FX
1120 HPLOT G,Y0+1 TO G,Y0-1
1130 NEXT G
1140 IF FY=1 THEN 1210
1150 FOR G=Y0 TO N STEP -FY
1160 HPLOT X0+1,G TO X0-1,G
1170 NEXT G
1180 FOR G=Y0 TO S STEP FY
1190 HPLOT X0+1,G TO X0-1,G
1200 NEXT G
1210 RETURN
3000 REM ------------------------
3010 REM DESSINER DES SYMBOLES
3020 REM ------------------------
3030 XA=SX(1)*S1+XS
    : YA=SY(1)*S2+YS
3040 XC=XA
    : YC=YA
3050 FOR I=2 TO 4
3060 XB=SX(I)*S1+XS
    : YB=SY(I)*S2+YS
3070 HPLOT XA,YA TO XB,YB
3080 XA=XB
    : YA=YB
3090 NEXT I
3100 HPLOT XB,YB TO XC,YC
3110 RETURN
```

Version Olivetti M20

```
10 CLEAR
20 MP=10
30 W?=WINDOW(2,40)
40 DEF FN R(X)=A*X+B
50 CLS%2
60 LINE%2(0,0)-(511,255),3,B
70 CLS
80 SCALE%2,-50,50,-50,50
90 GOSUB 270
100 INPUT "PREMIER POINT" ; X1,Y1
110 XA=X1
    : YA=Y1
    : GOSUB 3000
120 INPUT "DEUXIEME POINT" ; X2,Y2
130 XA=X2
    : YA=Y2
    : GOSUB 3000
140 P=(X2-X1)/MP
```

```
150 A=(Y2-Y1)/(X2-X1)
160 B=((Y1*X2)-(Y2*X1))/(X2-X1)
170 X=X1
    : Y=FN R(X)
180 FOR XX=X1+P TO X2 STEP P
190 YY=FN R(XX)
200 LINE%2(X,Y)-(XX,YY),2
210 X=XX
    : Y=YY
220 NEXT
230 INPUT "POURSUIVRE? (O/N)" ; A$
240 IF A$="O" THEN 10
250 CLEAR
    : END
270 REM **** TRACE DES AXES ****
280 LINE%2(0,-50)-(0,50)
290 LINE%2(-50,0)-(50,0)
300 FOR Q=0 TO 49 STEP 2
310 LINE%2(-.5,Q)-(.5,Q) 320 LINE%2(Q,-.5)-(Q,.5)
330 NEXT Q
340 FOR Q=0 TO -49 STEP -2
350 LINE%2(-.5,Q)-(.5,Q)
360 LINE%2(Q,-.5)-(Q,.5)
370 NEXT Q
380 RETURN
3000 REM **** DESSINER DES SYMBOLES ****
3010 LINE%2(XA-2,YA-2)-(XA+2,YA+2),2,B
3020 RETURN
```

HCOLOR, HGR et les contrôles de la ligne 210, en tenant compte des valeurs numériques de l'écran utilisé.
On a donné, à titre d'exemple, un listing convenant à un Olivetti M20. Ce programme fait appel à certaines instructions particulières que nous détaillerons par la suite.

Régression linéaire des moindres carrés

L'analyse d'un phénomène consiste à observer les valeurs prises par les diverses variables, au cours de l évolution dudit phénomène, à condition qu'il n'y ait qu'une seule source de variation.
Le cas le plus simple est déterminé par deux variables dont l'une est indépendante. L'ensemble des valeurs prises par cette variable montre, sous forme graphique, l'évolution du phénomène observé.
Partons d'un exemple fondé sur 5 observations ayant donné les valeurs suivantes pour les variables X et Y.

Observation	X	Y
a	2,5	3
b	4	3,5
c	5	5
d	7	4,5
e	9	6

On constate qu'on ne peut déduire immédiatement la valeur de Y correspondant à une valeur de X non observable. Ces informations sont obtenues selon une méthode — déjà présentée — qui consiste à extrapoler linéairement chaque paire de valeurs observées.
L'évolution réelle est ainsi assimilée à une série de segments joignant les points a, b, c, d, e. Le résultat est un graphique (voir ci-contre) qui peut être très éloigné de la réalité, avec des valeurs déduites grossières. Dans cette méthode, les équations représentant le phénomène sont celles des quatre droites qui joignent deux à deux les points connus.
Chaque équation (donc la forme analytique décrivant le phénomène) est déduite de la méthode développée dans le paragraphe précé-

dent (droite passant par deux points donnés). Il existe une seconde méthode appelée « régression linéaire des moindres carrés » qui, sous certaines conditions, fournit des résultats plus fiables.

Supposons, pour l'instant, que l'évolution du phénomène puisse être symbolisée par une droite. Grâce à une série d'observations sur les variables, on obtiendra des valeurs qui n'appartiennent pas toutes à la droite. Même si la loi régissant le phénomène est représentée par une droite, les valeurs mesurées pourraient toutefois être encore les points a, b, c, d, e (ligne rouge) lesquels, à une exception près (b), ne coïncident pas avec la droite. Cette « dispersion » dépend de nombreux facteurs, liés à la nature du phénomène observé.

Dans un phénomène physique, les erreurs peuvent être instrumentales et accidentelles. Quel que soit son degré de sophistication, un équipement ne fournit jamais qu'une valeur approximative de la grandeur.

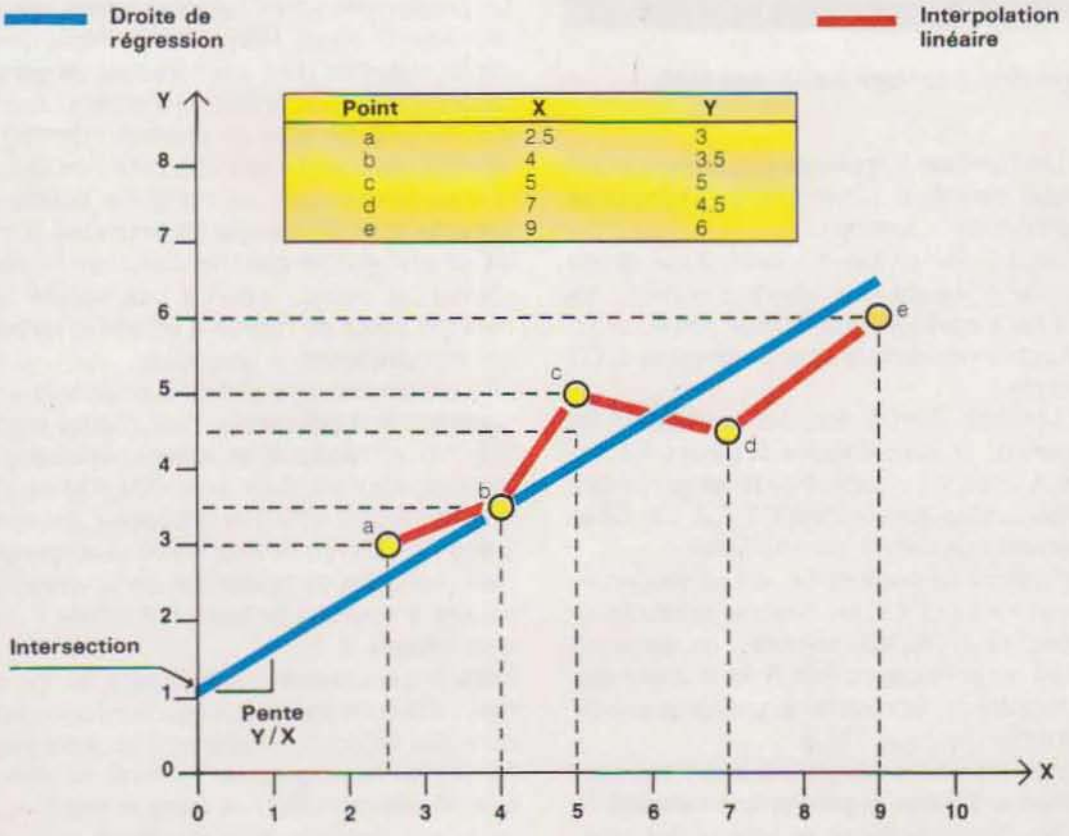
Autre cause d'erreur : l'influence de facteurs extérieurs tels que vibrations mécaniques, variations de température ou d'humidité, etc.

Ces erreurs se combinent et ne s'annulent que d'une manière aléatoire. D'ailleurs, la vérification de cette dernière situation n'est pas prévisible : même si les valeurs sont exactes à un moment donné, il n'existe pas de moyen de le vérifier. Les données doivent donc de toute manière être considérées comme entachées d'erreur.

En statistiques, les erreurs sont plus complexes et plus difficiles à repérer.

La méthode des moindres carrés permet de rechercher la forme analytique de la fonction qui se rapproche le plus du phénomène exami-

EVOLUTION D'UN PHENOMENE RESULTANT DE 5 OBSERVATIONS



Marka

Application graphique sur Olivetti M20.

né. Les formules à appliquer proviennent d'une analyse complexe dépendant de l'hypothèse de départ de la fonction.

Le cas le plus simple est celui d'une droite (régression linéaire des moindres carrés) : les formules à appliquer pour obtenir son équation sont contenues dans le sous-programme 8000 ci-contre.

Les variables A et B sont les coefficients de l'équation ; ils sont calculés à travers les valeurs X (.) et Y (.) observées, N est le nombre d'observations, les variables C1, C2, C3, C4 et D servant aux calculs intermédiaires.

Pour utiliser ce programme, il faut ranger en mémoire X (.) et Y (.) les mesures et les observations, et en N leur nombre ; en sortie on obtient les coefficients A et B de la droite qui, par hypothèse, représente le phénomène (voir organigramme page 1452).

La première phase consiste à tracer les axes cartésiens. Ensuite, le programme demande le nombre de points (N) et les valeurs des coordonnées des points observés (boucle avec indice I). Pour chaque paire de coordonnées, le sous-programme 5900 dessine un carré sur le point correspondant à la valeur observée. Après la boucle d'introduction, on appelle le sous-programme 8000 (ci-contre), qui calcule les coefficients A et B. Le résultat est matérialisé par une droite.

En réalité, le déroulement du programme est beaucoup plus complexe, en particulier au niveau des facteurs d'échelle.

En phase d'introduction, on peut avoir des valeurs de X ou de Y qui, d'après l'échelle adoptée, « sortiraient » des limites de l'écran. De même, après avoir calculé l'équation de la droite (Y=B * X+A), dessinée à partir des valeurs X=XN et X=XM (respectivement minimum et maximum sur l'axe X), il peut arriver que les valeurs correspondantes de Y dépassent les limites de l'écran.

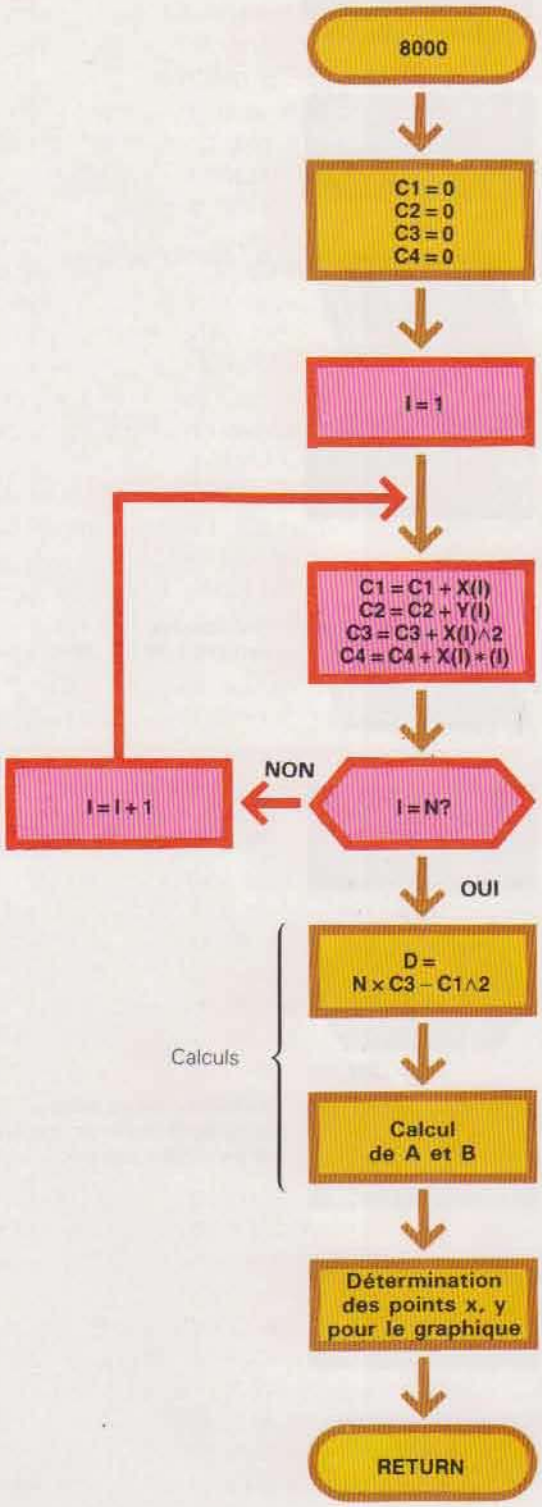
Le premier contrôle (limites des valeurs X et Y) s'effectue après chaque entrée, tandis que celui sur la droite est réalisé par le sous-programme 8000 (dont le listing est différent de l'organigramme présenté).

Le programme (dont l'organigramme détaillé se trouve en pages 1453, 1454 et 1455) révèle des complexités dues à la nécessité de prévoir un changement automatique d'échelle. Au début, l'origine des axes est placé en X0=140 et Y0=80, avec un facteur d'échelle F = 0,7. A chaque introduction, on lance une boucle de contrôle pour vérifier que les nouveaux points se situent encore dans le cadre de l'échelle choisie par défaut : sinon, il faut calculer une nouvelle valeur du facteur d'échelle et redessiner complètement le graphique.

La logique exposée s'obtient avec une série de contrôles et d'indicateurs. Pour chaque entrée X(I), Y(I), on recalcule les valeurs maximales et minimales sur les deux axes (XN, XM et YN, YM, lignes 270 à 360) et la longueur des intervalles (DX, DY) parmi lesquels on sélectionne le plus grand. En se basant sur cette valeur, on calcule à nouveau le facteur d'échelle F, qui sera inférieur à 1.

Dans le cas contraire, cela signifie qu'une erreur a été commise et que les données ne sont donc pas valides. Le traitement est alors interrompu et le programme reprend au début, avec l'instruction RUN et après le test F < 1. Le facteur d'échelle étant déterminé, on passe au calcul des nouvelles coordonnées X0, Y0

CALCUL DES COEFFICIENTS A ET B DE LA DROITE DE REGRESSION



Initialisation

Boucle sur les points observés

Calculs

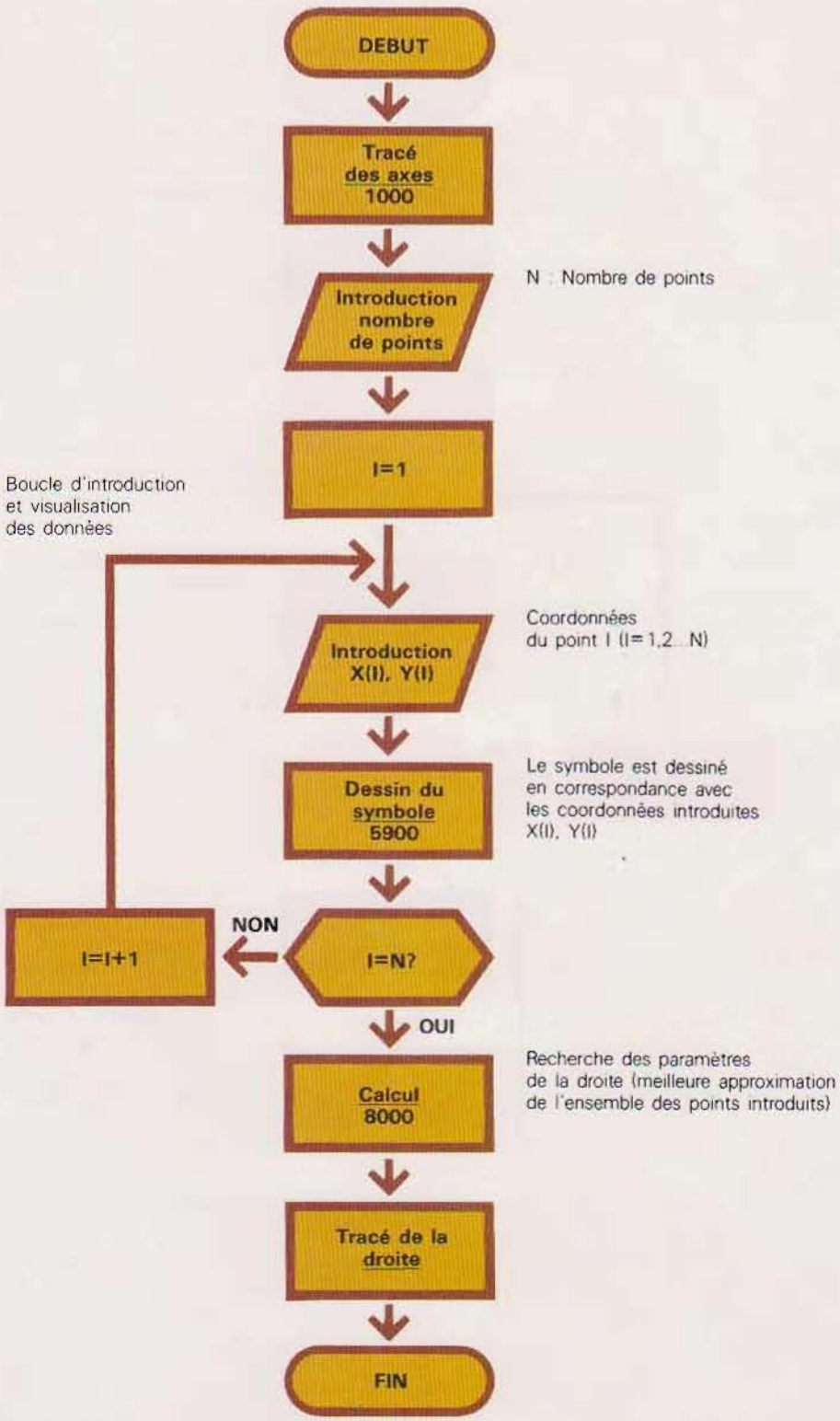
Paramètres de la droite de régression des moindres carrés

$$A = \frac{C2 * C3 - C1 * C4}{D}$$

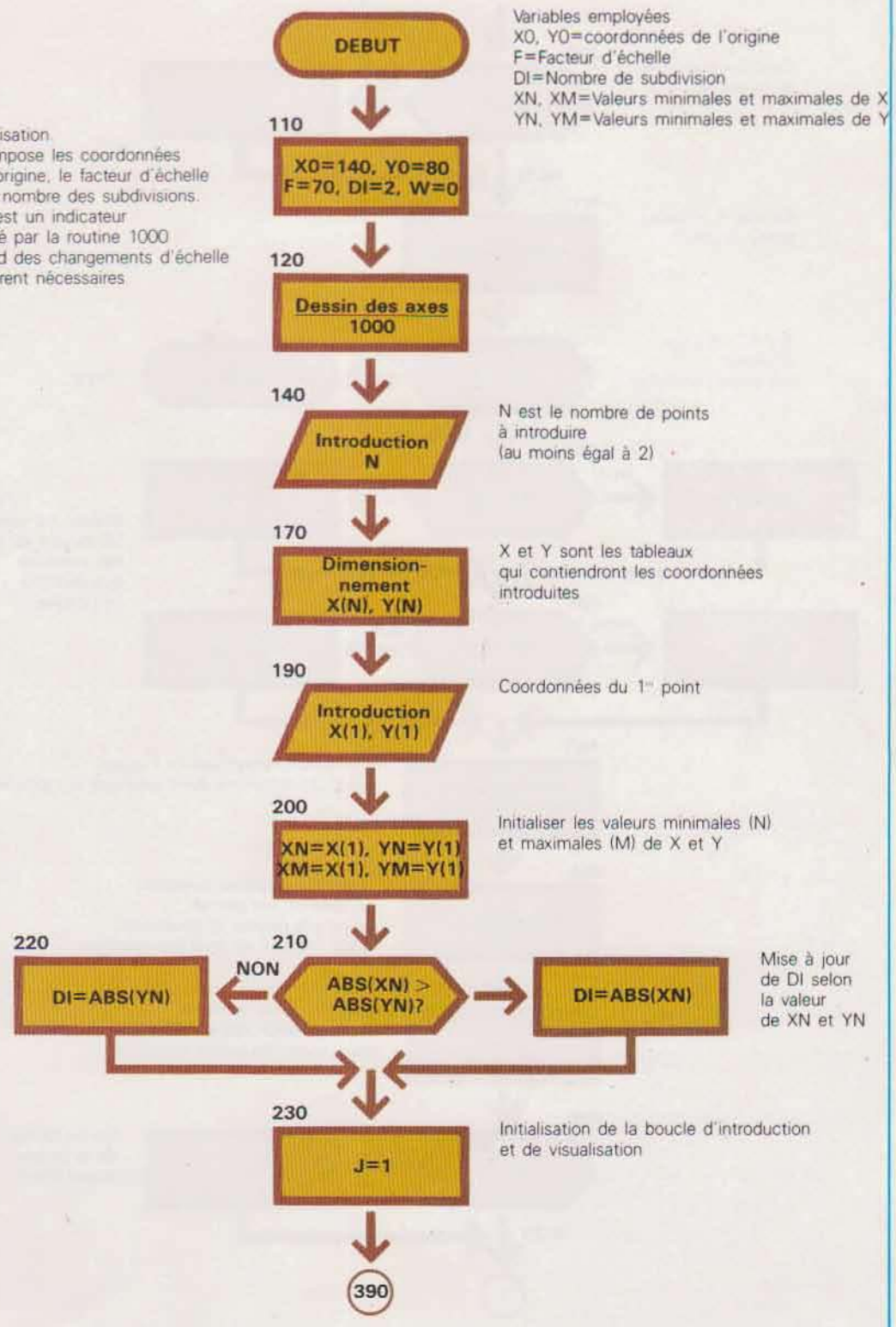
$$B = \frac{N * C4 - C1 * C2}{D}$$

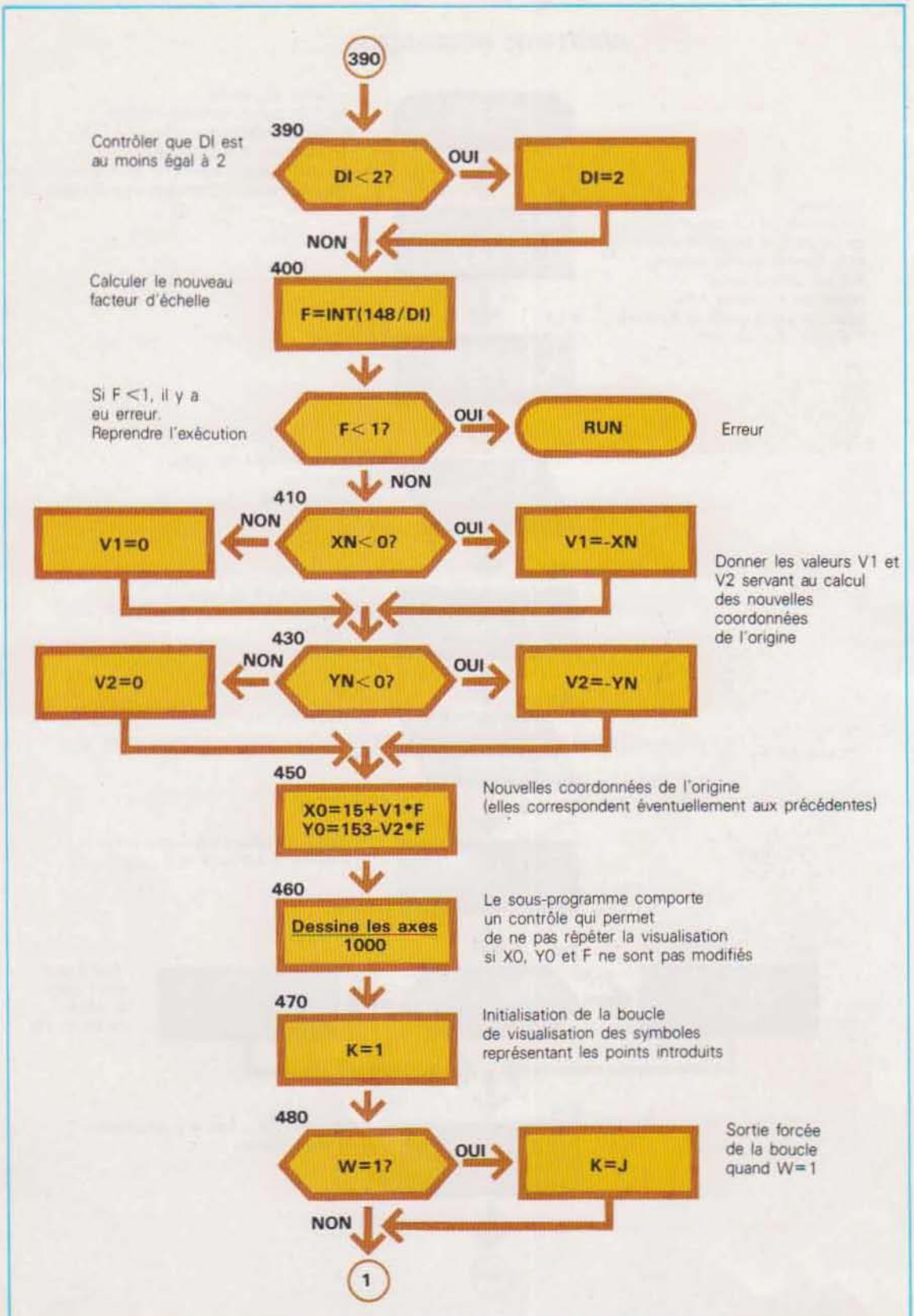
$$Y = A * X + B$$

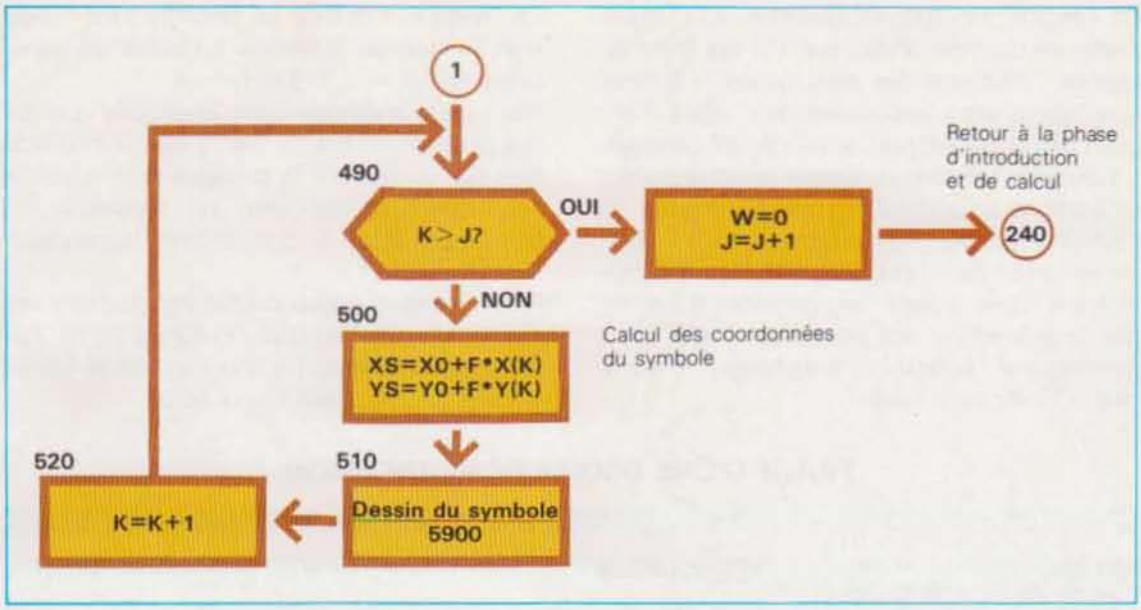
REGRESSION LINEAIRE DES MOINDRES CARRES

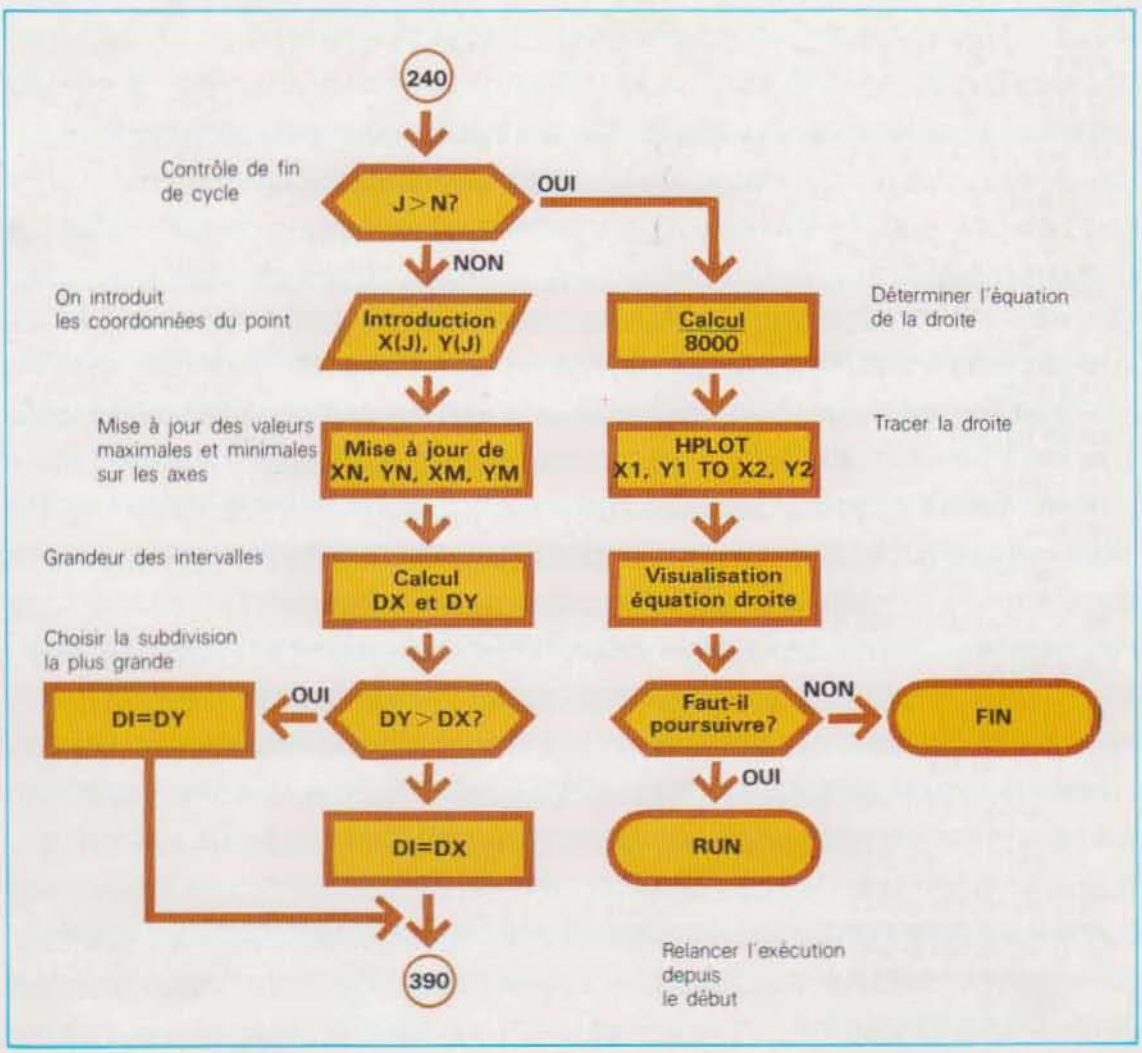


DROITE DE REGRESSION




390
390
Contrôler que DI est au moins égal à 2
DI<2?
OUI
DI=2
NON
400
Calculer le nouveau facteur d'échelle
F=INT(148/DI)
Si F<1, il y a eu erreur. Reprendre l'exécution
F<1?
OUI
RUN
Erreur
NON
410
NON
V1=0
XN<0?
OUI
V1=-XN
Donner les valeurs V1 et V2 servant au calcul des nouvelles coordonnées de l'origine
430
NON
V2=0
YN<0?
OUI
V2=-YN
450
X0=15+V1*F
Y0=153-V2*F
Nouvelles coordonnées de l'origine (elles correspondent éventuellement aux précédentes)
460
Dessine les axes
1000
Le sous-programme comporte un contrôle qui permet de ne pas répéter la visualisation si X0, Y0 et F ne sont pas modifiés
470
K=1
Initialisation de la boucle de visualisation des symboles représentant les points introduits
480
W=1?
OUI
K=J
Sortie forcée de la boucle quand W=1
NON
1


1
490
K > J?
OUI
W=0
J=J+1
240
Retour à la phase d'introduction et de calcul
NON
500
XS=X0+F*X(K)
YS=Y0+F*Y(K)
Calcul des coordonnées du symbole
520
510
K=K+1
Dessin du symbole
5900


240
Contrôle de fin de cycle
J > N?
OUI
NON
On introduit les coordonnées du point
Introduction X(J), Y(J)
Calcul 8000
Déterminer l'équation de la droite
Mise à jour des valeurs maximales et minimales sur les axes
Mise à jour de XN, YN, XM, YM
HPLOT X1, Y1 TO X2, Y2
Tracer la droite
Grandeur des intervalles
Calcul DX et DY
Visualisation équation droite
Choisir la subdivision la plus grande
DI=DY
OUI
DY > DX?
Faut-il poursuivre?
NON
FIN
OUI
DI=DX
RUN
390
Relancer l'exécution depuis le début

de l'origine. Le sous-programme 1000 comporte un contrôle (indicateur W) qui évite de répéter l'affichage des axes quand le facteur d'échelle et les coordonnées de l'origine n'ont pas changé par rapport au précédent passage. La dernière fonction redessine tous les points, toujours sous la condition d'une variation du facteur d'échelle. Au passage, notons bien la sortie forcée de la boucle. Si le facteur d'échelle varie, une boucle représentant tous les points précédents est nécessaire ; elle est à éviter quand l'échelle reste inchangée, c'est-à-dire si l'indicateur vaut 1.

La fonction s'obtient en posant, sous condition, l'indice de la boucle à l'extrémité supérieure (IF W = 1 THEN K = J).

Mais cette méthode n'est applicable que sur une quantité limitée de machines. La modification qui en permet la généralisation consiste simplement à introduire une dérivation (by pass) pour toute la boucle avec la condition W=1.

On trouvera ci-dessous le listing du programme, en version Personal Kid (Siprel 2010, Apple et compatibles). La transposition se fait en remplaçant certaines instructions.

TRACE D'UNE DROITE DE REGRESSION

```
10 REM ------------------------
20 REM DROITES DE REGRESSION
30 REM ------------------------
40 REM
50 REM
60 REM
70 FOR J=1 TO 4
80 READ XS(J),SY(J)
90 NEXT J
100 DATA -1,1,1,1,1,-1,-1,-1
110 X0=140
    : Y0=80
    : E=70
    : D1=2
    : W=0
120 GOSUB 1000
130 HOME
    : VTAB 22
140 INPUT "NOMBRE DE POINTS. " ; N
150 N=INT(N)
160 IF N<2 THEN 130
170 DIM X(N),Y(N)
180 HOME
    : VTAB 22
190 INPUT "POINT N.1: " ; X(1),Y(1)
200 XN=X(1)
    : XM=X(1)
    : YN=Y(1)
    : YM=Y(1)
210 IF ABS(XN)>ABS(YN) THEN D1=ABS(XN)
    : GOTO 230
220 D1=ABS(YN)
    : GOTO 230
230 J=1
    : GOTO 390
240 IF J>N THEN 540
250 HOME
    : VTAB 22
260 PRINT "POINT N."J" "
    : INPUT " " ; X(J),Y(J)
270 IF X(J)>XM THEN XM=X(J)
280 IF X(J)<XN THEN XN=X(J)
290 IF Y(J)>YM THEN YM=Y(J)
300 IF Y(J)<YN THEN YN=Y(J)
310 IF XN>0 AND XN<0 THEN DX=XM-XN
    : GOTO 340
```

```
320 IF XN>=0 AND XM>0 THEN DX=XM
    : GOTO 340
330 DX=-XN
340 IF YN<0 AND YM>0 THEN DY=YM-YN
    : GOTO 370
350 IF YN>=0 AND YM>0 THEN DY=YM
    : GOTO 370
360 DY=-YN
370 IF DY>DX THEN DI=DY
    : GOTO 390
380 DI=DX
390 IF DI<2 THEN DI=2
400 F=INT(140/DI)
    : IF F<1 THEN RUN
410 IF XN<0 THEN U1=-XN
    : GOTO 430
420 U1=0
430 IF YN<0 THEN U2=-YN
    : GOTO 450
440 U2=0
450 X0=U1*F+15
    : Y0=153-U2*F
460 GOSUB 1000
470 K=1
480 IF W=1 THEN K=J
490 IF K>J THEN W=0
    : GOTO 530
500 XS=X(K)*F+X0
    : YS=Y0-Y(K)*F
510 GOSUB 5900
    : REM SYMBOLE
520 K=K+1
    : GOTO 490
530 J=J+1
    : GOTO 240
540 GOSUB 8000
    : REM CALCUL
550 HPLOT X1,Y1 TO X2,Y2
560 S$="-"
570 IF A>0 THEN S$="+"
580 HOME
    : VTAB 22
590 PRINT "LA DROITE LA PLUS PROCHE EST: "
600 IF A=0 AND B=0 OR D=0 THEN INVERSE
    : PRINT "X= "XN
    : GOTO 630
610 IF B=0 THEN INVERSE
    : PRINT "Y= "A
    : GOTO 630
620 INVERSE
    : PRINT "Y= "B " *X "S$ " "ABS(A)
630 NORMAL
640 VTAB 24
650 PRINT "POURSUIVRE? (O/N)"
    : GET A$
660 IF A$="O" THEN RUN
670 TEXT
    : HOME
```

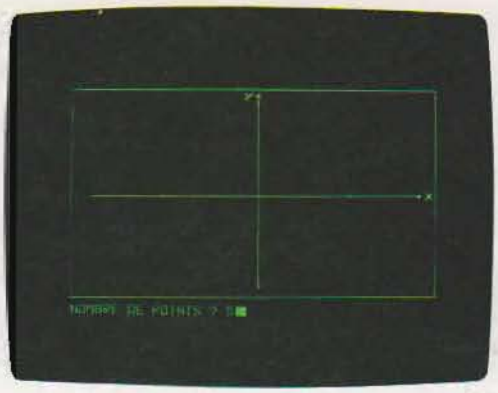
```
    : END
997 REM ---------------------
998 REM DESSINER LES AXES
999 REM ---------------------
1000 IF X0=XP AND Y0=YP AND F=FF THEN W=1
    : RETURN
1010 HGR
    : HCOLOR=3
1020 HPLOT 0,0 TO 279,0 TO 279,159 TO 0,159 TO 0,0
1030 HPLOT 15,Y0 TO 256,Y0
    : HPLOT X0,153 TO X0,5
1040 N2=5
    : FOR N1=2 TO 0 STEP -1
    : HPLOT X0-N1,N2 TO X0+N1,N2
    : N2=N2-1
    : NEXT N1
1050 N2=265
    : FOR N1=2 TO 0 STEP-1
    : HPLOT N2,Y0-N1 TO N2,Y0+N1
    : N2=N2+1
    : NEXT N1
1060 HPLOT 271,Y0-2 TO 276,Y0+3
    : HPLOT 276,Y0-2 TO 271,Y0+3
1070 HPLOT X0-10,3 TO X0-7,6
    : HPLOT X0-4,3 TO X0-10,8
1080 IF F=1 THEN 1220
1090 FOR G=X0 TO 15 STEP -F
1100 HPLOT G,Y0+1 TO G,Y0-1
1110 NEXT G
1120 FOR G=X0 TO 265 STEP F
1130 HPLOT G,Y0+1 TO G,Y0-1
1140 NEXT G
1150 FOR G=Y0 TO 5 STEP -F
1160 HPLOT X0+1,G TO X0-1,G
1170 NEXT G
1180 FOR G=Y0 TO 153 STEP F
1190 HPLOT X0+1,G TO X0-1,G
1200 NEXT G
1210 XP=X0
    : YP=Y0
    : FF=F
1220 RETURN
5899 REM ---------
5900 REM SYMBOLE
5901 REM ---------
5910 S=F/4
    : IF S>5 THEN S=5
5920 IF S<2 THEN S=2
5930 XA=SX(1)*S+XS
    : YA=SY(1)*S+YS
5940 XC=XA
    : YC=YA
5950 FOR T=2 TO 4
5960 XB=SX(T)*S+XS
    : YB=SY(T)*S+YS
5970 HPLOT XA,YA TO XB,YB
5980 XA=XB
    : YA=YB
5990 NEXT T
6000 HPLOT XB,YB TO XC,YC
6010 RETURN
```

