LTC

n°8

LE JOURNAL DES AMATEURS DE PROGRAMMATION

ISSN 0761-9936

AVRIL 1985

AVEC BASIC

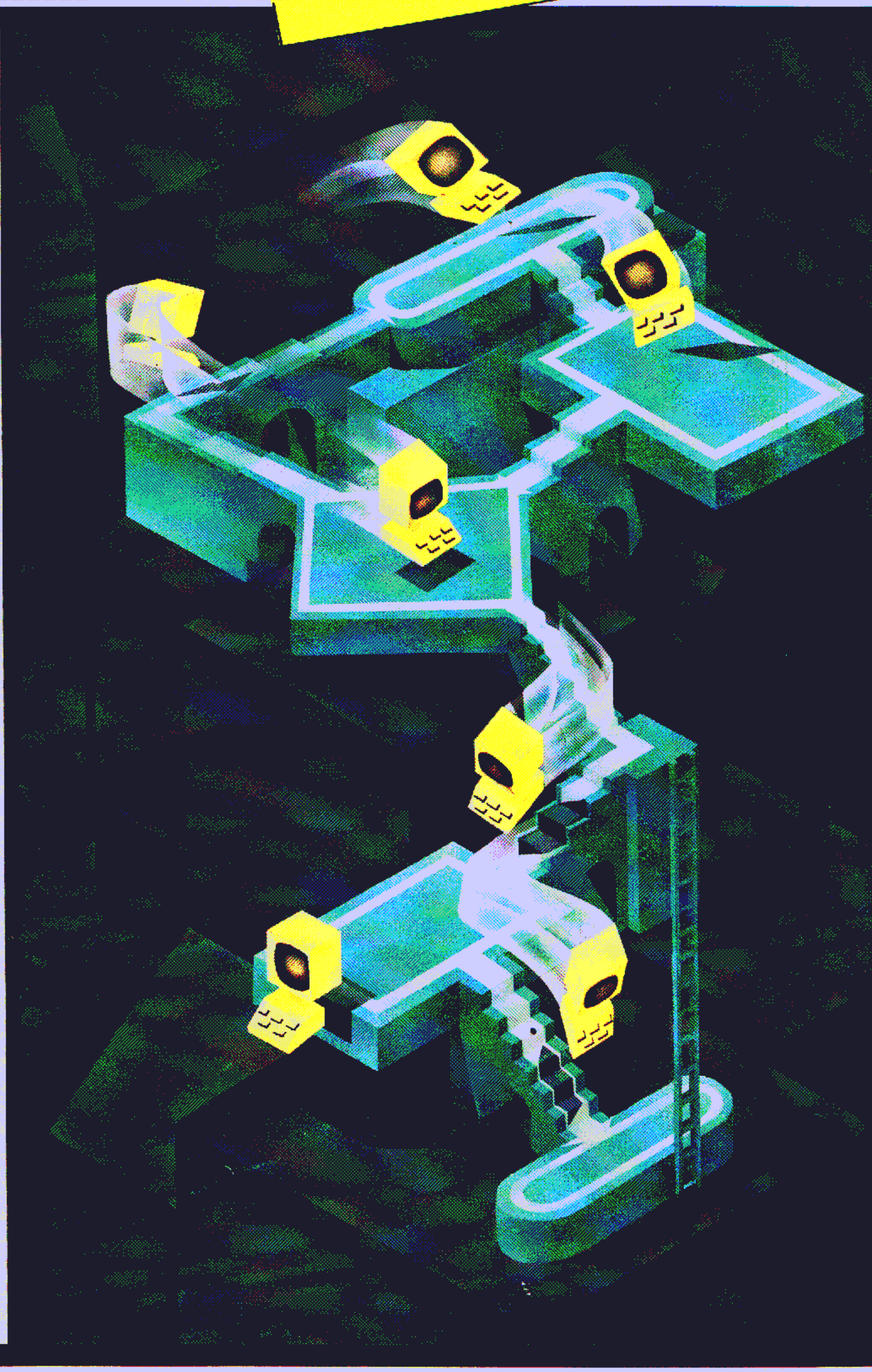
- L'avenir est structuré
- Atteindre la récursivité
- Programmez Picologo sur votre Amstrad

GROS PLAN SUR QUATRE LOGICIELS

- Découvrir Logo sur TO7 et TO7-70
- La troisième dimension sur Spectrum
- Hadès, Assembleur pour Oric-1 et Atmos
- Master 64, un vrai plus pour C.64

A L'ESSAI

- Le Basic du Guépard, du classique et du solide
- VG 5000 : pour les petits budgets



M 2712-8-20 F

Belgique : 154 FB - Canada : 2.95$C - Suisse : 6 FS

20 F

CASIO

INITIATION BASIC
PB 410

PB 410 : MICRO-ORDINATEUR DE POCHE

- Apprentissage d'une véritable PERSONNELLE au BASIC
- Programmation de tous équipements informatiques actuels du PB 410 (format, vitesse, mémoire, BASIC, clavier, écran, etc...)
- Sauvegarde des programmes SUR CASSETTE ORDINAIRE (une mémoire de 1K ou 2K ou 4K ou 8K ou 16K ou 32K supplémentaire)
- Exercices pratiques progressifs de programmation
- Explications complètes (jeux, calculs scientifiques, gestion de fichiers, etc...)
- Nombreux programmes d'application (gestion, jeux, etc...)
- Annexes : Glossaire, Table des codes ASCII, Messages d'erreur, Index
- Un ouvrage indispensable à tous les possesseurs d'un PB 410

EDITIONS SYBEX
6, rue de Mouzaïa
75019 PARIS
Tél. : 200.32.32

LIST
LE JOURNAL DES AMATEURS DE PROGRAMMATION

N° 8 - AVRIL 1985

SOMMAIRE

Ce numéro contient en encart des bulletins d'abonnement paginés 11, 12, 77 et 78.

Index des annonceurs p. 13

RÉDACTION-RÉALISATION
Directeur de la rédaction : Bernard Savonet
Rédacteur en chef : Jean Baptiste Comiti
Responsable de rubrique : Anne-Sophie Dreyfus
Conception graphique et secrétariat de rédaction : Eliane Gueylard
Assistante de rédaction : Maryse Gros
Administration : Marie-Hélène Muniz

Ont collaboré à ce numéro : *Gérald Anfossi, Antoine, Pierre Barnouin, François J. Bayard, Robin Bois, Christian Boyer, Daniel Briant, Michel Brochand, Laura Campagnet, Jean-Paul Carré, Thierry Chamoret, Frédéric Charles, Alain Chauvaux, Marcel Cottini, Raymonde Coudert, Robert Daguesse, Rémi Ducros, Christophe Fautres, Gérard Foucault, Florence Gautier-Louette, Francis Goudard, Max Hagenburger, Bernard Kokanosky, Jean-Christophe Krust, Xavier de La Tullaye, Jacques Labidurie, Jean-Pierre Lalevée, Patrick Leclercq, Eric Levenez, Thierry Lévy-Abégnoli, Alain Mariatte, Christophe Maurin, Pierrick Moigneau, Alain Normand, Edouard Rencker, Didier Renevot, Benoît Thonnart, Henri Volleau, André Warusfel.*

Illustrations : *Philippe Burel, Chimulus, Frapar, Bernard Helme, Renée Koch, Fabien Lacaf, Sylvain Lemaire, Philippe Mairesse, Alain Mangin, Alain Prigent, Nicolas Spinga.*

ÉDITION-PUBLICITÉ-PROMOTION
Éditeur : Jean-Pierre Nizard
Éditeur-adjoint : Jean-Daniel Belfond
Administration : Maryse Marti, assistée d'Anne Stolkowski
Publicité : Béatrice Ginoux Defermon
assistée de Nadine Schops

VENTES
Diffusion NMPP : Béatrice Ginoux Defermon
Abonnements : Muriel Watremez assistée de Denise Martinon, Cécilia Mollicone et Sylvie Trumel.

Directeur de la publication : Jean-Luc Verhoye

LIST est une publication du groupe tests

5 place du Colonel Fabien - 75491 Paris Cédex 10
Téléphone : (1) 240 22 01 - Télex : LORDI 215 105 F

La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'Art. 41, d'une part que « les copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective », et, d'autre part, que les analyses et les courtes citations dans un but d'exemples et d'illustrations, « toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants-droit ou ayant-cause est illicite » (alinéa 1er de l'Art. 40). Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contre-façon sanctionnée par les Art. 425 et suivants du Code Pénal.

BVP Bureau de Vérification de la Publicité.

Notre publication contrôle les publicités commerciales avant insertion pour qu'elles soient parfaitement loyales. Elle suit les recommandations du Bureau de Vérification de la Publicité. Si, malgré ces précautions, vous aviez une remarque à faire, vous nous rendriez service en écrivant au BVP, BP 4508, 75362 PARIS CEDEX 08.

Limité par Basic ?
Impatient, lorsqu'un compilateur classique prend 5 mn pour compiler 100 lignes ?
TURBO Pascal a été conçu pour vous.

TURBO PASCAL

625 F HT

avec manuel en français

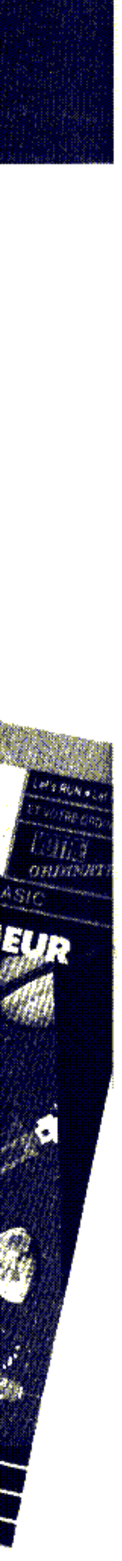
DE NOMBREUSES EXTENSIONS

vous permettent d'utiliser à fond votre ordinateur.

- *appels aux fonctions du DOS*
- *opérations sur la mémoire, les ports d'entrée/sortie*
- *fonctions AND OR, XOR, sur les entiers*

TURBO PASCAL EST UN SYSTÈME COMPLET

Il comprend un éditeur et un un compilateur dans le même programme. (28 K CPM. 36 K sous MS DOS).



Vous passez de l'un à à l'autre par une touche : plus besoin de jongler avec disquettes ou fichiers.

LA COMPILATION SE FAIT EN MÉMOIRE

Un compilateur classique utilise des fichiers intermédiaires sur disque; jusqu'à 90 % du temps peut être consacré aux opérations de lecture/ écriture sur disque.

Avec TURBO Pascal, la compilation se fait en mémoire en une seule passe : le temps de compilation est réduit au strict minimum.

Par exemple, Microcalc, programme de démonstration de 1.200 lignes fourni avec TURBO Pascal est compilé en 30 secondes (à 4 Mhz).

Si une erreur survient lors de la compilation, l'emplacement de l'erreur est retrouvé dans le code source et le mode éditeur activé : corriger un point-virgule oublié ne prend que quelques secondes.

L'ÉDITEUR INTÉGRÉ EST CONFIGURABLE.

Vous pouvez redéfinir toutes les commandes. Il lit les programmes écrits avec d'autres traitements de texte.

- *type String avec fonctions de traitement de chaînes*
- *procédures de gestion de l'écran*
- *8 fichiers prédéfinis*
- *modules de recouvrement (overlays) permettant d'écrire de très grands programmes*
- *fichiers à accès aléatoire avec "seek"*
- *constantes structurées permettant d'initialiser rapidement ensembles et tableaux*
- *identificateurs de 127 caractères*
- *programmes chaînés avec partage des données*
- *variables absolues placées à 1 adresse précise en mémoire.*

REVUES...

"Des performances à faire pâlir"
« LIST », Nov. 1984

"TURBO Pascal offre tout ce qu'un utilisateur de Pascal peut attendre en dépassant même largement ses espérances".
« ORDI », Nov. 1984

"The best cost less"
« Creative Computing », juillet 1984

"TURBO Pascal appears to violate the laws of thermodynamics".
« SOFTALK », Mars 1984

"This dynamic new language compiler is a "VolksPascal", with most of Pascal plus a few extras. It introduces a new programming environment and runs like magic".
« PC MAGAZINE », Nov. 1984

***TURBO 87** permet d'utiliser le coprocesseur 8087 sur les machines 16 bits.*

Remboursement possible par retour de la disquette non ouverte sous 15 jours.

ENVOYEZ-MOI DE SUITE :

- ☐ **TURBO PASCAL 625 F + 116,25 F TVA**
- ☐ **TURBO 87 1.150 F + 213,90 F TVA**
- ☐ **JE PRÉFÈRE LE MANUEL ANGLAIS**
- ☐ **ORDINATEUR :** .....

DISQUES ☐ **3 1/2"** ☐ **51/4** ☐ **8"**

SYSTÈME D'EXPLOITATION :
☐ **MS-DOS** ☐ **CMP-80**
☐ **PC-DOS** **CPM 86**

RÈGLEMENT :
☐ *CHÈQUE JOINT*
☐ *CONTRE-REMBT. (+ 50 F)*

- Remboursement garanti si renvoi de la pochette non ouverte sous 15 jours.
- Remplissez soigneusement le bon de commande.

NOM .....

ADRESSE .....

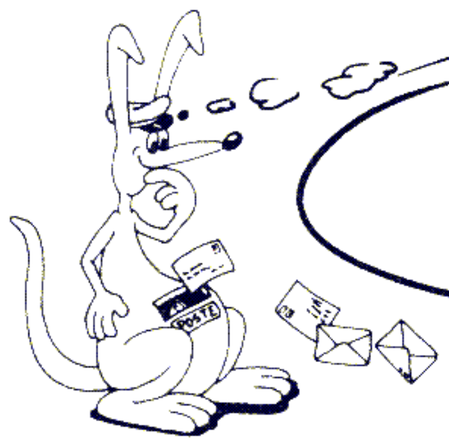
.....

TÉL. .....

SIGNATURE :

FRACIEL (47) 64.08.52
42, rue des Prébendes
37000 TOURS

A VOS CLAVIERS



Écrivez à LIST
5 place du Colonel Fabien
75491 Paris Cedex 10

UN EXEMPLE DE BON ÉDITEUR

SUITE à l'article publié dans *LIST* 6 (page 25), « L'exemple d'un bonne correction », concernant le PC-1251, je tiens à vous signaler l'éditeur du New-Brain qui me semble encore plus performant.

Il s'agissait pour moi de transformer la ligne 250 IMPT (''NOMBRE SUIVANT ?'')S, en la ligne 250 INPUT(''PREMIER NOMBRE ?'')A.
Si on joue le jeu de ne réécrire aucune lettre figurant sur la ligne à modifier, cela donne : 7 touches pour LIST 250 ; 2 touches pour le déplacement du curseur sur M et appui sur N ; 3 touches pour le déplacement sur T, l'insertion et l'appui sur U ; 10 touches pour le déplacement sur N, l'insertion et l'appui sur PREMIER _ ; 3 touches pour le déplacement sur S et l'effacement répétitif avec Shift flèche à droite ; 2 touches pour le déplacement du curseur sur S et l'appui sur A ; enfin, une touche de validation par New-Line.

Au total, 28 touches auront été actionnées et la correction aura duré 20 secondes. Aucun espace n'est à ménager pour les insertions, les compressions de ligne se faisant automatiquement.

Cela étant, *LIST* a choisi un créneau qu'il ne doit surtout pas modifier. Votre revue est une mine d'or pour tout programmeur amateur. Veillez à ce qu'elle le reste.

Michel TRICOCHE
72 Beaumont-sur-Sarthe

■ *Les résultats donnés par le New-Brain sont en effet intéressants. Merci de nous les avoir communiqués.*

Un bon éditeur doit permettre de corriger facilement et rapidement une ligne de programme. Le seul moyen de le juger est d'en faire l'expérience soi-même. Reste à trouver des « tests d'édition » qui permettraient de porter un premier jugement sur l'éditeur de son ordinateur.

Le test proposé dans l'article « L'exemple d'une bonne correction » est un premier pas. D'autres tests (ou idées) pour apprécier l'éditeur sont les bienvenus. Qu'on se le dise !

INDEX DES ANNONCEURS

FERMETURE AUTOMATIQUE

SUR le Canon X-07, l'instruction OPEN n'existe pas. Je pense que je dois la remplacer par INIT. Mais qu'en est-il de l'équivalent de CLOSE ?

Roger NOCETTI
11 Alzonne

■ *L'ouverture d'un fichier se fait bien avec INIT... sur le Canon X-07. Et il n'existe pas d'ordre de fermeture de fichier, pour la simple raison qu'il n'est pas possible d'en ouvrir plusieurs à la fois. Aussi, un équivalent de CLOSE est-il inutile, la fermeture intervenant automatiquement à la fin de l'exécution du programme, ou lorsqu'un autre fichier est ouvert à l'aide d'une instruction INIT.*

DE MEILLEURES SOLUTIONS POUR LES JEUX ET CASSE-TÊTE

DANS chaque numéro de *LIST*, vous trouvez l'énoncé de jeux et casse-tête informatiques dont la solution paraît dans le numéro suivant (dans ce numéro, voir page 82). Mais ce n'est pas toujours l'unique solution. Vous êtes nombreux à nous en proposer d'autres, souvent meilleures. Nous vous soumettons ici celles des jeux 10 et 15, parus respectivement dans *LIST* 4 (page 83) et *LIST* 5 (page 84).

L'objet du jeu 10, « Gagner des microsecondes », était d'écrire un programme chargé d'initialiser à zéro les 100 éléments d'un tableau de façon à ce que son temps d'exécution ne dépasse pas 0,15 seconde. Cela est possible en « dépliant » la boucle, c'est-à-dire en écrivant :

```
10 for I=1 to 100 step 2
20   T(I)=0
30   T(I+1)=0
40 next I
```

dont le temps d'exécution total est de 141500 microsecondes.

Olivier Martin et Douglas Rutledge, proposent de réduire encore ce temps de calcul en dépliant la boucle non pas d'un facteur 2,

mais d'un facteur plus élevé. Par exemple, avec un pas égal à 4, notre programme devient :

```
10 for I=1 to 100 step 4
20   T(I)=0
30   T(I+1)=0
40   T(I+2)=0
50   T(I+3)=0
60 next I
```

Le temps de calcul s'élève alors à : 25 × 850 pour la ligne 10, 25 × 880 pour la ligne 20, 25 × 1100 pour la ligne 40 et 25 × 1100 pour la ligne 50. Soit : 21 250 + 22 000 + 27 500 + 27 500 = 125 750 microsecondes.

Ainsi, plus la boucle est dépliée, plus le temps de calcul de notre programme se réduit. Il faut toutefois noter que le pas de la boucle doit être un diviseur de sa borne supérieure. Dans le cas qui nous intéresse, seuls les pas 1, 2, 4, 5, 10, 20, 25, 50 et 100 sont donc possibles.

Olivier Martin et Douglas Rutledge ont déterminé l'équation du temps de calcul de ce programme. Ce temps, en microsecondes, est :

$T = (850 \times (100/pas)) + (880 \times (100/pas)) + (1100 \times (pas-1) \times (100/pas))$.

Ce qui peut encore s'écrire : $T = (100/pas) \times (850 + 880 + (1100 \times pas) - 1100) = 100 \times (1100 + (630/pas))$.

La représentation graphique d'une telle équation, où le temps est fonction du pas, est une hyperbole qui tend vers une valeur égale à 110 000 microsecondes lorsque le pas tend vers l'infini. Toutefois, comme le pas ne peut pas être supérieur à 100, le programme le plus court s'exécute, d'après cette équation, en 110630 microsecondes. Ce programme est le suivant :

```
10 for I=1 to 100 step 100
20   T(I)=0
30   T(I+1)=0
.....
1010   T(I+99)=0
1020 next I
```

Mais dans ce cas, il est possible de supprimer les lignes 10 et 1020, puisque la boucle n'est exécutée qu'une fois. Cela permet de gagner encore 850 microsecondes, et le temps de calcul passe à 109780 microsecondes.

Enfin, une dernière amélioration consiste à supprimer la variable I. Le programme suivant :

```
10   T(1)=0
20   T(2)=0
.....
1000   T(100)=0
```

s'exécute en 88000 microsecondes. Il paraît difficile de faire plus court, en temps de calcul.

En revanche, ce temps gagné risque bien d'être perdu par l'écriture de ce programme. A moins d'avoir prévu un logiciel qui génère automatiquement les boucles dépliées...

Quant au jeu 15, « Une autre fonction aléatoire », il consistait à trouver une formule générant un nombre aléatoire compris entre A et B, à l'aide de la fonction RANDOM qui retourne un nombre pseudo-aléatoire compris entre 0 et 32767.

Un petit oubli a fait sauter, dans la première formule qui était proposée, la fonction INT. En effet, comme les nombres aléatoires doivent être entiers, il fallait bien entendu écrire :

```
INT ((B-A+1) * RANDOM/32768.0 + A)
```

Toutefois, Henri Bourboulon (de Dunkerque), a découvert que cette formule n'a pas un comportement satisfaisant. En effet, elle fournit des nombres aléatoires qui ne sont pas parfaitement centrés sur l'intervalle de A à B. Ce défaut de centrage peut être observé sur un grand nombre d'essais. Il se situe environ à 0,003 % de la valeur moyenne.

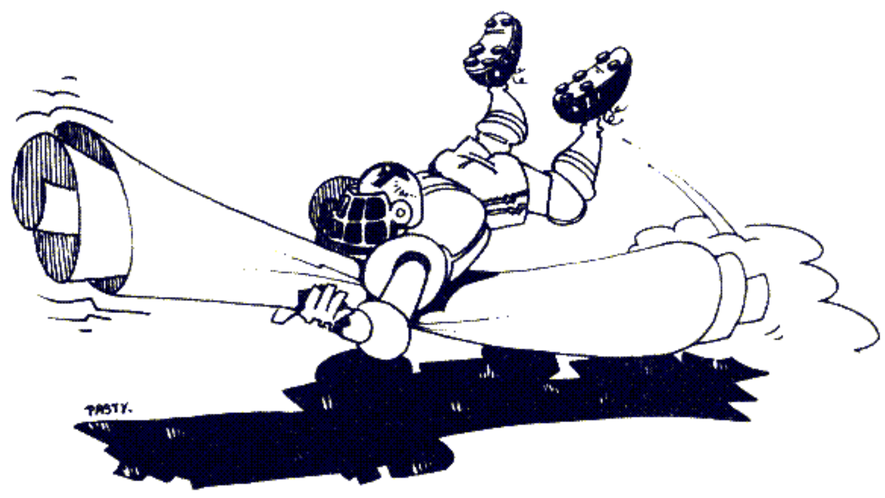
Le défaut de cette formule provient de deux effets qui se compensent. Le premier, en passant par l'expression (B − A + 1) et la division par 32768, étend l'intervalle de A à B en un autre intervalle allant de A à B+1. Le second effet, en prenant systématiquement la partie entière des nombres calculés, a tendance à en diminuer la valeur, et donc à ramener l'intervalle des nombres obtenus de A à B. Mais la répartition en est altérée.

Pour corriger cette erreur, Henri Bourboulon a modifié la formule. Elle devient :

```
ROUND ((B-A) * RANDOM/32767.0 + A)
```

où la fonction ROUND prend l'arrondi d'un nombre réel. Ainsi, la répartition des nombres s'effectue bien entre A et B, l'arrondi permettant de ne pas faire glisser cet intervalle vers des valeurs inférieures.

Les résultats expérimentaux confirment ces faits. Sur un grand nombre de tirages (cent millions), la valeur moyenne obtenue avec la première formule est égale à 4999,612, alors que la formule de Henri Bourboulon donne un meilleur résultat, puisqu'il est égal à 4999,763, l'erreur résiduelle provenant sans doute du générateur de nombres aléatoires.



LA PROGRAMMATION DES JEUX VOUS PASSIONNE !

Des algorithmes en Pascal pour construire des programmes en s'appuyant sur une classification logique des jeux de réflexion les plus classiques.

"La programmation des jeux de réflexion" par Louis Jardonnet
106 pages - 110,00 FF.

LJ4

Envoyer ce bon accompagné de votre règlement à :

P.S.I. DIFFUSION
B.P. 86-77402
LAGNY/MARNE CEDEX
Tél. (6) 006.44.35

Nom _____
Prénom _____
Adresse _____
_____
Code Postal _____
Ville _____

Je commande "La programmation des jeux de réflexion" et joins un chèque de F 110,00.

LA GAZETTE DE LIST

Edit-Plus pour les mordus de Basic

ISOSOFT lance sur le marché un nouveau logiciel d'édition de programmes, Edit-Plus destiné aux possesseurs d'Oric-1 et d'Atmos.

Editeur pleine page accompagné d'un Basic francisé, le nouveau produit de la société française Isosoft devrait faciliter la tâche des « mordus » de la programmation Basic en leur permettant d'écrire en anglais ou en français leurs programmes.

Le nouveau logiciel numérote également automatiquement les lignes d'écriture de programmes et permet une recherche facile de mots ainsi que leur remplacement éventuel. Le même logiciel dispose en outre des fonctions de suppression de lignes REM et de suppression des espaces superflus dans les lignes de programmes. ■

Baisse de prix chez Ediciel

EDI-LOGO, le Logo de **Matra/Hachette** pour la gamme Apple II (voir notre essai dans *LIST* 6) passe de 1 495 FF (prix public ttc) à 950 FF.

Dans le même temps, et toujours chez **Ediciel,** le traitement de texte Papyrus pour Apple passe de 850 à 680 FF. ■

Un nouveau clavier pour les TO7

LA société française **Peritek** a annoncé la sortie d'un nouveau clavier mécanique destiné aux micro-ordinateurs TO7 et TO7-70 de la firme **Thomson.**

Caractérisé par une grande ergonomie, le nouveau clavier devrait être commercialisé à environ 10 000 exemplaires durant l'année 1985. Actuellement, le parc français de TO7 et de TO7-70 s'élèverait à quelque 70 000 appareils.

Grâce à une disposition des touches identique à celle d'un clavier au standard Azerty le nouveau produit de la firme lyonnaise Peritek évite à son utilisateur tout effort d'adaptation en lui offrant une sécurité et un confort de frappe très nettement supérieurs aux claviers d'origine... Le clavier Peritek est actuellement disponible au prix de 750 FF ttc. ■

Le Stratos, enfin...

OFFICIEL : le *Stratos* arrive en France. Il y a si longtemps que la société britannique **Oric** annonçait un débarquement imminent que l'on commençait à douter sérieusement...

Le Stratos sera disponible très prochainement à un prix voisin de 3 000 FF. Le dernier-né d'Oric possède une mémoire vive de 64 Ko et fonctionne grâce à un microprocesseur 6502. Il peut utiliser les logiciels compatibles avec l'Atmos.

Son système de fonctionnement est basé sur deux connecteurs de cartouches. L'une destinée au langage de programmation, l'autre à une cartouche d'application.

Livrée avec une cartouche *Basic super étendu,* la machine pourra fonctionner, entre autres, sous Logo et sous Forth. ■

Un nouveau Multitech

APRÈS divers essais (que les mauvaises langues qualifient de malheureux), **Multitech** ne se décourage pas et lance sa troisième génération d'ordinateurs d'initiation. Principale différence avec les modèles précédents, le MPF 1/88 est destiné aux programmeurs déjà aguerris.

Débogueur, assembleur, gestion des entrées-sorties, sous-routines de programmation, le nouveau Multitech se veut un parfait outil d'apprentissage des subtilités que doit connaître un programmeur.

Il dispose en outre d'un mini écran de contrôle à cristaux liquides (deux lignes de 24 caractères) en plus du moniteur, de 4 Koctets de mémoire vive extensibles à 24 Ko, et de 16 Koctets de mémoire morte. ■

UN LIVRE

La solution RS-232
Joe Campbell
Editions Sybex
Paris, 1984
Broché, 208 pages
Prix : 148 FF

VOICI le livre que tout informaticien, fût-il amateur ou professionnel, devrait avoir dans sa bibliothèque, surtout s'il désire, un jour ou l'autre, enrichir l'environnement de son ordinateur de périphériques connectables à la « norme » RS-232.

Contrairement à ce que le titre laisserait supposer, l'ouvrage ne se contente pas d'exposer doctement les fondements théoriques de l'interfaçage série : il donne une foule de détails *pratiques* sur l'art de la connectique RS-232. Or, il faut bien l'avouer, c'est la pierre d'achoppement sur laquelle butent bien des utilisateurs de matériel informatique, et l'auteur du livre a le mérite de montrer que, si la notion de périphérique *compatible* avec tel ou tel ordinateur est à prendre avec prudence, en revanche, l'idée d'*incompatibilité,* totale ou partielle, entre deux appareils à liaison série n'est qu'une idée reçue : il y a toujours moyen d'arriver à les interfacer. Il suffit de savoir comment, et l'ouvrage est précisément conçu dans cette intention !

Après avoir défini la notion d'interface série, de son principe à la « quincaillerie » (le fameux et mystérieux UART), l'auteur passe à la pratique, en analysant cinq exemples concrets de réali-

Les anciens numéros de

LIST

sont disponibles à la

Librairie Informatique d'Aujourd'hui

253, rue Lecourbe
75015 PARIS

☎ (1) 828 72 88
(de 9 h à 19 h sauf dimanche :
métro Convention ou Boucicaut)

sation de cordons reliant un ordinateur à un périphérique, imprimante et modem. Sur ces cinq cas, trois étaient réputés « difficiles », voire « impossibles ». Or, l'auteur prouve le contraire, et montre, par la même occasion, l'inutilité de l'acquisition du coûteux cordon complet à 25 fils : la plupart du temps, trois ou quatre fils suffisaient, ce qui place la réalisation d'un tel cordon à la portée de tout amateur.

Ecrit dans un style agréable et humoristique (l'auteur remarque malicieusement que le prix de l'ouvrage sera amorti dès la confection du premier cordon !), ce livre ne se consulte pas, il se *lit :* l'analyse des cas rebelles prend des allures de « thriller » et c'est avec plaisir que le lecteur découvre que la solution était... élémentaire, mon cher Watson !

Quant à l'outillage et aux appareils de mesure, l'auteur montre qu'il n'est point besoin de disposer d'un laboratoire pour réussir une connexion, et indique comment, pour quelques francs, il est possible de fabriquer un analyseur de port série plus efficace que bien des notices d'utilisation.

Lisez vite ce livre : vous y découvrirez le moyen infaillible de *déterminer le sexe des DTE et des DCE,* et autres merveilles. Cela vaut la peine, non ?

AM ■

UN LIVRE

Musique sur Commodore 64
James Vogel et Nevin B. Scrimshaw
Editions Cedic-Nathan
Paris, 1984
Broché, 160 pages
Prix : 85 FF

VOICI une adaptation française du livre *The Commodore 64 Music Book* écrit par les mêmes auteurs, et édité aux Etats-Unis.

Le but de cet ouvrage, tel qu'il est annoncé par ses auteurs, est d'enseigner aux novices en programmation comme aux musiciens débutants comment tirer parti des capacités musicales du C. 64. La démarche est donc originale : initier à l'informatique par le biais de la musique ne se pratique pas couramment.

Cinq chapitres subdivisent l'ouvrage. Le premier expose les théories de base : qu'est-ce qu'un son, une fréquence, une hauteur ; forme des ondes sonores, enveloppes, filtres, modulations et synchronisations... Toutes choses connues des habitués de musique synthétique, mais dont la découverte sera fort utile aux débutants. Et cela d'autant plus que des explications claires (assorties de courts programmes démonstratifs) sont fournies pour chacun des points abordés.

Les chapitres suivants entrent plus directement dans le vif informatique du sujet. Depuis la programmation des paramètres musicaux « de base » sur le C.64 jusqu'au traitement des effets sonores dans lesquels cette machine est remarquable, les explications demeurent claires et précises. Remarquons toutefois que les carences du Basic Commodore, qui oblige à programmer la musique à grands renforts de POKE et autres FOR...NEXT, rendent délicate la tâche envisagée par les auteurs. Initier à l'informatique en commençant par ce type d'instructions et en poursuivant par les READ, DATA et RESTORE, complétés de l'usage des tableaux assortis d'une pincée de GET, tout ceci est peut-être un peu « pointu » pour les vrais débutants. Aussi, bien que judicieusement conçu, l'ouvrage s'adresse plutôt à ceux qui ont déjà fait leurs premières armes en matière de logique informatique.