```
7999 REM --------
8006 REM CALCUL
8010 REM --------
8020 C1=0
   : C2=0
     C3=0
   : C4=0
8030 FOR I=1 TO N
8040 C1=C1+X(I)
8050 C2=C2+Y(I)
8060 C3=C3+X(I)^2
8070 C4=C4+X(I)*Y(I)
8080 NEXT I
8090 D=N*C3-C1^2
8100 IF D=0 THEN Y1=Y0-YN*F
   : Y2=Y0-YM*F
   : X1=X0+XN*F
   : X2=X1
   : RETURN
8110 A=(C2*C3-C1*C4)/D
8120 B=(N*C4-C1*C2)/D
8130 IF B=0 THEN Y1=Y0-A*F
   : Y2=Y1
   : X1=X0+XN*F
   : X2=X0+XM*F
     RETURN
8140 IF A=0 AND B=0 THEN Y1=Y0-YN*F
   : Y2=Y0-YM*F
   : X1=X0+XN*F
   : X2=X1
   : RETURN
8150 WS=0
8160 X=(YM-A)/B
   : Y=YM
8170 IF X>=XN AND X<=XM THEN GOSUB 8270
8180 X=(YN-A)/B
   : Y=YN
8190 IF X>=XN AND X<=XM THEN GOSUB 8270
8200 X=XM
   : Y=B*XM+A
8210 IF Y>=YN AND Y<=YM THEN GOSUB 8270
8220 X=XN
   : Y=B*XN+A
8230 IF Y>=YN AND Y<=YM THEN GOSUB 8270
8240 X1=X0+X1*F
   : X2=X0+X2*F
8250 Y1=Y0-Y1*F
   : Y2=Y0-Y2*F
8260 RETURN
8270 IF WS=1 THEN GOTO 8290
8280 X1=X
   : Y1=Y
   : WS=1
   : RETURN
8290 X2=X
   : Y2=Y
   : RETURN
```

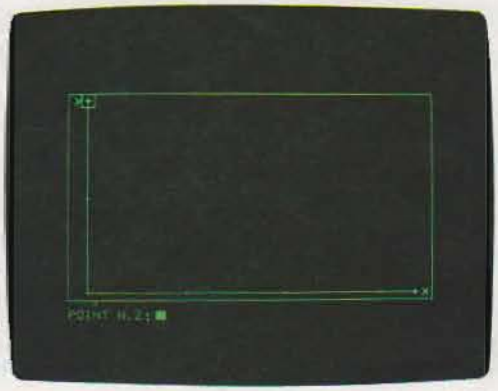
REGRESSION LINEAIRE DES MOINDRES CARRES

Le système présente des axes cartésiens et demande le nombre de points.
Pour vérifier la précision du logiciel, on va introduire 5 points appartenant à la droite Y=3*X+2.

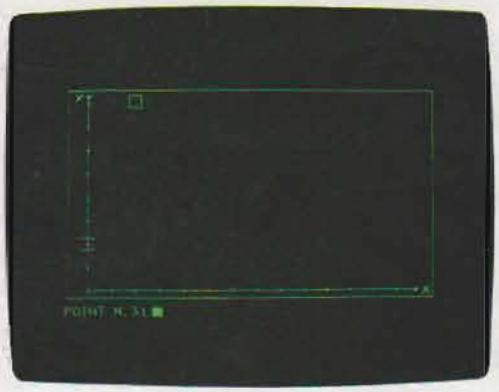
	X	Y
Point 1	0	2
Point 2	2	8
Point 3	3	11
Point 4	5	17
Point 5	10	32



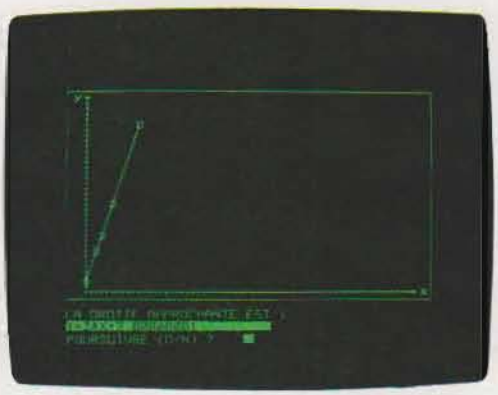
Dans le système, on a déjà introduit le 1er point de coordonnées 0,2. Le programme adapte la position des axes cartésiens de manière à disposer de toute la surface de l'écran pour la représentation des valeurs introduites.
Par rapport aux écrans précédents, l'origine s'est déplacée en bas à gauche, dans la mesure où les valeurs introduites (pour l'instant seulement le point 0,2) ne contiennent pas de nombres négatifs : l'écran est donc entièrement dédié au cadre positif du système de référence, donc la valeur maximale de Y est égale à la valeur maximale entrée.



Après l'introduction du point 2 (de coordonnées 2,8) le programme réduit l'échelle de l'axe Y de manière à l'adapter à la nouvelle abscisse (Y=8). Le point précédent semble avoir été déplacé vers le bas, tout en correspondant aux mêmes coordonnées. En effet, dans l'écran précédent, l'axe Y ne comportait que deux parties (petits traits horizontaux) et le symbole occupait la position 2 (c'est-à-dire l'ordonnée Y=2) ; dans la nouvelle situation, le programme a automatiquement modifié l'échelle en la réglant sur la nouvelle valeur maximale entrée (Y=8).



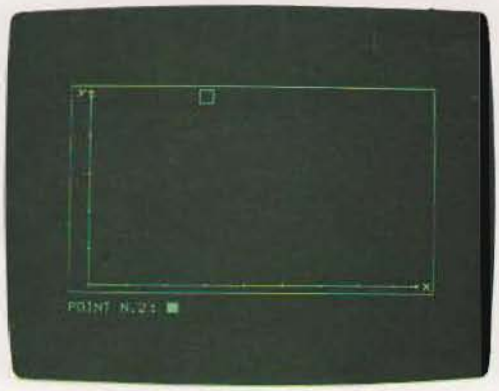
A la fin de l'introduction des données (dont le nombre a été déclaré au début) le programme présente le graphique final, qui comprend les points et la régression. De plus, il visualise l'expression analytique de la droite. Les coefficients calculés ne présentent aucune différence par rapport aux valeurs employées pour déterminer les points. La valeur 1, qui apparaît à la suite d'une série de zéros après la virgule, est due à l'absence d'arrondi dans la présentation des coefficients. Dans les applications pratiques, on tronque le résultat à la 2e ou à la 3e décimale.



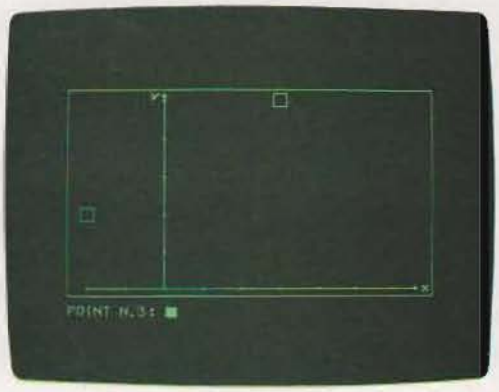
Contrairement au premier, le second exemple se réfère à un calcul de régression proprement dit, effectué sur les valeurs suivantes

	X	Y
Point 1	3	5
Point 2	- 2	
Point 3	10	6
Point 4	- 5	3
Point 5	1	- 2

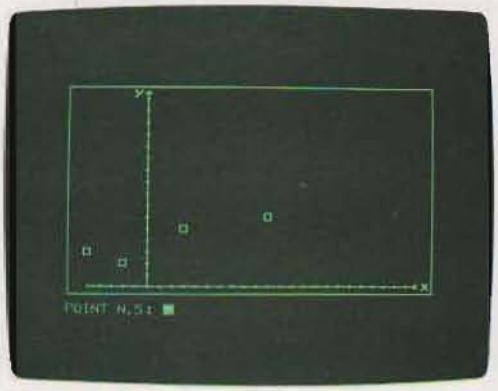
Les coordonnées du premier point sont entrées et le programme demande la 2° donnée.



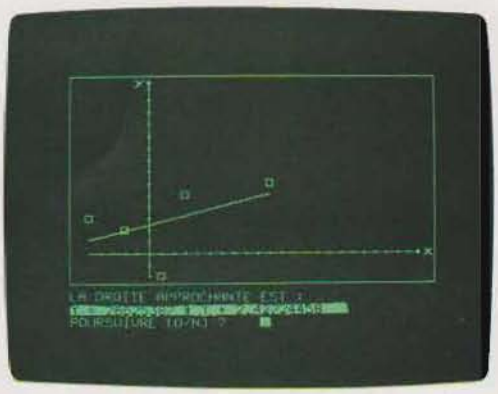
L'introduction de la première valeur négative (point 2, coordonnées -2,2) provoque le déplacement de l'axe Y vers la droite, car on a besoin d'une portion négative de l'axe Y (-2).



Les introductions suivantes déterminent l'adaptation du champ de vision (position de l'origine et facteur d'échelle). Dans la photo, on vient d'introduire le 4° point.



L'introduction du dernier point (1, -2) lance le calcul et la visualisation de la droite de régression.
Comme on peut le remarquer, les valeurs dans ce cas sont très dispersées et la droite de régression fournit une approximation relativement satisfaisante.



Programme du tracé graphique d'une fonction

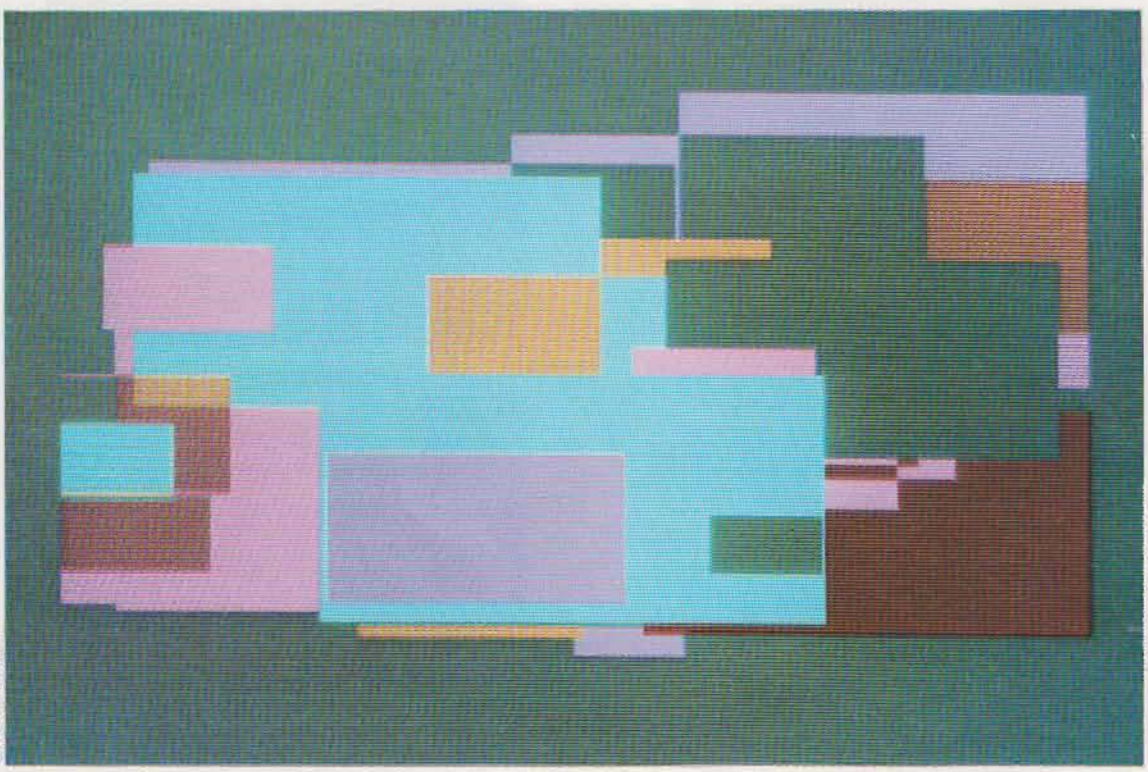
La représentation graphique d'une fonction générale exige plus de précautions que pour une simple droite.
En effet, on doit prévoir les points suivants :

— calcul du facteur d'échelle et sa normalisation
— contrôle et corrections d'erreurs
— choix des symboles par l'usager

Le facteur d'échelle se calcule très simplement comme le quotient entre les dimensions de l'écran (selon les deux axes) et les intervalles de valeurs correspondants.
Ainsi, pour représenter une fonction qui prend des valeurs comprises entre 20 et 70 sur un écran de 200 points sur l'axe Y, le facteur d'échelle sera 200/(70 -20)=200/50=4.
A partir de cette valeur, on calcule la coordonnée Y0 de l'origine qui doit coïncider avec la valeur minimale de la Y(20), plus une marge qui permettra d'éviter que les axes soient dessinés trop près du bord de l'écran.

En général, le résultat du quotient n'est pas un nombre entier. Dans l'exemple précédent, il suffit que les valeurs prises par la fonction varient entre 20 et 71 pour fournir un facteur d'échelle avec des valeurs décimales, donc difficile à interpréter.
Pour pallier cet inconvénient, on prévoit une routine de normalisation qui sélectionne la valeur la mieux adaptée parmi d'autres valeurs définies par l'utilisateur. Dans le cas précédent (valeurs 20 et 71), le facteur d'échelle 200/51=3,9 doit être arrondi à 3. Le résultat est un graphique de dimensions plus réduites (on n'exploite pas tout l'écran) mais plus facile à interpréter.
La fonction prend des valeurs soit négatives, soit positives : il convient par conséquent d'ajuster la position de l'origine (X0, Y0) de manière à permettre l'exploitation optimale de l'écran. La fonction peut, en outre, avoir une forme telle qu'elle fournit des valeurs très petites ou très grandes, supérieures aux limites de présentation. On résout ce problème en introduisant un facteur multiplicateur égal à une puissance de 10.

Visualisation de rectangles superposés sur écran couleur.



Centro THC/Marka

PRINCIPALES VARIABLES EMPLOYEES DANS LE PROGRAMME

V1	= Facteur d'échelle maximale prévu
SX(4), SY(4)	= Déplacements pour dessiner le symbole
VN/10)	= Facteurs d'échelle prévus (pour arrondis)
K$	= Indicateur d'axe (X ou Y)
XS,YS	= Coordonnées exprimées en points écran
XM,XN	= Valeurs maximales et minimales axe X
YM,YN	= Valeurs maximales et minimales de la fonction (Y)
FX,FY	= Facteurs d'échelle axe X et axe Y
DX	= Pas axe X
NP	= Nombre de points à présenter
LX,LY	= Longueur des axes
ER	= Code de la dernière erreur apparue
LOC	= Numéro de la ligne où l'erreur s'est vérifiée
K	= Puissance de 10 dans le facteur d'échelle
S1,S2	= Facteur d'échelle pour les symboles
T$	= Type de graphique (1 pour points, 2 pour segments, 3 pour symboles)
XO,YO	= Coordonnées de l'origine des axes.

Dans le cas du graphique d'une fonction variant entre 0,03 et 0,07 on introduit le facteur $K = 10^2$. Naturellement, sur le graphique il faut indiquer la valeur K utilisée ou, mieux encore, la valeur par laquelle il faut multiplier les coordonnées.

Le contrôle et la correction des erreurs sont nécessaires quand la fonction prend des valeurs indéterminées.

Ainsi, la fonction :

$$Y = \frac{1}{X-3}$$

prendra une valeur infinie au point X = 3. Un programme correct signalera cette anomalie, en évitant la valeur de l'abscisse 3. Sans ces contrôles, il résulterait une erreur de dépassement avec, pour conséquence, un arrêt de programme.

Les symboles définis par l'usager constituent un élément qui, s'il n'est pas indispensable, n'en est pas moins utile pour obtenir une présentation graphique agréable.

Dans le programme que nous présentons dans ces pages, on a prévu 3 systèmes de visualisation :

— par points
— par segments
— par symboles

Dans le premier cas, chaque point de la fonction est représenté par un point écran, et le graphique comme une succession de points plus ou moins rapprochés.

Dans la deuxième forme, chaque point est uni à celui qui le précède et qui le suit par deux segments. L'évolution de la courbe se présente alors comme une ligne brisée.

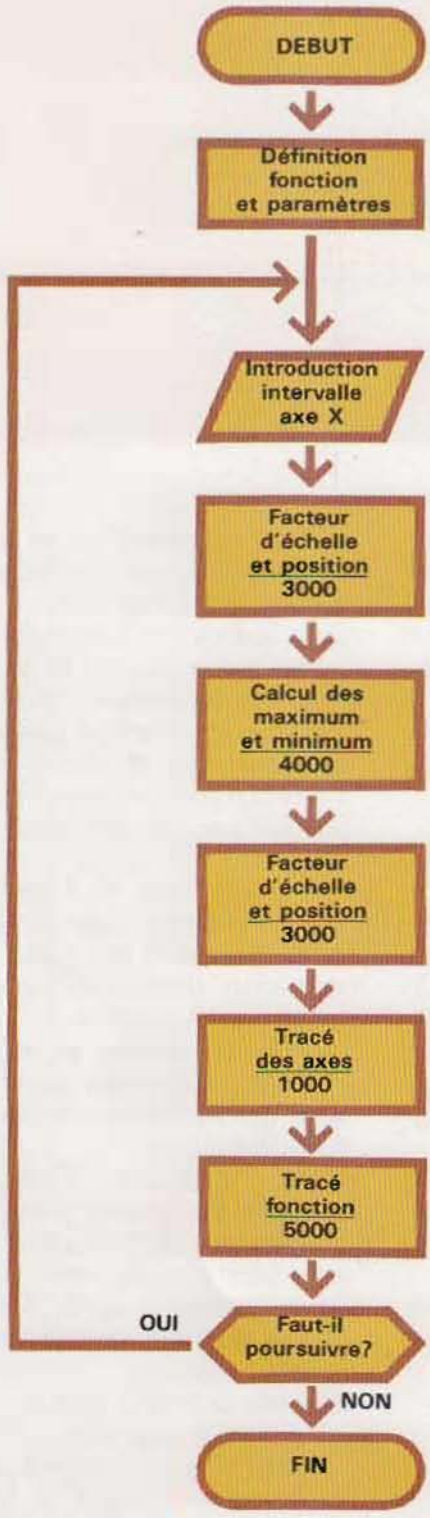
Le dernier type de représentation s'obtient en traçant un petit carré sur le point écran correspondant aux coordonnées de chaque point de la courbe.

Le premier bloc de l'organigramme (page suivante) décrit les paramètres caractéristiques de la machine et de l'application (dimensions de l'écran, facteurs d'échelle...) ainsi que la fonction à représenter. La variation de cette dernière est réalisée en remplaçant la ligne contenant la définition (ligne 100 DEF FN... A noter que ce procédé n'est autorisé qu'avec le Basic interprété).

Ensuite, le programme demande l'intervalle XM, XN des valeurs prises par la variable indépendante ; à partir de ces valeurs, il calcule le facteur d'échelle sur l'axe X (sous-programme 3000). Dans le champ des valeurs XN à XM, on effectue la recherche de la valeur maximale et minimale de la variable dépendante (4000), puis le calcul du facteur d'échelle correspondant.

Les deux dernières fonctions concernent le tracé des axes (1000) et du graphique (5000) avec les symboles sélectionnés.

ORGANIGRAMME DU PROGRAMME DE REPRESENTATION GRAPHIQUE D'UNE FONCTION



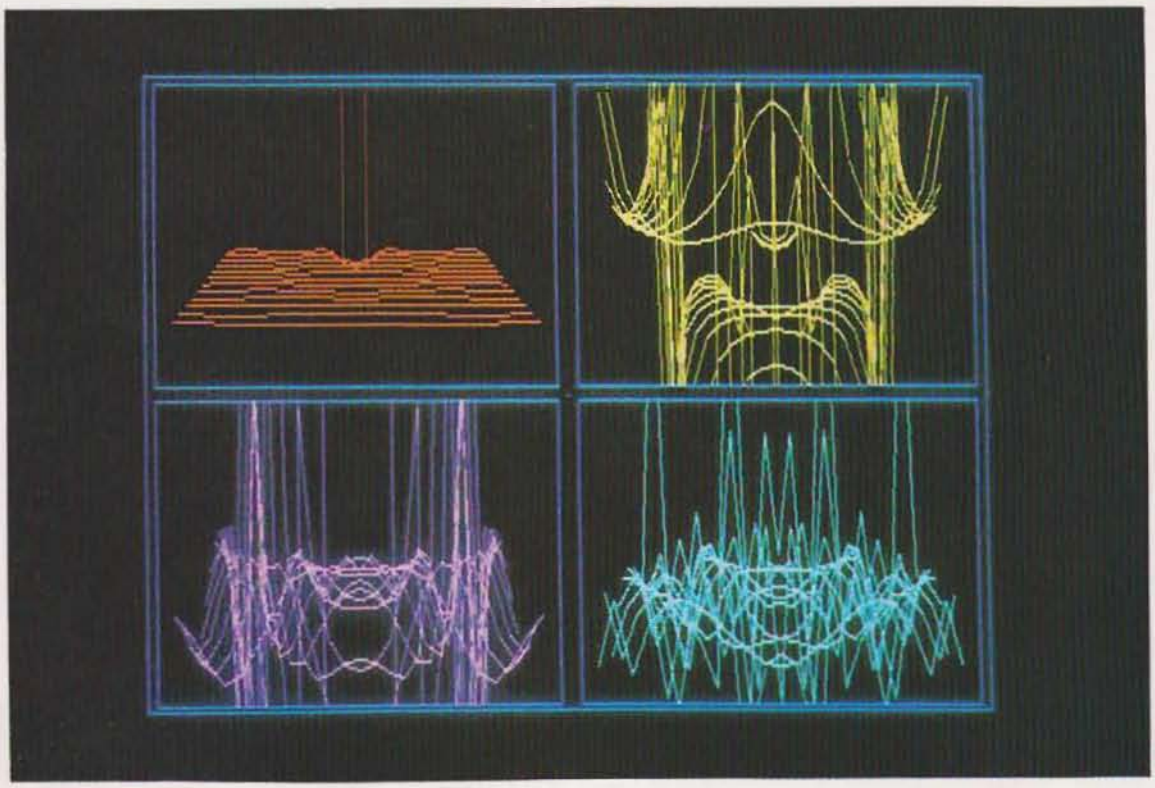
Calcul et arrondi du facteur d'échelle. Le sous-programme de calcul du facteur d'échelle (voir organigramme page suivante) utilise l'indicateur K$ pour sélectionner l'axe (X ou Y) à traiter. Il fournit, en sortie le facteur d'échelle de chacun des axes (FX, FY), les coordonnées X0, Y0 de l'origine et le facteur de multiplication K. Dans ce sous-programme, les valeurs maximale et minimale sont également arrondies en nombres entiers.
L'arrondi s'effectue par appel du sous-programme 6000.

Recherche des valeurs extrêmes de Y (sous-programme 4000) (voir p. 1468).
Le traitement consiste à calculer la valeur de Y en différents points de l'intervalle et à mémoriser, à chaque fois, la valeur la plus élevée et la valeur la plus basse.

Présentation du graphique (sous-programme 5000 et 7000). Elle est obtenue selon l'un des trois modes prévus (points, segments, symboles).
La présentation par points est du ressort du sous-programme 7000, alors que le mode segments ou symboles est exécuté par le sous-programme 5000 (l'organigramme de la page 1469 illustre cette dernière solution).
Le sous-programme 7000 est identique, à l'exclusion de l'appel à la ligne 5900, qui sert à tracer le symbole (voir page 1470).

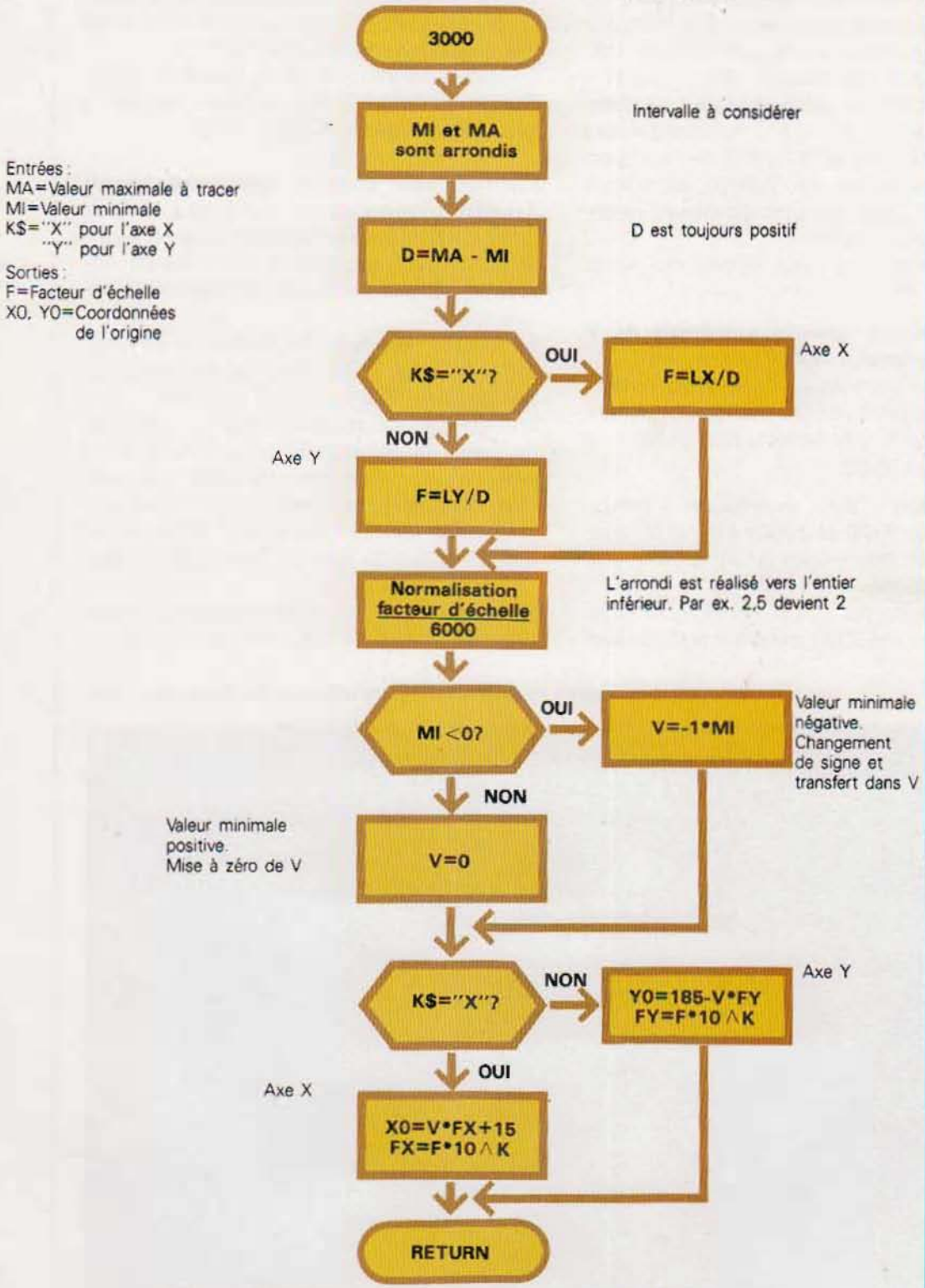
Gestion des erreurs (sous-programme 10000). L'organigramme des pages 1471 et 1472 décrit le sous-programme de gestion des erreurs. C'est la seule partie dépendant étroitement de la machine et donc difficilement généralisable.
La logique employée est propre au Personal Kid, au Siprel 2010, ainsi qu'aux Apple et compatibles. Pour les autres machines, il faut changer certains codes d'erreur, les positions de mémoire et, parfois, adopter une autre logique. Lorsqu'une erreur se produit, le code numérique correspondant est écrit dans une position de mémoire particulière (222), tandis que le numéro de ligne de l'erreur se trouve à une autre position.
Si l'on appelle ER la variable qui devra contenir le code d'erreur et LOC celle qui contiendra

Images graphiques tridimentionnelles utilisant les fenêtres vidéo.



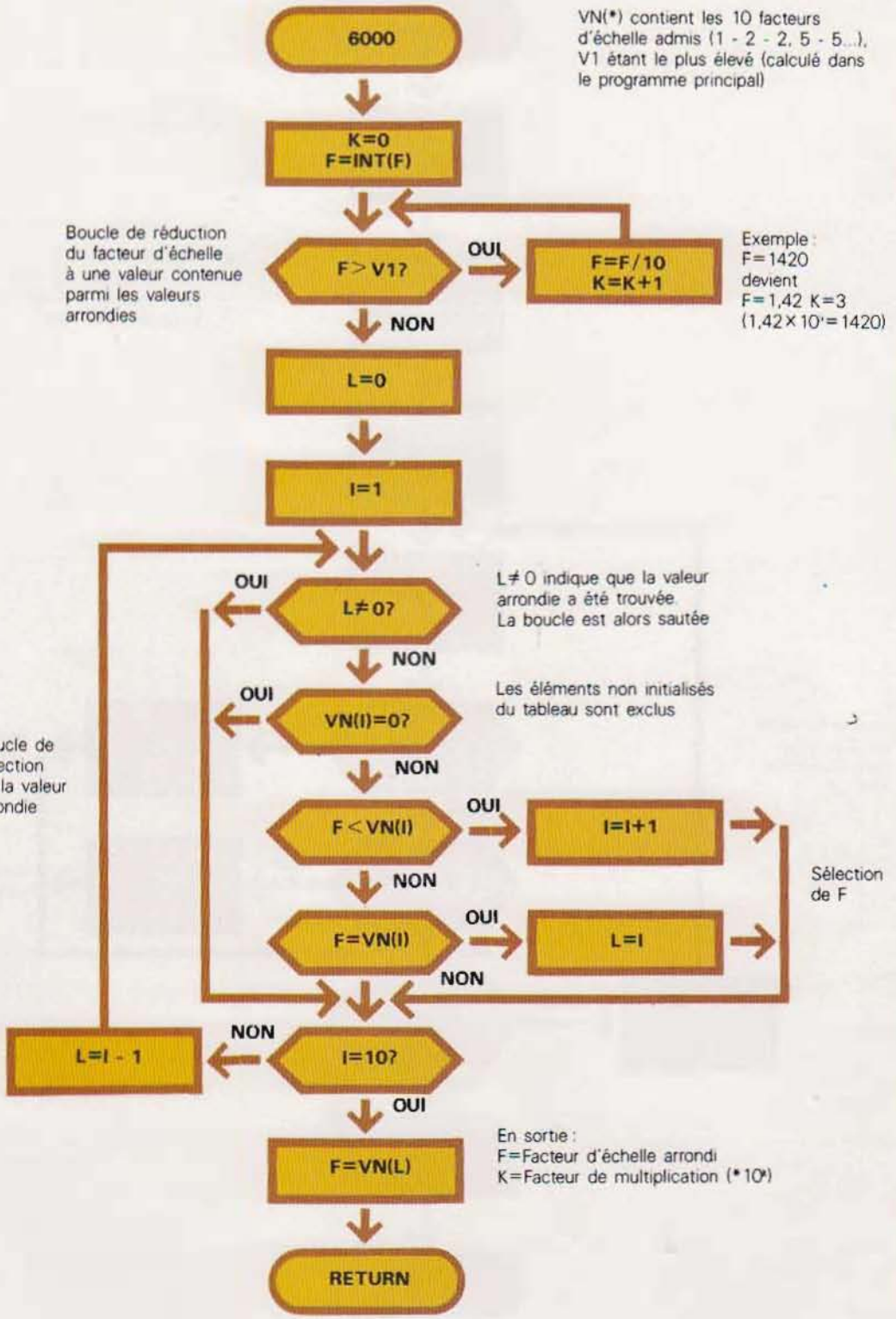
Marka

CALCUL ET ARRONDI DU FACTEUR D'ECHELLE

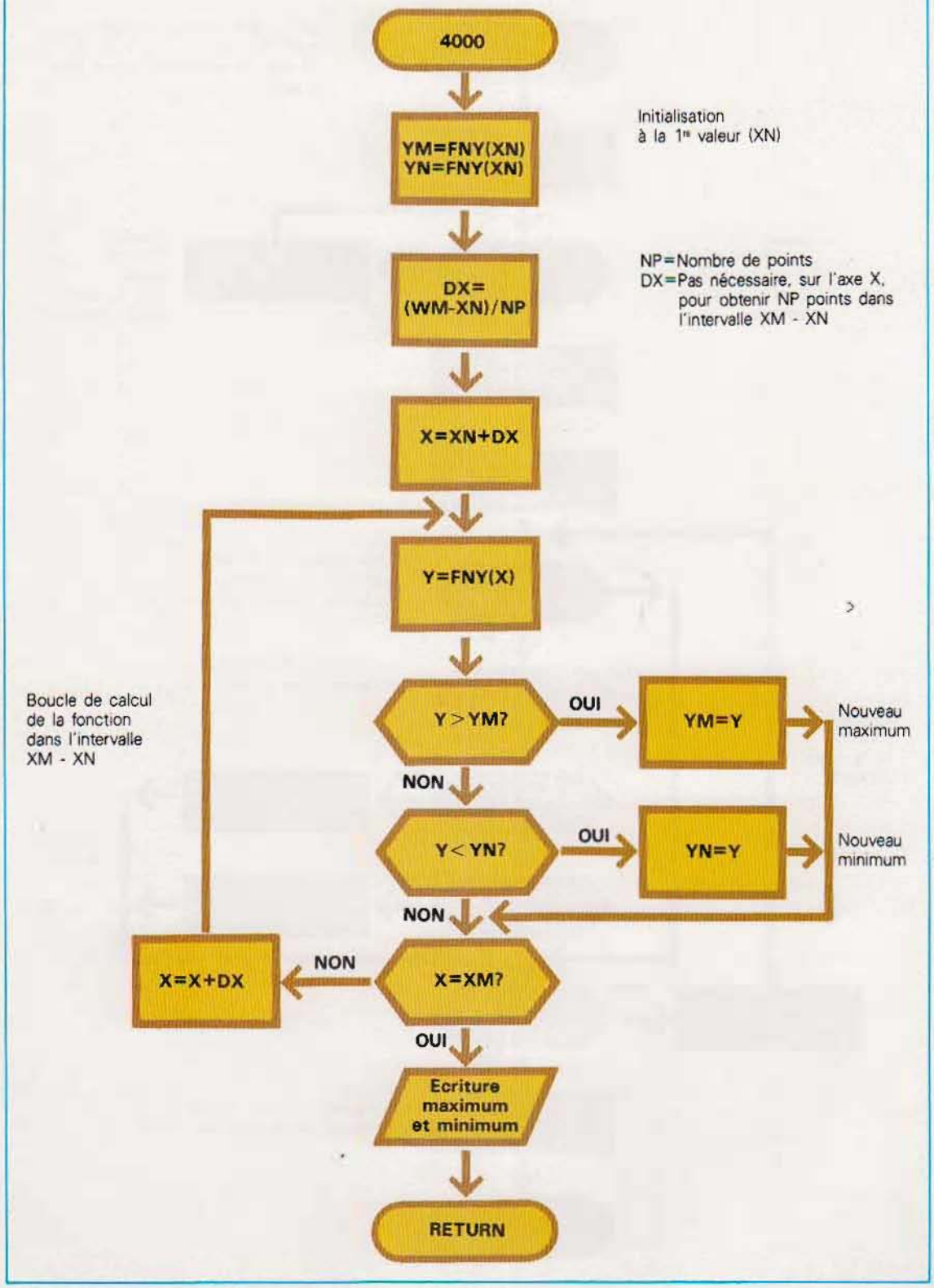


Ce sous-programme est utilisé 2 fois, une pour chaque axe.
La sélection se fait à l'aide de l'indicateur K$

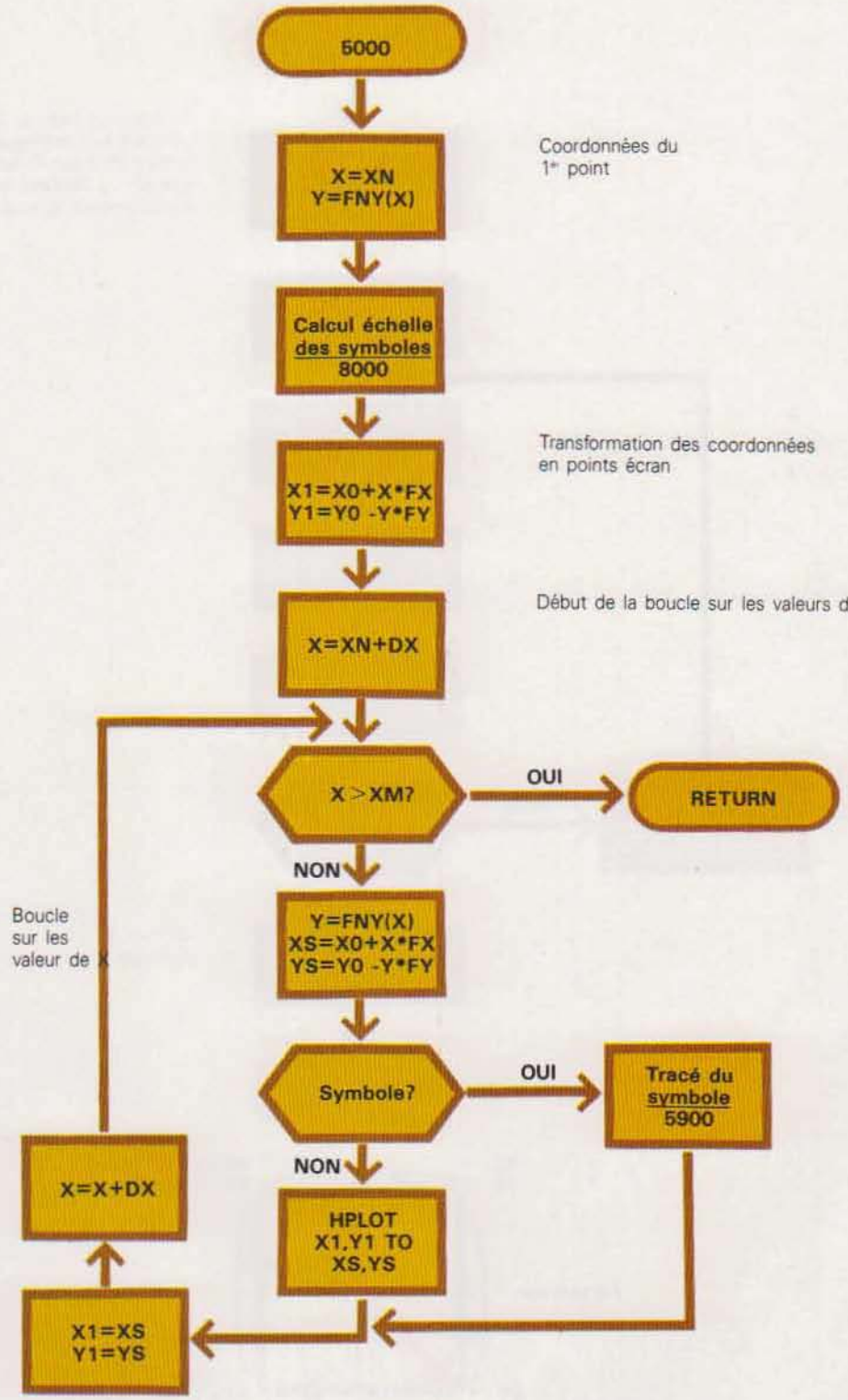
ARRONDI DU FACTEUR D'ECHELLE



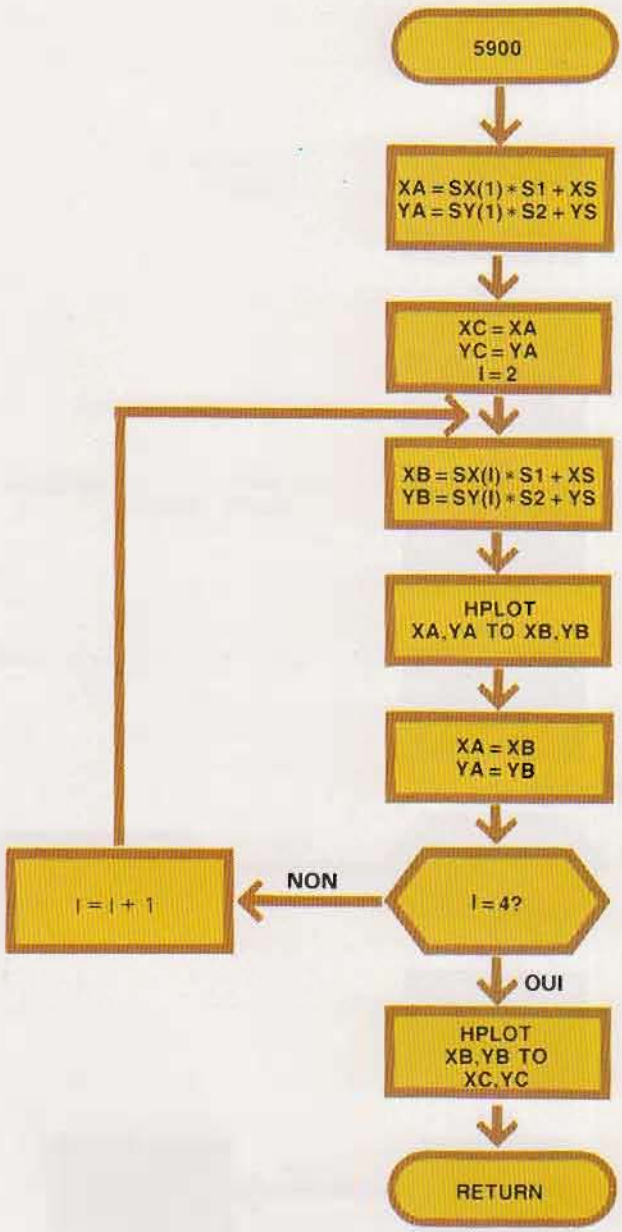
RECHERCHE DU MAXIMUM ET DU MINIMUM



PRESENTATION DE LA COURBE PAR SEGMENTS OU PAR SYMBOLES

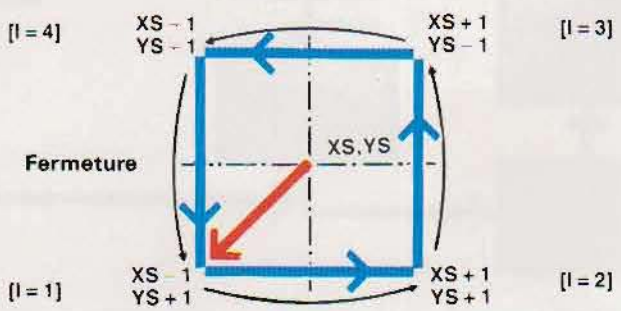


TRACE D'UN SYMBOLE CENTRE EN XS, YS

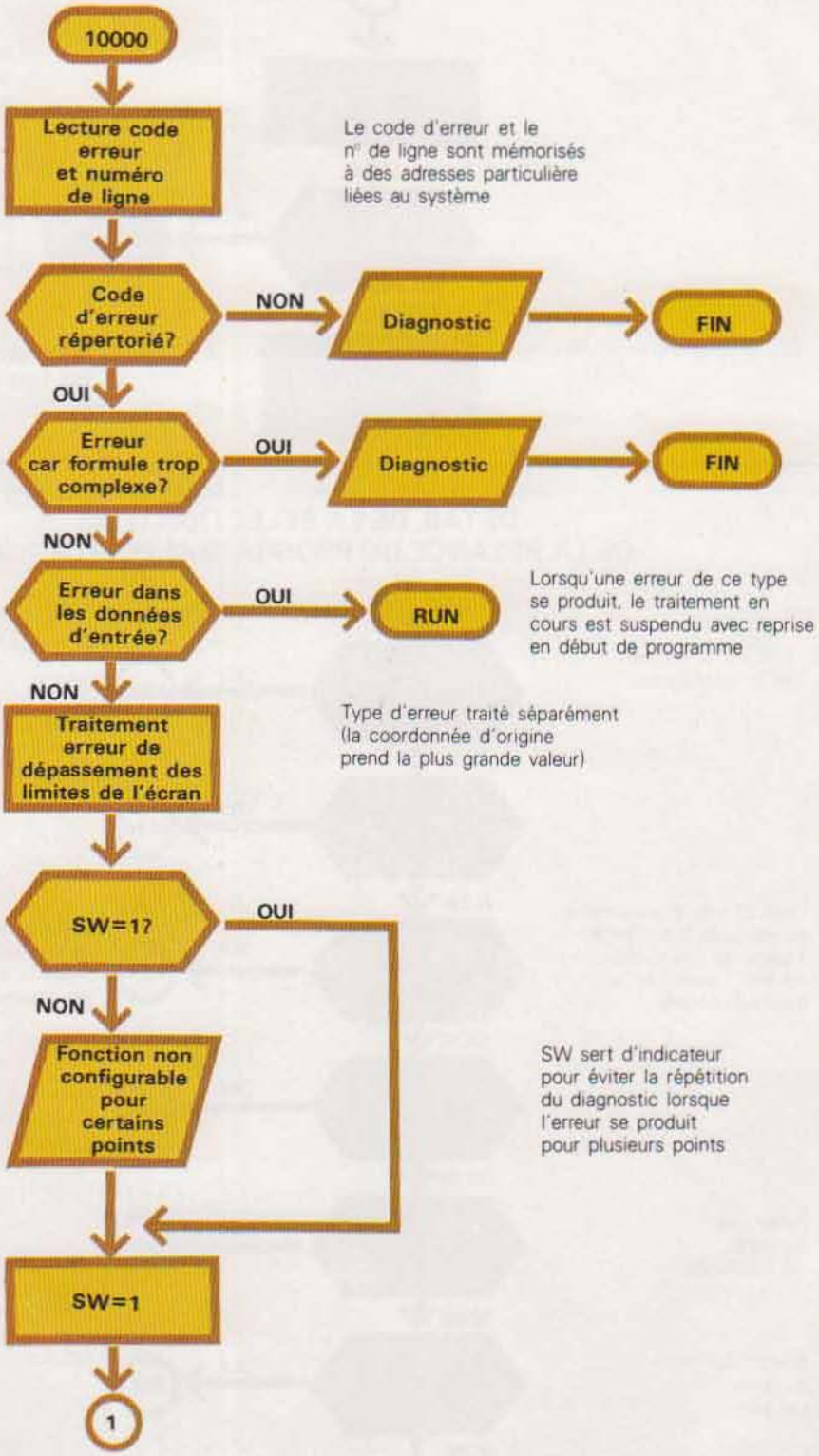


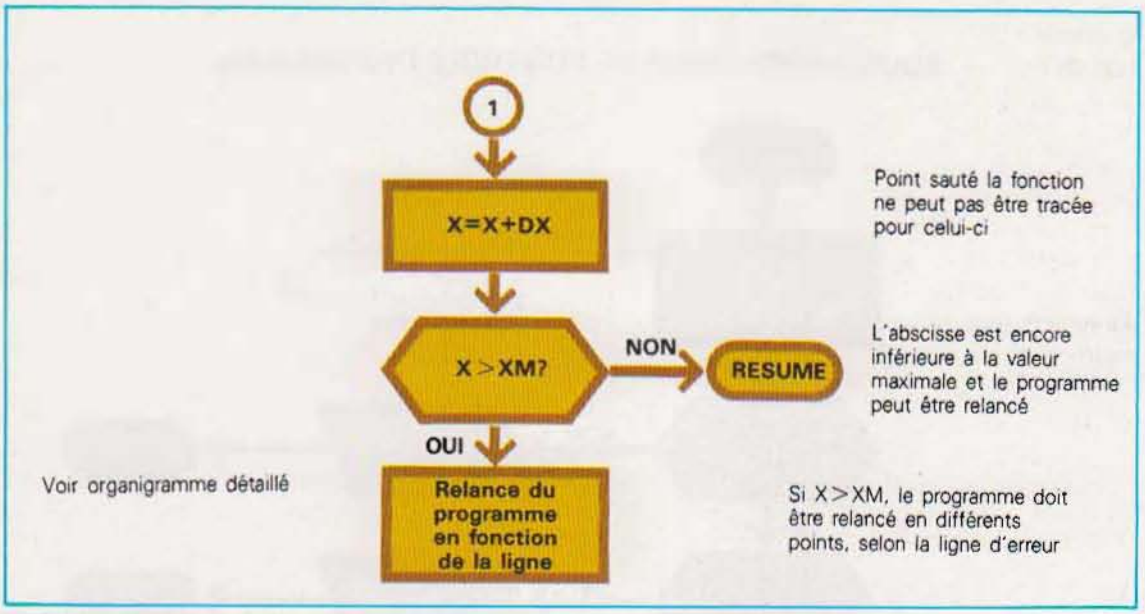
1er point. Le tableau SX ( ) contient le déplacement le long de l'axe X, tandis que SY ( ) contient les déplacements le long de Y

Clôture du symbole

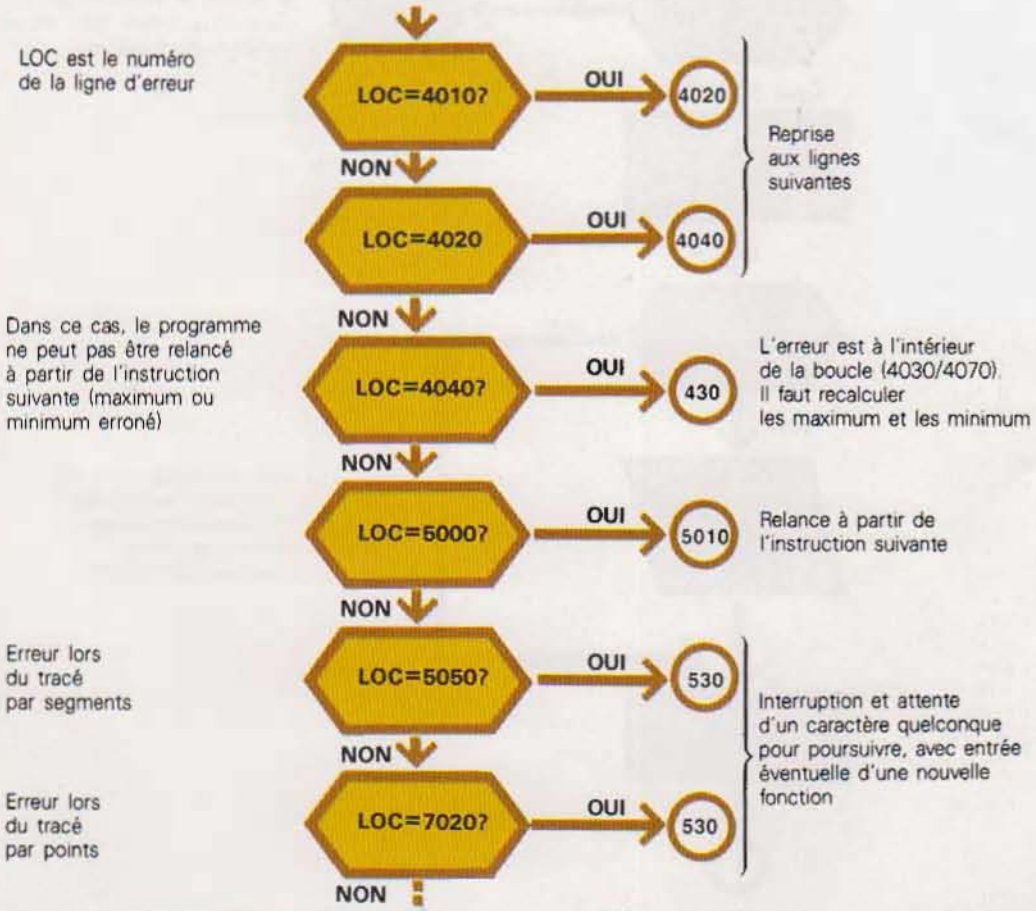


SOUS-PROGRAMME DE CONTROLE DES ERREURS





DETAIL DE LA SELECTION LORS DE LA RELANCE DU PROGRAMME POUR X > XM



le numéro de ligne, les instructions d'acquisition de ces valeurs seront :

```
ER=PEEK (222)
LOC=PEEK (218 )+PEEK (219)*256
```

La première est immédiatement compréhensible : PEEK extrait le contenu de l'adresse mémoire 222 et l'affecte à la variable ER.
Dans la seconde, la donnée (le numéro de ligne) doit être extraite de deux adresses mémoire successives (218 et 219) pour être reconstituée. En effet, l'adresse ne contient que 8 bits alors qu'il en faut 16 pour l'adressage du n° de ligne. Le système divise donc ce numéro en deux parties qu'il range à deux adresses consécutives.

La multiplication par 256 est nécessaire pour donner à la première partie le juste poids par rapport à la seconde (déplacer de 8 bits signifie multiplier par 256).
Les codes d'erreur reconnus, qui ne sont valables que pour la machine spécifiée, sont les suivants :

53=Quantité incorrecte
133=Division par zéro
254=Erreur d'entrée
69=Débordement
191=Formule trop complexe.

Pour ce qui concerne les autres machines, si la gestion des erreurs est proche de celle décrite, il convient de vérifier la correspondance des codes et, éventuellement, de les modifier.

TRACE D'UNE FONCTION

```
 90 REM ----------------------------
 91 REM FONCTIONS MATHEMATIQUES
 92 REM ----------------------------
 93 REM
 94 REM
 95 REM
 96 REM
 97 REM ----------------
 98 REM INITIALISATION
 99 REM ----------------
100 DEF FNY(X)=X*SIN(X)
110 LX=250
    : LY=180
    : SW=0
120 FOR J=1 TO 4
    : READ SX(J),SY(J)
    : NEXT J
130 DATA -1,1,1,1,1,-1,-1,-1
140 ONERR GOTO 10000
150 V1=1
160 FOR J=1 TO 10
    : READ VN(J)
    : IF NV(J)>1 THEN V1=VN(J)
170 NEXT J
180 DATA 1,2,3,4,5,6,7,8,9,0
190 REM ------
200 REM MENU
210 REM ------
220 TEXT
    : HOME
    : GOSUB 2000
230 HTAB 3
    : VTAB 5
    : INPUT "INTERVALLE AXE X? " ; XN,XM
    : HTAB 39
    : VTAB 5
    : PRINT "*"
240 IF XM<=XN THEN 220
250 IF XM-XN>250 THEN 220
260 HTAB 3
    : VTAB 10
    : INPUT "NOMBRE DE POINTS? " ; NP
    : HTAB 39
    : VTAB 10
    : PRINT "*"
```

```
270 HTAB 3
    : VTAB 15
    : PRINT "TYPE DE GRAPHIQUE? "
280 HTAB 3
    : PRINT "1) PAR POINTS"
290 HTAB 3
    : PRINT "2) PAR SEGMENTS"
300 HTAB 3
    : PRINT "3) PAR SYMBOLES"
310 VTAB 15
    : HTAB 30
    : INPUT " " ; T$
    : VTAB 15
    : HTAB 39
    : PRINT "*"
320 GOSUB 9000
330 REM ----------------------
340 REM PROGRAMME PRINCIPAL
350 REM ----------------------
360 IF XN<0 AND XM<=0 THEN MI=XN
    : MA=0
370 IF XN<0 AND XM>0 THEN MI=XN
    : MA=XM
380 IF XN>=0 AND XM>0 THEN MI=0
    : MA=XM
390 IF MA>250 THEN MA=250
400 IF MI<-250 THEN MI=-250
410 K$="X"
    : GOSUB 3000
420 GOSUB 4000
430 IF YN<0 AND YM<=0 THEN MI=YN
    : MA=0
440 IF YN<0 AND YM>0 THEN MI=YN
    : MA=YM
450 IF YN>=0 AND YM>0 THEN MI=0
    : MA=YM
460 IF MA>250 THEN MA=250
470 IF MI<-250 THEN MI=-250
480 K$="Y"
    : GOSUB 3000
490 HGR2
    : HCOLOR=3
500 GOSUB 1000
510 IF T$="1" THEN GOSUB 7000
    : GOTO 530
520 GOSUB 5000
530 PRINT CHR$(7)
    : GET A$
540 TEXT
    : HOME
    : VTAB 10
    : INPUT "VOULEZ-VOUS CHANGER DE FONCTION? (O/N)" ; A$
550 IF LEFT$(A$,1)="N" THEN RUN
560 PRINT
570 INVERSE
    : PRINT "REMPLACER LA FONCTION DE LA LIGNE 100 ET FAIRE <RUN>"
    : NORMAL
580 LIST 100
    : END
997 REM ------------------
998 REM DESSIN DES AXES
```

```
999 REM ------------------
1000 HPLOT 0,0 TO 279,0 TO 279,191 TO 0,191 TO 0,0
1010 IF D>191 THEN Y0=96
1020 HPLOT 15,Y0 TO 265,Y0
     : HPLOT X0,185 TO X0,5
1030 N2=5
     : FOR N1=2 TO 0 STEP -1
     : HPLOT X0-N1,N2 TO X0+N1,N2
     : N2=N2-1
     : NEXT N1
1040 N2=265
     : FOR N1=2 TO 0 STEP -1
     : HPLOT N2,Y0-N1 TO N2,Y0+N1
     : N2=N2+1
     : NEXT N1
1050 HPLOT 271,Y0-2 TO 276,Y0+3
     : HPLOT 276,Y0-2 TO 271,Y0+3
1060 HPLOT X0-10,3 TO X0-7,6
     : HPLOT X0-4,3 TO X0-10,8
1070 IF FX=1 THEN 1140
1080 FOR G=X0 TO 15 STEP -FX
1090 HPLOT G,Y0+1 TO G,Y0-1
1100 NEXT G
1110 FOR G=X0 TO 256 STEP FX
1120 HPLOT G,Y0+1 TO G,Y0-1
1130 NEXT G
1140 IF FY=1 THEN 1210
1150 FOR G=Y0 TO 5 STEP -FY
1160 HPLOT X0+1,G TO X0-1,G
1170 NEXT G
1180 FOR G=Y0 TO 185 STEP FY
1190 HPLOT X0+1,G TO X0-1,G
1200 NEXT G
1210 RETURN
1997 REM ---------------
1998 REM PRESENTATION
1999 REM ---------------
2000 A$="*"
2010 FOR J=1 TO 23
2020 HTAB 1
     : VTAB J
     : PRINT A$
2030 NEXT J
2040 FOR J=2 TO 39
2050 HTAB J
     : VTAB 23
     : PRINT A$
2060 NEXT J
2070 FOR J=23 TO 1 STEP -1
2080 HTAB 39
     : VTAB J
     : PRINT A$
2090 NEXT J
2100 FOR J=39 TO 2 STEP -1
2110 HTAB J
     : VTAB 1
     : PRINT A$
2120 NETX J
2130 VTAB 1
     : HTAB 10
     : INVERSE
     : PRINT "FONCTIONS MATHEMATIQUES"
     : NORMAL
2140 RETURN
2997 REM ---------
2998 REM ARRONDI
2999 REM ---------
3000 MA%=INT(MA)
     : MI%=INT(MI)
```

```
3010 IF MA>0 AND MA%<>(MA) THEN MA=MA%+1
     : GOTO 3030
3020 MA=MA%
3030 IF MI>0 AND MI%<>MI THEN MI=MI%+1
     : GOTO 3050
3040 MI=MI%
3047 REM --------------------------------
3048 REM CALCUL DU FACTEUR D'ECHELLE
3049 REM --------------------------------
3050 D=MA-MI
3060 IF K$="X" THEN F=LX/D
     : GOTO 3080
3070 F=LY/D
3080 GOSUB 6000
     : REM ARRONDI DU FACTEUR D'ECHELLE
3090 IF MI<0 THEN V=-MI
     : GOTO 3110
3100 V=0
3110 IF K$="X" THEN FX=F*10^K
     : X0=U*FX+15
     : RETURN
3120 FY=F*10^K
     : Y0=185-V*FY
     : RETURN
3997 REM ------------------------
3998 REM MAXIMUM ET MINIMUM
3999 REM ------------------------
4000 DX=(XM-XN)/NP
     : X=XN
4010 YN=FNY(X)
4020 YM=FNY(X)
4030 FOR X=XN+DX TO XM STEP DX
4040 Y=FNY(X)
4050 IF Y>YM THEN YM=Y
4060 IF Y<YN THEN YN=Y
4070 NEXT X
4080 REM ECRITURE DU MAXIMUM ET DU MINIMUM
4090 RETURN
4997 REM ----------------------------
4998 REM GRAPHIQUE PAR SEGMENTS
4999 REM ----------------------------
5000 X=XN
     : Y=FNY(X)
5010 GOSUB 8000
5020 X1=X0+X*FX
5030 Y1=Y0-Y*FY
5040 FOR X=XN+DX TO XM STEP DX
5050 Y=FNY(X)
5060 XS=X0+X*FX
5070 YS=Y0-Y*FY
5080 IF T$="3" THEN GOSUB 5900
     : GOTO 5100
5090 HPLOT X1,Y1 TO XS,YS
5100 X1=XS
     : Y1=YS
5110 NEXT X
5120 RETURN
5897 REM ------------------------
5898 REM DESSIN DES SYMBOLES
5899 REM ------------------------
5900 XA=SX(1)*S1+XS
     : YA=SY(1)*S2+YS
5910 XC=XA
     : YC=YA
5920 FOR I=2 TO 4
5930 XB=SX(I)*S1+XS
     : YB=SY(I)*S2+YS
```

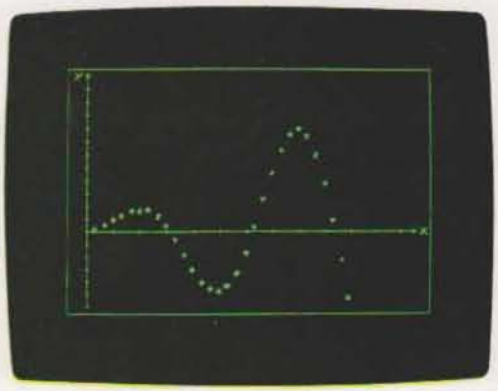
```
5940 HPLOT XA,YA TO XB,YB
5950 XA=XB
     : YA=YB
5960 NEXT I
5980 HPLOT XB,YB TO XC,YC
5990 RETURN
5997 REM ---------
5998 REM ARRONDI
5999 REM ---------
6000 K=0
     : F=INT(F)
6010 IF F>V1 THEN F=F/10
     : K=K+1
     : GOTO 6010
6020 L=0
6030 FOR I=1 TO 10
6040 IF L<>0 OR VM(I)=0 THEN 6070
6050 IF F<VM(I) THEN L=I-1
6060 IF F=VM(I) THEN L=I
6070 NEXT I
6080 F=VM(L)
6090 RETURN
6997 REM ------------------------
6998 REM GRAPHIQUE PAR POINTS
6999 REM ------------------------
7000 GOSUB 8000
7010 FOR X=XM TO XM STEP DX
7020 Y=FNY(X)
7030 XS=X0+X*FX
7040 YS=Y0-Y*FY
7050 HPLOT XS,YS
7060 NEXT X
7070 RETURN
7997 REM -------------------------
7998 REM ECHELLE DES SYMBOLES
7999 REM -------------------------
8000 S1=INT(FX/4)
     : IF S1<1 THEN S1=1
8010 S2=INT(FY/4)
     : IF S2<1 THEN S2=1
8020 IF S1<S2 THEN S2=S1
     : RETURN
8030 S1=S2
8040 RETURN
8997 REM ------
8998 REM PAUSE
8999 REM ------
9000 HOME
     : HTAB 12
     : VTAB 12
     : FLASH
     : PRINT "UN INSTANT, SVP"
     : NORMAL
     : RETURN
9997 REM -------------------------
9998 REM GESTION DES ERREURS
9999 REM -------------------------
10000 ER=PEEK(222)
     : LOC=PEEK(218)+PEEK(219)*256
10010 IF ER<>53 AND ER<>133 AND ER<>254 AND ER<>69 AND ER<>191 THEN HOME
     : PRINT "ERREUR N° " ; ER
     : END
10020 IF ER=191 THEN HOME
     : HTAB 12
     : VTAB 12
     : INVERSE
```

```
      : PRINT "FORMULE TROP COMPLEXE"
      : NORMAL
      : STOP
10030 IF ER=254 THEN RUN
10040 REM GRANDEUR INCORRECTE
10050 IF XS>279 THEN XS=279
      : RESUME
10060 IF XS<0 THEN XS=0
      : RESUME
10070 IF YS>191 THEN YS=191
      : RESUME
10080 IF YS<0 THEN YS=0
      : RESUME
10090 IF XA>279 THEN XA=279
      : RESUME
10100 IF XA<0 THEN XA=0
      : RESUME
10110 IF YA>191 THEN YA=191
      : RESUME
10120 IF YA<0 THEN YA=0
      : RESUME
10130 IF XB>279 THEN XB=279
      : RESUME
10140 IF XB<0 THEN XB=0
      : RESUME
10150 IF YB>191 THEN YB=191
      : RESUME
10160 IF YB<0 THEN YB=0
      : RESUME
10170 IF X1>279 THEN X1=279
      : RESUME
10180 IF X1<0 THEN X1=0
      : RESUME
10190 IF Y1>191 THEN Y1=191
      : RESUME
10200 IF Y1<0 THEN Y1=0
      : RESUME
10210 IF XC>279 THEN XC=279
      : RESUME
10220 IF XC<0 THEN XC=0
      : RESUME
10230 IF YC>191 THEN YC=191
      : RESUME
10240 IF YC<0 THEN YC=0
      : RESUME
10250 IF SW=1 THEN 10300
10260 TEXT
      : HOME
      : VTAB 12
      : INVERSE
      : PRINT "LA FONCTION N'EST DEFINIE EN AUCUN POINT DE L'INTERVALLE"
      : NORMAL
10270 VTAB 23
      : PRINT "APPUYEZ SUR UNE TOUCHE POUR POURSUIVRE"
      : GET A$
10280 GOSUB 9000
10290 SW=1
10300 X=X+DX
      : IF X>XM THEN 10320
10310 RESUME
10320 IF LOC=4010 THEN 4020
10330 IF LOC=4020 THEN 4030
10340 IF LOC=4040 THEN 430
10350 IF LOC=5000 THEN 5010
10360 IF LOC=5050 THEN 530
10370 IF LOC=7020 THEN 530
```

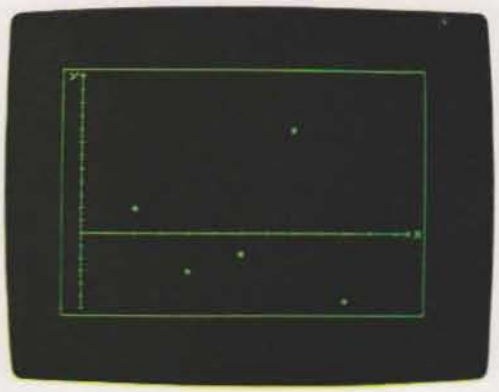
VISUALISATION DU GRAPHIQUE D'UNE FONCTION

Une fois l'exécution du programme demandée et la partie des initialisations (ligne 100 à 180) terminée, le sous-programme 2000 (ligne 220) est appelé. Celui-ci présente un cadre entouré d'astérisques et le titre en vidéo inversée. En fin de sous-programme 2000, on revient au programme principal qui présente, dans l'ordre, les différentes possibilités proposées. La photographie montre la situation finale, une fois qu'il a été répondu à toutes les questions et avant la sélection du type de graphique. Les données entrées se rapportent à la fonction X*SIN(X) (ligne 100). Les écrans suivants montrent, pour la même fonction, les différents aspects du graphique en fonction des paramètres NOMBRE DE POINTS et TYPE DE GRAPHIQUE.

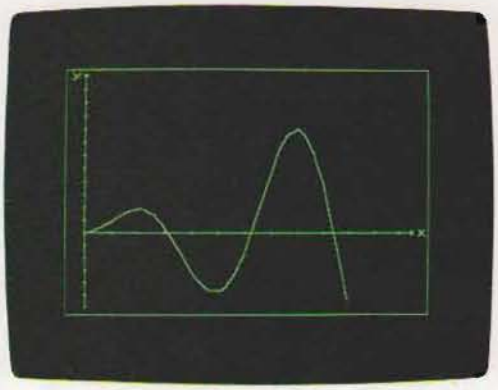
Le graphique par points (type de graphique 1) est le mode sélectionné. Le programme présente les axes cartésiens et les points correspondant aux 30 couples de coordonnées. La valeur de X est entrée au clavier tandis que Y est calculée par le programme.



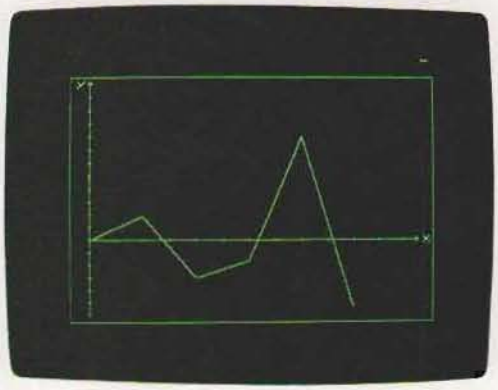
Dans ce cas précis, l'utilisateur a demandé la visualisation du graphique par points pour un nombre de valeurs de la variable X réduit (5). La courbe de la fonction est toujours la même, mais elle est moins nette.



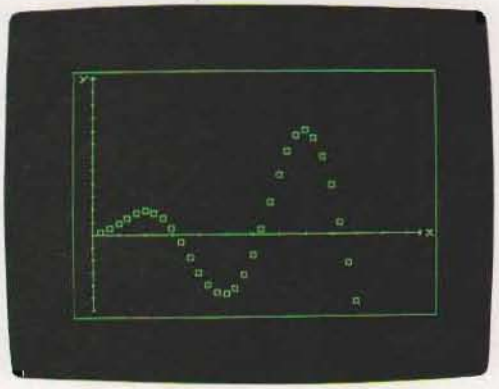
Visualisation par segments de 30 valeurs de la variable X. La définition de la courbe est excellente en raison du grand nombre de données. Le programme a tracé une série de segments joignant deux à deux les points dont les coordonnées sont connues.



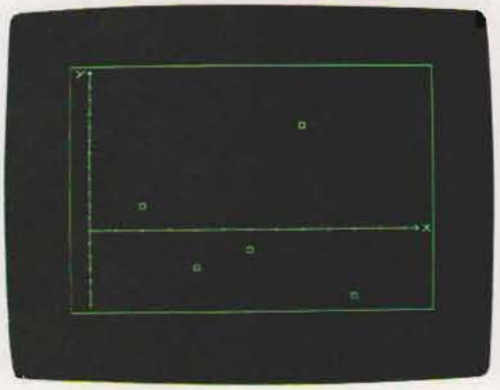
La visualisation se poursuit par segments, mais avec 5 valeurs d'entrée de X. La définition du graphique est très approximative et les segments sont visibles.



L'utilisateur a demandé la visualisation par symboles de 30 valeurs de X. Cette représentation est plus indiquée quand il s'agit de visualiser l'allure d'une grandeur sujette à des erreurs d'approximation. La grandeur du symbole utilisé peut, dans ce cas, couvrir ces erreurs.



Visualisation par symboles pour 5 valeurs de la variable indépendante. L'allure de la courbe est à peine perceptible.

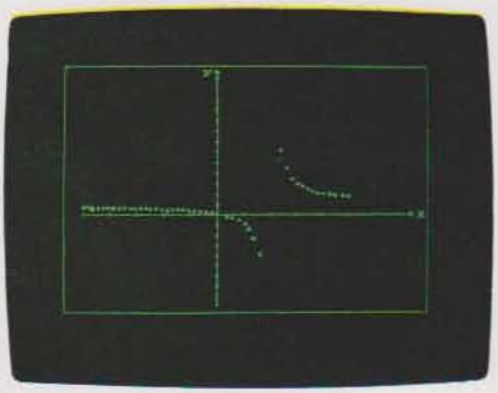


Une fois le graphique visualisé, il suffit d'activer une touche quelconque (ligne 540) pour que le programme fasse évoluer la fonction. Noter la ligne 580, qui permet de présenter le contenu de la ligne 100 et, par suite, de rappeler la fonction traitée. La ligne 580 est suivie de l'instruction END qui met fin au programme. Le système revient alors à l'état commandes (cet état est mis en évidence par l'affichage du caractère guide-utilisateur) et l'utilisateur peut changer la fonction en réécrivant la ligne 100. Pour finir, la commande RUN lance l'exécution du graphique de la nouvelle fonction X/(X-2).

Le programme présente à nouveau le cadre dans lequel il demande à l'utilisateur de luir fournir les paramètres nécessaires à l'exécution.

Une fois le type de graphique sélectionné, le programme passe à la phase de calcul, au cours de laquelle il découvre que la fonction contient un point de discontinuité pour X=2. En effet, pour cette valeur, on a Y=2/(2-2)=2/0.
Le quotient 2/0 donne un nombre infini, qui ne peut être représenté dans le graphique. Le programme signale cette anomalie (il ne s'agit pas d'une erreur mais d'une caractéristique de la fonction).
Le diagnostic qui apparaît est très général et peut même parfois être inexact d'un point de vue mathématique. Ce défaut est dû à la structure du sous-programme d'erreur qui regroupe plusieurs causes en un seul message.

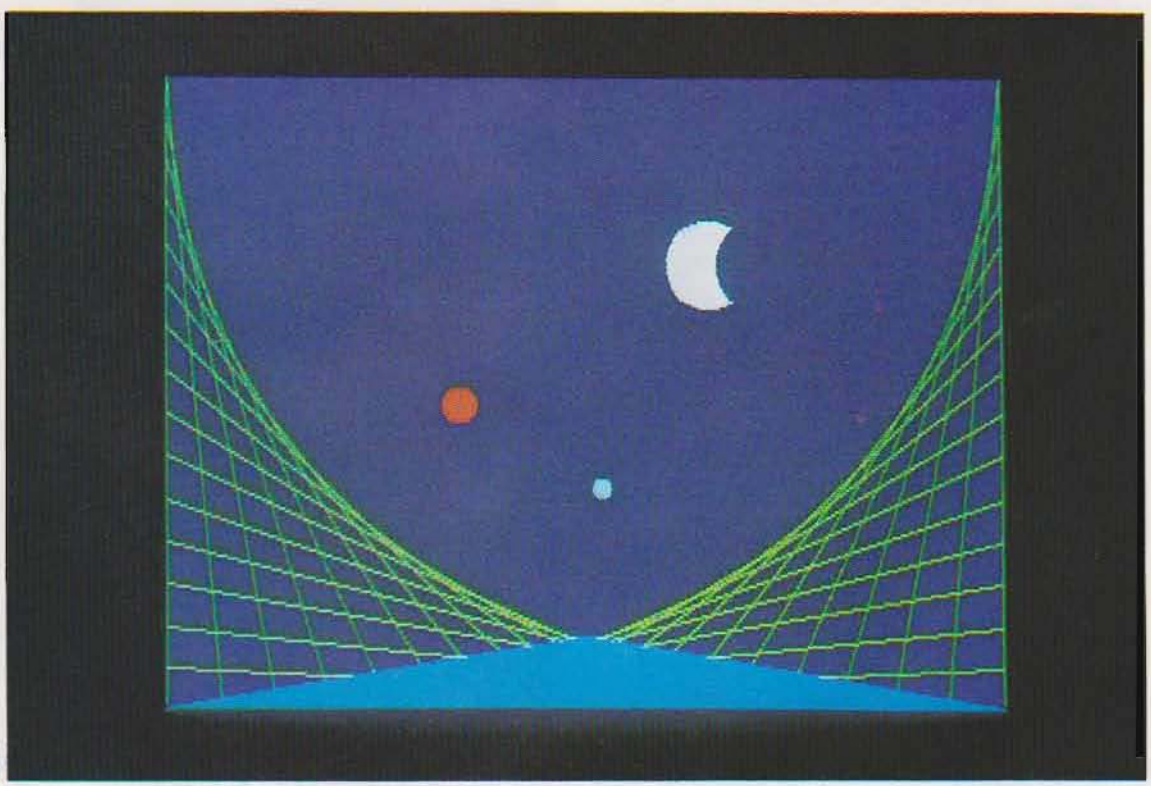
La fonction est tracée une fois que l'utilisateur a confirmé son choix en appuyant sur une touche quelconque (ligne 10250 et suivantes). La présence d'une asymptote pour X=2 (point de discontinuité de la fonction) est indiquée.
La définition de la courbe est améliorée en demandant sa visualisation par segments.



Visualisation des caractères en mode graphique

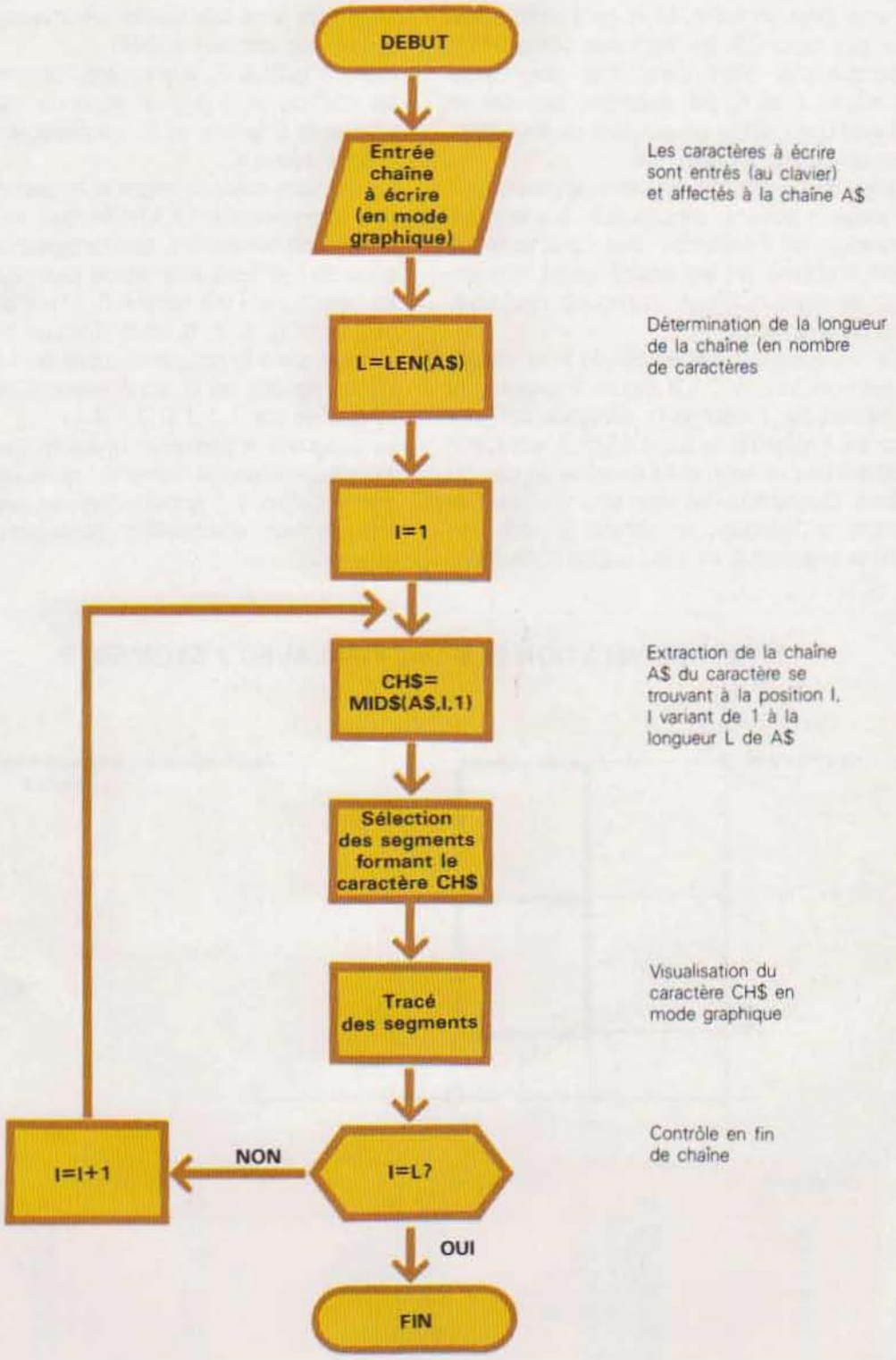
Les applications graphiques demandent souvent, elles aussi, l'insertion de caractères alphabétiques ou numériques dans l'image vidéo. C'est le cas, par exemple, des légendes explicatives associées à un graphique.
Si le moniteur haute résolution est conçu pour gérer séparément chaque point de l'écran, il n'est plus possible de visualiser les caractères ASCII standard utilisés en mode texte. En effet, dans ce dernier mode, l'écran est géré par matrices de caractères : il est donc possible d'y placer un caractère par « case » se trouvant au croisement d'une ligne et d'une colonne. Une case est constituée d'une matrice de points écran qui sont automatiquement activés par le système, en fonction du caractère à visualiser.
En mode graphique, la correspondance entre le caractère et la configuration de points n'existe plus, précisément pour autoriser cette gestion individuelle de tous les points écran.
Le terminal vidéo visualisera donc exclusivement des textes (mode texte) ou des graphiques (mode graphique), sans possibilité de coexistence. Toutefois, une forme de mixité a été mise au point dans certaines machines, réservant quelques lignes (généralement en bas de l'écran) au mode texte, le reste étant affecté au mode graphique.
La visualisation d'un caractère alphabétique ou numérique dans un page graphique nécessite une construction logicielle point par point de ce caractère. La plupart des ordinateurs personnels et des micros ne sont pas dotés de sous-programmes adéquats, laissés au bon soin de l'utilisateur.
La technique de construction consiste à représenter les lettres et les nombres sous des formes schématisables au moyen des déplacements élémentaires d'un crayon imaginaire. Pour obtenir un « tracé » quelconque en mode graphique, il faut lancer le programme correspondant lettre par lettre, ou nombre par nombre (voir organigramme ci-contre).
Le programme ne présente aucune difficulté majeure en-dehors de la sélection des segments composant chacun des caractères. Une

Image graphique obtenue en utilisant des fonctions mathématiques.



Marka

AFFICHAGE DE CARACTERES EN MODE GRAPHIQUE



méthode consisterait à écrire un sous-programme pour chaque caractère : une routine pour écrire la lettre « a », une autre pour « b » et ainsi de suite. Mais ce système, qui n'est pas optimisé, ne tient pas compte des ressemblances entre caractères graphiques. Les lettres F et E, par exemple, peuvent se déduire l'une l'autre, en ajoutant ou en supprimant un segment horizontal.

Pour les autres caractères moins apparentés, il est toujours possible de procéder à une schématisation de l'ensemble des caractères de façon à obtenir un équivalent assez ressemblant de chacun d'eux, composé exclusivement de segments.

Cette méthode est très employée pour visualiser les nombres de 0 à 9 (figure 1 ci-dessous) au moyen de 7 segments désignés par une lettre de l'alphabet (a,b,c,d,e,f,g). L'activation sélective de ces segments visualise un des 10 chiffres. Quand tous les segments sont visibles (comme ci-dessous), on obtient 8 ; en éteignant le segment g, on a 0 ; supprimons encore les segments f, a, b et c et 1 s'affiche, et ainsi de suite.

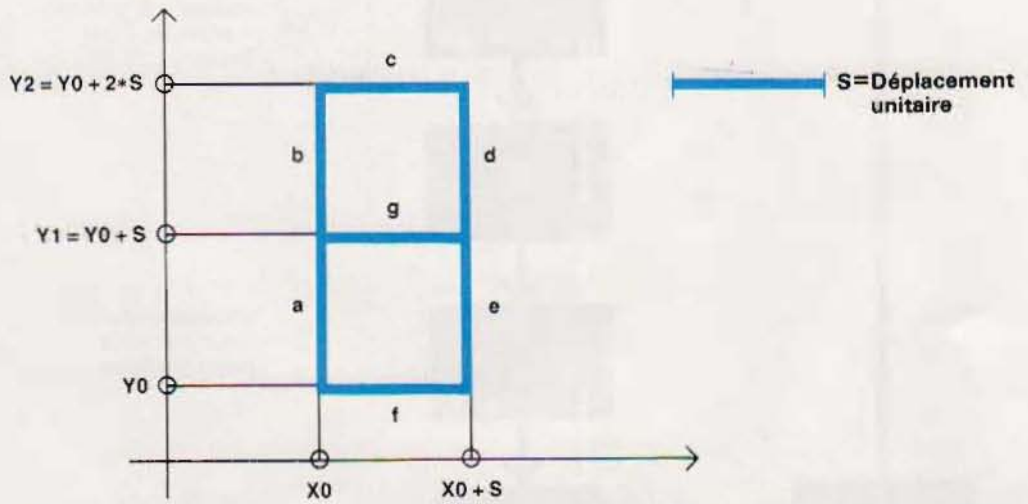
Dans les systèmes d'affichage numériques, les segments sont des diodes électroluminescentes ou des cristaux liquides.

Dans le tableau 2, la première colonne indique les chiffres de 1 à 0, la seconde montre les segments à activer et la troisième le symbole affiché (résultat).

La dernière colonne reprend la seconde, non pas selon notation OUI/NON mais en utilisant la notation binaire 0/1, qui occupe moins d'espace en mémoire et effectue plus rapidement les sélections. Pour obtenir 6, il faut activer les segments g, f, e, b, et a. Chacun d'eux est indiqué, dans le programme par un 1 (OUI), et les autres par un 0. Le nombre 6 sera ainsi symbolisé par 1 1 1 0 0 1 1.

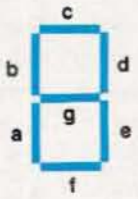
Le programme passe en revue le contenu du tableau position par position ; quand il rencontre le chiffre 1 il appelle l'un des sept sous-programmes d'activation correspondant au segment.

1/REPRESENTATION DES CHIFFRES AVEC 7 SEGMENTS



Segment	Coordonnées de	Coordonnées à	Incréments de	Incréments à
a	X0,Y0	X0,Y0 + S	0,0	0,1
b	X0,Y0 + S	X0,Y0 + 2 * S	0,1	0,2
c	X0,Y0 + 2 * S	X0 + S,Y0 + 2 * S	0,2	1,2
d	X0 + S,Y0 + 2 * S	X0 + S,Y0 + S	1,2	1,1
e	X0 + S,Y0 + S	X0 + S,Y0	1,1	1,0
f	X0 + S,Y0	X0,Y0	1,0	0,0
g	X0,Y0 + S	X0 + S,Y0 + S	0,1	1,1

2/REPRESENTATION DES CHIFFRES AVEC 7 SEGMENTS



Chiffre à représenter	Segment activé 7 g	6 f	5 e	4 d	3 c	2 b	1 a	Symbole tracé	Contenu du tableau
1	—	—	OUI	OUI	—	—	—		0011000
2	OUI	OUI	—	OUI	OUI	—	OUI		1101101
3	OUI	OUI	OUI	OUI	OUI	—	—		1111100
4	OUI	—	OUI	—	—	OUI	—		1010010
5	OUI	OUI	OUI	—	OUI	OUI	—		1110110
6	OUI	OUI	OUI	—	—	OUI	OUI		1110011
7	—	—	OUI	OUI	OUI	—	—		0011100
8	OUI	OUI	OUI	OUI	OUI	OUI	OUI		1111111
9	OUI	—	OUI	OUI	OUI	OUI	—		1011110
0	—	OUI	OUI	OUI	OUI	OUI	OUI		0111111

En réalité, il n'est pas nécessaire d'en avoir sept différents, un seul, correctement paramétré, est suffisant.
La seconde colonne du tableau 1 indique les déplacements nécessaires par rapport au point d'origine (X0, Y0) pour obtenir chacun des segments. Ainsi, le segment est tracé à l'aide du déplacement de X0, Y0 vers X0, Y0 + S (S étant la longueur de chaque segment). Ce système (déplacement unitaire S) permet deux paramétrages importants :

- Tous les déplacements — par conséquent tous les segments — sont identifiés par des coordonnées non pas absolues, mais relatives à un point initial. Pour déplacer la position de visualisation du caractère, il suffit de modifier les coordonnées du point d'origine.
- Les dimensions des caractères sont aisément modifiables au moyen du paramètre S.

Dans le cas du segment a, par exemple, la première position n'a subi aucune variation par rapport à l'origine. Autrement dit, son incrément est de 0 sur l'axe X ainsi que sur l'axe Y.

En revanche, les coordonnées finales du segment (X0 et Y0 + S) ont pour incrément S sur l'axe Y (et 0 sur X). Ce paramétrage permet d'écrire un programme généralisé capable de générer tous les segments.
Si (X1, Y1) est l'origine d'un segment et (X2, Y2) son autre extrémité, on a :

$X1 = X0 + N_X S$ (incrément sur l'axe X)
$X1 = X0 \pm N_Y S$ (incrément sur l'axe Y)

Il en est de même pour les coordonnées X2, Y2 (n'oublions pas que X0 et Y0 sont les coordonnées de référence). Dans ce cas particulier, pour les premiers incréments, on a N = 0 (origine), c'est-à-dire :

$X1 = X0$
$Y1 = Y0$

En revanche, pour le second point nous avons NX = 0 et Ny = 1, d'où :

$X2 = X0$
$Y2 = Y0 \pm S$

X1, Y1 et X2, Y2 sont les coordonnées des extrémités par rapport à l'origine (X0, Y0) prise comme référence. Le signe à utiliser dans la deuxième expression (+ ou -) dépend de l'orientation de l'axe Y. Dans ce cas des terminaux vidéo dont l'origine de l'écran est en bas, le signe est + (les Y augmentent vers le haut) et inversement.

Pour obtenir un programme généralisé, il suffit de mémoriser les incréments (voir figure 1 page 1484) de chaque segment et les segments (figure 2) composant chaque nombre. L'organigramme ci-contre illustre un programme de visualisation d'un nombre aux dimensions désirées (grâce au paramètre S), situé à n'importe quel endroit de l'écran (en faisant varier l'origine X0, Y0).

Les incréments nécessaires pour obtenir un segment sont mémorisés dans 4 tableaux :

XS(I),YD(I)=coordonnées du point initial d'un segment

XA(I),YA(I)=coordonnées du point final d'un segment

Les valeurs sont affectées à la ligne 60 à l'aide des instructions DATA des lignes 910 à 916 (comparer les données à celles du tableau 1, page 1484).

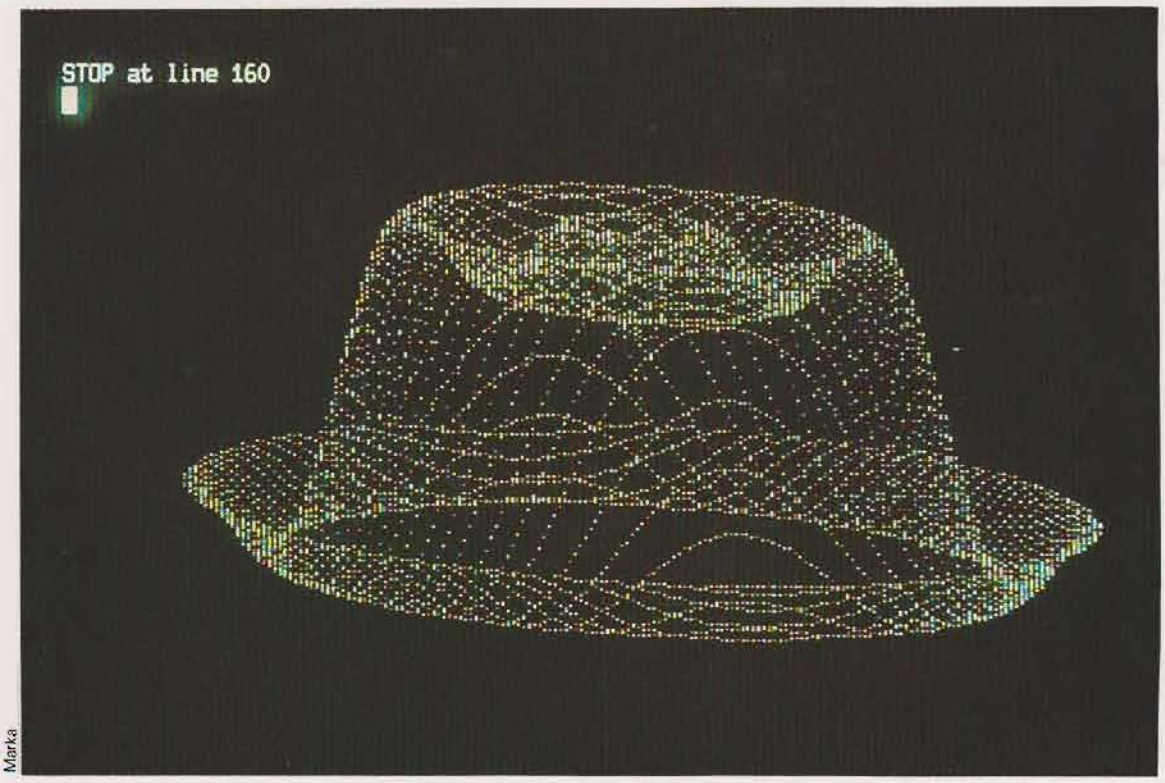
Les segments nécessaires à la visualisation des dix chiffres sont rangés dans le tableau à deux dimensions C% (10,7) dont les cases sont remplies au moyen de l'instruction READ (ligne 76) et des instructions DATA (lignes 918 à 927).

La première dimension (10) indique un des dix chiffres représentables (0,1,2...) tandis que le deuxième (7) représente les segments qui sont à activer.

Le programme utilise un tableau intermédiaire (V, de dimension 7), dans lequel sont transférés les éléments de C% correspondant aux segments à représenter (ligne 116) ; c'est donc le contenu de ce tableau qui est contrôlé (sous-programme 1000, figure inférieure de la page 1488).

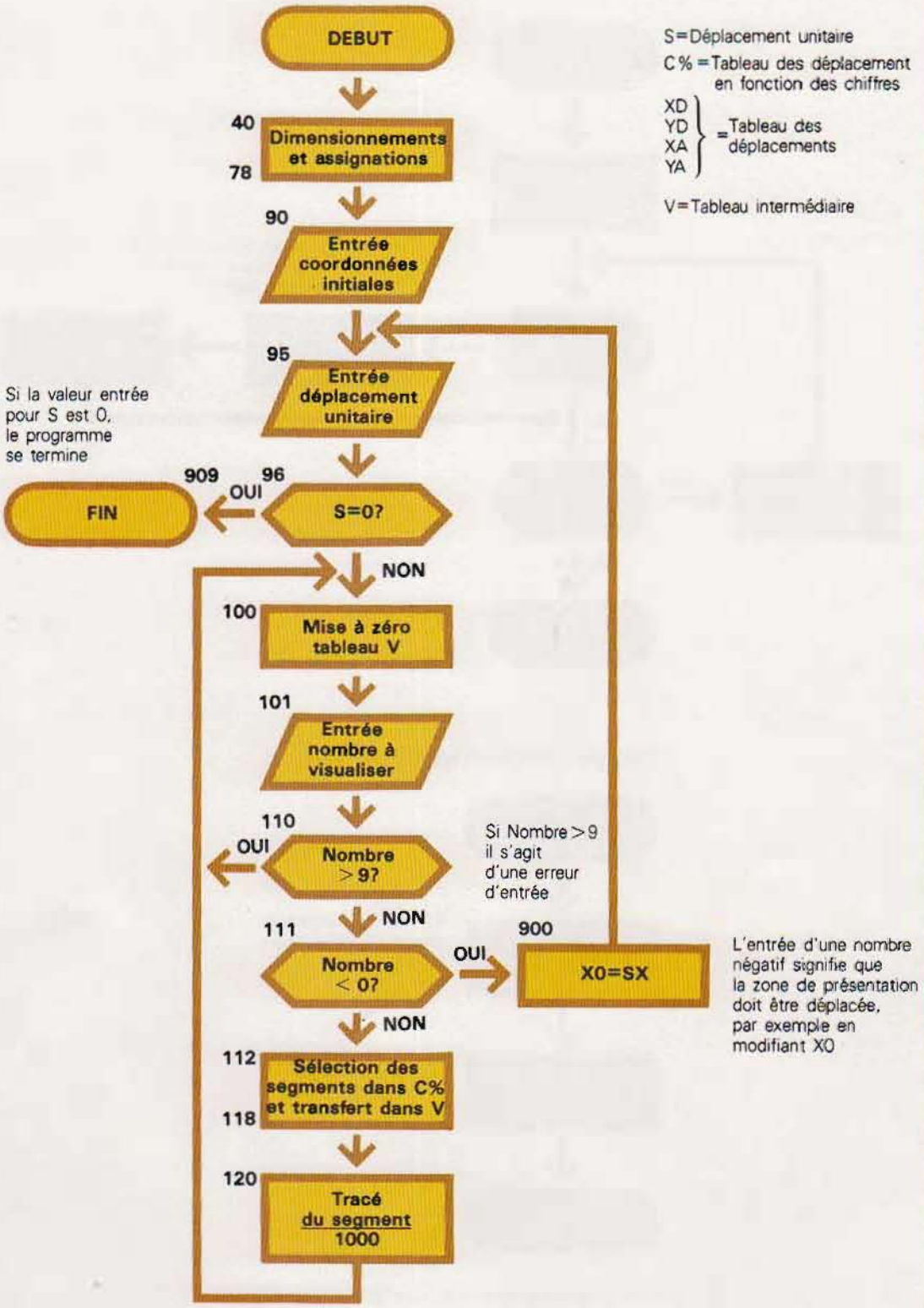
Le sous-programme 1000 (figure 1, page 1488) contient un appel au sous-programrne 2000 (figure 2) qui calcule les positions de début (X1, Y1) et de fin (X2, Y2) du segment et qui, ensuite, le trace.

Graphique tridimentionnel utilisant des courbes de niveau.

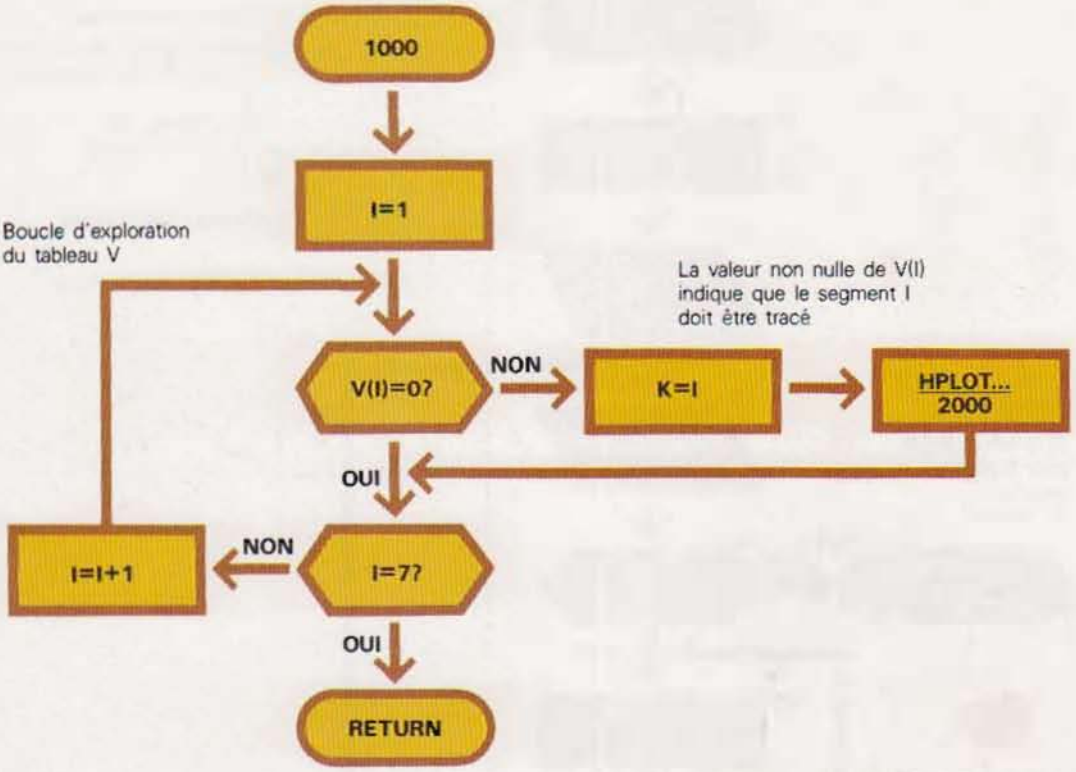


Marka

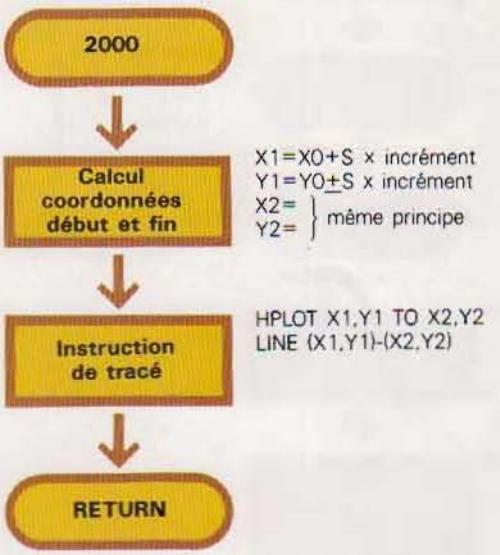
VISUALISATION D'UN NOMBRE EN MODE GRAPHIQUE



1/SOUS-PROGRAMME DE VISUALISATION D'UN SEGMENT



2/SOUS-PROGRAMME DE TRACE



Ce sous-programme dépend du type de machine utilisé

Pour des machines différentes d'Apple et de Siprel 2010 (programme ci-dessous), il suffit de modifier le sous-programme 3000. Dans la version Olivetti M20, le reste demeurant inchangé, les lignes 3010, 3030 et 3040 deviennent :