L'échelonnement des difficultés est bien étayé par des exemples de programmes de plus en plus élaborés. Les amateurs de musique seront heureux de pouvoir pratiquer la modulation en anneau, le filtrage ou la synchronisation. Tout est prévu d'ailleurs pour cela, et les explications sont détaillées. Le seul problème réside dans l'ensemble des connaissances informatiques préalables qu'il vaut mieux avoir avant de se lancer dans les applications proposées.

C'est donc un livre d'accès (relativement !) facile qui apportera à son utilisateur les connaissances qui pouvaient lui manquer en matière de musique de synthèse, et qui lui servira à progresser dans le domaine de la programmation avancée.

RB ■

Spectrum Plus... moins cher

LE constructeur britannique **Sinclair** a décidé de procéder à une baisse significative du prix de son ZX Spectrum Plus. L'appareil sera désormais commercialisé à 1660 FF ttc contre 2330 FF précédemment, ce qui le rend évidemment beaucoup plus attractif.

Autre baisse spectaculaire : les cartouches de bande magnétique, dites « microdisquettes », qui tournent sur les microdrives des Spectrum, Spectrum Plus et QL voient leur prix passer de 76 à 39 FF ttc. Cette chute des prix, selon Sinclair, a été rendue possible grâce à une forte croissance des capacités de production de la firme ainsi qu'à de nouvelles techniques de fabrication. ■

UN CLUB... UN SALON... UNE FOIRE...

En Saintonge, informatique et télématique s'exposent

A Saintes, les 26, 27 et 28 avril prochain, se tiendra le deuxième *Salon de l'informatique et de la télématique.* Il est organisé par l'Association pour le développement de l'informatique en Poitou-Charentes et s'installera au Parc des Expositions de la ville.

Si vous voulez en savoir davantage :
ADICHAP
Hôtel de ville
17100 Saintes
Tél. : (46) 93 34 45

Belgique : un bulletin pour le Dai

SUITE au meeting international du Dai, à Nivelles, le 20 octobre dernier, où près de 300 entrées ont été enregistrées, les organisateurs ont pris accord avec les responsables des différents clubs Dai de Belgique, pour créer un bulletin entièrement consacré à cet ordinateur.

Le bulletin s'appellera DAIclic et sera publié par l'association IDC (International Dai Club).

Les clubs associés (ils sont actuellement au nombre de trois : Carola Dai, Daic et Dainamur) abonneront directement leurs membres. Pour plus de renseignements, vous pouvez contacter :
Fabrice Duluins
4 allée Tour Renard
B-1400 Mivelles
Marc Vandermeersch
17 avenue du Vert Bocage
B-1410 Waterloo
Christian Poels
10 rue des Bas-Sarts
B-4100 Seraing

Une Foire aux Puces dans le Morbihan

L'Association pour la Promotion et la Diffusion de l'Informatique organise la 1ere Foire aux Puces du Centre Bretagne. Elle se tiendra à Pontivy le dimanche 21 avril 1985, dans le Château des Ducs de Rohan, qui fête cette année son 500e anniversaire.

De 10 à 19 heures, on pourra échanger, troquer, admirer tout matériel informatique (micros, jeux vidéo cassettes, logiciels). Vous pouvez vous renseigner auprès de Brigitte Milou, (97) 25 34 00.
APDI
Centre Informatique de Pontivy
Place des Ducs de Rohan
56300 Pontivy ■

Thomson, l'informatique s'inscruste sur le petit écran

QUE nous réserve la dernière nouveauté de **Thomson,** enfin disponible et qui fonctionne à la fois sur MO5 et sur TO7-70 ? Avec ce module dit « d'incrustation », il devient possible d'afficher sur un téléviseur les messages de l'ordinateur alors même que se poursuit une émission. Il en coûte 490 FF.

Le petit boîtier d'extension se connecte à l'arrière du MO5. Le branchement est donc extrêmement simple, mais si vous possédez déjà l'extension pour manettes de jeux, dans le cas du MO5, il vous faudra choisir...

A la mise sous tension de l'ordinateur, rien ne paraît avoir changé. Pour que l'incrustation soit effective, il faut positionner à 1 le cinquième paramètre de SCREEN. C'est grâce à un SCREEN 4,0,0,,1 que l'image télé nous est apparue, le texte venant du MO5 étant alors affiché en surimpression bleue du plus bel effet. Et SCREEN 4,6,6,,1 a fait revenir au mode normal.

Pour une meilleure compréhension, rappelons la signification des différents paramètres de SCREEN. Dans l'ordre : couleur de l'écriture, couleur du fond, couleur du pourtour, permutation des couleurs d'écriture et de fond (0 ou 1), et incrustation (0 ou 1). Les 16 couleurs disponibles sont représentées par les nombres de 0 à 15. Ainsi, le 0 correspond au noir, le 4 au bleu, le 6 au bleu clair, le 8 au gris, etc.

Le positionnement à 1 de l'incrustation a pour effet de rendre « transparents » à l'image télé tous les points de l'écran qui ont été définis en noir. Le SCREEN 4,0,0,,1 s'explique donc : le fond et le pourtour étant définis noirs, l'image de l'émission passe partout sauf à l'endroit du texte (bleu).

Si l'on veut suivre des cours d'informatique diffusés par la télévision et en même temps les pratiquer, il est intéressant de se garder quelques lignes pour travailler tout en laissant passer l'émission sur tout le reste de l'écran. Il suffit pour cela d'exécuter ces quelques lignes de Basic :

```
10 CONSOLE 0,24
20 SCREEN 0,0,0,,1
30 CLS
40 CONSOLE 19,24
50 COLOR 4,6
60 CLS
```

Vous pouvez aussi vous amuser à de petits effets comme, par exemple, laisser passer l'image uniquement à travers le texte. Et vous pouvez aussi, comme le suggère le *Guide du MO5* livré avec la machine, « placer des moustaches sur le visage de votre commentatrice favorite ».

Et le son dans tout cela ? Dès que l'incrustation est demandée, le son de la télé passe lui aussi. Mais, contrairement à l'image, la superposition n'est pas possible : les bips du clavier et les ordres PLAY deviennent muets. C'est dommage.

Quoi qu'il en soit, ce module d'incrustation est une extension très originale dont la principale (et unique ?) application reste pour l'instant les cours d'informatique de la télévision. CB ■

UN PETIT TOUR CHEZ LE LIBRAIRE

Graphismes en Kits
Michel Rousselet
Editions Techniques et Scientifiques Françaises
Paris, 1984
Broché, 262 pages
Prix : 122 FF

Jeux sur VG 5000 Philips
Benoît Amsler
Olivier Villemaud
Editions Edimicro
Paris, 1984
Broché, 214 pages
Prix : 88 FF

Le Basic sur le bout des doigts
Herbert Peckham
Traduit par Léon Collet
Editions Mac Graw Hill
Paris, 1984
Reliure spirale
Prix : 135 FF

Pascal pour programmeurs
Olivier Lecarme
et Jean-Louis Nebut
Editions Mac Graw Hill
Paris, 1985
Broché, 394 pages
Prix : 170 FF

Le langage C
Brian W. Kernighan
et Dennis M. Ritchie
Traduit par Thierry Buffenoir
Editions Masson
Paris, 1984
Broché, 218 pages
Prix : 140 FF

Dessiner, peindre... et jouer avec Alice
Louis Gros
Editions Eyrolles
Paris, 1984
Broché, 142 pages
Prix : 79 FF

Du Logo pour Apple
Nicole Bréaud-Pouliguen
Editions du PSI
Lagny, 1984
Broché, 114 pages
Prix : 80 FF

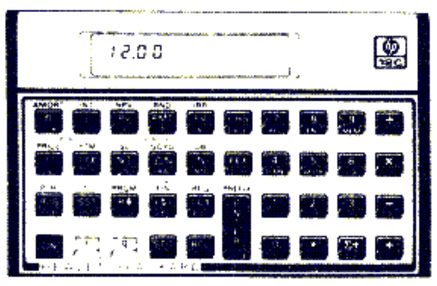
Lire Logo
André Myx
Editions Cédic-Nathan
Paris, 1984
Broché, 110 pages
Prix : 75 FF

ORDINATEURS de POCHE et CALCULATRICES

HEWLETT - PACKARD

SUPER PRIX DE PRINTEMPS !

ORDINATEURS de POCHE
HP 41 CV
HP 41 CX
HP 71 B



SCIENTIFIQUES HP 11 HP 15
FINANCIÈRES HP 12
PERIPHERIQUES et ACCESSOIRES

MAUBERT ELECTRONIC 49, bd. St Germain. PARIS 5° TEL. 325.88.80 PLACE ET M° MAUBERT

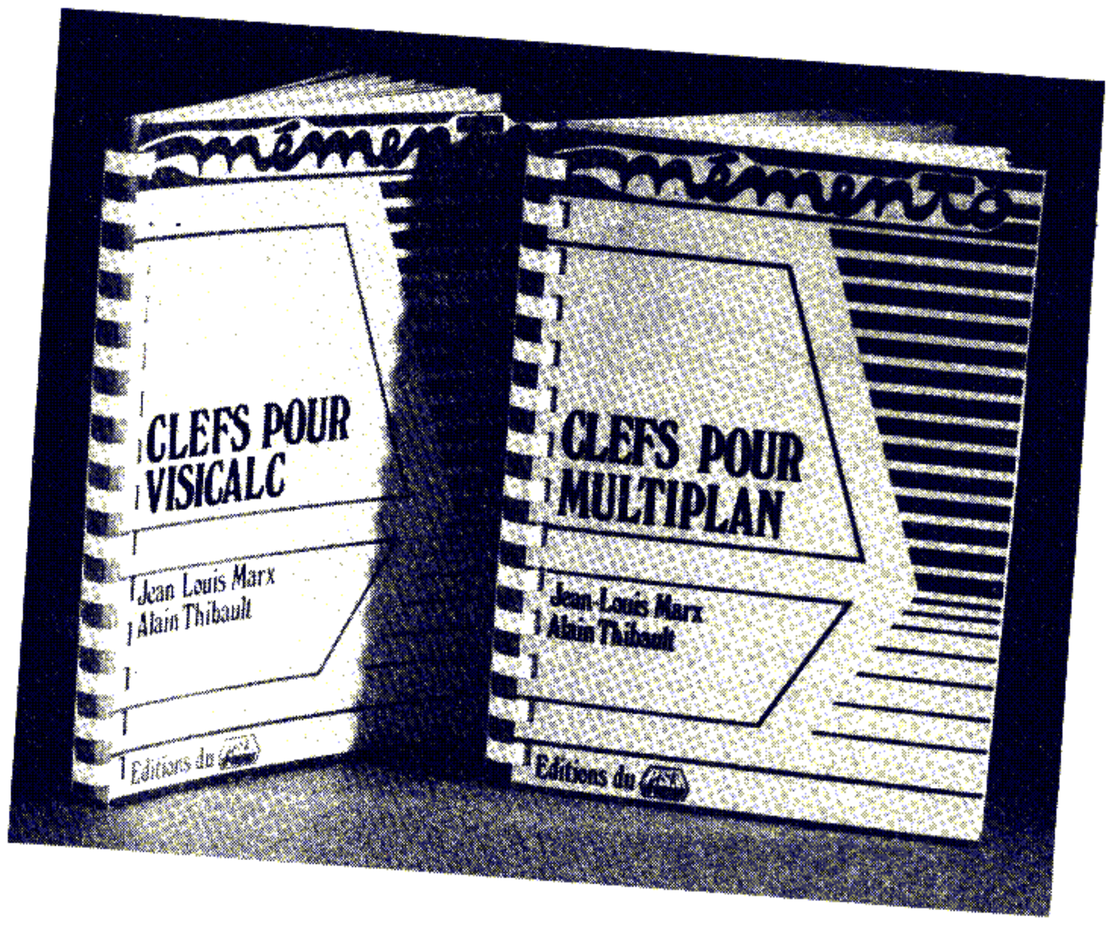
black

LA GAZETTE DE LIST

DEUX LIVRES

Clefs pour Visicalc
Jean-Louis Marx
et Alain Thibault
Éditions du PSI
Lagny, 1984
Reliure spirale, 102 pages
Prix : 105 FF

Clefs pour Multiplan
Jean-Louis Marx
et Alain Thibault
Éditions du PSI
Lagny, 1984
Reliure spirale, 128 pages
Prix : 105 FF



ON sait bien que l'informatique individuelle n'a vraiment pris son essor que lorsque sont apparus des logiciels directement utilisables sans aucune connaissance en programmation. Ou plus exactement lorsque Visicalc est né, en septembre 1979. S'il n'est plus tout à fait aujourd'hui le plus répandu, son nom est toujours un symbole : celui du *Tableur* et il reste évidemment très répandu. Son grand, et heureux, rival s'appelle Multiplan. Si Visicalc a surtout été le cheval de bataille (et l'entraîneur) d'Apple, Multiplan est celui de l'IBM-PC. A eux deux, ces programmes miracles, feuilles de calcul à tout faire, figurent sur la majorité des bureaux professionnels mais aussi de nombreux amateurs « éclairés ».

Il est certain que la très grande majorité de leurs utilisateurs ne connaît guère l'informatique. Ce qu'ils désirent est très simple : un outil performant dont ils ne maîtrisent à la rigueur que les 30 % des performances adaptées à leur travail quotidien. Leur plus grand besoin est évidemment celui d'un répertoire rapide, de consultation immédiate, écrit avec un grand souci pédagogique. Sinon, à quoi bon recommencer l'écriture du guide officiel ? Certes, quand on voit le pavé indigeste et majestueux qui accompagne les disquettes de Multiplan, on comprend que certains reculent devant sa consultation et préfèrent même payer (ou faire payer par leur entreprise) très cher un stage d'initiation de deux ou trois jours... Certains complètent cette démarche par l'achat, parfois justifié, de volumes spécialisés qui donnent des tableaux tout prêts, répondant aux besoins estimés de telle ou telle part du public.

Quoi qu'il en soit, il reste absolument indispensable à tous les utilisateurs de Visicalc et/ou Multiplan d'avoir constamment, à portée de main, un guide de taille très raisonnable à consulter du bout du doigt, avec des rubriques claires (une par page par exemple pour chaque mot clé), de nombreux exemples élémentaires, etc. Il faut aussi que ce manuel contienne quelques indications, même succinctes, sur des notions complexes (insuffisantes pour une maîtrise complète), mais assez fournies et claires pour qu'on sache ensuite aborder les chapitres correspondants du mode d'emploi sans être complètement perdu. Ces concepts difficiles paraîtront déjà moins hermétiques, si ce n'est vaguement familiers.

Si de plus ce guide est bourré, malgré sa taille, de « trucs et comment » (jeu de mots autant que traduction de « know-how » : savoir-faire), nul doute qu'on ne se trouve devant un utilitaire, au vrai sens du mot, dont la consultation est si facile et si payante qu'il devient — je peux en témoigner, ayant utilisé le second titre pour m'initier à Multiplan — indispensable à chaque instant.

Il existe donc deux de ces guides, dans la série bien connue « Clefs pour... », écrits d'après la même maquette par deux auteurs parfaitement rompus à la fois à la vulgarisation — pas à la simplification — et aux performances de ces *calques* pratiquement universels. Jean-Louis Marx et Alain Thibault ont su dégager de l'utilisation de Visicalc et de Multiplan les instructions et manipulations les plus courantes, qui couvrent 80 % des besoins, et de les classer avec efficacité et souplesse. Bien qu'ils soient encore tout récents, on retrouve déjà ces deux livres glissés dans bien des paquets de disquettes. On ne prend sûrement pas un gros risque à leur promettre un avenir très sûr, car leurs qualités d'écriture ont rencontré ici un besoin très évident dans le public. Il ne reste qu'à attendre d'autres « Clefs pour... » : à quand Lotus 1-2-3 ou Framework, ou des éditions adaptées aux petits nouveaux Visicalc Advanced Version ?

AW ■

1985 : année du Mac ?

S'IL est un matériel qui a su se forger rapidement non seulement des adeptes, mais encore des admirateurs, des fans, c'est bien le Macintosh. Il aura suffi de quelques mois en effet pour que cet ordinateur devienne à la mode. Dans certains milieux d'ailleurs, ce succès ne vas pas sans une pointe de snobisme. A en croire plus d'un utilisateur, tout ce qui est Mac serait par essence irréprochable. Le fin du fin dans cet ordre d'idée consiste à estimer que le prix de la machine est dérisoire. Et c'est tout juste si l'on ne doit pas s'étonner qu'**IBM** ait commis l'erreur de ne pas rendre son PC « compatible Macintosh ».

Quoi qu'il en soit, le Macintosh est promis à un bel avenir, et l'on peut déjà se demander si 1985 ne sera pas l'année du Mac. Encore que toute prospective soit une activité à haut risque, essayons de voir les tendances qui se dégagent concernant les logiciels, pour ce petit ordinateur.

Bien entendu, il est hors de question d'en dresser un catalogue complet, mais nous pouvons tout de même faire un premier tour non sans mettre en garde, comme il convient quand on rapporte des bruits. Restons en effet circonspects : en informatique, l'expérience montre qu'il vaut mieux, surtout au moment d'acheter un matériel, jouer les Saint-Thomas, considérer que les promesses ne seront pas tenues et que les produits annoncés ne viendront jamais.

Cela dit, si la tendance se confirme, la diversité et l'originalité devraient distinguer les nouveaux logiciels du petit **Apple** ; qu'on en juge plutôt. Aux jeux rebattus qui tournent sur la plupart des ordinateurs, plusieurs concepteurs espèrent substituer des logiciels de divertissement destinés à un public « plus mûr et plus sérieux ». Dans cette optique, deux éditeurs britanniques seraient en train de mettre la dernière main à des programmes de distraction pour hommes (noms de code : MacAdam et Mac-Wait) ou de préparation des vacances et voyages (Mac-Route). Toujours en Grande-Bretagne, mais pour des fins plus austères, on serait sur le point d'annoncer la commercialisation d'un ensemble de trois logiciels intégrés de conception révolutionnaire : Mac-1, Mac-2 et Mac-3 ; ce serait, et de loin, ce qui se fait de plus rapide dans la catégorie.

Chez nous, on paraît spécialement attentif à l'enseignement assisté par ordinateur, mais il ne s'agit pas d'apprendre à des enfants les tables d'addition ou le pluriel des mots composés : plusieurs de ces didacticiels seraient du niveau de la licence. La littérature est à l'honneur, en

particulier celle des humanités grecques avec (nous ne citons que les deux plus célèbres) Andro-Mac (version d'Euripide) et Mac-Homère ; plus près de nous, le roman français contemporain est également à l'honneur avec l'inévitable Mac Orlan. Dans les deux premiers cas, le logiciel devrait redéfinir en partie le clavier du Macintosh en apportant un jeu de caractères helléniques.

Enfin, nous avons gardé pour la bonne bouche un logiciel de simulation que nous avons pu essayer en avant-première et qui nous a fait une forte impression. Rien de plus facile que de transformer son Macintosh en une jolie frégate et de prendre le large. On regrettera seulement que l'éditeur n'ait pas prévu une version en français (l'original est en flamand !). Destiné principalement aux marins de plaisance, Mac-Row permet, entre autres choses, de s'exercer à la pêche à la traîne. ■

Interface série/parallèle

LA société **Néol** annonce la commercialisation d'une interface (mémoire-tampon de 8 Ko en option) permettant de relier un périphérique équipé d'une liaison parallèle type *Centronics* à la sortie V24 d'un ordinateur.

Sont déjà disponibles une interface parallèle graphique pour le C.64, une interface parallèle pour Apple IIc et pour Atari 600 et 800. Prix avec le câble d'entrée : 1365 FF ttc sans mémoire-tampon, et 1625 FF ttc avec 8 Ko.

Pour tout renseignement, s'adresser à
Néol
4a rue Nationale
F 67800 Bischheim-Strasbourg
Tél. : (88) 62 37 52 ■

APL sur un portable

AMPÈRE, le dernier cri des micro portables japonais a de quoi tenir le choc : 64 Koctets de mémoire vive extensibles à 512 Ko, 128 Ko de mémoire morte, une unité de micro-cassette intégrée de 300 Ko de capacité, plus des disquettes et plus tard, un disque dur. Une chose est certaine, on ne manquera pas de mémoire. La principale originalité de l'Ampère ne réside cependant pas dans ses capacités mais dans son langage. C'est effectivement le premier portable à disposer d'APL en version standard.

Selon **Sofrémi** (Société distributrice de l'ordinateur), la raison en est simple : APL est, avec ses 40 signes, un des langages de programmation les plus simples, les plus évidents et également les plus compacts.

De plus, APL ne sera pas le seul atout de l'Ampère qui devrait prochainement supporter LPA, le langage naturel développé par le CEA (Commissariat à l'Energie Atomique). L'âge du capitaine, les rendez-vous de la semaine, LPA permet tout type d'interrogation dans le langage le plus simple possible, pour peu que l'ordinateur ait la réponse en mémoire. ■

Poignées de jeux et Modem pour Alice

DE nouvelles extensions sur Alice 90 et Alice 32. Une boîte d'extension (250 FF) permet désormais aux amateurs de jeux d'arcades de disposer de poignées.

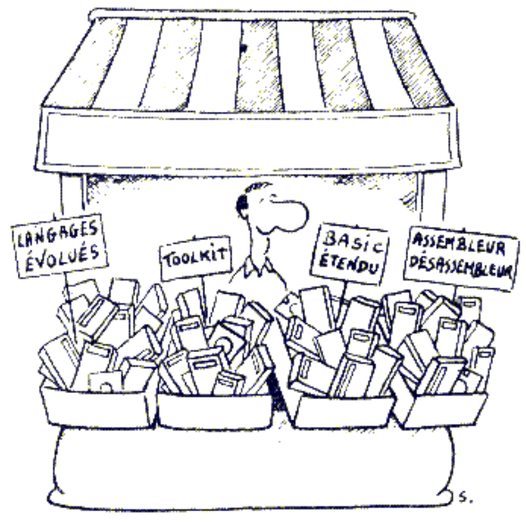
A partir du mois de mai devraient être aussi disponibles un lecteur de disquette 3 pouces 1/2 (2 950 FF) et un Modem (1 495 FF), qui permettra de connecter Alice à une ligne téléphonique ou à un serveur *Vidéotex*. ■

C'est du solide, et on le prouve !

DEPUIS le lancement de sa série 10, à la fin de 1981, **Hewlett-Packard** insiste à juste titre sur la fiabilité et la robustesse des calculatrices de cette gamme : HP-11C, 12C, 15C et 16C. Selon le constructeur, le degré de fiabilité de ces machines atteint en moyenne, actuellement, des résultats exceptionnels : 1 panne/50 ans.

Joignant le geste à la parole, Hewlett-Packard offre, à partir du 1er mars 1985 cinq ans de garantie sur les calculateurs programmables de la gamme 10. Evidemment, ce n'est pas cinquante ans, mais c'est nettement mieux que la plupart des garanties couvrant les matériels de poche. ■

CASSETTES ET DISQUETTES



X PER
Gestion de bases de connaissance
Disquettes pour C.64 et Apple II
Edité par Micro Application
Version pour C.64 : 950 FF
Version pour Apple II : 1950 FF

Graphique Exhibitor
Permet de re-combiner dessins ou textes et de les imprimer
Disquette pour Apple II
Edité par BIP
Prix : 439 FF

Kit écran
Utilitaire d'animation et de création d'écrans graphiques basse résolution
Cassette pour Oric-1 et Atmos
Edité par ARG Informatique
Prix : 120 FF

Basic Turbo
Utilitaire d'aide à la programmation
Cassette pour Oric-1 et Atmos
Edité par Cobra Soft
Prix : 140 FF

Supercopy Ecran
Copies d'écran
Cassette pour Oric Atmos et GP 100 A
Edité par ARG Informatique
Prix : 120 FF

DE LA METHODE POUR INVENTER VOS PROGRAMMES

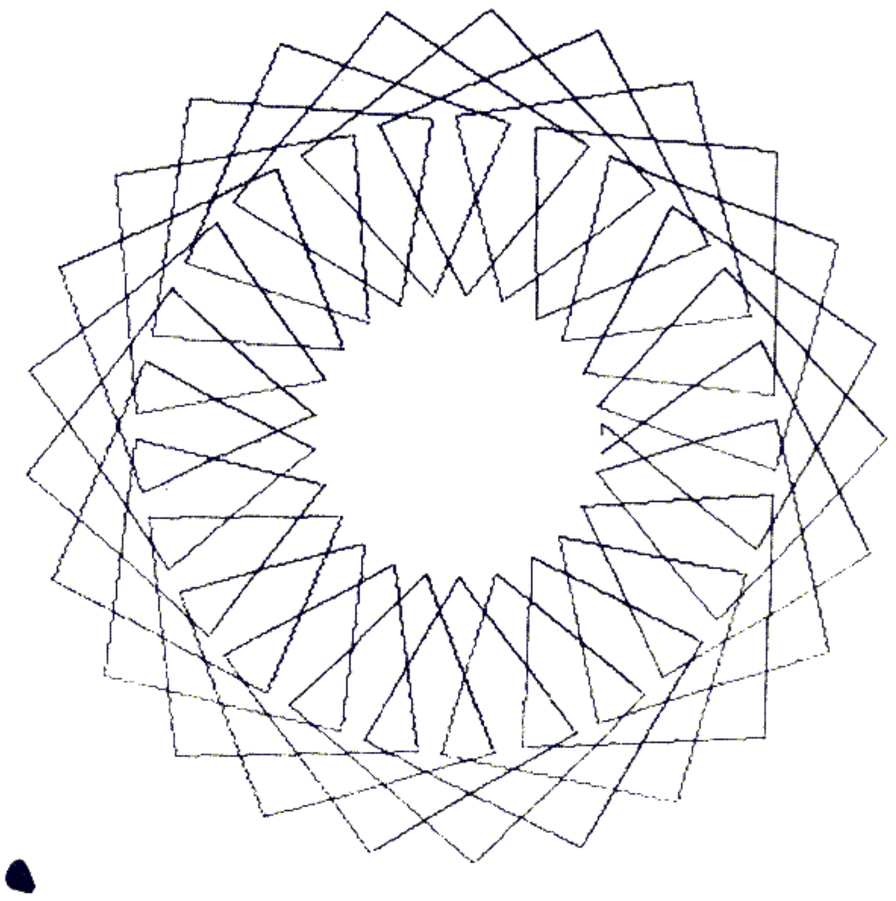
"Programmation inventive" par Xavier de la Tullaye
160 pages - 100,00 FF.

- **mieux utiliser les mots de la programmation.**
- **mettre au point un organigramme.**
- **réaliser le programme, de son invention à son utilisation.**

Envoyer ce bon accompagné de votre règlement à :
P.S.I. DIFFUSION
B.P. 86-77402
LAGNY/MARNE CEDEX
Tél. (6) 006.44.35
LP4

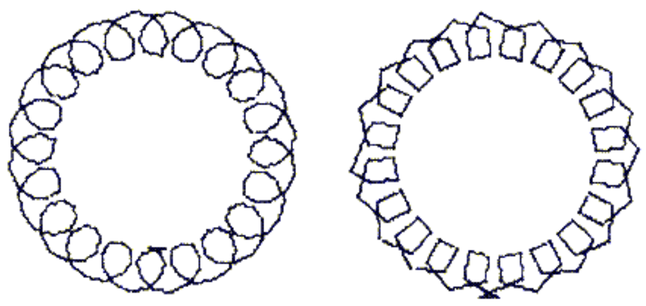
Nom _____ Prénom _____
Adresse _____
Code Postal _____ Ville _____
Je commande la "Programmation Inventive" et joins un chèque de F 100.00.

DES COURBES

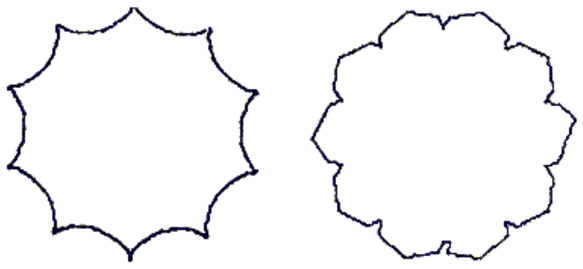


TRÈS ANGULEUSES

A l'origine, ce programme était conçu pour dessiner des développantes de cercles qui sont des courbes très harmonieuses. Mais l'imprimante ne trace que des lignes droites. Alors tant pis pour les courbes ! On a tiré parti des lignes brisées.



En exagérant à peine, on peut considérer qu'un cercle n'est qu'une succession de traits plus ou moins longs. C'est tout à fait exact pour les cercles que dessine la petite table traçante du Canon X-07. Des traits plus ou moins longs... C'est ce « plus ou moins » qui nous réserve des surprises.



Ainsi, dans le cas du cercle, si l'on s'en tient à 4 traits, on obtient un carré. Avec 6 traits, c'est un hexagone, etc. Et dans le cas de la développante ? Par développante ici, nous entendons la trajectoire d'un point quelconque d'une roue qui tourne autour d'une autre. Eh bien, avec les développantes, on obtient des graphismes inattendus. Je n'en dis pas plus, car l'utilisation du programme est suffisamment éloquente. Un petit avertissement cependant : dès le premier jour, j'ai usé quelques mètres de papier et quelques stylos.

Laisser faire le hasard

Vous trouverez, page suivante, le mode d'emploi du programme. Quand vous aurez obtenu un dessin, sachez que vous pouvez en faire tracer un autre concentrique au premier, puis un troisième, etc.