```
3010 Y1=Y0+YD (K)*S
3030 Y2=Y0+YA (K)*S
3040 LINE (X1,Y1) — (X2,Y2)
```

Le problème de la visualisation des caractères alphabétiques est également résolu sans difficulté grâce à un sous-programme qui représente chaque lettre comme un dessin.
Comme pour les chiffres, il s'agit donc de schématiser les caractères avec une série de segments. Mais l'analogie n'est que formelle. Pour les chiffres on peut définir 7 segments de base qui, selon qu'ils sont éteints ou allumés, génèrent tous les nombres. Pour les lettres, une telle procédure exigerait un nombre trop élevé de segments de base. Il faut donc employer une autre méthode pour concevoir n'importe quel type de figure.
Chaque lettre est schématisée par une série de segments (11 segments au maximum dans notre application) et visualisée en positionnant le pinceau électronique sur le point choisi comme début, puis en traçant les segments nécessaires.

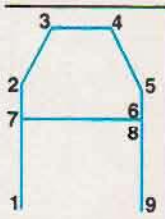
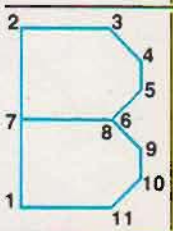
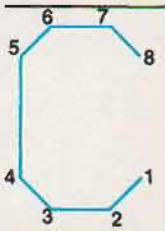
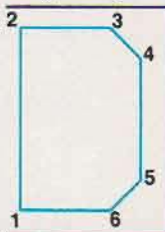
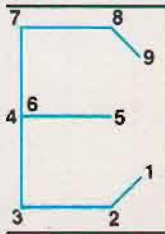
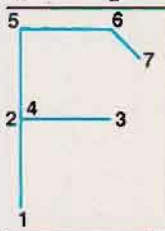
VISUALISATION DES CHIFFRES EN MODE GRAPHIQUE

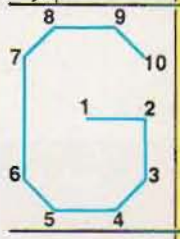
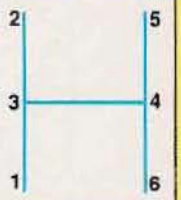
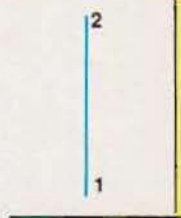
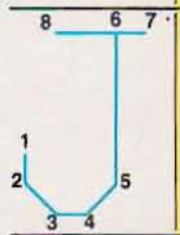
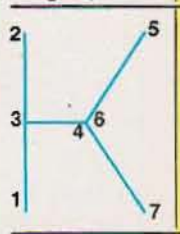
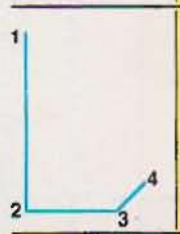
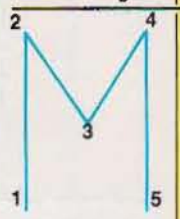
```
10 REM ---------
20 REM NOMBRES
30 REM ---------
40 DIM XD(7),YD(7),XA(7),YA(7)
45 DIM V(7),C%(10,7)
46 HGR
50 FOR I=1 TO 7
60 READ XD(I),YD(I),XA(I),YA(I)
70 NEXT I
72 FOR I=1 TO 10
74 FOR J=7 TO 1 STEP -1
76 READ C%(I,J)
   : NEXT J
78 NEXT I
90 PRINT "COORDONNEES INITIALES"
91 INPUT "X0 = ?" ; X0
92 INPUT "Y0 = ?" ; Y0
93 SX=X0
95 INPUT "DEPLACEMENT UNITAIRE? " ; S
96 IF S=0 GOTO 909
97 Y0=Y0+3*S
100 FOR I=1 TO 7
   : V(I)=0
   : NEXT I
101 INPUT "NOMBRE? " ; N
110 IF N>9 GOTO 100
111 IF N<0 GOTO 900
112 L%=N
   : IF L%=0 THEN L%=10
114 FOR I=1 TO 7
116 V(I)=C%(L%,I)
118 NEXT I
120 GOSUB 1000
160 X0=X0+2*S
170 GOTO 100
900 X0=SX
902 GOTO 95
909 END
```

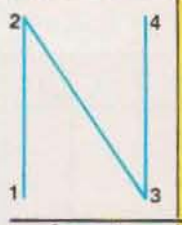
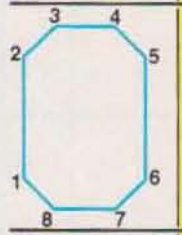
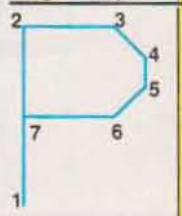
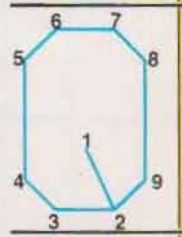
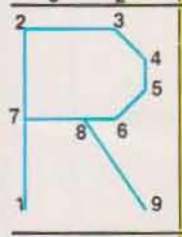
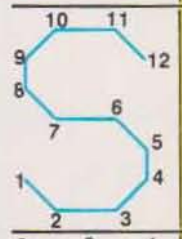
```
910 DATA 0,0,0,1
    : REM =a
911 DATA 0,1,0,2
    : REM =b
912 DATA 0,2,1,2
    : REM =c
913 DATA 1,2,1,1
    : REM =d
914 DATA 1,1,1,0
    : REM =e
915 DATA 1,0,0,0
    : REM =f
916 DATA 0,1,1,1
    : REM =g
918 DATA 0,0,1,1,0,0,0
    : REM =1
919 DATA 1,1,0,1,1,0,1
    : REM =2
920 DATA 1,1,1,1,1,0,0
    : REM =3
921 DATA 1,0,1,0,0,1,0
    : REM =4
922 DATA 1,1,1,0,1,1,0
    : REM =5
923 DATA 1,1,1,0,0,1,1
    : REM =6
924 DATA 0,0,1,1,1,0,0
    : REM =7
925 DATA 1,1,1,1,1,1,1
    : REM =8
926 DATA 1,0,1,1,1,1,0
    : REM =9
927 DATA 0,1,1,1,1,1,1
    : REM =0
1000 FOR I=1 TO 7
1020 IF V(I)=0 GOTO 1040
1025 K=I
1030 GOSUB 3000
1040 NEXT I
1050 RETURN
3000 X1=X0+XD(K)*S
3010 Y1=Y0-YD(K)*S
3020 X2=X0+XA(K)*S
3030 Y2=Y0-YA(K)*S
3040 HPLOT X1,Y1 TO X2,Y2
3050 RETURN
```

Le tableau ci-contre et les suivants listent les déplacements qui permettent de visualiser les caractères alphabétiques. Chaque lettre est contenue dans un rectangle de base 4 et de hauteur 6 : sa position est définie par l'angle en bas à gauche et le sens des déplacements est positif vers la droite pour l'axe X ; positif vers le haut pour l'axe Y. La première colonne du tableau indique le déplacement à effectuer pour se positionner sur le premier point de départ. Ce déplacement est nécessaire dans la mesure où la position du caractère sur l'écran doit être définie par l'utilisateur. Pour la lettre A par exemple, le tracé commence au point de référence (point 1), le déplacement initial est donc zéro. Pour tracer le C, il faut, au contraire, commencer par un point différent et prévoir un déplacement initial (4,1). Dans les tableaux, les colonnes numérotées (1, 2..., 11) repèrent les déplacements qui ne sont pas égaux à ceux figurant sur chaque lettre. Ils précisent le parcours effectué. Pour la lettre A par exemple, le premier déplacement, qui va du point 1 au point 2, est 0,4, le deuxième est 1,2, etc.

VISUALISATION DES CARACTERES ALPHABETIQUES EN MODE GRAPHIQUE

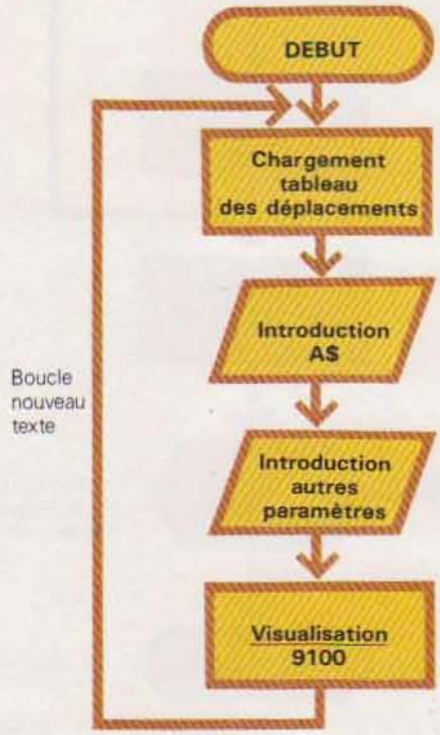
Lettre	Position	Déplacements 1	2	3	4	5	6	7	8	9	10	11
	0,0	0,4	1,2	2,0	1,-2	0,-1	-4,0	4,0	0,-3	-	-	-
	0,0	0,6	3,0	1,-1	0,-1	-1,-1	-3,0	3,0	1,-1	0,-1	-1,-1	-3,0
	4,1	-1,-1	-2,0	-1,1	0,4	1,1	2,0	1,-1	-	-	-	-
	0,0	0,6	3,0	1,-1	0,-4	-1,-1	-3,0	-	-	-	-	-
	4,1	-1,-1	-3,0	0,+3	3,0	-3,0	0,3	3,0	1,-1	-	-	-
	0,0	0,3	3,0	-3,0	0,3	3,0	1,-1	-	-	-	-	-

		1	2	3	4	5	6	7	8	9	10	11
	2,3	2,0	0,-2	-1,-1	-2,0	-1,1	0,4	1,1	2,0	1,-1	-	-
	0,0	0,6	0,-3	4,0	0,3	0,-6	-	-	-	-	-	-
	2,0	0,6	-	-	-	-	-	-	-	-	-	-
	0,2	0,-1	1,-1	1,0	1,1	0,5	1,0	-3,0	-	-	-	-
	0,0	0,6	0,-3	2,0	2,3	-2,-3	2,-3	-	-	-	-	-
	0,6	0,-6	3,0	1,1	-	-	-	-	-	-	-	-
	0,0	0,6	2,-3	2,3	0,-6	-	-	-	-	-	-	-

	1	2	3	4	5	6	7	8	9	10	11
 0,0	0,6	4,-6	0,6	-	-	-	-	-	-	-	-
 0,1	0,4	1,1	2,0	1,-1	0,-4	-1,-1	-2,0	-1,1	-	-	-
 0,0	0,6	3,0	1,-1	0,-1	-1,-1	-3,0	-	-	-	-	-
 2,2	1,-2	-2,0	-1,1	0,4	1,1	2,0	1,-1	0,-4	-1,-1	-	-
 0,0	0,6	3,0	1,-1	0,-1	-1,-1	-3,0	2,0	2,-3	-	-	-
 0,1	1,-1	2,0	1,1	0,1	-1,1	-2,0	-1,1	0,1	1,1	2,0	1,-1
 2,0	0,6	-2,0	4,0	-	-	-	-	-	-	-	-

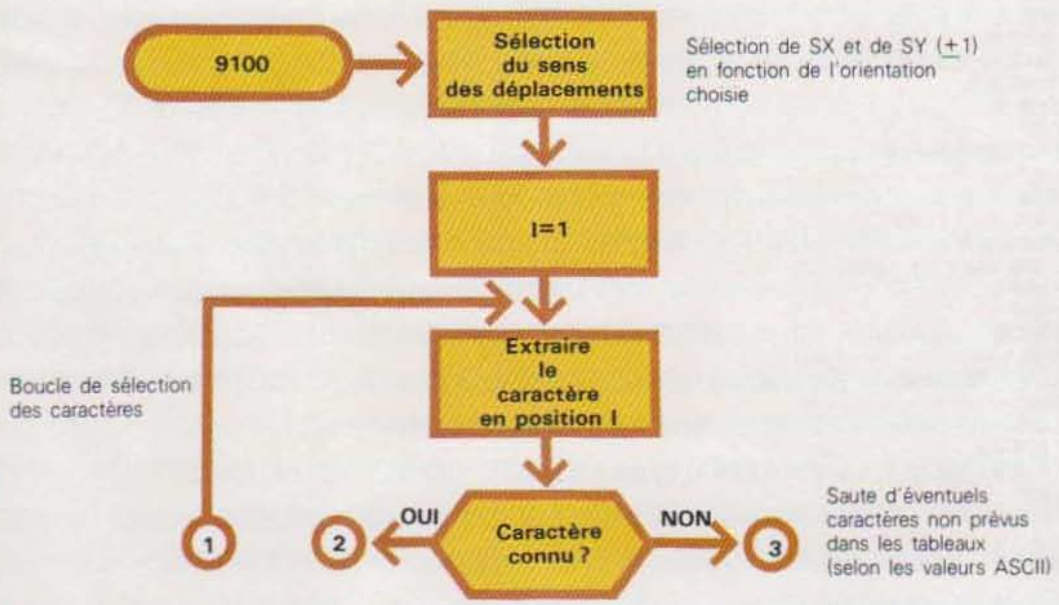
		1	2	3	4	5	6	7	8	9	10	11
1 6 2 5 3 4	0,6	0,-4	1,-2	2,0	1,2	0,4	-	-	-	-	-	-
1 5 2 4 3	0,6	0,-3	2,-3	2,3	0,3	-	-	-	-	-	-	-
8 4 9 7 5 3 6 2 10 1 11	0,0	0,1	4,4	0,1	0,-1	-2,-2	-2,2	0,1	0,-1	4,-4	0,-1	-
3 5 4 2 1	2,0	0,3	-2,3	2,-3	2,3	-		-	-	-	-	-
1 5 3 2 4	0,6	0,-6	2,3	2,-3	0,6	-	-	-	-	-	-	-
5 4 6 1 3 2	4,1	-1,-1	-3,0	4,6	-3,0	-1,-1	-	-	-	-	-	-

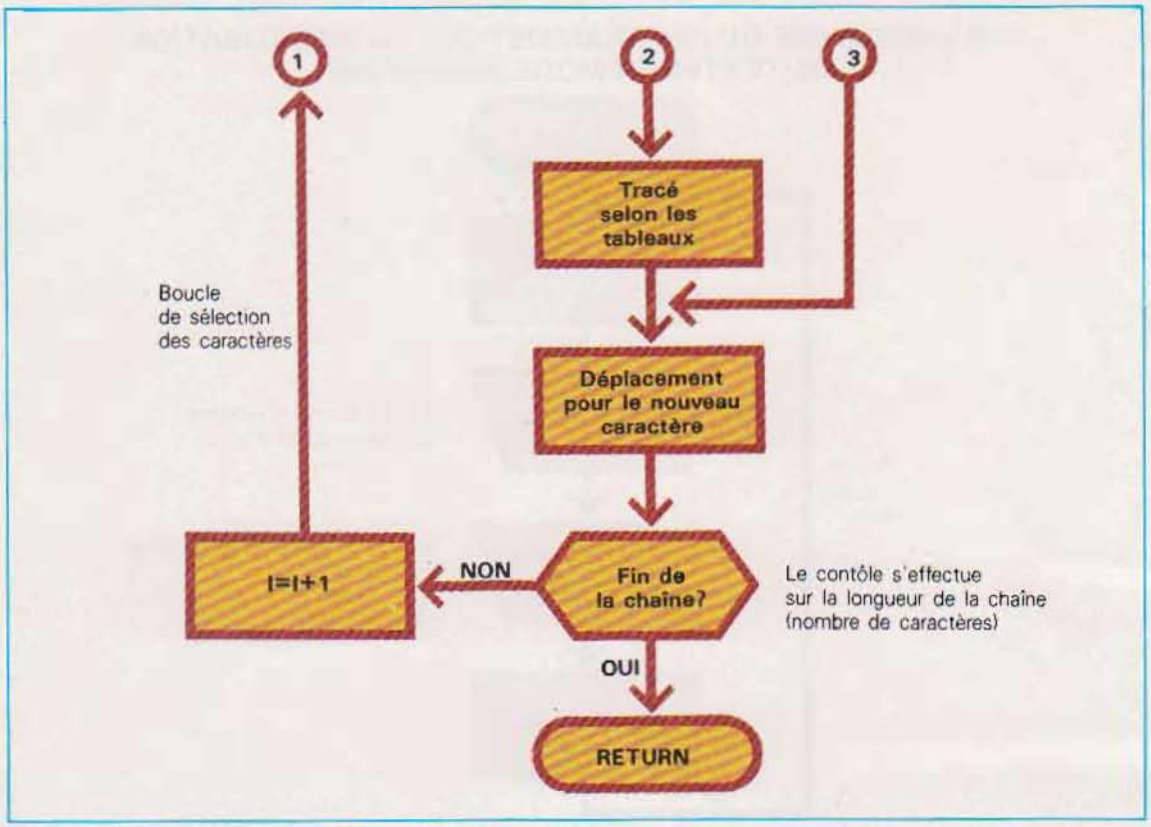
ORGANIGRAMME DU PROGRAMME POUR LA VISUALISATION DE TEXTES EN MODE GRAPHIQUE



Le programme n'a qu'une valeur d'exemple et ne prévoit pas de point de sortie

SOUS-PROGRAMME DE VISUALISATION DES CARACTERES





VISUALISATION DES CARACTERES EN MODE GRAPHIQUE

```
10 HGR
   : HCOLOR=3
20 DIM PO%(26,2),CAR%(26,11,2)
30 GOSUB 350
40 HOME
50 VTAB 22
60 INPUT A$
70 INPUT X0,Y0,S,D,MOV
80 GOSUB 9100
90 GOTO 40
320 REM ----------------
330 REM INITIALISATION
340 REM ----------------
350 FOR I=1 TO 26
360 READ PO%(I,1),PO%(I,2)
370 FOR J=1 TO 11
380 READ CAR%(I,J,1),CAR%(I,J,2)
390 NEXT
400 NEXT
410 RETURN
420 REM ----------------------
430 REM        DONNEES
440 REM ----------------------
450 REM.          A
460 DATA 0,0
470 DATA 0,4,1,2,2,0,1,-2,0,-1,-4,0,4,0,0,-3,0,0,0,0,0,0
480 REM           B
490 DATA 0,0
500 DATA 0,6,3,0,1,-1,0,-1,-1,-1,-3,0,3,0,1,-1,0,-1,-1,-1,-3,0
510 REM           C
```

```
520 DATA 4,1
530 DATA -1,-1,-2,0,-1,1,0,4,1,1,2,0,1,-1,0,0,0,0,0,0,0,0
540 REM          D
550 DATA 0,0
560 DATA 0,6,3,0,1,-1,0,-4,-1,-1,-3,0,0,0,0,0,0,0,0,0,0,0
570 REM          E
580 DATA 4,1
590 DATA -1,-1,-3,0,0,3,3,0,-3,0,0,3,3,0,1,-1,0,0,0,0,0,0
600 REM          F
610 DATA 0,0
620 DATA 0,3,3,0,-3,0,0,3,3,0,1,-1,0,0,0,0,0,0,0,0,0,0
630 REM          G
640 DATA 2,3
650 DATA 2,0,0,-2,-1,-1,-2,0,-1,1,0,4,1,1,2,0,1,-1,0,0,0,0
660 REM          H
670 DATA 0,0
680 DATA 0,6,0,-3,4,0,0,3,0,-6,0,0,0,0,0,0,0,0,0,0,0,0
690 REM          I
700 DATA 2,0
710 DATA 0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
720 REM          J
730 DATA 0,2
740 DATA 0,-1,1,-1,1,0,1,1,0,5,1,0,-3,0,0,0,0,0,0,0,0,0
750 REM          K
760 DATA 0,0
770 DATA 0,6,0,-3,2,0,2,3,-2,-3,2,-3,0,0,0,0,0,0,0,0,0,0
780 REM          L
790 DATA 0,6
800 DATA 0,-6,3,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
810 REM          M
820 DATA 0,0
830 DATA 0,6,2,-3,2,3,0,-6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
840 REM          N
850 DATA 0,0
860 DATA 0,6,4,-6,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
870 REM          O
880 DATA 0,1
890 DATA 0,4,1,1,2,0,1,-1,0,-4,-1,-1,-2,0,-1,1,0,0,0,0,0,0
900 REM          P
910 DATA 0,0
920 DATA 0,6,3,0,1,-1,0,-1,-1,-1,-3,0,0,0,0,0,0,0,0,0,0,0
930 REM          Q
940 DATA 2,2
950 DATA 1,-2,-2,0,-1,1,0,4,1,1,2,0,1,-1,0,-4,-1,-1,0,0,0,0
960 REM          R
970 DATA 0,0
980 DATA 0,6,3,0,1,-1,0,-1,-1,-1,-3,0,2,0,2,-3,0,0,0,0,0,0
990 REM          S
1000 DATA 0,1
1010 DATA 1,-1,2,0,1,1,0,1,-1,1,-2,0,-1,1,0,1,1,1,2,0,1,-1
1020 REM         T
1030 DATA 2,0
1040 DATA 0,6,-2,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

```
1050 REM          U
1060 DATA 8,6
1070 DATA 8,-4,1,-2,2,0,1,2,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1080 REM          V
1090 DATA 8,6
1100 DATA 8,-3,2,-3,2,3,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1110 REM          W
1120 DATA 8,6
1130 DATA 8,-6,2,3,2,-3,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1140 REM          X
1150 DATA 8,8
1160 DATA 8,1,4,4,4,0,1,0,-1,-2,-2,-2,2,0,1,0,-1,4,-4,0,-1,0,0
1170 REM          Y
1180 DATA 2,0
1190 DATA 8,3,-2,3,2,-3,2,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1200 REM          Z
1210 DATA 4,1
1220 DATA -1,-1,-3,0,4,6,-3,0,-1,-1,0,0,0,0,0,0,0,0,0,0,0,0
9000 REM ---------------------------------
9001 REM LETTRES EN HAUTE RESOLUTION
9002 REM ---------------------------------
9023 REM ---------------------------------
9100 D=D+4
9110 IF MOV=1 THEN SX=1
     : SY=1
     : M1=1
     : M2=2
9120 IF MOV=2 THEN SX=1
     : SY=-1
     : M1=2
     : M2=1
9130 IF MOV=3 THEN SX=-1
     : SY=-1
     : M1=2
     : M2=2
9140 IF MOV=4 THEN SX=-1
     : SY=1
     : M1=2
     : M2=1
9150 LU=LEN(A$)
9160 FOR I=1 TO LU
9170 LE$=MID$(A$,I,1)
9180 COD=ASC(LE$)
     : NL=COD-64
9190 IF COD<65 OR COD>90 THEN 9260
     : REM A MODIFIER POUR LES NOUVEAUX CARACTERES
9200 X1=X0+(S*PO%(NL,M1)*SX)
     : Y1=Y0-(S*PO%(NL,M2)*SY)
9210 X2=X1+(S*CAR%(NL,J,M1)*SX)
     : Y2=Y1-(S*CAR%(NL,J,M2)*SY)
9230 HPLOT X1,Y1 TO X2,Y2
9240 X1=X2
     : Y1=Y2
9250 NEXT
9260 IF MOV=2 OR MOV=4 THEN Y0=Y0-(D*S*SY)
     : GOTO 9200
9270 X0=X0+(D*S*SX)
9280 NEXT
9290 RETURN
```

Le programme (1495 à 1498) se compose surtout de 3 blocs. Le principal (lignes 10, 90) contient la définition des zones de mémoire (DIM) et les appels de sous-programmes. Parmi ces derniers, la ligne 350 définit, par une série d'instruction DATA, les valeurs des déplacements ; la 9100 les exécute, en visualisant le caractère.

Soit l'exemple d'une chaîne représentée par A$ et des paramètres XO, YO, S, D, MOV lus à partir du clavier. Pour généraliser le programme et le rendre exécutable dans toutes les applications, il suffit de le transformer en sous-programme et d'y entrer les valeurs acquises ou définies ailleurs.

Paramètres utilisés :

XO, YO : Coordonnées du point de référence du rectangle contenant la droite (début de l'écriture).

S : Facteur d'échelle qui détermine les dimensions de visualisation des caractères. En multipliant par S les déplacements on obtient le même caractère agrandi.

D : Espace qui définit la distance entre deux caractères.

MOV : Indique l'orientation du texte. Les orientations codifiées par des numéros allant de 1 à 4 sont précisées dans le listing.

REPRESENTATION DES CARACTERES NUMERIQUES EN MODE GRAPHIQUE

L'écran en bas de page illustre une représentation graphique des caractères numériques.

Ci-contre : phase d'introduction des coordonnées initiales et de la valeur du déplacement unitaire.

Le programme active maintenant le mode graphique, en conservant une fenêtre de 4 lignes pour le dialogue.
L'utilisateur a introduit une série de chiffres à représenter et à afficher sur l'écran.
L'introduction d'un nombre négatif implique une nouvelle sélection du déplacement unitaire.

Après avoir sélectionné un déplacement plus petit, les dimensions des chiffres se réduisent. Notons bien, cependant, que les proportions restent les mêmes.

Dans ce cas, on a changé les valeurs des coordonnées de départ et le déplacement unitaire.

Les chiffres visualisés avec un déplacement 4 sont toujours plus grands que les chiffres ASCII standard, que l'on voit dans la partie réservée au dialogue.

Le dernier écran montre la réduction d'échelle progressive que l'on obtient au fur et à mesure que l'on diminue la valeur du déplacement unitaire.

VISUALISATION DES CARACTERES ALPHABETIQUES EN MODE GRAPHIQUE

Exemples d'emploi de caractères alphabétiques en mode graphique.

Ci-contre, le résultat de l'exécution du programme. On a introduit la chaîne BASIC.

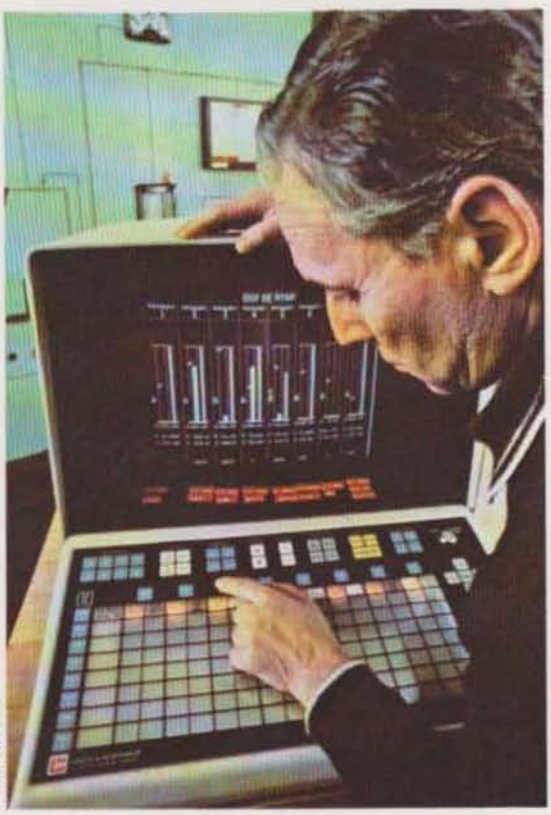
Le deuxième écran montre les différentes possibilités du programme. Le texte peut être positionné sur n'importe quel point de l'écran, avec l'orientation désirée.

En travaillant davantage le plan de tracé des caractères, il est possible de reproduire divers caractères typographiques. L'écran ci-contre en fournit quelques échantillons.

Histogrammes et diagrammes circulaires

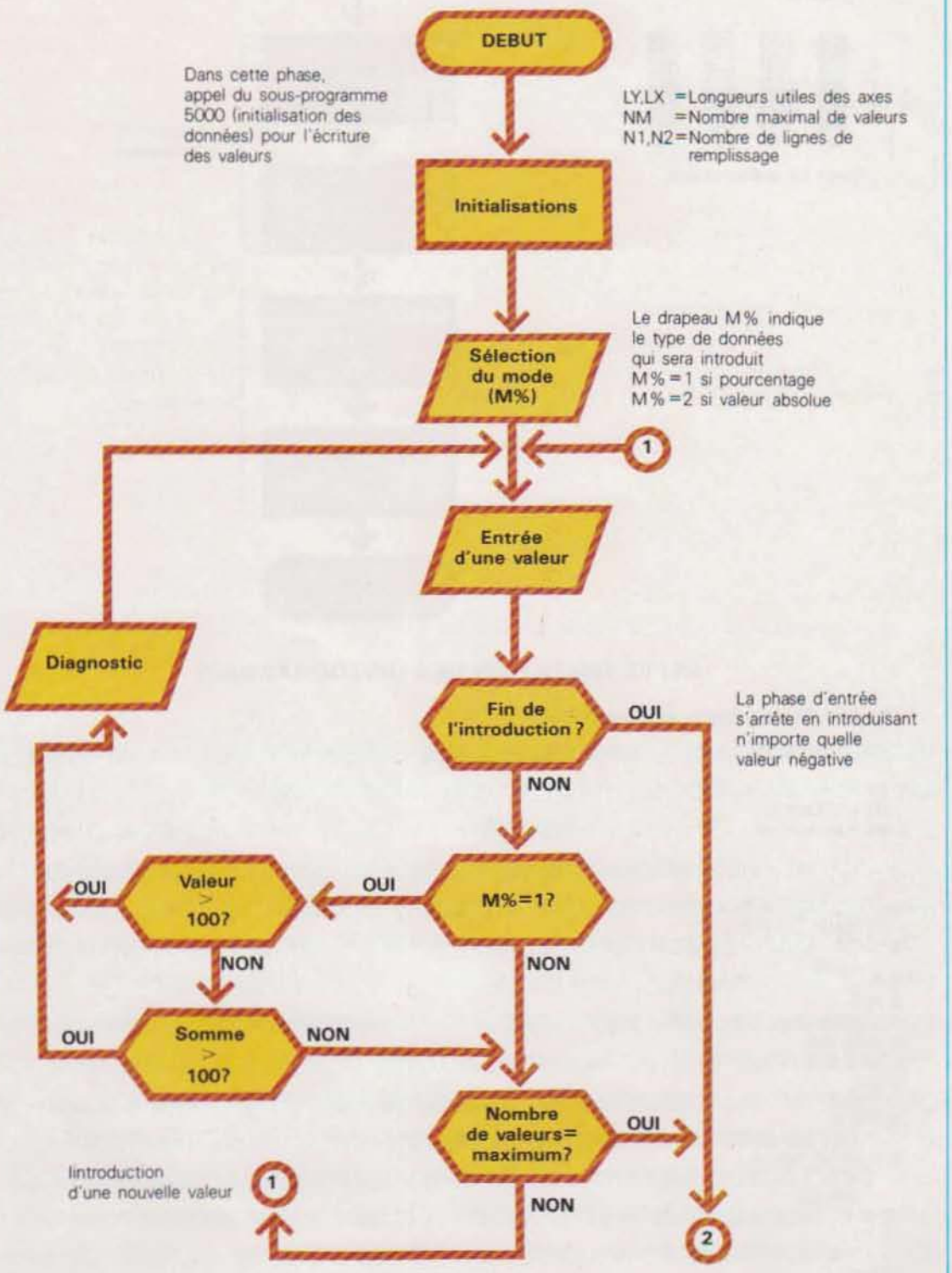
Une représentation graphique quelconque s'obtient en traçant la progression de la fonction mathématique qui le décrit (à condition qu'elle existe). Certains phénomènes ne peuvent pas être décrits au moyen d'une fonction, car la quantité à représenter ne dépend pas d'une variable (indépendante) qui varie de manière continue, ou parce que la variable indépendante n'est pas de type numérique.
Nous avons déjà donné un exemple se rapportant aux ventes d'un article pour différents vendeurs ; il n'était pas possible de considérer, dans ce cas, les vendeurs (A, B, C...) comme des données numériques. Un autre exemple très courant concerne la représentation du bilan mensuel sur une année. Dans ce cas, "la variable indépendante" est le nom du mois (janvier, février...), qui n'est certes pas une variable numérique.
Pour résoudre ces problèmes de représentation graphique, il existe deux méthodes très répandues : les histogrammes et les diagrammes circulaires.
Les histogrammes se présentent comme une série de colonnes d'une hauteur proportionnelle aux valeurs à représenter selon l'axe Y et distantes, sur l'axe X, d'une grandeur proportionnelle aux valeurs de la variable indépendante (X). En général, le pas est constant, dans la mesure où le phénomène est observé à des intervalles réguliers. Dans le cas d'un bilan, la variable indépendante est le temps et la variable dépendante le montant. Elles sont reportées sur une période mensuelle, l'axe X étant divisé en 12 parties (alors que l'axe Y représente le chiffre mensuel, soit en valeur absolue, soit en pourcentage du total). Cette dernière forme est la plus usitée car elle fournit une vision immédiate du phénomène et facilite la construction du diagramme.
Le diagramme circulaire est une représentation proche de l'histogramme, où chaque valeur est convertie en une portion de cercle de grandeur proportionnelle. L'ensemble du cercle représente la valeur totale et chaque secteur la valeur d'une de ses composantes. Exemple : pour représenter le bilan précédent sous cette forme, il faut construire un cercle divisé en 12 secteurs, un par mois, d'une grandeur proportionnelle au pourcentage du montant de ce mois sur le total annuel.
L'histogramme fournit donc une vision de l'évolution du bilan dans le temps, alors que le diagramme circulaire montre comment l'ensemble du bilan se répartit entre les diverses parties (mois). Il s'agit donc de deux représentations qui se complètent. Pages 1503-1504, on a construit l'organigramme d'un programme de représentation d'histogrammes (voir listings en pages 1504-1510). Les données à représenter sont soit des valeurs absolues, soit des pourcentages. Dans le premier cas, le programme les convertit lui-même en pourcentages. L'utilisateur choisit entre 5 formes de représentation différentes sur le même graphique. Dans le programme, les 5 modes de présentation se réfèrent à un écran monochromatique.
Ce fait complique le sous-programme de présentation car il oblige à prévoir divers modes de représentation des colonnes de l'histogramme. Sur un écran couleur on peut associer, à chaque mode de représentation, une couleur différente et dessiner les divers histogrammes.

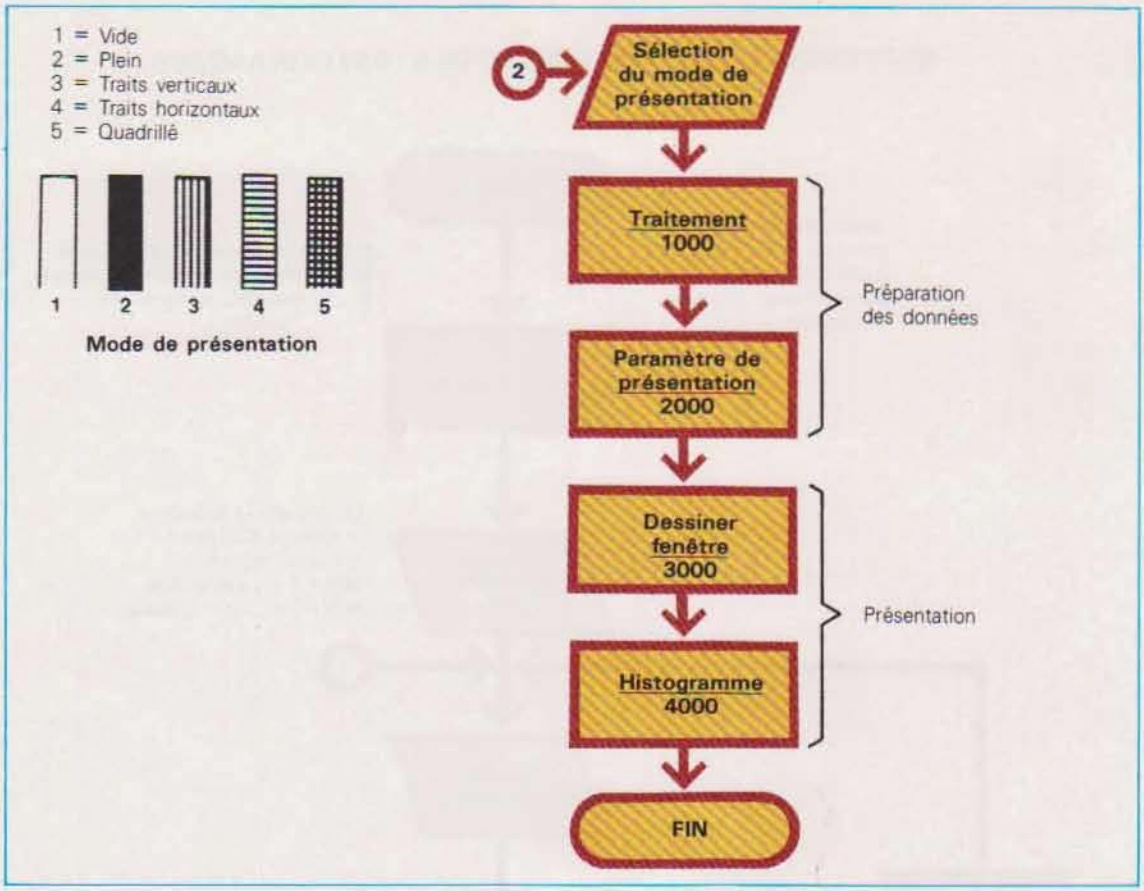
Contrôle de processus sur écran graphique



K. Reese/Marka

REPRESENTATION GRAPHIQUE DES HISTOGRAMMES





REPRESENTATION DES HISTOGRAMMES

Version Siprel, Apple et compatibles

```
4 REM --------------
5 REM HISTOGRAMMES
6 REM --------------
7
    :
8
    :
9 DIM PO%(26,2)CAR%(26,11,2)
10 LX=230
    : LY=150
20 NM=20
30 P%=0
50 DIM D(NM),P(NM),S(NM),ES(NM)
60 GOSUB 5000
65 GOSUB 9120
70 TEXT
    : HOME
80 VTAB 12
    : INPUT "LES DONNEES SONT EN POURCENTAGE? (O/N)" ; A$
90 IF A$="O" THEN M%=1
    : P%=1
    : GOTO 110
100 M%=2
110 I=1
    : TT=0
120 HOME
```

```
    : VTAB 12
130 PRINT "INTRODUIRE LA DONNEE N " ; I
140 INPUT "(NEGATIF POUR FINIR): " ; D$
    : IF D$="" THEN 120
150 D=VAL(D$)
160 IF D>99999 THEN 120
170 IF D<0 THEN 260
180 D(I)=INT(D)
190 TT=TT+D(I)
200 IF M%=2 THEN 220
210 IF D(I)>100 OR TT>100 THEN GOSUB 6000
    : HOME
    : GOTO 110
    : REM          ERREUR
220 VTAB 15
    : INPUT "ETIQUETTE (MAX 3 CARACTERES): " ; E$
230 E$(I)=LEFT$(E$,3)
240 I=I+1
    : IF I>NM THEN 260
250 GOTO 120
260 IF TT=0 THEN RUN
270 NV=I-1
280 HOME
290 VTAB 5
    : PRINT "MODES DE PRESENTATION: "
300 VTAB 10
310 PRINT "1) VIDE"
320 PRINT "2) PLEIN"
330 PRINT "3) TRAITS VERTICAUX"
340 PRINT "4) TRAITS HORIZONTAUX"
350 PRINT "5) INCRUSTE"
360 VTAB 22
    : INPUT "CHOISIR: " ; MP$
    : IF MP$="" THEN 360
370 MP=VAL(MP$)
380 IF MP<1 OR MP>5 THEN 280
1000 REM ------------
1010 REM TRAITEMENT
1020 REM ------------
1030 IF M%=1 GOTO 1140
1040 HOME
    : VTAB 12
1050 PRINT "VOULEZ-VOUS UNE REPRESENTATION EN POURCENTAGE? ";
1060 INPUT "(O/N): " ; R$
1070 IF R$="O" THEN P%=1
    : GOTO 1140
1080 MA=D(1)
1090 FOR K=2 TO NV
1100 IF D(K)>(MA) THEN MA=D(K)
1120 NEXT K
1130 TT=MA
1140 FOR I=1 TO NV
1150 P(I)=D(I)/TT*100
1160 S(I)=INT(LY*P(I)/100)
1170 NEXT I
1180 LA=INT(LX/(2*NV))
2000 REM ----------------------------
2001 REM PARAMETRES DE PRESENTATION
2002 REM ----------------------------
2010 SX=1
    : SY=1
2040 ON MP GOTO 3000,3000,2060,2070,2080
2060 SX=2
    : GOTO 3000
2070 SY=2
    : GOTO 3000
```

```
2000 SY=2
     : SX=2
3000 REM ----------------------
3010 REM DESIN DE LA FENETRE
3020 REM ----------------------
3030 HGR2
     : HCOLOR=3
3040 HPLOT 34,0 TO 279,0 TO 279,159 TO 34,159 TO 34,0
3050 FOR I=1 TO 10
3060 V=159-I*15
3070 HPLOT 34,V TO 39,V
3075 HPLOT 274,V TO 278,V
3080 NEXT
3090 S=3
     : Y0=159
3100 IF P%=0 THEN 3170
3110 FOR NN=0 TO 100 STEP 10
3120 X0=9*S
3130 GOSUB 5500
3140 Y0=Y0-15
3150 NEXT
3160 GOTO 3250
3170 ST=MA/10
     : IF ST<>INT(ST) THEN ST=INT(ST)+1
3175 Y0=9
3180 FOR NN=MA TO 0 STEP -ST
3190 X0=9*S+3
3200 GOSUB 5500
3210 Y0=Y0+15
3220 NEXT
3230 X0=9*S+3
3235 IF NN+ST=0 THEN 3250
3240 NN=0
     : GOSUB 5500
3250 REM
4000 REM ----------------------------
4001 REM DESSIN DES HISTOGRAMMES
4002 REM ----------------------------
4005 N=0
4007 Y1=159
4010 FOR I=1 TO NV
4025 X1=50+N*LA
4027 X2=X1+LA
4030 HPLOT X1,Y1 TO X1,Y1-S(I) TO X2,Y1-S(I) TO X2,Y1
4040 IF MP=1 THEN 4900
4050 IF MP=4 THEN 4300
4060 Y=Y1
     : X=X1+SX
4070 IF X>X2 THEN 4200
4080 HPLOT X,Y TO X,Y-S(I)
4090 X=X+SX
4100 GOTO 4070
4200 IF MP=5 THEN 4300
4210 GOTO 4900
4300 X=X1
     : Y=Y1-SY
4310 IF Y<Y1-S(I) THEN 4900
4320 HPLOT X1,Y TO X2,Y
4330 Y=Y-SY
4340 GOTO 4310
4900 N=N+2
4905 A$=ES(I)
     : H0=X2
     : V0=181
     : S=1
     : D=2
     : MOV=4
     : GOSUB 10270
```

```
4910 NEXT I
4920 GET F$
4930 TEXT
   : HOME
   : VTAB 12
   : INPUT "DESIREZ-VOUS UN AUTRE GRAPHIQUE? (O/N)" ; R$
4940 IF LEFT$(R$,1)="O" THEN 70
4950 END
5000 REM ------------------------------
5010 REM INITIALISATION DES NOMBRES
5020 REM ------------------------------
5030 DIM XD(7),YD(7),XA(7),YA(7)
5040 DIM V(7),C%(10,7)
5050 FOR I=1 TO 7
5060 READ XD(I),YD(I),XA(I),YA(I)
5070 NEXT I
5080 FOR I=1 TO 10
5090 FOR J=7 TO 1 STEP -1
5100 READ C%(I,J)
   : NEXT J
5110 NEXT I
5120 RETURN
5500 REM ---------------------
5510 REM DESSIN DES NOMBRES
5520 REM ---------------------
5530 NN$=STR$(NN)
5540 LN=LEN(NN$)
5550 FOR KK=LN TO 1 STEP -1
5560 N=VAL(MID$(NN$,KK,1))
5570 X0=X0-2*S
5580 FOR I=1 TO 7
   : V(I)=0
   : NEXT I
5590 L%=N
   : IF L%=0 THEN L%=10
5600 FOR I=1 TO 7
5610 V(I)=C%(L%,I)
5620 NEXT I
5630 FOR I=1 TO 7
5640 IF V(I)=0 GOTO 5710
5650 K=1
5660 X1=X0+XD(K)*S
5670 Y1=Y0-YD(K)*S
5680 X2=X0+XA(K)*S
5690 Y2=Y0-YA(K)*S
5700 HPLOT X1,Y1 TO X2,Y2
5710 NEXT I
5720 NEXT KK
5730 RETURN
6000 VTAB 20
   : PRINT "INTRODUCTION ERRONEE: "
6010 PRINT
   : PRINT "LA SOMME DES DONNEES DEPASSE LA VALEUR '100'"
6015 PRINT
   : PRINT "FRAPPER UNE TOUCHE POUR RECOMMENCER"
6030 GET Q$
6040 RETURN
7945 REM --------------
7950 REM    DONNEES
7955 REM --------------
8000 DATA 0,0,0,1
   : REM =a
8010 DATA 0,1,0,2
   : REM =b
8020 DATA 0,2,1,2
   : REM =c
8030 DATA 1,2,1,1
   : REM =d
```

```
8040 DATA 1,1,1,0
    : REM =e
8050 DATA 1,0,0,0
    : REM =f
8060 DATA 0,1,1,1
    : REM =g
8070 DATA 0,0,1,1,0,0,0
    : REM =1
8080 DATA 1,1,0,1,1,0,1
    : REM =2
8090 DATA 1,1,1,1,1,0,0
    : REM =3
8100 DATA 1,0,1,0,0,1,0
    : REM =4
8110 DATA 1,1,1,0,1,1,0
    : REM =5
8120 DATA 1,1,1,0,0,1,1
    : REM =6
8130 DATA 0,0,1,1,1,0,0
    : REM =7
8140 DATA 1,1,1,1,1,1,1
    : REM =8
8150 DATA 1,0,1,1,1,1,0
    : REM =9
8160 DATA 0,1,1,1,1,1,1
    : REM =0
9090 REM ----------------
9100 REM INITIALISATION
9110 REM ----------------
9120 FOR I=1 TO 26
9130 READ PO%(I,1),PO%(I,2)
9140 FOR J=1 TO 11
9150 READ CAR%(I,J,1),CAR%(I,J,2)
9160 NEXT J
9170 NEXT I
9180 RETURN
9190 REM -----------------
9200 REM      DONNEES
9210 REM -----------------
9220 REM          A
9230 DATA 0,0
9240 DATA 0,4,1,2,2,0,1,-2,0,-1,-4,0,4,0,0,-3,0,0,0,0,0,0
9250 REM          B
9260 DATA 0,0
9270 DATA 0,6,3,0,1,-1,0,-1,-1,-1,-3,0,3,0,1,-1,0,-1,-1,-1,-3,0
9280 REM          C
9290 DATA 4,1
9300 DATA -1,-1,-2,0,-1,1,0,4,1,1,2,0,1,-1,0,0,0,0,0,0,0,0
9310 REM          D
9320 DATA 0,0
9330 DATA 0,6,3,0,1,-1,0,-4,-1,-1,-3,0,0,0,0,0,0,0,0,0,0,0
9340 REM          E
9350 DATA 4,1
9360 DATA -1,-1,-3,0,0,3,3,0,-3,0,0,3,3,0,1,-1,0,0,0,0,0,0
9370 REM          F
9380 DATA 0,0
9390 DATA 0,3,3,0,-3,0,0,3,3,0,1,-1,0,0,0,0,0,0,0,0,0,0
9400 REM          G
9410 DATA 2,3
9420 DATA 2,0,0,-2,-1,-1,-2,0,-1,1,0,4,1,1,2,0,1,-1,0,0,0,0
9430 REM          H
9440 DATA 0,0
9450 DATA 0,6,0,-3,4,0,0,3,0,-6,0,0,0,0,0,0,0,0,0,0,0,0
```

```
9460 REM          I
9470 DATA 2,0
9480 DATA 0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
9490 REM          J
9500 DATA 0,2
9510 DATA 0,-1,1,-1,1,0,1,1,0,5,1,0,-3,0,0,0,0,0,0,0,0,0
9520 REM          K
9530 DATA 0,0
9540 DATA 0,6,0,-3,2,0,2,3,-2,-3,2,-3,0,0,0,0,0,0,0,0,0,0
9550 REM          L
9560 DATA 0,6
9570 DATA 0,-6,3,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
9580 REM          M
9590 DATA 0,0
9600 DATA 0,6,2,-3,2,3,0,-6,0,0,0,0,0,0,0,0,0,0,0,0,0
9610 REM          N
9620 DATA 0,0
9630 DATA 0,6,4,-6,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
9640 REM          O
9650 DATA 0,1
9660 DATA 0,4,1,1,2,0,1,-1,0,-4,-1,-1,-2,0,-1,1,0,0,0,0,0,0
9670 REM          P
9680 DATA 0,0
9690 DATA 0,6,3,0,1,-1,0,-1,-1,-1,-3,0,0,0,0,0,0,0,0,0,0,0
9700 REM          Q
9710 DATA 2,2
9720 DATA 1,-2,-2,0,-1,1,0,4,1,1,2,0,1,-1,0,-4,-1,-1,0,0,0,0
9730 REM          R
9740 DATA 0,0
9750 DATA 0,6,3,0,1,-1,0,-1,-1,-1,-3,0,2,0,2,-3,0,0,0,0,0,0
9760 REM          S
9770 DATA 0,1
9780 DATA 1,-1,2,0,1,1,0,1,-1,1,-2,0,-1,1,0,1,1,1,2,0,1,-1
9790 REM          T
9800 DATA 2,0
9810 DATA 0,6,-2,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
9820 REM          U
9830 DATA 0,6
9840 DATA 0,-4,1,-2,2,0,1,2,0,4,0,0,0,0,0,0,0,0,0,0,0,0
9850 REM          V
9860 DATA 0,6
9870 DATA 0,-3,2,-3,2,3,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0
9880 REM          W
9890 DATA 0,6
9900 DATA 0,-6,2,3,2,-3,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0
9910 REM          X
9920 DATA 0,0
9930 DATA 0,1,4,4,0,1,0,-1,-2,-2,-2,2,0,1,0,-1,4,-4,0,-1,0,0
9940 REM          Y
9950 DATA 2,0
9960 DATA 0,3,-2,3,2,-3,2,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0
9970 REM          Z
9980 DATA 4,1
9990 DATA -1,-1,-3,0,4,6,-3,0,-1,-1,0,0,0,0,0,0,0,0,0,0,0,0
```

```
10000 REM ------------------------------
10010 REM LETTRES EN HAUTE RESOLUTION
10020 REM ------------------------------
10030 REM
10040 REM
10050 REM EN ENTREE
10060 REM
10070 REM A$ - CHAINE DE CARACTERES
10080 REM
10090 REM H0,V0 - POSITION DEBUT ECRITURE
10100 REM
10110 REM S - ECHELLE (>0)
10120 REM
10130 REM D - ESPACEMENT (>0)
10135 REM
10140 REM MOV - TYPE D'ECRITURE:
10150 REM
10160 REM 1=HORIZONTALE
10170 REM 2=DESCENDANTE
10180 REM 3=INVERSEE
10190 REM 4=MONTANTE
10200 REM
10210 REM
10220 REM
10230 REM ------------------------------
10270 D=D+4
10280 IF MOV=1 THEN PX=1
      : PY=1
      : M1=1
      : M2=2
10290 IF MOV=2 THEN PX=1
      : PY=-1
      : M1=2
      : M2=1
10300 IF MOV=3 THEN PX=-1
      : PY=-1
      : M1=1
      : M2=2
10310 IF MOV=4 THEN PX=-1
      : PY=1
      : M1=2
      : M2=1
10320 LU=LEN(A$)
10330 FOR II=1 TO LU
10340 LE$=MID$(A$,II,1)
10345 IF LE$=" " THEN II=LU
      : NEXT II
      : RETURN
10350 COD=ASC(LE$)
      : NL=COD-64
10360 IF COD<65 OR COD>90 THEN 10430
      : REM A MODIFIER POUR LES NOUVEAUX CARACTERES
10370 H1=H0+(S*PO%(NL,M1)*PX)
      : V1=V0-(S*PO%(NL,M2)*PY)
10380 FOR J=1 TO 11
10390 H2=H1+(S*CAR%(NL,J,M1)*PX)
      : V2=V1-(S*CAR%(NL,J,M2)*PY)
10400 HPLOT H1,V1 TO H2,V2
10410 H1=H2
      : V1=V2
10420 NEXT J
10430 IF MOV=2 OR MOV=4 THEN V0=V0-(D*S*PY)
      : GOTO 10450
10440 H0=H0+(D*S*PX)
10450 NEXT II
10460 RETURN
```

L'ordinateur et les aiguilleurs du ciel

Vous avez tous entendu parler des contrôleurs du trafic aérien — les aiguilleurs du ciel — mais peu de gens savent vraiment en quoi consiste leur activité et comment les technologies nouvelles sont devenues indispensables au bon déroulement de leurs tâches. Aujourd'hui, on trouve dans le monde des exemples d'automatisation complète du trafic aérien, depuis le contrôle de procédure jusqu'à l'échange de données entre ordinateurs.

Le choix de tel ou tel système est bien évidemment lié au rapport entre le coût de l'automatisation et les bénéfices que l'on peut en tirer. Dans le cas de la surveillance radar, les coûts d'entretien, la formation d'un personnel hautement qualifié sont tels qu'ils dissuadent encore de nombreux pays de se lancer dans l'aventure. Côté avantages, il faut citer une plus grande sécurité et de meilleures conditions de travail. Tout cela dépend, naturellement, du volume du trafic à contrôler.

Le contrôle du trafic aérien (CTA) est un service dont la finalité est d'assurer un trafic sûr, bien organisé et rapide, dans l'intérêt à la fois des voyageurs et des compagnies aériennes. L'organisation mondiale qui supervise et coordonne les problèmes complexes du contrôle aérien (CTA) est l'ICAO (International Civil Aviation Organization).

Pour faciliter le trafic, l'espace aérien mondial est divisé en FIR (Flight Information Region) dont le contrôle est confié aux nations géographiquement concernées.

Les problèmes de coordination sont réduits par un tracé, dans le ciel, de ***routes de navigation*** *idéales, d'une largeur d'environ 10 miles, et comportant plusieurs niveaux d'altitude.*

Contrairement à ce que l'on pourrait penser, le service de contrôle couvre ***toute la durée du vol****, depuis le décollage jusqu'à l'approche de la piste et l'atterrissage, qui restent toujours les phases opérationnelles les plus délicates.*

La tâche du contrôleur consiste à guider les avions par radio, le long des voies aériennes, en tenant compte des demandes des pilotes eux-mêmes ainsi que de la situation du trafic en cours. Seul le contrôleur se trouve en possession de l'ensemble de ces données car la vision du pilote est limitée par divers facteurs, dont la vitesse (un avion de ligne parcourt environ 10 km par minute).

Les trois aires de vol (FIR) composant l'espace aérien italien.

Pour faciliter le travail du contrôleur, les pilotes sont obligés, avant le départ, de rédiger un ***plan de vol*** *(FPL, Flight Plan) qui consigne toutes les informations concernant l'avion, ses caractéristiques techniques (type), sa destination, ainsi qu'une estimation des horaires. Dans le plan de vol, l'itinéraire est exprimé comme une succession de voies aériennes ou de points caractéristiques (FIX), qui bien souvent correspondent à des balises radio au sol, repérées par des noms conventionnels sur les cartes internationales.*

Le plan de vol est transmis à toutes les régions de vol (FIR) concernées par l'itinéraire, grâce à un réseau de ***télex spécial*** *appelé AFTN (Aeronautical Fixed Telecommunication Network).*

Dans chaque région, les assistants de l'aiguilleur du ciel tirent, de chaque plan de vol, les

informations se rapportant à leur propre région. Pour chaque FIX concerné par l'itinéraire, ils reçoivent une bande télex (strip) avec les données essentielles de l'appareil, l'altitude demandée, le nom du FIX et l'horaire de survol prévu. A titre d'exemple, la FIR de Rome prépare 10 à 15 strips par vol; donc pour une moyenne d'environ 900 vols par jour, on arrive en gros à environ 10 000 strips quotidiens.

Le système traditionnel, qui permet au contrôleur de connaître à tout instant le trafic correspondant à un FIX donné, consiste à classer chronologiquement les strips concernant ce point, afin d'éviter les ***croisements d'avions*** *sur un même niveau.*

Pour un contrôle de ce type (dit de procédure), la distance requise entre deux avions est en général de 10 minutes, s'ils volent à la même altitude, ou de 300 mètres (1 000 pieds), s'ils volent à des altitudes différentes.

Un moment très délicat, du point de vue opérationnel, est celui où un avion franchit la limite entre deux régions de vol.

Un réseau téléphonique spécial permet, aux contrôleurs de régions limitrophes, de coordonner le point, l'altitude et le moment du transfert de contrôle.

Grâce à cette communication, le nouveau contrôleur peut évaluer la correction à effectuer par rapport aux horaires estimés des strips, pour synchroniser la ***trajectoire prévue*** *et la* ***véritable position*** *de l'avion.*

Diverses communications avec le pilote au moment du survol des FIX permettent d'évaluer d'éventuels retards ou avances par rapport aux prévisions.

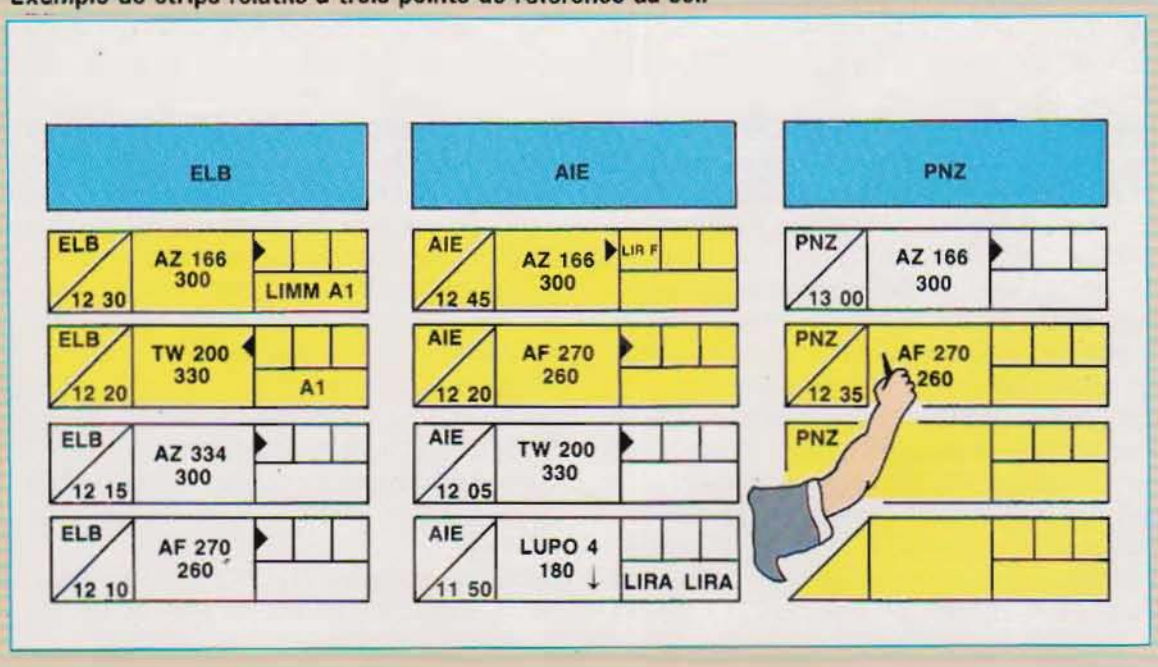
La première application technologique importante du CTA a eu lieu dans les années 40. Il s'agissait du ***radar****. Celui-ci a permis de visualiser le trafic en cours, en générant des images qui n'étaient qu'une série de points lumineux sur un écran (les échos radar).*

Entre 1958 et 1960, on commence à utiliser un second radar dit SSR (Secondary Surveillance Radar), qui interroge un appareil récepteur-émetteur, embarqué dans l'avion, le ***transpondeur****. La réponse numérique que le SSR obtient du transpondeur véhicule une série d'informations concernant l'identité de l'appareil (exprimée avec un code de 64 bits) et l'altitude de vol.*

Ces communications sont traitées par un ordinateur pour l'identification des vols sur écrans PPI (Plan Position Indicator).

Le premier degré d'automatisation est ce que l'on appelle le labelling (étiquetage), qui consiste à associer, à l'écho radar primaire, une étiquette donnant notamment le code de l'avion et son altitude.

Exemple de strips relatifs à trois points de référence au sol.



Après un premier contact radio avec le pilote, le contrôleur remplace le code d'identification (envoyé à l'ordinateur par le transpondeur) par la désignation normale du vol (par exemple AZ 128), qui facilitera la **comparaison** entre la **position** de l'avion, telle qu'elle ressort des informations radar et les **fiches de progression** (voir graphique page précédente).

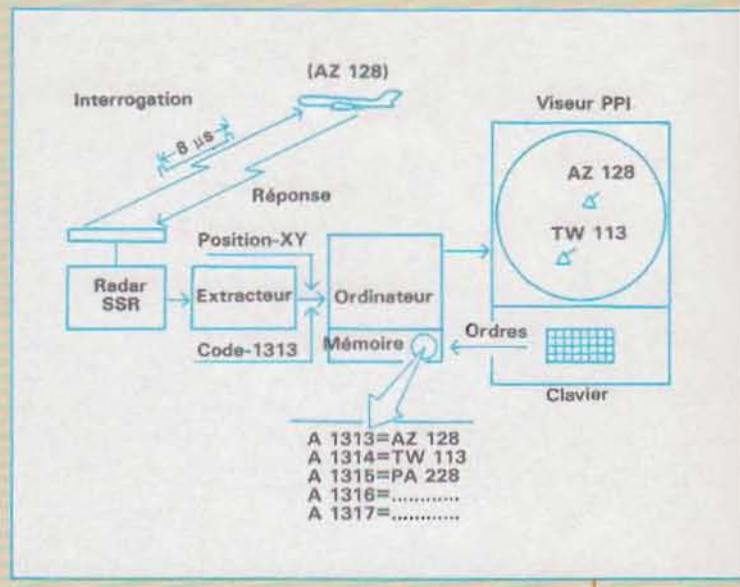
Grâce à l'ordinateur, l'image-écran du moniteur PPI (Plan Position Indicator) est enrichie de plusieurs fonctions destinées au contrôleur, comme par exemple :

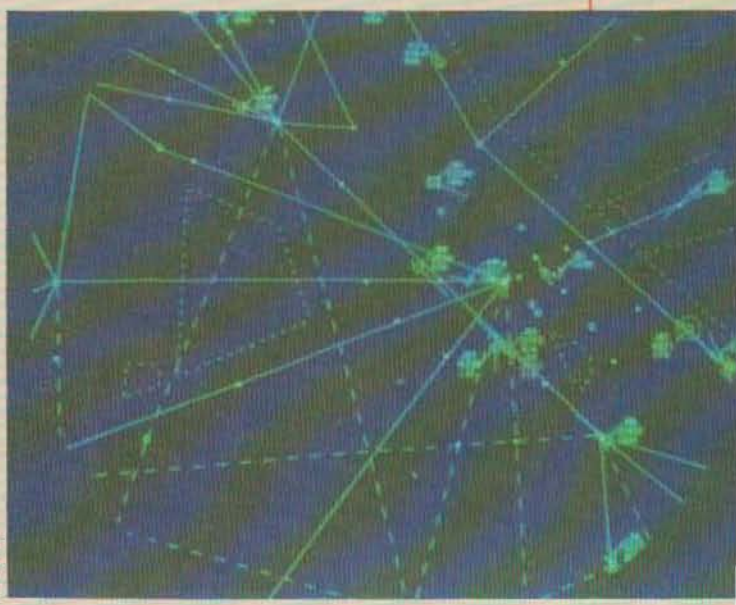
- Une vue schématique des routes aériennes,
- L'ensemble des FIX (points caractéristiques) et des zones particulières (aéroports, zones interdites à la navigation civile, altitudes des principaux reliefs proches d'un aéroport).
- Une carte des itinéraires d'approche des pistes d'atterrissage.

A noter aussi, d'autres fonctions disponibles :

- Calcul de la distance entre deux avions.
- Changement d'échelle de la représentation à l'écran.

La fonction de **tracking** (poursuite), qui réalise le traitement des informations radar, marque une nouvelle phase dans l'automatisation. En effet, les informations fournies par le transpondeur et les échos de réponse aux radars primaire et secondaire (généralement coaxiaux et corotatifs) ne concernent que l'altitude (pour le premier) et la distance (pour les seconds).
Le tracking est la fonction de corrélation de ces données dans le temps pour en déduire une **valeur de vitesse** (intensité, direction, côté) qui est ensuite représentée sur le PPI à l'aide d'un symbole (vecteur) associé au symbole d'identification de l'avion sur l'écran (photo ci-contre).
Le tracking est une fonction complexe, obtenue à partir des données radar, extrapolées avec un taux d'erreurs plus ou moins important. D'autres erreurs tiennent aux équipements et aux programmes quand, par exemple, un espace aérien est couvert par plusieurs





B. Liotta

Identification d'un avion grâce au transpondeur et au tracking vidéo.

radars placés en des points éloignés (multiradar tracking). Si les données en provenance des différents radars ne sont pas erronées, il reste la difficulté de les réduire à la même échelle dans le temps, pour en faire la synthèse.
Une fois calculées la position et la vitesse d'un avion, les données de sorties sont affichées sur le PPI ou réexploitées en entrées pour une opération particulièrement intéressante : l'**analyse des conflits.** En effet, il est possible, à

B. Liotta

Ci-dessus, déroulement d'un strip (fiche de progression) sur l'écran, comportant les divers FIX (balises radio).

Ci-dessous, contrôleur de vol à sa console.

partir des données cinématiques d'un avion, d'effectuer certaines prévisions concernant sa trajectoire, donc des risques de se retrouver à un point déterminé en même temps qu'un autre avion. Cette situation, dangereuse, entraîne le déclenchement d'une alarme et l'intervention urgente du contrôleur.

L'automatisation de cette phase de contrôle assure une aide immense aux contrôleurs et les aide beaucoup à améliorer la sécurité.

Une fois passés les contrôles, les avions volent à des intervalles de moins de 3 minutes, ce qui permet, en phase d'atterrissage, de décongestionner le trafic et de supprimer les ***files d'attente en altitude*** *(holding), très coûteuses pour les compagnies (l'heure peut atteindre 25 000 francs pour un Boeing 747).*

D'autres aspects du service ont également été automatisés pour soulager les contrôleurs aériens. C'est le cas de l'interprétation des plans de vol notamment, et de l'impression des fiches de progression.

Il a donc fallu ajouter, au système, des fonctions et des bases de données qui facilitent le décodage des informations.

S. Senn/EGS

L'une des principales **bases de données** contient la **description géographique** complète des différentes zones de circulation aérienne intégrant les routes aériennes, tous les FIX désignés par leurs noms et leurs coordonnées, les parcours d'approche des pistes des aéroports ainsi que les parcours de décollage, les paramètres des pistes, les caractéristiques concernant le trafic sur les différents segments de route, la liste des régions de vol limitrophes et les FIX correspondants.

Ces informations doivent être facilement modifiables pour tenir compte des aménagements apportés, des ouvertures de nouvelles routes aériennes, etc.

Une autre base de données porte sur les courriers consignant sur 6 mois (il s'agit des RPL, Repetitive Plan) la route et les horaires prévus. Ces prévisions constituent près de 50% des vols quotidiens à contrôler. La base de données est balayée à intervalles réguliers (en général toutes les heures) pour détecter l'entrée d'avions dans la région de vol.

Les plans de vols des 50% restants — hors RPL, donc — sont fournis par le réseau AFTN, connecté à l'ordinateur. Le système lit les télex, les interprète et corrige les éventuelles erreurs. Les messages non corrigés sont présentés à l'opérateur sur l'écran, pour qu'on y apporte les modifications nécessaires (voir photo p. 1517).

Les RPL et les plans de vol AFTN servent d'entrées à une deuxième fonction : la **reconstitution automatique de la route.**

Celle-ci doit, à partir des données géographiques, sélectionner, dans l'ensemble de la route du plan de vol, la partie concernant le contrôleur de la région et la traduire en une suite de FIX à survoler.

Le recours à d'autres bases moins importantes (vitesses et caractéristiques des avions) et aux indications horaires portées dans les plans de vol permet de calculer, pour chaque FIX, le temps de survol prévisible.

Les résultats sont de nouveau traités par la fonction qui pilote les imprimantes (production des fiches de progression tout au long de la journée).

La troisième phase d'automatisation consiste à réunir les deux niveaux précédents en un seul système. Les informations proviennent simultanément **en direct** (du radar) et **par prévision** (des routes déduites du plan de vol). Ce dernier pas n'est possible qu'à condition de disposer d'un gros ordinateur capable d'effectuer des calculs à très grande vitesse.

S. Senni/E.G.S

Salle des opérations de contrôle aérien de Rome-Ciampino.

Certaines informations importantes, comme les temps de survol des FIX théoriques et réels et les prévisions de route, sont présentées aux contrôleurs sur des écrans spéciaux appelés EDD (Electronic Data Display) et placés à côté de l'écran radar afin de compléter les images du PPI (voir graphique p. 1517).

En reliant la base de données radar à celle des routes reconstruites, on peut comparer la position effective de l'avion à sa position théorique d'après la route déclarée, ce qui permet, par conséquent, de déclencher une **alarme en cas d'incohérence.**

Entrées du tracking et vitesse sont ensuite visualisées sous forme numérique sur les EDD, à côté des données déduites du plan de vol, pour fournir au contrôleur, et de façon compacte, le plus grand nombre d'informations.

Cette troisième phase d'automatisation ouvre un champ d'application extrêmement vaste, avec notamment une gamme de commandes permettant d'interroger le système.

Le contrôleur peut ainsi modifier les bases de données pour tenir compte de mises à jour des plans de vol, demander un nouveau calcul des routes particulières et réimprimer les fiches de progression.

La sécurité apportée par l'automatisation a poussé de nombreux pays à adopter le système de CTA.
En Europe, c'est l'échange d'informations entre ordinateurs de différents pays qui assure une nouvelle dimension au contrôle de la navigation aérienne.
L'OACI ou ICAO (International Civil Aviation Organization) a déjà prévu des protocoles pour l'échange normalisé de messages entre ordinateurs, le but étant de construire un réseau informatique spécialisé qui remplacerait l'ancien AFTN.
De telles ***interconnexions*** *pourraient étendre le concept de régulation de la navigation à l'****ensemble du trafic européen*** *en coordonnant les éventuelles modifications de route vers des destinations autres que celles initialement prévues (et non plus en se bornant à chercher une solution à l'intérieur d'une seule FIR ce qui n'est pas toujours aisé, surtout pour des raisons relevant de la météo).*
Ce projet ambitieux, n'a pu, jusqu'ici, être réalisé qu'aux Etats-Unis où une organisation unique, la Federal Aviation Administration, coordonne le contrôle des 51 états.
Du point de vue de la configuration du système, les automatisations radar nécessitent l'usage de ***mini-ordinateurs*** *(les 16 bits peuvent suffire), sans trop de mémoire de masse. En effet, le travail de l'ordinateur est limité aux calculs sur un nombre de données restreint, les données radar étant déjà traitées une première fois par des équipements spécialisés. Le traitement des plans de vol et des plans répétitifs (RPL) n'exige pas non plus de gros ordinateurs, mais d'****importantes bases de données*** *sont indispensables pour stocker les informations géographiques, les caractéristiques des avions et les plans de vol.*
Si l'analyse des RPL et l'impression des fiches de progression sont réalisées séparément du traitement des données radar, il est possible de stocker l'ensemble des données sur disque, car les temps d'E/S ne sont pas critiques.
En revanche les niveaux de contrôle intégré requièrent des sytèmes plus complexes pour augmenter la vitesse de calcul et la capacité mémoire.
Ils constituent une application typique du ***traitement en temps réel,*** *puisqu'ils doivent acquérir périodiquement les tracés et les traiter de telle sorte qu'ils soient présentés de façon synchronisée avec la rotation des radars (moins de 10 secondes par rotation).*
Pour cela, les performances des ordinateurs doivent atteindre 1 Mips (million d'instructions par seconde).
Pour réduire les temps morts d'E/S, les bases de données sont directement chargées en mémoire au démarrage du système, même si cela implique une occupation de 1 à 2 Moctets en mémoire centrale. C'est d'ailleurs le ***coût des mémoires*** *qui explique le retard pris au cours des décennies passées dans le développement de ces systèmes.*
Aujourd'hui, on s'achemine vers une gestion, par l'ordinateur central de la base contenant les données dynamiques sur le trafic, du dialogue avec le contrôleur et du tracking.
C'est un réseau de mini-ordinateurs qui établit les plans de vol, les fiches de progression et commande l'affichage sur les PPI et les EDD.
Les éléments du système sont dédoublés pour éviter les risques de pannes, toujours possibles. Elles sont de trois sortes : pannes d'alimentation, pannes de matériel et pannes des logiciels.
Le premier type couvre les coupures d'électricité. On a alors recours à des groupes électrogènes (eux aussi dédoublés) complétés de batteries de secours pour suppléer au temps mort entre l'interruption de l'alimentation secteur et le démarrage des groupes.
Il suffit, en effet, de quelques dixièmes de seconde pour perdre les informations contenues dans les mémoires.
Le deuxième type de panne concerne l'ordinateur. Le redémarrage est généralement plus long (environ 1 minute). Pour qu'aucune information entrée par les contrôleurs ne soit perdue, une ***copie sur disque*** *des bases de données contenues dans la mémoire centrale est effectuée lors de l'envoi de chaque commande.*
Quand un ordinateur primaire tombe en panne, l'ordinateur secondaire (qui peut accéder aux mêmes disques) en est informé et charge, lors de son démarrage, toutes les informations relatives au trafic en cours.
En cas de panne des logiciels, les informations sont également enregistrées sur disque.
En Italie, la mise au point d'un système de contrôle aérien automatisé remonte à la fin des

années 60, date de la commande, par les militaires, du système ATCAS (Air Traffic Control Automatic System), résultat de la collaboration entre Selenia et IBM, qui devait permettre de contrôler la région aérienne de Rome.
Le programme de développement prévoyait la fourniture en 10 ans (de 1975 à 1985) de tous les niveaux d'automatisation nécessaires. A partir de 1980, la société Datamat Ingegneria dei Sistemi, de Rome, s'est jointe au consortium pour concevoir les logiciels des derniers niveaux d'automatisation.
Dans sa configuration finale, le système prévoit l'emploi d'un gros ordinateur central du type IBM 3033, qui a une vitesse d'exécution de 4,5 mips. Il est connecté à deux GP-16, à un CDG (Selenia) et à un deuxième IBM 3033 servant d'« esclave » (graphique ci-contre).

Les principaux périphériques gérés par l'ordinateur central sont les suivants :

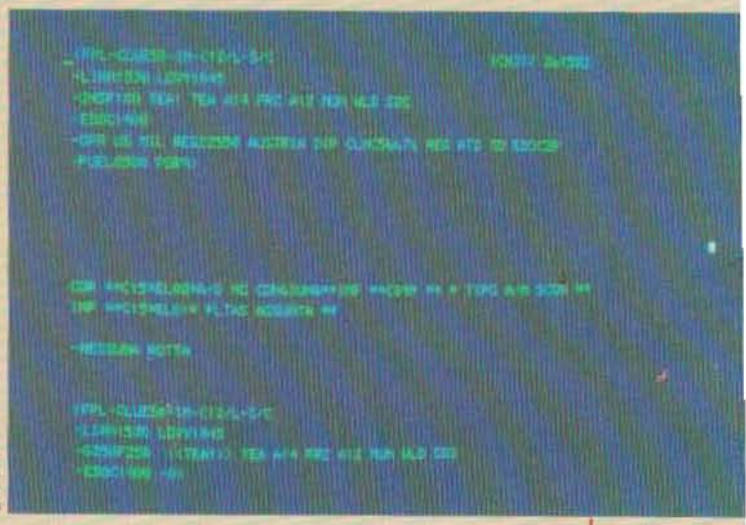
- ***Unités de disque** de 400 Moctets chacun pour enregistrer la situation en cours et les « repetitive plans »*

- ***Dérouleurs de bande magnétiques** pour l'enregistrement légal de toutes les communications entre les divers contrôleurs et le système*

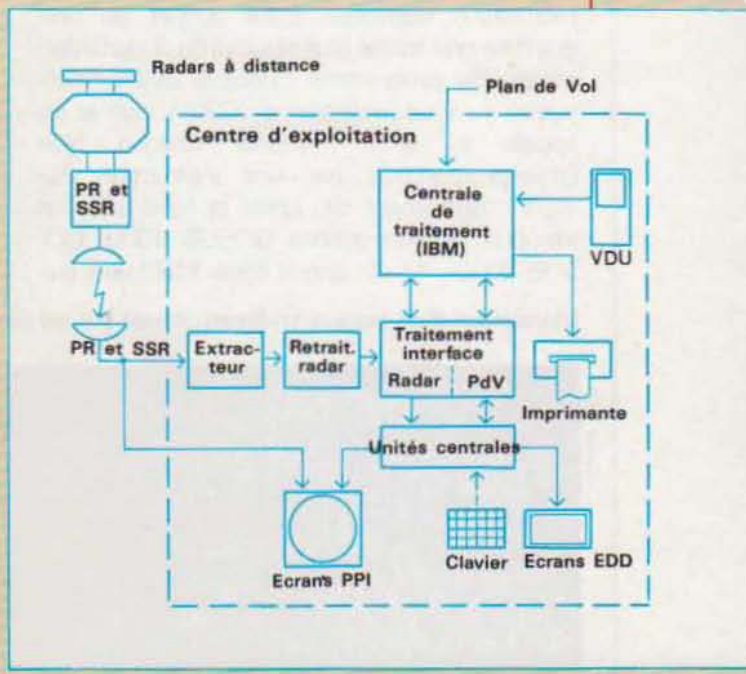
- ***Imprimantes** permettant d'imprimer les fiches de progression et de conserver une copie papier des télex reçus du réseau AFTN*

- ***Unités vidéo** de correction des messages télex et d'entrée des plans de vol ou des repetitives plans*

De plus, l'ordinateur est ***directement connecté au réseau AFTN*** *afin de recevoir l'ensemble des messages concernant les plans de vol.*
La gestion des unités de visualisation (PPI/EDD) dont disposent les contrôleurs est effectuée par les GP-16, qui reçoivent de l'ordinateur central les données traitées. Les mini-ordinateurs font également fonction d'interface entre les opérateurs et le maître en gérant les claviers ordinaires ou les dispositifs d'entrée rapides.



B. Liotta



En haut : télex transmis par l'AFTN. En-dessous, schéma du système de collecte des données.

En cas de panne, qu'elle soit matérielle ou logicielle, de l'IBM 3033, il est prévu que le CDG assurera l'étiquetage des données radar pendant la minute nécessaire au redémarrage automatique.

Umberto Corà et Bruno Liotta,
Datamat Ingegneria dei Sistemi, Rome.

L'écran couleur permet d'associer une couleur différente à chaque mode de représentation. On trace tous les histogrammes sous forme de colonnes pleines dont seule la couleur varie. Avec l'écran monochrome, il faut remplir différemment les colonnes.

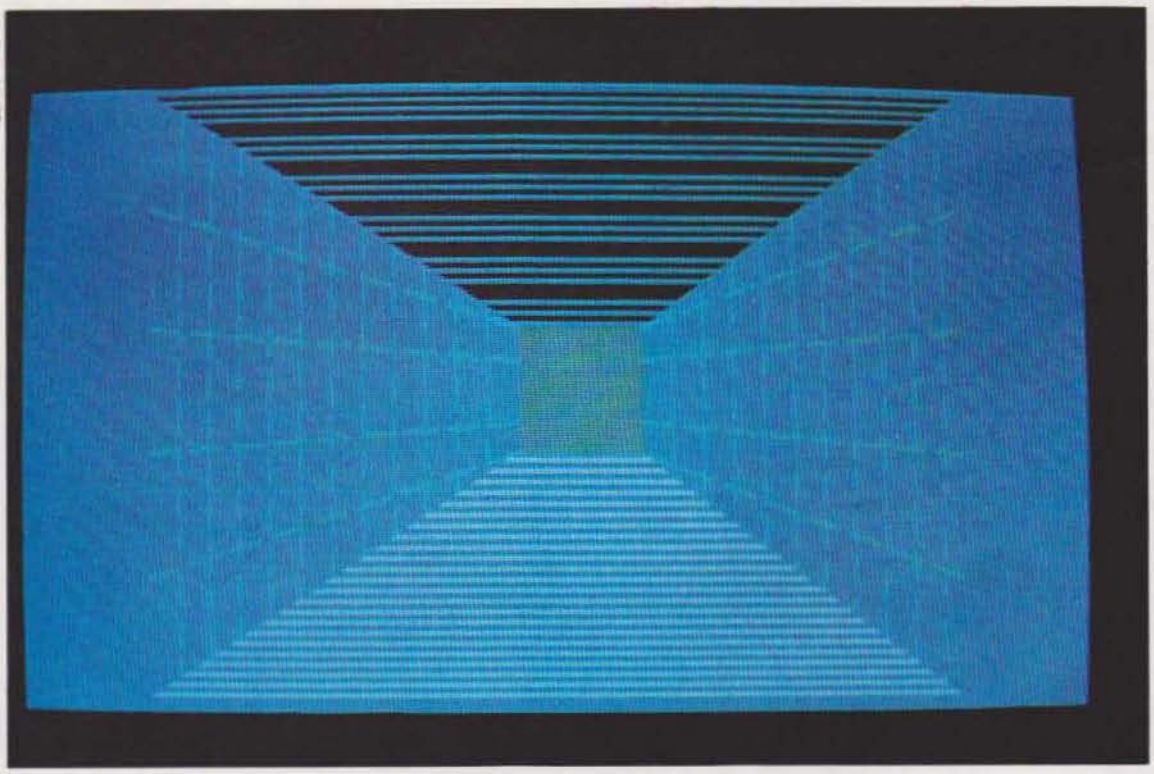
Le programme est divisé en deux parties : la première est consacrée à l'entrée des données, la seconde à leur présentation. L'entrée est insérée dans le programme principal, pour autoriser toutes les adaptations.

La présentation comporte des sous-programmes spécialisés chacun dans une tâche spécifique. La subdivision en sous-programmes ne se retrouve pas dans le listing car les fonctions se suivent : mais les organigrammes (pages 1519-1522) reprennent la numérotation habituelle. Pour donner au programme une forme plus structurée, il faut interrompre le programme principal avant l'exécution du sous-programme 1000, y insérer les appels et enfin terminer chaque bloc (sous-programme) par une instruction RETURN. Autrement dit, après la ligne 380, on introduit les instructions GOSUB 1000, GOSUB 2000... tandis que la ligne 1180 sera suivie de RETURN (de même que la 2085, etc.). Chaque bloc est alors considéré comme un sous-programme.

Le programme contient également les sous-programmes de visualisation des caractères alphanumériques ; les données correspondantes sont initialisées à partir des lignes 60 et 65. Ensuite vient la phase d'entrée des données, au cours de laquelle est effectué un contrôle de validité. Si on a choisi d'entrer les valeurs en pourcentage, aucun chiffre ne peut dépasser 100 (100 %) pas même la somme des valeurs entrées (lignes 210). Noter que le sous-programme 6000 contient simplement l'affichage du diagnostic. La sortie du programme est réalisée en entrant un nombre de données égal au maximum prévu (NM = 20, ligne 20) ou une valeur négative. La limite maximale de 20 données, correspondant à 20 colonnes à répartir sur l'axe X, est due aux dimensions de l'écran utilisé et aux formes particulières de remplissage des histogrammes.

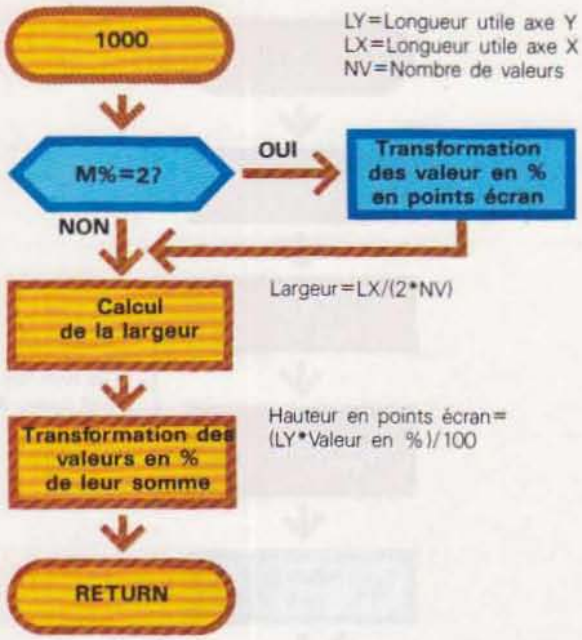
Le transfert de ce programme sur d'autres machines nécessite certaines modifications d'instructions et notamment :

Simulation d'un espace tridimentionnel sur un moniteur couleur.

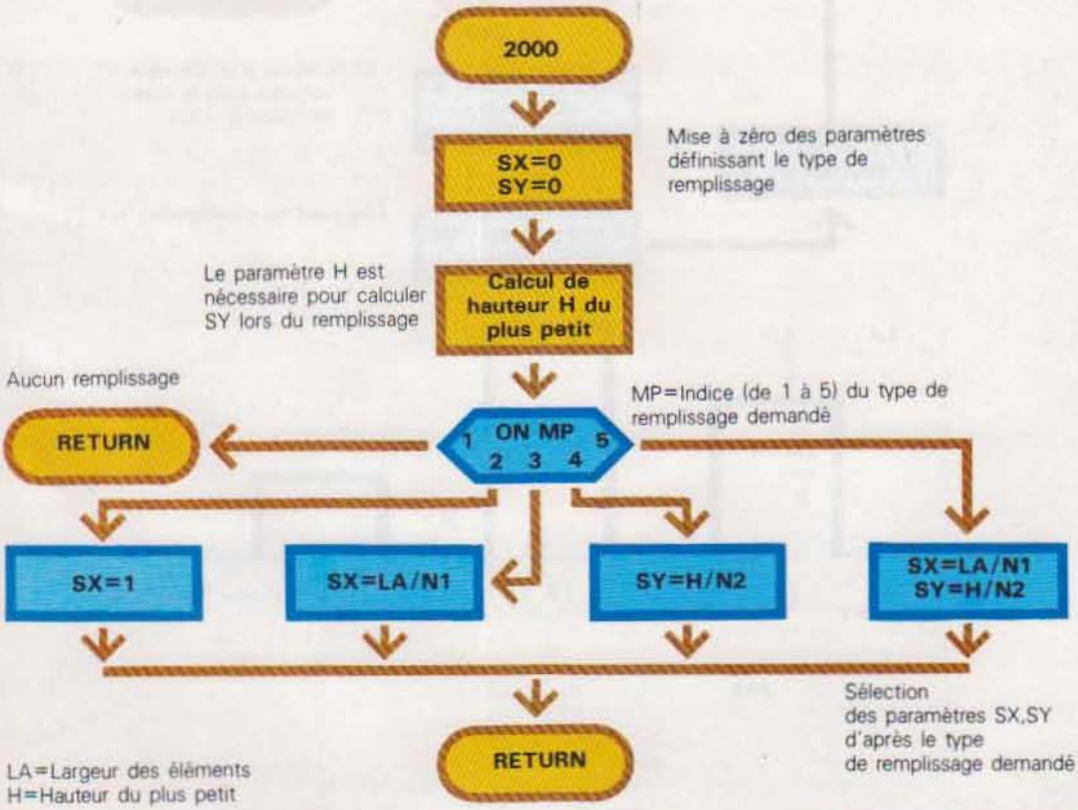


Centro THC

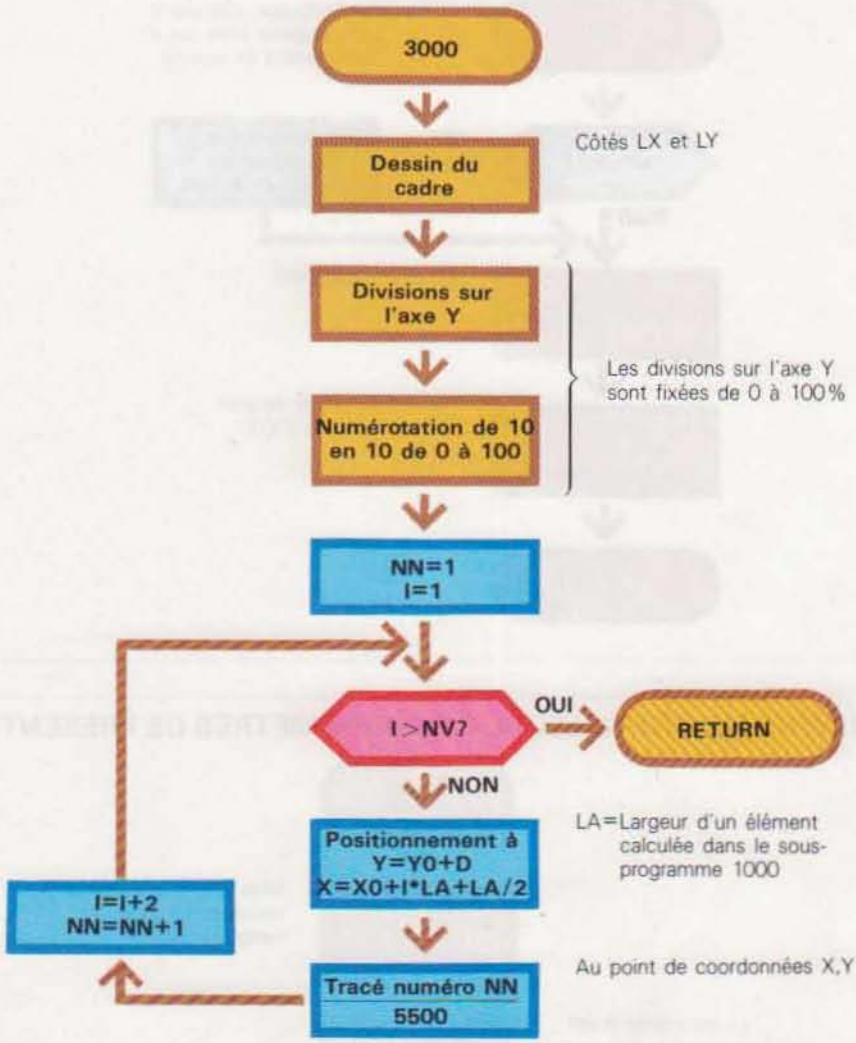
1/ TRAITEMENT DES DONNEES POUR L'HISTOGRAMME



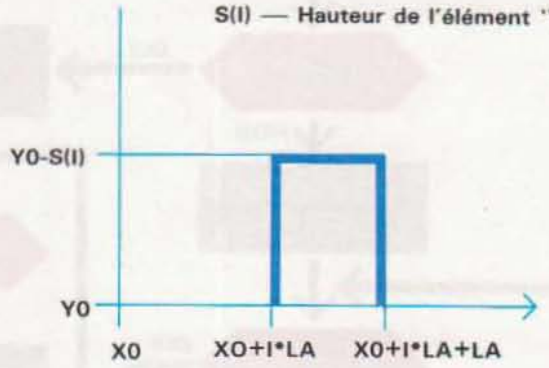
2/ ASSIGNATION DE VALEURS AUX PARAMETRES DE PRESENTATION

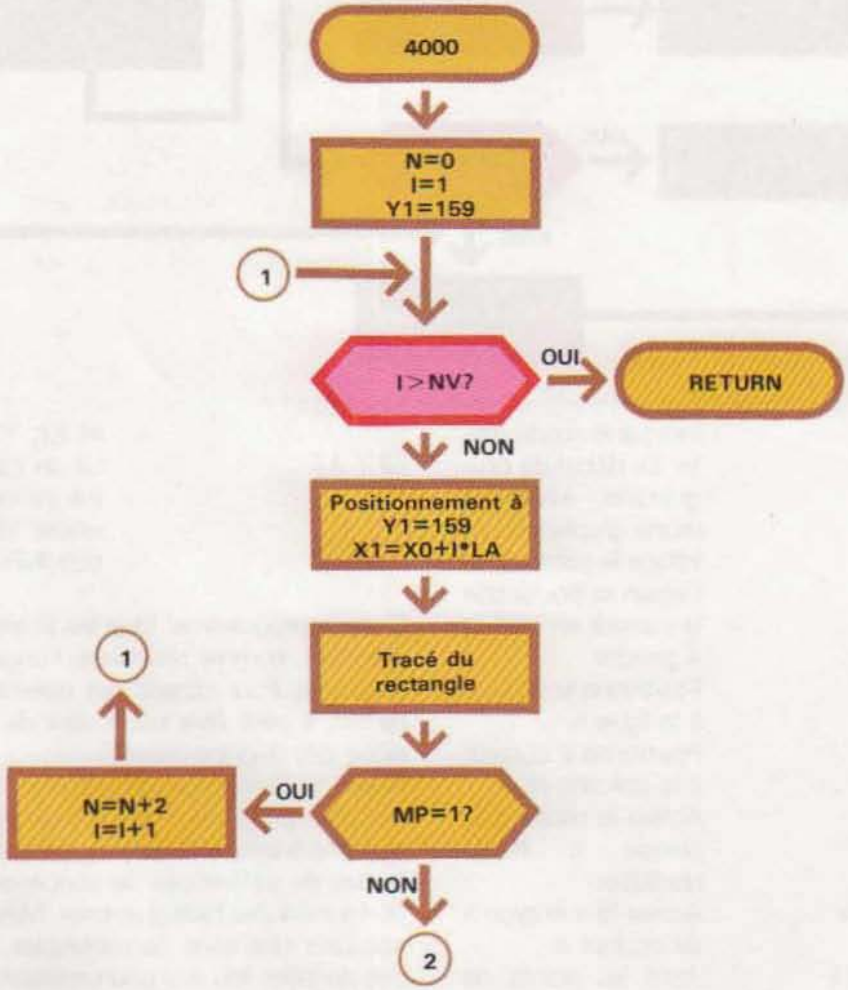


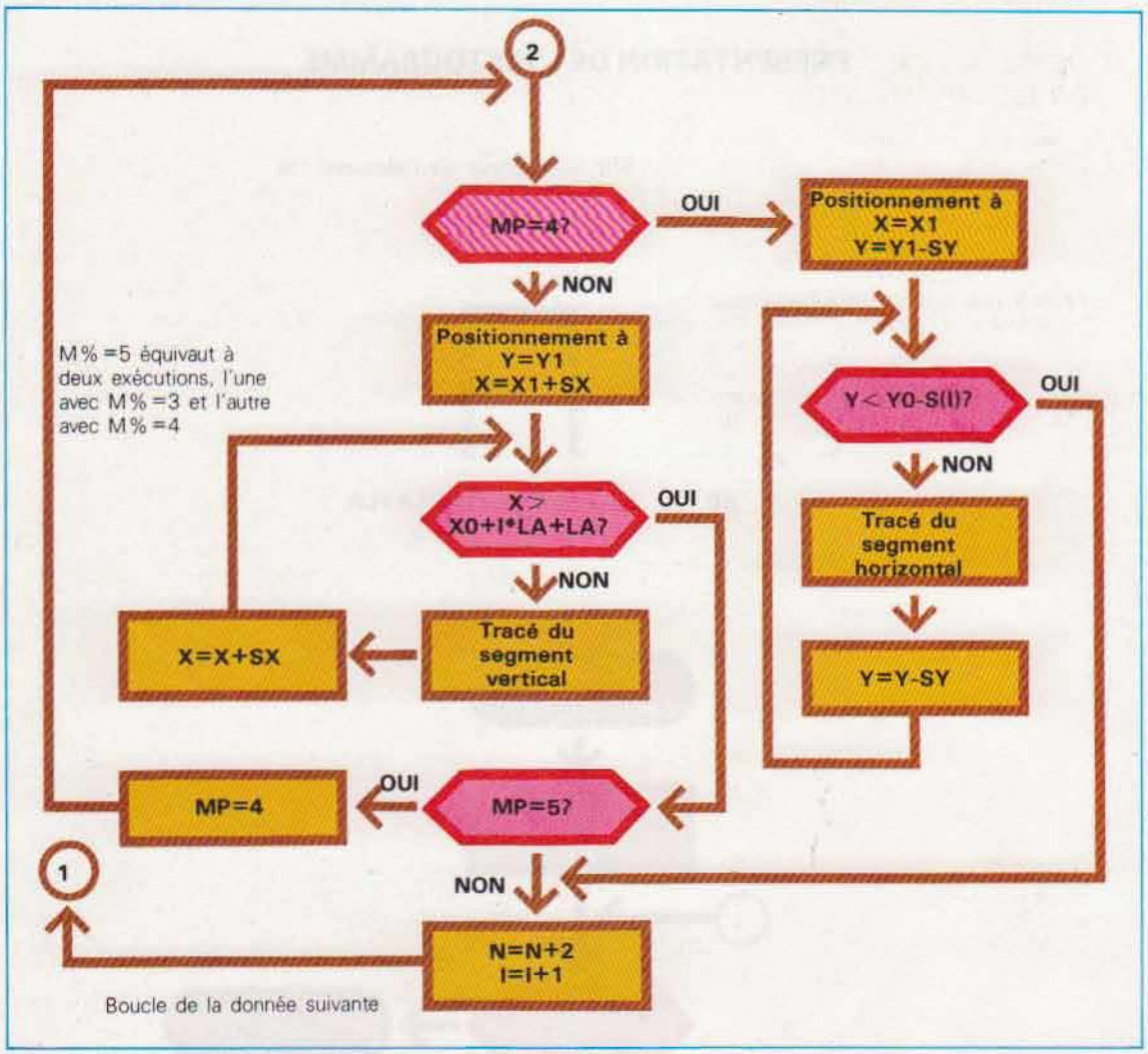
PRESENTATION DE LA FENETRE



PRESENTATION DE L'HISTOGRAMME







TEXT : Indique le mode texte. En début de programme, inhibe le mode graphique.

HOME: Efface le contenu de l'écran et positionne le curseur en haut et à gauche

VTAB n : Positionne le curseur à la ligne n

HTAB m : Positionne le curseur à la colonne m

HGR2 : Active le mode graphique à haute résolution

HCOLOR = n : Active le « crayon » de couleur n

HPLOT X1, Y1 TO X2, Y2 : Joint les points de coordonnées X1, Y1 et X2, Y2.

GET A$: Lit un caractère entré au clavier. Equivalent de l'instruction INPUT $ (1).

Dans le programme, tous les positionnements prennent, comme référence, l'origine, en haut à gauche. Pour obtenir des orientations différentes, il peut être nécessaire de modifier le signe des déplacements.

Pages 1525 et 1526, on trouvera l'organigramme d'un programme de visualisation de diagramme à secteurs. Ce programme ne présente pas de différences de conception par rapport à celui des histogrammes. Mais, au lieu de visualiser une série de rectangles aux valeurs des données (ou aux pourcentages de ces valeurs par rapport au total), le programme doit

REPRESENTATION D'HISTOGRAMMES

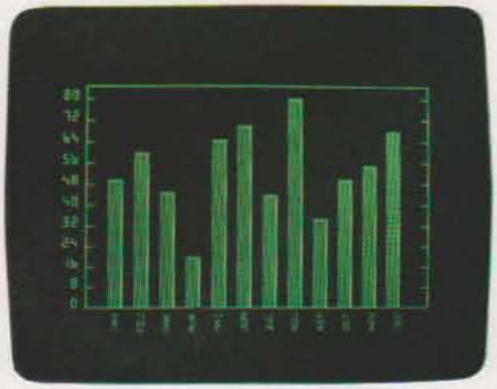
Quelques phases de fonctionnement du programme de représentation des histogrammes.

Après la question de la ligne 80, à laquelle l'utilisateur a répondu en indiquant qu'il entrerait des données absolues (et non des pourcentages), le programme est passé à la phase d'entrée des données à représenter. Ici, l'utilisateur a entré la valeur 50 comme première donnée. Elle doit être marquée à l'aide de l'étiquette JAN.

Les lignes 310 à 360 provoquent, au terme de la phase d'entrée, l'affichage du menu des différentes formes de représentation.
L'opérateur à choisi la représentation à colonnes remplies de lignes verticales.

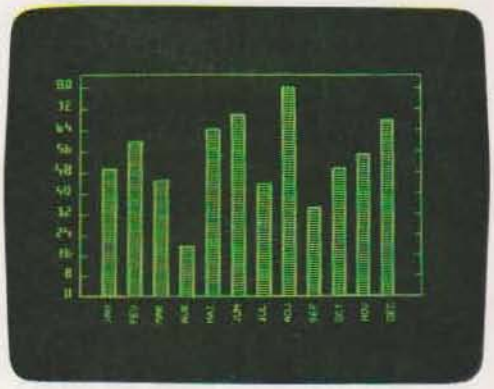
Le programme demande ensuite à l'utilisateur s'il désire représenter les données sous forme de pourcentage.
La réponse est négative.

Le programme a activé la représentation en lignes verticales des valeurs absolues entrées.

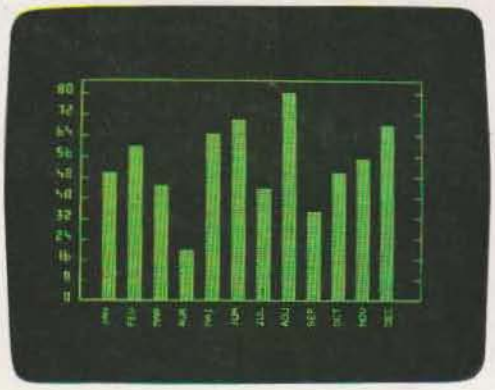


L'écran ci-contre et les 3 autres illustrent les différents modes de visualisation.

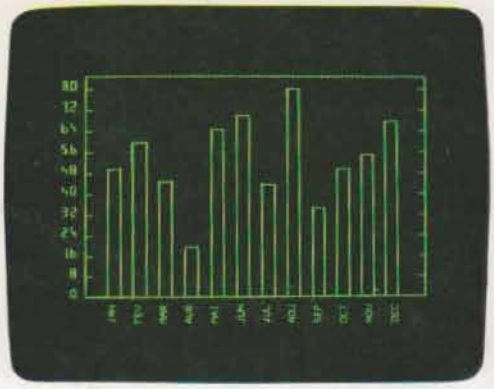
Lignes horizontales...



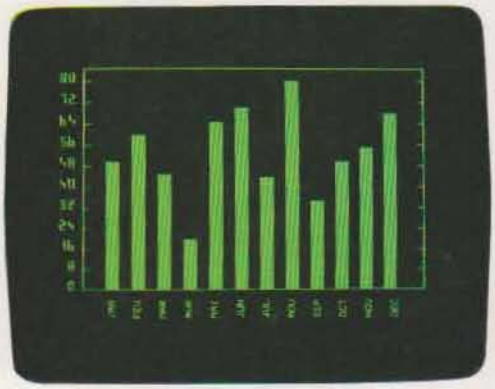
Quadrillage...



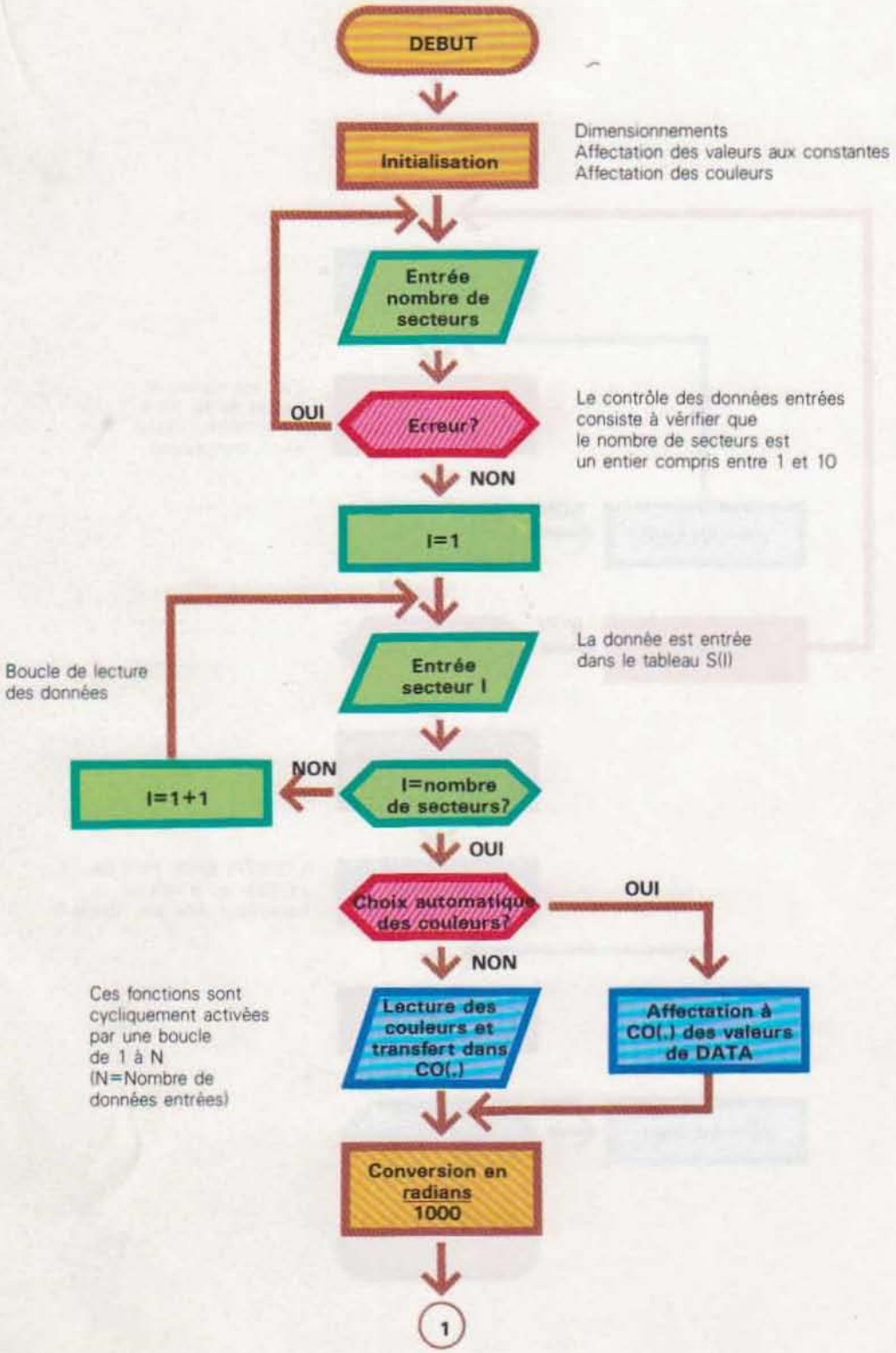
Colonnes vides...

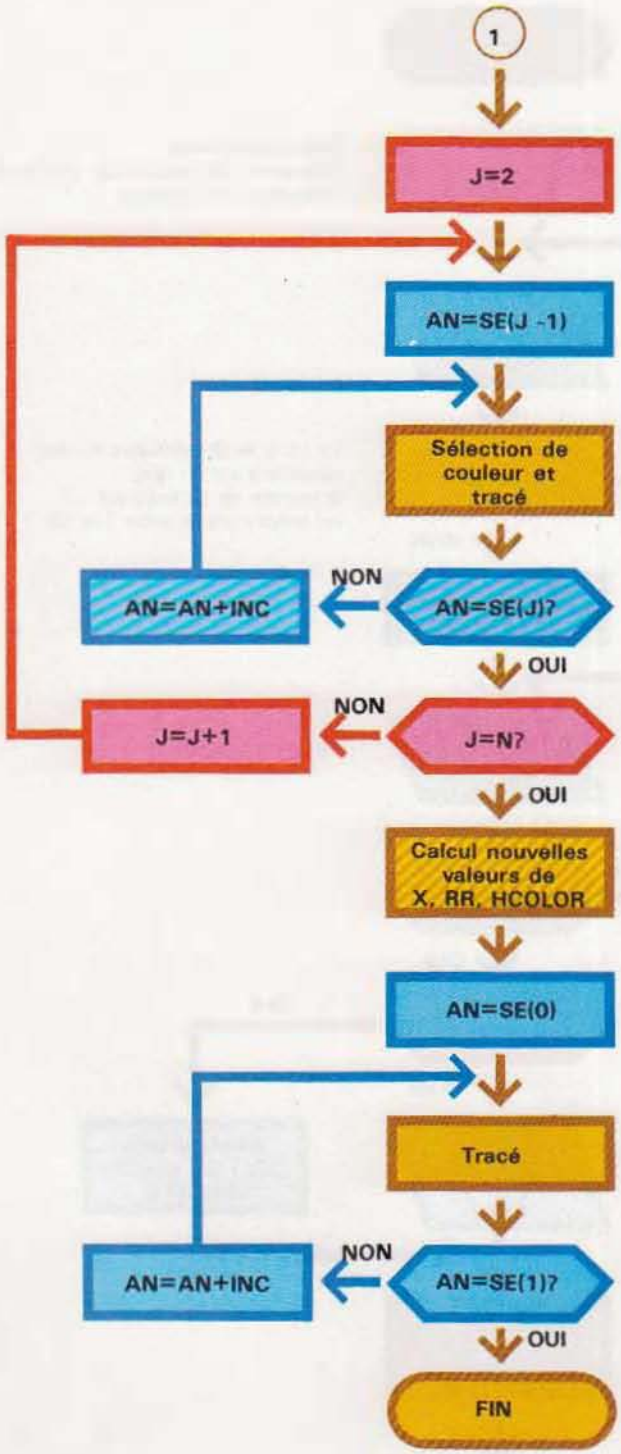


Colonnes pleines...



PROGRAMME DE PRESENTATION DE DIAGRAMMES SECTEURS




1
J=2
AN=SE(J -1)
Sélection de couleur et tracé
L'arc est obtenu en traçant de Xc, Yc à
X=Xc+RR*COS(AN)
Y=Yc-RR*SIN(AN)
AN=AN+INC
NON
AN=SE(J)?
OUI
J=J+1
NON
J=N?
OUI
Calcul nouvelles valeurs de X, RR, HCOLOR
AN=SE(0)
L'OPTION BASE 1 n'a pas été prévue ; le tableau commence donc par l'indice 0
Tracé
AN=AN+INC
NON
AN=SE(1)?
OUI
FIN

VISUALISATION DE DIAGRAMMES A SECTEURS

```
10 REM ---------------------
20 REM DIAGRAMMES A SECTEURS
30 REM ---------------------
40 REM
50 REM
60 XC=130
   :YC=90
   :R=90
   :PI=3.1415926
   :INC=PI/300
70 DIM SE(10),C(10),CO(10),S(10)
80 FOR I=1 TO 10
   :READ C(I)
   :NEXT
90 DATA 5,6,7,1,2,3,5,6,7,1
100 REM
110 REM ------------------
120 REM ENTREE DES DONNEES
130 REM ------------------
140 REM
150 HOME
160 VTAB 4
    :INPUT "Nombre de secteurs (max. 10) "
    ;A$
170 A=VAL(A$)
180 IF A<1 OR A>10 OR INT(A)<>(A) THEN 150
190 VTAB 6
    :PRINT "Entrez les données (nombres
     positifs) de chaque secteur."
200 N=A
    :TT=0
210 FOR I=1 TO N
220 VTAB (10+I)
    :PRINT "Secteur"I;
    :INPUT ": ";A$
230 S(I)=VAL(A$)
    :IF S(I)<0 THEN 220
240 TT=TT+S(I)
    :S(I)=TT
250 NEXT
260 HOME
270 VTAB 4
    :PRINT "Le diagramme contient 'n'
     secteurs."
280 REM
290 REM ----------------------
300 REM SELECTION DES COULEURS
310 REM ----------------------
320 REM
330 VTAB 6
    :INPUT "Voulez-vous choisir vous-même
     les couleurs ou que l'ordinateur s'en
     charge (1/2) ";A$
340 IF A$<>"1" AND A$<>"2" THEN 260
350 IF A$="1" THEN 390
360 FOR I=1 TO N
    :CO(I)=C(I)
    :NEXT
370 IF CO(N)=5 THEN CO(N)=1
380 GOTO 550
390 HOME
400 VTAB 4
    :PRINT "Choisissez entre vert, rouge,
     orange, bleu et blanc en tapant <V>,
     <R>, <A>, <B> ou <W>."
410 FOR I=1 TO N
420 VTAB (6+I)
    :PRINT "Couleur du secteur"I;
    :INPUT ": ";A$
430 IF A$="V" THEN CO(I)=1
    :GOTO 490
```

```
440 IF A$="R" THEN CO(I)=2
    :GOTO 490
450 IF A$="A" THEN CO(I)=3
    :GOTO 490
460 IF A$="B" THEN CO(I)=4
    :GOTO 490
470 IF A$="W" THEN CO(I)=5
    :GOTO 490
480 GOTO 420
490 NEXT
500 REM
510 REM ------------------------
520 REM PRESENTATION DU SECTEUR
530 REM ------------------------
540 REM
550 HOME
560 VTAB 4
    :INPUT "Voulez-vous faire ressortir le
     secteur 1 (O/N) ";A$
570 FL=0
    :IF LEFT$(A$,1)="O" THEN FL=1
580 GOSUB 1000
590 REM
600 REM --------------------
610 REM DESSIN DU DIAGRAMME
620 REM --------------------
630 REM
640 HGR2
650 FOR J=2 TO N
660 FOR AN=SE(J-1) TO SE(J) STEP INC
670 HCOLOR=CO(J)
680 HPLOT XC,YC TO XC+R*COS(AN),YC-R*SIN(AN)
690 NEXT AN,J
700 X=XC+15*FL
    :HCOLOR=CO(1)
    :RR=R-8*FL
710 IF SE(1)>.5 THEN RR=R-3*FL
    :X=XC+12*FL
    :IF SE(1)>1 THEN RR=R
    :XC=XC+9*FL
720 FOR AN=SE(0) TO SE(1) STEP INC
730 HPLOT X,YC TO X+RR*COS(AN),YC-RR*SIN(AN)
470 NEXT
750 END
1000 REM
1010 REM ----------------------
1020 REM CONVERSION EN RADIANS
1030 REM ----------------------
1040 REM
1050 IF TT=360 THEN 1090
1060 FOR I=1 TO N
1070 S(I)=S(I)/TT*360
1080 NEXT
1090 FOR I=1 TO N-1
1100 SE(I)=S(1)-S(1)/2
1110 NEXT
1120 SE(0)=-S(1)/2
     :SE(N)=360+SE(0)
1130 FOR I=0 TO N
1140 SE(I)=SE(I)/180*PI
1150 NEXT
1160 RETURN
```

visualiser un cercle divisé en un nombre de secteurs égal à celui des données à représenter. Il attribue, à chaque secteur, une surface proportionnelle à la donnée représentée.

Le programme (listing pages 1527 et 1528) prévoit l'emploi d'un écran couleur. Les couleurs des secteurs du diagramme sont automatiquement choisies par le programme, sauf

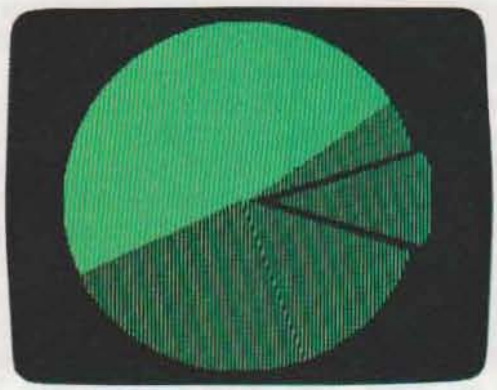
DIAGRAMMES A SECTEURS

Le programme (ligne 160) demande à l'utilisateur combien de secteurs (de données) il doit représenter.
L'entrée du nombre de secteurs (ici 5) est suivie de la phase de demande des données (lignes 190 à 250).

Au terme de l'entrée des données, le programme indique, à l'utilisateur, le nombre de données entrées (ligne 270) et lui demande des précisions sur les choix de couleurs (ligne 330).

L'utilisateur ayant opté pour la sélection automatique des couleurs, le contrôle passe à la ligne 560, qui demande s'il faut mettre en valeur le premier secteur.
La réponse est affirmative.

Présentation du diagramme à secteurs. Le secteur "découpé" est le premier ; les autres suivent dans le sens inverse des aiguilles d'un montre.



indication contraire de l'utilisateur. Seule particularité importante : le sous-programme 1000, qui convertit les valeurs entrées par l'utilisateur en valeurs angulaires (grandeurs des secteurs) exprimées en radians. Pour le reste, les instructions sont analogues à celles utilisées pour les histogrammes.

Gestion de la mémoire en mode graphique

L'écran est l'unité de sortie la plus courante dans les applications graphiques. Or ce dispositif ne garde pas en mémoire les images qui apparaissent. La visualisation d'un texte ou d'un dessin, possible à l'aide d'une seule instruction (PRINT, LINE, etc.) demande, en réalité, une activité continue de régénération de la même image, à des intervalles très courts, afin de réactiver chacun de ses points. C'est pourquoi l'image graphique est rangée dans une zone mémoire spéciale.

A des intervalles réguliers, les programmes du système prélèvent le contenu de cette zone pour le transférer sur l'écran en redessinant continuellement le graphique. Ceci vaut également pour un texte en Basic ou dans tout autre langage. Il faut donc disposer d'une zone de mémoire qui conserve l'état de chaque point de l'écran. La dimension de cette mémoire varie en fonction du type d'écran et du système d'exploitation.

Si l'écran est monochrome, un seul bit par point suffit : à 0, il signifie que le point n'est pas visible ; à 1, que le point est activé.

Avec les écrans couleur, l'extension de la mémoire spéciale est plus grande car, pour chaque point, en plus de l'état (ON/OFF), il faut définir la couleur. Pour l'affichage d'un dessin, on procède donc en deux étapes : chargement de l'image binaire en mémoire graphique, activation du processus de présentation.

Par exemple, les instructions de tracé d'un segment entre 2 points (LINE, HPLOT, etc.) calculent, à partir des coordonnées en points écran, les positions mémoire correspondantes. Pour cela, elles mettent à l'état ON tous les bits relatifs à ce segment. Dans une machine 8 bits, chaque position mémoire contrôle un maximum de 8 points écran. Si la valeur 1 est écrite dans l'une d'elles, le premier des points contrôlés est activé ; par contre, avec la valeur 3, deux points contigus (1 + 2) le seront, et ainsi de suite jusqu'à la valeur maximale prévue.

La qualité graphique d'un dessin dépend fortement de la résolution de l'écran. Plus le nombre de points est élevé, plus la zone mémoire nécessaire est importante ; aussi doit-on, même dans le cas d'une haute résolution, réduire le nombre de points gérés pour éviter l'occupation d'un trop grand espace. Un écran de 280 positions horizontales par 190 positions verticales demande environ 53 000 indicateurs d'état correspondant chacun à un bit. L'occupation mémoire est donc d'environ 7,5 Koctets, en supposant que pour chaque position, seulement 7 bits parmi les 8 possibles sont employés.

L'existence de cette zone mémoire spéciale offre des facilités de programmation remarquables. Pour construire des figures d'une certaine complexité, on peut recourir à l'accès direct aux adresses mémoire sans être limité par les instructions graphiques. Inversement, on lira aisément le contenu de la mémoire pour connaître l'état de chaque point vidéo (allumé ou éteint) et pour traiter ou stocker le dessin. Avec cette méthode, on considère le graphique comme un tableau de valeurs numériques sur lequel on opère toutes sortes de traitements. La présentation à l'écran est soit automatique (au fur et à mesure des modifications du tableau) soit sur commande (quand le mode graphique est activé). Toutefois, cette gestion n'est pas propre à toutes les machines. Dans certaines d'entre elles, la mémoire graphique n'est accessible qu'au système ; il faut donc recourir à des instructions de haut niveau.

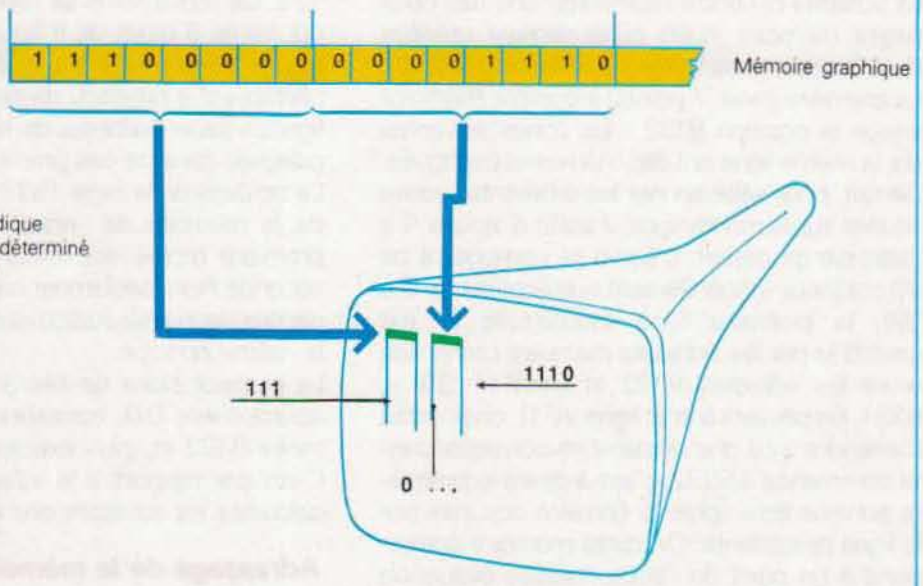
Pour d'autres machines, seule la connaissance de l'organisation interne de la mémoire permet de travailler directement sur l'image binaire.

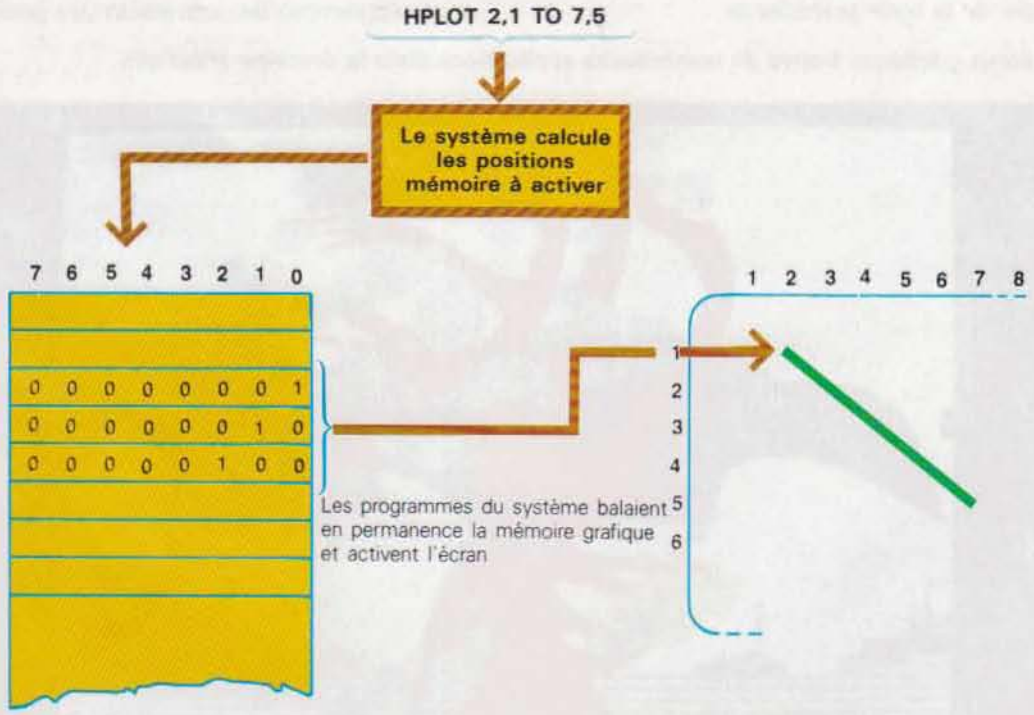
La mémoire graphique

La relation unissant un point écran à l'adresse mémoire qui le gère n'est ni simple, ni linéaire. Le mécanisme de gestion de l'écran est lié aux différents paramètres et, parfois, au type d'unité centrale utilisée. Nous allons étudier un exemple de partition de la mémoire couramment employé dans les ordinateurs personnels et dans les micros dotés d'une unité centrale Rockwell 6502 (machines du type Apple et compatibles).