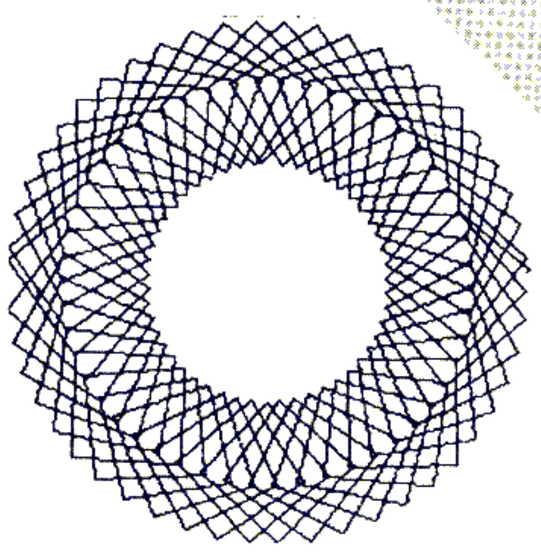
Principales variables utilisées

B = C/D	Rapport des rayons de C1 et de C2
N	Nombre de tours de C1 nécessaire pour fermer la courbe
F	Finesse du tracé, nombre de traits par tour de C1 ($\neq$ de 0)
TT	Nombre total de traits = F × N
R1	Somme des rayons de C1 et C2
R2	Distance du centre de C2 au point M
R	Somme des valeurs absolues de R1 et R2 ($<$ 240)
E	Couleur (E = E MOD 4, pour les étourdis)
A$	Inkey$, boucle d'attente
P	3.1415926535898 × 2
X, Y	Coordonnées successives du tracé
A	Compteur de boucle (angle de rotation de R1)
K	Coefficient : lorsque R1 tourne de A, R2 tourne de K × A − P/2

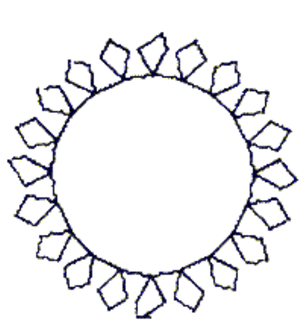
```
10000 CLEAR 100:DEFSTR Ƶ:Ƶ="N"
10010 LPRINT CHR$(18)
10015 CLS:INPUT"Rapport des  cercles C/D
   C=";C:IF INT(C)<>C THEN 10015
10017 INPUT"        D=";D:IF D=0 OR INT(D
)<> D THEN 10017
10020 B=C/D:K=1+B:N=0
10025 N=N+1:IF INT(N*B) <> N*B AND N < D
 THEN 10025
10030 INPUT"  Finesse";F:IF F=0 THEN 100
30
10035 TT=N*F
10040 INPUT"        R1";R1
10050 INPUT"        R2";R2
10055 R=ABS(R1)+ABS(R2)
10060 IF R>240 THEN 10040
10070 INPUT"  Couleur";E:E=EMOD4
10075 CLS:PRINT "C/D:"C"/"D," F="F"R1="R
1,"R2="R2"TT="TT
10077 PRINT "OK=RETURN NON=AUTRE";
10080 BEEP 2,2:A$=INKEY$:IF A$="" THEN 1
0080 ELSE IF A$<>CHR$(13) THEN 10015
10085 PRINT "":LPRINT "C"+STR$(E)
10090 P=2*3.1415926535898
10100 IF Ƶ="N" THEN LPRINT "R240,"STR$(-
R):LPRINT "I"
10110 LPRINT "M"+STR$(R1-R2)+",0"
10150 FORA=0 TO N*P+P/F/2 STEP P/F
10170 X=R1*COS(A)+R2*COS(P/2-K*A)
10180 Y=R1*SIN(A)-R2*SIN(P/2-K*A)
10190 LPRINT "D"+STR$(X)+","+STR$(Y)
10200 NEXT
10205 CLS:PRINT"Dessin concentrique   OU
I = O   NON = N"
10206 Ƶ=INKEY$:IF Ƶ<>"O" AND Ƶ<>"N" THEN
10206
10210 IF Ƶ = "N" THEN LPRINT "M-240,"+ST
R$(-R):GOTO 10010
10220 GOTO 10015
```

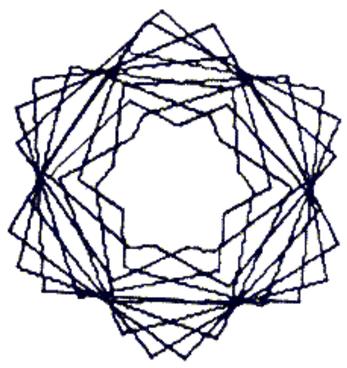
Des courbes très anguleuses
Programme pour X-07 + table traçante

Auteur Rémi Ducros
Copyright *LIST* et l'auteur



Laissez faire votre imagination ou le hasard. Vous pouvez aussi, en cours de tracé, changer de couleur. N'hésitez pas : cela peut être d'un bel effet. Il suffit d'appuyer, pendant que l'imprimante dessine, sur le deuxième bouton blanc en partant de la droite (il est surmonté par 4 traits).





Vous avez enfin la faculté d'appuyer sur la touche Break de votre Canon et de laisser là une œuvre inachevée. Le programme boucle sur lui-même en conservant ceux des paramètres qui ne changent pas d'un dessin à l'autre.

Rémi DUCROS

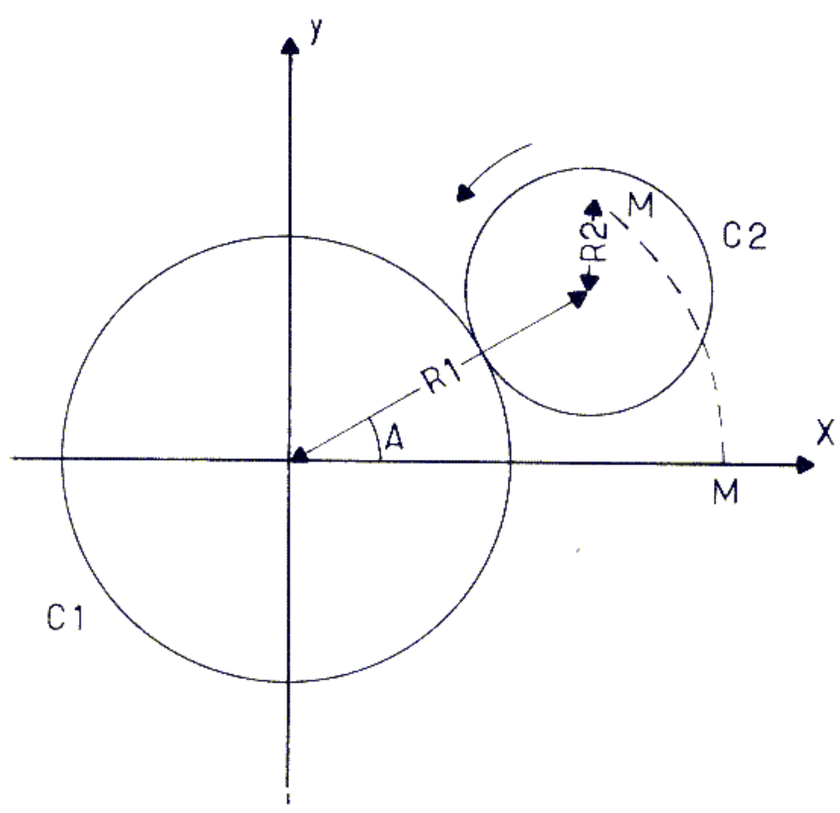
Utilisation du programme

Rentrée des paramètres.

- Rapport des deux cercles C/D. C et D doivent être des nombres entiers positifs ou négatifs (D ≠ 0). Ce rapport détermine le nombre de tours effectués par les deux roues. La ligne 10025 calcule N, le nombre de tours de C1.
- Le paramètre F, comme Finesse, fixe le nombre de traits à tirer par tour. La durée du dessin dépend donc directement de ce nombre. Par ailleurs, plus il est grand, plus la courbe est fine.
- R1 est la somme des rayons des deux roues C1 et C2.
- R2 est la distance du centre de la roue C2 au point M. La somme de la valeur absolue de R1 et de la valeur absolue de R2 doit être inférieure à 240, 480 étant la largeur du papier en unités graphiques. R1 et R2 peuvent être négatifs.
- C, enfin, désigne la couleur du tracé : on a le choix entre les quatre couleurs disponibles.

Le programme affiche ensuite les paramètres entrés et le nombre total de traits qui composeront la figure. La durée du tracé est évidemment fonction de ce nombre (il faut compter environ un à deux segments de droite à la seconde).

Après le Beep de la ligne 10080, la machine attend soit l'annulation (on appuie sur une touche différente de RETURN), soit votre feu vert : une pression sur RETURN et le tracé commence.



Un problème qui n'est pas sans rappeler celui de la lune tournant autour de la terre qui tourne autour du soleil.

UNE MANIÈRE D'ÊTRE RÉCURSIF

MÊME avec un Basic classique comme celui de l'Oric, il est possible de concevoir des fonctions récursives. Elles n'utilisent pas d'instructions spécifiques, mais font appel à une pile. Une technique aisément gérée par le Basic.

Commençons par une définition : la récursivité est le procédé qui définit une fonction ou un objet par une référence à elle ou à lui-même.

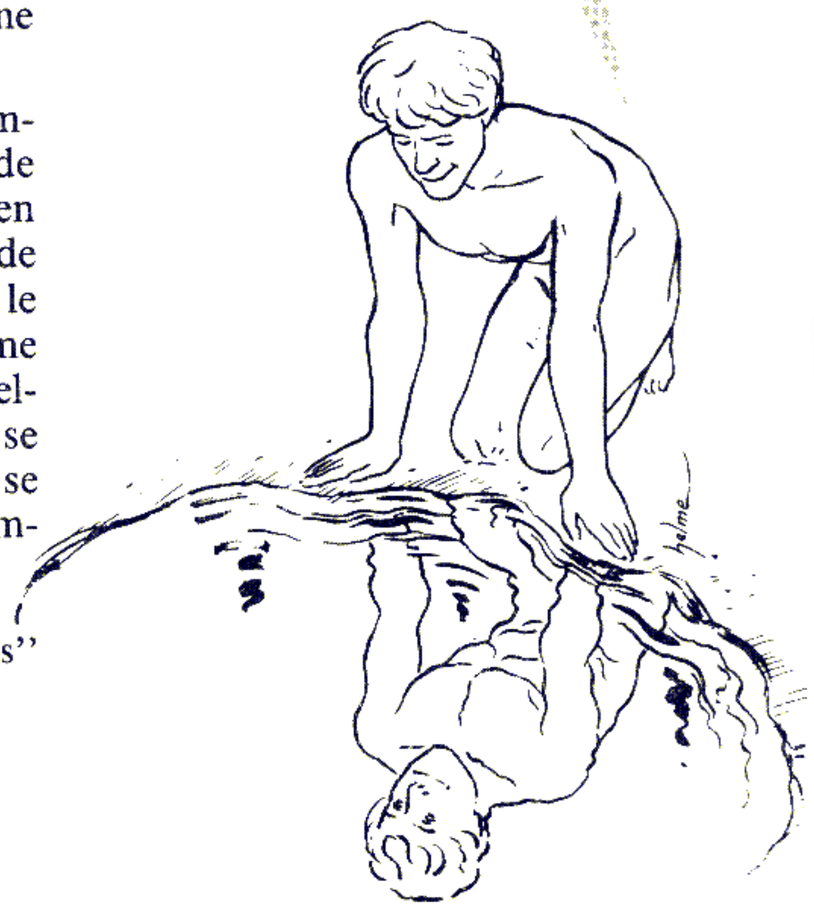
Bien que peu utilisée par les programmeurs, la récursivité s'emploie de manière naturelle. Il suffit, pour s'en convaincre, d'observer le mécanisme de l'exécution des sous-programmes par le compilateur Basic. Un sous-programme peut en appeler un autre qui en appellera un autre, etc. Le Basic doit alors se souvenir du retour et surtout ne pas se tromper pour chacun d'eux. Par exemple :

```
  0 PRINT "appels sous/programmes"
 10 GOSUB 100
 20 PRINT "fin programme"
 30 END
100 PRINT "ss/prog 100"
110 GOSUB 1000
120 PRINT "fin ss/prog 100"
130 RETURN
1000 PRINT "ss/prog 1000"
1010 RETURN
```

En Basic, nous utilisons l'appel à un sous-programme par GOSUB. C'est encore lui qui fera l'appel de fonction. Ce mécanisme est profondément récursif. Essayez :

```
  0 REM recursivite de GOSUB
 10 NBRE=9
 20 GOSUB 100
 30 END
100 REM ss/prog recursif
110 I=I+1 :PRINT "("I;
120 IF I<NBRE THEN GOSUB 100
130 PRINT ")"I; :I=I-1
140 RETURN
```

La récursivité a ses règles. La première à retenir est la suivante : le nombre d'appels d'une fonction à elle-même doit être fini pour que le programme puisse se terminer. Dans le cas d'un GOSUB qui s'appelle lui-même, ce nombre peut aller jusqu'à 24 (chacun peut mesurer, sur sa propre machine, la taille de la pile des retours de GOSUB).



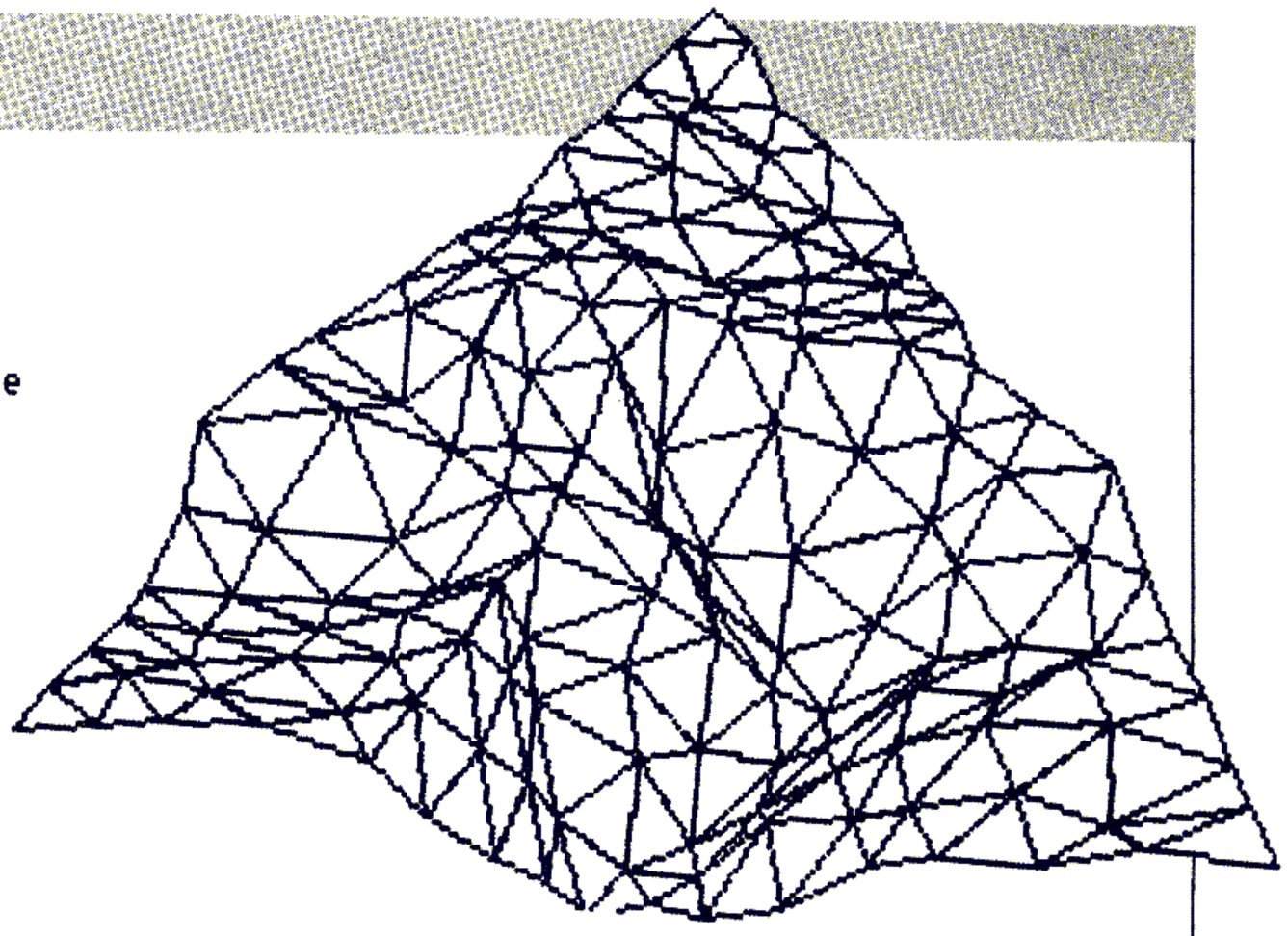
Fonction récursive en pleine action

Des retours délicats

Si le compilateur utilise la récursivité pour son propre usage, celle des programmes n'est pas assurée. En particulier, l'environnement des sous-programmes (les valeurs des variables) n'est pas conservé pour le retour. Seule l'adresse de retour du GOSUB est conservée.

La principale difficulté dans la programmation en Basic sera cette conservation de l'environnement nécessaire au

La fonction fractale
Programme pour Oric-1
Auteur Max Hagenburger
Copyright *LIST* et l'auteur



L'exécution du programme

```
0 REM FRACTALE la fonction recursive
10 GOSUB 100
20 GOSUB 50 :IF DESSIN THEN 20
30 GOSUB 900
40 REM -----
50 GOSUB 200 :REM dessin
60 GOSUB 300 :REM FRACTALE
70 GOSUB 800 :RETURN
90 REM =====
100 REM initialisations
110 GOSUB 700
120 DIM ALEA(19)
130 PRINT "Fonction FRACTALE recursiv
ite du"
140 PRINT " fractionnement aleatoire
(exemple: 0)"
150 PROF$="0"
190 RETURN :REM -----
200 REM debut un dessin
210 PROF=VAL(PROF$)
220 X1=10 :Y1=130
230 X2=220 :Y2=145
240 X3=120 :Y3=25
250 FOR I=0 TO 19
260 ALEA(I)=RND(1)
270 NEXT
290 RETURN :REM -----
300 REM FRACTALE (recursive)
310 IF PROF=0 THEN GOSUB 400 :RETURN
320 P1L=P1L+1
330 GOSUB 500 :REM empile et segmente
340 X1=X1(P1L):Y1=Y1(P1L):X2=X4(P1L):
Y2=Y4(P1L):X3=X6(P1L):Y3=Y6(P1L)
345 PROF=PROF(P1L)-1 :GOSUB 300
350 X1=X2(P1L):Y1=Y2(P1L):X2=X5(P1L):
Y2=Y5(P1L):X3=X4(P1L):Y3=Y4(P1L)
355 PROF=PROF(P1L)-1 :GOSUB 300
360 X1=X3(P1L):Y1=Y3(P1L):X2=X6(P1L):
Y2=Y6(P1L):X3=X5(P1L):Y3=Y5(P1L)
365 PROF=PROF(P1L)-1 :GOSUB 300
370 X1=X4(P1L):Y1=Y4(P1L):X2=X5(P1L):
Y2=Y5(P1L):X3=X6(P1L):Y3=Y6(P1L)
375 PROF=PROF(P1L)-1 :GOSUB 300
380 P1L=P1L-1
390 RETURN :REM -----
400 REM trace du triangle (/ordinat.)
410 IF X2<>X1 OR Y2<>Y1 THEN CURSET X
1,Y1,1 :DRAW X2-X1,Y2-Y1,1
420 IF X3<>X2 OR Y3<>Y2 THEN CURSET X
2,Y2,1 :DRAW X3-X2,Y3-Y2,1
430 IF X1<>X3 OR Y1<>Y3 THEN CURSET X
3,Y3,1 :DRAW X1-X3,Y1-Y3,1
490 RETURN :REM -----
500 REM empile et segmente
510 PROF(P1L)=PROF
520 X1(P1L)=X1:Y1(P1L)=Y1
530 X2(P1L)=X2:Y2(P1L)=Y2
540 X3(P1L)=X3:Y3(P1L)=Y3
550 REM milieux des 3 cotés
560 XA=X1:YA=Y1:XB=X2:YB=Y2:GOSUB 600
565 X4(P1L)=XM:Y4(P1L)=YM
570 XA=X2:YA=Y2:XB=X3:YB=Y3:GOSUB 600
575 X5(P1L)=XM:Y5(P1L)=YM
580 XA=X3:YA=Y3:XB=X1:YB=Y1:GOSUB 600
585 X6(P1L)=XM:Y6(P1L)=YM
590 RETURN :REM -----
600 REM segmentation
610 L=ABS(XA-XB)+ABS(YA-YB)
620 K=(XA+XB)*2+YA+YB
630 DIST=(ALEA(K-INT(K/20)*20)-.5)/5
640 XM=INT(DIST*L+(XA+XB)/2)
650 K=(YA+YB)*2+XA+XB
660 DIST=(ALEA(K-INT(K/20)*20)-.5)/5
670 YM=INT(DIST*L+(YA+YB)/2)
690 RETURN :REM -----
700 REM efface ecran (/ordinateur)
710 CLS :PAPER 0:INK 7: HIRES
790 RETURN :REM -----
800 REM fin un ordre
810 PRINT CHR$(7);
820 PRINT "Profondeur de 0 a 6 ou 'F'
in ?";
830 GET PROF$ :IF PROF$<"0" OR PROF$>
"6" AND PROF$<>"F" THEN 830
840 DESSIN=(PROF$<>"F")
850 IF DESSIN THEN GOSUB 700 :PRINT P
ROF$
890 RETURN :REM -----
900 PRINT :PRINT "Fin du programme";
990 END :REM ==========
```

UNE MANIÈRE D'ÊTRE RÉCURSIF

déroulement des sous-programmes ou des fonctions. Prenons par exemple, une fonction « milieu » qui coupe un segment en deux, coupe alors chacun des segments obtenus, et continue ainsi. Cette fonction doit pouvoir conserver les extrémités du segment et le nombre NBRE de fois qu'elle s'appelle elle-même. NBRE détermine donc le nombre de découpages et diminue à chaque appel.

Dès le début, prévoir la sortie

Deuxième règle à retenir : il faut trouver, dès le début, le cas qui permet une action directe, sans faire appel au sous-programme. Dans notre exemple, cela arrive si l'on ne coupe plus le segment, si NBRE = 0 (ligne 110) :

```
100 REM X2 "milieu" de X1, X3
110 IF NBRE = 0 THEN GOSUB 200 :
    RETURN
140 X2 = (X1 + X3)/2
150 X1 = X1 :X3 = X2 :NBRE = NBRE - 1
    :GOSUB 100
160 X1 = X2 :X3 = X3 :NBRE = NBRE - 1
    :GOSUB 100
190 RETURN
200 REM tracé graphique
... ...
```

Les variables X1, X2, X3 ou NBRE sont modifiées par chaque appel de GOSUB, et n'auront donc plus de sens au retour, pour la suite. Il faut alors ajouter une séquence de conservation des valeurs pour chaque GOSUB 100.

D'un Basic à l'autre

Ce programme est écrit pour Oric-1. Sur l'Atmos, les tests sur les coordonnées avant CURSET et DRAW ne sont pas nécessaires.

Pour l'Apple, par exemple, il faut remplacer les lignes 410, 420, 430 et 710 par :

```
410   HPLOT X1,Y1 TO X2,Y2 TO
      X3,Y3 TO X1,Y1
710   HGR : HOME : VTAB 22 :
      HCOLOR = 7
```

C'est le procédé d'empilement :

```
100 REM "milieu" de X1,X3
110 IF NBRE = 0 THEN GOSUB 200
    :RETURN
120 P1L = P1L + 1
130 X1(P1L) = X1 :X3(P1L) = X3
    :NBRE(P1L) = NBRE
140 X2(P1L) = (X1 + X3)/2
150 X1 = X1(P1L) :X3 = X2(P1L)
    :NBRE = NBRE(P1L) - 1 :GOSUB
    100
160 X1 = X2(P1L) :X3 = X3(P1L)
    :NBRE = NBRE(P1L) - 1 :GOSUB
    100
170 P1L = P1L - 1
190 RETURN
```

Une pile est une mémoire fonctionnant d'après le principe *dernier entré-premier sorti :* quand un élément entre dans la pile, il prend la place du « sommet » de pile (P1L, P-un-L), immédiatement après les éléments déjà entrés. L'élément sortant est le sommet de pile.

Pour illustrer la technique récursive, nous allons étudier la fonction FRACTALE utilisée par exemple dans le dessin de relief montagneux, notamment pour des décors de films.

Le principe topographique de cette fonction consiste à approcher n'importe quelle forme (une montagne, par exemple) par un assemblage de formes géométriques (des triangles). La précision est directement déterminée par le nombre de découpages, c'est-à-dire la profondeur.

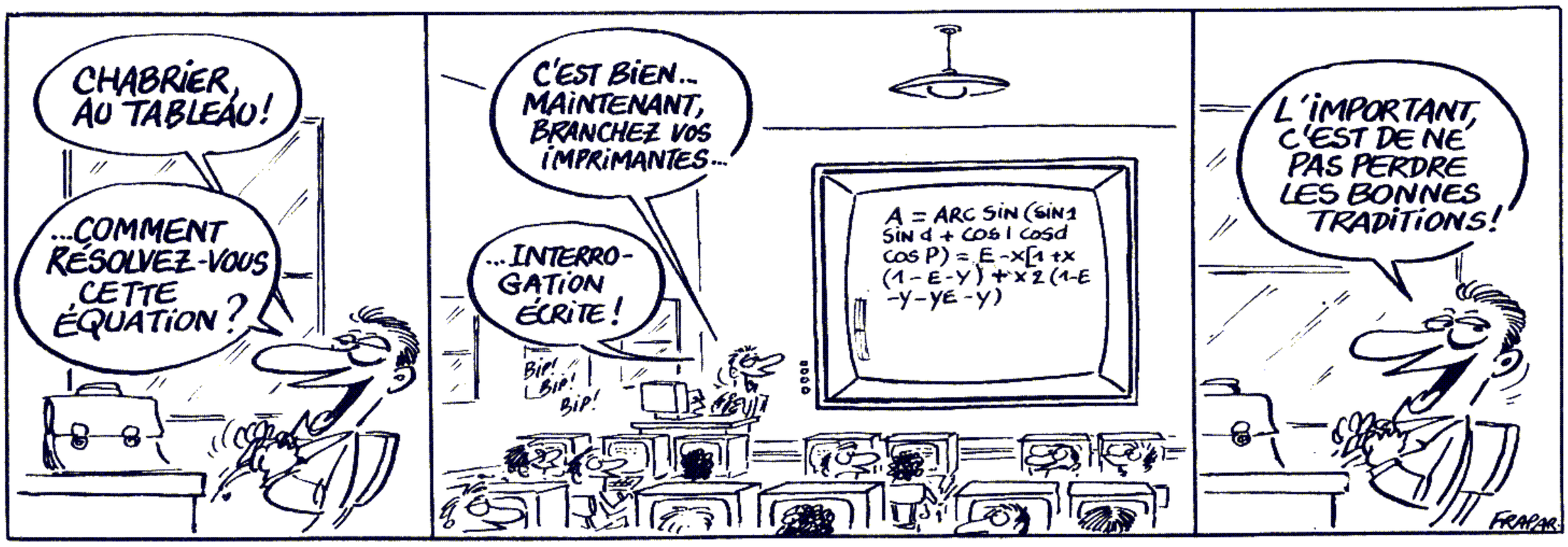
A l'inverse, un assemblage de triangles recrée une montagne avec impression de relief. Le découpage se fait par le fractionnement aléatoire d'un triangle en plusieurs autres, chacun d'eux étant à nouveau découpé, etc.

Dans le programme, la fonction récursive (GOSUB 300) contient sa propre structure, celle de s'appeler elle-même un nombre variable de fois, suivant la valeur de la profondeur (PROF$).

Les valeurs X et Y du début du dessin déterminent la dimension de la montagne. Le tableau ALEA contient des valeurs données au hasard et utilisées pour le calcul de la segmentation. P1L est le sommet de la pile. Dans la fonction empilement, chaque côté d'un triangle est segmenté. On obtient ainsi quatre nouveaux triangles qui vont donc appeler quatre fois la fonction FRACTALE. La segmentation se calcule avec un facteur de distorsion aléatoire (+ ou − 0.1).

Le programme présenté est écrit sur Oric, mais il est facilement adaptable à d'autres machines. Quant à la fonction FRACTALE, elle n'est qu'un des nombreux exemples utilisant la récursivité.

Max HAGENBURGER



PASCAL EST CONTAGIEUX : LE BASIC SE STRUCTURE

LE Basic structuré : on en parle de plus en plus comme d'un langage d'avenir. Voyons sur quelques exemples concrets les principaux avantages qu'apportent les versions structurées du « vieux » Basic.

L'apparition des machines MSX munies du Basic standard Microsoft amène l'utilisateur de micro-ordinateurs à se demander si le langage Basic ne va pas ainsi se trouver figé pour longtemps dans une version somme toute proche de celle des années 70. Dans le même temps, on commence à voir apparaître sur quelques machines des Basic plus puissants : les Basic dits « structurés ».

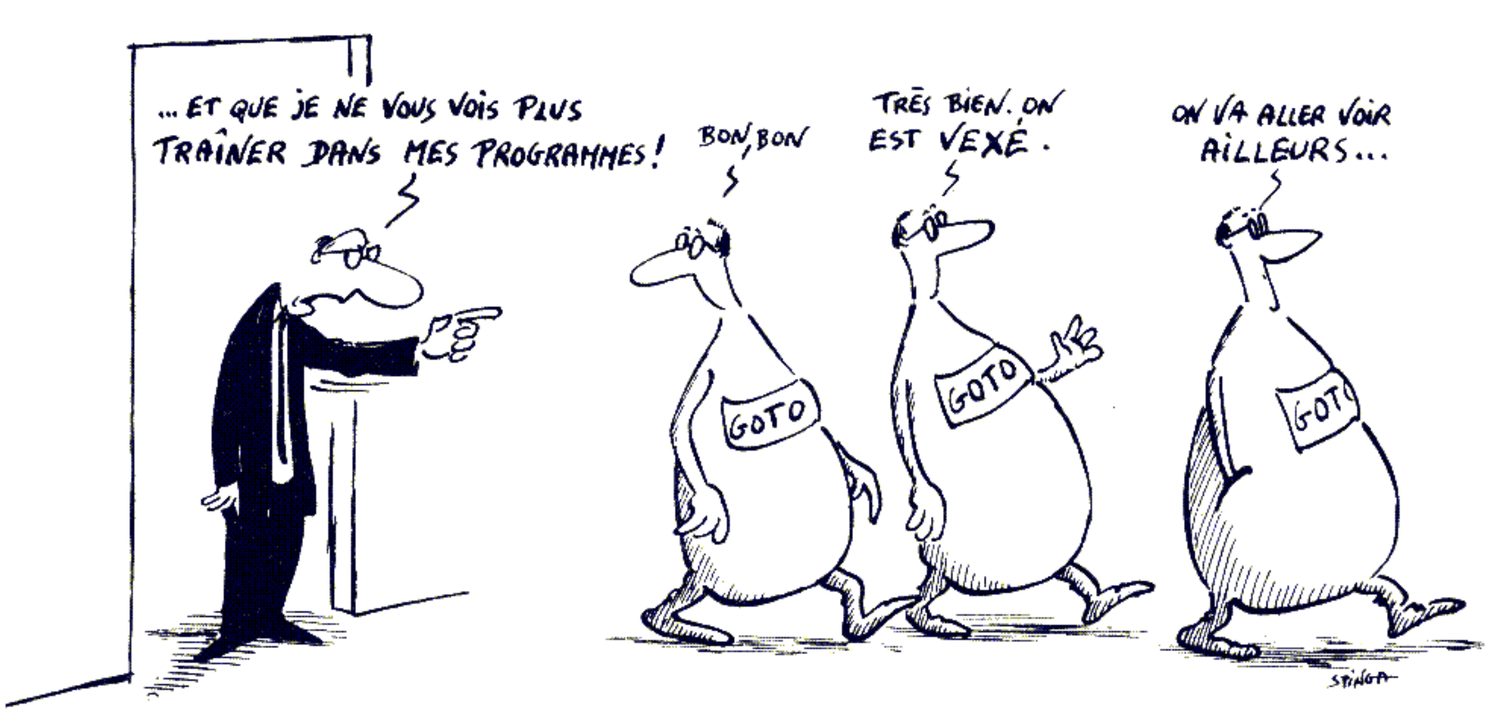
Rien que des avantages

Première constatation : qui peut le plus peut le moins. Dans l'ensemble, le Basic structuré reste compatible avec le Basic standard. Autrement dit, il est presque toujours possible de faire fonctionner avec un Basic structuré un programme écrit en Basic standard. Reste donc le « plus ». Nous allons voir justement quelques-uns des avantages qu'un Basic structuré apporte à l'utilisateur.

Qui n'a pas, un jour, recopié un programme paru dans une revue en prenant bien soin de n'y rien changer, et cela de peur de le voir mal fonctionner ? Cette crainte, le plus souvent justifiée, est due au fait que le mode de fonctionnement du programme est loin d'être évident : une lecture, même très attentive, de la liste ne suffit pas. Il faut, pour bien appréhender le fonctionnement du programme, plusieurs heures de travail assidu. Mais d'où peut provenir cette difficulté de compréhension ? On évoquera, entre autres raisons :

- l'absence de documentation ; par manque de place ou de temps, on a publié un programme sans REM ;
- l'absence d'organigramme (mais dès qu'un programme devient long, qui le dessine encore ?) ;
- la présence de GOSUB conduisant à des sous-programmes dont on ne connaît pas la fonction, ce qui rejoint le premier point ;
- plus grave : la présence de GOTO dont le débutant à trop tendance à user ! Cet ordre déroute le flux normal du programme, le rend plus difficile à comprendre. A une ligne donnée (surtout si plusieurs GOTO y conduisent), on peut très bien ne plus savoir du tout quel est l'état des variables, où elles ont été initialisées, modifiées, etc. ;
- enfin, il y a les astuces de programmation qu'on ne doit employer qu'à bon escient : si elles permettent d'écrire un programme plus court, plus rapide à l'exécution, elles nuisent à sa clarté, le rendent difficile (ou impossible) à lire à tout autre que l'auteur, et même à celui-ci quelques mois plus tard !