La zone mémoire correspondant à l'écran n'est

GESTION SIMPLIFIEE DE L'ECRAN EN MODE GRAPHIQUE





pratiquement jamais constituée d'adresses contiguës, pour des raisons d'ordre essentiellement matériel. Ainsi, deux zones distantes de 100 positions ne correspondent pas à deux adresses ayant une différence de 100.
Le schéma ci-contre représente une des deux pages mémoire vidéo généralement utilisées dans les systèmes basés sur le Rockwell 6502. La première zone (7 points) a comme mémoire image la position 8192 ; les zones suivantes sur la même ligne ont des mémoires contiguës. De fait, pour sélectionner les différentes zones situées sur la même ligne, il suffit d'ajouter 1 à l'adresse de départ. L'écran se composant de 40 colonnes (généralement numérotées de 0 à 39), la première ligne (numérotée 0) est contrôlée par les adresses mémoire comprises entre les adresses 8192 et 8192 + 39 = 8231. En passant à la 2e ligne (n° 1), on pourrait s'attendre à ce que la mémoire correspondante commence à 8232, c'est-à-dire à la première adresse libre après la dernière occupée par la ligne précédente. Or, cette mémoire correspond à un point de l'écran localisé beaucoup plus bas. La 2e ligne (n° 1) commence en fait à l'adresse 9216, distante de 1024 de l'adresse initiale de la ligne précédente.

Comme pour les autres lignes, chaque zone de l'écran correspond à une mémoire contiguë à la précédente. L'incrément de 1024 reste vrai pour 7 lignes (nos 1 à 7), tandis que l'adresse de la 8e ligne est égale à celle de la 1re (n° 0) plus 128. Ce mécanisme se répète ainsi 7 fois, ce qui forme 8 blocs de 8 lignes chacun. La mémoire correspondant à la deuxième partie de l'écran est à nouveau divisée en 8 blocs de 8 lignes : seule l'adresse de la 1re position varie, puisque, dans ce cas précis, il s'agit de 8232. Le schéma de la page 1533 indique la partition de la mémoire de l'ensemble de l'écran. La première moitié est visualisée en totalité, la seconde étant seulement suggérée ; les mécanismes de numérotation des adresses suivent le même principe.
Le premier point de l'écran, identifié par les coordonnées 0,0, correspond à l'adresse mémoire 8192 et, plus exactement, à son bit 0. C'est par rapport à la valeur 8192 que sont calculées les adresses des autres points.

Adressage de la mémoire vidéo

Pour expliquer le mécanisme d'adressage, nous prendrons un exemple de calcul et un sous-programme de conversion des positions

Un écran graphique trouve de nombreuses applications dans le domaine artistique.

Centro THC

TOPOGRAPHIE DE LA MEMOIRE VIDEO : SCHEMA GENERAL

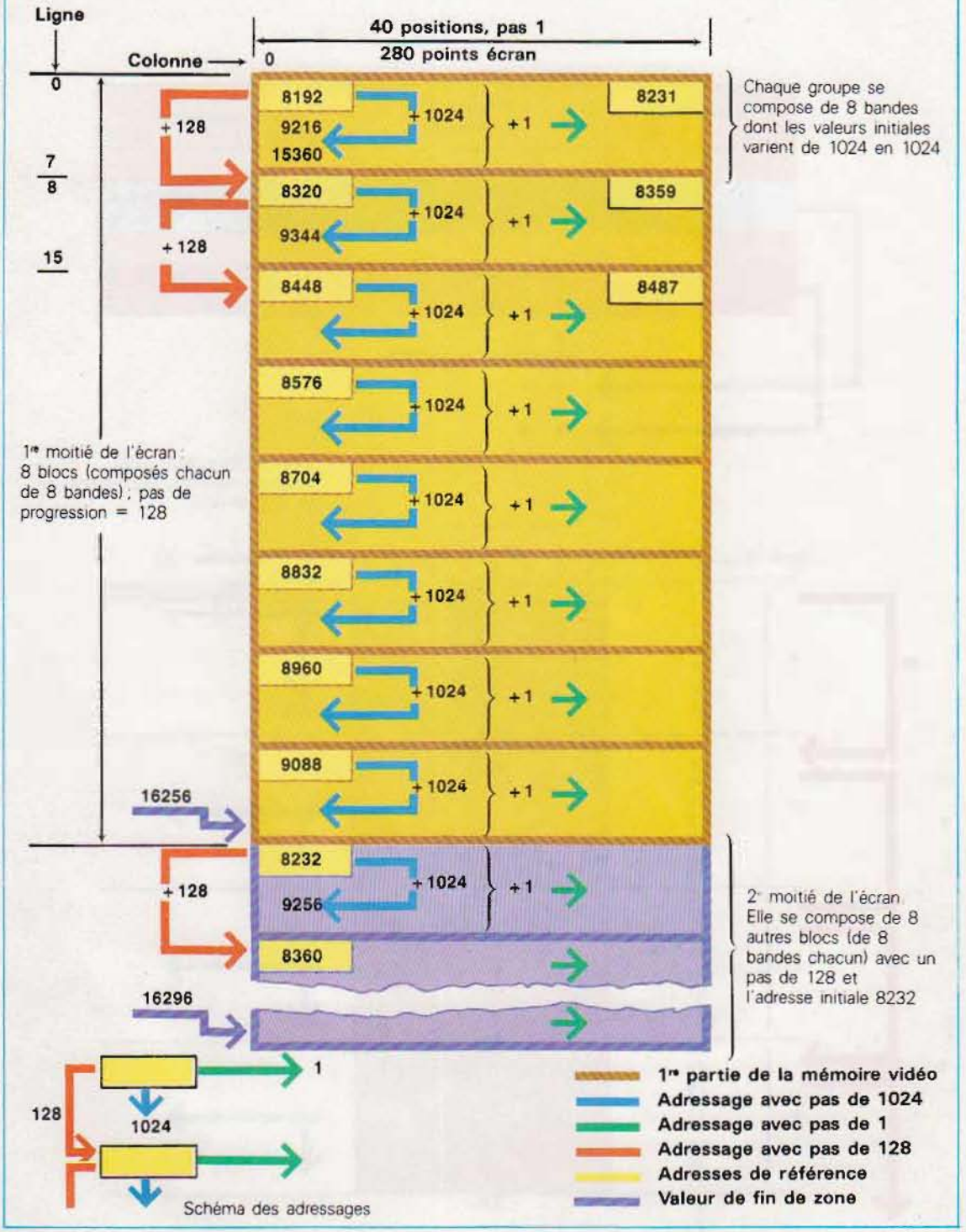


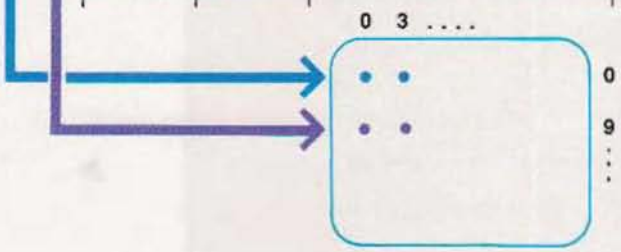
Schéma des adressages

mémoire en points écran, et inversement.

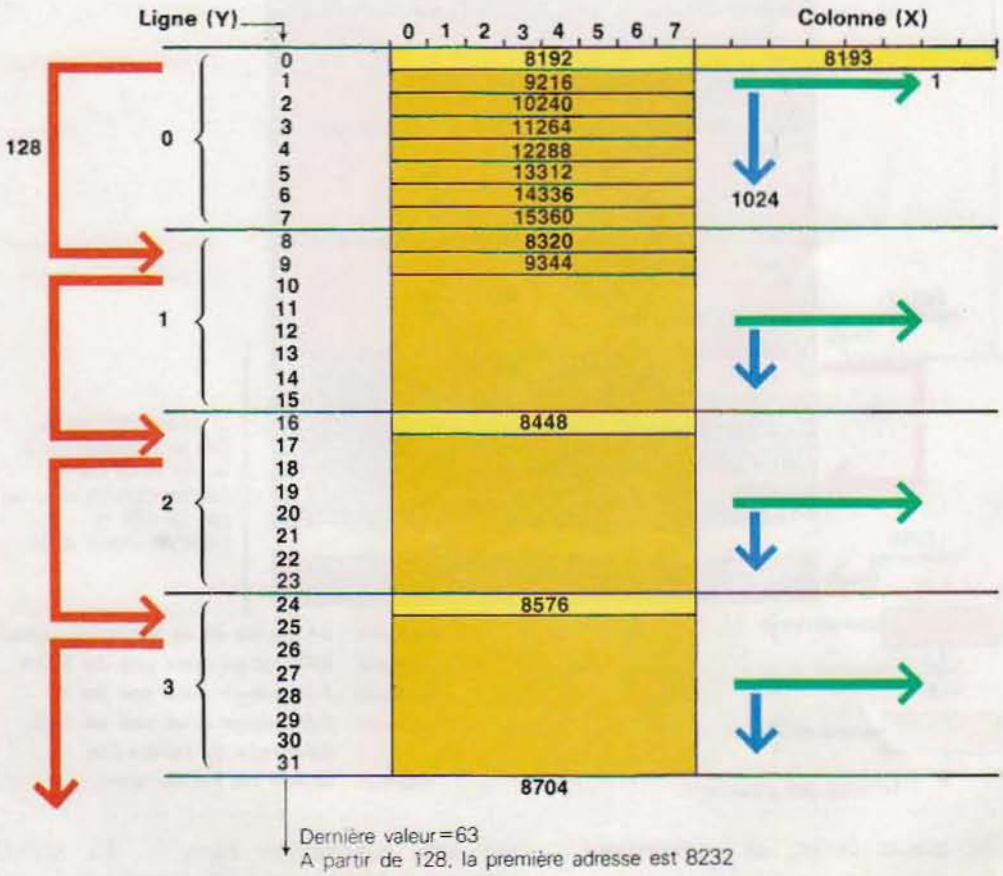
Adressage le long de l'axe. Soit l'adressage d'un point situé dans la colonne zéro (X = 0) sur une quelconque ligne Y. La variable d'entrée est l'ordonnée (Y), à partir de laquelle il faut trouver la position mémoire correspondante. Pour Y = 7, la position est 15360.

ADRESSAGE SUR L'AXE Y

Coordonnées X	Coordonnées Y	Positions mémoire	Contenu de la mémoire 0 1 2 3 4 5 6 7
3	0	8192	0 0 0 1 0 0 0 0
5	0	8192	0 0 0 0 0 1 0 0
3	9	9344	0 0 0 1 0 0 0 0
5	9	9344	0 0 0 0 0 1 0 0



Exemple de rapport entre l'adresse mémoire, son contenu et le point écran



Dernière valeur = 63
A partir de 128, la première adresse est 8232

La partie inférieure de la figure de la page 1534 reprend la partie de la figure précédente concernant les points ayant X=0. Pour des raisons de place, seuls les 4 premiers blocs sont représentés (Y compris entre 0 et 31) ; le mécanisme est identique pour les autres.
Pour déterminer la position initiale correspondant à une valeur de Y donnée, il faut :

A - Déterminer à quel bloc B (groupe de 8 lignes) elle appartient. B s'obtient en divisant Y par 8 (puisque chaque bloc contient 8 valeurs de Y) et en retenant la partie entière du résultat.

B - Déterminer l'adresse initiale du bloc B auquel Y appartient. La position initiale de chaque groupe variant de 128 en 128, l'adresse de B est donnée par la formule 8192+B*128.

C - Déterminer la position de Y dans B.

Chaque groupe commence par une valeur déterminée de Y (0, 8, 16, etc.) et se termine par une valeur +7 (puisqu'il contient 8 lignes, de 0 à 7). Une ligne Y quelconque occupe dans le bloc la position :Y-B*8.

En effet, B est son groupe d'appartenance, et le produit B*8 indique le nombre de lignes précédant la ligne Y et appartenant à d'autres blocs. La différence entre Y et cette dernière valeur fournit la position de la ligne Y dans le bloc B.

Sachant que le pas de progression dans le bloc est de 1024, pour trouver la position mémoire, il faut multiplier la valeur précédente (position de Y dans le bloc B) par 1024 et y ajouter l'adresse initiale du bloc B, calculé au point B. La deuxième moitié de l'écran, dont la mémoire commence à l'adresse 8232, peut être traitée de la même façon, en remplaçant évidemment la valeur initiale utilisée au point B par cette valeur.

TABLEAUX DES POSITIONS DE REFERENCE SUR L'AXE Y

1re partie

Base 8192 - Ordonnée (Y) entre 0 et 63

Bloc	Valeurs de Y de	à	Adresse initiale du bloc
0	0	7	8192
1	8	15	8320
2	16	23	8448
3	24	31	8576
4	32	39	8704
5	40	47	8832
6	48	55	8960
7	56	63	9088

Algorithme : Adresse mémoire=8192+128*Bloc
Bloc =INT (Y/8)

2e partie

Base 8232 - Ordonnée (Y) entre 64 et 127

Bloc	Valeurs de Y de	à	Adresse initiale du bloc
8	64	71	8232
9	72	79	8360
10	80	87	8488
11	88	95	8616
12	96	103	8744
13	104	111	8872
14	112	119	9000
15	120	127	9128

Algorithme : Adresse mémoire=8232+128*(Bloc-8)
Bloc =INT (Y/8)

Pour des raisons pratiques, on a mis en page 1535 deux tableaux, un pour chaque zone. Chacun d'eux fournit, en fonction de la valeur de Y, le bloc d'appartenance et la position initiale de celui-ci. Par exemple, la valeur Y = 35 appartient au bloc 4, qui commence à la position 8704. Il correspond à la position :

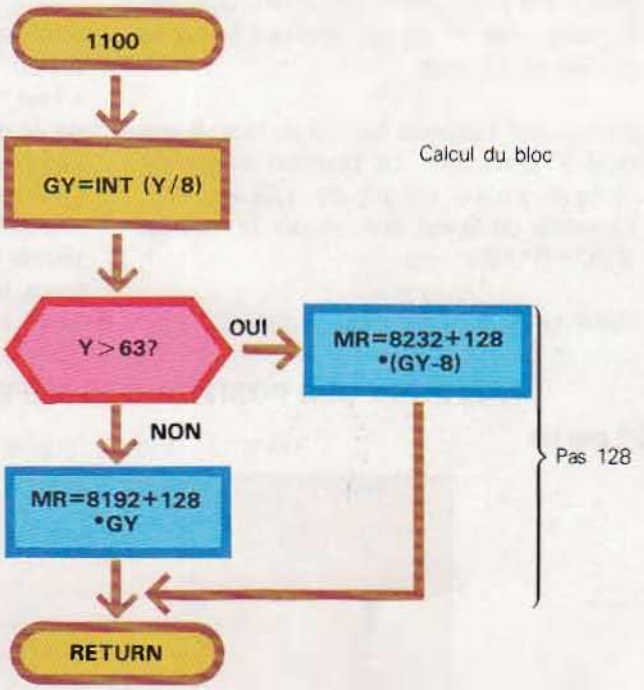
8704+1024*(35-4*8)
=8704+1024*3=11776.

Les organigrammes ci-dessous montrent le déroulement de deux sous-programmes qui, à partir d'une valeur Y trouvent l'adresse correspondante (toujours avec X = 0).

CALCUL D'ADRESSE SUR L'AXE Y

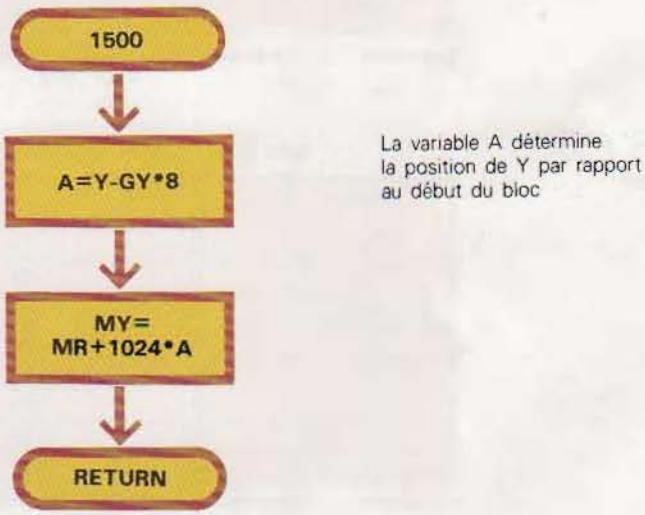
Calcul de l'adresse de référence sur l'axe Y

Entrée : Y=Valeur de l'ordonnée
Sortie : MR=Adresse de référence du bloc
BY=Bloc



Calcul de l'adresse correspondant à l'ordonnée

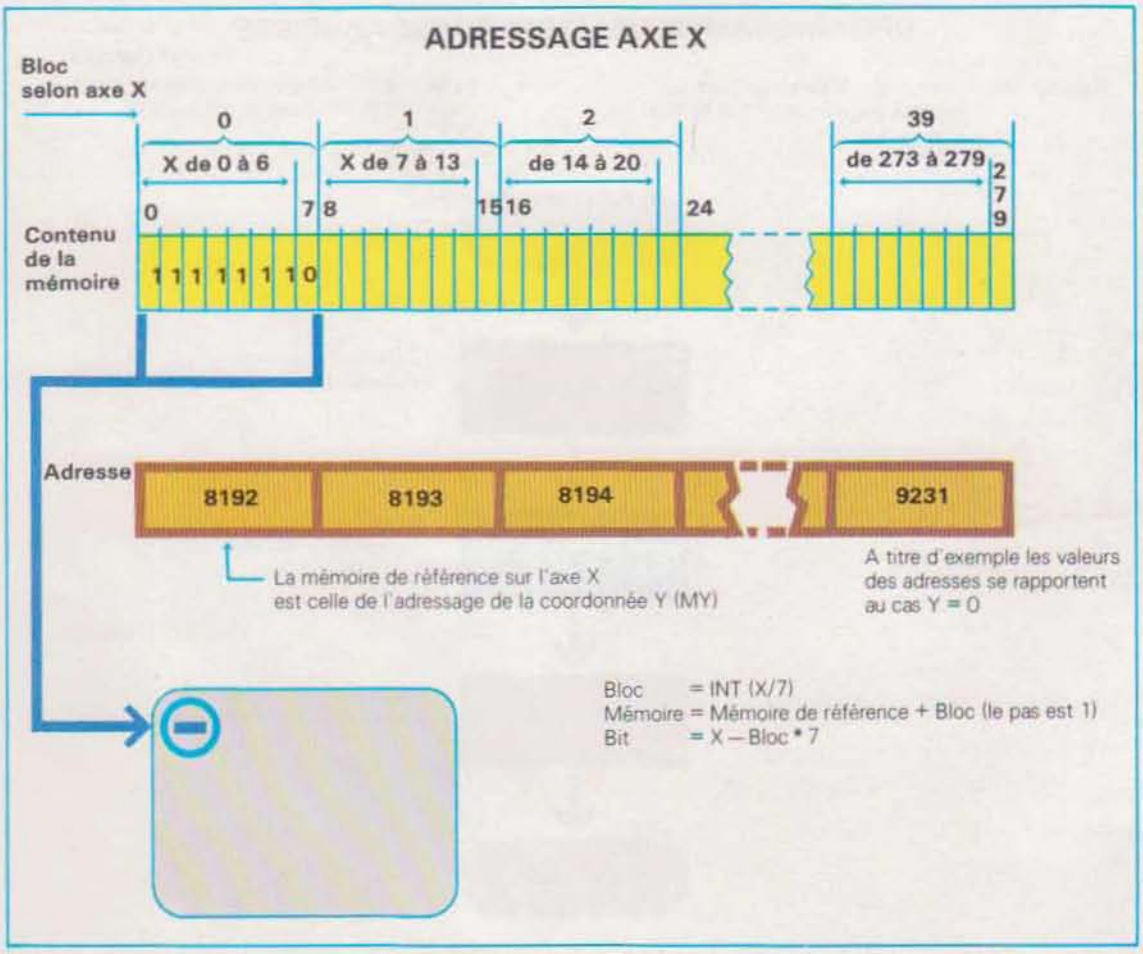
Entrées : Y=Ordonnée
MR=Adresse de référence du bloc
BY=Bloc
Sortie : MY=Adresse mémoire correspondant à Y



Dans le premier sous-programme (1100), on détermine automatiquement la valeur de référence (8192 ou 8232) en fonction de celle de Y. Les opérations, pour parvenir à l'adresse de la mémoire, sont simples, mais la technique n'est pas la plus directe.
On a choisi une méthode fractionnée pour permettre à l'opérateur d'intervenir par variation des valeurs numériques.

Adressage salon l'axe X. Pour chaque ligne (valeur de Y), les colonnes de l'écran (de 0 à 39) correspondent à des mémoires contiguës. Connaissant la valeur initiale correspondant à une ligne Y et à la colonne 0 (ou mieux à l'abscisse X = 0), les autres adresses s'obtiennent de manière séquentielle en ajoutant 1.
Chaque emplacement mémoire contenant 8 bits, il indique l'état de 8 points écran. Le bit 0 correspondant au point X = 0, le bit 1 à X = 1 et ainsi de suite.
Dans notre machine, le bit 7 (la numérotation commence à 0) n'est pas employé ; ainsi, chaque emplacement de mémoire contient l'état (ON/OFF) de 7 points (bits 0 à 6). Dans le graphique ci-dessous, on trouve les adresses des points écran appartenant à la ligne Y = 0. L'adresse de départ de la ligne est 8192. Les suivantes s'obtiennent en ajoutant 1, jusqu'à un maximum de 39, qui est le nombre de colonnes prévues (les colonnes sont, en fait, au nombre de 40, numérotées de 0 à 39). Chaque colonne est divisée en 7 points écran, pour un total de 7 × 40 = 280 points (de X = 0 à X = 279).
Pour adresser un point d'abscisse X, il faut d'abord déterminer son bloc (colonne) ; à partir de cette valeur, calculer l'adresse mémoire et enfin la position du bit dans la mémoire.
Par exemple, le point d'abscisse X = 16 appartient au bloc 2.
En effet :

Bloc = INT (X/7) = 2



1537

et son adresse équivaut au début de ligne (8192) plus 2 soit :

Mémoire = 8192+2 = 8194

Dans l'emplacement ainsi repéré, le 1er bit (numéro 0) représente le point écran X = 14 ; le point étudié étant X = 16, il sera représenté par le 3e bit, c'est-à-dire par le n° 2 (bit 0 = point 14, bit 1 = point 15, bit 2 = point 16). En général, pour le calcul du bit, on a :

Numéro du bit = X — Bloc * 7

L'organigramme ci-dessous et le suivant décrivent deux sous-programmes réalisant ces traitements. Au premier (1800), il faut fournir l'adresse de référence de la ligne (déterminée par la valeur de Y) et la valeur de l'abscisse X. On obtient, en sortie, l'emplacement mémoire MX qui contient l'état du point écran (coordonnées X, Y) et sa position (le n° du bit), à l'intérieur de l'emplacement mémoire.

Le second sous-programme (1900) active un point écran en écrivant la valeur 1 dans le bit correspondant.

L'activation du bit s'effectue avec l'opérateur OU (OR) pour conserver la position précédente, sinon cette activation entraînerait l'effacement des bits précédemment activés. Ce sous-programme lit également l'état d'un point : dans ce cas, il ne contiendra que l'instruction PEEK.

La technique de partition de mémoire est très usitée dans les programmes d'animation de dessins. Avant de déplacer un dessin, on contrôle l'état de l'écran au point d'arrivée pour savoir s'il est déjà activé ou non.

Grâce à cette donnée, le programme d'application peut sélectionner l'opération adéquate, par exemple en refusant le déplacement.

ORGANIGRAMME DE L'ADRESSAGE ABSCISSE

Entrées : MY = Adresse de référence calculée par les sous-programmes 1100 et 1500
X = Valeurs abscisse

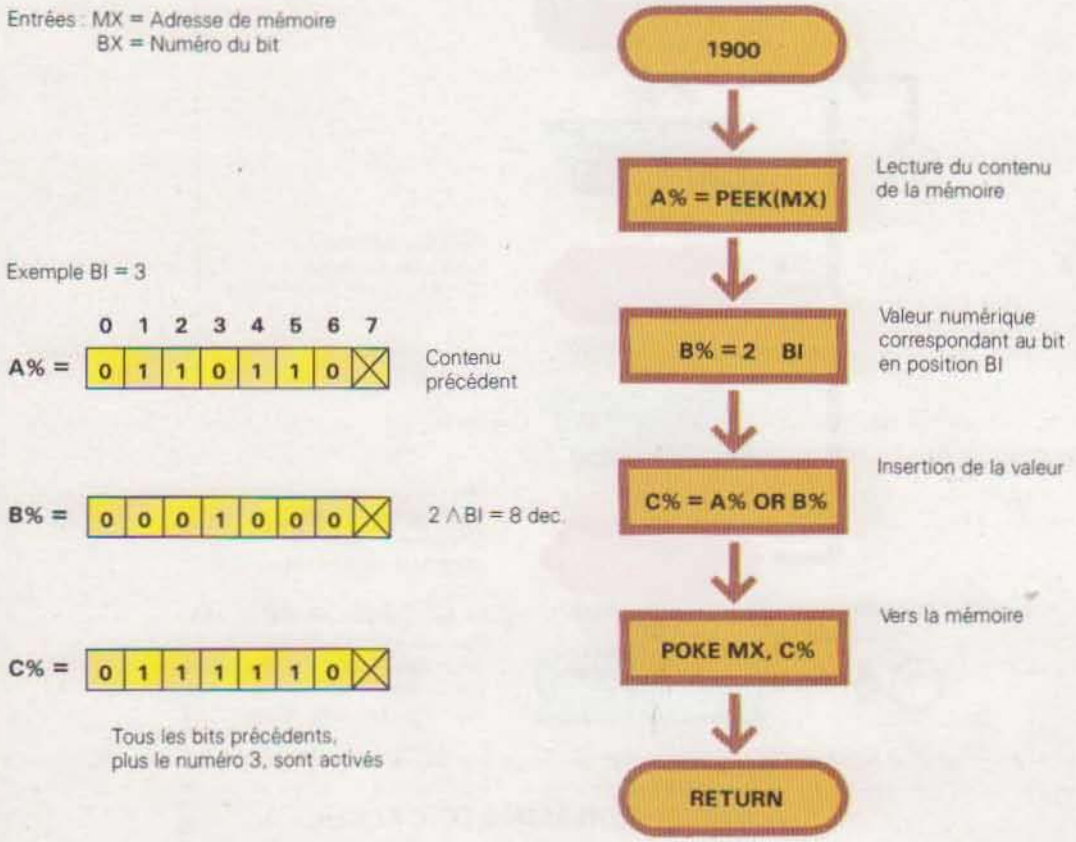
Sortie : MX = Adresse de la mémoire axe X
BI = Numéro du bit à activer

Sélection du groupe appartenant à la coordonnée X

Pas = 1
à partir de la référence MY

Calcul du bit à activer dans la mémoire d'adresse MX

ORGANIGRAMME D'ACTIVATION D'UN BIT



Dans les graphiques des pages 1540 et 1541, on trouvera l'organigramme d'un programme de démonstration qui emploie cette technique. Dans le listing de la page 1542, nous avons omis les sous-programmes dans la mesure où ils ont déjà été présentés. La structure en points écran (caractéristique de la technique « raster ») génère, dans certains cas, une représentation très approximative.

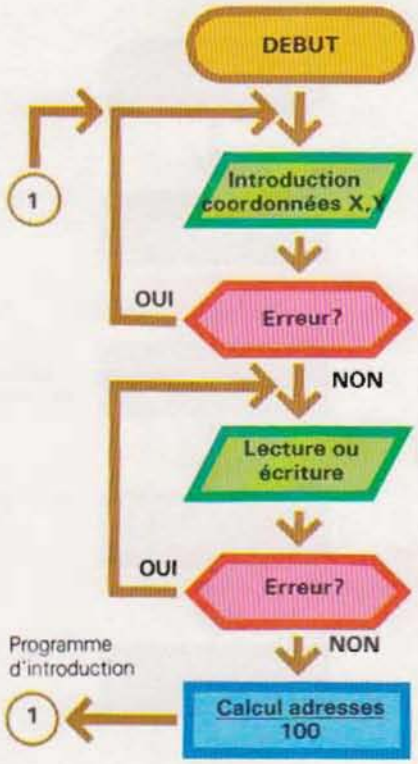
Le résultat est un tracé de segment incliné par rapport aux axes (p. 1543). A cause de cette inclinaison, certaines parties du segment (1 et 3) sont représentées par des traits parallèles à l'un des axes : ils sont respectivement horizontaux et verticaux; en effet, à cause de la pente assez forte, ils concernent plusieurs bits de la même mémoire. Avec des inclinaisons proches de 45°, on a, pour chaque point écran, une ligne et une colonne différentes avec un bien meilleur résultat. Ce défaut est strictement proportionnel à la résolution de l'écran et il ne peut être éliminé.

L'adressage direct de la mémoire

La préparation (ou la reconnaissance) d'une figure, en utilisant l'adressage de la mémoire écran, offre davantage de possibilités que l'usage d'instructions de niveau élevé. Ces dernières sont, en effet, structurées de manière à satisfaire les besoins moyens d'un utilisateur désireux de générer des dessins. Cependant, elles ne prévoient pas de fonctions particulières, comme par exemple la possibilité d'analyser l'écran pour relever la présence d'éventuels dessins.

Pour ce type de fonctions, il est indispensable d'utiliser l'adressage de la mémoire écran pour en lire le contenu ou pour le modifier.

PROGRAMME DE DEMONSTRATION D'ADRESSAGE D'UNE MEMOIRE GRAPHIQUE

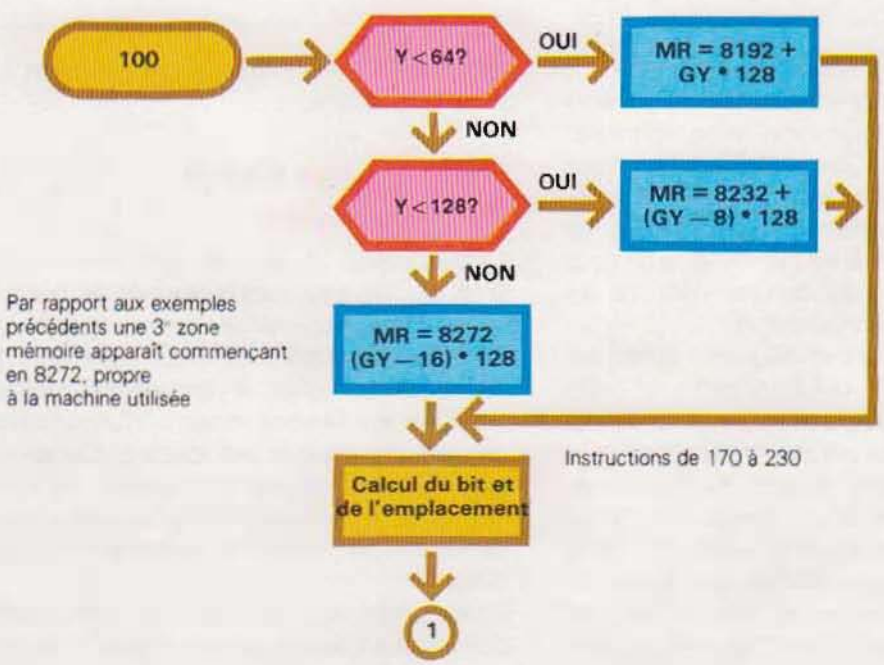


Coordonnées dans les limites de l'écran ($0 < X < 279$ et $0 < Y < 159$)

Réponse L (lecture) ou E (écriture). Les autres caractères sont écartés

Le sous-programme 100 contient les deux calculs (axe Y et X) précédemment rapportés dans deux sous-programmes séparés (1100 et 1500) (voir organigramme détaillé)

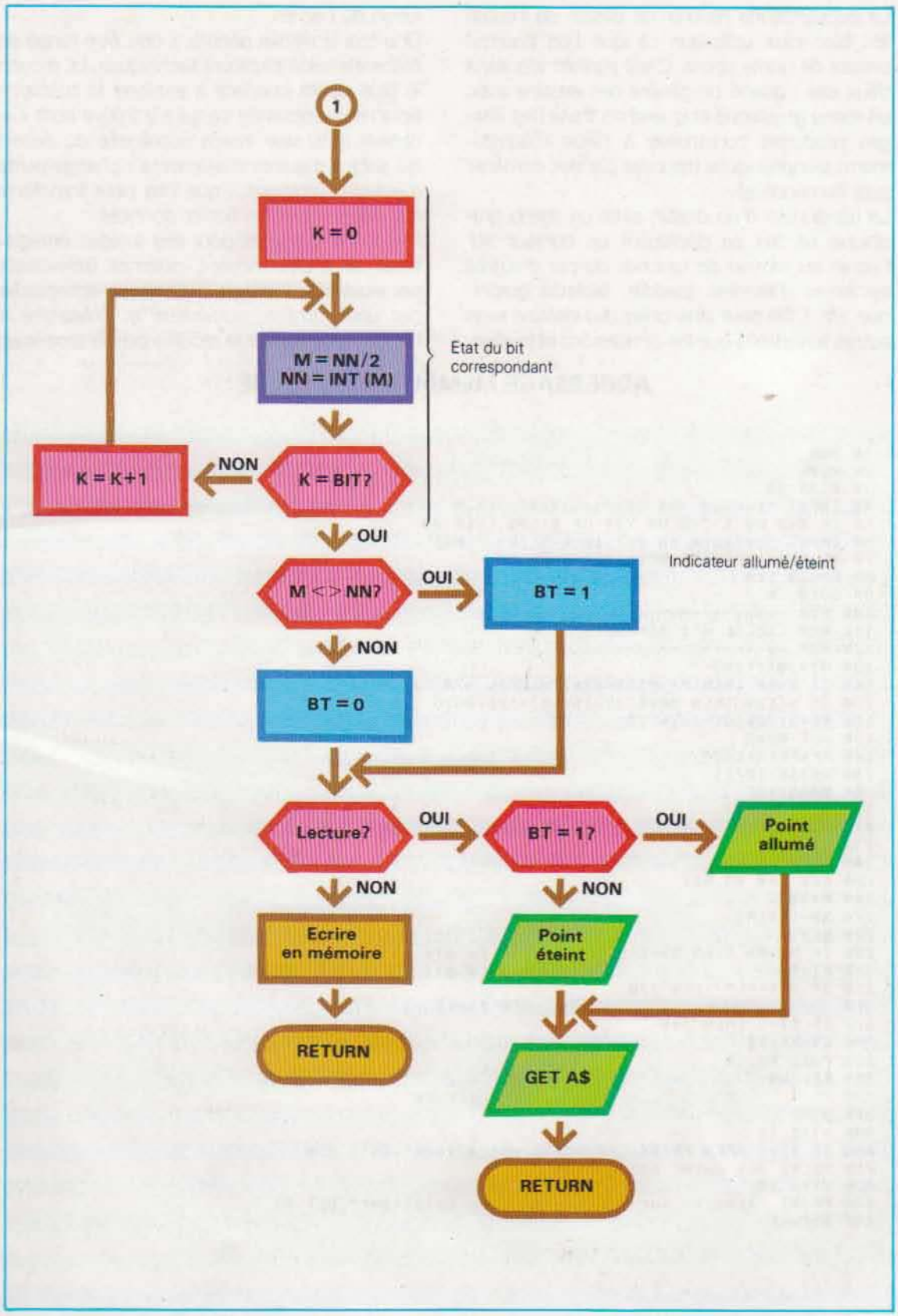
SOUS-PROGRAMME DE CALCUL



Ces deux calculs sont ceux du 1100

Par rapport aux exemples précédents une 3e zone mémoire apparaît commençant en 8272, propre à la machine utilisée

Instructions de 170 à 230


1
K = 0
M = NN / 2
NN = INT (M)
Etat du bit correspondant
K = K+1
NON
K = BIT?
OUI
M <> NN?
OUI
BT = 1
Indicateur allumé/éteint
NON
BT = 0
Lecture?
OUI
BT = 1?
OUI
Point allumé
NON
NON
Ecrire en mémoire
Point éteint
RETURN
GET A$
RETURN

La possibilité de repérer un dessin de l'écran est bien plus utile que ce que l'on pourrait penser de prime abord. C'est surtout vrai dans deux cas : quand on génère des dessins avec un menu graphique et quand on traite des images produites construites à l'aide d'équipements périphériques (reprises par des caméras puis numérisées).

La génération d'un dessin avec un menu graphique se fait en déplaçant un curseur sur l'écran au moyen de touches ou par d'autres systèmes d'entrées (paddle, tablette graphique, etc.). On peut ainsi créer des dessins sans autres limitations que les dimensions et la résolution de l'écran.

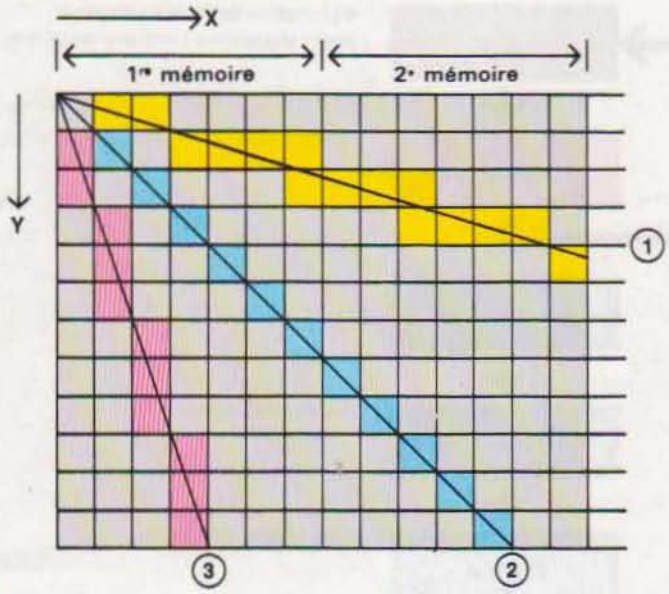
Une fois le dessin généré, il doit être rangé en mémoire selon plusieurs techniques. Le moyen le plus direct consiste à explorer la mémoire écran et à conserver ce qui s'y trouve écrit. On obtient ainsi une image numérisée du dessin qui subira d'autres traitements : changements d'échelle, rotations... que l'on peut transférer sur disque dans un fichier données.

Mêmes traitements pour des images enregistrées avec des moyens externes (télévisuels par exemple). L'image originale est enregistrée par une caméra, numérisée et présentée à l'écran. L'opérateur la modifie puis la stocke en

ADRESSAGE MEMOIRE GRAPHIQUE

```
10 HGR
20 HOME
30 VTAB 23
40 INPUT "Entrez les coordonnées: ";X,Y
50 IF X<0 OR X>279 OR Y<0 OR Y>159 THEN 20
60 INPUT "Lecture ou écriture (L/E) ";MD$
70 IF MD$<>"L" AND MD$<>"E" THEN 20
80 GOSUB 100
90 GOTO 20
100 REM --------------------
110 REM CALCUL DES ADRESSES
120 REM --------------------
130 GY=INT(Y/8)
140 IF Y<64 THEN MR=8192+GY*128:GOTO 170
150 IF Y<128 THEN MR=8232+(GY-8)*128:GOTO 170
160 MR=8272+(GY-16)*128
170 A=Y-GY*8
180 MY=MR+A*1024
190 GX=INT(X/7)
200 MX=MY+GX
210 BIT=X-7*GX
220 A%=PEEK(MX)
230 B%=2BIT
240 NN=A%
250 FOR K=0 TO BIT
260 M=NN/2
270 NN=INT(M)
280 NEXT
290 IF M<>NN THEN BT=1:GOTO 310:REM Le bit est allumé
300 BT=0:                          REM Le bit est éteint
310 IF MD$="L" THEN 370
320                                REM Ecriture
330 IF BT=1 THEN 360
340 C%=A%+B%
350 POKE MX,C%
360 RETURN
370                                REM Lecture
380 HOME
390 VTAB 23
400 IF BT=1 THEN PRINT "Le point est allumé":GOTO 420
410 PRINT "Le point est éteint"
420 VTAB 24
430 PRINT "Appuyez sur une touche pour continuer":GET A$
440 RETURN
```

IMPACT DE LA RESOLUTION DE L'ECRAN VIDEO SUR LA PRESENTATION DES SEGMENTS NON PARALLELES AUX AXES



L'adressage minimum est un bit, auquel correspond un petit point lumineux. Dans le schéma, on a tracé 3 lignes inclinées. Pour une seule d'entre elles (45 degrés) la figure de l'écran se rapproche de la tendance. Pour les autres, on obtient une série de segments parallèles. Dans la figure, chaque point écran est représenté par un carré

mémoire toujours selon la méthode de l'adressage direct. Ces possibilités, associées à l'emploi d'un écran couleur, ont ouvert de nouvelles perspectives au dessin assisté par ordinateur, notamment dans la publicité où la création utilise les effets spéciaux. Dans le domaine de l'art, elles ont donné naissance à une véritable école de peinture grâce au pinceau électronique. L'apport de l'ordinateur consiste en sa grande rapidité et dans la facilité des modifications. La variation d'une couleur ou de la forme d'un détail n'exige d'autres opérations que l'envoi d'une commande à l'ordinateur, là où les moyens traditionnels demanderaient un temps considérable.

Notons aussi la possibilité de mettre en mémoire certaines formes standard qui, lors de la préparation du dessin, seront rappelées et utilisées pour un travail de composition. On peut ainsi créer un menu graphique pour des applications artistiques.

Stockages des images graphiques

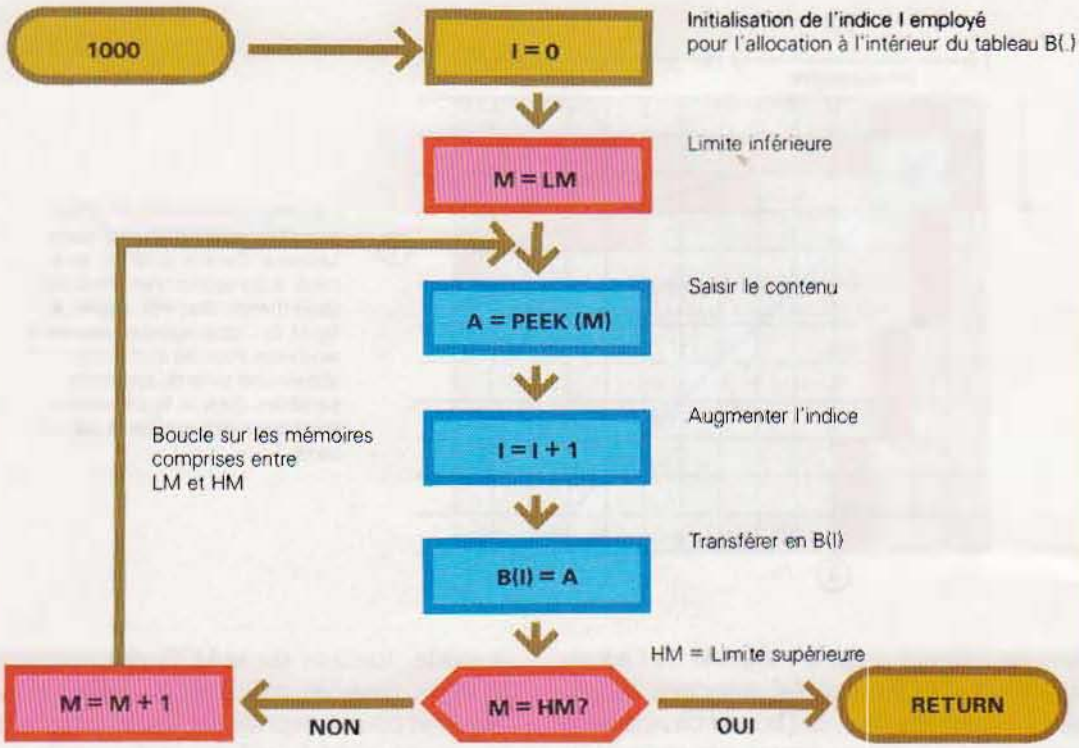
Un traitement graphique commence par la saisie de la figure présentée à l'écran. Pour ce faire, certaines machines disposent d'instructions spéciales. Ainsi, dans l'Olivetti M20, l'instruction POINT (X, Y) restitue la couleur du point de coordonnées X, Y, ainsi que son état.

Il existe, toujours sur le M20, des instructions pour la mise en mémoire de toute la page écran et pour sa représentation. Pour les autres machines, c'est à l'utilisateur d'écrire ses propres sous-programmes.

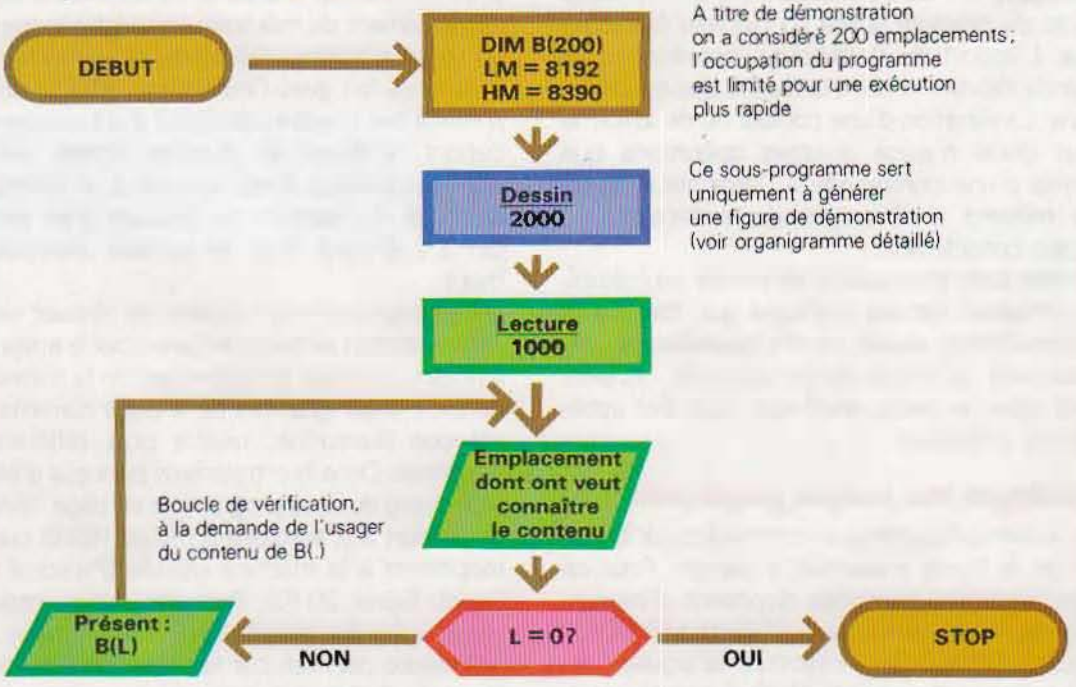
La méthode la plus simple consiste à saisir, avec l'instruction PEEK, le contenu de chaque emplacement de mémoire écran et à le tranférer dans un tableau défini par l'utilisateur. Cette lecture se fait avec l'instruction PEEK quand il n'existe pas d'autre possibilité d'adressage en-dehors du direct. En d'autres termes, aucun nom symbolique n'est associé à la mémoire écran et l'utilisateur, ne pouvant s'en servir, devra l'adresser avec le numéro d'emplacement.

Le sous-programme capable de réaliser cette tâche est tout simplement une boucle entre les limites maximales et minimales de la mémoire écran. L'organigramme de la page suivante en est une illustration, valable pour différentes machines. Dans le programme principal d'essai (voir listing du sous programme en page 1546), on recourt aux adresses 8192 et 16383 qui se rapportent à la machine utilisée (Personal Kid Siprel, Siprel 2010). Pour les autres ordinateurs, il faudra modifier les valeurs selon les indications données par le manuel d'utilisation. Dans le listing, pourraient apparaître deux

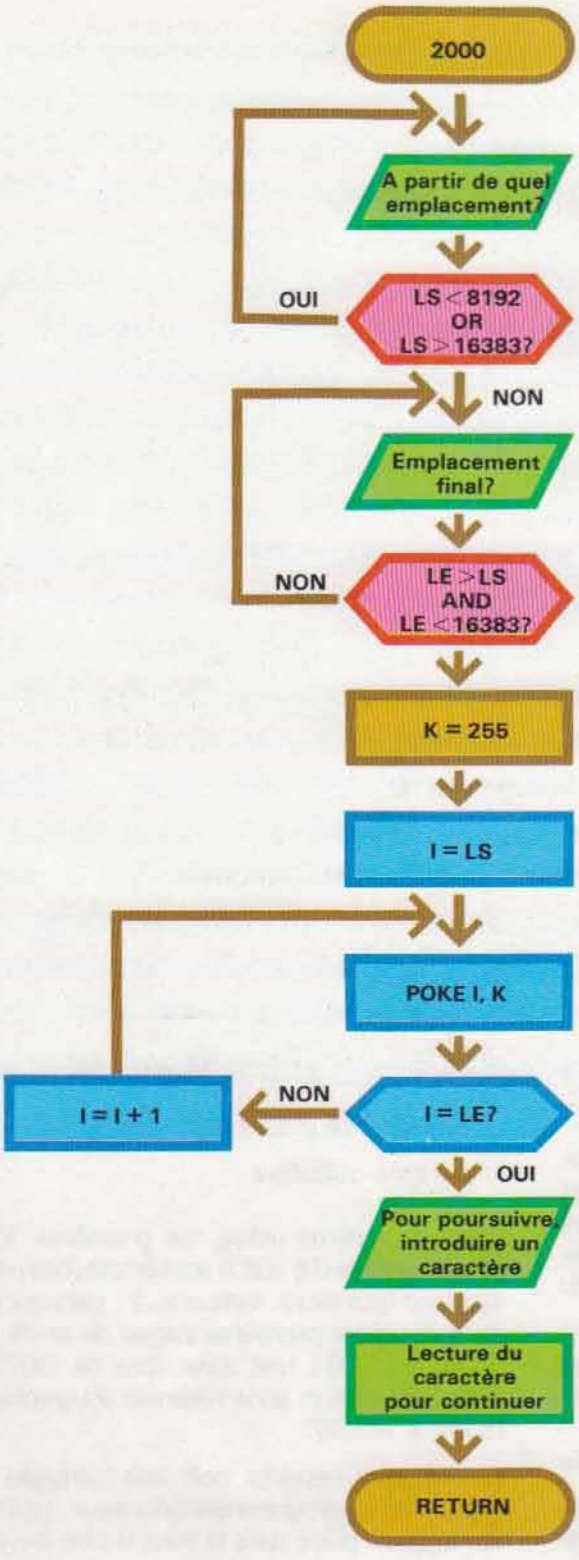
LECTURE DE LA MEMOIRE VIDEO



Programme principal d'essai



PREPARATION D'UN SEGMENT D'ESSAI



Le sous-programme n'a qu'un but pédagogique et se limite à la présentation de segments horizontaux. Les deux emplacements de mémoire doivent appartenir à la même ligne.
En introduisant des adresses n'appartenant pas à la même ligne, on peut activer tous les points écran compris entre un minimum et un maximum

Active les bits de 0 à 7

Boucle qui écrit la valeur 255 dans les mémoires graphiques choisies

LECTURE ET VISUALISATION DU CONTENU DE LA MEMOIRE

VERSION Siprel 2010, Apple et compatibles

```
50 HGR
100 DIM B(200):LM=8192:HM=8390
110 GOSUB 2000
120 GOSUB 1000
130 HOME:VTAB 22:INPUT "Entrez localisation ";L
140 IF L=0 THEN TEXT:END
150 HOME:VTAB 22:PRINT "Contenu = "B(L)
155 PRINT "Appuyer une touche":GET Q$
160 GOTO 130
1000 REM ===========
1001 REM * Lecture *
1002 REM ===========
1003 :
1010 I=0
1015 FOR M=LM TO HM
1020 A=PEEK(M):I=I+1
1030 B(I)=A
1040 NEXT M
1050 RETURN
2000 REM ==========
2001 REM * Dessin *
2002 REM ==========
2003 REM
2010 HOME:VTAB 22:INPUT "Localisation initiale ";LS
2015 IF LS<8192 OR LS>16383 THEN 2010
2020 INPUT "Localisation finale ";LE
2025 IF LE>LS AND LE<16383 THEN 2040
2030 GOTO 2020
2040 K=255
2045 FOR I=LS TO LE
2050 POKE I,K
2060 NEXT I
2070 HOME:VTAB 22:PRINT "Frappez une touche":GET Q$
2080 RETURN
```

instructions reliées étroitement à la machine : HIMEM et LOMEM. Ce type d'instructions indique la zone occupée par le programme. Elle est donc limitée par HIMEM (adresse de mémoire la plus élevée) et par LOMEM (adresse de mémoire la plus basse) selon la configuration du matériel et du logiciel du système.

La RAM du système est divisée en 3 fonctions ; en voici un résumé :

— système d'exploitation, par exemple le DOS

— fonctions particulières (par exemple mémoire écran de texte)

— mémoire graphique

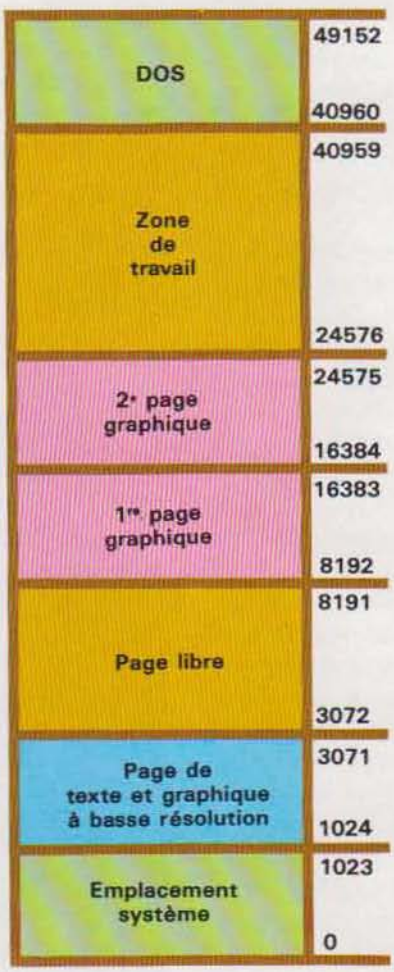
— zone utilisateur

Dans le système utilisé, les premières 1024 allocations (de 0 à 1023) sont employées par le système (pointeurs, tampons...) ; viennent ensuite les deux premières pages de texte (de 1024 à 3071) ; une zone libre de (3072 à 8191) et enfin la zone réservée au graphique (8192 à 16383).

La mémoire restante doit être partagée en 3 DOS et en programmes utilisateur. Le DOS est toujours placé dans la zone la plus élevée ; donc, reste pour le programme utilisateur, une

zone comprise entre la fin de la page graphique et le début du DOS (schéma ci-dessous).
Ce système offre, en outre, une 2e page graphique gérée et présentée indépendamment de la première (emplacements 16384 à 24575). Le programme utilisateur part donc de l'allocation 24576 et s'achève avant le début du DOS (emplacement 40960).
Les deux paramètres HIMEM et LOMEM établissent ces bornes pour éviter d'empiéter sur la zone réservée au graphique.
Dans ce cas particulier, les deux paramètres ne sont pas nécessaires, étant donné le faible encombrement du programme : nous ne les avons mentionnés que pour faire ressortir ce problème d'encombrement de mémoire, particulièrement important dans les applications graphiques. Notre programme a cependant besoin de certaines fonctions supplémentaires pour la mise en mémoire et le rappel de pages graphiques.
La méthode la plus avantageuse consiste à utiliser un tampon qui ne contient qu'une seule ligne à la fois (40 emplacements de mémoire) ; à chaque nouvelle ligne, son contenu est transféré sur disque ; le tampon est donc à nouveau libre pour recevoir la ligne suivante.
Cette méthode se trouve illustrée par l'organigramme de la page 1551.
Puisqu'il faut disposer d'un dessin sur l'écran pour procéder au test du programme (dont le sous-programme saisit les points), nous

PARTITION DE LA MEMOIRE RAM DANS LE SIPREL 2010



Radioscopie de l'ordinateur personnel

Bien souvent, on trouve chez l'utilisateur d'un ordinateur personnel une bonne connaissance du langage de programmation. En revanche, il ignore la structure de sa machine. Le châssis de l'ordinateur est souvent considéré comme une barrière impénétrable.

L'attitude d'embarras à l'égard de la machine et de ses composants, risque cependant d'avoir des conséquences fâcheuses, car l'apparition d'une panne ou d'un mauvais fonctionnement du système prend souvent l'utilisateur au dépourvu.

Toute personne qui s'est trouvée dans la triste nécessité de devoir faire réparer une défaillance matérielle de son ordinateur personnel aura certainement expérimenté le manque de diligence avec lequel les centres d'assistance émettent un diagnostic, ainsi que la longueur des délais de réparation.

C'est le moment de rappeler avec insistance que la structure interne d'un ordinateur personnel est très simple, bien plus simple que celle d'un circuit de téléviseur ou de radio. En effet, ceux-ci traitent des signaux analogiques et non numériques.

L'utilisateur courageux, qui décide un beau jour de retirer le capot qui recouvre les circuits, reste toujours surpris par le peu d'éléments et par la modularité du système. Il peut constater que chaque fonction est réalisée par des composants (cartes) bien séparés, facilement identifiables, dont les connections sont, à première vue, élémentaires.

Sur chaque carte, les circuits intégrés ne sont pas soudés mais insérés par pression sur des plots prévus à cet effet.

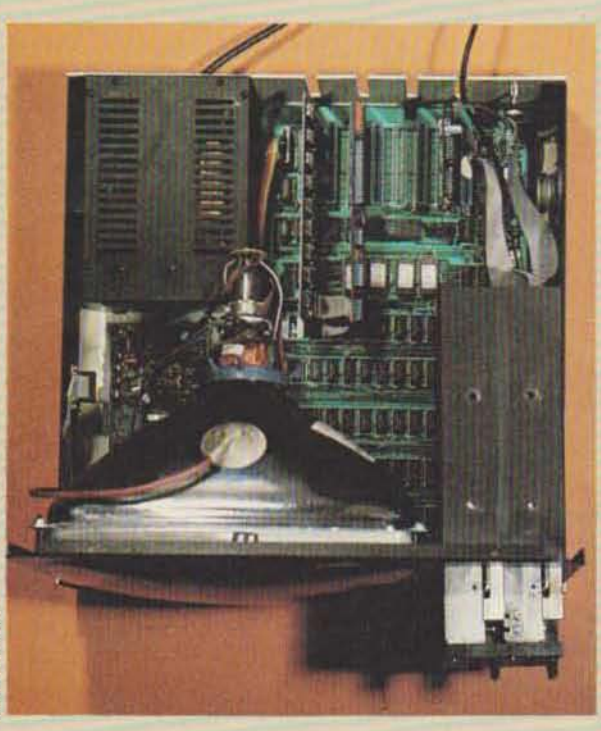
Dans ces conditions, une réparation ne devrait pas demander plus de 15 minutes pour le diagnostic et moins d'une minute pour le changement du circuit intégré en cause, lequel coûte rarement plus de quelques dizaines de francs. Donner un coup d'oeil à la structure interne de votre ordinateur vaut donc la peine, ne serait-ce que pour se rendre compte que ce que nous venons de dire est vrai.

Le système Siprel 2040 (Personal Kid) se prête fort bien à une étude générale de l'architecture d'un ordinateur personnel. Il s'agit d'une machine où l'écran, les lecteurs et les cartes se partagent une même entité physique à laquelle est relié un clavier séparé.

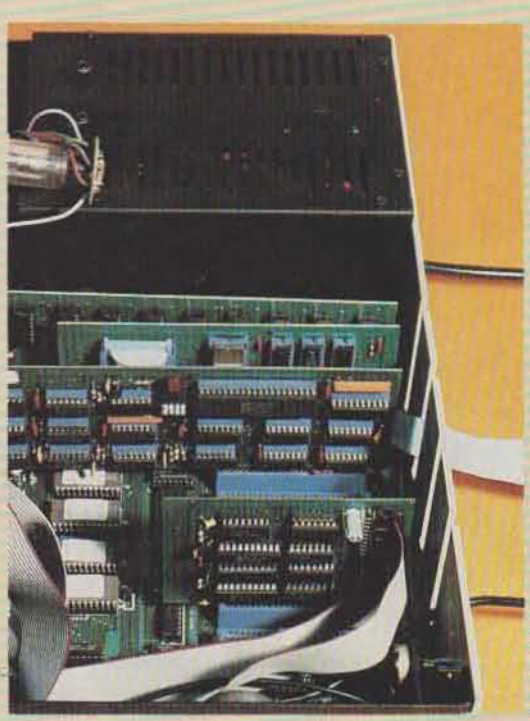
Même s'il existe des versions de ce système où les unités sont séparées, nous allons examiner cette configuration justement parce que, au premier abord, elle apparaît inabordable.

Dans la première photo ci-dessous, nous avons enlevé le couvercle du châssis pour extraire les deux lecteurs : il nous a suffi de retirer 8 vis. La seconde (haut de la page 1549) montre le système vu du dessus. Outre les lecteurs de type « slim », on y voit clairement le tube à rayons cathodiques et, au-dessus, la boîte d'alimentation, avec les fentes pour la ventilation du transformateur. Sur le fond on entrevoit la fiche mère, avec les slots d'insertion des diverses cartes fonctionnelles (visibles dans la partie haute de la photo). La troisième photo montre ces cartes montées et, au premier plan, la fiche d'interface avec les lecteurs et les deux câbles plats qui en sortent (un pour chaque unité de disque). Au second plan, on distingue une partie de la carte qui contient l'unité centrale (Zilog Z80) ; puis la carte interface avec l'écran et une carte extension de la mémoire RAM. Les mêmes cartes sont visibles dans la quatrième

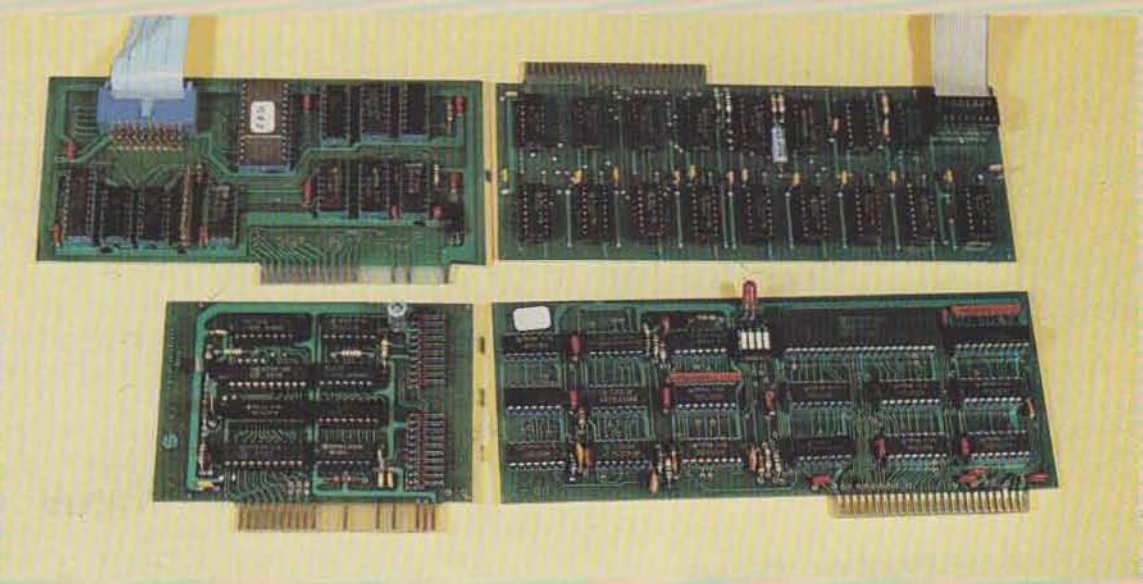
Il Dagherotipo



Il Dagherrotipo



Il Dagherrotipo



Il Dagherrotipo

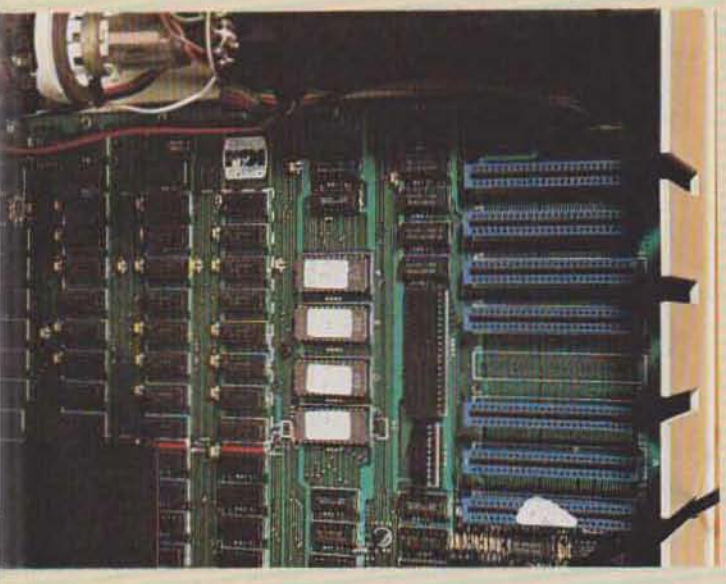
photo : en haut à gauche, la carte de gestion d'écran 80 colonnes et, à côté, l'extension mémoire ; en bas à gauche, la carte interface avec les lecteurs et, à côté, la carte avec le processeur Z80.

La cinquième photo (page 1550) met en évidence un détail interne après séparation des cartes fonctionnelles et de la fiche mère. Les 7 slots sont maintenant bien visibles et on peut entrevoir, dans le bas, le sabot pour la connexion des dispositifs externes (joystick, tablette graphique, paddle, etc.)

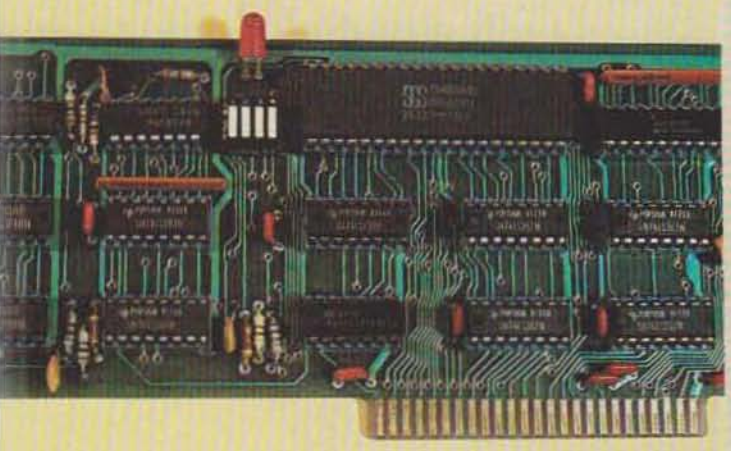
Une tablette graphique est reliée au sabot dans notre exemple : le câble de connexion sort du côté droit de la photo.