Ce dernier point pose le problème de la maintenance des programmes, problème assez souvent mineur pour l'amateur, mais d'une importance cruciale pour les entreprises ou les administrations susceptibles de changer de matériel... ou de programmeurs. Il a été en partie résolu par l'apparition de la



PASCAL EST CONTAGIEUX : LE BASIC SE STRUCTURE

« programmation structurée » et du langage Pascal qui en fait grand usage.

Jusqu'à ces dernières années, l'amateur désirant programmer « mieux » avait évidemment la solution de se mettre au Pascal, quand ce langage existait sur son ordinateur. La situation est en train de changer avec l'apparition des Basic structurés qui, comme leur nom l'indique, permettent la programmation structurée, mais n'en sont pas moins des langages interprétés et, par conséquent, plus souple d'emploi que Pascal (généralement compilé).

L'essentiel : les boucles

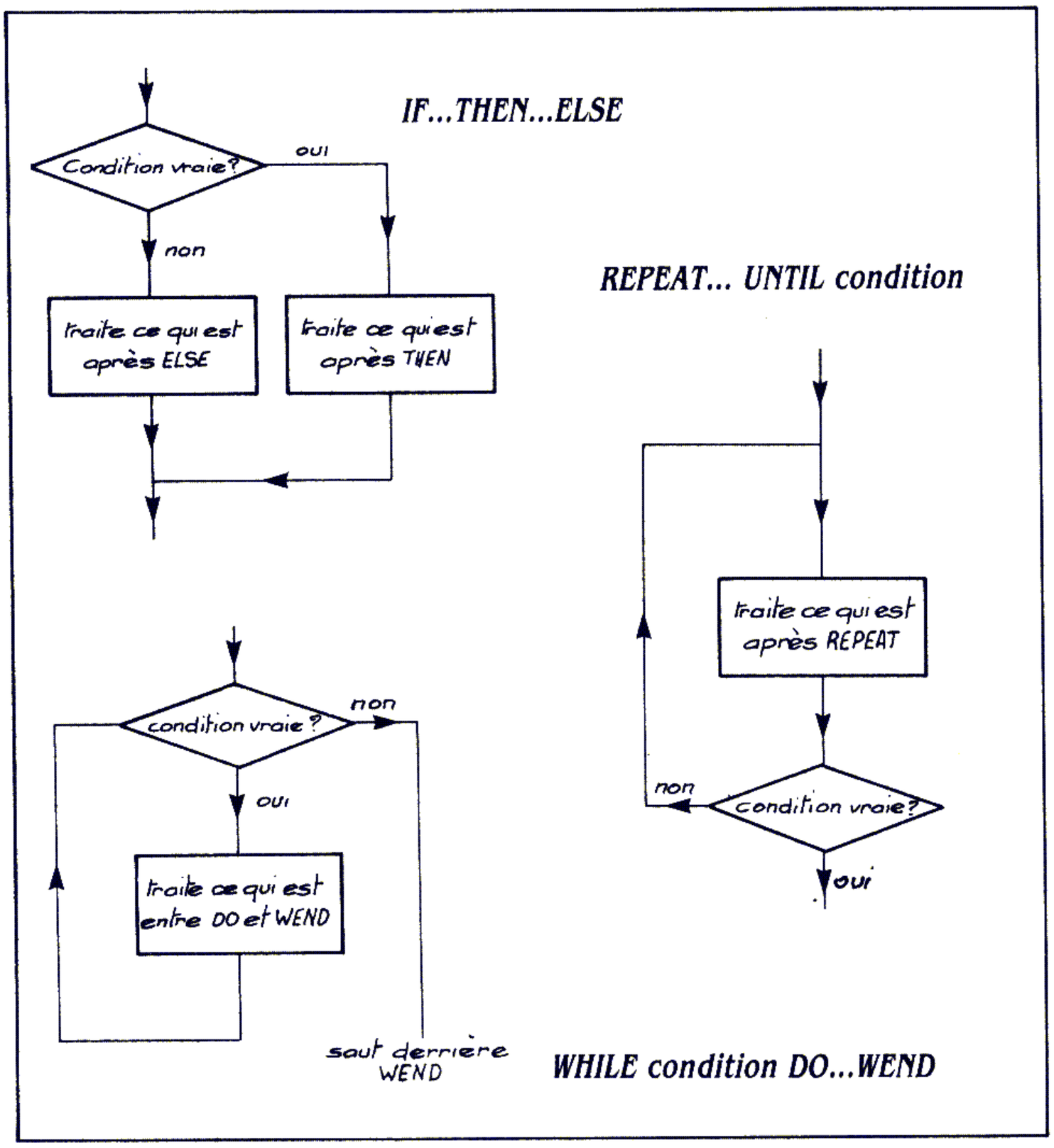
Dans ces Basic, le programme sera constitué d'une suite d'appels à des modules, chaque module ayant un seul point d'entrée et un seul point de sortie (ce qui facilite énormément la compréhension du programme). Avant de voir ce que sont ces modules, disons qu'ils sont construits exclusivement avec des instructions séquentielles, comme en Basic standard, avec les structures de test IF... THEN... ELSE, et surtout avec les boucles (structures « itératives » pour faire savant) : REPEAT... UNTIL condition et WHILE condition DO... WEND.

Les organigrammes correspondant à ces trois structures ont été reproduits ci-contre. On remarquera en passant que la classique boucle FOR...NEXT (qui existe aussi en Basic structuré) n'est qu'un cas particulier de la boucle REPEAT... UNTIL. Selon leur degré de complexité, les Basic structurés possèdent ou non d'autres structures facilitant la programmation, comme les tests multiples (IF... ELSIF... ELSIF... ELSE... ENDIF) ou la structure « cas » du Pascal : CASE OF... WHEN... WHEN... OTHERWISE... ENDCASE.

A titre d'exemples, on trouvera (listes nos 1 et 2) deux versions d'un court programme écrit dans le même Basic. Il s'agit d'un programme très classique dans lequel on doit deviner le nombre tenu secret par l'ordinateur. Seules ins-

Liste n° 1
Le jeu du nombre secret en Basic standard

```
10 CLS:SUP=99:INF=0
20 NOMBRE=INT(100*RND(1))
30 INPUT"Votre choix:";X
40 IF X<=NOMBRE GOTO 90
50 PRINT"Trop grand !"
60 SUP=X MIN SUP
70 PRINT"Entre ";INF;" et ";SUP
80 GOTO 30
90 IF X=NOMBRE GOTO 140
100 PRINT"Trop petit !"
110 INF=X MAX INF
120 PRINT"Entre ";INF;" et ";SUP
130 GOTO 30
140 PRINT"Bravo !"
150 PRINT"Voulez vous rejouer (O/N)?"
160 INKEY R$
170 IF R$<>"O" AND R$<>"N" GOTO 160
180 IF R$="O" GOTO 10
190 END
```



Liste n° 2
Même jeu, mais écrit en Basic structuré : plus de GOTO ni de GOSUB

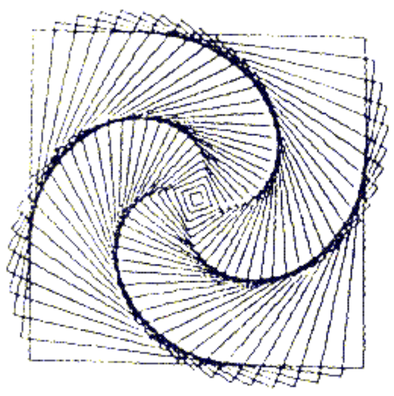
```
10 REPEAT
20 :   CLS:SUP=99:INF=0
30 :   NOMBRE=INT(100*RND(1))
40 :   REPEAT
50 :     INPUT "Votre choix:";X
60 :     IF X>NOMBRE DO
70 :       PRINT"Trop grand !"
80 :       SUP=X MIN SUP
90 :       PRINT"Entre ";INF;" et ";SUP
100 :    ELSIF X<NOMBRE DO
110 :      PRINT"Trop petit !"
120 :      INF=X MAX INF
130 :      PRINT"Entre ";INF;" et ";SUP
140 :    ENDIF
150 : UNTIL X=NOMBRE
160 : PRINT"Bravo !"
170 : PRINT"Voulez vous rejouer (O/N)?"
180 : REPEAT
190 :   INKEY R$
200 : UNTIL R$="O" OR R$="N"
210 UNTIL R$="N"
220 END
```

tructions qui ne soient pas très courantes : A MAX B donne le maximum d'A et B, A MIN B retourne le minimum. A part ces deux exceptions, la première version ne comporte que des instructions standard.

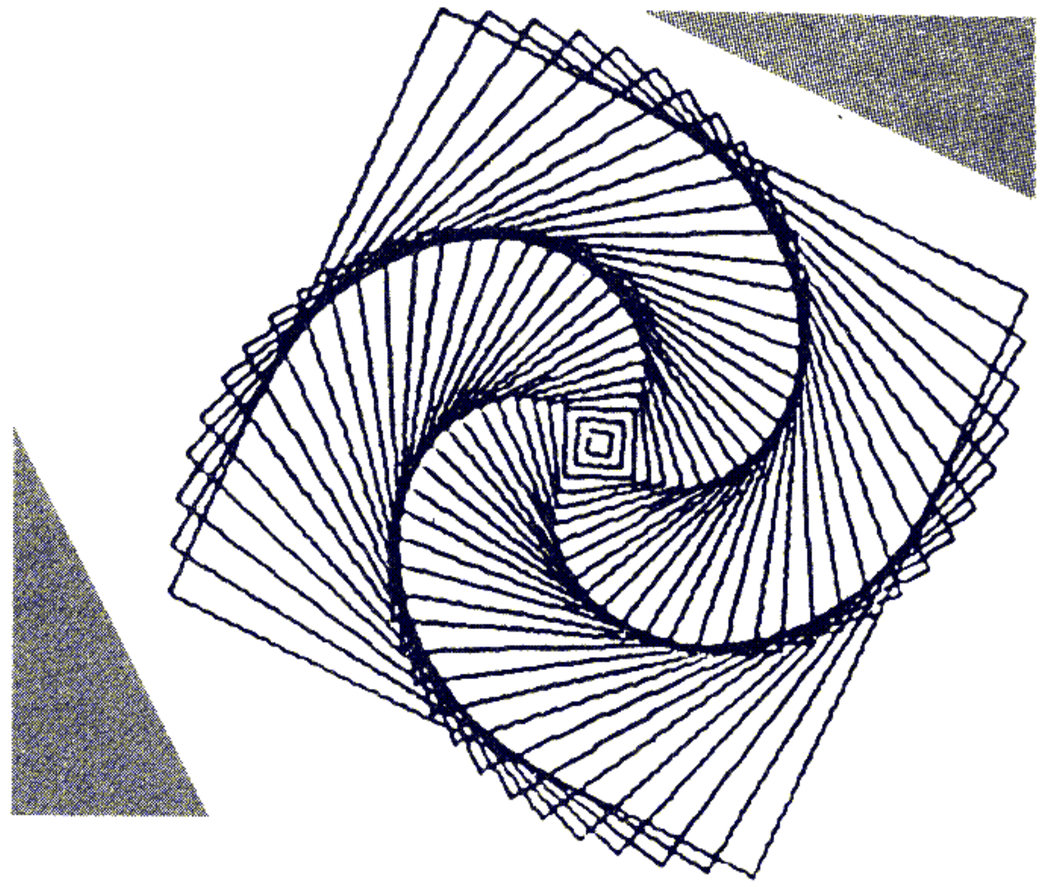
Dans la seconde version, qui est structurée, on notera l'absence de tout branchement par GOTO ou GOSUB. L'indentation des lignes, d'autre part, fait ressortir l'organisation du programme.

Enfin, la construction modulaire est assurée par la présence des procédures et des fonctions qui généralisent respectivement les GOSUB et les DEF FN du Basic standard. On peut leur donner un nom (PROC "Trace le graphe" est bien plus parlant que GOSUB 1000). A titre d'exemple, on a reproduit dans l'encadré ci-contre le début d'un programme traçant une courbe définie par ses équations paramétriques X(T) et Y(T).

Liste n° 3
Un court programme graphique et la courbe qu'il trace



```
10 MODE GR
20 FOR I=1 TO 30
30 : PROC"Carre"(240,-240,I/10,1+5*I)
40 NEXT I
50 MODE TN
60 END
70 REM ********************************
80 DEF PROC"Carre"(X,Y,A,D)
90 LOCAL DX,DY
100 DX=D*COS(A):DY=D*SIN(A)
110 MOVE X+DX,Y+DY
120 PROC"Ligne"(-DY, DX)
130 PROC"Ligne"(-DX,-DY)
140 PROC"Ligne"( DY,-DX)
150 PROC"Ligne"( DX, DY)
160 ENDPROC
170 REM ********************************
180 DEF PROC"Ligne"(A,B)
190 LINE X+A,Y+B
200 ENDPROC
210 REM ********************************
```



Même sans explications supplémentaires, le caractère modulaire du programme et les grandes lignes de son fonctionnement apparaissent clairement dès le début de la liste. Mais les procédures et les fonctions ont aussi un énorme avantage sur les GOSUB et sur

```
 10 PROC"Initialisation"
 20 REPEAT
 30 :     PROC"Titre"
 40 :     PROC"Entre fonctions"
 50 :     REPEAT
 60 :       PROC"Titre"
 70 :       PROC"Affiche fonc-
            tions"
 80 :       PROC"Entre bornes"
 90 :       PROC"Calculs"
100 :       PROC"Dessine"
110 :       PROC"Indications"
120 :     UNTIL FN"Encore?"=
          "F"
130 UNTIL 0
```

FN : elles permettent le passage de paramètres entre le programme principal et le sous-programme (voir liste n° 3). La procédure définie à la ligne 80 permet de tracer un carré centré en (X,Y) et ayant l'une de ses diagonales (de demi-longueur D) faisant l'angle A avec l'axe des X. Lors de l'appel de cette procédure, à la ligne 30, X prendra automatiquement la valeur 240, Y prendra la valeur −240, A la valeur I/10 et D la valeur 1 + 5*I. Comme A et D augmentent légèrement avec I, le carré va tourner sur lui-même tout en s'agrandissant.

L'intérêt du passage de paramètres est qu'il a été possible de mettre au point la procédure indépendamment du reste : lors de l'écriture du programme principal (lignes 10 à 60), on a seulement dû se souvenir de l'ordre des paramètres à passer à la procédure. Avec un Basic standard, il aurait fallu aussi tenir

PASCAL EST CONTAGIEUX : LE BASIC SE STRUCTURE

compte du nom donné aux paramètres et écrire la ligne 30 comme suit :

```
30 X=240 : Y=-240 : A=I/10 :
   D=1+5*I : GOSUB 80
```

De plus, X, Y, A et D sont des variables « locales » à la procédure, c'est-à-dire qu'elles sont considérées par le Basic structuré comme différentes d'autres variables qui seraient elles aussi nommées X, Y, A et D mais qui seraient utilisées en dehors de la procédure.

Pour en avoir la démonstration, il suffit de rajouter au programme les trois lignes :

```
25 X=500
35 PRINT X
95 PRINT X
```

La ligne 95 imprimera 240 (valeur de X dans la procédure) alors que la ligne 35 donnera 500 (valeur de X en dehors de la procédure, définie ligne 25). Il y a donc sauvegarde de la valeur 500 de X lors de l'entrée dans la procédure et restitution lors du END PROC de la ligne 160. Un autre exemple de cette notion se trouve à la ligne 180 où l'on voit que A est de nouveau utilisé comme paramètre. La variable A, angle d'inclinaison du carré, utilisé dans PROC "Carré" est sauvée lors de l'entrée dans PROC "Ligne" et restituée lors du END PROC de la ligne 200. Enfin, pour des variables utilisées dans une procédure (ou une fonction) et qui ne seraient pas des paramètres, il sera toujours possible de les rendre locales grâce à l'instruction LOCAL (ligne 90).

Chaque procédure fait un tout

C'est cette notion de variables locales qui permet, en Basic structuré, de mettre au point les procédures et les fonctions indépendamment du programme principal, sans risque de « télescopage » entre les variables de ce programme et celles qui sont utilisées dans la procédure ou la fonction.

Il devient ainsi possible de se constituer une bibliothèque de procédures ou de fonctions (ré-)utilisables dans tout programme. Cela dit, certains Basic structurés sont dotés de ce que l'on appelle le « passage de paramètres par référence » qui permet la sortie de plusieurs résultats d'une procédure ou d'une fonction, mais nous n'entrerons pas dans les détails.

Pour terminer, la liste 4 nous montre une fonction calculant une intégrale par la méthode de Simpson. L'exemple est tiré (et adapté sur Sharp MZ 700) de l'*Introduction à la programmation systématique* par Niklaus Wirth (le père du langage Pascal) publié chez Masson. La fonction calcule l'intégrale de F$, écrite sous forme de chaîne de caractères et calculée par EVAL entre les bornes A et B. On remarquera que toutes les variables sont locales et, par conséquent, que la fonction se comporte par rapport au programme qui l'appelle comme une boîte bien fermée dans laquelle on entre A, B, F$ et qui retourne l'intégrale.

Voilà exposées les principales qualités des Basic structurés. On comprend les énormes avantages qu'ils ont sur le Basic standard. Bien entendu, presque tous les programmes peuvent être écrits dans n'importe quel langage, et l'acheteur d'une machine destinée à des jeux ou à l'utilisation de programmes tout faits n'a pas à se poser la question du type de Basic à utiliser : une version standard sera tout à fait adaptée à ses besoins. Mais, à mon avis, celui qui veut faire de sa machine un véritable outil d'apprentissage de la programmation et, par la suite, de développement intellectuel, a tout intérêt à choisir le Basic structuré.

Le seul véritable problème est, à l'heure actuelle, son manque de diffusion et le nombre restreint de machines permettant de l'utiliser. J'espère, quant à moi, que les constructeurs et les concepteurs de logiciels se rendent compte qu'il n'est pas plus difficile de créer un Basic structuré qu'un Basic standard. Et je ne parle pas en l'air : j'ai moi-même écrit plusieurs interpréteurs Basic structurés sur des ordinateurs Sharp, en modifiant des Basic standard. Il est grand temps de s'apercevoir qu'il y a une demande pour ce genre de langage. Le Basic des années 70 a fait son temps.

Liste n° 4
Calcul d'une intégrale par la méthode de Simpson

```
10 INPUT "Fonction:";Z$
20 REPEAT
30 :   INPUT"a=";X$:X=EVAL(X$)
40 :   INPUT"b=";Y$:Y=EVAL(Y$)
50 UNTIL X<>Y
60 PRINT"Integrale:";FN"Simpson"(X,Y,Z$)
70 END
80 REM ********************************
100 DEF FN"Simpson"(A,B,F$)
110 LOCAL N,H,X,S,S1,S2,S4,SS,I
120 N=2:H=(B-A)/2
130 X=A:S1=EVAL(F$)
140 X=B:S1=H*(S1+EVAL(F$)):S2=0
150 X=A+H:S4=4*H*EVAL(F$):S=S1+S4
160 REPEAT
170 : SS=S:N=2*N:H=H/2
180 : S1=S1/2:S2=S2/2+S4/4:S4=0
190 : FOR I=1 TO N STEP 2
200 :   X=A+I*H:S4=S4+EVAL(F$)
210 : NEXT I
220 : S4=4*H*S4:S=S1+S2+S4
230 UNTIL ABS(S-SS)<.0000001
240 RESULT=S/3
```

LIST · PAGE 28

Bernard KOKANOSKY

LE PC-1500 JOUE A LA PAIX DES ÉTOILES

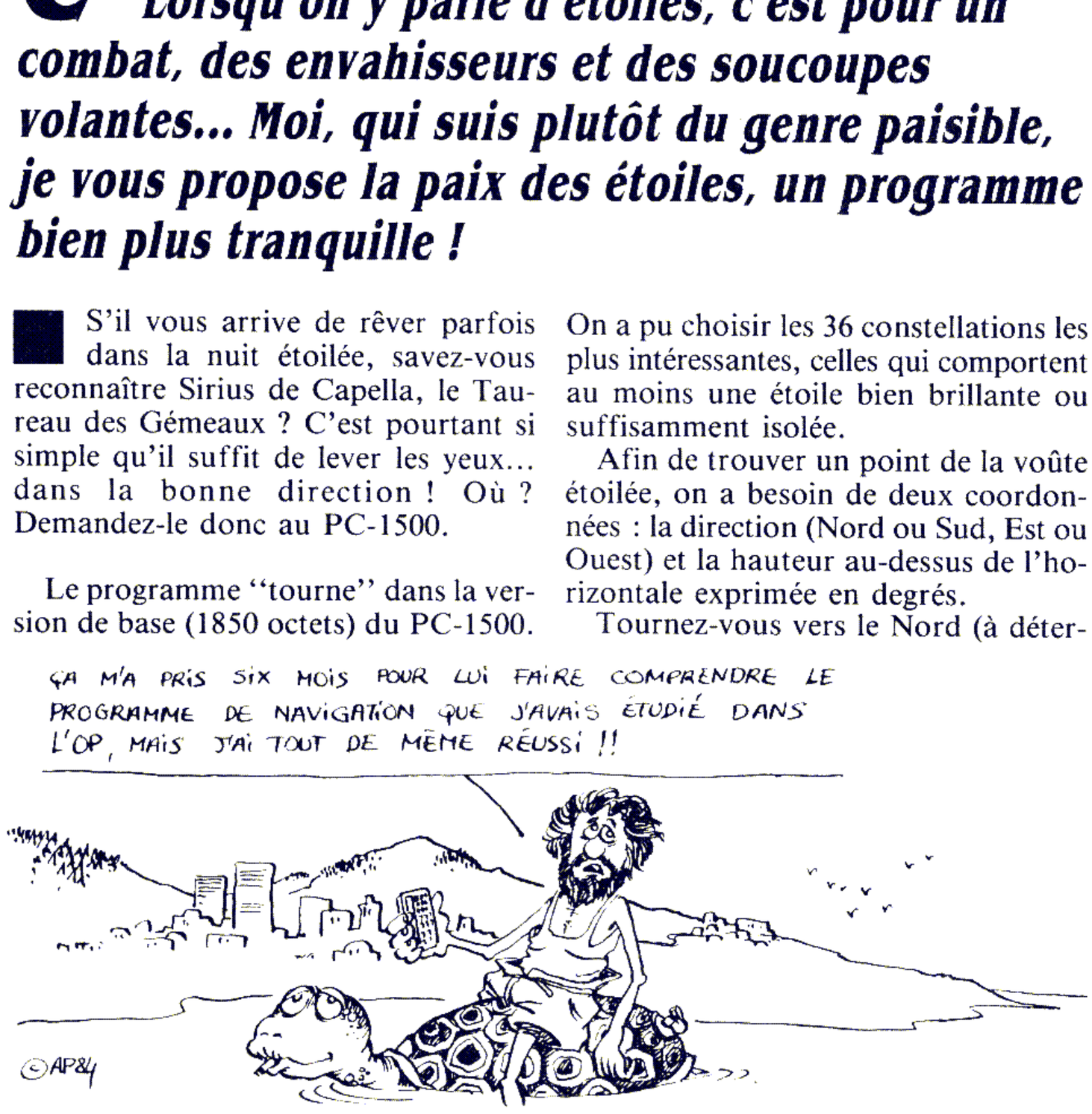
CINÉMA ou ordinateur : même combat ? Lorsqu'on y parle d'étoiles, c'est pour un combat, des envahisseurs et des soucoupes volantes... Moi, qui suis plutôt du genre paisible, je vous propose la paix des étoiles, un programme bien plus tranquille !

S'il vous arrive de rêver parfois dans la nuit étoilée, savez-vous reconnaître Sirius de Capella, le Taureau des Gémeaux ? C'est pourtant si simple qu'il suffit de lever les yeux... dans la bonne direction ! Où ? Demandez-le donc au PC-1500.

Le programme "tourne" dans la version de base (1850 octets) du PC-1500. On a pu choisir les 36 constellations les plus intéressantes, celles qui comportent au moins une étoile bien brillante ou suffisamment isolée.

Afin de trouver un point de la voûte étoilée, on a besoin de deux coordonnées : la direction (Nord ou Sud, Est ou Ouest) et la hauteur au-dessus de l'horizontale exprimée en degrés.

Tournez-vous vers le Nord (à déterminer avec une boussole ou, approximativement, sachant que le soleil se lève à l'Est et se couche à l'Ouest), on dira que vous faites face à "l'azimut 0°" (ou 360°, ce qui est la même chose). Si vous tournez sur vous-même de, disons 30 degrés vers la droite (soit un tiers d'un quart de tour...), vous faites face à "l'azimut 30°". Ainsi, le Nord correspond au 0 (ou 360), l'Est au 90, le Sud au 180 et, enfin, l'Ouest au 270.

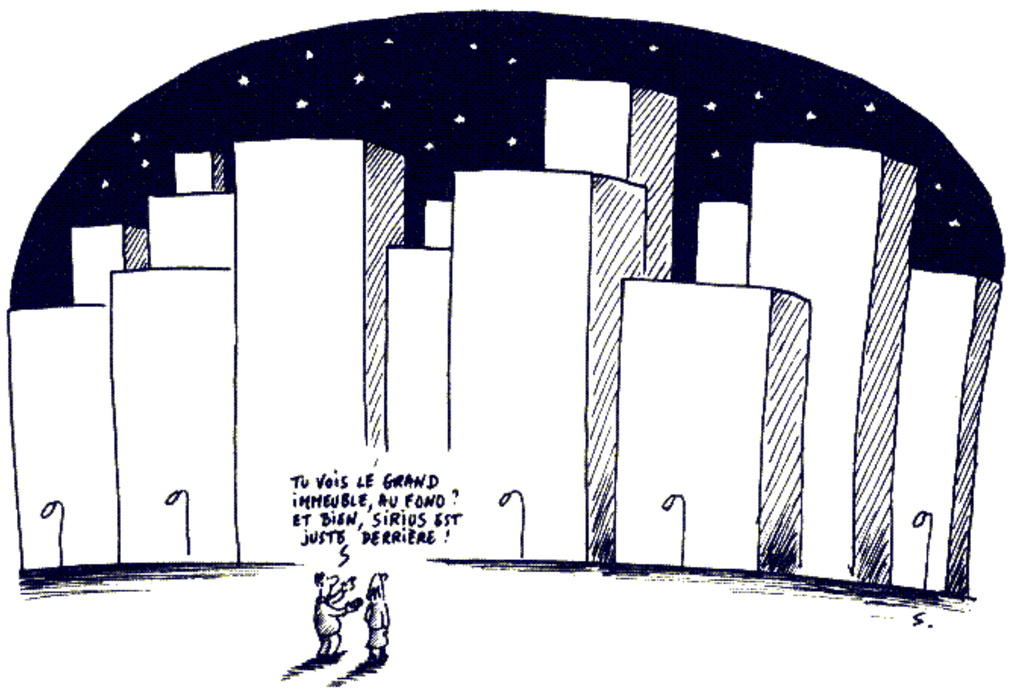


Où donner de la tête ?

Ainsi orienté dans une direction donnée, levez le nez de N degrés au-dessus de l'horizontale : la hauteur du point du ciel que vous fixez est de N degrés. L'horizontale correspond à 0° et la verticale à 90° (le zénith est au-dessus de votre tête).

Munis d'une bonne paire d'yeux et d'un cou suffisamment souple, vous voici en mesure de regarder dans le ciel une étoile dont vous connaissez les deux coordonnées : azimut (direction) et hauteur.

Une dernière chose à préciser : où êtes-vous ? Eh oui, l'évidence s'impose : dans l'hémisphère Nord ou dans l'hémisphère Sud, en URSS ou aux USA, à un instant donné, les étoiles visi-



La paix des étoiles
Programme pour PC-1500
Auteur Antoine
Copyright *LIST* et l'auteur

```
10:"S":CLEAR :
   CSIZE 2:COLOR
   1:LPRINT "TROU
   VEUR D/ETOILES
   ":LF 1:DIM Q$(
   0)*18:USING "#
   ###.##"
20:INPUT "LAT EST
   IMEE",L1:L=DEG
   L1
30:INPUT "N OU S?
   ",L$:IF (L$="N
   ")+(L$="S")<>1
   GOTO 30
40:INPUT "LONG. E
   STIMEE?",G1:G=
   DEG G1
50:INPUT "E OU O?
   ",G$:IF (G$="O
   ")+(G$="E")<>1
   GOTO 50
60:IF L$="S"LET L
   =-L
70:IF G$="O"LET G
   =-G
80:LPRINT "LATITU
   DE  ";L1;L$;"L
   ONGITUDE ";G1;
   G$:LF 2:INPUT
   "DECALAGE HORA
   IRE?",DO
90:T=TIME :M=INT
   (T/10000):J=
   INT (T/100)-10
   0*M:I=T-10000*
   M-100*J:I=DEG
   I-DO
100:K=INT ((30.6*M
   )-32.4):A=1984
   :IF INT (A/4)=
   A/4LET K=K+1
110:IF M<=2LET K=3
   1*M-31
120:H0=99.84:H1=0.
   9856164:H2=15.
   04107
130:FOR Z=1TO 36:
   RESTORE (300+Z
   ):READ Q$(0),V
   ,D
140:H=H0+H1*(K+J-1
   )+I*H2:IF H>36
   0LET H=H-360
150:H=H+V:B=H+G
160:Y=ASN (SIN D*
   SIN L+COS D*
   COS L*COS B)
170:S=ASN (COS D*
   COS Y*SIN B):R
   =(TAN D*COS L-
   SIN L*COS B)/
   SIN B
180:IF R<0AND S<0
   LET S=-S:GOTO
   210
190:IF R*S<0LET S=
   180+S:GOTO 210
200:S=360-S
210:IF Y>0LPRINT Q
   $(0):USING "##
   ##":LPRINT DMS
   Y;"  AU ";S
220:NEXT Z:BEEP 5.
   END
301:DATA "PEGASE/A
   LPHERATZ",358,
   29
302:DATA "CASSIOPE
   E/SCHEDIR",350
   ,56
303:DATA "BALEINE/
   DEN.KAITOS",34
   9,-18
304:DATA "FL.ERIDA
   N/ACHERNAR",33
   6,-57
305:DATA "BELIER/H
   AMAL",328,23
306:DATA "PERSEE/M
   IRFAK",309,50
307:DATA "TAUREAU/
   ALDEBARAN",291
   ,16
308:DATA "ORION/RI
   GEL",282,-8
309:DATA "COCHER/C
   APELLA",281,46
310:DATA "CARENE/C
   ANOPUS",264,-5
   3
311:DATA "GD-CHIEN
   /SIRIUS",259,1
   7
312:DATA "P.-CHIEN
   /PROCYON",245,
   5
313:DATA "GEMEAUX/
   POLLUX",244,28
314:DATA "VOILES/S
   UHAIL",223,-43
315:DATA "LION/REG
   ULUS",208,12
316:DATA "GDE-OURS
   E/DUBHE",194,6
   2
317:DATA "CROIX DU
   S/ACRUX",174,
   -63
318:DATA "VIERGE/S
   PICA",159,-11
319:DATA "BOUVIER/
   ARCTURUS",146,
   19
320:DATA "CENTAURE
   /RIGIL.KENT",1
   40,-61
321:DATA "PTE-OURS
   E/KOCHAB",137,
   74
322:DATA "COUR.BOR
   ./ALPHECCA",12
   6,27
323:DATA "SCORPION
   /ANTARES",112,
   -26
324:DATA "OPHIUCUS
   /RASALHAG.",96
   ,13
325:DATA "LYRE/VEG
   A",81,39
326:DATA "SAGITTAI
   RE/NUNKI",76,-
   26
327:DATA "AIGLE/AL
   TAIR",63,9
328:DATA "PAON/PEA
   COCK",54,-57
329:DATA "CYGNE/DE
   NEB",50,45
330:DATA "POIS.AUS
   /FOMALHAUT",16
   ,-29
331:DATA "PLEIADES
   ",304,23
332:DATA "VERSEAU"
   ,35,-4
333:DATA "CANCER",
   231,19
334:DATA "CAPRICOR
   NE",47,-16
335:DATA "POISSONS
   ",348,7
336:DATA "BALANCE"
   ,135,-12
1000:'Z"ON ERROR
   GOTO 1003:
   DIM A$(0)*18
   :WAIT 0
1001:READ A$(0),A
   ,B:LPRINT A$
   (0);A;B:GOTO
   1001
1003:BEEP 1:PAUSE
   :END
```

bles ne sont pas les mêmes. Aussi, doit-on connaître les coordonnées terrestres du lieu géographique où l'on se trouve. Vous trouverez dans un atlas géographique la longitude et la latitude du lieu qui vous concerne.

DEF S et le programme débute. Introduire à la demande du PC-1500 les latitude et longitude du lieu d'observation, préciser Nord, Sud, Est ou Ouest.

Du côté du temps, l'heure du programme est celle de Greenwich, le TU (temps universel) et pour l'obtenir à partir de l'heure fournie par la fonction TIME, il faut retrancher, ou ajouter (selon le lieu et l'heure légale en vigueur) un décalage horaire.

Le nez dans les étoiles

Par exemple, en France, en été, l'heure légale (celle de TIME) est supérieure de deux heures à celle de Greenwich (contre une seulement en hiver). Donc le décalage horaire est +2. A New York, en hiver, de décalage est −5 (5 heures de moins). On introduira ce décalage à la demande de l'ordinateur.

Et après avoir tant travaillé, la récompense est là : la liste des coordonnées des 36 constellations est imprimée, seulement pour celles qui sont visibles au lieu et à l'heure de l'observation (il n'est pas tenu compte d'éventuels arbres, montagnes ou buildings qui restreindraient le champ de vision...).

La liste fournie donne deux noms : ceux de la constellation et de son étoile la plus brillante. Ensuite, les coordonnées sont, dans l'ordre, la hauteur et l'azimut de l'étoile.

Le reste du programme, c'est à vous qu'il incombe de le réaliser : sortir, lever la tête dans la bonne direction et... contempler ces étoiles.

Si vous ne possédez pas d'imprimante, il convient de remplacer chaque LPRINT en PRINT et d'éliminer les instructions comme LF n, CSIZE n, COLOR n,... qui sont celles du CE-150. A la ligne 100, on donnera à A la valeur 1985 et on remplacera l'expression H0 = 99.8 par H0 = 100.59 en ligne 120.

Et si votre imagination vous permet de modifier le programme, peut-être un jour trouverez-vous Syrée, l'étoile des navigateurs facétieux... Comment ? Vous ne connaissez pas l'étoile Syrée ?

ANTOINE

LE BASIC DU GUÉPARD

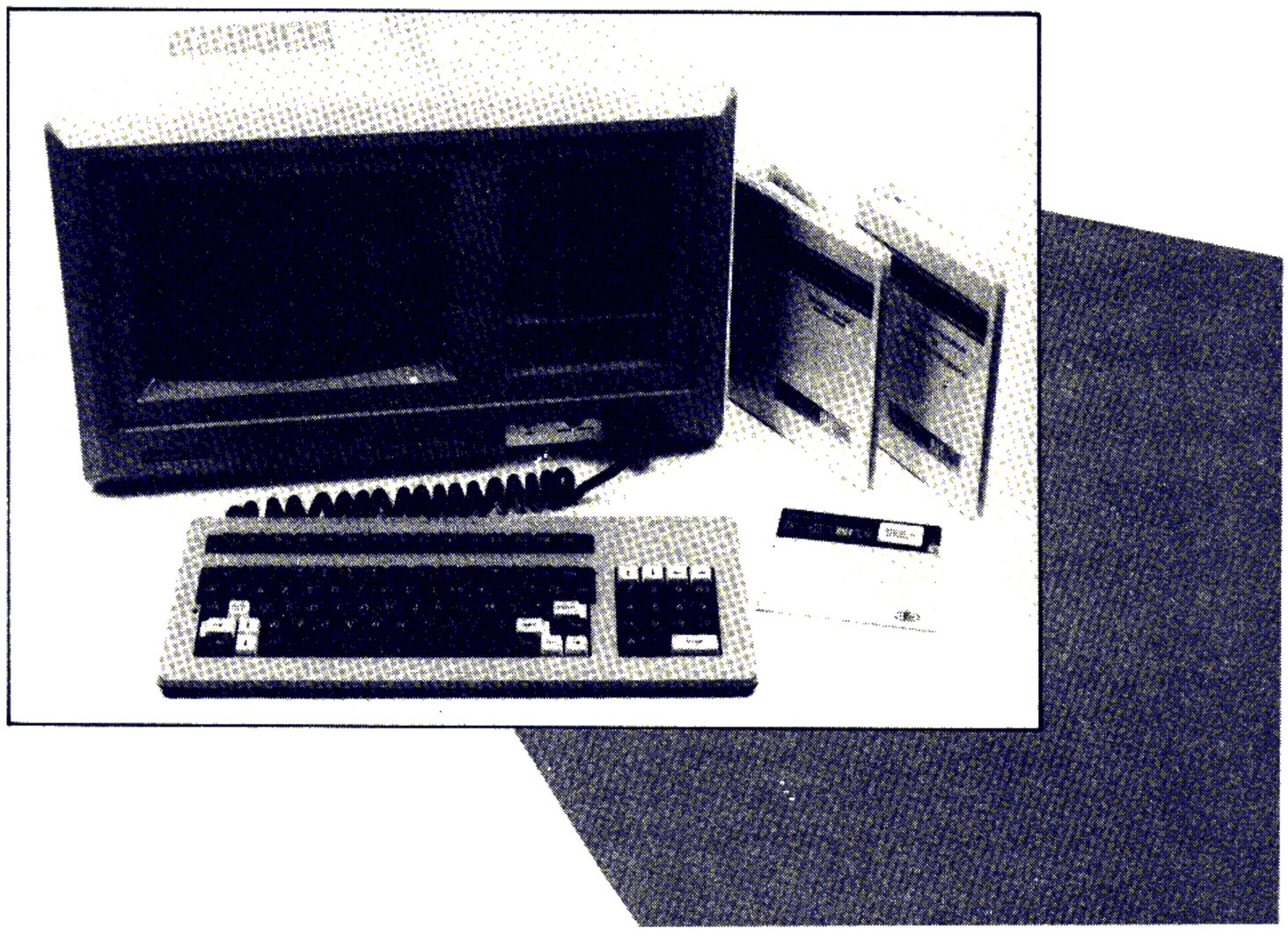
LES astuces du Guépard, ordinateur français, feraient rougir bien des machines étrangères : clavier mixte, fonctionnement autonome sous batterie, deux systèmes d'exploitation renommés... Et le Basic, malgré quelques archaïsmes, se montre bien séduisant.

L'allure massive du boîtier de l'unité centrale se justifie par l'intégration de deux lecteurs de disquette, d'un moniteur vert de 30 cm, des interfaces nécessaires. Il reste même de la place pour en ajouter d'autres. De plus, de lourds accus sont embarqués à bord. Ils assurent une autonomie complète de fonctionnement pendant près d'une heure. La fin des angoisses dans les zones à coupures fréquentes du secteur électrique. Conséquence de ce remplissage du boîtier : le volume et le poids s'en ressentent.

New Dos 80 et CP/M3

Le clavier contraste par sa faible épaisseur et son encombrement réduit. Mais il s'avère très complet, avec son pavé numérique, son double jeu de flèches de curseur et ses 15 touches de fonction (sans compter les 27 autres obtenues par ESC et une touche alphabétique). Encore une originalité : il est configurable en Azerty ou Qwerty, selon les besoins. Il suffit d'échanger les cabochons de touches et d'exécuter un petit programme de configuration. Cela ne prend qu'une ou deux minutes. Avantages : la lisibilité est meilleure qu'une gravure multiple et l'accès reste ouvert à d'autres dispositions (entre autres, le nouvel arrangement Dvorak pour lequel il suffirait de réaliser le petit programme d'exploitation en langage-machine).

Les lecteurs de disquette demi-épaisseur exploitent, selon les versions, 40 ou 80 pistes en double face. Cela donne au maximum 720 Ko par disquette. Une capacité très professionnelle. Et les SED (systèmes d'exploitation des disquettes) pour en tirer parti ne sont pas, eux non plus, des joujoux : il en existe deux qui me plaisent autant l'un que l'autre. Le premier donne du génie à cet ordinateur, c'est NewDos 80 d'Apparat. Le second apporte une bibliothèque de programmes fabuleuse : c'est CP/M 3 (aussi appelé CP/M+). Grâce à lui, j'ai pu jouer avec Dbase 2 et SuperCalc 2. La programmathèque disponible sous NewDos n'est pas ridicule pour autant puisque ce SED donne accès aux logiciels écrits pour TRS-80. La lecture des disquettes de ce dernier oblige cependant à définir les bonnes caractéristiques de PDRIVE (commande permettant de lire des disquet-



LE BASIC DU GUÉPARD

tes écrites dans des formats différents) et éventuellement à en réécrire le répertoire (WRDIRP).

Le Basic n'est disponible que sous NewDos. Les utilisateurs habitués à CP/M seront donc contraints de renoncer à leur SED préféré pour écrire leurs programmes. Contrairement à ce qui se passe sur le TRS modèle III, le SBasic du Guépard est entièrement chargé à partir de la disquette et il laisse 38200 octets de mémoire utilisateur. Son appel est effectué par la commande BASIC, suivie de deux paramètres facultatifs déterminant le nombre de zones tampons réservées pour les fichiers, et la zone mémoire à protéger pour les routines en langage-machine. Une commande Basic est également exécutable conjointement à l'appel de l'interpréteur.

Version élaborée du Basic

Le SBasic s'appuie sur des fondations Microsoft, gage de puissance et de standardisation. De fait, la base syntaxique et lexicale se révèle classique et complète. Le ELSE donne l'alternative à IF...THEN, on trouve les ON GOTO et ON GOSUB, DEF USR, VARPTR et autres PRINT USING et LINE INPUT qui caractérisent les versions élaborées du Basic. Et l'on remarque quelques signes de recherches visant à apporter du confort au programmeur. RND ne se contente pas de produire des nombres aléatoires entre 0 et 1, il sait aussi générer directement des nombres entiers. PAUSE paramétré en centièmes de seconde remplace avantageusement les boucles d'attente.

La pseudo-variable TIME$ n'a pas été oubliée pour l'exploitation directe de l'horloge interne. Pour la causerie en langage-machine, PEEK et POKE sont prêts à s'adresser au Z80 qui équipe le Guépard et INP et OUT se réservent le dialogue avec les ports. Pas d'oubli donc dans les instructions générales.

Un peu moins de raffinement, en revanche, dans le domaine des fonctions mathématiques. La trigonométrie ne se joue que sur des angles délivrés en radians et les logarithmes décimaux ne peuvent s'exprimer qu'à partir de leurs cousins népériens. La manipulation de nombres en hexadécimal ou octal a par contre été prévue. Pas de problème dans la précision des calculs puisque les trois grands types de variables numériques sont présents : entiers, simple ou double précision (16 chiffres significatifs).

Les définitions de type se font soit au coup par coup avec les suffixes %, ! et #, soit globalement avec DEF. Les identificateurs restent limités à deux caractères significatifs, ce qui n'exclut pas des noms vraiment parlants mais laisse moins de choix dans les désignations (par exemple, une variable Longueur est valide, mais Longitude désigne la même « case » de stockage).

Les chaînes comportent jusqu'à 255 caractères. Leurs manipulations s'opèrent avec les instructions habituelles du Basic Microsoft : LEFT$, RIGHT$, MID$, STRING$, STR$, VAL, INSTR et LEN. Et l'on retrouve la possibilité de contrôler l'espace qu'elles occupent en mémoire vive avec FRE(x$), FRE(x) indiquant, lui, la place utilisée par le programme.

La gestion des fichiers s'opère comme dans le Basic Microsoft, pour le séquentiel et l'accès direct classique. Cinq autres types de fichiers à accès direct sont également réalisables donnant une grande souplesse dans la réalisation des programmes. Le type "MU" permet les enregistrements de longueur variable et l'accès direct grâce à des marques de séparation. Les types "MF" et "FF" réalisent des enregistrements de longueur fixe pouvant aller jusqu'à 4095 caractères, le premier avec marqueur de repérage, le second sans (à la charge du programmeur). Les deux derniers types, "MI" et "FI" produisent des fichiers dont la position des variables est contrôlée par le programmeur (pas de segmentation des enregistrements).

Jusque-là, un Basic propre et complet, mais pas vraiment génial. La gestion des graphismes et sons nous fait entrer dans des zones plus originales. Disons tout de suite que la résolution graphique n'est pas des meilleures dans la version de base. Sans la carte haute résolution optionnelle, la définition se limite à 160 × 72 points. Mais l'affichage est bien exploité par le Basic. Il donne déjà le choix en mode texte entre 16 lignes de 64 colonnes (compatibilité TRS) et 25 lignes de 80 colonnes. Des zones épargnées par le déroulement de l'image (scroll) peuvent être délimitées en haut ou en bas de l'écran. On a d'autre part accès à la vidéo inverse, clignotante ou demi-intensité. A condition de trouver le bon câble péritel, l'exploitation couleur devient accessible : 32 couleurs disponibles grâce à l'interface péritel incluse dans l'ordinateur.

Le confort assuré

Pour le son, toutes les fantaisies sont permises sur les trois voies existantes. L'instruction PLAY se paramètre avec le numéro du canal, l'octave (0 à 6), la note (1 à 12), le volume et la durée. Une modulation d'enveloppe est même prévue, mais elle nécessite une programmation directe des registres du générateur. Celle-ci s'effectue par SOUND suivi du numéro de registre et d'une valeur de 0 à 255.

La vraie richesse du SBasic se trouve dans le domaine des commandes. Elles assurent un pilotage grand confort et sont complétées d'un éditeur bien agréable. Numérotation automatique des lignes de programme (de 0 à 65529) et renumérotation disposant de deux

Les 10 tests de LIST*

Test		
1.	Boucle vide :	12 s
2.	Sous-programme :	36 s
3.	Calcul matriciel :	49 s
4.	Chaînes de caractères :	100 s
5.	Arithmétique :	72 s
6.	Calcul scientifique (16 chiffres) :	230 s
7.	Affichage :	630 s
8.	Tracé d'une ligne graphique :	non réalisable sans interface haute résolution (absence de LINE)
9.	Écriture fichier (disquette) :	290 s
10.	Lecture fichier (disquette) :	160 s

(Voir le détail des tests dans LIST 7, page 83)

Liste des mots clés du Basic du Guépard

	CMD "S"	ERL	INT	NEXT	RIGHT$
	CMD "T"	ERR	INV	NOR	RND
&H	CMD	ERROR	KILL	ON	RSET
&O	CONSB	EXP	LEFT$	OPEN	RUN
ABS	CONSH	FIELD	LEN	OR	SAVE
AND	CONT	FIX	LINE INPUT #	OUT	SET
ASC	COS	FLA	LINE INPUT	PAUSE	SGN
ATN	CVD	FN	LIST	PEEK	SIN
AUTO	CVI	FONCAR	LLIST	PLAY	SOUND
CAR	CVS	FONECR	LOAD	POINT	SQR
CHR$	DATA	FOR	LOC	POKE	STEP
CLEAR	DEF FN	FRE	LOF	POS	STOP
CLM	DEFUSRn	GET	LOG	PRINT #	STR$
CLOSE	DEFxxx	GOSUB	LPRINT	PRINT	STRING$
CLS	DELETE	GOTO	LSET	PUT	TAB
CLV	DI	GRE	MEM	READ	TAN
CMD "C"	DIM	IF	MERGE	REF	THEN
CMD "E"	DU	INKEY$	MID$	REM	TIME$
CMD "F"	EDIT	INP	MKD$	RENEW	TO
CMD "J"	ELSE	INPUT #	MKI$	RENUM	TROFF
CMD "O"	END	INPUT	MKS$	RESET	TRON
CMD "R"	EOF	INSTR	NAME	RESTORE	USING
			NEW	RESUME	USRn
				RETURN	VAL
					VARTPR

options U et X. La première vérifie qu'il n'existe pas d'impossibilité à la renumérotation. Elle constitue un moyen très simple pour contrôler la présence de branchements sur des numéros de ligne non définis. La seconde option, X, accepte, elle, les numéros indéfinis, ce qui évite un « plantage » si l'on décide de renuméroter avant d'avoir fini d'écrire le programme.

La suppression de lignes, par DELETE (ou D) s'accompagne également de deux options permettant de transférer une ligne d'un numéro à l'autre avec ou sans effacement de l'original. La commande LIST autorise de nombreux raccourcis, opérant grâce à une seule touche. Le point liste la ligne courante, les flèches verticales la suivante ou la précédente, le point-virgule ramène à la première ligne du programme, tandis que la barre de fraction conduit au listage de la dernière ligne. Les deux-points et le "à" affichent une page écran complète, avec la ligne courante en bas ou en haut de l'écran. Toutes ces astuces semblent peut-être insignifiantes lorsqu'on en lit la description, mais je peux vous garantir qu'à l'usage, c'est bigrement agréable.

Même confort d'utilisation pour l'éditeur, bien qu'il ne soit pas réellement de type écran. La commande EDIT provoque le passage en mode édition avec affichage d'un symbole graphique en tête de chaque ligne. Le déplacement obéit aux flèches, l'insertion et la suppression étant réalisées en « shiftant » les flèches latérales. La sortie de ce mode s'obtient par Enter ou Break, et dans ce dernier cas, les modifications effectuées ne sont pas prises en compte.

Une aide précieuse pour la mise au point et l'archivage des programmes est apportée par la commande REF. Elle établit une liste sur écran ou sur imprimante des références de variables, d'entiers, de numéros de ligne, d'instructions Basic ou de chaînes de caractères. Facile ainsi de débusquer les doubles emplois de variables ou de mettre en évidence les branchements incorrects. En faisant par exemple REF = GOTO, vous affichez tous les numéros de ligne où se niche un GOTO. La réalisation du tableau des variables pour l'archivage du programme devient un jeu d'enfant. Ah, si tous les Basic disposaient d'une telle commande !

Fiche technique du Guépard

Constructeur et distributeur : *HBN Electronic S.A. BP 2739 · 51060 Reims Cedex*
Prix public : *environ 1 500 FF*
Processeur : *Z80 à 4 MHz*
Mémoire vive disponible : *64 Ko (38 Ko après chargement du SBasic)*
Mémoire morte : *2 Ko (extensible à 16)*
Langage : *SBasic sur disquette fournie avec la machine*
Précision : *calculs sur 16 chiffres significatifs*
Nombre de mots clés du Basic : *140*
Variables : *entières de –32768 à 32767 ; 6 chiffres significatifs pour la simple précision et 16 pour la double précision*

Le SED commandé depuis Basic

Autre commande que je trouve géniale : CMD "C". Elle supprime de la liste les espaces inutiles et/ou les remarques. L'optimisation d'un programme est ainsi réalisée automatiquement. Et l'on ne craint plus de mettre au point sur une version délayée pour exploiter ensuite une liste concentrée. Tous les programmeurs rigoureux apprécieront...

La dernière commande citée n'est pas la seule à réclamer le préfixe CMD. Et, ici nous apparaît un trait de génie de l'interface NewDos-SBasic. En effet toutes les commandes du système d'exploitation sont accessibles directement depuis Basic. Il suffit d'y écrire "CMD" devant la commande. Et l'on parvient ainsi à formater des disquettes, à dupliquer des fichiers, à exécuter toutes sortes de tâches normalement réservées au SED, depuis le Basic, sans perdre le programme en mémoire. D'autres commandes obéissant à CMD sont encore disponibles pour permettre le tri de variables, pour définir des fonctions spéciales (remise à zéro des pointeurs d'adresse de retour de sous-programmes ou de piles FOR...NEXT, échange de variables...) et même pour convertir des dates en numéro de jour de l'année.

Toutes les astuces de commande que nous venons d'examiner mettent en évidence un dialogue réussi entre un système d'exploitation bien pensé et un Basic astucieux. Et l'ordinateur qui accueille ces hôtes de marque possède, nous l'avons vu, sa dose de génie.

Cela montre que les systèmes 8 bits ne sont pas du tout périmés et que les ordinateurs français ne sont pas ridicules comparés aux productions étrangères. Il reste au Guépard à se faire connaître, à compléter sa collection de logiciels et à améliorer peut-être son aspect esthétique. Si ces trois points sont résolus, Guépard fera du chemin.

Xavier de LA TULLAYE

CONCEPTION DE JEUX GRAPHIQUES SUR APPLE II

DANS le cœur de tout « Applemane », il y a un créateur de jeux qui sommeille. Peut-être vous êtes-vous déjà perdu dans la contemplation de ces jeux d'arcade, à l'animation tellement impressionnante. Et peut-être aimeriez-vous aussi en percer le secret ? Voici une méthode d'animation que vous pourrez mettre à profit.

Le graphisme de l'Apple possède une organisation assez particulière. Les pages graphiques sont composées de 192 lignes de 280 points. Le nombre de lignes est standard, mais à quoi correspond ce nombre de points ? Chaque ligne est composée de 40 octets et chaque octet est formé de huit bits, numérotés de 0 à 7. Or, les bits 0 à 6 permettent d'allumer un point sur l'écran, le bit 7 étant réservé à la définition de la couleur. On retrouve donc 40 fois les 7 bits (numérotés de 0 à 6), ce qui fait bien 280 points.

En outre, les colonnes de points ne sont pas équivalentes. Les colonnes paires permettent de dessiner des points violets ou bleus, les colonnes impaires autorisant des points oranges ou verts. Deux points horizontaux consécutifs allumés donnent un point blanc, par addition de couleurs (voir figure 1).

Pour observer ce phénomène, tapez : HGR:VTAB 21:CALL-151 (passage dans le moniteur) 2000:55.

Vous pouvez voir alors l'octet en haut à gauche s'illuminer en violet (si vous avez la couleur !).

Les économies de Steve Wosniak

Des remarques s'imposent :

- les bits à l'écran sont dans le sens inverse de leur écriture binaire ;
- la page graphique numéro 1 s'étend des octets $2000 à $3FFF, la page numéro 2 allant de $4000 à $5FFF ;
- la première ligne d'écran va de $2000 à $2027 (27 en hexadécimal est égal à 39 en décimal), mais l'octet $2028 n'est pas le premier de la ligne numéro 2, comme on aurait pu le penser.

En entrant le programme suivant, on voit ce qui se passe :

```
10 HGR
20 FOR I=8192 TO 16384
30 POKE I,255
40 NEXT I
```

Ce programme se contente d'écrire, dans tous les octets de la page haute résolution, la valeur 255 ($FF) qui donne la couleur blanche. Après avoir rempli la première ligne, il ne passe pas à la seconde. Il va plus bas, à la ligne numéro 64 pour être exact. Voilà qui complique singulièrement les choses.

Les raisons en sont pourtant simples : quand Wosniak et ses amis ont construit l'Apple, ils l'ont conçu de la manière la plus économique possible afin d'offrir un ordinateur avec haute résolution au prix le plus bas. Faire que les lignes d'écran soient consécutives en mémoire aurait augmenté le nombre de composants et donc, le prix de l'Apple.

Si vous avez laissé tourner le programme de remplissage d'écran, vous avez dû remarquer son impressionnante lenteur. Aussi, pour apprécier la vitesse du langage-machine, tapez : HGR: HCOLOR=3:HPLOT 0,0:CALL 62452. Et ça devient remarquable !

Passons à des choses sérieuses : le dessin d'un avion dans le coin en haut

LIST - PAGE 34

à gauche de l'écran. Pour cela, sortons l'arme absolue de l'Applemane graphiste, le papier quadrillé.

Quand les calculs sont trop lents

On choisit les dimensions de notre avion : trois octets de long (21 points) sur huit lignes de hauteur (voir figure 2).

Figure 1
Couleur des points en fonction de la valeur des octets

Octet								Valeur binaire	Valeur hexa	Couleur des colonnes	
0	1	2	3	4	5	6	7			Paires	Impaires
								00000000	$00	Noir 1	Noir 2
							*	10000000	$80	Noir 2	Noir 2
*	*	*	*	*	*	*		01111111	$7F	Blanc 1	Blanc 1
*	*	*	*	*	*	*	*	11111111	$FF	Blanc 2	Blanc 2
*		*		*		*		01010101	$55	Violet	Vert
*		*		*		*	*	11010101	$D5	Bleu	Orange
	*		*		*			00101010	$2A	Vert	Violet
	*		*		*		*	10101010	$AA	Orange	Bleu

L'étoile (*) représente un bit allumé (mis à 1).

Figure 2
Représentation de l'avion

1er octet							2e octet							3e octet							Table de l'avion
					*	*	*	*	*	*	*	*	*	*	*	*					60 7F 07
*	*							*					*							*	03 42 40
*	*	*					*	*	*	*	*	*	*	*	*	*	*	*		*	07 7F 5F
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	7F 7F 7F
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*		*	7F 7F 5F
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*			*	7F 7F 4F
			*										*	*							08 40 01
	*	*												*	*						06 00 03

Et pour le faire apparaître à l'écran, il faut taper : HGR:VTAB 21:CALL-151

```
2000:60 7F 07
2400:03 42 40
2800:07 7F 5F
2C00:7F 7F 7F
3000:7F 7F 5F
3400:7F 7F 4F
3800:08 40 01
3C00:06 00 03
```

Le problème maintenant est de placer cet avion à un endroit quelconque de l'écran. Ce qui revient à trouver les adresses de la ligne initiale et des suivantes où seront écrits ces octets. La méthode qui consiste à faire calculer ces adresses n'est pas adéquate du fait de sa lenteur. Il va donc falloir utiliser ce que les Américains appellent la méthode du « Table Look Up ». Elle consiste à former une table en deux parties :

- dans la première partie, on place les octets hauts des adresses des lignes telles qu'elles apparaissent à l'écran ;
- dans la seconde, de la même manière, on place les octets bas de ces adresses.

Par exemple :

TABREFH:20 24 28 30 38 3C...
TABREFB:00 00 00 00 00 00...

Ce qui donne la table de la figure 3.

Figure 3
Table Look Up

```
]06000.617F

6000- 20 24 28 2C 30 34 38 3C
6008- 20 24 28 2C 30 34 38 3C
6010- 21 25 29 2D 31 35 39 3D
6018- 21 25 29 2D 31 35 39 3D
6020- 22 26 2A 2E 32 36 3A 3E
6028- 22 26 2A 2E 32 36 3A 3E
6030- 23 27 2B 2F 33 37 3B 3F
6038- 23 27 2B 2F 33 37 3B 3F
6040- 20 24 28 2C 30 34 38 3C
6048- 20 24 28 2C 30 34 38 3C
6050- 21 25 29 2D 31 35 39 3D
6058- 21 25 29 2D 31 35 39 3D
6060- 22 26 2A 2E 32 36 3A 3E
6068- 22 26 2A 2E 32 36 3A 3E
6070- 23 27 2B 2F 33 37 3B 3F
6078- 23 27 2B 2F 33 37 3B 3F
6080- 20 24 28 2C 30 34 38 3C
6088- 20 24 28 2C 30 34 38 3C
6090- 21 25 29 2D 31 35 39 3D
6098- 21 25 29 2D 31 35 39 3D
60A0- 22 26 2A 2E 32 36 3A 3E
60A8- 22 26 2A 2E 32 36 3A 3E
60B0- 23 27 2B 2F 33 37 3B 3F
60B8- 23 27 2B 2F 33 37 3B 3F
60C0- 00 00 00 00 00 00 00 00
60C8- 80 80 80 80 80 80 80 80
60D0- 00 00 00 00 00 00 00 00
60D8- 80 80 80 80 80 80 80 80
60E0- 00 00 00 00 00 00 00 00
60E8- 80 80 80 80 80 80 80 80
60F0- 00 00 00 00 00 00 00 00
60F8- 80 80 80 80 80 80 80 80
6100- 28 28 28 28 28 28 28 28
6108- A8 A8 A8 A8 A8 A8 A8 A8
6110- 28 28 28 28 28 28 28 28
6118- A8 A8 A8 A8 A8 A8 A8 A8
6120- 28 28 28 28 28 28 28 28
6128- A8 A8 A8 A8 A8 A8 A8 A8
6130- 28 28 28 28 28 28 28 28
6138- A8 A8 A8 A8 A8 A8 A8 A8
6140- 50 50 50 50 50 50 50 50
6148- D0 D0 D0 D0 D0 D0 D0 D0
6150- 50 50 50 50 50 50 50 50
6158- D0 D0 D0 D0 D0 D0 D0 D0
6160- 50 50 50 50 50 50 50 50
6168- D0 D0 D0 D0 D0 D0 D0 D0
6170- 50 50 50 50 50 50 50 50
6178- D0 D0 D0 D0 D0 D0 D0 D0
*
```

CONCEPTION DE JEUX GRAPHIQUES SUR APPLE II

Ne soyez pas effrayé à l'idée de rentrer cette table de 384 octets sans faire de fautes (fautes qui seraient particulièrement ennuyeuses !). Un petit programme Basic (programme 1) fait le travail d'assemblage et de sauvegarde de cette table.

L'octet en haut à gauche de l'écran sur lequel on souhaite dessiner notre avion est repéré par un décalage horizontal HORI et par la ligne de départ VERT. Pour trouver l'adresse de cet octet, on utilise la routine suivante (il faut ici que le registre Y contienne le numéro de la ligne verticale) :

```
ADRESSE  LDA TABREFB, Y ; charge adresse octet bas
         CLC            ; prépare l'addition
         ADC HORI       ; ajoute le décalage horizontal
         STA GRAFB      ; écrit pointeur en page zéro
         LDA TABREFH, Y ; charge adresse octet haut
         STA GRAFH      ; écrit pointeur suivant de GRAFB
         LDY #0
         RTS
```

Il faut maintenant charger les différents octets qui composent notre avion, pour les écrire à l'emplacement correct sur l'écran. Tout d'abord, inscrivons tous ces octets à la suite quelque part en mémoire. Sur l'écran, les huit lignes de trois octets devront être reconstituées. Pour ce faire, il faut initialiser la figure comme suit, avant de la dessiner :

```
INIT     LDA #3      ; charge l'adresse
         STA FIGH    ; de l'avion dans
         LDA #0      ; des pointeurs
         STA FIGB    ; en page zéro
         LDA #8      ; 8 dans la
         STA FIGPROF ; profondeur de figure
         LDA #3      ; 3 dans la longueur
         STA FIGLONG ; de figure
         STA TEMP    ; et un pointeur temporaire
```

Programme 1
Assemblage et sauvegarde de la table Look Up

```
10 T=24576: REM $6000
20 FOR Y=0 TO 191
30 Y1=INT(Y/8):YA=Y-Y1*8
40 Y2=INT(Y1/8):YB=Y1-Y2*8
50 YY=8192+Y2*40+YB*128+YA*1024
60 POKE T+Y,INT(YY/256)
70 POKE T+192+Y,YY-INT(YY/256)*256
80 NEXT Y
90 PRINT CHR$(4)"BSAVETABREF,A$6000,L$180"
```

On utilise alors deux modes d'adressage du 6502 : l'indexé indirect et l'indirect indexé. Par la routine INIT, nous avons stocké la table dans le format bas-haut en FIGB,FIGH (ce sont deux adresses consécutives page zéro). Le registre X étant chargé avec la valeur zéro, un LDA (FIGB,X) charge l'octet pointé par FIGB.FIGH (octet de l'avion) dans l'accumulateur.