Le circuit intégré de grande taille, qui se trouve dans le sens contraire des autres, est l'UC (Rockwell 6502), alors que les circuits intégrés plus petits, occupant la partie gauche de l'image, sont des mémoires RAM.

Les quatre circuits intégrés recouverts par des étiquettes sont les EPROM (mémoires de



Il Daghenotipo



Il Daghenotipo

Il Daghenotipo

lecture seulement, qui sont programmables et effaçables) où se trouve le système d'exploitation. Les étiquettes qui recouvrent les fenêtres permettent d'effacer le contenu des mémoires au moyen d'une exposition aux rayons ultraviolets et ont pour but d'empêcher l'effacement accidentel à la suite d'une exposition prolongée à la lumière solaire.

Le système Siprel utilise, comme base, l'unité centrale Rockwell 6502 (la même qu'Apple et que Commodore 64) qui travaille en DOS mais peut être complétée par un processeur Zilog Z80, travaillant normalement sous CP/M (photo centrale).

Le processeur 6502 est contenu dans la carte mère sur laquelle ont peut insérer une deuxième qui contient la Z80. En insérant la carte optionnelle, toutes les unités fonctionnelles (mémoires, dispositifs d'E/S, etc.) peuvent être gérées par la nouvelle unité centrale et on obtient ainsi un système qui travaille sous CP/M. Le passage d'un mode de fonctionnement à l'autre est automatique. A la mise en route le système est prêt à fonctionner sous DOS, avec le processeur 6502. En insérant la disquette CP/M dans le lecteur et en tapant, à partir du clavier, la commande DOS de chargement (IN = 6) le DOS est « remplacé » par le CP/M et le système passe sous le contrôle de l'unité Z80.

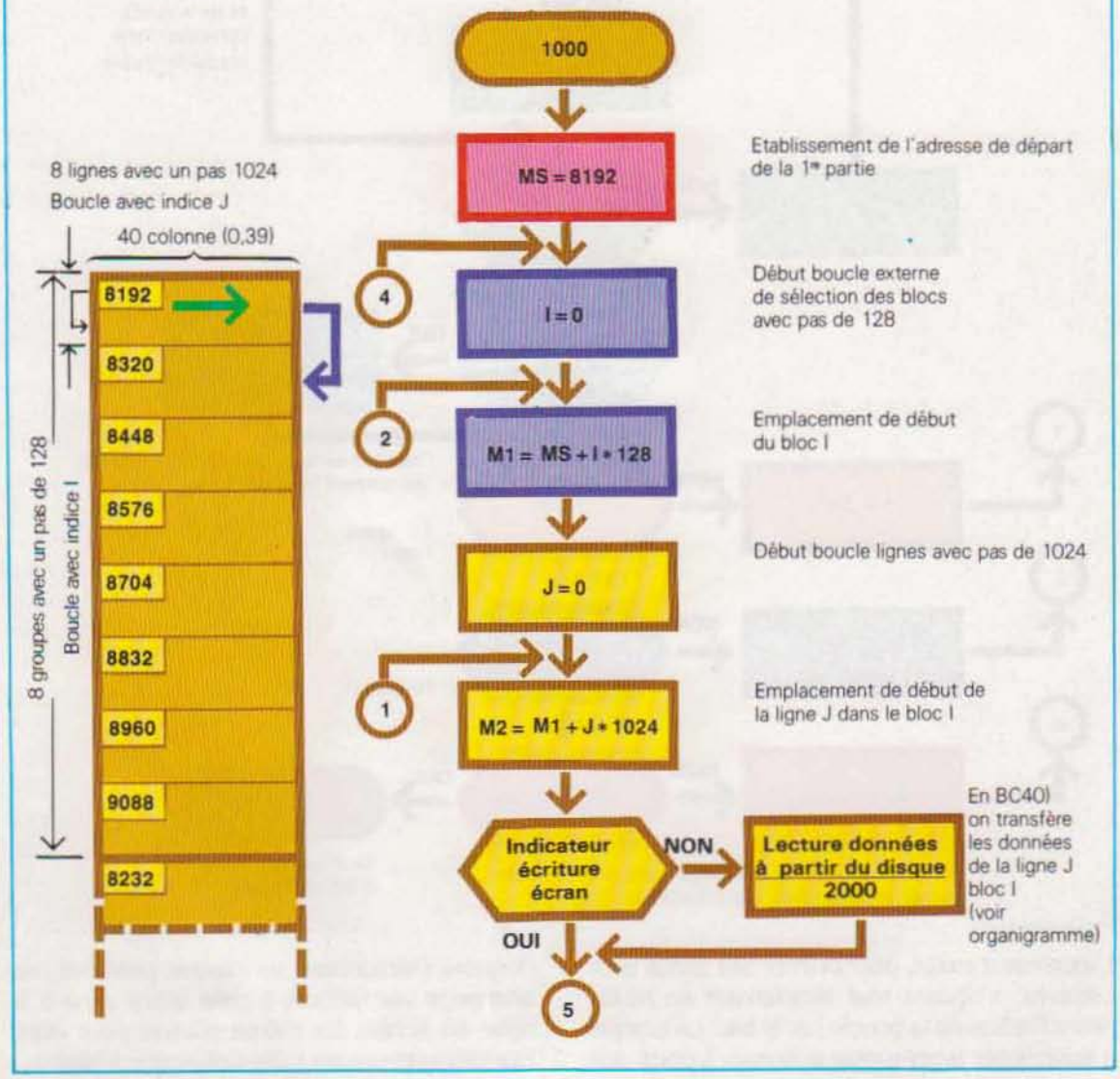
Cette dualité est très pratique dans la mesure où elle permet de faire exécuter par la machine tous les programmes écrits sous l'un ou l'autre des systèmes d'exploitation qui sont parmi les plus utilisés.

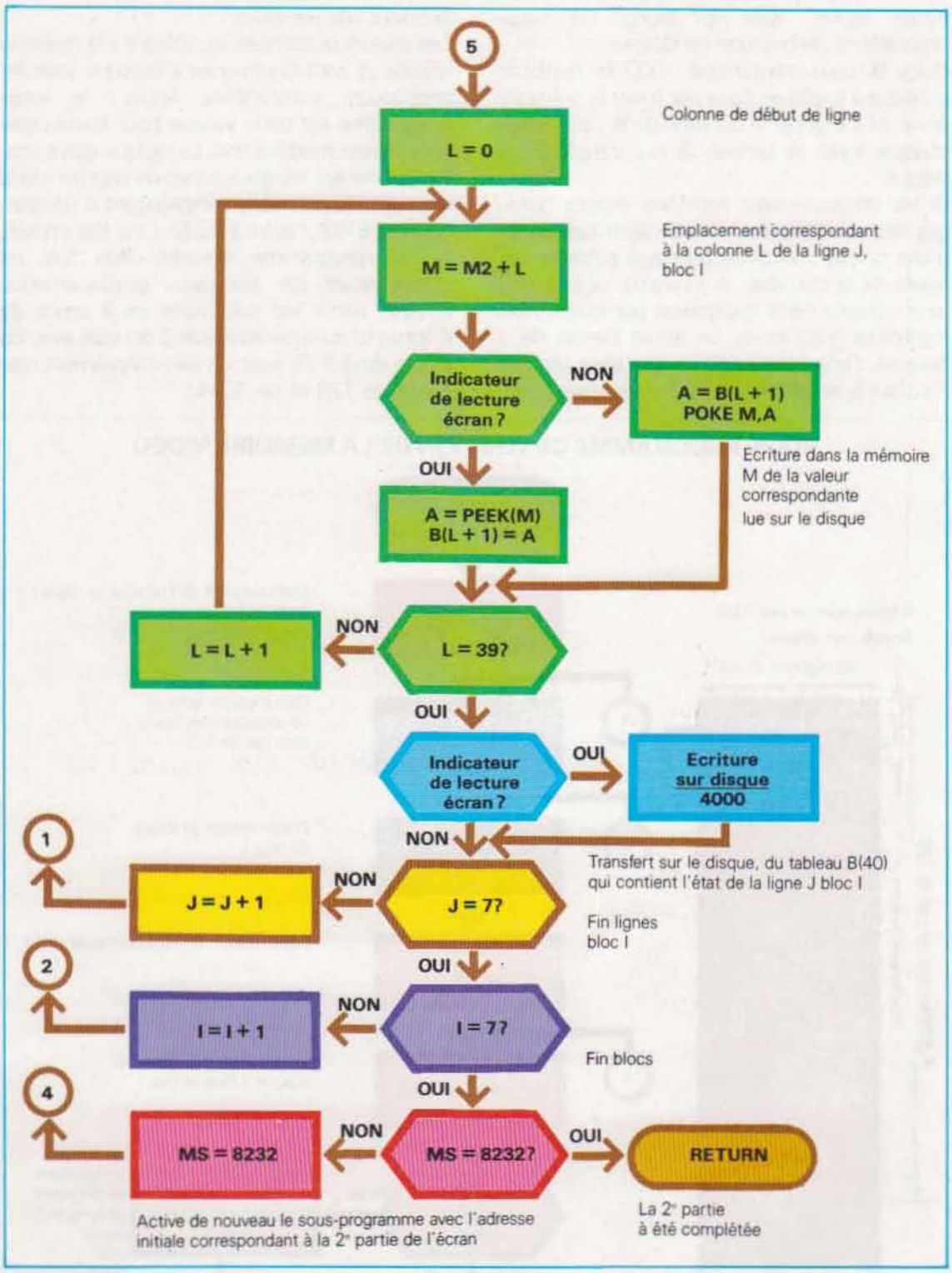
La dernière photo montre le clavier du système. Dans la partie centrale on distingue le circuit intégré destiné à la gestion du dispositif. Ajouter que le circuit intégré plus petit, à côté de l'attache du câble, sert à isoler le circuit principal de la ligne.

Nous pensons ainsi avoir illustré la transparence d'un système de traitement bien conçu et surtout d'avoir donné envie aux utilisateurs d'approfondir, de leur côté, l'étude de l'anatomie du matériel. De cette expérience, on pourra prendre conscience du fait que chacun de nous est en mesure de résoudre seul bien des problèmes qui apparaissent normalement dans la vie d'un ordinateur ou, au moins, d'orienter de manière claire un éventuel travail de réparation.

avons inclus, dans le listing, un sous-programme de création de dessin.
Dans le sous-programme 1000 la méthode consiste à explorer, ligne par ligne, la mémoire écran et à stocker le contenu de B ; à la fin de chaque ligne, le tableau B est transféré sur disque.
Si les emplacements mémoire étaient contigus, les sous-programmes seraient constitués d'une simple boucle de balayage entre la première et la dernière. A l'inverse, si l'on doit tenir compte de la disposition particulière des mémoires graphiques, on aurait besoin de 3 boucles, l'une pour explorer une ligne (de 0 à 39 c'est-à-dire 40 mémoires) et les autres pour le calcul des adresses.
Les valeurs rapportées se réfèrent à la machine utilisée et sont communes à presque tous les ordinateurs compatibles Apple ; le sous-programme est donc valable pour toute cette famille sans modification. La logique suivie traite séparément les deux zones de partition de la mémoire, la première commençant à l'emplacement 8192, l'autre à 8232. Une fois appelé, le sous-programme travaille deux fois en commençant par ces deux emplacements. Chaque partie est subdivisée en 8 zones de 8 lignes (d'où la nécessité de 2 boucles avec un indice de 0 à 7), avec un pas d'incrément respectif de 128 et de 1024.

SOUS-PROGRAMME DE GESTION DE LA MEMOIRE VIDEO

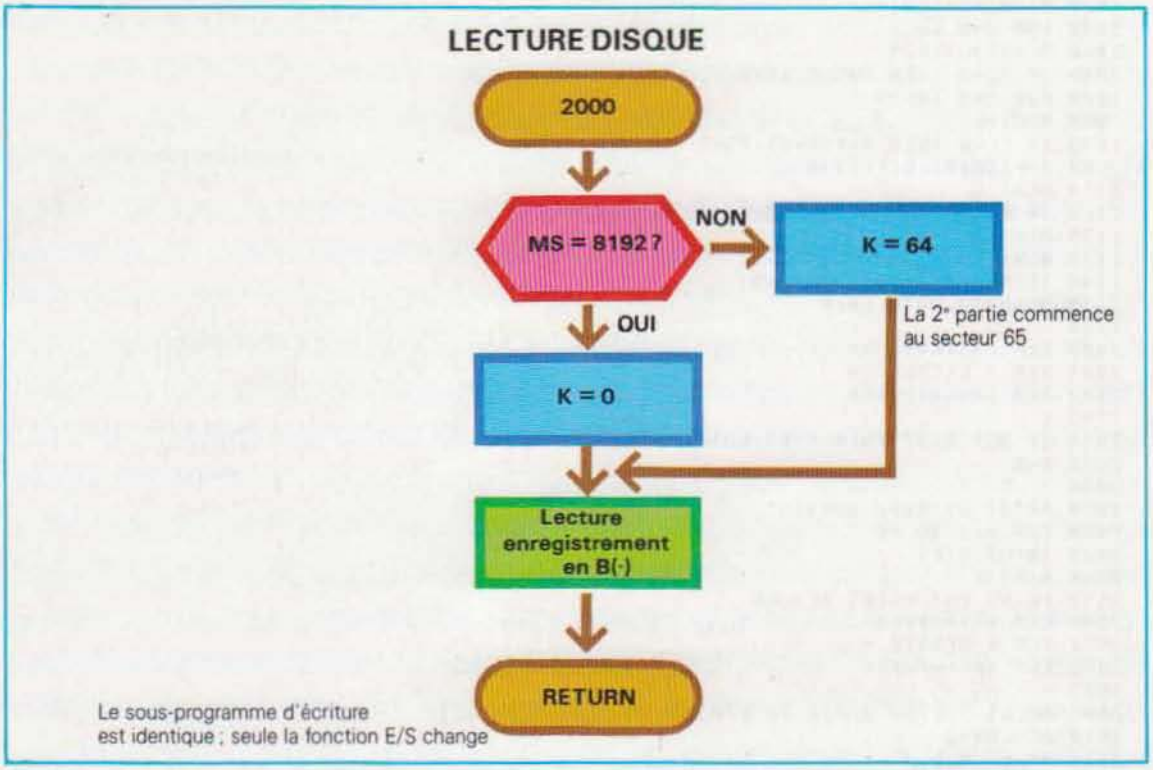




L'incrément exact, pour pointer des zones successives, s'obtient tout simplement en multipliant l'indice de la boucle par le bas. La logique à suivre pour recomposer le dessin, à partir des données mémorisées sur disque, présente une altenance par rapport à celle ayant servi à la mise en fichier. La même routine peut donc alors être utilisée pour deux objectifs différents :

l'un ou l'autre mode de fonctionnement est indiqué par la valeur d'un indicateur activé lors de l'appel.

L'emploi de ce sous-programme sur d'autres machines exige parfois d'importantes modifications ; par exemple, si on dispose d'une ins-



MEMORISATION DES IMAGES GRAPHIQUES SUR DISQUE

```
1 REM ====================
2 REM *     PROGRAMME     *
3 REM *     DE DESSIN     *
4 REM * ET MEMORISATION *
5 REM *     SUR DISQUE    *
6 REM ====================
7 :
30 :
40 HIMEM:8191
45 :
50 DIM B(40):Y=90:X=140:S=5
55 D$=CHR$(4)
60 PRINT D$"Open dessin"
65 HCOLOR=3
100 HOME:VTAB 22:INPUT "Lecture/Ecriture (L/E) ";A$
110 IF A$="L" THEN FL=0:HGR:GOTO 150
120 IF A$="E" THEN FL=1:HGR:GOTO 140
130 IF A$="" THEN 200
135 GOTO 100
140 GOSUB 3000
150 GOSUB 1000:GOTO 100
200 PRINT D$"Close dessin":TEXT:END
999 :
1000 REM ==================
```

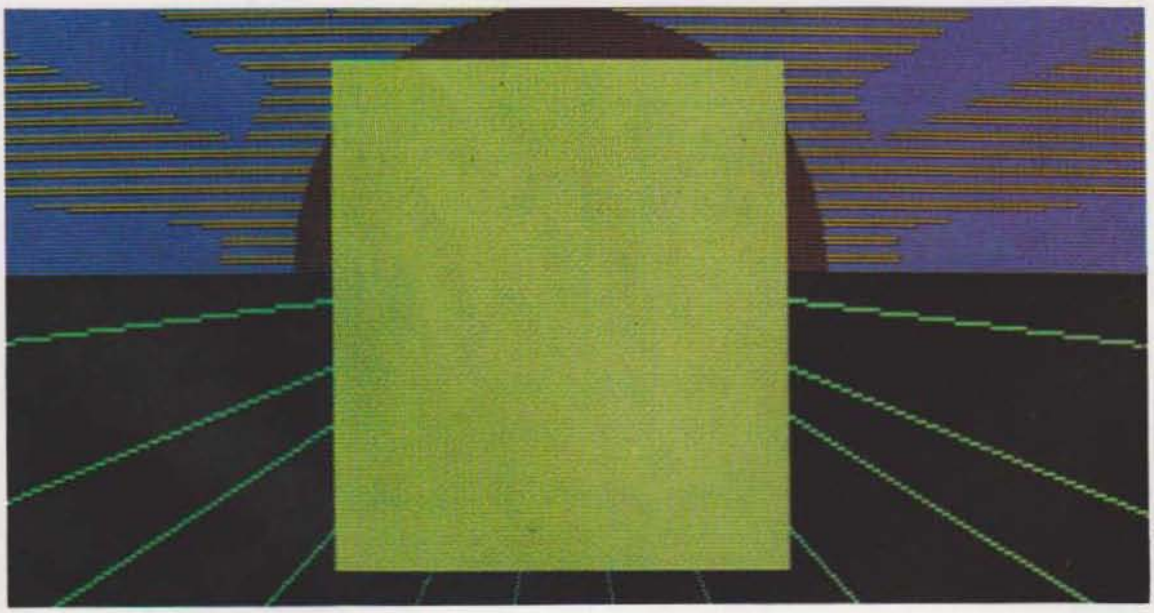
```
1001 REM * MEMOIRE ECRAN *
1002 REM ==================
1003 :
1010 MS=8192
1020 FOR I=0 TO 7
1030 M1=MS+I*128
1040 FOR J=0 TO 7
1050 M2=M1+J*1024
1060 IF FL=0 THEN GOSUB 2000
1070 FOR L=0 TO 39
1080 M=M2+L
1090 IF FL=0 THEN A=B(L+1):POKE M,A:GOTO 1110
1100 A=PEEK(M):B(L+1)=A
1110 NEXT L
1120 IF FL=1 THEN GOSUB 4000
1130 NEXT J
1135 NEXT I
1140 IF MS=8232 THEN RETURN
1150 MS=8232:GOTO 1020
1160 :
2000 REM ============
2001 REM * LECTURE *
2002 REM ============
2003 :
2010 IF MS<>8192 THEN K=64:GOTO 2030
2020 K=0
2030 :
2050 PRINT D$"Read dessin"
2060 FOR P=1 TO 40
2070 INPUT B(P)
2080 NEXT P
2150 PRINT D$":PRINT:RETURN
3000 REM ===========
3001 REM * DESSIN *
3002 REM ===========
3003 :
3005 HPLOT 1,1 TO 278,1 TO 278,159 TO 1,159 TO 1,1
3010 HCOLOR=3
3015 HPLOT X,Y
3020 HOME:VTAB 22:PRINT "I=haut / M=bas / K=droite / J=gauche"
3025 PRINT "S=sauvegardez"
3030 VTAB 1:GET Q$:PRINT
3040 IF Q$="I" THEN Y=Y-S:GOTO 3200
3050 IF Q$="M" THEN Y=Y+S:GOTO 3200
3060 IF Q$="K" THEN X=X+S:GOTO 3200
3070 IF Q$="J" THEN X=X-S:GOTO 3200
3080 IF Q$="C" THEN 3010
3090 IF Q$="S" THEN PRINT:RETURN
3100 GOTO 3030
3200 HPLOT TO X,Y:GOTO 3030
4000 REM ============
4001 REM * ECRITURE *
4002 REM ============
4003 :
4010 IF MS<>8192 THEN K=64:GOTO 4030
4020 K=0
4030 :
4050 PRINT D$"Write dessin"
4060 FOR P=1 TO 40
4070 PRINT B(P)
4080 NEXT P
4150 PRINT D$:PRINT:RETURN
```

truction capable de lire l'état d'un point (ou la couleur), il faudra restructurer les boucles en éliminant la boucle externe et en conservant les deux autres, une pour les lignes, l'autre pour les colonnes.

Calcul de zones et de périmètres

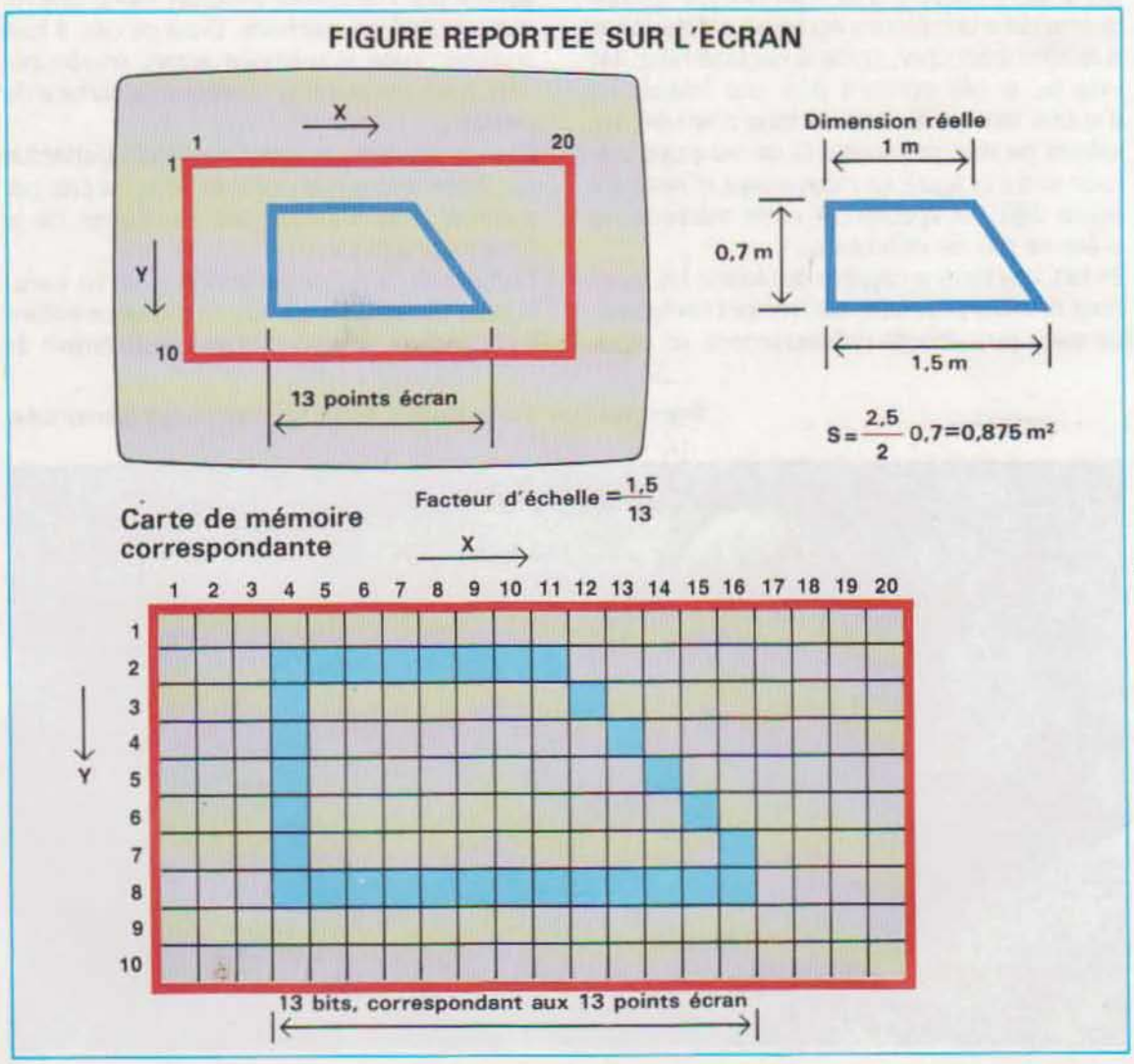
La lecture de l'état d'un point, ou mieux, de la mémoire qui le contient, peut se révéler indispensable dans certaines applications comme l'écriture de logiciels pour les jeux vidéo. Dans ces programmes, il est fréquent de déplacer une figure en vérifiant d'éventuelles collisions avec d'autres figures présentes à l'écran.
Dans certaines machines, ce contrôle s'effectue avec une instruction Basic, qui existe rarement dans les ordinateurs personnels, d'emploi plus général.
On a alors recours à la méthode de lecture directe dans la mémoire écran pour déterminer si la zone à occuper, après le déplacement, est vide ou si elle contient déjà une image. En d'autres termes, si, dans la zone d'arrivée, les valeurs ne sont pas toutes 0, on aura une collision entre la figure en mouvement et celle qui existe déjà. En apparence, cette méthode ne présente pas de difficultés.
En fait, une routine capable de réaliser les fonctions décrites peut être relativement complexe. Ce sujet sera abordé ultérieurement, en montrant la différence entre la réalisation de la fonction au moyen du logiciel d'application et ce que l'on peut obtenir de machines disposant de dispositifs spéciaux.
Un problème semblable à celui de la collision, mais bien plus simple, concerne la mesure des surfaces et des périmètres. Dans beaucoup d'applications graphiques, après avoir exécuté le dessin, on doit procéder à une phase de traitement pour mesurer les dimensions de l'objet représenté. Le dessin une fois terminé, toutes les dimensions sont présentes dans la mémoire graphique et il n'est donc pas difficile de concevoir quelques sous-programmes pour le calcul des caractéristiques géométriques.
Le problème peut être abordé de deux façons, selon le mode de saisie du dessin.
La première méthode prévoit que le dessin, généré par l'utilisateur avec un menu graphique, est déjà en mémoire. Dans ce cas, il faut analyser toute la mémoire écran, en récupérant, point par point, le contour et la surface du dessin.
Dans la seconde, le calcul du dessin s'effectue en même temps que sa saisie : c'est le cas, par exemple, des introductions au moyen de la tablette graphique.
Comme on l'a vu, ce périphérique est un transducteur de position qui fournit à chaque instant à l'ordinateur les coordonnées de l'élément de

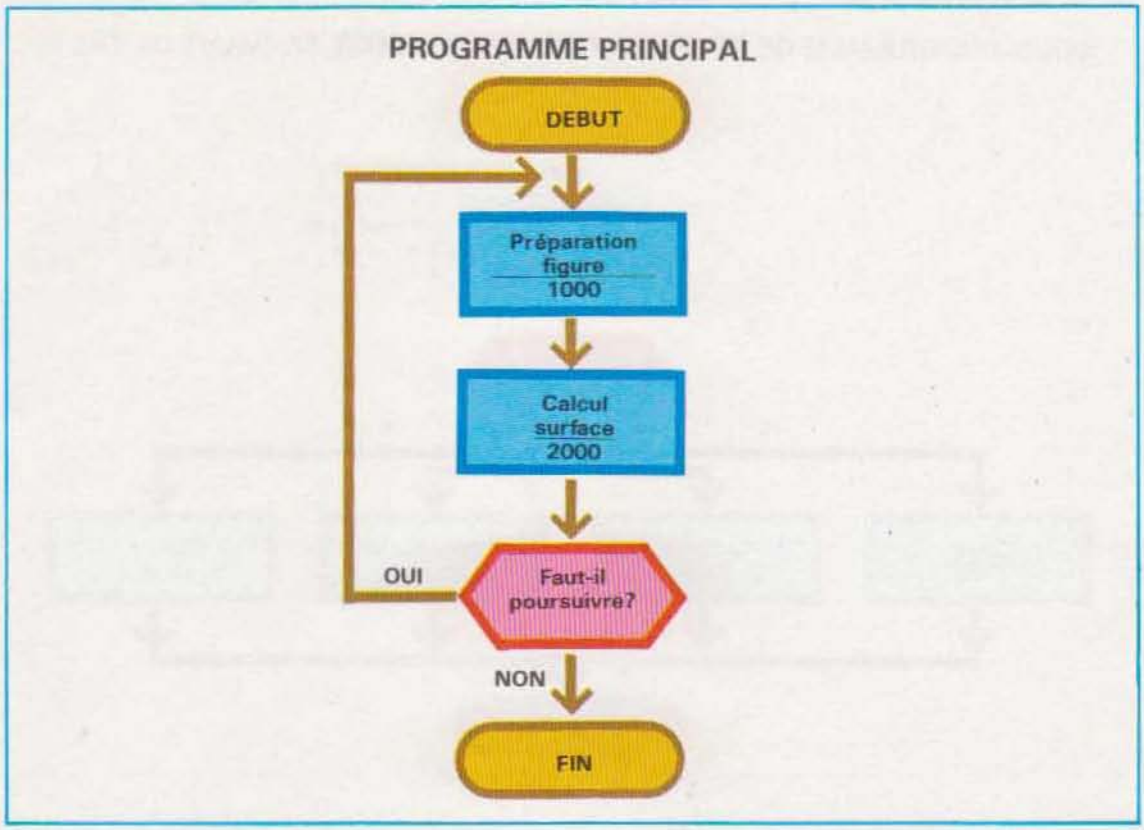
Superposition d'une fenêtre écran sur une image mémorisée.



Centro THC

contact ; le dessin est saisi en déplaçant cette référence le long du contour de la figure.
De cette manière, le calcul du périmètre est immédiat. Il consiste à mesurer la distance entre les divers points, en schématisant chaque déplacement par un segment de droite. Il est donc d'autant plus précis que les déplacements sont petits.
En réalité, l'utilisateur ne se rend pas compte de ce caractère discret le long du contour, le mouvement de référence est continu : c'est le rôle du logiciel de le diviser en éléments suffisamment petits pour assurer une bonne précision, sans toutefois trop ralentir l'exécution du programme.
La surface est calculée une fois que le dessin est terminé. Dans ce cas aussi, la méthode la plus simple consiste à examiner la mémoire écran et à effectuer le calcul sur l'image mémorisée (voir graphique en bas de page).
Pour illustrer la méthode, nous avons choisi un trapèze dont la surface calculée mathématiquement vaut 0.875 m² (les dimensions sont en mètres). Supposons l'image saisie et représentée avec la grande base (13 points écran) et un facteur d'échelle de 1,5/13 ; chaque point écran vaut 0,11...m et la hauteur en points écran 6,06... (hauteur en point écran = 13/1.5 * 0.7).
Ce dimensionnement, choisi pour des raisons graphiques, se révèle inefficace. En réalité, si l'on doit représenter ce dessin, l'échelle la plus adaptée est 100 ; ainsi la base devient 1.5 × 100 = 150 points écran et la hauteur 70 ; on obtient une meilleure précision.
Dans la partie inférieure du schema ci-dessous,





on montre qu'on obtient une carte mémoire correspondant à l'image. Les numérotations ne sont pas celles de l'écran, mais sont relatives à une zone de l'écran autour de la figure ; cependant, la présentation de la méthode n'en demeure pas moins valable. Le calcul de la surface consiste à examiner l'aire de mémoire concernée et à déterminer si chaque point se trouve, ou non, à l'intérieur de la figure. La somme des points intérieurs fournit la surface à l'échelle.

L'exploration de la mémoire se fait selon deux boucles, la plus externe selon l'axe Y (dans l'exemple de 1 à 10) et la plus interne selon l'axe X (de 1 à 20).

En posant Y = 1, et en examinant tous les emplacements de mémoire appartenant à cette ligne, on ne repère aucun bit actif ; donc sur la ligne Y = 1 il n'existe pas de points dessinés. Pour Y = 2, on rencontre en colonne 4 le 1er bit actif, puisque c'est le 1er mémorisé (dans la variable X1) comme début d'une ligne interne de la figure. En continuant l'exploration de la ligne, chaque fois que l'on rencontre un bit actif (après le 1er) sa position est rangée dans la variable XF. A la fin de la boucle interne (X de 1 à 20), on aura XF = 11 et la différence XF — XI fournit la longueur de la 1re ligne interne (en particulier pour Y = 2 : cette ligne est la petite base du trapèze examiné).

L'échelle du dessin étant 1,5/13, la longueur du segment en dimensions réelles est (1.5/13) * (XF-XI) = (1.5/13) * 8 = 0.92... m.

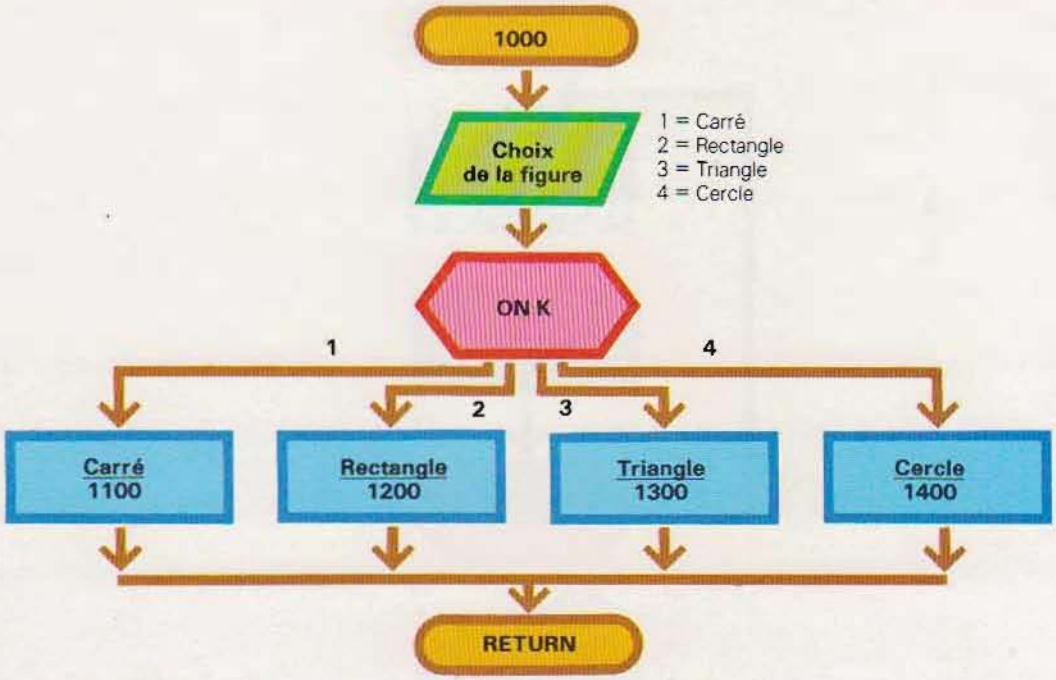
Remarquons la différence avec la mesure réelle de la petite base (1 m), due au facteur d'échelle adopté. Avec la valeur 1,5/13, la petite base devrait être longue 8.66... points écran, mais elle doit être ramenée à 8, avec l'erreur qui en découle. (o)

En choisissant une échelle mieux adaptée, cette erreur devient négligeable. Le segment ainsi mesuré (longueur : 0,92) a une dimension, selon l'axe Y, égale à un point écran, c'est-à-dire, 1,5/13 mètres. En multipliant cette quantité par la longueur, on obtient la valeur du premier élément de la surface.

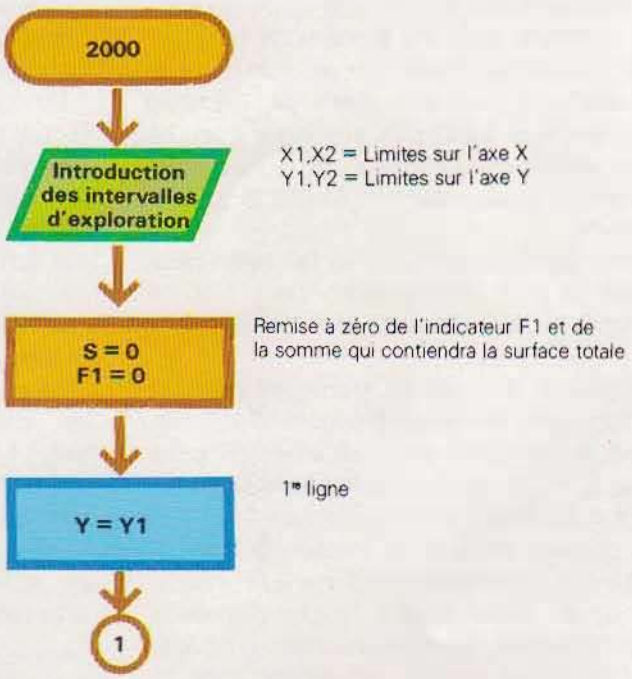
Le processus se poursuit pour toutes les lignes en additionnant chaque élément de surface ainsi trouvé.

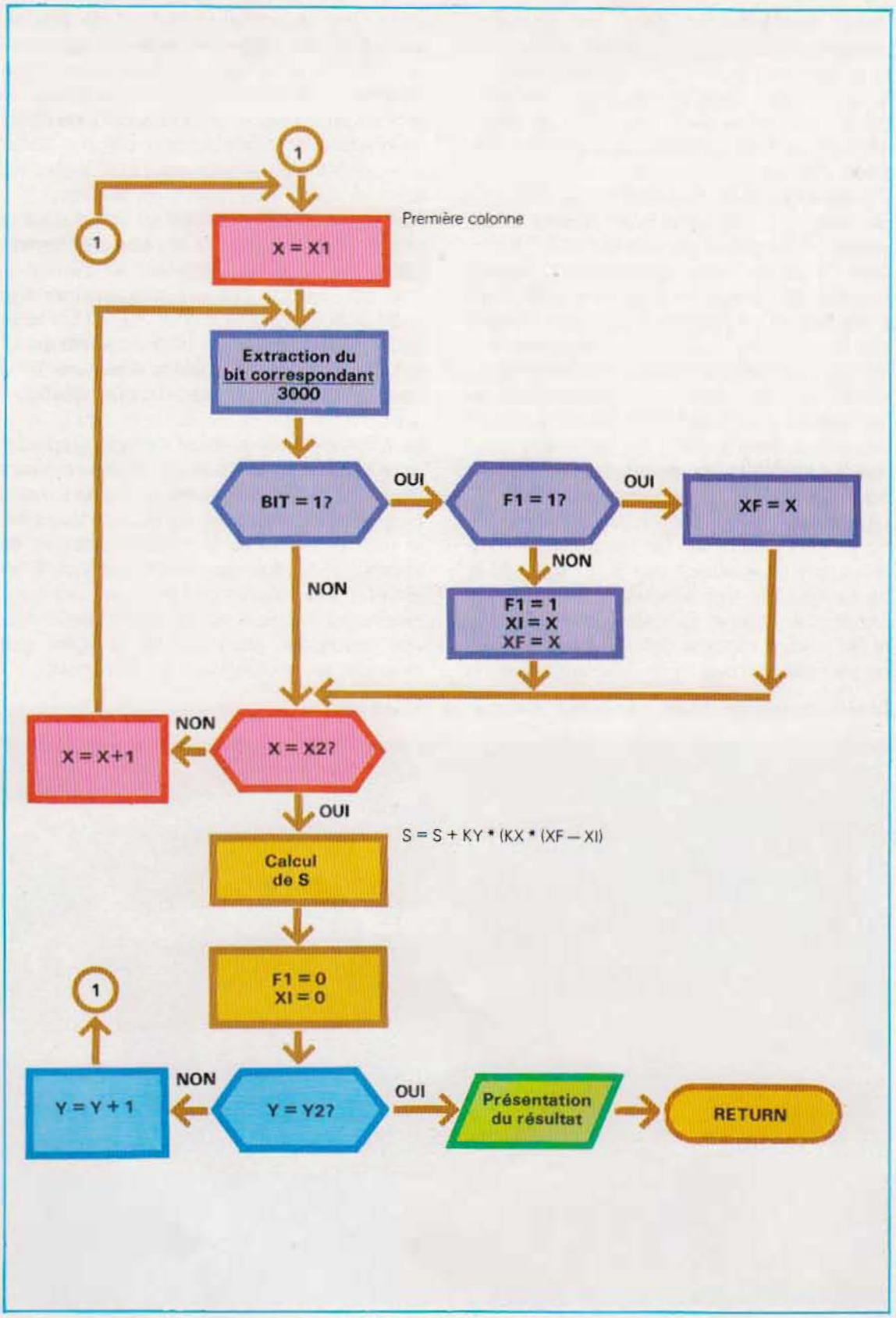
(o) La mémoire de l'écran et l'écran ne peuvent être adressés avec des valeurs réelles.

SOUS-PROGRAMME DE PREPARATION DE LA FIGURE SERVANT DE TEST



SOUS-PROGRAMME DE CALCUL DE LA SURFACE

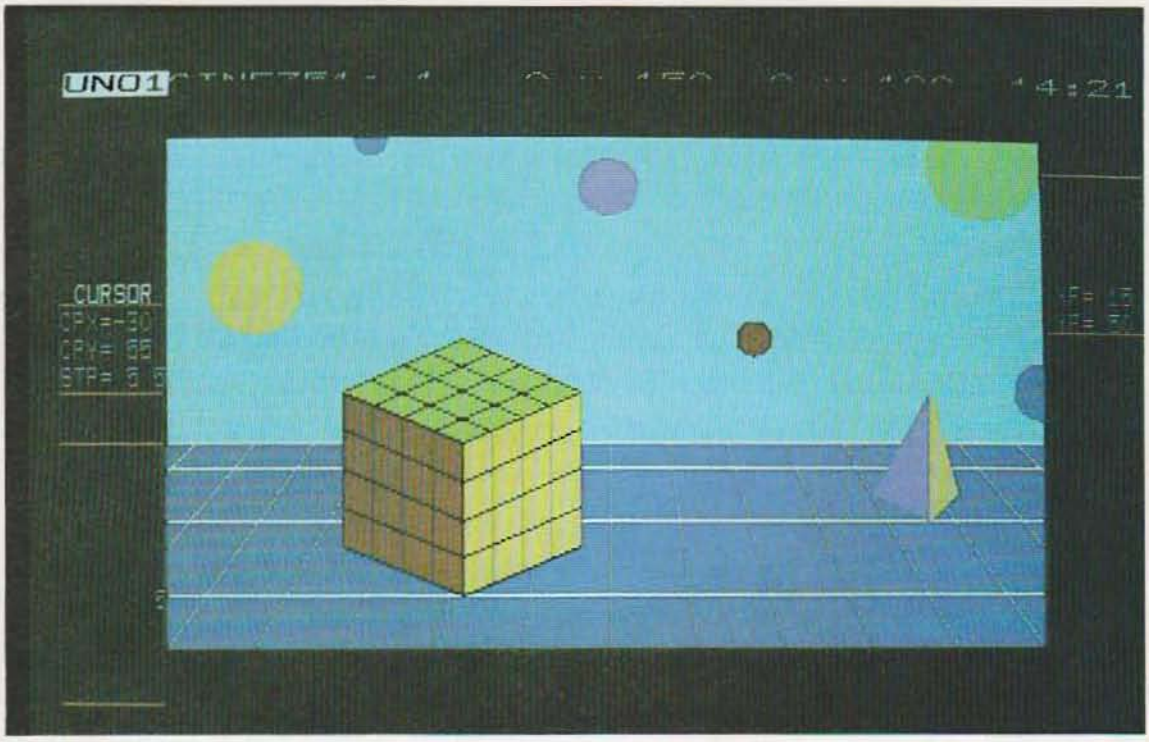



1
1
Première colonne
X = X1
Extraction du bit correspondant 3000
BIT = 1?
OUI
F1 = 1?
OUI
XF = X
NON
NON
F1 = 1
XI = X
XF = X
X = X+1
NON
X = X2?
OUI
S = S + KY * (KX * (XF — XI)
Calcul de S
F1 = 0
XI = 0
1
Y = Y + 1
NON
Y = Y2?
OUI
Présentation du résultat
RETURN

Notre problème est traité par le sous-programme 2000, les autres étant chargés de la préparation de certaines figures géométriques à l'écran. Dans le programme, on a recours, avec quelques variantes, au sous-programme déjà présenté pour le calcul de l'emplacement de mémoire.
Contrairement à la précédente qui avait un but démonstratif, cette version est réduite à l'essentiel à l'exception des lignes 3142 à 3150. Dans ce cas particulier, le temps de traitement est important surtout en Basic interprété. Pour éviter de longues attentes, nous avons introduit ces lignes (3142 à 3150) qui remplissent le contour de l'image au fur et à mesure du traitement. Toujours en suivant la même logique, on a introduit l'indicateur F2 qui peut être supprimé, même dans la 2071. Cet indicateur interrompt l'activation des points écran quand les coordonnées sont extérieures à la figure.
La logique employée est volontairement simple et peu fonctionnelle. Le remplissage n'est exact que pour autant que la 1re ligne de la figure n'est pas trop éloignée de la verticale.
De plus, les figures géométriques présentées et les valeurs relatives des surfaces calculées ne sont pas précises et en adéquation avec le reste du programme. Le but est de montrer comment se servir simplement du sous-programme 2000 tout en évitant des complexités en d'autres points. En particulier, le dessin des figures et la surface calculée par le programme ne correspondent pas aux surfaces calculées analytiquement. La différence est liée à la manière de réaliser les dessins.
Pour le carré, par exemple, en introduisant la valeur 10 pour le côté, et, comme origine de la figure, le point de coordonnées 10, 10, le premier côté vertical s'obtient en traçant un segment entre l'origine et le point 10, 20. De cette manière, la longueur du côté en points écran est 11 ; la surface résultante diffère de 10% de celle calculée : cet écart disparaît si la figure est correctement dessinée.
La méthode présentée est de type graphique dans la mesure où la surface obtenue provient du traitement graphique de la figure. Il existe aussi les méthodes analytiques avec lesquelles le résultat (calcul de la surface) s'obtient en appliquant des formules mathématiques. Mais elles ne sont valables que dans des cas particuliers (par exemple, polygônes) et demandent une description analytique de la figure (par exemple, les coordonnées des sommets).

Représentation graphique d'un cristal cubique sur ordinateur.

J. Pickerell/Marka



Centre THC

Image graphique tridimensionnelle (3 D) sur écran couleur.

PROGRAMME DE CALCUL DE SURFACES

Version Siprel 240, Apple et compatibles

```
10 REM
20 REM fichier=surfaces
30 REM
40 KY=1:KX=1
50 GOSUB 1000
60 GOSUB 2000
70 REM
80 INPUT "Poursuivre (O/N) ";A$
90 IF A$="O" GOTO 50
92 END
1000 REM figure d'essai
1005 REM
1010 HGR
1015 PRINT "Sélection(1-2-3-4)?"

1020 GET R
1025 IF R<1 OR R>4 GOTO 1015

1030 ON R GOSUB 1100,1200,1300
     ,1400
1040 RETURN
1050 REM
1052 REM ■■■■■
1054 REM
1100 REM carré
1105 REM
1110 INPUT "Coté ";L
1115 INPUT "Position (X,Y) ";
     X0,Y0
1120 HPLOT X0,Y0 TO X0+L,Y0
     TO X0+L,Y0-L TO X0,Y0-L
     TO X0,Y0
1121 A=L*L
1122 PRINT "Surface = ";A
1125 RETURN
1130 REM
1132 REM ■■■■■
1134 REM
1200 REM rectangle
1210 INPUT "Côtés (L,l) ";LX,LY
1215 INPUT "Position (X,Y) ";X0
     ,Y0
1220 HPLOT X0,Y0 TO X0+LX,Y0
     TO X0+LX,Y0-LY TO X0,Y0-LY
     TO X0,Y0
1221 A=LX*LY
1222 PRINT "Surface = ";A
1225 RETURN
1230 REM
1232 REM ■■■■■
1234 REM
1300 REM triangle
1305 REM
1310 INPUT "Base et hauteur ";B
     ,H
1315 INPUT "Position (X,Y) ";X0
     ,Y0
1320 HPLOT X0+B,Y0 TO X0,Y0 TO
     X0,Y0+H TO X0+B,Y0
1321 A=B*H/2
1322 PRINT "Surface = ";A
1325 RETURN
1330 REM
1332 REM ■■■■■
```

```
1334 REM
1400 REM cercle
1405 REM
1410 INPUT "Rayon ";R
1415 INPUT "Centre (X,Y) "
     ;X0,Y0
1520 X1=X0+R:Y1=Y0
1525 FOR A=-0.05 TO 6.5 STEP
     0.05
1530 X2=X0+R*COS(A)
1535 Y2=Y0+R*SIN(A)
1540 HPLOT X1,Y1 TO X2,Y2
1545 X1=X2:Y1=Y2
1550 NEXT A
1555 A=3.14*R^2
1556 PRINT "Surface = ";A
1560 RETURN
2000 REM
2005 REM surfaces
2010 REM
2015 INPUT "Limites axe X "
     ;X1,X2
2020 INPUT "Limites axe Y "
     ;Y1,Y2
2025 S=0:F1=0:XI=0:XF=0
2030 FOR Y=Y1 TO Y2
2035 FOR X=X1 TO X2
2040 GOSUB 3000
2045 IF BIT=0 GOTO 2050
2050 IF F1=1 THEN XF=X:GOTO
     2060
2055 XI=X:XF=X:F1=1
2060 NEXT X
2062 PRINT "Y = ";Y;"XI = "
     ;XI;"XF = ";XF
2065 S=S+KY*(KX*(XF-XI))

2070 F1=0:XI=0.XF=0
2071 F2=0
2075 NEXT Y
2080 PRINT "Surface = ";S
2082 GET D$
2085 RETURN
2090 REM
2092 REM *****
2094 REM
3000 REM extraction du bit
3005 REM
3010 REM entrées:
3015 REM     X,Y       X,Y
3016 :
3020 REM
3025 REM sorties:
3030 REM    BIT=0,1   BIT=0,1
3035 REM
3036 :
3040 GY=INT(Y/8)
3045 IF Y<64 THEN MR=8192+GY*
     128:GOTO 3070
3050 IF Y<128 THEN MR=8232+(GY
     -8)*128:GOTO 3070
3060 MR=8272+(GY-16)*128
3070 A=Y-GY*8
3080 MY=MR+A*1024
3090 GX=INT(X/7)
3100 MX=MY+GX:BIT=X-7*GX

3105 A%=PEEK(MX):B%=2^BIT

3110 NN=A%
3115 FOR K=0 TO BIT
3120 M=NN/2
3125 NN=INT(M)
3130 NEXT K
3135 BIT=0
3136 IF M<>NN THEN BIT=1
3142 IF F1=0 THEN RETURN
3143 C%=A%+B%
3145 IF F1=1 AND BIT=1 THEN
     F2=1
3146 IF F2=1 THEN RETURN
3147 IF BIT=1 THEN RETURN
3150 POKE MX,C%
3152 RETURN
3160 REM
3162 REM *****
3164 REM
```

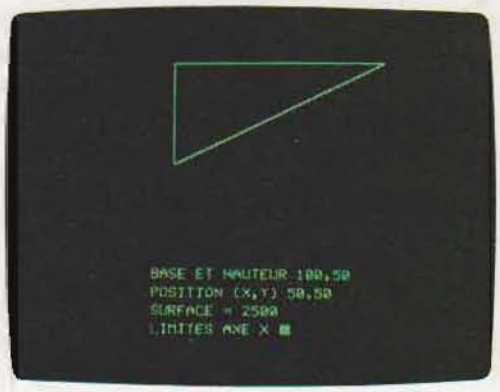
Les méthodes analytiques concernent donc plus la programmation scientifique que graphique car elles exigent quelques connaissances scientifiques.
La méthode présentée maintenant n'utilise aucune formule et s'applique à toute figure plane mais demande un temps de traitement certain des boucles de contrôle de la page graphique. Pour réduire ce temps d'attente on peut :

1/ se servir de la version compilée du programme
2/ et/ou circonscrire la zone de mémoire à explorer à l'intérieur de la figure.

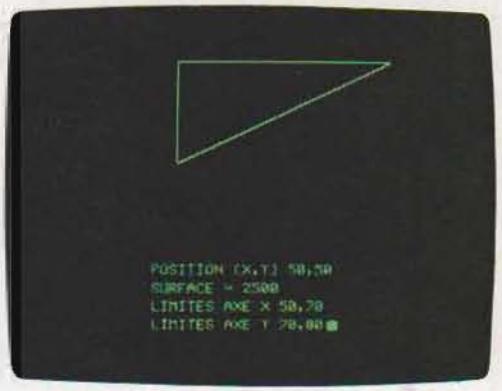
C'est pourquoi le sous-programme 2000 interroge sur les limites à poser dans l'exploration de l'écran le long des deux axes : ainsi les zones sans figures ne sont pas analysées. Autre avantage : ces limites permettent d'analyser plusieurs figures simultanées à l'écran. Il suffit d'appeler le sous-programme autant de fois qu'il y a de figures, en posant pour chacune d'elles les limites d'exploration appropriées. Attention : l'entrée des données doit bénéficier d'une attention particulière, car un mauvais calcul entraînerait de graves erreurs. Là encore, une série de contrôles s'avère nécessaire. Le plus simple est de dessiner, sur l'écran, une grille indiquant les coordonnées. Pour cela, il suffit de dessiner un rectangle en suivant les bords de l'écran et en marquant les côtés de référence, par exemple tous les dix points. Le programme exécutant ce tracé est le même que pour le tracé des axes cartésiens ou du cadre des histogrammes.
Une autre technique (analytique) consiste à mémoriser les coordonnées maximales et minimales au fur et à mesure que le dessin est créé.

PROGRAMME DE CALCUL DE SURFACES

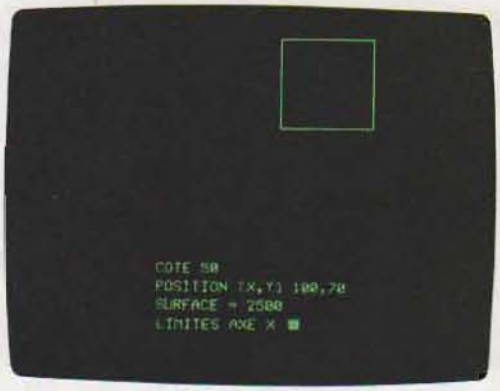
Examinons quelques exemples d'application du programme présenté pages 1561 et 1562.
L'écran montre ce qui apparaît après l'introduction de la valeur 3 en réponse à la ligne 1015. Le contrôle passe au sous-programme 1300, qui demande la base, la hauteur et la position du triangle et fournit la valeur analytique de la surface (1321, 1322).



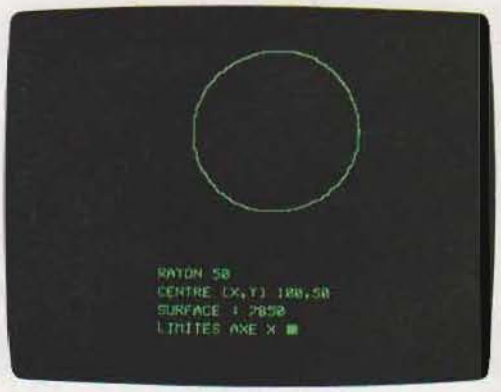
On passe ensuite la main au sous-programme 2000, qui calcule la surface en balayant la mémoire vidéo. Il demande les limites de balayage de l'écran le long des axes X et Y. Cet écran et le précédent illustrent la phase d'entrée de ces limites : l'axe X est exploré de la position 50 à 70, et l'axe Y de la 70 à la 80. A la fin s'affiche la valeur de la surface, déterminée par balayage.



Le 3e écran représente la même situation pour un carré positionné en 100,70 dont le côté mesure 50. La surface calculée analytiquement vaut 2500 ; la valeur déterminée par le programme sera affichée une fois entrées les limites d'analyse de l'écran.



Le dernier écran (cercle). Le système peut, même en mode graphique, réserver les 4 lignes inférieures pour dialoguer avec l'utilisateur.



Pour finir, on explore la surface rectangulaire, limitée par les valeurs maximales et minimales, avec une éventuelle marge. Dans le cas de plusieurs figures c'est à l'utilisateur de signaler au programme quand chacune d'elles est terminée.
Le fait de poser des limites permet d'éviter un autre type d'erreur. C'est le cas, notamment, d'un dessin aux contours mal définis. N'étant pas fermés, l'exploration et le calcul ultérieur du dessin ne s'achèvent qu'au moment où le programme rencontre le bord de l'écran. Résultat : la surface attribuée à cette partie serait bien supérieure à la réalité.
C'est pourquoi a été créée, dans le sous-programme 1400, la boucle utilisée pour tracer la circonférence.
Le cercle devrait être tracé en partant de l'angle 0 jusqu'à l'angle $2*\pi = 6{,}28$ (ce qui équivaut à un tour complet), mais la boucle utilise les limites - 0,05 et 6,3 afin d'assurer la clôture totale de la figure.

Instructions Basic pour l'archivage des figures

La méthode exposée pour obtenir une figure ne fait intervenir aucune instruction particulière. Elle est appliquée à toutes les machines dont les adresses mémoire graphiques sont connues.
Dans certains cas il existe des instructions de haut niveau et surtout transparentes, c'est-à-dire qui ne nécessitent aucun adressage, du moins en-dehors de la définition des coordonnées graphiques de la zone à mémoriser ou à redessiner.
Ce type d'instructions est étroitement lié à la structure de la machine. De ce fait, leur syntaxe, parfois même les fonctions exécutées, sont particulières.
Voici par exemple les instructions disponibles sur l'Olivetti M20 :

POINT : Enregistre la couleur d'un point
GET : Mémorise une zone écran
PUT : Visualise l'image enregistrée avec une instruction GET

Les instructions GET et PUT utilisées ici sont différentes — dans leur syntaxe et dans leur objet — de celles employées avec des fonctions d'E/S.

POINT. Son format est :
A% =POINT(X,Y)

A l'exécution, le programme assigne à la variable entière A% une valeur numérique dépendant de la couleur du point de coordonnées X,Y. Cette valeur est 0 ou 1 pour un écran monochrome (0 = éteint, 1 = allumé), ou comprise entre 0 et 3 pour un écran 4 couleurs ou 0 et 7 pour le 8 couleurs.

Les valeurs des coordonnées X,Y peuvent aussi bien se rapporter au matériel, c'est-à-dire correspondre à des points de l'écran, qu'être définies par l'utilisateur. Avec ce type de machine, il est en effet admis de recourir à un système de référence différent de celui qui définit les points écran ; il emploie alors la **référence utilisateur**.

POINT s'emploie également dans une autre instruction graphique, mais dans un autre sens pour avoir deux curseurs, l'un de texte et l'autre graphique. La gestion du premier est identique à celle des autres machines, bien que les instructions soient formellement différentes. Cependant, celle du curseur graphique est peu courante.
Le format le plus simple de cette instruction est :

CURSOR POINT(X,Y)N,M

N et M indiquant un point visible (N=1) ou non visible (N=0) ainsi que la fréquence du clignotement du curseur (pour M=0, il ne clignote pas).

GET. Son format simplifié est :

GET(X_1,Y_1) - (X_2,Y_2),A%(0)

X_1,Y_1 et X_2,Y_2 définissant les sommets de la portion rectangulaire d'écran à mémoriser, et A%(0) le premier élément du tableau dans lequel seront rangées les valeurs. (N'oublions pas qu'en Basic, l'indice initial d'un tableau est 0). Une fois l'instruction exécutée, tous les bits correspondant à la section de l'écran définie par les coordonnées sont stockés dans A%(.). Outre l'état de chacun des bits, A%(.) contient les informations suivantes :

A%(0) : Base du rectangle défini par X_1,Y_1 et X_2, Y_2
A%(1) : Hauteur du rectangle
A%(2) : Ecran monochrome ou couleur.

A partir de A% (3), les adresses sont utilisées pour stocker les bits de la zone graphique décrite. Le dimensionnement du tableau A%(.) dépend de l'extension, en pixels, de la zone à archiver ; elle se calcule comme suit :

Dimension=((⌈base/16⌉*hauteur)*K)+3

K valant 1 pour un écran monochrome, 2 pour un écran 4 couleurs, et 3 pour un écran 8 couleurs, tandis que le quotient base/16 doit toujours être arrondi par excès (vers le haut).
L'ordinateur M20 admet une partition de l'écran en portions appelées fenêtres. Chacune d'elles est utilisable indépendamment des autres, par simple indication de son numéro.
La syntaxe complète de l'instruction GET est :

GET W% (X_1,Y_1) - (X_2,Y_2), A%(0)

W% étant la valeur numérique de la fenêtre (ce sujet sera traité plus loin).

PUT. Son format est :

PUT W%(X_1,Y_1) - (X_2,Y_2), A%(0),V

Là encore, le paramètre W% (fenêtre) est optionnel. Par rapport à l'instruction précédente, PUT contient en plus le paramètre V, qui exprime une opération logique à effectuer entre les numéros de couleur contenus dans le tableau A%(.) et ceux des couleurs présentes à l'écran dans le rectangle défini par les coordonnées spécifiées.
Quand V est omis, le tableau A%(.) est visualisé tel qu'il a été mémorisé. Si V est spécifié, la visualisation obtenue sera le résultat de l'opération logique exprimée par V entre les couleurs prédéfinies et celles de A%(.). L'option V prend les valeurs AND, OR, XOR, NOT, PSET et PRESET. AND, OR et XOR ont leur sens habituel. NOT indique le complément des numéros des couleurs présentes à l'écran. PSET ressort le tableau tel qu'il a été enregistré. PRESET donne le complément des numéros de couleur du tableau.

Les fenêtres

Dans bon nombre d'applications, il n'est pas inutile d'avoir un écran divisé en zones gérées séparément les unes des autres. Appelées **fenêtres,** ces zones se comportent, du point de vue du programme d'application, comme autant de petits écrans séparés. Elles sont le plus souvent utilisées pour présenter, simultanément, un dessin d'ensemble et des détails de celui-ci (grossissements).
Par exemple, pour tracer le graphique d'une fonction, on peut diviser l'écran en deux ou en trois parties dont une, généralement la plus grande, pour le graphique entier, et les autres pour l'analyse détaillée de petites portions du graphique.
Dans les systèmes capables de générer des fenêtres, l'opération se déroule généralement en deux phases. On commence par assigner des valeurs aux paramètres définissant les fenêtres (dimensions, position, etc.) ; on passe ensuite à la phase de travail sur la fenêtre adressée à l'aide du nombre ou du symbole qui lui a été associé précédemment.
Les possibilités offertes par la gestion des fenêtres varient d'une machine à l'autre. En fait, il en existe 3 grandes catégories :

- les machines dans lesquelles les fenêtres ne sont pas prévues
- les ordinateurs ne permettant de créer que 2 fenêtres, l'une pour le graphique, et l'autre pour le texte
- les ordinateurs permettant une gestion complète des fenêtres, tant du point de vue de leur nombre que du mode (graphique ou texte)

Les deux premières catégories sont pratiquement équivalentes. En effet, la visualisation d'une fenêtre de texte, généralement limitée à quelques lignes, ne peut être utile que dans les cas où l'on veut avoir simultanément un dialogue avec l'utilisateur et l'affichage de la page graphique. C'est le cas, par exemple, des ordinateurs compatibles Apple qui conservent une fenêtre de texte d'une hauteur de 4 lignes (instruction HGR). Il ne s'agit donc pas d'une division en fenêtres graphiques, laquelle reste néanmoins réalisable par l'utilisateur à l'aide de programmes relativement simples.

Programme de génération de fenêtres

Une fenêtre est une zone de l'écran, le plus souvent carrée ou rectangulaire, identifiée à l'aide d'un symbole ou d'un numéro. Pour l'utiliser le programme graphique doit être paramétré, c'est-à-dire qu'il doit gérer tous les positionnements et les déplacements en fonction des paramètres de la fenêtre.

Ce paramétrage n'est évidemment nécessaire que lorsque la partition de l'écran est obtenue à l'aide de programmes écrits par l'utilisateur. Avec les machines prévoyant des fenêtres au niveau du système, le paramétrage est transparent à l'utilisateur; il n'a qu'à appeler la fenêtre de travail.

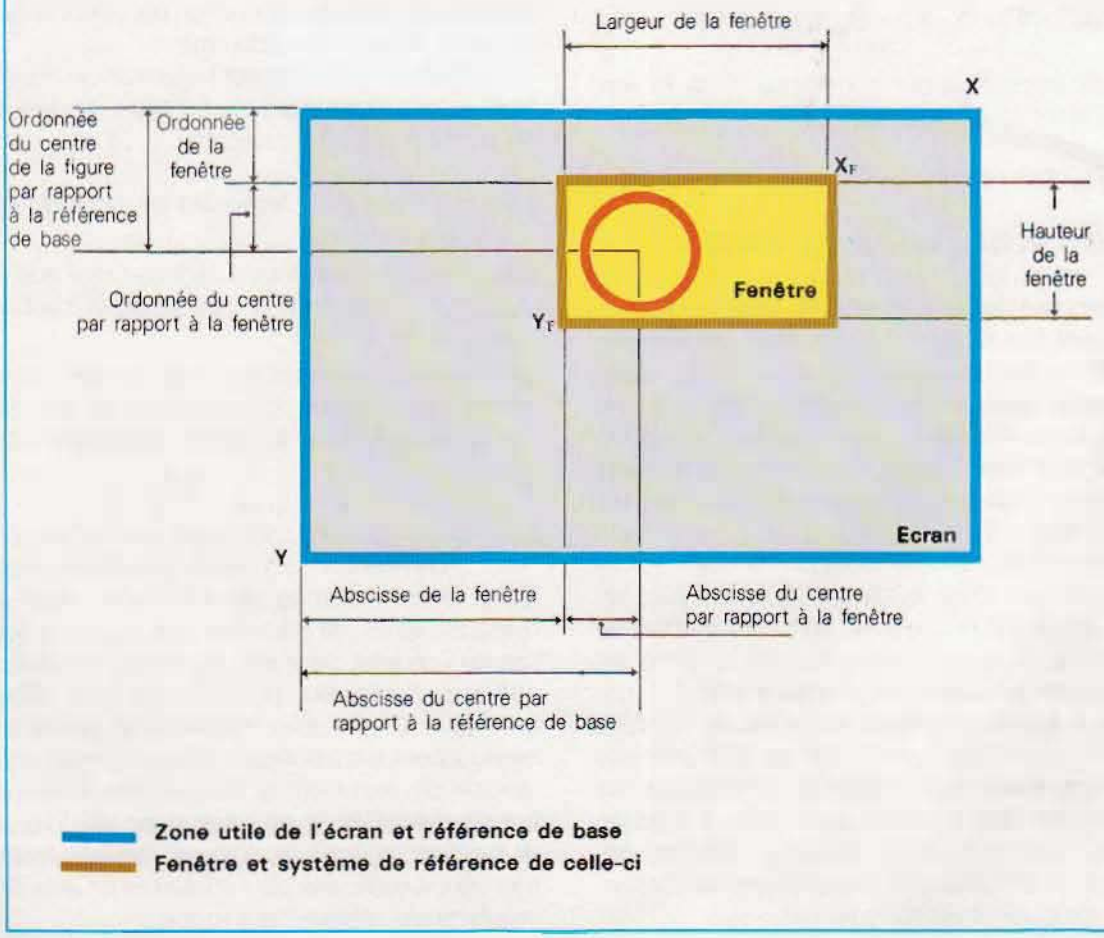
On a représenté ci-dessous le dessin d'une circonférence tracée à la fois par référence au système de base et à un système de référence coïncidant avec deux côtés d'une fenêtre. La gestion d'un dessin, à l'intérieur d'une fenêtre, demande que les coordonnées de construction soient exprimées par rapport au système de référence de la fenêtre.