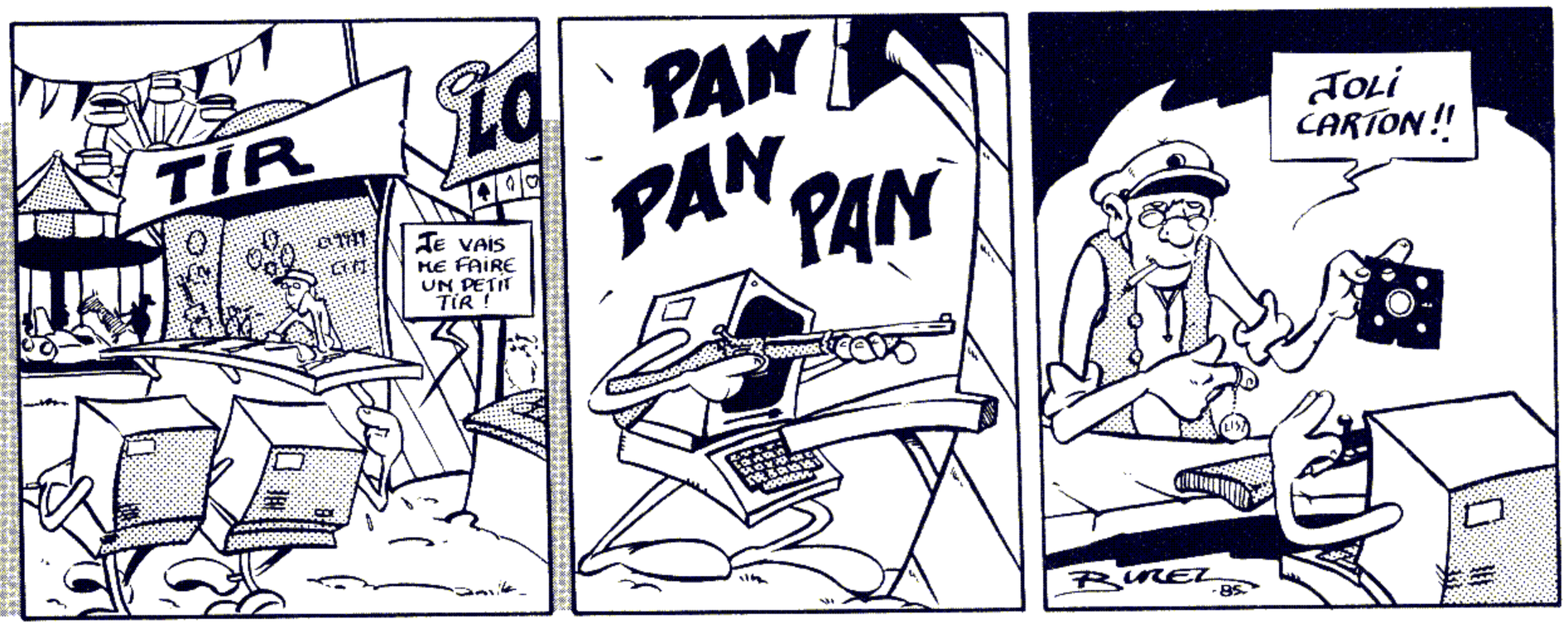
Avec la routine ADRESSE, nous avions stocké en GRAFB, GRAFH (deux adresses consécutives page zéro), l'adresse de l'octet d'écran. Le registre Y étant chargé avec la valeur zéro, on fait STA (GRAFB),Y pour écrire l'octet d'avion sur l'écran. Il reste à prendre garde d'augmenter la position verticale chaque fois que l'on aura écrit trois octets (figure 2).

Paré à décoller

L'avion est alors en mémoire d'écran. Mais pour le dessiner, comme notre écran est noir, il nous faut utiliser une petite astuce : l'emploi de EOR (ou exclusif). Il nous permettra d'utiliser la même routine pour dessiner l'avion aussi bien que pour l'effacer, deux écritures consécutives remettant l'écran en noir. D'où la séquence d'ordres :

```
LDA (FIGB, X)   ; charge octet d'avion
EOR (GRAFB),Y ; fait ou exclusif accumulateur/écran
STA (GRAFB),Y ; écrit le résultat à l'écran
```

Le choix de l'adressage indexé indirect pour le chargement des octets de l'avion est intéressant. Il permet d'employer la même routine de dessin et d'effaçage des formes, en écrivant différentes adresses pour différentes formes dans les pointeurs FIGB.FIGH. Un



Programme 2

Déplacement de l'avion, liste en Assembleur

```
SOURCE FILE: AVSON
0000:                1 *DEPLACEMENT D'AVION
----- NEXT OBJECT FILE NAME IS AVSON.OBJ0
6200:                2          ORG  $6200
0000:                3 GRAFB    EQU  $00
0001:                4 GRAFH    EQU  $01
60C0:                5 TABREFB  EQU  $60C0
6000:                6 TABREFH  EQU  $6000
00F8:                7 FIGB     EQU  $F8
00F9:                8 FIGH     EQU  $F9
00FA:                9 FIGPROF  EQU  $FA
00FB:               10 HORI     EQU  $FB
00FC:               11 FIGLONG  EQU  $FC
00FD:               12 TEMP     EQU  $FD
00FE:               13 VERT     EQU  $FE
00FF:               14 TVERT    EQU  $FF
FCA8:               15 DELAI    EQU  $FCA8
0018:               16 VTOUCHE  EQU  $18
C030:               17 HP       EQU  $C030
6200:               18 *
6200:               19 *EFFACEMENT DE L'ECRAN
6200:               20 *
6200:               21 *MET EN GRAPHIQUE
6200:8D 50 C0       22          STA  $C050      ;MODE GRAPHIQUE
6203:8D 52 C0       23          STA  $C052      ;PLEIN ECRAN
6206:8D 54 C0       24          STA  $C054      ;PAGE PRIMAIRE
6209:8D 57 C0       25          STA  $C057      ;MODE HGR
620C:A9 00          26          LDA  #$00
620E:A8             27          TAY
620F:85 00          28          STA  GRAFB
6211:A9 20          29 NET      LDA  #$20
6213:85 01          30          STA  GRAFH
6215:AA             31          TAX
6216:A9 00          32          LDA  #$00       ;NOIR 1
6218:91 00          33 NET1     STA  (GRAFB),Y ;ECRIS A L'ECRAN
621A:C8             34          INY             ;OCTET SUIVANT
621B:D0 FB          35          BNE  NET1       ;PAGE INCOMPLETE
621D:E6 01          36          INC  GRAFH      ;PAGE SUIVANTE
621F:CA             37          DEX             ;DECREMENTE COMPTEUR DE PAGES
6220:D0 F6          38          BNE  NET1
6222:A9 60          39          LDA  #$60       ;CENTRE VERT
6224:85 FE          40          STA  VERT
6226:85 FF          41          STA  TVERT
6228:A9 14          42          LDA  #$14       ;CENTRE HORI
622A:85 FB          43          STA  HORI
622C:85 18          44          STA  VTOUCHE
622E:20 46 62       45 BOUCLE   JSR  INIT       ;VA DESSINER
6231:A9 80          46          LDA  #$80
6233:20 A8 FC       47          JSR  DELAI      ;ATTENDS
6236:20 46 62       48          JSR  INIT       ;VA EFFACER
6239:20 F2 62       49          JSR  SON        ;FAIT CLIC
623C:20 8B 62       50          JSR  TOUCHE     ;LIS CLAVIER
623F:A5 18          51          LDA  VTOUCHE
6241:C9 9B          52          CMP  #$9B       ;ESCAPE ?
6243:D0 E9          53          BNE  BOUCLE     ;NON
6245:60             54          RTS             ;OUI RETOUR A L'APPELANT
6246:               55 *
6246:               56 *INIT AVION
6246:               57 *
6246:A9 03          58 INIT     LDA  #$03
6248:85 F9          59          STA  FIGH
624A:A9 00          60          LDA  #$00
624C:85 F8          61          STA  FIGB
624E:A9 08          62          LDA  #$08
6250:85 FA          63          STA  FIGPROF
6252:A9 03          64          LDA  #$03
6254:85 FC          65          STA  FIGLONG
6256:85 FD          66          STA  TEMP
6258:A4 FF          67 DESSIN   LDY  TVERT
625A:20 7B 62       68          JSR  ADRESSE
625D:A2 00          69          LDX  #$00
625F:A5 FD          70          LDA  TEMP
6261:85 FC          71          STA  FIGLONG
6263:A1 F8          72 DESSIN2  LDA  (FIGB,X)
6265:51 00          73          EOR  (GRAFB),Y
6267:91 00          74          STA  (GRAFB),Y
6269:E6 F8          75          INC  FIGB
626B:C8             76          INY
626C:C6 FC          77          DEC  FIGLONG
626E:D0 F3          78          BNE  DESSIN2
6270:E6 FF          79          INC  TVERT
6272:C6 FA          80          DEC  FIGPROF
6274:D0 E2          81          BNE  DESSIN
6276:A5 FE          82          LDA  VERT
6278:85 FF          83          STA  TVERT
627A:60             84          RTS
627B:B9 C0 60       85 ADRESSE  LDA  TABREFB,Y
627E:18             86          CLC
627F:65 FB          87          ADC  HORI
6281:85 00          88          STA  GRAFB
6283:B9 00 60       89          LDA  TABREFH,Y
6286:85 01          90          STA  GRAFH
6288:A0 00          91          LDY  #$00
628A:60             92          RTS
628B:AD 00 C0       93 TOUCHE   LDA  $C000
628E:30 02          94          BMI  FRAPPE
6290:A5 18          95          LDA  VTOUCHE
6292:C9 C9          96 FRAPPE   CMP  #$C9       ;"I"
6294:F0 13          97          BEQ  MONTE
6296:C9 CD          98          CMP  #$CD       ;"M"
6298:F0 1F          99          BEQ  DESCENT
629A:C9 CA         100          CMP  #$CA       ;"J"
629C:F0 2B         101          BEQ  GAUCHE
629E:C9 CB         102          CMP  #$CB       ;"K"
62A0:F0 33         103          BEQ  DROITE
62A2:C9 9B         104          CMP  #$9B       ;ESCAPE
62A4:F0 3F         105          BEQ  SORTIE
62A6:              106 *AUCUNE TOUCHE VALIDE
62A6:4C ED 62      107          JMP  CLAVLU
62A9:85 18         108 MONTE    STA  VTOUCHE
62AB:C6 FE         109          DEC  VERT
62AD:A5 FE         110          LDA  VERT
62AF:C9 FF         111          CMP  #$FF
62B1:D0 3A         112          BNE  CLAVLU
62B3:A9 B8         113          LDA  #$B8
62B5:85 FE         114          STA  VERT
62B7:30 34         115          BMI  CLAVLU
62B9:85 18         116 DESCENT  STA  VTOUCHE
62BB:E6 FE         117          INC  VERT
62BD:A5 FE         118          LDA  VERT
62BF:C9 B9         119          CMP  #$B9
62C1:D0 2A         120          BNE  CLAVLU
62C3:A9 00         121          LDA  #$00
62C5:85 FE         122          STA  VERT
62C7:F0 24         123          BEQ  CLAVLU
62C9:85 18         124 GAUCHE   STA  VTOUCHE
62CB:C6 FB         125          DEC  HORI
62CD:10 1E         126          BPL  CLAVLU
62CF:A9 25         127          LDA  #$25
62D1:85 FB         128          STA  HORI
62D3:D0 18         129          BNE  CLAVLU
62D5:85 18         130 DROITE   STA  VTOUCHE
62D7:E6 FB         131          INC  HORI
62D9:A5 FB         132          LDA  HORI
62DB:C9 26         133          CMP  #$26
62DD:D0 0E         134          BNE  CLAVLU
62DF:A9 00         135          LDA  #$00
62E1:85 FB         136          STA  HORI
62E3:F0 08         137          BEQ  CLAVLU
62E5:85 18         138 SORTIE   STA  VTOUCHE
62E7:2C 10 C0      139          BIT  $C010
62EA:8D 51 C0      140          STA  $C051
62ED:A5 FE         141 CLAVLU   LDA  VERT
62EF:85 FF         142          STA  TVERT
62F1:60            143          RTS
62F2:AD 30 C0      144 SON      LDA  HP
62F5:60            145          RTS

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

Programme principal

```
10 D$ =  CHR$ (4)
20  REM  TABLE DE L'AVION
30  PRINT D$"BLOAD AVION.BLOC"
40  REM  TABLE LOOK UP
50  PRINT D$"BLOAD TABREF"
60  REM  LANCE LE MODULE
70  PRINT D$"BRUN AVSON.OBJ0"
```

adressage indexé simple ne permettrait de mettre au travail qu'une seule forme, à une adresse fixée à l'avance.

Mais on peut faire encore mieux : faire voler l'avion et contrôler ce vol depuis le clavier, avec les touches I,J,K,M pour le déplacement et ESC pour sortir du programme, toute autre touche stoppant le mouvement. Il nous faut donc saisir une touche au clavier par LDA $C000, puis suivant le cas, incrémenter ou décrémenter VERT ou HORI, provoquer un petit clic dans le haut parleur au passage et le tour est joué (programme 2).

Le programme principal se chargera de mettre tous ces morceaux en mémoire et une fois lancé, il lancera l'avion.

Gérald ANFOSSI

UNE PROCÉDURE PASCAL

LECTURE MULTIPLE

DANS un programme, un même traitement doit souvent être appliqué à un grand nombre de données. La structure classique d'un tel programme lance le traitement après la lecture de chaque donnée ou groupe de données. Une autre manière de faire consiste à lire toutes les données, puis à réaliser les traitements en série. La création en Pascal d'une nouvelle fonction va permettre de choisir entre les deux manières de faire.

Le traitement des données dans un programme peut être effectué de deux manières différentes. La première (schématisée par la figure 1) est de la forme donnée-traitement-donnée-traitement-..., c'est-à-dire que le traitement est effectué après la lecture de chaque donnée. La seconde (figure 2) est de la forme données-traitements, c'est-à-dire que les traitements n'ont lieu qu'après la lecture de toutes les données. Avec une telle structure, les données sont introduites par groupes, un sous-programme permettant d'extraire les données individuelles de la liste.

Figure 1

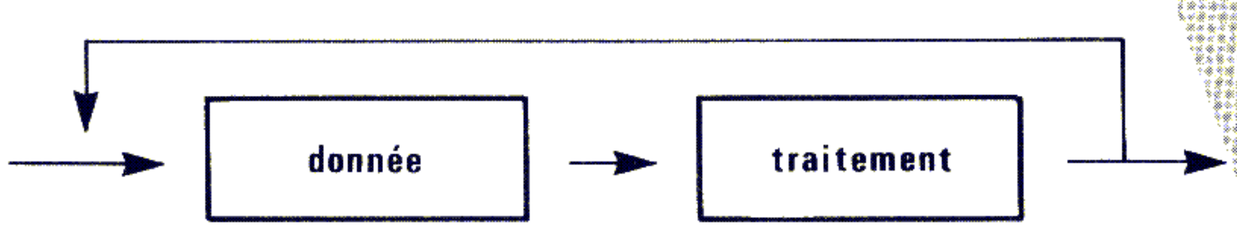
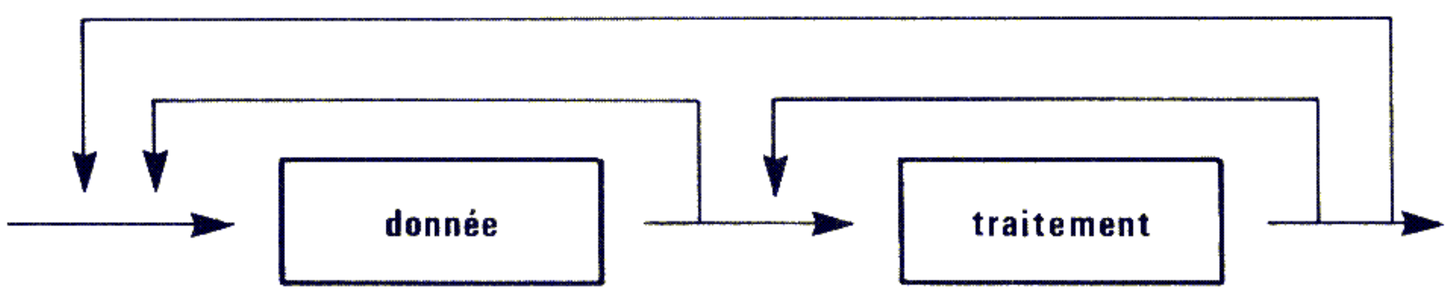


Figure 2



Une chasse aux nombres

La fonction Pascal décrite ici traite les nombres entiers. Elle s'applique donc essentiellement au traitement des références. Elle permet, à partir d'une chaîne de caractères contenant une liste de nombres, d'extraire ceux-ci un par un. Par exemple, appliquée à la liste "72 37 28 92", elle retournera successivement les valeurs 72, 37, 28 et 92.

La déclaration de cette fonction (première ligne du programme) fait apparaître les paramètres. Le premier, LISTE, est une chaîne de caractères contenant le ou les nombres que l'on désire transférer dans une variable de type INTEGER. Ce paramètre est modifié par la procédure car le nombre lu est effacé de la chaîne. Le second paramètre, VALEUR, représente le nombre extrait de LISTE.

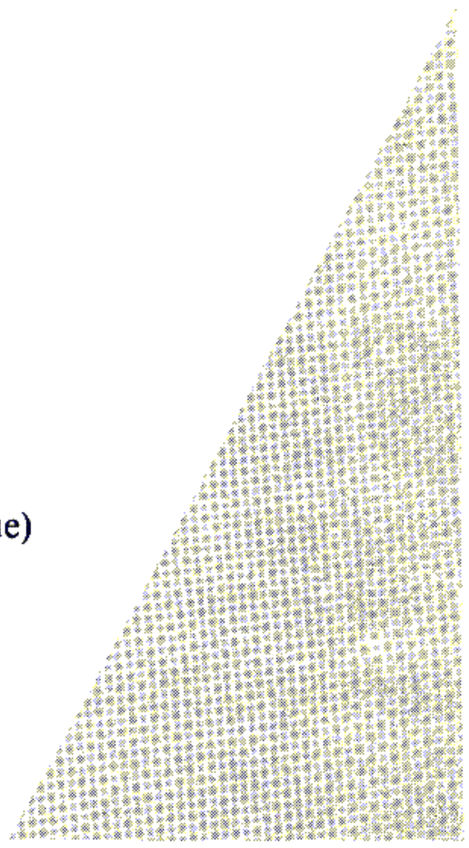
La fonction efface donc de LISTE le premier entier qu'elle retourne dans VALEUR. Son résultat est de type booléen. Il indique s'il a été possible d'extraire une valeur de LISTE : il est vrai (true) si la chaîne mémorisait un nombre, et faux (false) sinon.

Les nombres de la liste peuvent être séparés par n'importe quel caractère non numérique, et différent de " + " et " – ". D'une manière générale, le caractère séparateur sera l'espace ou le point-

Programme de lecture des données


```
function suivant (var liste : string; var valeur : integer) : boolean;
var     i          : integer;
        longueur   : integer;
        arret      : boolean;
        negatif    : boolean;
        numerique  : set of char;

begin
   numerique: = ['0'..'9','+','-'];
   suivant: = false;
   valeur: = 0;
   longueur: = length(liste);
   if longueur > 0
      then
         begin
   i: = 1;
   while not (liste[i] in numerique) and (i < longueur) do
      i: = i + 1;
   if liste[i] in numerique
      then
         begin
            suivant: = true;
            negatif: = false;
            arret: = false;
            repeat
              if liste[i] in ['0'..'9']
                then
                  valeur: = 10*valeur + ord(liste[i]) - ord('0')
                else
                  negatif: = liste[i] = '-';
              liste[i]: = chr(0);
              i: = i + 1;
              arret: = i > longueur;
              if not arret then arret: = not (liste[i] in numerique)
            until arret;
            if negatif then valeur: = - valeur
         end
   end;
```

traitement des données



Lecture de toutes les données

virgule (ils ne sont pas sujet à confusion, comme le sont la virgule et le point).

Si l'on a besoin à plusieurs reprises, dans le programme, de l'entier retourné par la fonction, il ne faut pas oublier de le stocker dans une variable intermédiaire. En effet, chaque appel de la fonction a pour objet, outre de fournir la valeur numérique, de supprimer cette valeur de la liste.

Si la fonction a pour résultat *faux*, le nombre contenu dans le paramètre VALEUR est égal à 0. La réciproque n'est pas vraie : VALEUR peut contenir 0 (si ce nombre appartient à la liste) alors que la fonction prend la valeur *vrai*.

Une telle fonction peut être utilisée, par exemple, lorsque l'on désire indiquer immédiatement plusieurs références (en vue de leur impression), sans attendre que la précédente ait été traitée pour indiquer la référence suivante. Dans ce cas, elle s'insère dans un programme d'impression de dossiers, de la manière suivante :

```
write ('Numeros des dossiers à imprimer (<return> pour sortir) : ');
readln(liste);
while suivant(liste,nodossier) do
   begin
      lire(dossier,nodossier);
      imprimer(dossier)
   end;
```

Dans le but de réduire le temps de calcul d'une telle fonction, un ensemble NUMERIQUE est créé dès le début de son activation. Il mémorise les caractères qui peuvent apparaître dans les nombres. Ceci évite au système de reconstruire régulièrement cet ensemble lors de l'exécution.

La fonction recherche le premier caractère numérique de la liste. Dès qu'il a été trouvé (ou bien lorsque la chaîne a été explorée en totalité), cette recherche se termine. Il est alors possible de connaître le résultat de la fonction. Si le caractère a été trouvé, elle prend la valeur *vrai* puisque le nombre suivant existe dans la liste. Sinon, elle prend la valeur *faux*, la liste ne contient plus de nombre.

Puis cette liste est à nouveau explorée. Dès qu'un chiffre est détecté, il est pris en compte dans la valeur entière. Il est ensuite effacé de la chaîne, ou plus exactement remplacé par le caractère NUL de code ASCII égal à 0. Le parcours de la liste s'arrête dès l'occurrence d'un symbole non numérique. Comme le caractère nul peut ne pas être affiché sur les écrans, il est possible d'indiquer à tout moment la liste des données à traiter. Ceci s'effectue simplement par l'affichage de la variable passée dans le paramètre LISTE.

Cette fonction simple rendra sans doute de grands services aux utilisateurs de programmes informatiques, en leur évitant d'attendre la fin d'un traitement avant de lancer le suivant. Elle contribue ainsi, de manière efficace, à rendre ces programmes plus clairs et plus agréables à utiliser.

Thierry CHAMORET

UNE HORLOGE PLUS PRÉCISE

SUR le TRS-80, le système d'exploitation des disquettes offre une horloge qui bat la seconde. Mais il est possible d'améliorer très nettement ces performances et de chronométrer au 40e de seconde.

Les possesseurs de TRS-80 avec disquette savent que les différents SED utilisables permettent un affichage optionnel des indications de l'horloge. Certains utilisent cette horloge interne pour afficher automatiquement des durées de calculs en encadrant ceux-ci par : UD$ = TIME$ (calculs) UF$ = TIME$. Il suffit alors de traiter UD$ et UF$ par un sous-programme :

```
10  DEF FNDU (A$,B$,I) = VAL
    (MID$(B$,I,2)) – VAL
    (MID$(A$,I,2))
20  DH = FNDU(UD$,UF$,10)
30  DM = FNDU(UD$,UF$,13)
40  DS = FNDU(UD$,UF$,16)
50  IF DS < 0 THEN
    DS = DS + 60:DM = DM – 1
60  IF DM < 0 THEN DM = DM +
    60 :DH = DH – 1
200 PRINT USING"DUREE : ##
    Heures ### Minutes ###
    Secondes";DH;DM;DS
```

qui affichera la durée à la seconde près.

Dans certains cas il peut être intéressant ou nécessaire de connaître cette durée avec plus de précision, ou de pouvoir évaluer des durées inférieures à la seconde. Or, sous DEBUG (TrsDos mais pas NewDos) l'instruction « U » permet de constater que l'horloge « bat » plus vite que la seconde.

Saisir à la seconde juste

En effet, si l'affichage position 4041H change à chaque seconde, l'affichage en 4040H change beaucoup plus vite, en fait tous les 1/40 de seconde (25 millisecondes) et les quelques lignes suivantes permettent de le vérifier à l'écran :

```
10 A = 0:B = 0:S1 = 0:UD$ = ""
20 CLS
30 A = PEEK(&H4041)
40 B = PEEK(&H4041)
50 IF A = B THEN 40
60 S1 = PEEK(&H4040)
70 UD$ = TIME$
80 PRINT RIGHT$(UD$,8)
90 GOTO 30
```

On notera à l'occasion de ce contrôle, car cela est important pour la suite, une remise à zéro de 4040H à chaque passage à 256.

Il est donc théoriquement possible, sous condition de pouvoir saisir cette valeur juste avant celle de l'horloge (donnée par UD$ = TIME$), de mesurer des durées au 1/40 de seconde.

En pratique cependant, si l'on se contente d'effectuer la saisie des données par : S1 = PEEK (&H4040):UD$ = TIME$, puis, après les calculs, S2 = PEEK(&H4040):UF$ = TIME$, on constate des anomalies lors du traitement de S2-S1. Il est en effet hautement improbable, par cette méthode, de réussir à saisir « S1 » exactement au moment du début de la seconde ; il en résulte une erreur aléatoire pouvant atteindre 39/40 de seconde et le problème n'est toujours pas résolu.

Il faut donc non pas déclencher le chronomètre au début du calcul, mais, au contraire, déclencher le calcul lorsque le chronomètre est exactement à la seconde juste. Ceci semble pouvoir être obtenu par :

```
n n A = PEEK(&H4041)
x x B = PEEK(&H4041)
m m IF A = B THEN x x
o o S1 = PEEK(&H4040):UD$ = TIME$
(calcul)
p p S2 = PEEK(&H4040):UF$ = TIME$
```

Le traitement de S2 – S1 découle du fait que S1 augmente de 40 à chaque seconde écoulée et qu'il est remis à zéro à chaque passage à 256. Ainsi, après traitement de UD$ et UF$, afin d'obtenir DH,DM et DS, il faut encore ajouter au sous-programme de traitement :

Au 40e de seconde
Programme pour TRS-80 Modèle 1
Auteur Henri Volleau
Copyright *LIST* et l'auteur

Partie en Basic

```
10 SAVE"TESTDURE/BAS:0":STOP
20 '     PGM DE DEMONSTRATION DE CALCUL DE DUREES
30 '       DUREE PRECISE à +/- 12.5 milliseconde
40 '
50 '=============================================
60 '
70 '---------------
100 CMD"LOAD USRDUREE/ASS"
110 '-------------
200 CLS
210 CLEAR 100
220 DEFUSR0=&HFE00
230 DEF FNDU(A$,B$,I)=VAL(MID$(B$,I,2))-VAL(MID$(A$,I,2))
240 DEF FNA1(A,B,C)=40*(3600*A+60*B+C)
250 DEF FNA2(A,B,C,D)=D+FNA1(A,B,C)-256*FIX(FNA1(A,B,C)/256)
260 '--------------
300 INPUT "Nb de  boucles ":IM
310 '-------------
400 CLS
410 S1=USR0(0)
420 UD$=TIME$
430 FOR I=1 TO IM
440 PRINT@ 0,I;
450 NEXT
460 S2=PEEK(&H4040)
470 UF$=TIME$
480 '-------------
500 CLS
510 PRINT UF$:S2
520 PRINT UD$:S1
530 PRINT
540 DH=FNDU(UD$,UF$,10)
550 DM=FNDU(UD$,UF$,13)
560 DS=FNDU(UD$,UF$,16)
570 IF DS<0 THEN DS=DS+60:DM=DM-1
580 IF DM<0 THEN DM=DM+60:DH=DH-1
590 S1=FNA2(DH,DM,DS,S1)
600 XS=S2-S1
610 IF XS<0 THEN XS=XS+256
620 DS=DS+XS/40
630 PRINT USING"DUREE : ## Heures ### Minutes ###.### Secondes":
    DH;DM;DS
640 GOTO 300
```

Routine en Langage-Machine