La présentation d'un cercle de rayon R est obtenue à l'aide d'une boucle qui calcule les coordonnées de chaque point au moyen des formules suivantes :

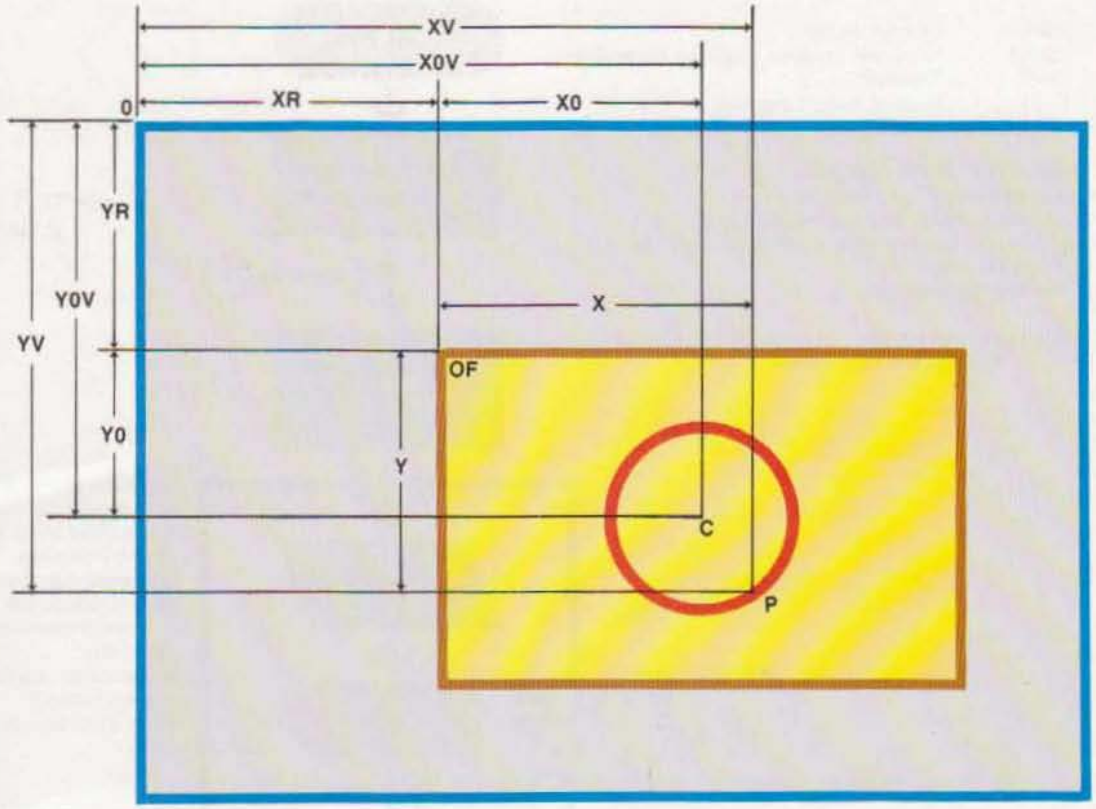
X=XO+R*COS(A)
Y=YO+R*SIN(A)

XO et YO étant les coordonnées du centre.

GESTION D'UNE FIGURE DANS UNE FENETRE



SYMBOLES UTILISES DANS LA GESTION DES FENETRES



Pour transférer le cercle dans la fenêtre, il suffit de définir les coordonnées du centre par rapport à celles de la fenêtre. Les symboles utilisés figurent ci-dessus. Les coordonnées X, Y d'un point quelconque du cercle par rapport à la fenêtre doivent être fournies sous forme de coordonnées XV, YV par rapport au système de base. Celui-ci est utilisé par l'ordinateur pour adresser les déplacements du crayon électrique. Donc :

XV=XR+X
YV=YR+X

De même, si elles se réfèrent au système de base, les coordonnées du centre deviennent :

Y0V=XR+X0
Y0V=YR+Y0

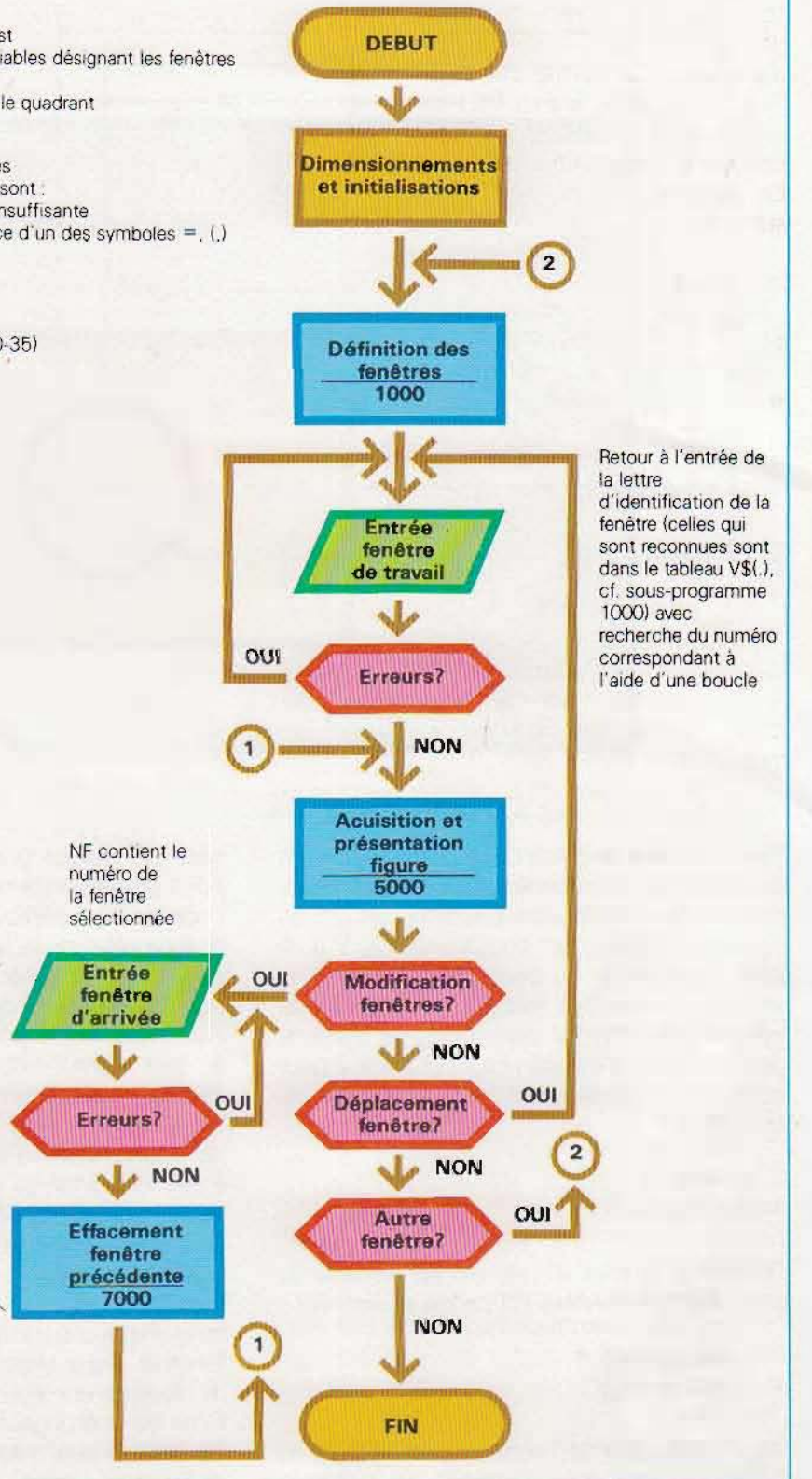
Pour positionner le cercle à l'intérieur de la fenêtre, il suffit donc d'ajouter, aux coordonnées de chaque point, celles de l'origine du point de référence de la fenêtre.
Dans ce cas particulier, les coordonnées de chaque point de la figure renvoient à celles de son centre, et la somme s'effectue aussi bien pour chaque point que pour le centre seul.
Autrement dit, le positionnement d'une figure au sein d'une fenêtre se réduit à une simple translation du système de référence. Ces deux façons de procéder (sur chaque point ou sur les coordonnées d'une seule référence) sont également valables pour des figures n'ayant pas de centre, à condition que les coordonnées de chaque point se réfèrent à un point fixe comme un sommet.
L'organigramme de la page suivante est un programme d'illustration qui manipule plusieurs fenêtres. Dans chacune d'elles, il est possible de dessiner une figure par référence à un système de coordonnées, solidaire de la fenêtre. On dispose ainsi d'autant d'écrans séparés que de fenêtres créées.

GENERATION DES FENETRES

D$(4)	Chaînes de test
V$(16)	Noms des variables désignant les fenêtres
Q(16)	Quadrant
P(16)	Position dans le quadrant
H(16),L(16)	Dimensions

Les chaînes de test, chargées dans une instruction DATA, sont :
1/ commande de longueur insuffisante
2/ erreur de syntaxe, absence d'un des symboles =, (,)
3/ mot-clé erroné
4/ erreur de paramètre

Format de l'instruction :
F=WINDOW(1-3-70-35)



Ce programme est structuré comme un interprétateur, puisque la définition des fenêtres est formulée sous forme de chaîne : le sous-programme 1000 l'interprète en déduisant les paramètres. La forme choisie coïncide avec l'instruction Basic la plus courante pour générer une fenêtre dans les systèmes ayant prévu ce type d'implémentation.
La syntaxe que l'on veut simuler est la suivante :

- une première lettre quelconque identifie la fenêtre en cours de génération
- le symbole =, informe le système que le mot suivant est le mot-clé
- un mot-clé (WINDOW) définit la fonction désirée (création d'une fenêtre)
- une série de paramètres entre parenthèses dessinent la géométrie de la fenêtre.

Ces paramètres sont :

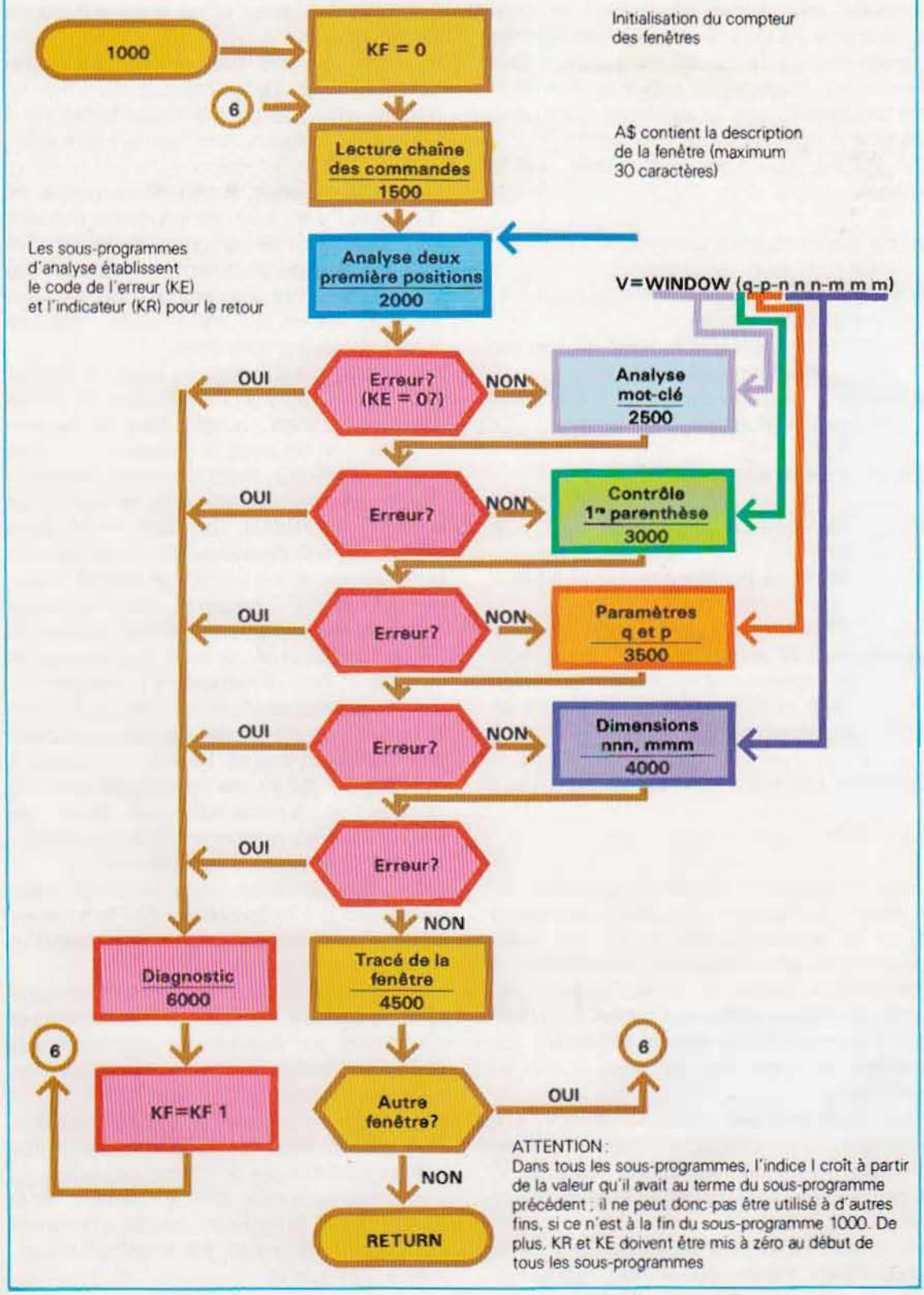
q : Identifie le quadrant de l'écran (1, 2, 3, 4)
p : Donne la position (de 1 à 4) à l'intérieur du quadrant
nnn, mmm : Valeurs numériques constituées chacune d'un maximum de trois chiffres et représentant respectivement la largeur et la hauteur de la fenêtre en points écran.

La chaîne à fournir a donc la forme :

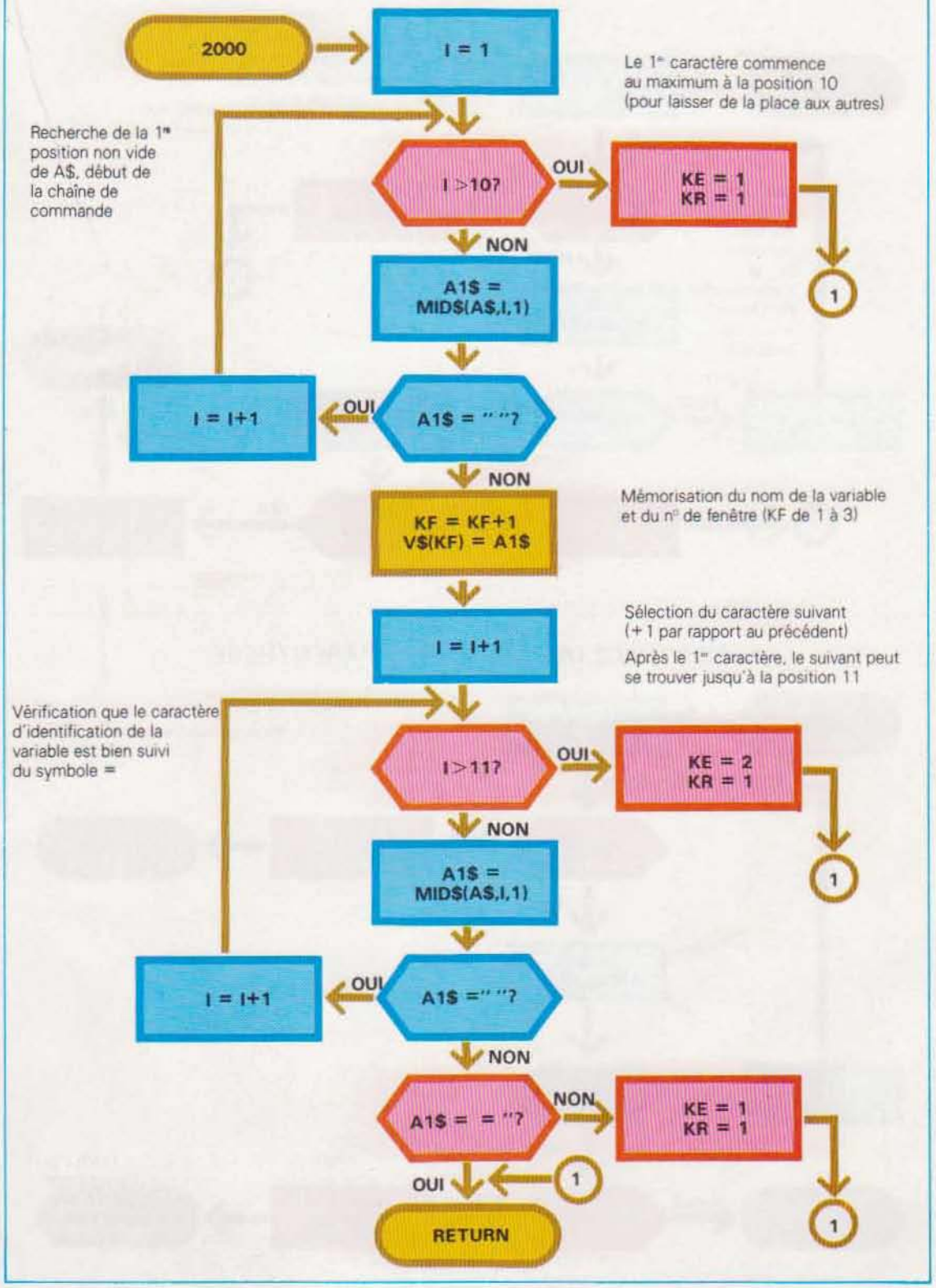
V=WINDOW(1 - 4 - 50 - 20)

Cette formule décrit une fenêtre identifiée par la lettre V, occupant le 1er quadrant en position 4, et de dimensions 50 × 20. Les sous-programmes qui prélèvent les paramètres sont structurés de manière à éliminer les éventuels espaces entre les différentes parties de la chaîne de commande : le mot-clé WINDOW, par exemple, peut être plus ou moins accollé au symbole =.
Seule limite imposée : l'absence d'espaces à l'intérieur des parenthèses contenant la définition des paramètres (ce choix résulte de la nécessité de ne pas compliquer excessivement l'exemple).
Les valeurs définissant les dimensions de la fenêtre (nnn, mmm) contiennent indifféremment de 1 à 3 caractères. Elles seront identifiées grâce à la position du symbole — séparation entre les zones et de la parenthèse de fermeture.
En réalité, une telle méthode n'est pas nécessaire pour créer une fenêtre. Cet exemple a été formulé ainsi afin d'illustrer une technique à laquelle on a recours dans l'écriture d'un interpréteur.
Dans notre exemple, le mot-clé est unique, ce qui permet d'en faire une constante utilisable comme élément de comparaison lors des diagnostics. Plus généralement, on peut créer un tableau contenant une série de mots-clés permettant, lors de leur identification, d'activer autant de sous-programmes.
Les modifications à apporter sont, en théorie, très simples : lors de l'initialisation, les mots-clés prévus sont chargés dans un tableau (comme les diagnostics d'erreur). Le sous-programme 1500 contient une boucle d'identification du mot-clé introduit. Si le résultat est positif (mot identifié), les autres termes de la chaîne sont ensuite analysés, la chaîne se comportant comme une instruction dans un langage particulier à l'utilisateur. Toutefois, cette méthode présente un inconvénient : elle entraîne une prolifération de sous-programmes de contrôle et de prélèvement de paramètres.
Dans le cas examiné (mot-clé unique), les paramètres suivant le mot-clé sont obligatoirement dans un format imposé. Dans le cas général, il faut appeler autant de sous-programmes de contrôle et d'extraction qu'il existe de possibilités ; leur nombre est donc directement proportionnel à celui des mots-clés.
Même tres simplifiée, cette structure reflète néanmoins le fonctionnement de l'interpréteur Basic et est utilisable dans la construction d'un langage orienté graphique.
Le programme fourni est purement démonstratif et ne contient pas les contrôles nécessaires concernant, par exemple, les dimensions des fenêtres et celles du dessin ; ou encore les variations d'échelle. L'utilisateur doit fournir des valeurs correctes en entrée. Les modifications sont toutefois faciles : il suffit de vérifier que les coordonnées de chacun des points du dessin correspondent bien à l'intérieur de la fenêtre. Sinon, la figure est coupée à proximité des bords de la fenêtre (cette opération s'appelle « écrêtage »).

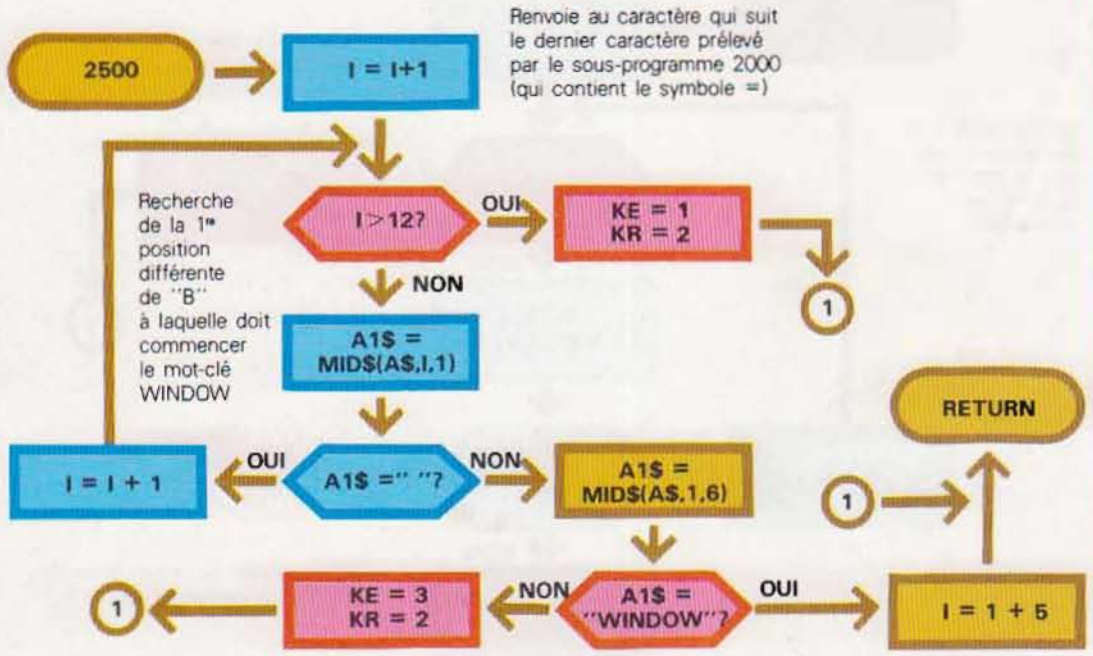
SOUS-PROGRAMME DE DEFINITION DES FENETRES



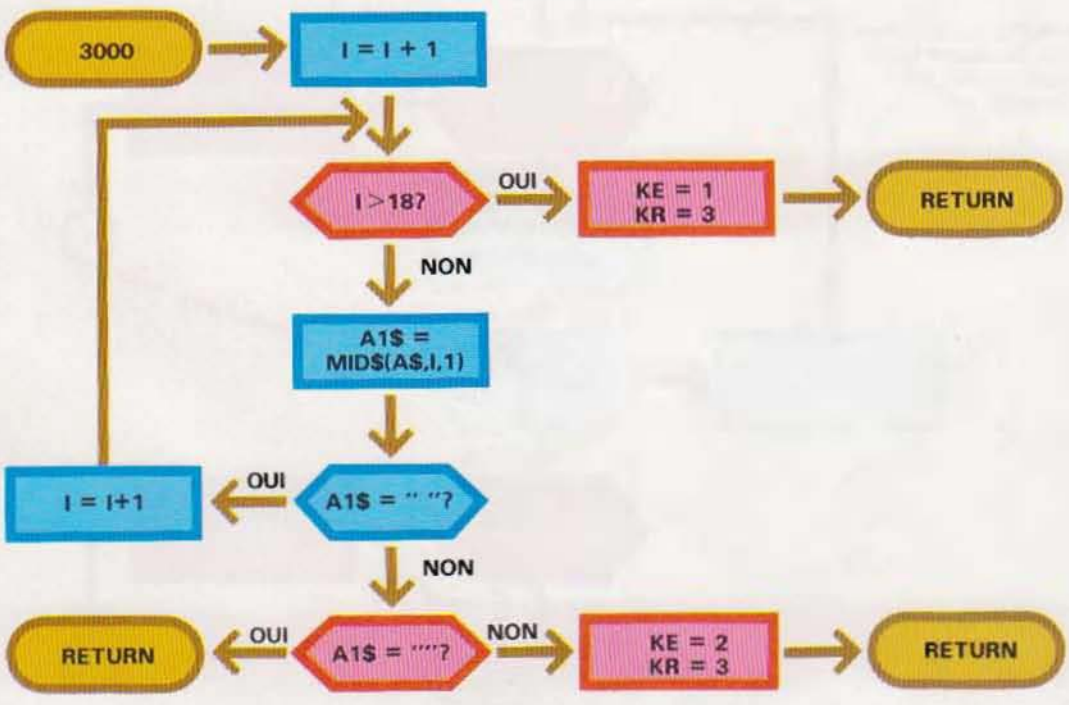
SOUS-PROGRAMME D'ANALYSE DES 2 PREMIERS CARACTERES



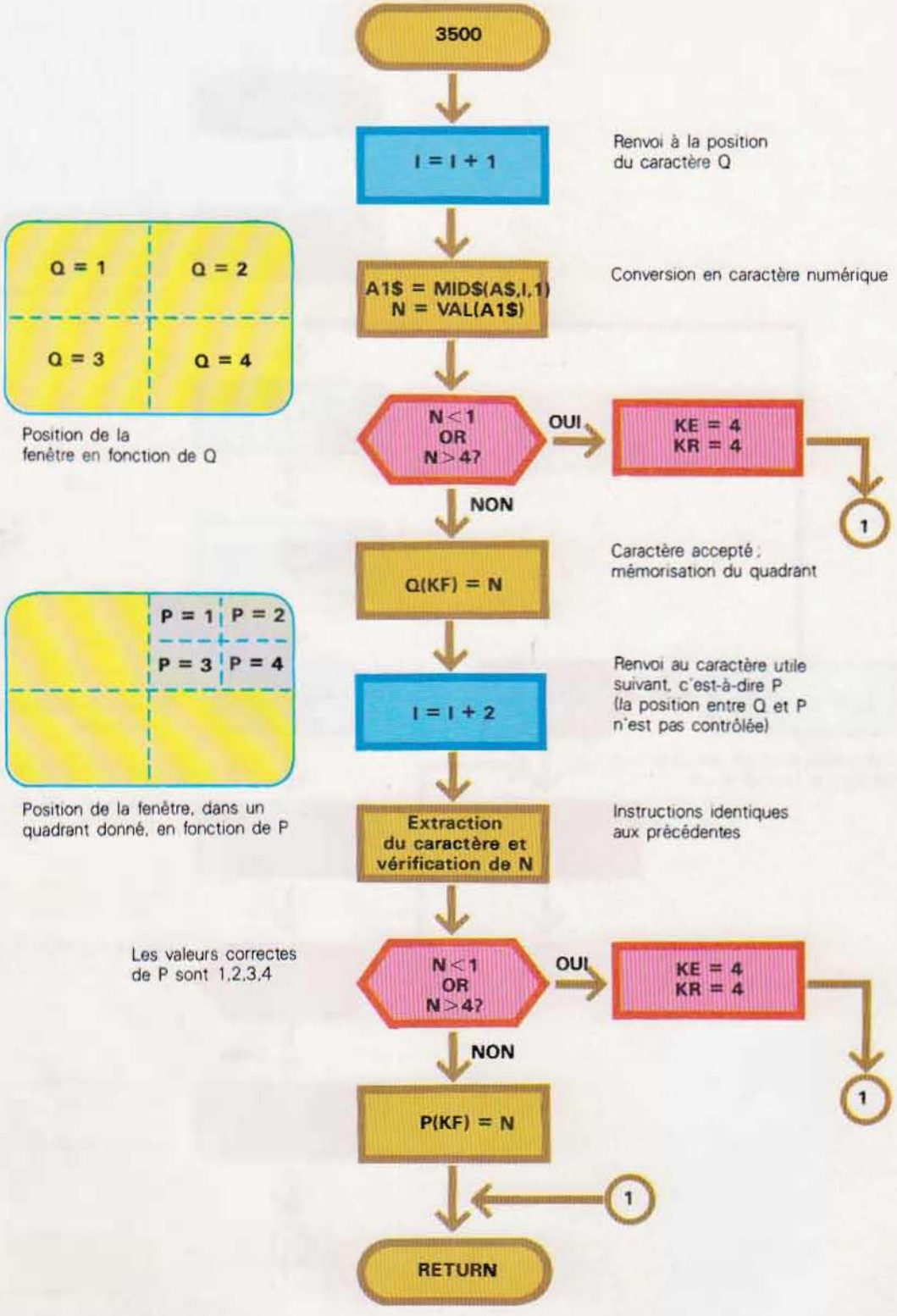
ANALYSE DU MOT-CLE



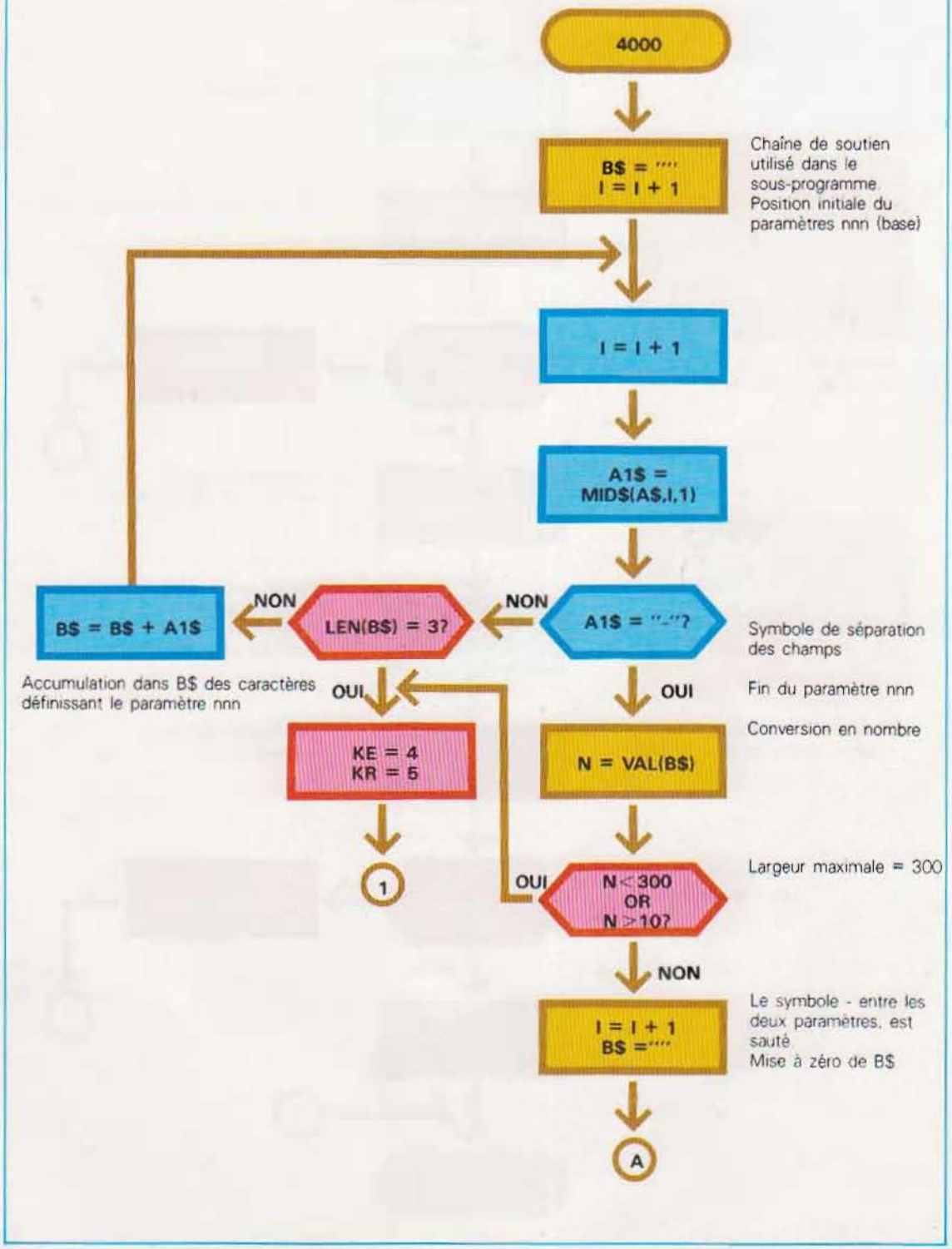
EXISTENCE DE LA PREMIERE PARENTHESE

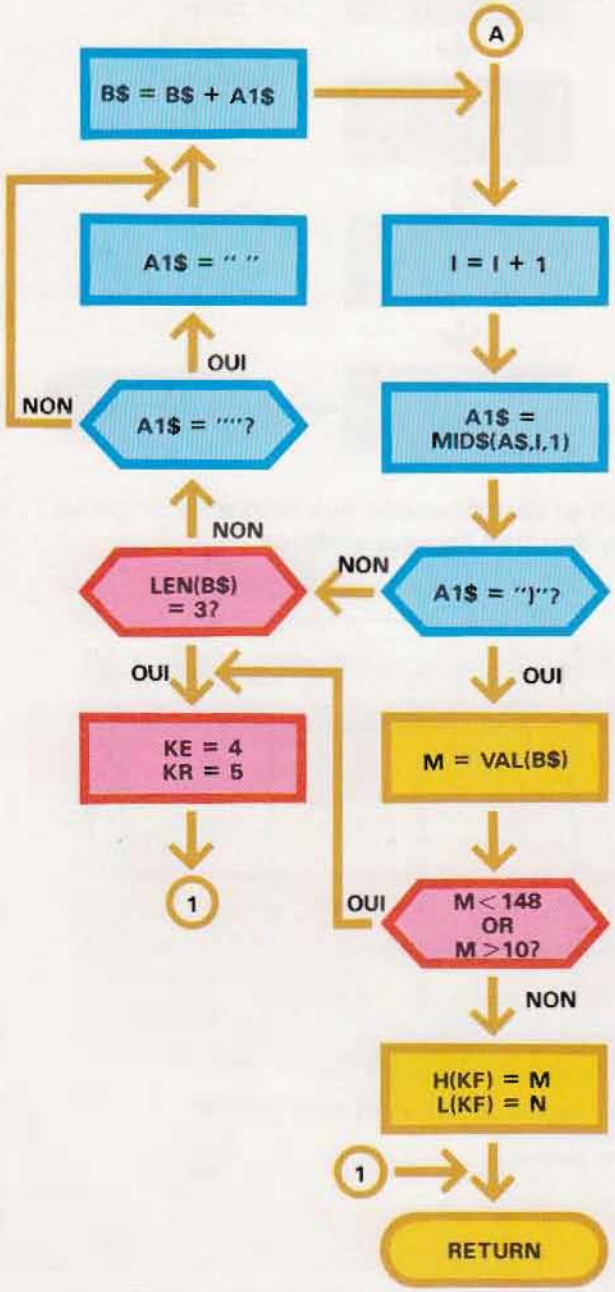


CONTROLE DES PARAMETRES Q ET P (QUADRANT ET POSITION)



CONTROLE DES DIMENSIONS (PARAMETRES nnn, mmm)

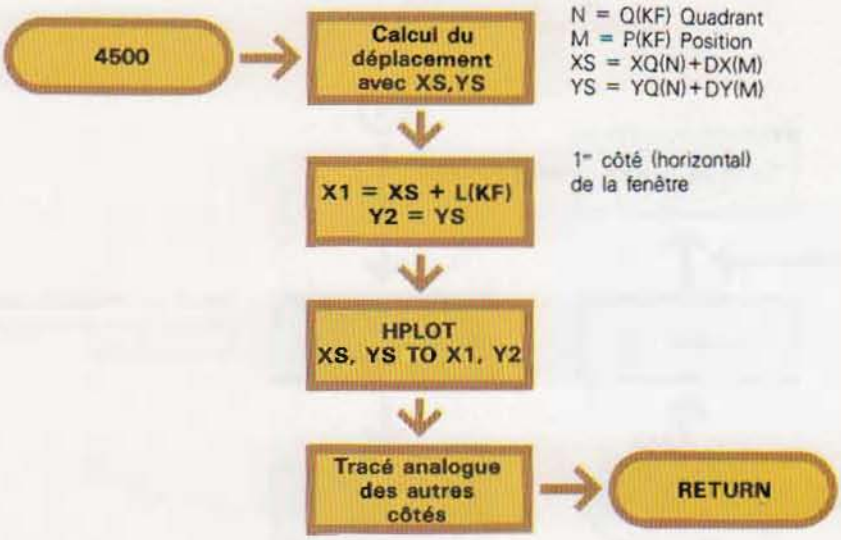




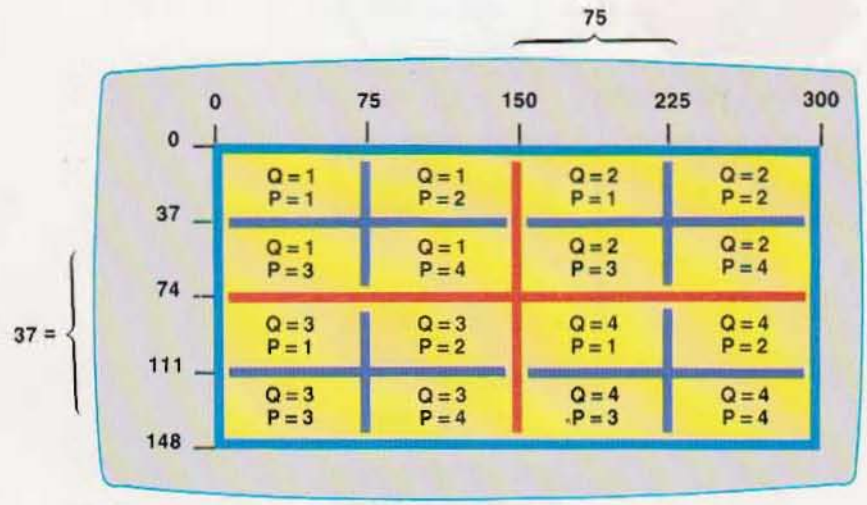
Nouvel incrément nécessaire pour la boucle d'extraction du 2e paramètre

Fin du paramètre mmm (hauteur)

SOUS-PROGRAMME DE PRESENTATION DE LA FENETRE



Division de l'écran et coordonnées des nouvelles origines en fonction des paramètres



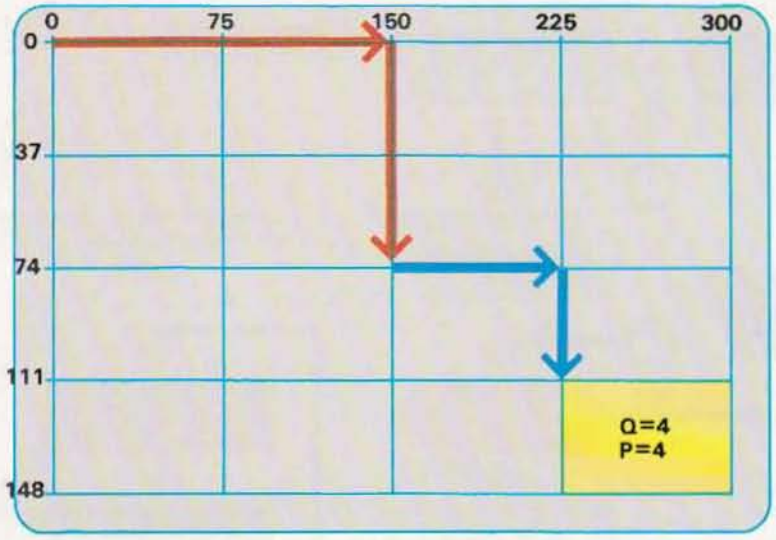
Le point de référence de chaque quadrant (subdivisions rouges) est le point extrême en haut à gauche. On obtient donc les valeurs suivantes :

Quadrant	Coordonnées du sommet
1	0,0
2	150,0
3	0,74
4	150,74

qui sont mémorisées (avec une instruction DATA) dans XQ(4) et YQ(4). De même, l'origine de la position, à l'intérieur d'un quadrant, est définie par les coordonnées du sommet en haut à gauche (mémorisée dans DX(4) et DY(4) par rapport à celle du quadrant. Les valeurs sont :

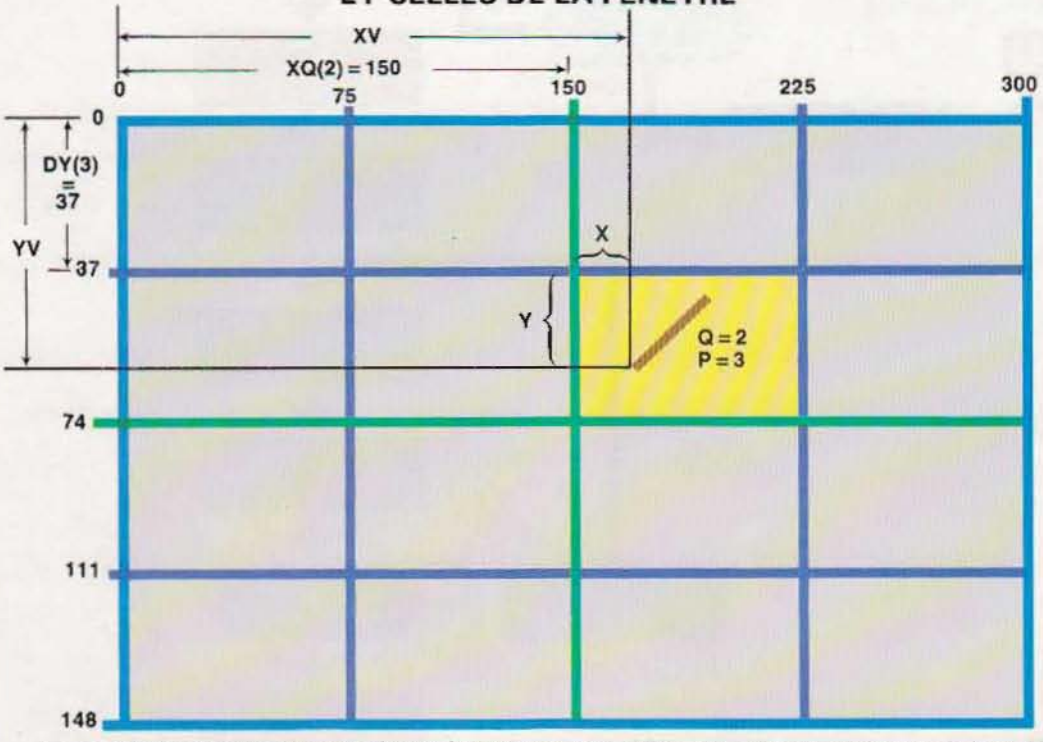
XQ(.) = 0,150,0,150
YQ(.) = 0,074,74
DX(.) = 0,75,0,75
DY(.) = 0,0,37,37

UTILISATION DES COORDONNEES DE QUADRANT ET DE POSITION



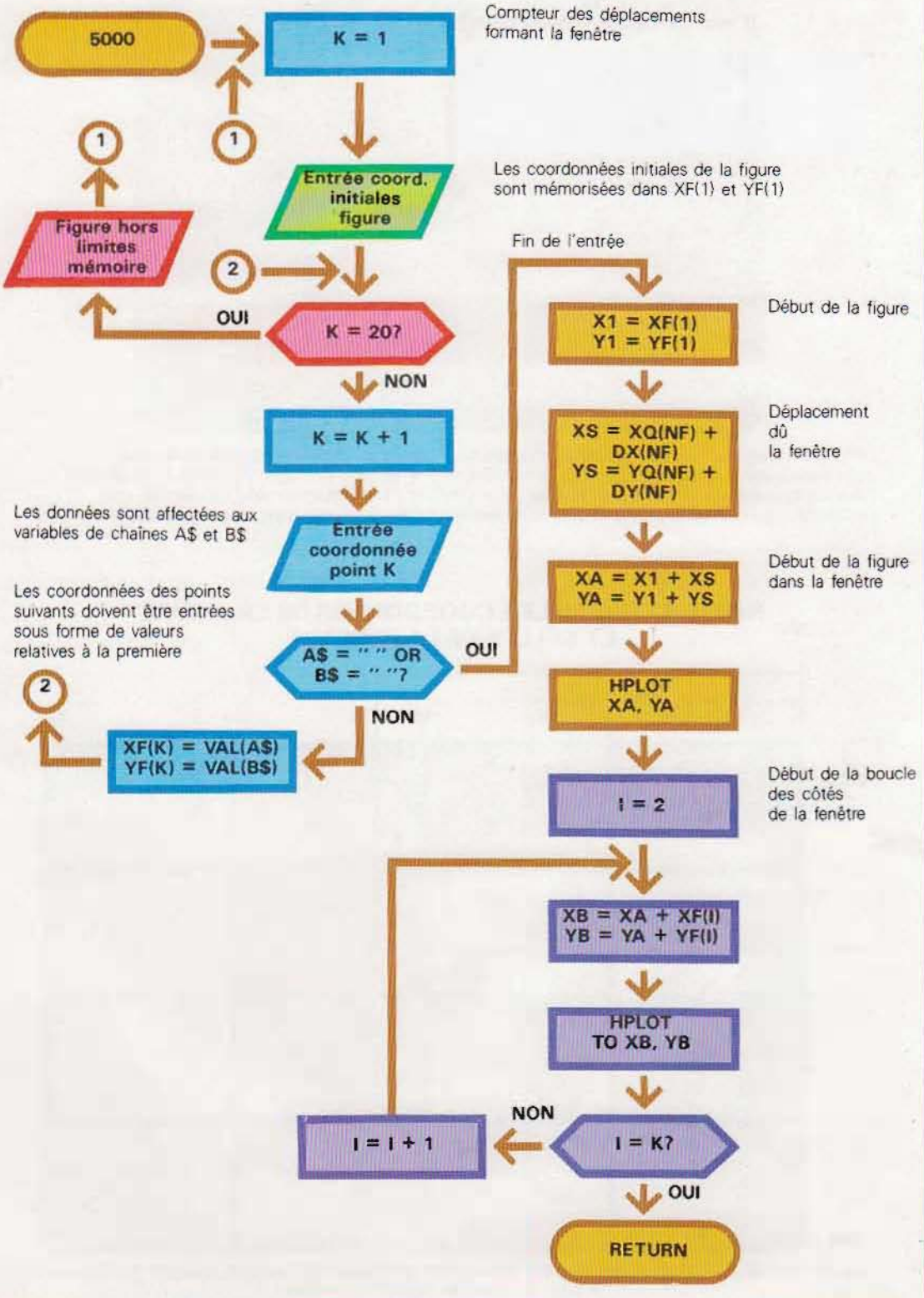
La fenêtre en position 4 dans le quadrant 4 s'obtient tout d'abord par déplacement jusqu'au quadrant, c'est-à-dire jusqu'au point de coordonnées X = XQ(4) et Y = YQ(4) (trait rouge). Il faut ensuite se déplacer, à l'intérieur du quadrant, jusqu'aux coordonnées X1 = DX(4) et Y1 = DY(4) trait bleu). On identifie ainsi l'extrémité supérieure gauche de la fenêtre. Si les dimensions de l'écran changent, il faut également changer les DATA.

RAPPORT ENTRE LES COORDONNES DE LA FIGURE ET CELLES DE LA FENETRE



Les coordonnées réelles d'un point (XV, YV) s'obtiennent en additionnant celles du quadrant et de la position à celles du point par rapport à la fenêtre

SAISIE DES COORDONNEES ET PRESENTATION DE LA FIGURE



CREATION ET GESTION DES FENETRES

```
1 REM ==================
2 REM *   PROGRAMME    *
3 REM * DE SIMULATION  *
4 REM *  DE FENETRES   *
5 REM * EN PAGE  HGR2  *
6 REM ==================
7 :

8 :

10 HOME
100 GOSUB 10000
110 GOTO 500
115 :

120 REM ==================
130 REM * APPEL FENETRE *
140 REM ==================
145 :

147 TEXT
150 HOME:PRINT "Fenetres
    disponibles"
160 VTAB 3:PRINT I$

170 FOR J=1 TO KF
180 PRINT V$(J);:HTAB 11:PRINT
    Q(J);:HTAB 23:PRINT P(J);
    :HTAB 31:PRINT H(J)"*"L(J)

200 NEXT J
210 VTAB 22:INPUT "Introduisez
    le nom de la fenetre ":NM$

220 FOR J=1 TO KF
230 IF NM$=V$(J) THEN NF=J:J=KF
    :NEXT J:GOTO 250

240 NEXT J
245 GOTO 210
250 RETURN
499 :

500 REM ========================
501 REM *APPEL SOUS-PROGRAMMES*
502 REM ========================
503 :

505 TEXT
510 GOSUB 1000
520 GOSUB 120
530 GOSUB 5000
540 TEXT:HOME:VTAB 10:INPUT
    "Voulez-vous déplacer la
    figure (O/N) ":RS$

550 IF LEFT$(RS$,1)="O" THEN 640
560 VTAB 10:INPUT "Voulez-vous
    changer le dessin (O/N) "
    :RS$
570 IF LEFT$(RS$,1)="O" THEN 520
580 VTAB10:INPUT "Voulez-vous
    modifier les fenetres (O/N) "
    :RS$
590 IF LEFT$(RS$,1)="O" THEN 500

600 HOME:VTAB 22:END

640 GOSUB 120:GOSUB 7000

645 IF Q$=" " THEN 655
650 GOSUB 5125
655 GOTO 540
1000 :

1001 REM ========================
1002 REM *DEFINITION DES FENETRES*
1003 REM ========================
1004 :

1010 KF=0:HGR2

1020 GOSUB 1500
1030 GOSUB 2000
1040 IF KE<>0 THEN 6000
1050 GOSUB 2500
1060 IF KE<>0 THEN 6000
1070 GOSUB 3000
1080 IF KE<>0 THEN 6000
1090 GOSUB 3500
1100 IF KE<>0 THEN 6000
1110 GOSUB 4000
1120 IF KE<>0 THEN 6000
1130 GOSUB 4500
1140 TEXT
1150 HOME:VTAB 10:INPUT "Autre
     fenetre (O/N) ":RS$

1160 IF LEFT$(RS$,1)="O" THEN 1020
1170 RETURN
1499 :

1500 REM ========================
1501 REM * DESCRIPTION FENETRE *
1502 REM ========================
1503 :

1505 TEXT
1510 HOME:VTAB 10:INPUT "Fenetre "
     ;A$

1520 IF LEN(A$)>30 THEN 1510
1530 RETURN
2000 :

2001 REM ==================
2002 REM * CTRL POS. 1-2 *
2003 REM ==================
2004 :

2010 KR=0:KE=0:I=1

2020 IF I>10 THEN KE=1:KR=1
     :RETURN

2030 A1$=MID$(A$,I,1)
2040 IF A1$=" " THEN I=I+1
     :GOTO 2020
2050 KF=KF+1:V$(KF)=A1$

2060 I=I+1
2070 IF I>11 THEN KE=1:KR=1
     :RETURN
```

```
2080 A1$=MID$(A$,I,1)
2090 IF A1$=" " THEN I=I+1
     :GOTO 2070
2100 IF A1$="=" THEN RETURN
2110 KE=2:KR=1:RETURN

2499 :

2500 REM ================
2501 REM * CTRL WINDOW *
2502 REM ================
2503 :
2510 KR=0:KE=0:I=I+1

2520 IF I>12 THEN KE=1:KR=2
     :RETURN
2530 A1$=MID$(A$,I,1)
2540 IF A1$=" " THEN I=I+1
     :GOTO 2520
2550 A1$=MID$(A$,I,6)
2560 IF A1$="WINDOW" THEN
     I=I+5:RETURN
2570 KE=3:KR=2:RETURN

3000 :

3001 REM =============
3002 REM * CTRL "(" *
3003 REM =============
3004 :
3010 KE=0:KR=0:I=I+1

3020 IF I>18 THEN KE=1:KR=3
     :RETURN

3030 A1$=MID$(A$,I,1)
3040 IF A1$=" " THEN I=I+1
     :GOTO 3020
3050 IF A1$="(" THEN RETURN
3060 KE=2:KR=3:RETURN

3500 :

3501 REM =============
3502 REM * CTRL Q/P *
3503 REM =============
3504 :
3510 KR=0:KE=0:I=I+1

3520 A1$=MID$(A$,I,1)
3530 N=VAL(A1$)
3540 IF N<1 OR N>4 THEN KE=4
     :KR=4:RETURN
3550 Q(KF)=N:I=I+2

3560 A1$=MID$(A$,I,1)
     :N=VAL(A1$)
3570 IF N<1 OR N>4 THEN KE=4
     :KR=4:RETURN

3580 P(KF)=N:RETURN

4000 :

4001 REM ==================
4002 REM * CTRL PARAMET. *
4003 REM ==================
4004 :

4010 KE=0:KR=0:B$="":I=I+1

4020 I=I+1:A1$=MID$(A$,I,1)

4030 IF A1$="-" THEN 4070
4040 IF LEN(B$)=3 THEN KE=4
     :KR=5:RETURN

4050 B$=B$+A1$:GOTO 4020

4070 N=VAL(B$):IF N>300 OR
     N<10 THEN KE=4:KR=5
     .RETURN

4080 B$=""
4090 I=I+1:A1$=MID$(A$,I,1)

4100 IF A1$=")" THEN 4130
4110 IF LEN(B$)=3 THEN KE=4
     :KR=5:RETURN

4115 IF A1$="" THEN A1$=" "
4120 B$=B$+A1$:GOTO 4090

4130 N=VAL(B$):IF N>140 OR
     N<10 THEN KE=4:KR=5
     :RETURN

4140 H(KF)=N:L(KF)=N:RETURN

4500 :

4501 REM ==================
4502 REM * DESSIN FENETRE *
4503 REM ==================
4504 :
4505 POKE -16304,0:POKE -16297,0
     :POKE -16299,0

4507 HCOLOR=3
4510 N=Q(KF):M=P(KF)

4520 XS=XQ(N)+DX(M):YS=YQ(N)
     +DY(M)
4530 X1=XS+L(KF):Y2=YS

4535 IF X1>279 THEN X1=279
4537 IF Y2>191 THEN Y2=191
4540 HPLOT XS,YS TO X1,Y2
4550 Y2=YS+H(KF)
4555 IF Y2>191 THEN Y2=191
4560 HPLOT TO X1,Y2
4570 X1=XS
4575 IF X1>279 THEN X1=279
4577 HPLOT TO X1,Y2
4580 Y2=YS
4585 IF Y2>191 THEN Y2=191
4587 HPLOT TO X1,Y2
4600 GET Q$:RETURN

5000 :
```

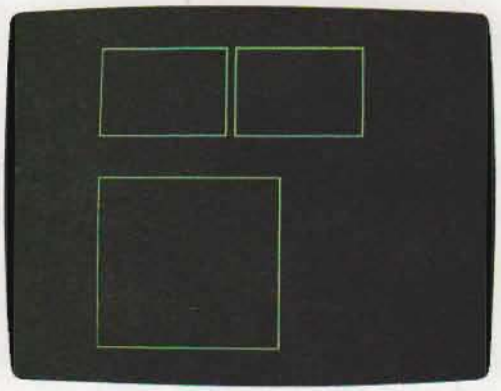
```
5001 REM ================
5002 REM * REPR. FIGURE *
5003 REM ================
5004 :

5005 TEXT
5010 K=1
5020 HOME:VTAB 10:PRINT
     "Introduire les coordonnées
     initiales"

5030 VTAB 10:HTAB 30:PRINT "X/Y"

5040 VTAB 11:HTAB 31:INPUT "":A$

5050 VTAB 11:HTAB 35:INPUT "":B$

5052 IF A$="" OR B$="" THEN 5020
5055 GOTO 5110
5060 IF K=20 THEN 5700
5070 K=K+1
5080 VTAB 10:HTAB 21:PRINT "Point
     "K

5090 VTAB 11:HTAB 31:INPUT "":A$

5100 VTAB 11:HTAB 35:INPUT "":B$

5110 IF A$="" OR B$="" THEN K=k-1
     :GOTO 5126

5120 XF(K)=VAL(A$):YF(K)=VAL(B$)
     :IF XF(K)>270 OR XF(K)<1
     THEN 5080

5121 IF YF(K)>191 OR YF<1 THEN
     5080
5122 GOTO 5060
5123 :

5124 REM >> DESSIN <<
5125 :

5126 HCOLOR=3
5127 POKE -16304,0:POKE -16297,0
     :POKE -16299,0

5130 X1=XF(1):Y1=YF(1)

5140 XS=XQ(Q(NF))+DX(P(NF)):YS
     =YQ(Q(NF))+DY(P(NF))
5145 CF=NF
5150 XA=X1+XS:YA=Y1+YS:I=2


5155 HPLOT XA,YA
5160 XB=XF(I)+XS:YB=YF(I)+YS

5170 REM
5175 IF XB>=XS+L(NF) THEN XB=XS
     +L(NF)-1
5176 IF YB>=YS+H(NF) THEN YB=YS
     +H(NF)-1
5180 HPLOT TO XB,YB
5190 IF I=K THEN GET Q$:RETURN

5200 I=I+1:GOTO 5160

5700 TEXT:HOME:VTAB 10


5710 PRINT "Max. 10 points":GET
     Q$
5720 GOTO 5010
6000 :

6001 REM ===============
6002 REM * DIAGNOSTIC *
6003 REM ===============
6004 :

6010 PRINT CHR$(7)
6020 VTAB 20:PRINT D$(KE):GET Q$

6030 VTAB 20:PRINT BK$

6040 KF=KF-1:GOTO 1020

6999 :

7000 REM ======================
7001 REM * EFFACEMENT FIGURE *
7002 REM ======================
7003 :

7010 NF=NF:NF=CF

7020 HCOLOR=0:GOSUB 5127

7030 NF=NF:RETURN

10000 :

10001 REM ===================
10002 REM * INITIALISATION *
10003 REM ===================
10004 :

10005 HIMEM 16383:HGR2:TEXT

10010 DIM D$(4),V$(16),Q(16)
      ,P(16),H(16),L(16),XF(20)
      ,YF(20)
10020 I$="Nom/Quadrant/Position
      /Dim."
10025 BL$="":FOR J=1 TO 30:BK$
      =BK$+" ":NEXT J

10030 FOR J=1 TO 4:READ D$(J)
      :NEXT J

10050 RETURN

10100 DATA 0,0,0,0,150,0,75,0,0
      ,74,0,3,7,150,74,75,37

10110 DATA "Données insuffisant
      es","Erreur de syntaxe"
      ,"Mot-clé erronné","Erreur
      de paramètres"
```

GENERATION ET GESTION DES FENETRES VIDEO

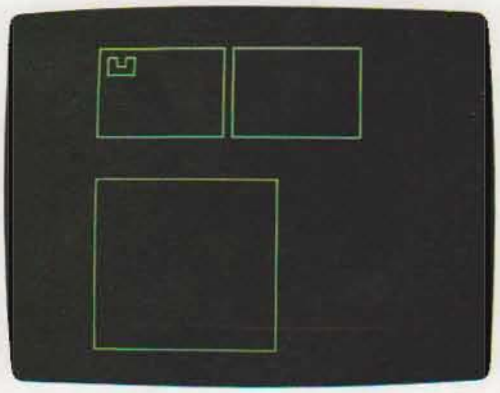
Voici quelques exemples de fonctionnement du programme présenté aux pages 1579 à 1581.
Sur l'écran ci-contre, on peut voir la phase de définition d'une fenêtre A, de 50 × 70 points écran, positionnée au point 1,1 (en haut à gauche). Le mot-clé WINDOW est suivi de ces paramètres dans un ordre inversé.

Après en avoir défini 2 autres, le programme liste les fenêtres.
L'activation d'une fenêtre se fait en introduisant le nom qui l'identifie.

A la demande de l'opérateur, 3 fenêtres définies au préalable sont affichées.



Une figure graphique a été dessinée. Elle peut être insérée dans n'importe laquelle des 3 fenêtre activées à l'aide de commandes de positionnement.



L'ordinateur et la voile

Qui ne connaît, ou tout du moins n'a entendu parler de la fameuse Coupe de l'America , ce trophée très convoité qu'a conquis, le 22 août 1851, la goélette américaine America lors d'une régate au large de l'île de Wight.

Ce fut la première victoire de la navigation à voile du Nouveau continent sur le Vieux. La coupe, qui s'appelait alors Coupe des Cent guinées, fut rebaptisées Coupe de l'America et, pendant 132 années consécutives, en dépit d'une suite de défis épiques qui ont renforcé sa renommée, elle est restée aux mains des Américains.

Avec l'édition de 1983, Coupe de l'America est devenue un symbole, bien plus important que le simple trophée. De l'avis de tous, cette année a représenté un ***tournant*** *dans l'histoire de cette compétition. Il y a d'abord eu la victoire d'Australia II sur Liberty, qui a quelque peu entamé l'invincibilité américaine.*

Il faut parler ensuite de l'intérêt de plus en plus marqué d'un plus grand public qui n'est pas forcément impliqué par la course.

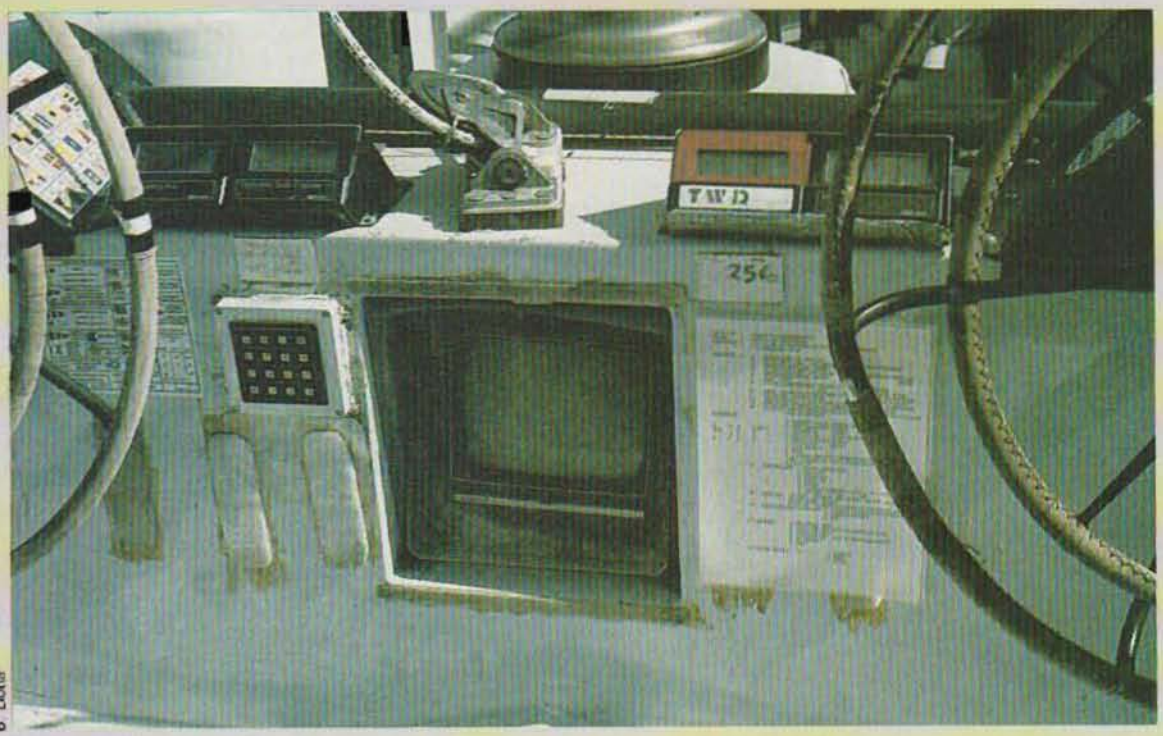
Autre facteur ayant fortement contribué à ce changement de vitesse de cette régate : ***l'utilisation croissante*** *de l'****électronique*** *et de l'****informatique.***

D'abord simples instruments de calcul ou outils d'aide à l'architecture et à la conception des voiliers, les ordinateurs sont « montés à bord » pour des essais en mer et en régate, afin de collecter et de traiter les données de bord. Nous assistons à un renouvellement complet de la compétition avec des voiliers de plus en plus sophistiqués.

Précisément, l'ordinateur a fait sa première apparition en 1963 sur la scène de la Coupe de l'America, avec l'embarquement d'un appareil capable de calculer l'angle réel du vent. En effet, les instruments d'alors ne fournissaient que des ***informations*** *— sens et vitesse du vent, notamment — sur le vent apparent (vent réel combiné au vent de vitesse engendré par le déplacement même de l'embarcation). Or, les données relatives au vent réel s'obtiennent en effectuant des* ***calculs vectoriels****. C'est pourquoi on est passé de l'emploi des ordinateurs comme simples instruments de calcul*

Départ de la Coupe de l'America.

B. Lotta



© Lotta

Le pont de Victory. Au premier plan, le "puits" du timonier : entre les timons, on aperçoit l'écran de l'ordinateur de bord, le clavier de commande et les viseurs à cristaux liquides permettant de visualiser les données.

pour résoudre des problèmes de navigation, à leur utilisation au cœur d'un système complexe de collecte, de traitement et de contrôle des données afin de déterminer les différents paramètres de performance.