```
                00100 ; PGM DE SAISIE DU 1/40 DE SECONDE
                00110 ;--------------------------------
                00120 ;
FE00            00130          ORG     0FE00H
FE00  214140    00140 DEB      LD      HL,4041H
FE03  7E        00150          LD      A,(HL)
FE04  BE        00160 TEST     CP      (HL)
FE05  28FD      00170          JR      Z,TEST
FE07  2B        00180          DEC     HL
FE08  6E        00190          LD      L,(HL)
FE09  2600      00200          LD      H,0
FE0B  C39A0A    00210          JP      0A9AH
FE00            00220          END     DEB
```

Le seul ordre spécifique au NewDos-80 est CMD "LOAD...". Les utilisateurs de Trs Dos devront donc supprimer la ligne 100 (qui permet le chargement de la routine directement sous Basic) et prévoir le chargement préalable de cette routine sous Dos.

70 S1 = S1 + 40*(3600*DH + 60*DM + DS)
80 S1 = S1 – 256*FIX(S1/256)
90 MS = S2 – S1
100 IF MS < 0 THEN MS = MS + 256
110 DS = DS + MS/40
et modifier le USING de la ligne 200 (###.### Secondes) pour obtenir l'affichage des millisecondes.

Tester en s'amusant

L'essai d'un tel programme montre cependant encore quelques petites anomalies sur les courtes durées, ce qui semble indiquer que le Basic est trop lent pour saisir le moment exact du passage de la seconde juste.

J'ai donc préféré d'une part confier cette tâche à un petit programme en langage-machine qui sera exécuté par USR et, d'autre part, effectuer les corrections de S1 par fonction.

Finalement le programme obtenu, qui permet de s'amuser à tester la durée de boucles simples (ou tout autre calcul exécuté entre les lignes 430/450), donne entière satisfaction. Il permet, entre autres, de déterminer les portions de calcul à optimiser ou plus radicalement à remplacer par des appels USR pour augmenter la rapidité d'exécution.

Henri VOLLEAU

3D MOVER
L'ESPACE A TROIS DIMENSIONS DANS LE ZX SPECTRUM

La troisième dimension attire de plus en plus les amateurs de programmation, si l'on en juge par l'abondance des nouveaux livres de micro-informatique traitant de ce sujet. Grâce au programme 3D Mover pour le Spectrum, il est très facile d'accéder à cette troisième dimension, sans aucune connaissance mathématique particulière.

Le logiciel *3D Mover* est fourni sur une cassette dont les deux faces sont enregistrées. Il est accompagné d'une légère brochure explicative, format agenda de poche, d'une quinzaine de pages.

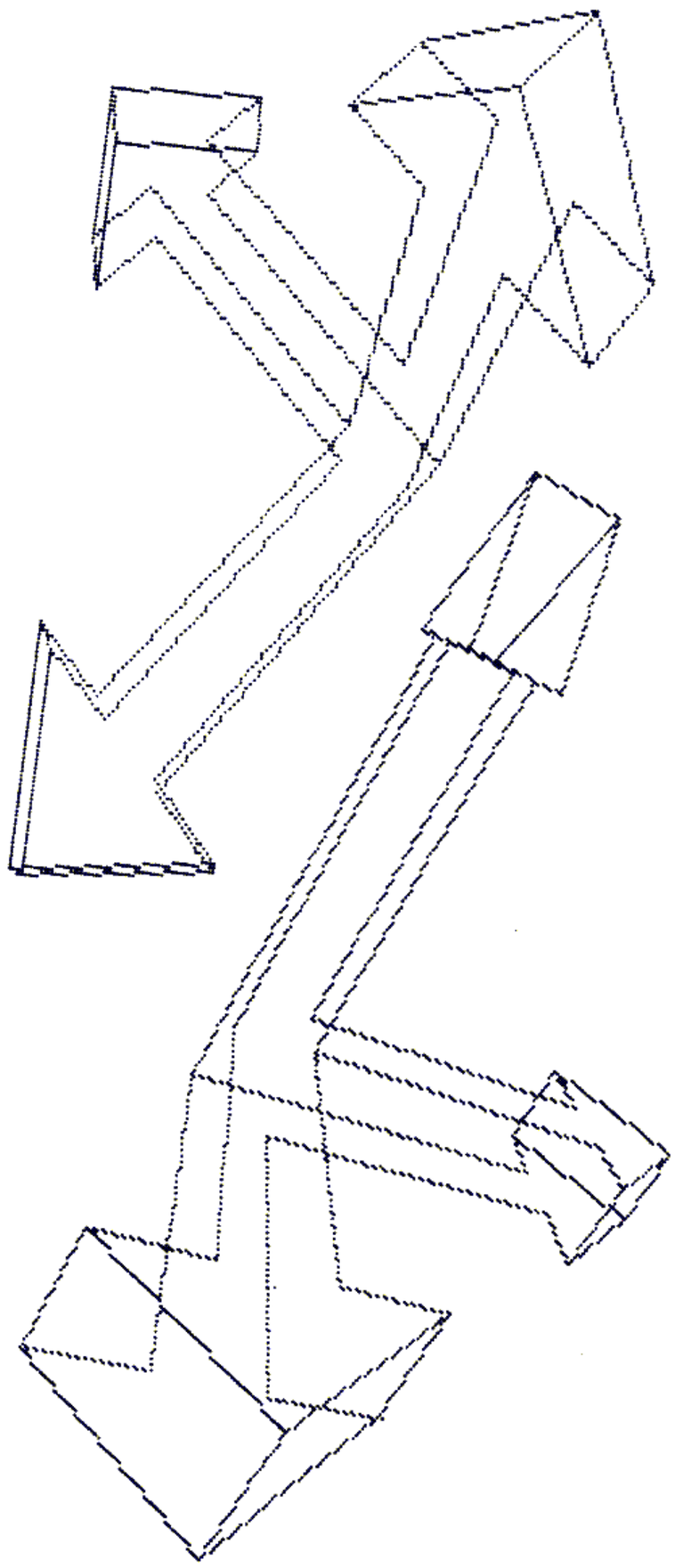
Sur une face, nous trouvons *3D Mover* proprement dit, tandis que sur l'autre, nous trouvons un autre programme nommé *3D Basic*. L'un et l'autre de ces programmes s'autolancent par une démonstration de leurs possibilités. Trois vecteurs définissent une base de l'espace et s'animent alors sur l'écran à bonne vitesse, au cours de rotations, translations, zooms avant et arrière. De quoi nous allécher avant de réaliser nous-même nos propres animations.

Avant de voir les possibilités respectives des deux programmes, il n'est pas inutile d'ouvrir une parenthèse pour parler de la qualité des enregistrements des logiciels vendus sur cassette.

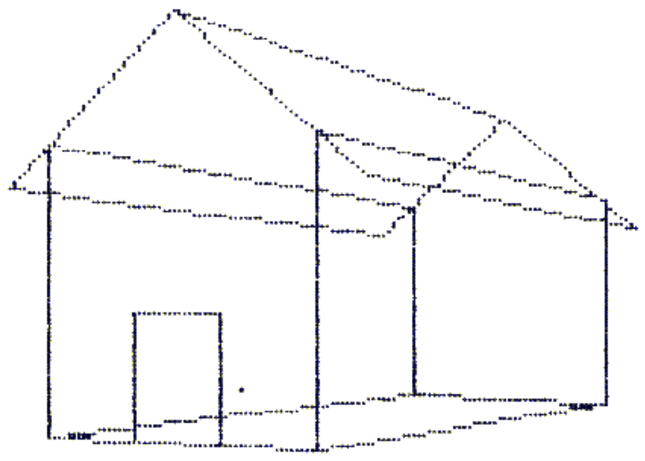
Bien que disposant de deux lecteurs de cassettes différents, je n'ai pas pu lire correctement la face contenant *3D Mover*. Le problème, dans ce cas, devait être dû à une amorce finale trop courte qui entraîne un ressaut en fin de lecture et donc le message désolant d'erreur de chargement. Ce n'est qu'après bien des acrobaties et après avoir compris que le dernier programme en cause ne contenait que les DATAs de la démonstration, que j'ai pu finalement disposer de l'essence même du programme *3D Mover*.

Les sociétés diffusant des logiciels ont un effort à faire quant à la qualité des supports sur lesquels ils enregistrent leurs programmes, surtout quand ils sont de qualité.

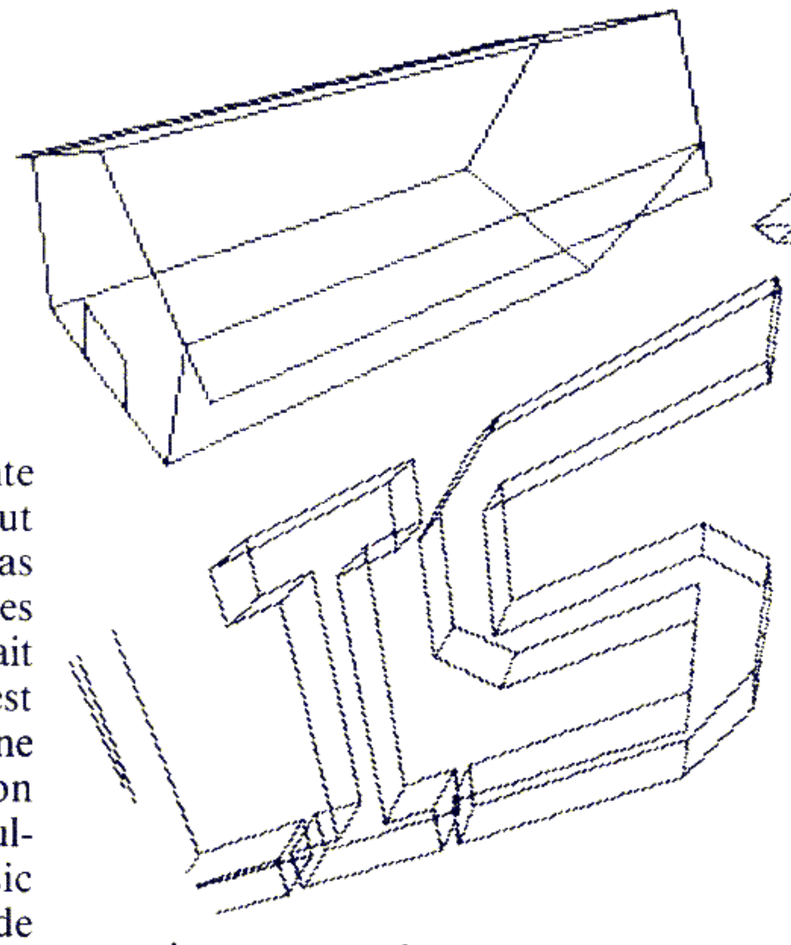
Au chargement, ce n'est pas moins de cinq programmes ou blocs d'octets qui vont être implantés en mémoire, dont deux contenant les DATAs de la

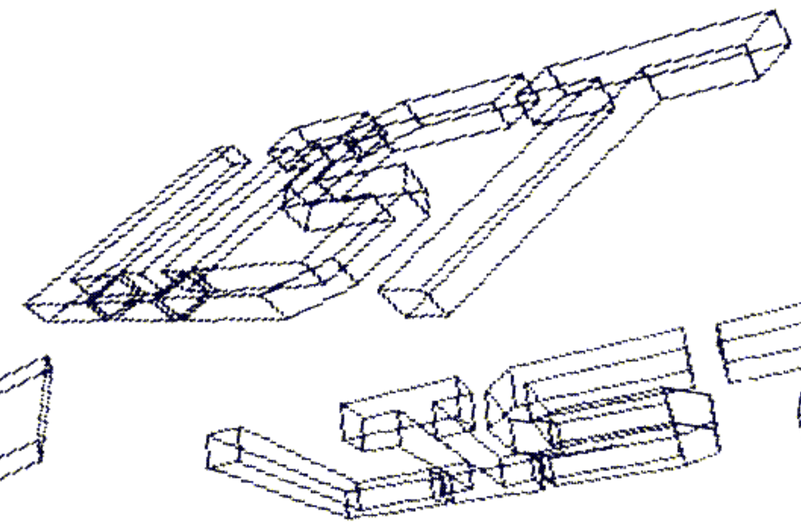


La démonstration alléchante du logiciel



Une création "maison"





démonstration. Donc une longue attente avant de pouvoir être actif. On peut regretter que les auteurs n'aient pas prévu une option pour implanter ces programmes sur microdrive. Cela aurait été très simple pour eux, alors qu'il est difficile de le faire soi-même. Il faut une bonne maîtrise du Spectrum et de son langage, et débusquer une à une les multiples embûches (programme Basic immergé sous forme de codes, codes de couleurs dans la liste du programme, désamorçage de la touche BREAK, etc.). Mais si on y arrive, quel agrément !

L'effet cinématographique

Après la démonstration, *3D Mover* est enfin tout à soi. Un menu de neuf options est proposé (dont la copie d'écran sur l'imprimante Sinclair, qui n'est pas répertoriée dans la notice).

Pour créer une figure, l'option 1 demande d'introduire une à une les trois coordonnées de points dans l'espace, en précisant si ce point doit être "PLOTé" ou bien relié par un trait au point précédemment défini (DRAW). Tout en introduisant les points ou vecteurs, on a la possibilité, au moyen des dix touches de chiffres et de Q et W, de positionner la figure que l'on est en train de définir pour une meilleure visualisation.

De retour au menu, vous pouvez demander l'option *Modification d'un vecteur* ou celle permettant de lister les vecteurs sur l'écran ou l'imprimante.

Deux options du menu servent à la communication avec le magnétophone, pour charger ou sauvegarder une figure. C'est un point qu'il vous faudra modifier si vous voulez adapter le programme aux microdrives.

L'option *Continuer une figure* permet de continuer la définition des vecteurs d'une figure et poursuit l'introduction des vecteurs après le dernier défini.

Avec *Utiliser 3D Mover*, les 12 touches précédemment citées vous entraîneront en pleine animation, par des rotations autour des trois axes, combinées ou non, des translations droite gauche, haut bas, avant arrière. Au cas où vous perdriez la figure au-delà de votre écran, l'option *Modification du cadrage* vous invite à faire une remise à zéro des variables de position (RAZ) ou à redéfinir les coordonnées du centre de rotation (SYM).

La particularité la plus importante et passionnante de l'utilisation de *3D Mover* est la possibilité d'enregistrer en mémoire (jusqu'à saturation de celle-ci) les mouvements de votre figure. Ceux-ci seront alors à votre ordre, restitués à bonne cadence ("cinématographique"). Vous pouvez même n'enregistrer qu'une image sur deux, trois, ..., pour obtenir un effet d'accéléré.

Le programme *3D Basic* se charge en cinq parties dont un affichage écran et deux pour la démonstration. Pour les utilisations ultérieures de *3D Basic*, il vous faudra patienter un peu car la démonstration vous sera encore imposée, avec un temps de chargement long.

Néanmoins, après la démonstration, il y a une réinitialisation par un NEW. Rassurez-vous, le code machine de *3D Basic* est bien protégé au-dessus de RAMTOP et l'accès se fait par RANDOMIZE USR 63490.

Les vecteurs utilisables par *3D Basic* sont préalablement définis et sauvegardés sur bande par *3D Mover*. L'adresse d'implantation de ces vecteurs peut être variable mais doit être précisée en initialisant une variable Basic DEFOB. D'autres variables Basic peuvent être définies pour commander l'animation d'une figure par *3D Basic* : les coordonnées de l'observateur, les arguments des rotations, le centre de rotation. Il est aussi possible de définir une fenêtre d'écran dans laquelle la figure s'animera. Les autres points de l'écran resteront invariants. Ceci permettra donc d'animer alternativement plusieurs figures sur l'écran en remplaçant à chaque fois les pointeurs par ceux de la figure à animer.

Le logiciel en quelques lignes

Nom : *3D Mover*
Ordinateur : *ZX Spectrum*
Forme : *Cassette*
Edité par : *Ere Informatique*
Prix public : *180 FF*
Principale orientation : *graphisme en trois dimensions*

Espace et mouvement

Les possibilités de *3D Basic* sont donc très larges et ouvertes. C'est votre programme Basic qui animera la ou les figures dans les directions définies par vous-même.

A noter que trois des feuillets de la notice vous donnent quelques éléments de géométrie tridimensionnelle pour vous aider à diriger votre figure par le Basic.

En conclusion, *3D Mover* vous ouvrira de façon simple l'univers de la troisième dimension. Il vous aidera à mieux percevoir ce que peut être la projection sur un écran d'un objet défini par ses trois dimensions et cela quel que soit le point de vue. *3D Basic* vous laissera l'initiative de l'animation, de façon moins rigide qu'avec *3D Mover*, en vous permettant d'incorporer des animations dans vos programmes Basic.

On peut regretter qu'il n'y ait pas d'option d'élimination des traits cachés. mais là, le problème est plus complexe.

Ces logiciels de bonne qualité permettent de comprendre une certaine vision des choses parfois difficile à appréhender.

Benoît THONNART

MASTER 64

UNE AIDE A LA PROGRAMMATION POUR COMMODORE 64

MASTER 64 se présente sous la forme d'une disquette protégée par une clé électronique. C'est un ensemble d'aide à la programmation qui est livré avec une documentation de qualité (150 pages). Le public visé n'est pas celui des débutants. Générateur d'écrans, générateur de fichiers, extension du Basic, moniteur, etc., le tout est impressionnant de possibilités et devrait combler les programmeurs chevronnés ou professionnels du Commodore 64.

On ne fait pas de la navigation de plaisance avec un porte-avions, ni de balade dominicale avec une moissonneuse-batteuse. Pourquoi ? Parce que ce sont des outils de travail destinés à l'amiral ou à l'agriculteur.

Master 64 est, lui aussi, un outil de travail. Il est avant toute chose destiné aux professionnels qui créent des programmes d'application.

Un client vous demande un fichier bien spécialisé pour gérer un terrain de camping, des cultures en laboratoire, ou une bourse d'échange de coquillages. Ce client n'a rien d'un analyste, ni d'une opératrice de saisie, et il lui faut un programme où les données soient faciles à entrer, où des messages d'aide puissent apparaître et disparaître au bon moment, où les erreurs possibles soient prévues et rattrapables.

Ce programme, vous allez le lui vendre. Comme vous ne souhaitez pas qu'il se le rembourse en vendant sous le manteau (avant de rire sous cape) des copies clandestines, vous voudrez que ce programme soit protégé.

Vous, concepteur professionnel, aurez donc dans ce cas un certain nombre de problèmes précis à résoudre. Si vous travaillez sur un Commodore 64 depuis trente ans déjà, aucune des ficelles ne vous sera étrangère, et en assemblant (rien ne se perd) des bouts de programmes que vous avez déjà faits, vous arriverez sans doute à un résultat satisfaisant.

Mais si, par hasard, tel n'était pas votre cas, il faudrait que vous consacriez un temps certain à explorer les mémoires mortes du C. 64 pour y découvrir les itinéraires optimaux.

Protection par clé électronique

Je ne pense pas trahir la pensée des créateurs de *Master* en disant qu'ils vous apportent ici sur un plateau toute leur science d'experts.

Un exemple, avant d'examiner le « concept » de face, de profil et de trois-quarts : quiconque a un peu ramé dans le SED (Système d'Exploitation des Disquettes) de Commodore a eu un petit pincement au cœur en se lançant dans l'accès direct à un secteur de la disquette. Il a potassé la syntaxe d'OPEN-dièse, du BLOCK-READ et du BLOCK-WRITE, compté et recompté ses pointeurs, posé 255 et-je-retiens-zun, et perdu quelques Ko de précieuses informations par erreur de jeunesse ! Sous *Master*, vous n'ouvrez rien, vous ne fermez rien, vous frappez trois instructions, et il s'occupe du reste. Que

Proscreen : un éditeur d'écran particulièrement perfectionné.

contient le secteur 1 de la piste 18 ? Vous frappez en mode direct si vous voulez :
INBLOCK 18,1:TAKBUF A$,0,255: PRINT A$, et c'est tout. Mais commençons par le commencement.

C'est en 1981, au Sicob, si mes souvenirs sont exacts, que Micro Application présentait à un public perplexe le concept *Master* sur CBM 8000. Une version 4000 existait aussi, bientôt suivie d'un *Master II*. Aujourd'hui, sur vos écrans, le retour de la revanche du fils du pont de la rivière Kwaï, entendez *Master 64* dernière mouture : un classeur corail au format A5 contenant une disquette, une clé électronique, et 150 pages d'explications.

Car Micro Application recourt au système de protection de son programme par clé électronique. La protection est un sujet qui a fait couler beaucoup d'encre et tout a été dit ou presque. Il reste que la clé électronique a ceci de bon qu'elle permet à l'utilisateur de faire autant de copies de sécurité du programme qu'il le souhaite, et c'est là un excellent point. Évidemment, comme ces copies ne peuvent tourner qu'en présence du « dongle » (un mille-pattes sur un bout de circuit imprimé qui se branche sur le port cassette), on évite l'apparition d'une diffusion sauvage. D'autres constructeurs prévoient une clé mâle/femelle, qui permet de conserver l'usage du magnétophone. Celle-ci est close. Tant pis.

Une petite démonstration ?

Quatre chapitres, Master Screen, Master Print, Master File et Master Basic, vous font découvrir les quatre talents de société du programme : la génération d'écran, la génération d'édition, la génération de fichiers et un Basic gonflé à bloc ; sans préjudice du Basic 4.0 (à la suite de quelle retombée en enfance Commodore l'a-t-il abandonné sur le 64 ?) et d'un petit moniteur langage-machine réduit, mais qui a le mérite d'être là.

Sur la disquette, deux programmes *Master* avec leurs chargeurs, l'un suffixé « D » et destiné au développement des programmes, l'autre suffixé « U » et destiné à l'utilisateur des programmes. C'est ce dernier qui figurera sur la disquette du produit fini que vous proposerez.

Pour qui découvre *Master*, des programmes de démonstration montrent l'effet des nouvelles commandes, mais le programme de test m'a presque paru plus démonstratif.

Pour le concepteur, toujours, un utilitaire de copie de fichiers et un de Backup (copie de sécurité) reproduisant une disquette secteur par secteur, moyennant de fréquentes manipulations.

Pour l'utilisateur final, pas question de lui compliquer la vie en lui demandant de charger *Master*, de le lancer, puis de charger le ou les programmes que vous aurez mis au point pour lui. Il faut que tout s'enchaîne. Mais là encore, le concepteur n'aura pas à chercher de solution, c'est prévu : le chargeur du programme-utilisateur chargera bien entendu les deux parties de *Master* proprement dit, mais il cherchera auparavant une page de titre, nommée Page 64, que vous aurez composée et qui fera une jolie image à se mettre sous l'œil le temps que le reste entre en mémoire. De plus, quand tout le système sera en place, ce même chargeur lancera automatiquement l'exécution d'un programme de Boot que vous aurez préparé, afin d'appeler votre application.

Tout a donc été préparé pour qu'un simple LOAD et un simple RUN suffisent à l'utilisateur final. Enfin, LOAD et RUN plus une minute cinquante pour charger *Master*, plus le temps de charger le programme que vous aurez mis au point. Un logiciel comme *Mercure*, (une gestion de fichier développée sous *Master* par Micro Application) ne met pas moins de deux minutes trente-quatre à se charger avant de pouvoir opérer. C'est un peu lourd, mais la faute n'en revient pas à Micro Application. L'unité de disquette 1541 a sur les cassettes l'avantage de l'accès direct, mais pas vraiment celui de la rapidité.

Master Screen, le générateur d'écran, reprend les instructions de *Tool* (voir l'article de Jacques Deconchat dans le numéro 1 de *LIST*, page 40). Pour le programmeur, les instructions de mise en page permettent de tracer de petites boîtes sans compter sur ses doigts, mais ce sont surtout les instructions de saisie qui trouveront ici leur plein emploi : convenablement utilisées, elles permettent d'éviter à l'utilisateur le moins expérimenté tout risque d'erreur et d'arrêt du programme. Les instructions de gestion de page-écran, quant à elles, permettent, comme leur nom l'indique, de préparer en mémoire des pages-écran numérotées de 1 à 127 et de les appeler une par une au moment voulu. Comme il s'agit d'un simple transfert de mémoire, l'opération est extrêmement rapide. On peut aussi appeler des écrans préalablement stockés sur disquettes,mais là on me pardonnera de ne pas m'étendre sur le temps d'accès.

Avec une modestie touchante, le manuel d'instructions cite, sans développer, le programme Proscreen présent sur la disquette, un éditeur d'écran particulièrement perfectionné, permettant de manipuler avec une grande facilité les formes et les couleurs, grâce aux touches de curseur et aux touches de fonction.

Equivalent « gutenberg-ien » d'un Master Screen cathodique, Master Print permet à l'utilisateur averti de remplir comme en se jouant le formulaire administratif le plus vicieux. De même qu'à l'écran on pouvait prévoir des zones de saisie ou de sortie de données, de même, ici, on peut créer et mettre en boîte un format d'impression selon un schéma préétabli, que viendront remplir ensuite les données prélevées sur un fichier. Que

```
list 1000-1200
 1000 rem trace de la grille
 1010 sclear
 1020 tline 40,1,1←40,3,1←40,6,1←40,11,1
←40,9,1←40,13,1
 1030 tcol 13,1,1←13,1,40
 1100 out "Fiche Cinema no.",2,2←ne$,2,1
9:rev 2,19,1,3
 1110 out "Titre:",4,2:rev 4,2,1,6
 1120 out "Realisateur:",7,2:rev 7,2,1,1
2
 1130 out "Annee",10,2←"Duree",10,10←"NB
/CO",10,16←"S/M",10,22←"VO/VF",10,26
 1135 out"Cassette",10,32
 1140 tcol 5,9,9←5,9,15←5,9,21←5,9,25←5,
9,31
 1150 rev 10,2,1,5←10,10,1,5←10,16,1,5←1
0,22,1,3←10,26,1,5←10,32,1,8
 1200 return

ready.
```

Un masque de saisie vu du côté du programmeur (Master screen)

```
Fiche Cinema no.  1

Titre:
Le Master de la chambre jaune

Realisateur:
Cecil B. de LIST

| Annee | Duree | NB/CO | S/M | VO/VF | Cassette |
|---|---|---|---|---|---|
| 1984 | 128 | Coul. | Son | | |

Frappez = pour un film francais
          F pour un film en VF
          O pour un film en VO
          S pour un film en VO sous-titree
```

Le même masque tel qu'il apparaît à l'utilisateur

demandez de plus ? Là encore, un tel luxe ne se justifie que pour un usage professionnel, mais si votre bien-aimé Directeur des Ventes, pris d'une rage créatrice funeste, modifie tous les quinze jours la disposition des imprimés de la boîte, vous échappez au surmenage ou la calvitie précoce : vous n'aurez que le fichier-format à modifier.

Moi-même-personnellement-tout-seul-je-pense-que, et d'ailleurs je le partage, un C.64 qui aurait en mémoire morte le Basic revu et corrigé par *Master* ferait un malheur sur le marché.

Le Basic 4.0 est là, avec son DIRECTORY qui ne détruit pas le programme, ses DSAVE et DLOAD sans suffixe 8, ses DOPEN et ses DCLOSE, ses SCRATCH, HEADER, COLLECT, APPEND, CONCAT, COPY et autres RENAME qui relèguent aux oubliettes les DOS-SUPPORT, WEDGE et l'ouverture du redoutable canal 15, sans parler des messages d'erreur en mode direct sur un simple PRINT DS$.

Une richesse impressionnante

Outre les instructions façon *Tool*, on trouve les opérations en multiprécision, GOTO et GOSUB calculés, inversion majuscules/minuscules, l'accès direct susmentionné, le NOLIST (un programme enregistré avec NOLIST au lieu de SAVE provoque un RESET si l'on tente un LIST : le curieux se décourage vite), un contrôle de date et la recherche de caractères dans une chaîne.

Ce n'est pas tout. Pour le même prix, on trouve aussi EDEX, (comme *Tool*), avec AUTO, DELETE, RENU, DUMP, ERROR, FIND, HARDCOPY, PLOT, RESET, PRINT USING, IF-THEN-ELSE, TRACE, et une désactivation de la touche STOP sur un bête POKE 1001,1.

Bref, un superadditif comme dans les poudres à laver (faites chanter le Basic de votre 64 !) et sans lequel on a l'impression de faire décoller Concorde avec un élastique.

Mais le point fort de *Master*, et sans doute sa raison d'être, ce sont les instructions relatives aux fichiers. Vingt instructions supplémentaires, six variables réservées, et vous voilà débarrassé des basses besognes du programmeur pour pouvoir vous concentrer sur les nobles tâches de l'analyste. Si vous avez une formation classique, la traduction d'un organigramme tant soit peu détaillé se fait presque naturellement, dès que vous avez absorbé le vocabulaire nouveau qui vous est proposé. Un ONERROR consacré aux messages disques vous permet de faire face à toutes les situations. Cela ne veut pas dire qu'il s'agisse d'un programme de fichier tout prêt : tel n'est pas le propos des auteurs, et vous devrez vous-même organiser vos rubriques, créer vos tris, utiliser les clés, c'est votre problème.

La gestion de fichier Master File a pour origine, dit le manuel, la méthode séquentielle indexée (ISAM). C'est à vous de définir, de créer et de prévoir l'utilisation du fichier grâce à vingt instructions supplémentaires et six variables réservées. De plus, une collection de programmes utilitaires vous permet non seulement, bien sûr, de définir un fichier, mais aussi de lire une mémoire-tampon ou le contenu d'un fichier sous forme de dump-hexa, de vérifier le numéro de série de *Master* (pour provoquer un RESET en cas de non-conformité, par exemple), de transférer le contenu d'un fichier vers un canal de sortie (édition brute de données sur imprimante), de transférer un fichier séquentiel sur un fichier *Master* (utile quand on veut restructurer un fichier prévu trop juste), et de régénérer les tables d'index.

En équilibre sur le bord du clavier, un verre d'eau chante : deux comprimés d'aspirine y pétillent gaiement, mais je viens de vivre une aventure. Le petit classeur corail se referme à regret. Ah, si j'étais pro !

D'un intérêt limité pour les informaticiens du dimanche, *Master* semble être pour des professionnels chevronnés un investissement rentable leur permettant de créer un logiciel aussi sophistiqué qu'ils le voudront, mais directement utilisable par des débutants.

François J. BAYARD

Le logiciel en quelques lignes

Nom : *Master 64*
Ordinateur : *Commodore 64*
Forme : *disquette avec clé électronique*
Edité par : *Micro Application Software*
Distribué par : *Procep*
Prix public : *550 FF*
Principales orientations : *génération d'écran, génération d'édition, génération de fichiers*
Autres orientations : *extension du Basic, moniteur*

LOGOMONDE
POUR TO 7/TO 7-70

LOGOMONDE est un manuel de 80 pages accompagné d'une cassette de programmes Logo tout prêts pour TO 7 et TO 7-70. Après l'avoir testé à plusieurs niveaux (enseignants dans leur classe et en formation, enfants, programmeurs), nous retenons que Logomonde fait partie des quelques documents Logo à posséder, même sans matériel Thomson. Hatier aura-t-il la bonne idée de le traduire pour les autres versions de Logo ?

Pour un spécialiste de la pédagogie Logo, le premier réflexe est de penser que *Logomonde* est le type de documents à ne pas faire. Les programmes Logo, en effet, se construisent à partir de projets personnels. Dans cette optique, donner aux enfants des programmes tout faits n'est pas souhaitable. Et pourtant...

Des réactions différentes

Dans le cadre d'une école, nous avons pu constater que *Logomonde* déclenchait deux réactions très différentes. La première est celle de l'instituteur ayant bien compris la pédagogie Logo. Il a utilisé le livre en essayant de trouver dans les exemples proposés des programmes analogues à ceux que ses élèves avaient réalisés. Cela lui a permis de leur présenter une autre approche ou une extension de leurs propres projets.

La seconde réaction est celle de l'instituteur concevant l'ordinateur comme un moyen de faire de l'enseignement assisté par ordinateur. Après avoir sélectionné quelques programmes donnés aux élèves consommateurs, il s'est aperçu que les élèves pouvaient aussi devenir autonomes en programmant eux-mêmes avec Logo grâce à la liste accompagnant chaque exemple.

Pour les enseignants en formation, *Logomonde* est un document « rassurant ». Les adultes ont envie de maîtriser rapidement Logo. Souvent, quand ils débutent, ils ont l'impression de perdre leur temps en réalisant des projets simples. Ils sont mieux disposés à progresser pas à pas lorsqu'ils ont feuilleté le livre de *Logomonde*. De plus, certaines pages sont très pédagogiques. L'ordre dans lequel sont proposées les procédures est un élément important : il a été bien pensé.

Côté programmation, nous rencontrons un problème inhérent à la presque totalité des documents écrits : les fautes de frappe, et parfois même les lignes manquantes... D'une façon générale, les espaces — qui sont un signe au même titre qu'une lettre dans les programmes Logo — ne sont pas assez marqués typographiquement. De plus, les erreurs semblent s'être donné rendez-vous à certaines pages (principalement pages 32 et 33). Pour éviter de tels inconvénients, il aurait sans doute été préférable de clicher une véritable liste de chaque procédure, telle qu'elle sort de l'imprimante.

Si l'on met de côté ces quelques griefs, la présentation du produit est agréable, la mise en page est excellente et la progression bien étudiée. *Logomonde* est donc un document utile, même pour ceux qui n'ont pas de machine ou ceux qui ont une machine autre que le TO 7 / TO 7-70.

Peut-être même Hatier serait-il bien inspiré de commercialiser le livre seul, ou mieux, une série de livrets reprenant les différentes versions du Logo car ce langage n'est pas l'apanage du seul Thomson, et le parc d'Apple, Commodore, Amstrad et autres matériels « Éducation nationale » est loin d'être négligeable lorsque le produit est bon. A moins de s'en tenir à un *Logomonde* Thomson et un dictionnaire Logo, ouvrage en cours d'édition dont nous reparlerons dans un prochain numéro.

Robert DAGUESSE

Le logiciel en quelques lignes

Nom : *Logomonde*
Ordinateur : *Thomson TO 7 / TO 7-70*
Forme : *cassette et livret d'accompagnement*
Auteurs : *M. Bonneton, G. Godimier et A. Pré*
Edité par : *Hatier*
Prix public : *185 FF*
Principale orientation : *utiliser Logo grâce à des programmes commentés*

HADÈS
POUR ATMOS ET ORIC-1

AVEC la cassette Hadès, Ère Informatique propose un ensemble cohérent de logiciels (assembleur-désassembleur, moniteur, sourceur, débogueur) conçus pour la programmation en Assembleur et en Langage-Machine.

En pratiquant Hadès 1.0, on s'aperçoit qu'il est possible de programmer en Assembleur avec presque autant de confort qu'en Basic.

Comme cela se passe pour tous les langages symboliques, un programme écrit en Assembleur (programme-source) doit nécessairement être traduit en Langage-Machine (programme-objet). Et c'est ici que la traduction en deux passes prend tout son intérêt : on peut, avec Hadès, utiliser pratiquement tous les types de labels (jusqu'à huit caractères), y compris les labels locaux qui prolifèrent vite avec ce genre de langage.

On retrouve en nombre réduit les commandes les plus connues : *Liste n1, n2* pour lister le programme ou une partie du programme, *Insert n1* ou *Insert* seul pour insérer à la suite, *Delete n1, n2, Modifie n1, n2.* On ne tape que la ou les deux premières lettres (qui figurent ici en majuscules) suivies des numéros de ligne. On trouve aussi quelques commandes simplifiées mais particulièrement utiles : *Save, LOad* et *Merge* pour sauvegarder, charger ou fusionner des programmes sources, et *BSave* pour le langage-machine (objet du programme). Pour ces quatre derniers ordres, la vitesse lente est possible en option. Notons encore *Print* qui imprime tout ce qui est affiché à l'écran : bien pratique, sinon indispensable, dans le cas des programmes longs.

Pour faciliter encore la programmation ou la mise au point, Hadès offre un éditeur plus performant et plus rapide que l'éditeur Basic : répétition accélérée des touches, recopie de la ligne au-dessus, recopie du reste de la ligne avec ou sans validation, utilisation des flèches, Ctrl-A, etc.

Il est agréable de constater que, pratiquement, la seule contrainte dans l'écriture d'une ligne est le caractère espace obligatoire pour distinguer le label de l'instruction qu'il désigne. Tout commentaire est introduit avec un point-virgule.

Cet Assembleur offre aussi des pseudo-instructions qui le rendent proche d'un grand Assembleur : origine et adresse du programme ou d'un label, réservation de mémoire à l'intérieur du programme, entrée de code ou de texte et tables d'adresses.

Bien entendu, la commande la plus importante est *Asm* qui provoque l'assemblage. La première passe traduit les labels et les changements d'origine. A l'issue de la seconde passe, on dispose du programme-source et soit du programme-objet, soit (personne n'est à l'abri des bogues) d'un ensemble de messages d'erreur en clair. Quand tout est au point, *Exec* suivie d'une adresse lance l'exécution du programme-objet.

Par ailleurs, l'Assembleur peut renvoyer à un programme Basic pour lequel le logiciel réserve 200 octets libres. Le retour à l'Assembleur se faisant par le signe du point d'exclamation, ce qui rend malheureusement impossible l'utilisation des microdisques. Pour ma part, il n'y a que l'Assembleur qui m'oblige à travailler encore avec des cassettes.

Un moniteur très complet

A côté de l'Assembleur proprement dit (que les auteurs ont d'ailleurs enrichi d'un mini-moniteur très simplifié mais suffisant), Hadès comporte également un moniteur nettement plus important qui permet d'exploiter des programmes déjà écrits.

Ce deuxième logiciel peut être chargé en même temps qu'un autre programme. Pour bien l'utiliser, il est nécessaire d'apprendre dans un premier temps le langage codé assez complexe de ses commandes. Ce moniteur est lui-même très complet et d'une utilisation facilitée par un éditeur aussi performant que celui de l'Assembleur.

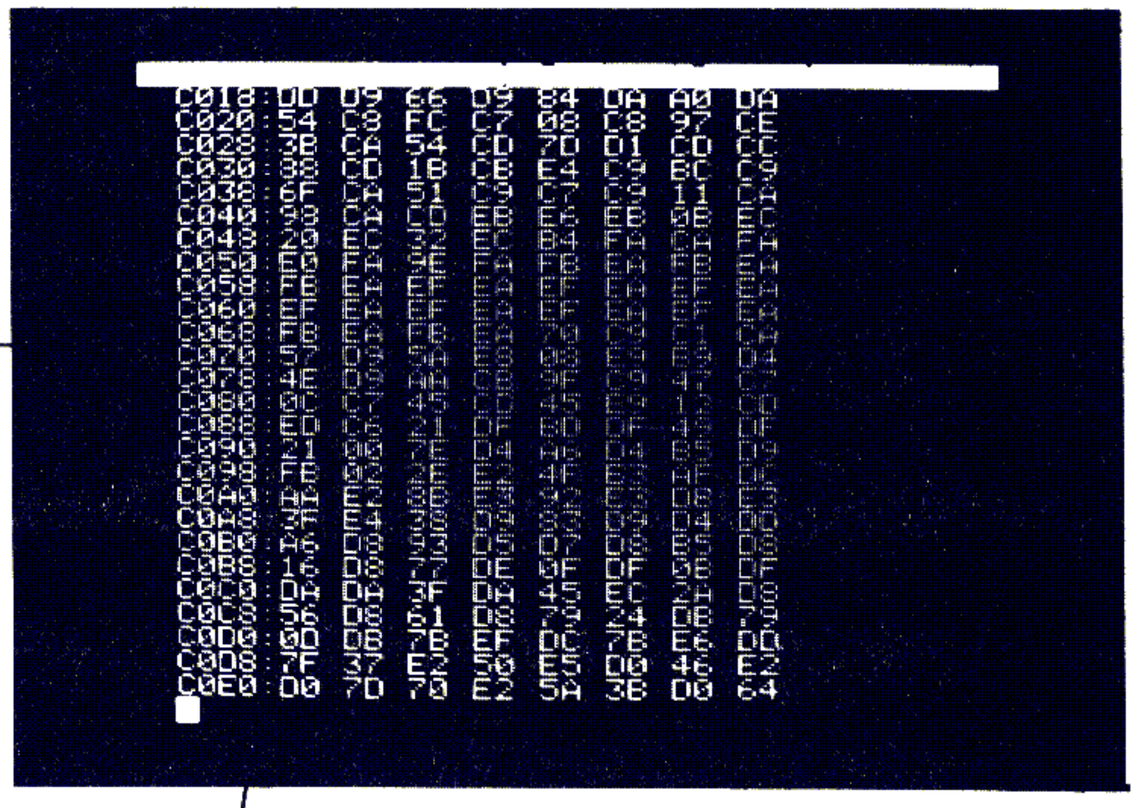
On trouve comme principales fonctions :

1. L'exploitation de la mémoire : visualisation, modification, recherche et déplacement de mémoire. A noter que

Exemple d'exécution d'un court programme écrit sous Hadès