Le développement technologique considérable réalisé dans le domaine électronique — et notamment en matière de ***miniaturisation,*** *de* ***puissance*** *de calcul et de* ***coûts*** *des ordinateurs — a été pour beaucoup dans l'accélération du processus d'intégration de l'ordinateur comme instrument indispensable à bord des 12 mètres.*

L'édition 1980 de la Coupe de l'America, à Newport, a marqué le début de l'emploi de l'ordinateur, la nouveauté n'étant plus la présence de celui-ci à bord, mais le support externe constitué par les ***puissants systèmes de calcul installés à terre et reliés par radio.*** *Pendant les entraînements et les régates, toutes les données intéressantes de l'embarcation — la vitesse et la direction du voilier, la vitesse et le sens du vent, etc. — relevées par des capteurs de bord étaient d'abord affichées sur les viseurs des instruments de bord pour une exploitation immédiate par l'équipage, ensuite recueillies et traitées par le calculateur de bord, et, enfin, intégralement enregistrées sur disque ou bande magnétique. A la fin de la journée, les données étaient transmises à terre et entrées dans l'ordinateur central, afin d'être analysées pour qu'on puisse en tirer le plus grand nombre d'informations possible sur les performances de l'embarcation.*

Toujours à l'occasion de l'édition 1980, une autre nouveauté est apparue : l'introduction de la ***télématique****, qui permet de transmettre directement les données du voilier au centre de traitement à terre. L'Indépendance Syndicate (un « syndicate » est l'équivalent d'un consortium) du New York Yacht Club, propriétaire du Defender Clipper (defender, signifiant défenseur, est le nom que l'on donne — ou plutôt que l'on donnait — aux 12 mètres américains, qui devaient défendre la coupe détenue), avait adopté cette nouvelle méthode afin de rendre le traitement des données plus* ***rapide*** *et plus* ***efficace.***

Il n'y avait donc plus à attendre le retour à la base de l'embarcation pour disposer des données recueillies, celles-ci étant ***directement traitées à bord.***

Les données sont transmises en permanence par radio, de l'ordinateur de bord à celui du centre terrestre. Leur traitement est, quasiment, instantané. Au retour de l'embarcation, les données, ainsi analysées par le calculateur, sont immédiatement disponibles.

Cette innovation s'est avérée une aide précieuse, notamment lors de la mise en service des nouveaux bateaux. En effet, l'équipage doit apprendre à connaître les performances du voilier en mer : l'**ordinateur** constitue un auxiliaire de grande valeur et un moyen d'**accélérer** au maximum la **phase d'apprentissage**.

L'épreuve de 1983 a confirmé la tendance au recours de plus en plus fréquent à l'ordinateur de bord et des systèmes de traitement de soutien à terre, non seulement pour les defenders américains, qui sont, historiquement et techniquement, les mieux lotis en la matière, mais aussi pour presque tous les 12 mètres engagés. Il faut d'ailleurs dire que l'un des systèmes les plus sophistiqués jamais utilisés au cours de cette course était installé à bord de l'embarcation anglaise Victory 83.

A travers cet exemple, nous examinerons en détail l'emploi d'un ordinateur à bord d'un 12 mètres. Ce système se décompose en plusieurs éléments :

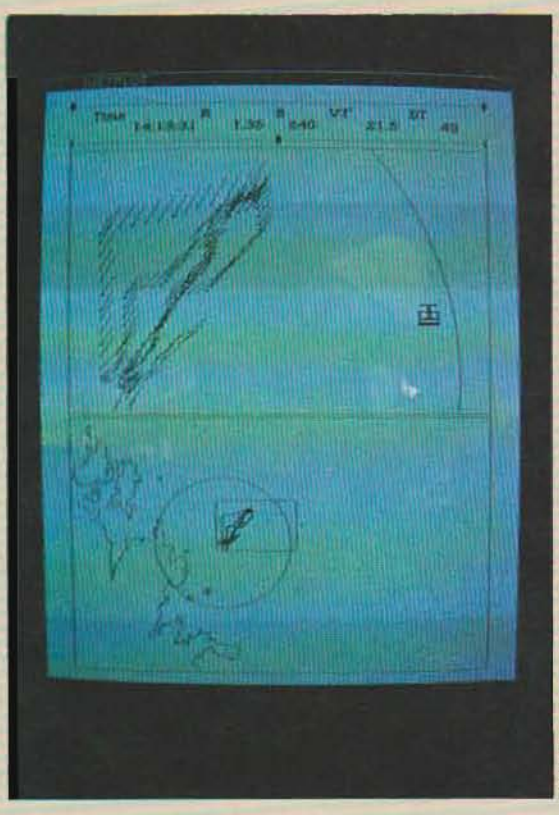
■ ***Le système de relevé, de préparation et de visualisation des données*** (transducteurs et capteurs) qui mesure les données liées au pilotage de l'embarcation (par exemple, vitesse et direction du vent, vitesse et route de l'embarcation, angle du timon, position, etc.).

Il faut y ajouter des unités d'interface ainsi qu'un microprocesseur dont la tâche est de préparer les données brutes et d'effectuer quelques calculs simples. Les résultats sont affichés sur des viseurs à cristaux liquides. Ils sont donc immédiatement exploitables par l'équipage, mais, de toute façon, transmis à l'unité centrale pour traitement.

■ ***Le système d'acquisition et de traitement des données*** comporte un ordinateur de bord qui effectue les traitements parti-

Le système d'acquisition, de traitement et de mémorisation des données à bord de l'embarcation auxiliaire de Victory.

B. Lotz



B. Liotta

Présentation, à l'écran, de la zone de régate et reconstitution de la route suivie par Victory. En haut de l'écran, la zone est agrandie.

culiers, comme la corrélation des données, la production de nouveaux résultats à partir de données transmises par les capteurs, la visualisation et la comparaison des données et des résultats avec ceux qui ont été mémorisés au cours des essais ou des précédentes régates.

Jusque-là, pour des raisons d'ordre plus historique que technique, la puissance du système de bord était limitée. On a préféré confier le travail d'analyse statistique, de production de fonctions, de mémorisation... à un autre système, plus important, installé à terre ou, comme dans le cas du Victory, à bord d'une vedette chargée d'assister le 12 mètres jusqu'aux minutes précédant le début de la régate. Les données ne sont donc pas confinées à bord de l'embarcation engagée, mais sont ***transmises par radio*** *au système principal. Ce type de transmission est désormais admis dans le monde des 12 mètres.*

Outre l'ordinateur, le noyau du système intègre une unité de disque, un écran vidéo, une imprimante graphique et un clavier. Le système possède une banque de données à la disposition de l'embarcation ; celle-ci constitue une aide réelle pour l'équipage qui peut, grâce aux instruments embarqués, contrôler efficacement les performances du voilier.

Les données prélevées par les capteurs de bord sont enregistrées ***en continu*** *et, le cas échéant, traitées* ***en temps réel.*** *L'opérateur se trouve ainsi en mesure de « piloter » les événements, par exemple de signaler par radio à l'équipage les éventuelles anomalies... et d'enregistrer les données s'il le juge utile. Ceci lors des essais seulement, car il n'est pas permis de communiquer avec les 12 mètres pendant la régate.*

La prochaine édition de la Coupe de l'America se déroulera à Perth, en Australie, en 1987. On peut déjà assurer, sans risque d'erreurs, que d'autres systèmes de bord se trouvent d'ores et déjà à l'étude.

Prenons le cas d'un pays européen. L'Italie, par exemple, a déjà lancé trois défis pour l'année 1987.

Le premier l'a été par le Yacht Club Costa Smeralda, soutenu par le Consorzio Azzurra, Sfida Italian America's Cup. Le second est celui du Yacht Club Italiano, avec le Consorzio Italia, qui a acheté Victory 83 pour s'en servir comme « lièvre » du nouveau 12 mètres qui sera conçu et construit par le consortium (l'embarcation-lièvre est utilisée pour l'entraînement d'une autre). Le règlement de la Coupe prévoit, en effet, que l'embarcation et son équipement doivent obligatoirement être conçus et réalisés par la nation engagée.

Le dernier défi est celui du Club Nautico Marina di Carrara, parrainé par le Consorzio Futura.

De nouvelles idées et de nouveaux projets entourés du plus grand secret seront probablement mis à jour. Il est néanmoins certain que la ***capacité des systèmes de bord*** *installés sur les 12 mètres sera* ***augmentée,*** *et que les études météorologiques de la zone de régate (analyse statistique des vents et d'autres données, à l'aide de satellites météorologiques du type Météostat) seront améliorées.*

Alors vivent le sport et la technologie ! Conjointement, car tous deux retirent d'appréciables avantages réciproques de leur étroite coopération.

Bruno Liotta

Instructions Basic de gestion des fenêtres vidéo

Le Basic standard, implémenté sur certaines machines, contient les instructions nécessaires à la gestion des fenêtres vidéo. On procède en deux étapes :

— génération de la fenêtre, en en définissant les dimensions et la position ;
— adressage de graphiques ou de texte par appel du nom symbolique associé à la fenêtre.

Une fenêtre est décrite à l'aide d'instructions proches de l'instruction WINDOW de l'exemple précédent. Toutefois, tous les sous-programmes nécessaires au décodage de l'instruction et à son exécution font partie intégrante du langage et sont totalement transparents à l'utilisateur.
Dans notre exemple, les sous-programmes de décodage sont écrits en Basic et génèrent des appels à d'autres sous-programmes, également en Basic. Ces derniers sont traduits par l'interpréteur avant d'être exécutés.
Quand les instructions nécessaires font partie du langage, le décodage fournit directement la version finale.
En revanche, si c'est l'utilisateur qui doit les implémenter, le temps d'exécution sera plus long (sauf si le programme est compilé).
Nous allons maintenant examiner la syntaxe des instructions de gestion des fenêtres dans l'Olivetti M20.
Dans cette machine, l'écran vidéo offre deux modes de visualisation : 512 × 256 pixels ou 480 × 256 pixels à sélectionner au moment de la configuration du système. Chacun des points de l'écran (pixel) est identifié, à l'aide du système de coordonnées de base dont l'origine se trouve en bas à gauche. Ce système de référence peut également être défini par l'utilisateur.
Dans ce dernier cas, il peut arriver que les coordonnées d'un point ne correspondent à aucun pixel (la position de chacun des pixels est définie au niveau du matériel). Le système active alors le pixel le plus proche des coordonnées indiquées.
Ce mode de fonctionnement n'oblige pas à recourir aux références des points écran, mais peut entraîner des imprécisions parfois importantes dans la présentation des images graphiques.
L'instruction qui définit une fenêtre à la forme :

A=WINDOW (q,p,v,h)

comprenant les éléments suivants :

A : C'est la variable (entière) associée à la fenêtre. Le système lui affecte une valeur croissante de 2 à 16, 16 étant le plus grand nombre de fenêtres susceptibles d'être ouvertes simultanément, tandis que 1 désigne l'intégralité de l'écran. La numérotation croissante commence à la première valeur disponible et progresse au fur et à mesure de la création de fenêtres mais en réutilisant les numéros laissés libres par d'éventuelles fermetures. Par exemple, si les fenêtres 2, 3, et 4 sont créées, la suivante sera la 5, à moins que l'une des précédentes ne soit effacée (fermée). La prochaine fenêtre prendra alors la valeur dégagée et non pas 5.

q : Cet élément indique la position occupée par la fenêtre en cours de création à l'intérieur de la fenêtre « mère ». Cette fonction se distingue de celle de l'exemple précédent, puisque la fenêtre ne pouvait être positionnée que par rapport à l'écran tout entier.
A l'inverse, avec cette machine, une fenêtre est considérée comme un écran (l'écran tout entier est la fenêtre 1). Les valeurs admises pour le paramètre sont les suivantes :

0 Positionnement de la nouvelle fenêtre dans la partie supérieure de la fenêtre mère
1 Positionnement dans la partie inférieure
2 Positionnement à gauche
3 Positionnement à droite

p : Il sert à noter la position de subdivision de la fenêtre mère (ou de l'écran entier) pour créer les nouvelles fenêtres.
La valeur à affecter au paramètre p dépend de celle de q. Si q vaut 0 ou 1 (fenêtre dans la partie supérieure ou inférieure de la fenêtre mère), la valeur de p variera de 1 à 255 pour créer une fenêtre ; elle indique donc le nombre de lignes de balayage de la

nouvelle fenêtre*. Un exemple de subdivision de l'écran est montré ci-dessous. Si le paramètre q (quadrant) vaut 2 ou 3, la division est verticale. Le paramètre p (position) peut alors varier entre 1 et la longueur de la fenêtre mère est décrémentée de 1.

v : C'est le paramètre optionnel, indiquant le nombre de lignes de balayage par ligne de texte. Il varie de 1 à 16. La valeur par défaut est celle de la fenêtre mère.

h : Il sert à indiquer la distance, en pixels, entre deux caractères de texte. Il peut prendre les valeurs 6 ou 8. Dans le premier cas, chaque ligne comporte un maximum de 80 caractères, contre 64 dans le second. Ce paramètre est optionnel. Sa valeur par défaut est celle de la fenêtre mère.

* L'expression « ligne de balayage » désigne une ligne de l'écran graphique. Elle ne doit pas être confondue avec les termes « rangée » ou « ligne », qui se réfèrent au mode texte. Dans ce cas, une ligne est séparée des autres de la hauteur d'un caractère au moins, et elle se compose de plusieurs lignes de balayage, généralement de 7 à 12.

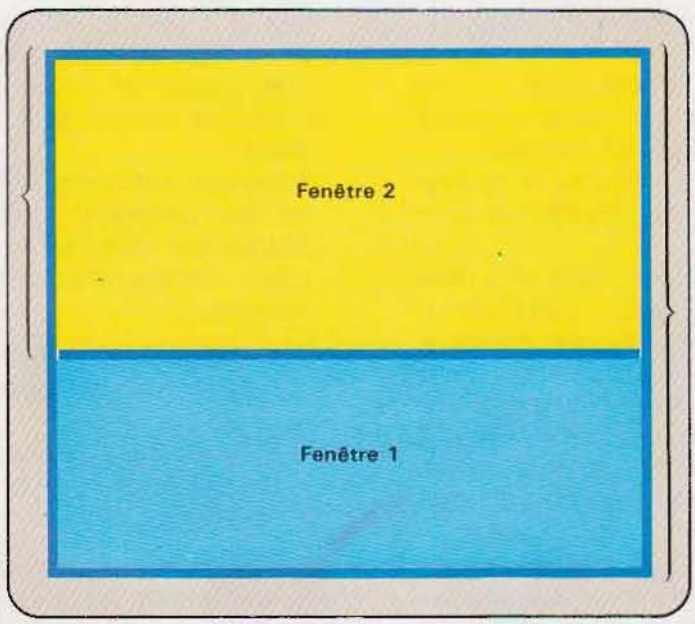
La page ci-contre montre quelques exemples de l'instruction WINDOW, en fonction des variations de v et de h. Les autres paramètres sont omis, ce qui signifie que le système prend en compte les valeurs de la fenêtre mère. Les paramètres v et h peuvent être modifiés à tout moment (même une fois la fenêtre ouverte) à l'aide de la même instruction WINDOW. Elle prend alors la forme :

A=WINDOW(0,0,v,h)

Dans ce cas, la variable A est fictive, car le système lui affecte automatiquement la valeur correspondant à la fenêtre sélectionnée.
La sélection d'une fenêtre reprend une fois encore l'instruction WINDOW, sous la forme :

WINDOW %N

EXEMPLE DE PARTAGE DE L'ECRAN EN FONCTION DES PARAMETRES q et p

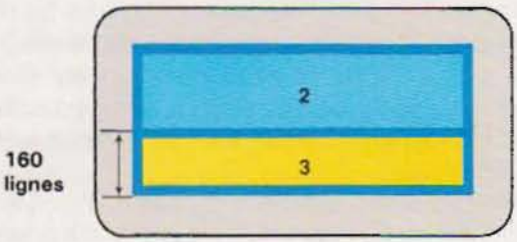


Quand une fenêtre (par ex. la n° 2) est ouverte selon les valeurs :

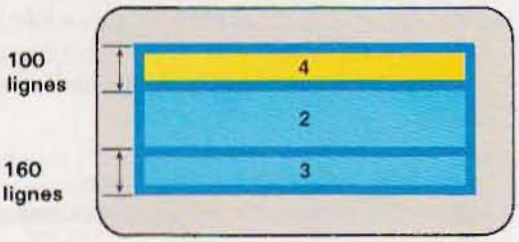
q = 0 la fenêtre est positionnée dans la partie haute de la fenêtre mère (ici l'écran dans son ensemble) et la division est horizontale

p= 180 la hauteur de la fenêtre, à partir du bord supérieur, est de 180 lignes

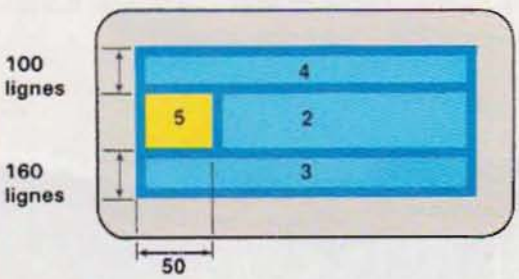
EXEMPLES D'APPLICATION DE L'INSTRUCTION WINDOW



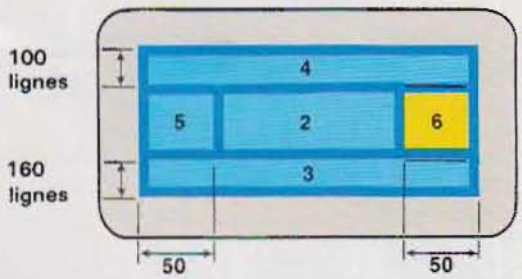
WINDOW (1,160). Séparation de la partie basse (q = 1), d'une hauteur de 160 lignes de balayage



WINDOW (0.100). Séparation de la partie haute, d'une hauteur de 100 lignes



WINDOW (2,50). Division dans la partie gauche. Dans ce cas, la valeur 50 du paramètre p indique la largeur de la fenêtre



WINDOW (3,50). Division analogue opérée dans la partie droite (q = 3) de la fenêtre mère

La fenêtre choisie correspond à la valeur numérique de la variable entière N.
Par exemple, les instructions :

```
N=2
WINDOW %N
```

provoquent la sélection de la fenêtre n° 2.

La valeur numérique peut également être fournie directement ; les instructions sont alors regroupées dans WINDOW %2.
WINDOW ferme une fenêtre devenue inutile, et rend cet espace disponible pour d'autres applications.

Sa forme devient alors :

CLOSE WINDOW %A

A indiquant la fenêtre à effacer. Si A est omis, toutes les fenêtres présentes à l'écran sont fermées.

Vecteurs graphiques et tables des figures

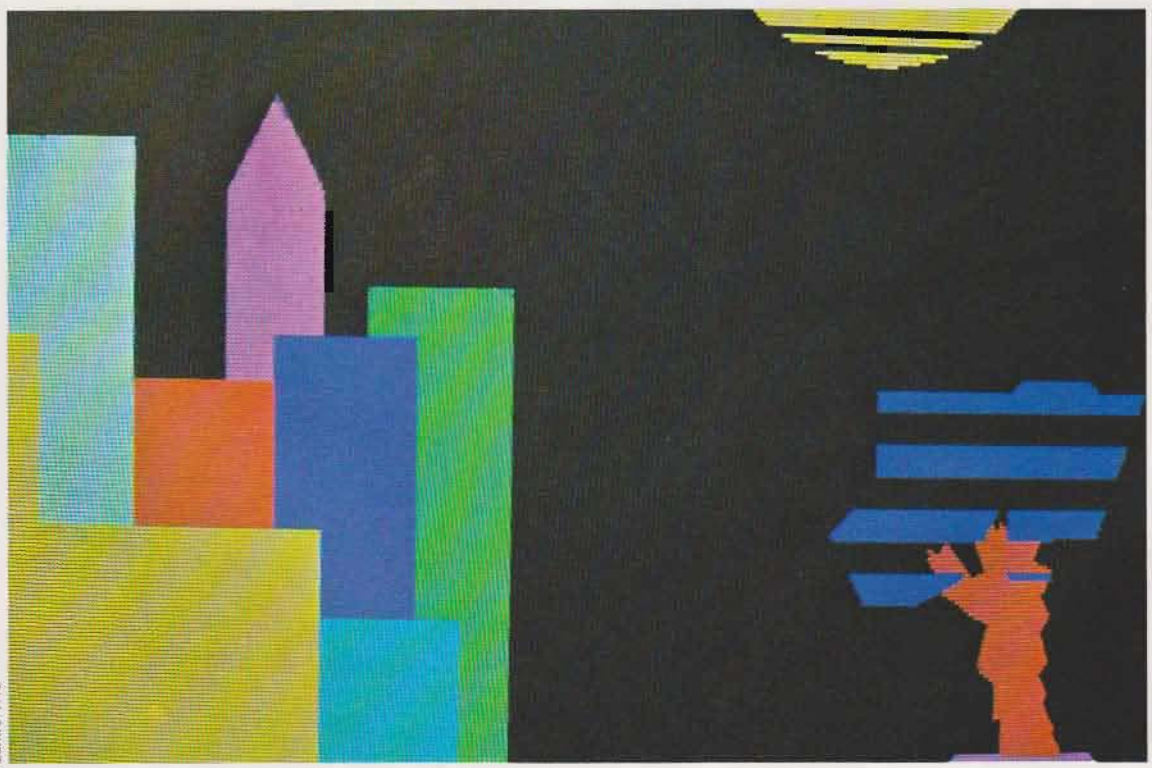
Tout dessin se décompose en une série de traits linéaires élémentaires. Cette technique permet, avec certains systèmes, de définir un ensemble de déplacements élémentaires du stylet constituant la figure. Ces ensembles sont généralement désignés par **tables des figures** à l'image des tables des déplacements servant à générer des chiffres ou des lettres.
Dans les systèmes qui y font appel, les tables sont gérées par le logiciel de base. Celui-ci contient des instructions de changement d'échelle ou de rotation.
Il est toutefois possible de créer un logiciel très simple, assurant les mêmes fonctions, avec des machines pour lesquelles ces instructions n'ont pas été initialement prévues.

Codage des vecteurs de déplacement

Pour produire une table des figures, il suffit de mémoriser, à l'échelle appropriée, tous les déplacements nécessaires au tracé du dessin. Il y a quatre types de déplacements (haut, bas, droite, gauche), les lignes inclinées ou courbes étant représentées à l'aide de lignes brisées dégradées.
Une figure de la table n'est pas mémorisée à ses dimensions réelles, mais selon un système qui assimile un déplacement à la longueur du côté le plus court.
Ainsi, pour mémoriser un rectangle dont la hauteur est double de sa base, il faudra utiliser la table suivante (dans notre machine, l'origine des axes se trouve en haut à gauche de l'écran) :

1 / Un déplacement horizontal (base)
2 / Deux déplacements verticaux (hauteur = 2 × base)
3 / Un déplacement horizontal (dans le sens opposé au précédent)
4 / Deux déplacements verticaux (dans le sens contraire des précédents)

Une ville "interprétée" par l'ordinateur en mode graphique.



Centro THC

Si les symboles suivants :

↑ = déplacement unitaire vertical négatif (haut)

↓ = déplacement vertical positif (bas)

→ = déplacement horizontal positif (droite)

← = déplacement horizontal négatif (gauche)

sont utilisés pour indiquer les déplacements, ce rectangle sera décrit par la suite de symboles :

→ ↑↑ ← ↓↓

Chaque déplacement est appelé **vecteur graphique ;** le rectangle précédent sera donc constitué de 6 vecteurs. Lors de la reconstitution du dessin, chacun d'eux indique la direction du déplacement à effectuer pour tracer un côté, tandis que le pas de déplacement est donné par le facteur d'échelle. Les changements d'échelle sont donc très simples à effectuer : il suffit de suivre le même tracé, mais en multipliant les quantités en fonction de la nouvelle échelle.

Si chaque figure est schématisée comme une combinaison des quatre déplacements élémentaires, le gain d'espace obtenu à la mémorisation est considérable. Les choix possibles à chaque pas sont effectivement limités à quatre (↑ , →, ← , ↓ ,) et sont représentés à l'aide de deux bits, par exemple comme il est indiqué dans le tableau ci-dessous :

Déplacement	Valeur décimale	Valeur binaire
↑	0	0 0
→	1	0 1
↓	2	1 0
←	3	1 1

Le codage indiqué est le plus courant dans les machines reprenant cette technique (Apple et compatibles). Dans tous les cas, les vecteurs peuvent toujours être représentés avec deux bits.

Le tableau 1 montre un codage possible des vecteurs graphiques mais n'indique pas leurs éventuels attributs. Il faut notamment que soit indiqué si le déplacement spécifié par le vecteur est seulement un positionnement (sans que rien n'apparaisse sur l'écran) ou la visualisation d'un segment. Dans le premier cas, le déplacement décrit par le vecteur doit être tracé à l'aide d'une couleur identique à celle du fond ; dans le second cas (visualisation), la couleur doit être visible. Cette information nécessite le recours à un 3e bit indiquant si le segment est visible ou non. Si on lui associe la valeur O pour indiquer qu'il s'agit d'un simple déplacement et la valeur 1 pour signifier que le trait est visible, on obtient les correspondances présentées dans le tableau 2 en bas de page. Comme on peut le voir, pour passer d'un déplacement non visible au même déplacement visible, il suffit d'ajouter 4. Par exemple, le déplacement horizontal positif (→) a le code décimal 1 s'il est invisible et 5 dans le cas contraire. D'après ce décodage, le rectangle considéré précédemment est représenté par les valeurs décimales 5 (→), 4 (↑), 4 (↑), 7 (←), 6 (↓), 6 (↓) et par les valeurs binaires correspondantes :

101 (5, →)
100 (4, ↑)
100 (4, ↑)
111 (7, ←)
110 (6, ↓)
110 (6, ↓)

La mémorisation de cette table nécessiterait

CODAGE DES VECTEURS DE DEPLACEMENT

Déplacement (vecteur)	Visible			Non visible			Valeur décimale visible	Valeur décimale non visible
	2	1	0	2	1	0		
↑	1	0	0	0	0	0	4	0
→	1	0	1	0	0	1	5	1
↓	1	1	0	0	1	0	6	2
←	1	1	1	0	1	1	7	3

6 positions de mémoire, soit une par déplacement. L'espace mémoire serait inutilement occupé car chaque position se compose de 8 bits, alors qu'un vecteur graphique n'en utilise que 3. Il est donc normal de rechercher une forme de compactage permettant de mémoriser plusieurs vecteurs à la même adresse.

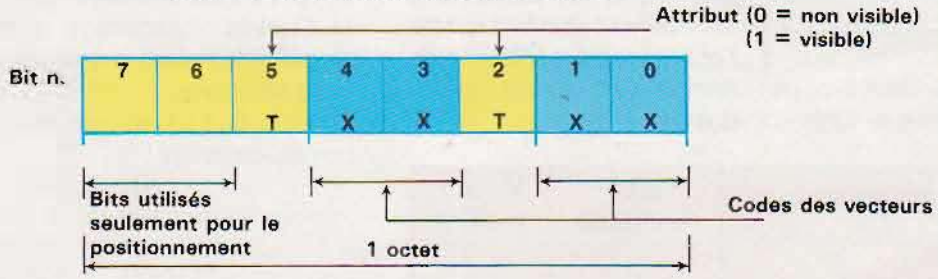
La plus usitée (figure 1 de cette page) consiste à occuper, avec deux vecteurs successifs, 6 des 8 bits d'une adresse mémoire ; les bits 3, 4 et 5 pour le premier et 0, 1 et 2 pour le second. Les bits 6 et 7 de l'adresse restent donc inactifs. Ils pourront, dans certains cas, servir à désigner un vecteur de déplacement sans affichage (il y manque le bit indiquant l'attribut). Leur emploi ne peut toutefois pas être généralisé.

La figure 2 de cette page illustre une autre forme de mémorisation d'un rectangle, exploitant le système de compactage précédemment décrit. Les bits 6 et 7 sont toujours à 0 tandis que le 5 et le 2 sont toujours à 1 (ils indiquent un déplacement visible). Pour signaler à la machine que la table est finie, une adresse tout entière est mise à 0 (position 4). Une fois la table de la figure mémorisée, il suffit, pour exécuter le tracé, d'appeler un sous-programme spécifique. Celui-ci analyse le contenu de chaque adresse et dessine un segment dans la direction indiquée par les bits 3, 4 et 0, et dans la couleur représentée par les bits 5 et 2.

La boucle est interrompue quand le contenu de la mémoire est à zéro (valeur choisie, conventionnellement, pour indiquer la fin de la table). On trouvera pages 1593 et 1595 les organigrammes d'un programme d'illustration et le listing correspondant (page 1596). Dans cet exemple, la table de la figure (un rectangle), introduite à l'aide d'une instruction DATA, est fixe, tandis que la phase de présentation (sous-programme 1000) peut être utilisée pour des emplois généraux.

Le programme n'est valable que du point de vue qualitatif. Il y manque la possibilité d'inclure plusieurs figures dans la même table, ensuite les sous-programmes de générations et de mémorisation des figures.

COMPACTAGE DES CODES CORRESPONDANT AUX VECTEURS GRAPHIQUES



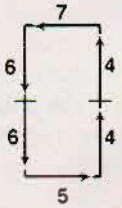
MEMORISATION D'UN RECTANGLE

Position mémoire	7	6	32 5 T	16 4	8 3	4 2 T	2 1	1 0	Valeur décimale
1	0	0	1	0	1	1	0	0	44
2	0	0	1	0	0	1	1	1	39
3	0	0	1	0	1	1	0	1	45
4	0	0	0	0	0	0	0	0	0

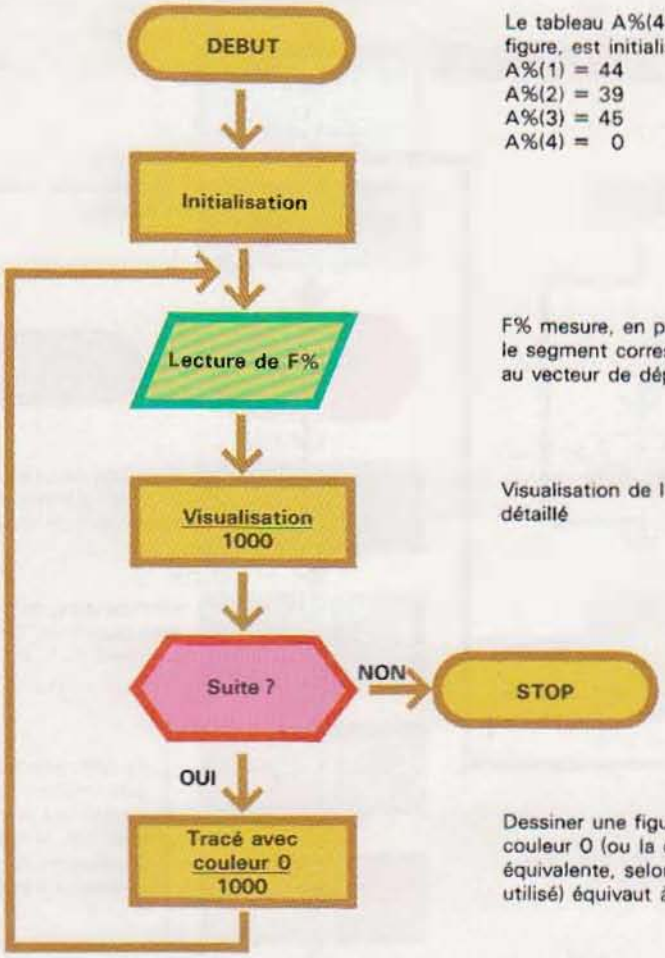
5(→) + 4(↑)

4(↑) + 7(←)

6(↓) + 6(↓)



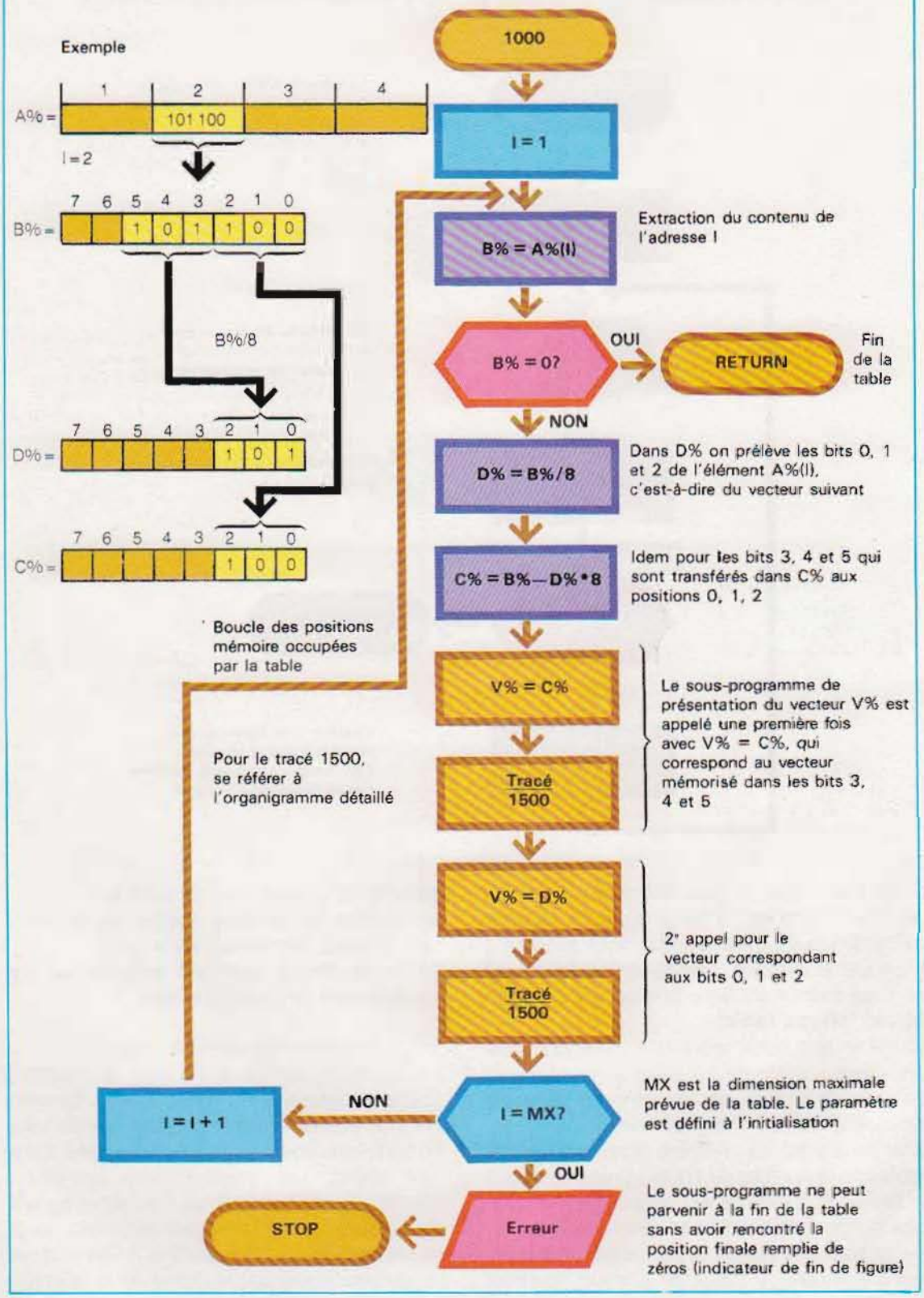
ORGANIGRAMME DE PRESENTATION D'UNE TABLE DE FIGURES



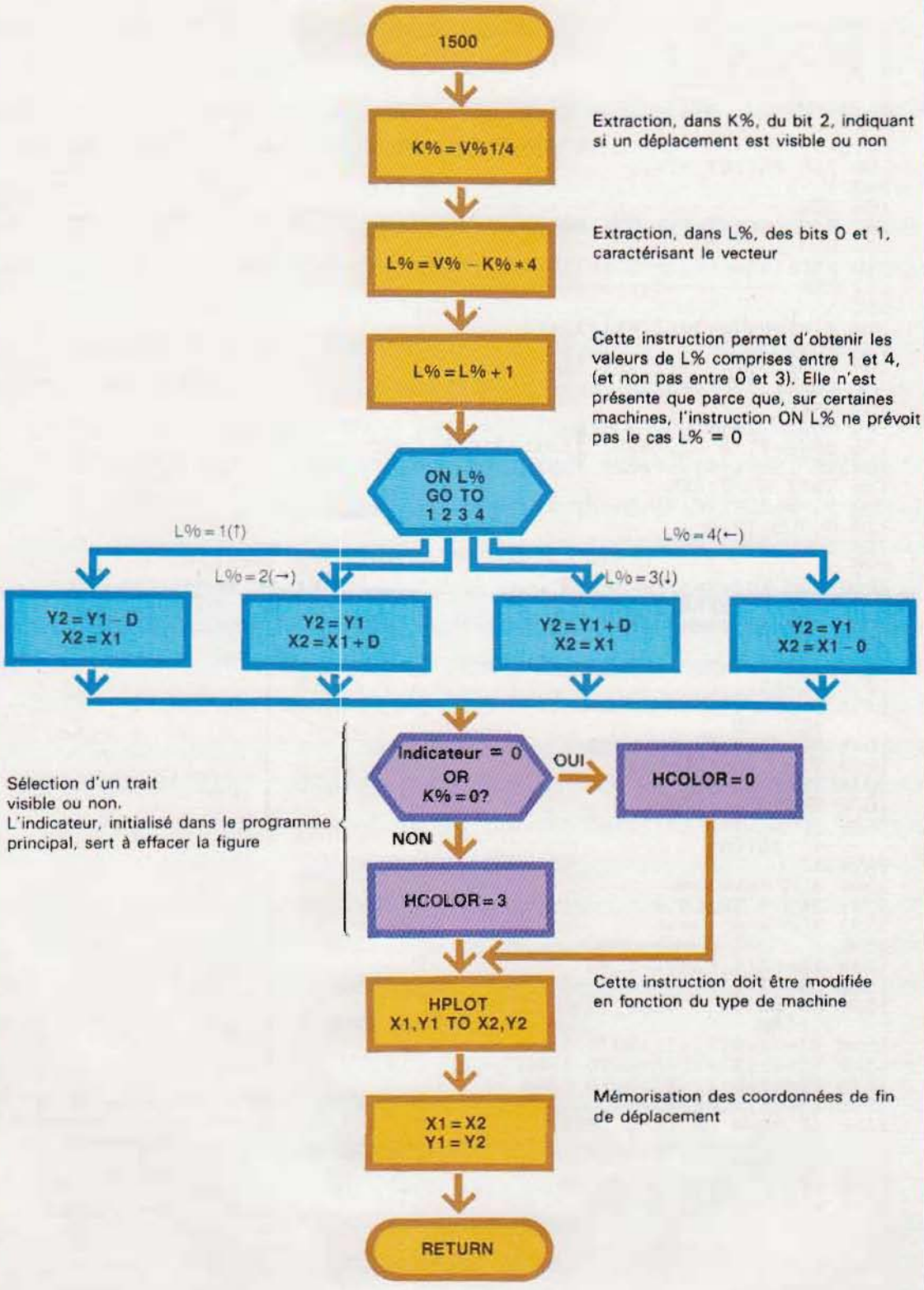
Dans la pratique, on peut difficilement réaliser un dessin complexe à partir d'une seule série de symboles. La meilleure approche consiste à découper le dessin en figures simples (**shapes**) et à les mémoriser dans une unique table de figures (**shape table**).
La table doit avoir alors une structure plus complexe.
La figure de la page 1597 montre une structure typique utilisée dans les machines dotées de ce type de logiciel. La première adresse (octet 0) renferme le nombre de figures présentes dans la table ; la deuxième est toujours nulle.
Les autres sont divisées en deux groupes :
— la zone des adresses, qui contient les pointeurs indiquant le début de chaque figure et constituant le répertoire de la table,
— la zone des vecteurs graphiques représentant chacune des figures de la table.
La fin de chaque figure est indiquée par une position dont le contenu est nul.

Voici comment fonctionne la table.
Le système examine le contenu de l'octet 0 (première adresse) et s'informe ainsi du nombre des figures présentes (3 dans l'exemple). Le deuxième octet étant toujours à zéro, il doit être sauté. Les adresses des différentes figures, au nombre de 3, se trouveront à partir de l'adresse 2, en avançant d'un pas de 2. Autrement dit, les adresses 2 et 3 contiendront le pointeur désignant le début de la première

SOUS-PROGRAMME DE PRESENTATION



SOUS-PROGRAMME DE TRACE

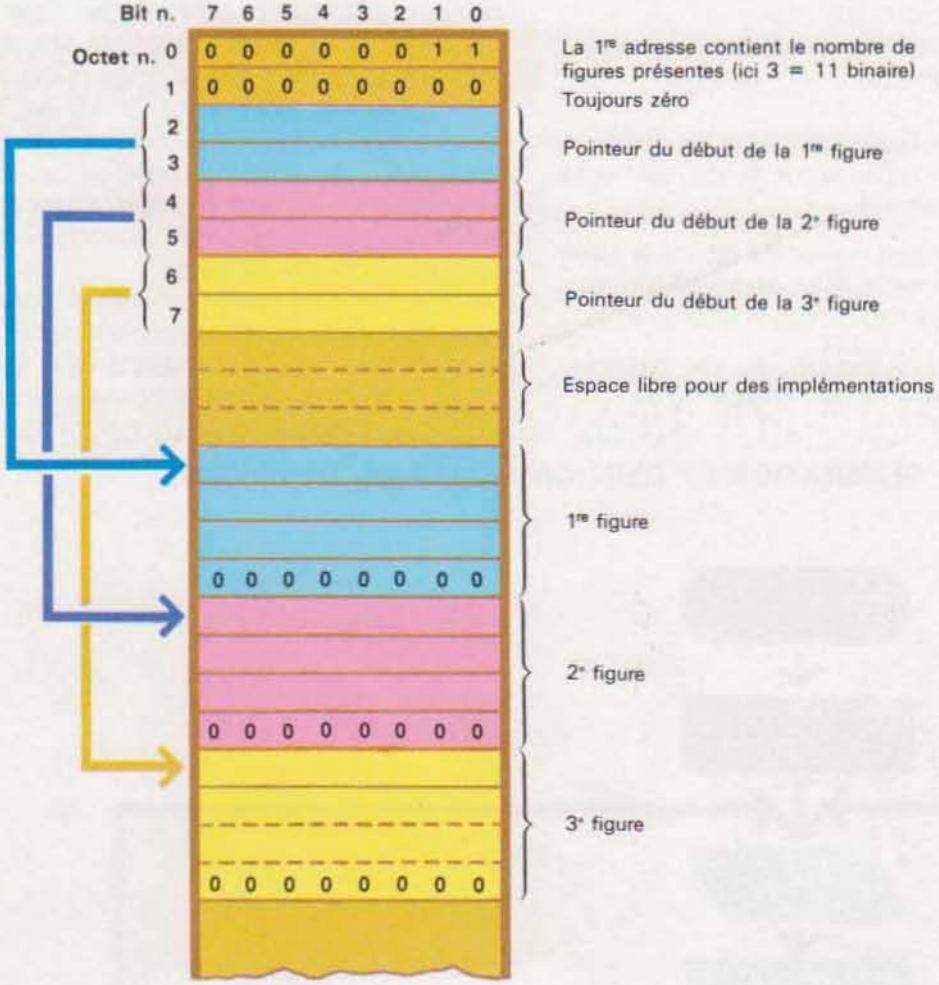


GESTION D'UNE TABLE DE FIGURES

Version Siprel, Apple et compatibles

```
1 REM ===========
2 REM *  TABLE  *
3 REM * FIGURES *
4 REM ===========
5 :
6 :
50 HGR
100 DIM A%(20):MX=20
103 :
106 REM ----------------------
107 REM CHARGEMENT DES VECTEURS
108 REM :
110 A%(1)=36:A%(2)=55:A%(3)=46
112 REM ------------------------
115 :
120 X1=100:Y1=50:X2=X1:Y2=Y1
125 FL=1
150 HOME:VTAB 22:INPUT "Longueur
    segment? ";F%
155 D=F%
160 GOSUB 1000
170 HOME:VTAB 22:INPUT "Suite? ";A$
180 IF LEFT$(A$,1)="O" THEN 200
190 TEXT:HOME:END
200 FL=0:XI=100:Y1=50:X2=X1:Y2=Y1
210 GOSUB 1000
220 GOTO 125
999 :
1000 REM ================
1001 REM * PRESENTATION *
1002 REM ================
1003 :
1010 FOR I=1 TO MX
1020 B%=A%(I)
1030 IF B%=0 THEN I=MX:NEXT I
     :RETURN
1040 D%=B%/8:C%=B%-D%*8
1050 V%=C%:GOSUB 1500
1060 V%=D%:GOSUB 1500
1070 NEXT I
1080 HOME:VTAB 22:PRINT "Erreur
     !":RETURN
1499 :
1500 REM =========
1501 REM * TRACE *
1502 REM =========
1503 :
1510 F%=V%/4:L%=V%-K%*4
1520 L%=L%+1
1530 ON L% GOTO 1550,1560,1570
     ,1580
1550 Y2=Y1-D:X2=X1:GOTO 1600
1560 Y2=Y1:X2=X1+D:GOTO 1600
1570 Y2=Y1+D:X2=X1:GOTO 1600
1580 Y2=Y1:X2=X1-D
1600 IF FL=0 OR K%=0 THEN
     HCOLOR=0:GOTO 1620
1610 HCOLOR=3
1620 HPLOT X1,Y1 TO X2,Y2
1630 X1=X2:Y1=Y2:RETURN
```

STRUCTURE D'UNE TABLE DE FIGURES



figure, les 4 et 5 le pointeur de la deuxième, et les 6 et 7 celui de la troisième. Une adresse est généralement laissée à zéro entre les figures et les tables pour permettre l'introduction de nouveaux pointeurs si les dimensions et la complexité du dessin augmentaient.

Programme de génération et de gestion de tables des figures

Il y a deux méthodes de création d'une table de figures. La première (et la plus simple) prévoit un dialogue avec l'utilisateur qui indique directement les déplacements à effectuer. La seconde (plus complexe du point de vue de la programmation, mais beaucoup plus utile) associe 4 touches aux 4 déplacements principaux ; de plus, chacun des déplacements est visualisé et mémorisé au fur et à mesure qu'il est activé.

Du point de vue de la production de la table, le résultat est le même, mais l'utilisateur voit progressivement le dessin se construire.

Examinons un programme de génération et d'utilisation de tables de figures utilisables dans les systèmes dépourvus de ces fonctions.

Avec les modifications adéquates, ce même programme peut servir dans les systèmes prévoyant la gestion des tables de déplacements. Dans ce type de machines, les fonctions accessoires, comme le changement d'échelle et les translations de la figure, seront obtenues à l'aide d'instructions spéciales.

Les principales fonctions exécutées par ce programme concernent :

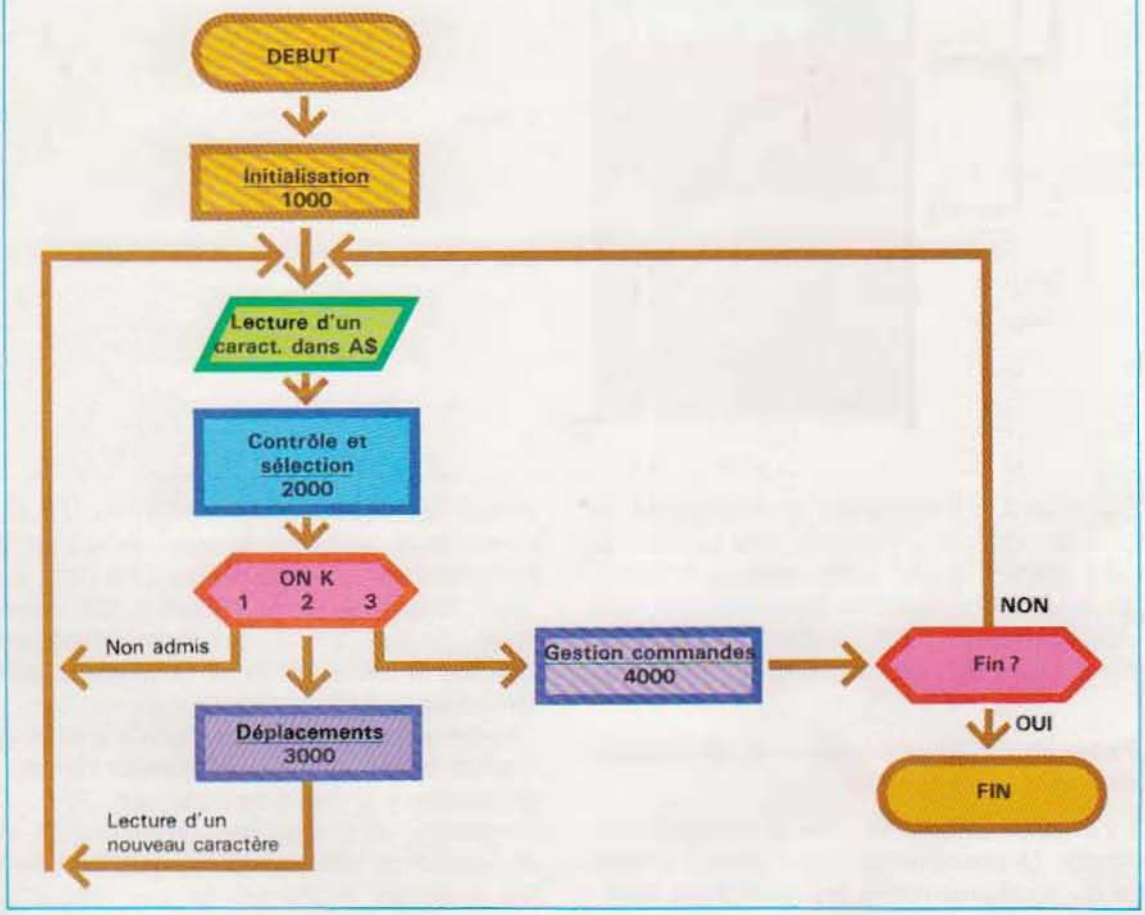
- La saisie des figures constituant le dessin, la mémorisation sous forme de déplacements et la visualisation.
- L'enregistrement sur disquette de la table des déplacements, avec possibilité de relecture pour des modifications ultérieures.
- La possibilité de changer l'échelle, de déplacer ou de faire tourner chaque figure.

L'organigramme du programme principal (ci-dessous), dont les variables sont énumérées dans le tableau de la page ci-contre, comporte quatre sous-programmes essentiels ou blocs fonctionnels.

Le premier (1000) initialise les variables, les instructions DATA et les indicateurs. Vient ensuite la lecture d'un caractère (A$), effectuée dans le programme principal, et qui doit utiliser l'instruction INPUT$(1) (sous CP/M) ou GET (ou un équivalent). Ces instructions, n'ayant pas d'écho, permettent d'entrer un caractère seul, sans avoir à fermer l'enregistrement comme avec l'instruction INPUT normale (il faut actionner la touche RETURN).

Le caractère ainsi entré est analysé dans le sous-programme 2000, qui répond par 3 valeurs contenues respectivement dans K, TS et TP. La valeur de K indique la touche enfoncée :

GENERATION ET GESTION DE TABLES DE FIGURES



VARIABLES ET SYMBOLES EMPLOYES DANS LE PROGRAMME

A$	= Caractère lu. Indique un caractère ou une commande.
A1$	= Deuxième caractère d'exécution de la commande
NM$	= Nom du fichier mémorisé ou rappel de la figure
K	= Indicateur du type de commande activée (non valide, déplacement, ordre)
TS	= Type de déplacement demandé
TP	= Type de commande
S(10)	= Tableau contenant les codes ASCII des touches de déplacement
O(10)	= Tableau contenant les codes ASCII des touches de commande
X1, Y1	= Coordonnées avant le déplacement
X2, Y2	= Coordonnées après le déplacement
DX, DY	= Valeurs du déplacement unitaire (points écran)
HC	= Indicateur de déplacement visible ou non visible. Activé par TP=1 ou 2
B(100)	= Tableau des déplacements constitutifs de la figure
IN	= Indice de la dernière position occupée dans B(100)
X0, Y0	= Coordonnées de début du dessin
FL	= Indicateur de déplacement
FR	= Indicateur de rotation
FS	= Indicateur d'échelle

Touches de commande

S = Déplacement	D = Disque (sauvegarde)
R = Rotation	C = Effacement
L = Chargement	+ = Agrandissement
	— = Réduction

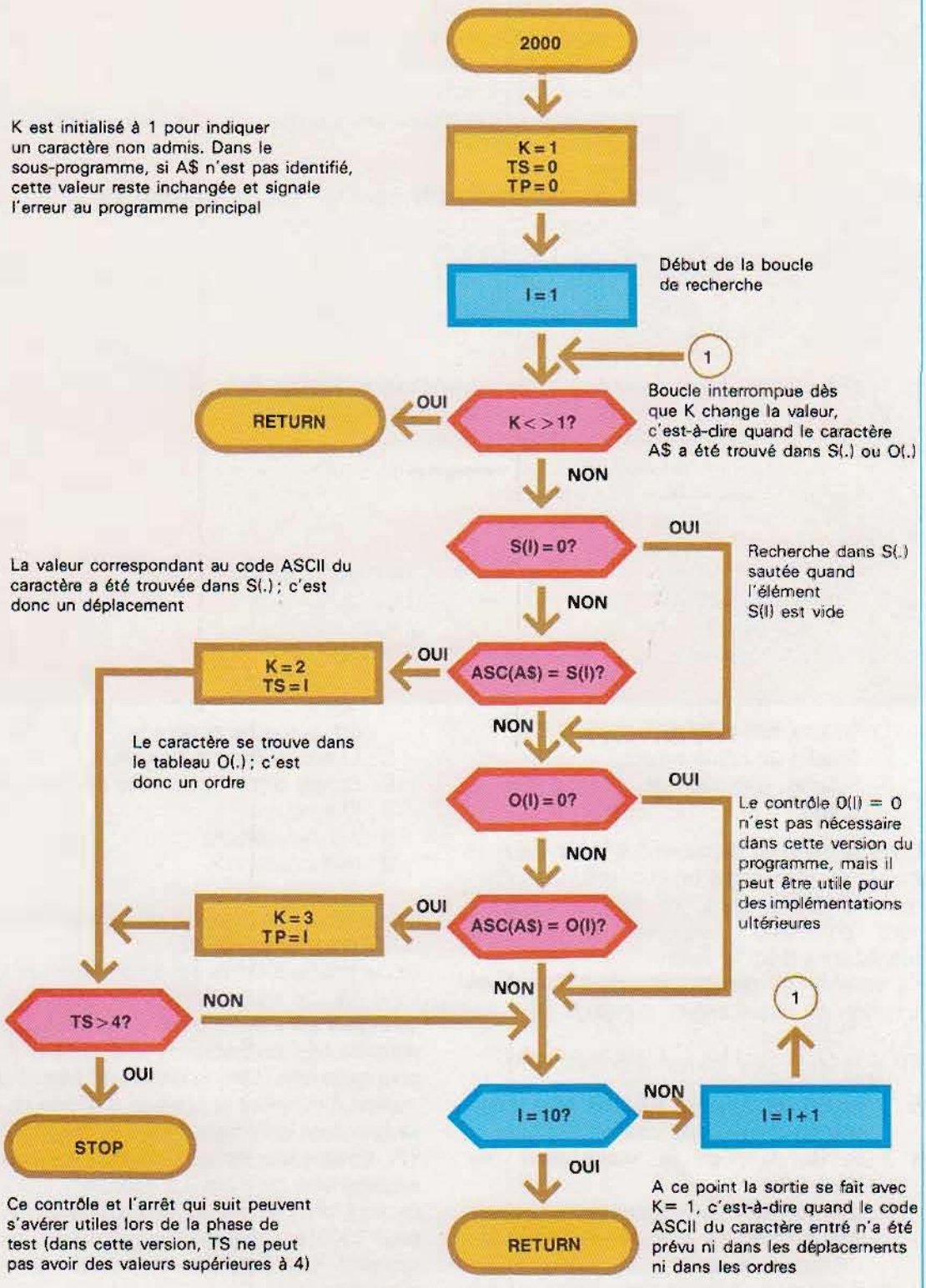
1 : Touche non prévue
2 : Touche de déplacement
3 : Touche d'entrée de commandes et ordres

Dans le cas d'un déplacement (pour tracer la figure) ou d'un ordre (mémorisation, changement d'échelle, etc.), les valeurs associées sont contenues dans les variables TS (déplacement) et TP (ordre).
La variable TS prendra la valeur 1 à 4 en fonction du déplacement demandé (↑, →, ↓, ←);
TP aura une valeur comprise entre 1 et 9 :

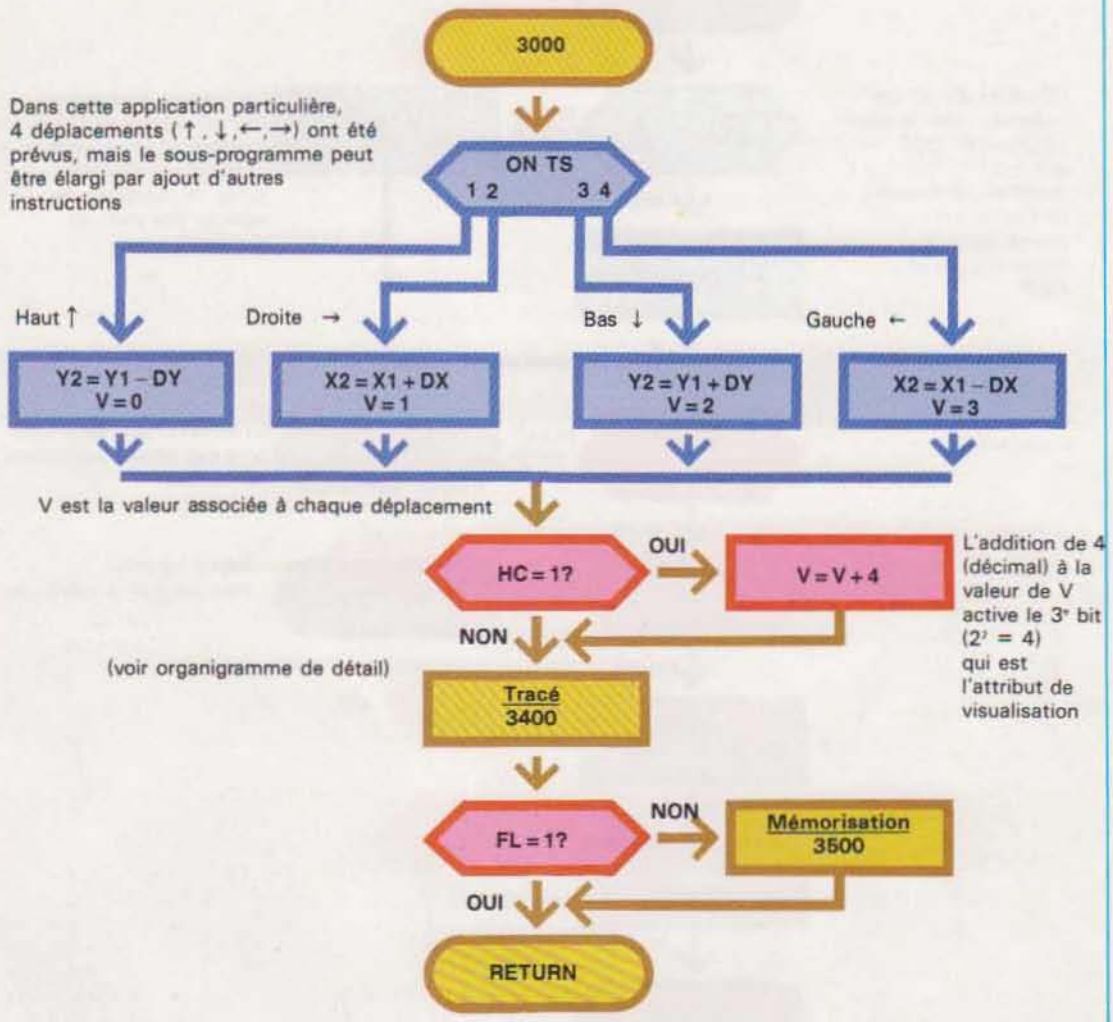
1 : Les déplacements suivant cette commande ne sont pas visibles
2 : Restauration de la visualisation des déplacements
3 : Déplacement de la figure dans la direction indiquée par la touche enfoncée immédiatement après
4 : Rotation de la figure dans le sens indiqué par la touche actionnée
5 : Mémorisation de la figure
6 : Rappel d'une figure mise en mémoire
7 : Effacement
8 : Agrandissement
9 : Réduction

L'organigramme de détail du sous-programme 2000 est en page 1600.
Le paramètre d'entrée est le caractère contenu dans A$. Le sous-programme vérifie que ce caractère est l'un de ceux qui ont été prévus pour les déplacements (dans le tableau S) ou pour les ordres (dans le tableau O). Dans l'affirmative, il transfère la position d'origine du caractère dans la variable correspondante (TS ou TP). Cette valeur indique également le type de déplacement ou d'ordre à exécuter.
Le sous-programme 3000 (organigramme en page 1601) est chargé de la gestion des déplacements. Les fonctions exécutées sont les suivantes :

SOUS-PROGRAMME DE CONTROLE DU CARACTERE ENTRE



ORGANIGRAMME DU SOUS-PROGRAMME DE GESTION DES DEPLACEMENTS



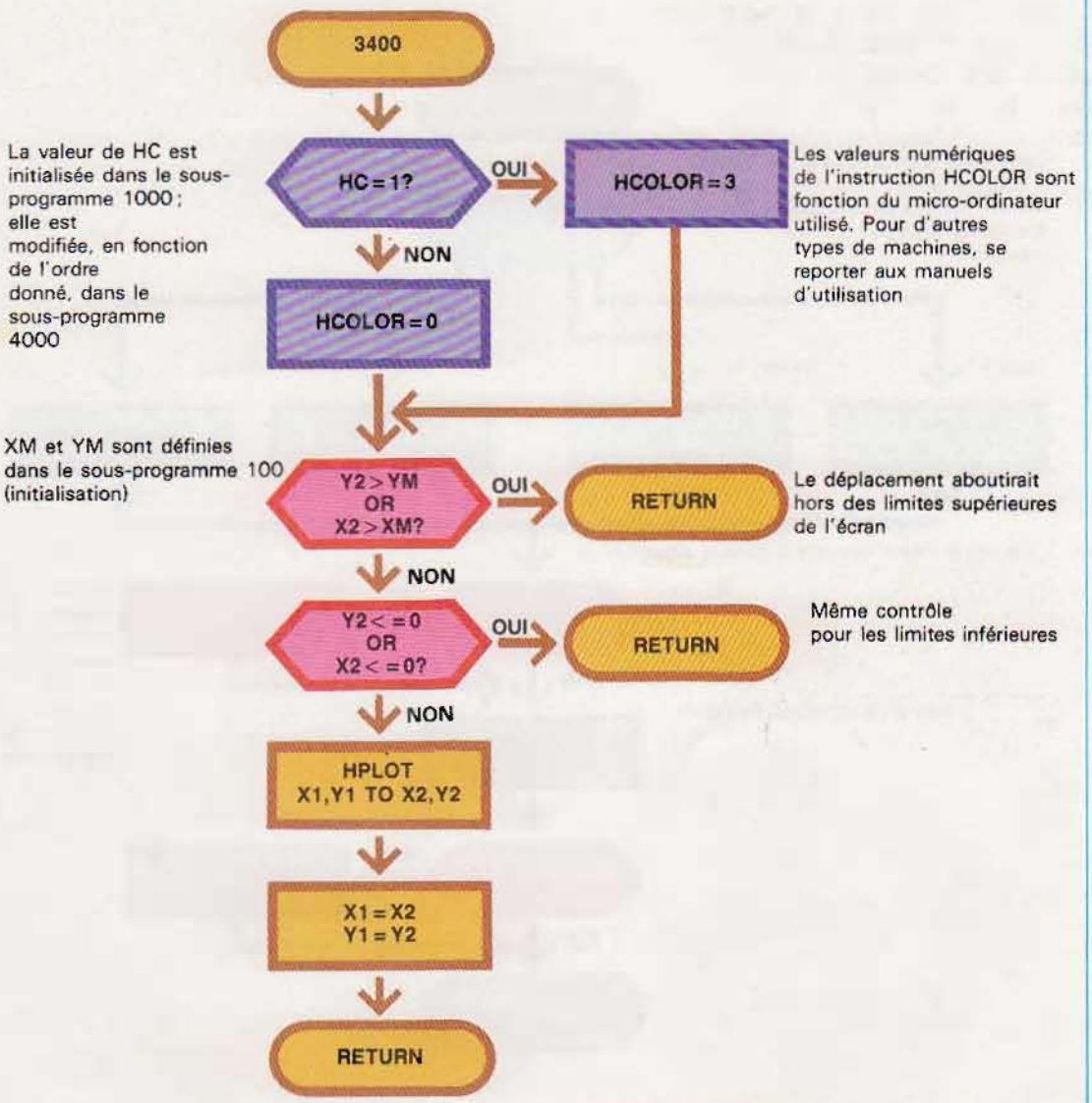
- Calcul des coordonnées après le déplacement et de la valeur V à mémoriser dans la table
- Transfert du déplacement dans la table (sous-programme 3500)
- Tracé du déplacement (sous-programme 3400).

Le calcul des coordonnées après le déplacement s'obtient en ajoutant à X1, Y1 les valeurs DX et DY. Les valeurs initiales de X1 et Y1 doivent être introduites dans le sous-programme 1000.

La valeur de V à mémoriser dans la table des déplacements n'est montrée que pour des raisons de clarté. On pourrait, en effet, l'obtenir simplement à l'aide de l'équation V = K—1, ou en faisant varier K de 0 à 3.

La nouvelle variable V ne serait alors plus nécessaire (K se trouve dans la plage de mémorisation et pourrait être utilisé directement). Mais des complications se produiraient au niveau du programme (par exemple, dans l'instruction ON K... certaines machines ne prévoient pas le cas K = 0).

SOUS-PROGRAMME DU TRACE D'UN DEPLACEMENT ELEMENTAIRE



La condition de déplacement visible ou non est testée à l'aide de l'indicateur HC, qui prend la valeur 1 quand le caractère correspondant à TP = 2 est entré, et la valeur O pour TP = 1. Pour mettre à 1 le 3e bit du déplacement (attribut), il suffit d'ajouter 4 à la valeur de V. Au cours de ce même sous-programme (3000) sont appelés les sous-programmes 3500, qui mémorise le déplacement (V) dans le tableau B (100) et, 3400, qui visualise le déplacement.
Le sous-programme 3400 contrôle également les valeurs de X2 et Y2. Au terme de cette opération, si les valeurs excèdent les limites de l'écran, le déplacement n'est pas visualisé mais seulement mémorisé (avec éventuellement une indication non portée dans l'organigramme). Cette méthode a été choisie parce que, comme nous allons le voir, ce programme permet de changer d'échelle. Si un déplacement sort des limites de l'écran, il suffit d'activer l'état commandes et de modifier l'échelle de représentation pour y remédier.
Le sous-programme 3500 est, quant à lui, plus complexe car il tient compte de « l'encombre-