```
ASM hades 1.0   Gruber&Bockelee  CAPS
:B
 10 DIM PR$(3,3)
 20 DOKE 0,ASC("P")+256*(ASC("R")+128)
 30 CALL #9000
 40 PRINT "adresse tableau "DEEK(2)
!
:L
  1           ORG $9000 ; (call)
  2           LDX $9E ;   pointeur-
  3           LDA $9F ;     tableaux
  4 TAbleau   STX $2 ;     range
  5           STA $3 ;     dans 02
  6           LDY #$0 ;   charge
  7           LDA ($2),Y ; 1er octet
  8           INY
  9           CMP $0 ;     compare
 10           BNE SUivant;   si <>
 11           LDA ($2),Y ; 2e octet
 12           CMP $1 ;     compare
 13           BEQ Fin ;    si trouve
 14 SUivant   INY
 15           LDA ($2),Y ;pointeur
 16           CLC ;         offset
 17           ADC $2 ;     + sur 02
 18           TAX ;        (pointe
 19           INY ;         tableau
 20           LDA ($2),Y ;  suivant)
 21           ADC $3 ;     + sur 03
 22           BCC TAbleau
 23 Fin       RTS ;     retour Basic
:A
FIN DE LA PASSE 1
Fin des labels:$2AA6
000         1       ORG $9000 ; (call)
FIN DE LA PASSE 2
Sortie des labels ?
TAbleau  =$9004     SUivant  =$9017
Fin      =$9025
                FIN= $9026
:B
Ready
RUN
adresse tableau =  1373
```

On reconnaîtra dans la liste, successivement :
- une partie Basic avec réservation d'un tableau et appel de la séquence en langage-machine ;
- une partie en Assembleur suivie de l'assemblage et des labels ;
- l'exécution du programme (recherche de l'adresse du tableau PR$ dans la mémoire centrale).



le mnémonique de la commande est placé derrière les données manipulées.
2. Un Assembleur simplifié à une passe (sans possibilités d'utiliser des labels).
3. Un débogueur qui exécute les programmes pas à pas.
4. Un sourceur enfin qui, à l'inverse de l'Assembleur, convertit un programme en langage-machine, en un texte Assembleur. Il utilise deux passes dont la première crée des labels. De plus, le programme-source est automatiquement stocké sur cassette.

Si le manuel d'emploi est assez détaillé (une quarantaine de petites pages), quelques points délicats manquent de clarté. Toutefois, après une bonne lecture, on s'y retrouve.

En conclusion, on peut conseiller Hadès aux programmeurs qui désirent s'attaquer sérieusement à l'Assembleur. Le prix, environ 250 FF, est justifié en regard des services que ce logiciel peut rendre.

Le logiciel en quelques lignes

Nom : *Hadès 1.0*
Ordinateur : *Atmos/Oric-1*
Auteur : *F. Bochelée et O. Gauber*
Édité par : *Ère Informatique*
Prix public : *250 FF*
Principale orientation : *programmation en Assembleur et en langage-machine (assembleur-désassembleur, moniteur, sourceur, débogueur)*

Terminons en signalant que la couleur retenue pour l'affichage ne sera pas du goût des utilisateurs d'écrans noir et blanc : le rouge, en effet, ne passe pas très bien... Aucun problème en revanche sur un écran couleur.

Max HAGENBURGER

UN PROGRAMME SOUS LES FLASHES

TRANSPOSER un programme de l'Apple IIe au PB-700, ou le contraire, c'est une idée qui peut paraître saugrenue, mais cela s'avère parfois nécessaire. Ces deux machines ont bien peu de points communs, l'essentiel est qu'elles « parlent » l'une et l'autre le Basic. Pour mettre en lumière certains des problèmes posés par une telle transposition, on a choisi ici un programme destiné aux photographes amateurs.

Les problèmes rencontrés lors de la transposition d'une machine à une autre sont de plusieurs types : langage, configuration, mémoire, vitesse, possibilités de calcul, affichage, etc. Ceci reste vrai pour la transposition d'un programme de l'Apple IIe au PB-700. Le programme qui sert de cobaye ici regroupe différents calculs se rapportant aux nombres-guides de flash.

Un menu à six options

Au chapitre des points communs, on trouve la conception du programme. A ce niveau, les particularités du langage ou de la configuration n'entrent pas en ligne de compte : le programme est relativement simple et il ne fait pas appel à des fonctions spéciales telles que les fichiers séquentiels par exemple.

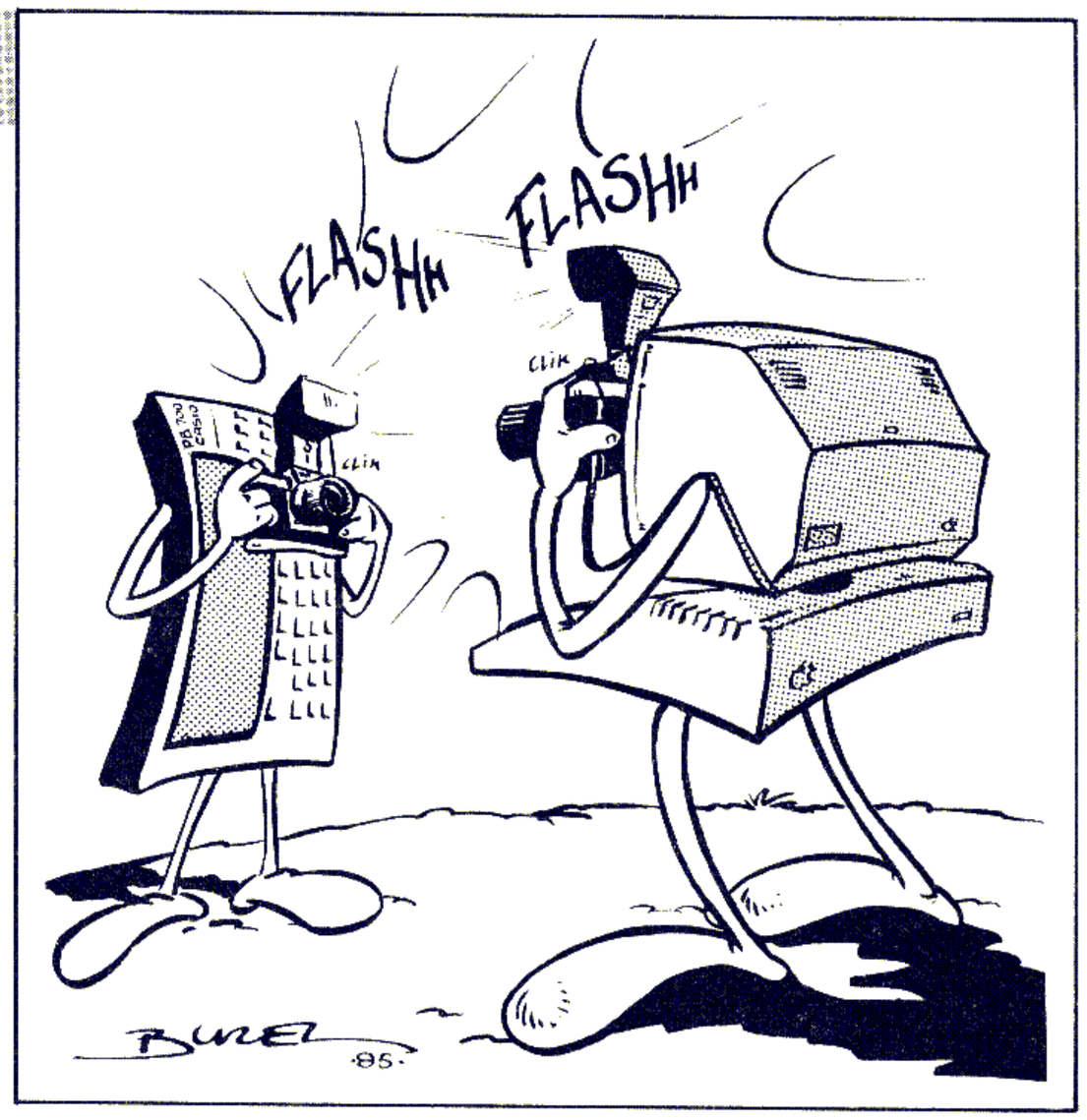
Son but est de permettre à un utilisateur de calculer rapidement les éléments suivants :

- nombre-guide d'un flash,
- diaphragme connaissant le nombre-guide et la distance flash-sujet,

Sous les flashes
Programme pour Apple IIe
Auteur Pierrick Moigneau
Copyright *LIST* et l'auteur

```
10  REM
20  REM  PROGRAMME NG FLASH
30  REM  ------------------
40  HOME
50  HTAB 6
60  PRINT "MENU PRINCIPAL NG FLASH"
70  VTAB 4
80  PRINT "A-CALCUL DU NOMBRE-GUIDE D'UN FLASH"
90  PRINT
100  PRINT "B-CALCUL DU DIAPHRAGME CONNAISSANT LE NG ET LA DISTANCE"
110  PRINT
120  PRINT "C-CALCUL DE LA DISTANCE CONNAISSANT LE  NG ET LE DIAPHRAGME"
130  PRINT
140  PRINT "D-CALCUL DU NOMBRE-GUIDE EN FONCTION DE LA SENSIBILITE"
150  PRINT
160  PRINT "E-CALCUL DE LA DISTANCE FLASH-SUJET EN FONCTION DU NG ET DU
GRANDISSEMENT"
170  PRINT
180  PRINT "F-CALCUL DU NOMBRE-GUIDE DE DEUX FLASHES COUPLES"
190  PRINT
200  PRINT "VOTRE CHOIX(A,B,C,D,E,F)?"
210  GET CH$
220 X = ( ASC (CH$) - 64)
225  IF (X > 6) OR (X < 1) THEN 200
230  ON X GOTO 250,410,560,720,900,1070
240  REM
250  REM  1/NOMBRE-GUIDE
260  REM  ---------------
270  HOME
280  HTAB 5: PRINT "CALCUL DE NOMBRE-GUIDE"
290  VTAB 5
300  INPUT "SENSIBILITE ISO DU FILM D'ESSAI? ";S
310  PRINT
320  INPUT "VALEUR DU DIAPHRAGME CORRECT? ";DP
330  PRINT
340  INPUT "DISTANCE FLASH-SUJET?";DS
350 NG = DP * DS: VTAB 12
360 NB = NG: GOSUB 1280
370 NG = NB
380  PRINT "NOMBRE-GUIDE DU FLASH: ";NG;" Pour ";S;" ISO"
390  GOSUB 1250: GOTO 20
400  REM
410  REM   2/DIAPHRAGME
420  REM   -------------
430  REM
440  HOME : HTAB 5
```

```
450  PRINT "CALCUL DU DIAPHRAGME"
460  VTAB 5
470  INPUT "NOMBRE-GUIDE?";NG
480  PRINT
490  INPUT "DISTANCE FLASH-SUJET?";DS
500 DP = NG / DS
510 NB = DP: GOSUB 1280
520 DP = NB
530  PRINT
540  PRINT "DIAPHRAGME: ";DP: GOSUB 1250: GOTO 20
550  REM
560  REM  3/DISTANCE
570  REM  ----------
580  HOME
590  HTAB 5
600  PRINT "CALCUL DE DISTANCE"
610  VTAB 5
620  INPUT "NOMBRE-GUIDE? ";NG
630  PRINT
640  INPUT "DIAPHRAGME DESIRE? ";DP
650 DS = NG / DP
660 NB = DS: GOSUB 1280
670 DS = NB
680  PRINT
690  PRINT "DISTANCE FLASH-SUJET: ";DS;" metres"
700  GOSUB 1250: GOTO 20
710  REM
720  REM  4/SENSIBILITE
730  REM  -------------
740  HOME
750  HTAB 5
760  PRINT "CALCUL DE NG SELON SENSIBILITE"
770  VTAB 5
780  INPUT "SENSIBILITE ISO DE DEPART?";S1
790  PRINT
800  INPUT "NOMBRE GUIDE CORRESPONDANT? ";NG
810  PRINT
820  INPUT "NOUVELLE SENSIBILITE? ";S2
830 N2 = NG * ( SQR (S2 / S1))
840 NB = N2: GOSUB 1280
850 N2 = NB
860  PRINT
870  PRINT "NOMBRE-GUIDE POUR ";S2;" ISO : ";N2
880  GOSUB 1250: GOTO 20
890  REM
900  REM  5/GRANDISSEMENT
910  REM  ---------------
920  HOME
930  PRINT "CALCUL DISTANCE FLASH-SUJET EN FONCTION DU GRANDISSEMENT ET
DU NOMBRE-GUIDE"
940  VTAB 5
950  INPUT "NOMBRE-GUIDE DU FLASH?";NG
960  PRINT
970  INPUT "DIAPHRAGME SOUHAITE? ";DP
980  PRINT
990  INPUT "GRANDISSEMENT?";G
1000 DS = (NG * 100) / (DP * (G + 1))
1010  VTAB 12
1020 NB = DS: GOSUB 1280
1030 DS = NB
1040  PRINT "DISTANCE FLASH-SUJET: ";DS;" Centimetres"
1050  GOSUB 1250: GOTO 20
1060  REM
1070  REM  6/FLASHES COUPLES
1080  REM  ------------------
1090  HOME
1100  HTAB 5
1110  PRINT "CALCUL DU NG DE DEUX FLASHES COUPLES"
1120  VTAB 5
1130  INPUT "NG FLASH NO1 ?";N1
1140  PRINT
1150  INPUT "NG FLASH NO2 ?";N2
1160 NG =  SQR ((N1 * N1) + (N2 * N2))
1170  PRINT
1180 NB = NG: GOSUB 1280
1190 NG = NB
1200  PRINT "NOMBRE-GUIDE RESULTANT: ";NG
1210  GOSUB 1250: GOTO 20
1220  REM
1230  REM  SOUS-PROGRAMME DE TEMPORISATION
1240  REM  -------------------------------
1250  FOR I = 1 TO 5000: NEXT I: RETURN
1260  END
1270  REM
1280  REM  SOUS-PROGRAMME DE FORMATAGE
1290  REM  ---------------------------
1300 K =  INT (NB * 10)
1310 K = K / 10
1320 NB = K
1330  RETURN
1340  END
```



- distance flash-sujet pour un diaphragme et un nombre-guide donnés,
- nombre-guide en fonction de la sensibilité du film,
- distance flash-sujet pour un diaphragme et un grandissement donnés,
- nombre-guide de deux flashes couplés.

Chaque calcul est réalisé à l'aide de formules classiques dont la principale est celle-ci : nombre-guide = diaphragme × distance (le nombre-guide sert à définir la puissance d'un flash de type amateur).

Les six calculs sont indépendants, chacun d'eux forme un petit programme séparé que l'on choisit avec un menu.

La structure va donc être très claire : un menu propose six options ; chaque option appelle un calcul différent ; l'utilisateur obtient le résultat voulu après avoir fourni les données numériques nécessaires ; ce résultat s'affiche quelques secondes, puis l'on revient au menu.

Sur Apple, le menu ne pose pas de problème particulier : le positionnement du titre et de la première ligne sur l'écran se fait par les instructions HTAB et VTAB, tabulation horizontale et tabulation verticale, allant de 1 à 24 pour VTAB et de 1 à 40 pour HTAB (de 1 à 80 avec la carte 80 colonnes). Le nom de ces instructions peut varier d'un matériel à l'autre (LOCATE, CURSOR, etc.), mais ces fonctions existent obligatoirement dès lors qu'il y a plus d'une ligne d'affichage.

Les textes de nos six options possibles tiennent sans problème sur l'écran ;

UN PROGRAMME SOUS LES FLASHES

chacun d'eux est précédé d'une lettre (A, B, C, D, E, F) servant à faire le choix. L'utilisateur n'a plus qu'à taper au clavier cette lettre qui est recueillie par l'instruction GET suivie du nom de la variable « réceptrice », CH$. En faisant la soustraction *code ASCII de la lettre – 64*, on obtient un chiffre compris entre 1 et 6. Ce chiffre, mémorisé dans la variable X, va permettre de faire le branchement vers les différents calculs par l'expression ON X GOTO < numéros de ligne > .

Formater les résultats

Pour le PB-700, ce n'est pas aussi simple... Il faut abréger les éléments du menu, car l'écran à cristaux liquides ne comprend que quatre lignes de 20 caractères. On doit également faire défiler les lignes ; pour cela on a choisi un défilement continu, dont le rythme est défini par une temporisation placée en sous-programme.

Le choix d'une option se fait, là aussi, par une lettre recueillie par la pseudo-variable INKEY$. Cette dernière a le même rôle que le GET de l'Apple, même si la syntaxe en est un peu différente. Là encore, la lettre devient un chiffre avec l'expression : X = ASC(CH$) – 64.

On peut remarquer cette fois que nous avons exactement la même expression pour les deux versions. L'instruction ON...GOTO n'existe pas sur le PB-700. En revanche, la disposition de la mémoire utilisateur en 10 zones (classique chez Casio) se prête bien à ce type de programme ; nous mettrons donc la partie « menu » dans la zone P0, puis chaque programme de calcul dans les zones allant de P1 à P6.

Notons au passage que, chaque zone fonctionnant de façon parfaitement autonome, les programmes pourront être testés séparément.

Toujours pour le PB-700, le branchement à la suite du menu devient on ne peut plus simple : c'est l'expression GOTO PROG X qui renvoie à la zone de programme numéro X.

Entrons maintenant dans les phases

Sous les flashes
Programmes pour PB-700
Auteur Pierrick Moigneau
Copyright *LIST* et l'auteur

```
P0
  10 REM *--PROGRAMME NG FLASH--*
  20 CLEAR :CLS
  30 PRINT "  MENU NG FLASH"
  40 GOSUB 500
  50 LOCATE 0,2
  60 PRINT "A/CALCUL NBRE-GUIDE"
  70 GOSUB 500
  80 PRINT "B/CALCUL DIAPHRAGME"
  90 GOSUB 500
 100 PRINT "C/CALCUL DISTANCE"
 110 GOSUB 500
 120 PRINT "D/CALC.NG/SENSIBTE"
 130 GOSUB 500
 140 PRINT "E/DIST.FL/GRANDISMT"
 150 GOSUB 500
 160 PRINT "F/CALC.NG.2 FLASHES"
 170 GOSUB 500
 180 GOTO 50
 200 X=ASC(CH$)-64
 210 IF X<1 THEN 50 ELSE IF X>6 THEN 50
 220 GOTO PROG X
 500 FOR I=1 TO 80:NEXT I
 510 CH$=INKEY$:IF CH$<>"" THEN 200 ELS
     E RETURN
 520 END
P1
  10 REM 1/NOMBRE-GUIDE
  20 REM ------------
  30 CLS
  40 PRINT "CALCUL NOMBRE-GUIDE"
  50 INPUT "Sensibte ISO";S
  60 INPUT "Diaph. correct";DP
  70 INPUT "Dist.flash-sujet";DS
  80 NG=DP*DS
  90 NG=ROUND(NG,-2)
 100 CLS
 110 PRINT "NOMBRE-GUIDE: ";NG
 120 PRINT
 130 PRINT " (pour ";S;" ISO)"
 140 GOSUB PROG 9
 150 RETURN
 160 END
P2
  10 REM 2/DIAPHRAGME
  20 REM ------------
  30 CLS
  40 PRINT "CALCUL DU DIAPHRAGME"
  50 INPUT "NBRE-GUIDE";NG
  60 INPUT "DIST.FLASH-SUJET";DS
  70 DP=NG/DS
  80 DP=ROUND(DP,-2)
  90 CLS
 100 LOCATE 0,2
 110 PRINT "DIAPHRAGME:";DP
 120 GOSUB PROG 9
 130 RETURN
 140 END
P3
  10 REM 3/DISTANCE
  20 REM ----------
  30 CLS
  40 PRINT "CALCUL DE DISTANCE"
  50 INPUT "NOMBRE-GUIDE";NG
  60 INPUT "DIAPH.DESIRE";DP
  70 DS=NG/DP
  80 DS=ROUND(DS,-2)
  90 CLS
 100 PRINT "DIST.FLASH-SUJET:"
 110 PRINT " ";DS;" metres"
 120 GOSUB PROG 9
 130 RETURN
 140 END
P4
  10 REM 4/NG SELON SENSIBLTE
  20 REM -------------------
  30 CLS
  40 PRINT "NG SELON SENSIBLTE"
  50 INPUT "  Sensibilite ISO de depart
     ";S1
  60 INPUT "NG correspondant";NG
  70 INPUT "Nouv. sensibte";S2
  80 N2=NG*(SQR(S2/S1))
  90 N2=ROUND(N2,-2)
 100 CLS
 110 PRINT " NOMBRE-GUIDE "
 120 PRINT "pour ";S2;" ISO: ";N2
 130 GOSUB PROG 9
 140 RETURN
 150 END
P5
  10 REM DIST/GRANDISSEMENT
  20 REM ------------------
  30 CLS
  40 PRINT "DISTANCE FLASH-SUJET SELON
     GRANDISSEMENT"
  50 INPUT "Nbre-guide flash";NG
  60 INPUT "Diaph.souhaite";DP
  70 INPUT "Grandissement";G
  80 DS=(NG*100)/(DP*(G+1))
  90 DS=ROUND(DS,-2)
 100 CLS
 110 PRINT "DISTNCE FLAH-SUJET:"
 120 PRINT " ";DS;" centimetres"
 130 GOSUB PROG 9
 140 RETURN
 150 END
P6
  10 REM FLASHES COUPLES
  20 REM ---------------
  30 CLS
  40 PRINT "   CALCUL DU NOMBRE-GUIDE D
     E 2 FLASHES"
  50 INPUT "NG flash no 1";N1
  60 INPUT "NG flash no 2";N2
  70 NG=SQR((N1*N1)+(N2*N2))
  80 NG=ROUND(NG,-2)
  90 CLS
 100 PRINT "NOMBRE-GUIDE TOTAL:"
 110 PRINT "    ";NG
 120 GOSUB PROG 9
 130 RETURN
 140 END
P9
  10 REM Temporisation
  20 FOR I=1 TO 500:NEXT I
  30 CLS
  40 RETURN
```

de calcul. Au départ, changer simplement le HOME de l'Apple en CLS, pour le PB-700, effacera l'écran. Nous avons ensuite des instructions et des symboles très classiques que l'on retrouve sur l'Apple et sur le PB-700 : INPUT, GOSUB, SQR, etc. Ce qui prouve que le Basic, malgré ses innombrables variantes, possède quand même un « tronc commun » reconnu par la plupart des machines.

Les problèmes arrivent quand il s'agit de présenter les résultats : il faut supprimer les décimales qui ne servent à rien pour cette application. Le PB-700, fort de son expérience de « super-calculatrice », propose deux solutions pour cela : ROUND et USING. La première arrondit un nombre à partir de l'endroit qu'on lui indique : ROUND(NB, – 3) va arrondir le nombre NB en supprimant toutes les décimales à partir de la troisième, et en effectuant un arrondi sur la deuxième. USING, utilisé avec PRINT, précise le nombre de chiffres à afficher au moyen de symboles « dièses » : PRING USING "###.##" NB.

Mais du côté de l'Apple, c'est le désert. Aucune instruction ne permet le formatage, il faut passer par un petit sous-programme pour y parvenir. La célèbre pomme présente là une faiblesse dans son Basic. Elle se fait damer le pion par des appareils moins chers, mais de conception plus récente (dans le même ordre d'idée on pourrait parler de l'éditeur de l'Apple, qui n'est pas vraiment un modèle de simplicité).

Temporisons, temporisons ...

Maintenant que nous avons notre résultat et que celui-ci est formaté, nous voulons l'afficher quelques secondes à l'écran. Sur ce plan, les deux ordinateurs ont la même lacune : pas de temporisation d'affichage de type WAIT ou PAUSE.

Il faut donc passer par une boucle de temporisation, placée en sous-programme. Celui-ci devra être bien repéré par des REM : ainsi les possesseurs de PC-1500 par exemple verront tout de suite qu'ils peuvent s'en passer grâce à WAIT. Attention, pour ces boucles de temporisation, il y a de grosses différences de vitesse de traitement d'un ordinateur à l'autre. Sur un Apple, une boucle FOR I = 1 TO 5000 affiche un résultat pendant six secondes environ. Pour une durée semblable, la boucle FOR I = 1 TO 500 est suffisante sur le PB-700.

En résumé, nous avons pu voir sur cet exemple quelques-unes des règles à respecter pour écrire un programme facilement adaptable d'un ordinateur à l'autre :

- d'abord, bien connaître le matériel de base, celui sur lequel sera conçu le programme au départ, comment le situer par rapport aux autres ; quels sont ses atouts et ses points faibles ; quelles sont ses particularités ; son Basic est-il « standard » ou non ?
- utiliser au maximum dans la version initiale du programme le noyau de Basic accepté par la majorité des machines, les INPUT, GOSUB, PRINT, FOR NEXT, etc. Même si dans certains cas cela fait une programmation un peu lourde ;
- ne pas utiliser les codes internes avec PEEK et POKE : il n'y a en effet aucune chance pour que les adresses soient les mêmes sur deux matériels différents. Si vraiment on ne peut pas s'en passer, expliquer leur usage par des REM (remarques) détaillées ;
- au niveau des différences de Basic, on peut les classer en deux catégories : d'une part les mots clés distincts qui correspondent à une même fonction (par exemple CLS et HOME) ; d'autre part les fonctions qui n'existent pas sur certains matériels et qu'il faut donc remplacer par un sous-programme (par exemple le formatage de résultats sur Apple).

Dans tous les cas, il sera nécessaire de repérer ces différences par des REM et si possible de les isoler sur une ligne ; éventuellement, d'indiquer les transformations à faire sur ces lignes dans un document joint au programme.

D'une manière générale, les différentes phases doivent apparaître clairement sur la liste du programme, les transformations sont alors plus faciles parce que bien délimitées. Dans notre exemple, le menu devait être complètement refondu à cause des différences d'affichage, il fallait donc que la partie « menu » du programme soit bien séparée du reste.

On retrouve finalement des impératifs bien connus : programmation structurée, « documentation » du programme, écriture simple et « aérée ». Tout cela répond à un vieux principe qui n'est pas seulement destiné à l'informatique : « ce qui se conçoit bien s'énonce clairement ».

Pierrick MOIGNEAU

Mode d'emploi du programme NG FLASH

La plupart des flashes sont vendus pour un nombre-guide donné à 100 ISO ; vous pouvez vérifier celui-ci en prenant plusieurs clichés d'un sujet de valeur moyenne à des diaphragmes différents et à une distance fixe de l'ordre de 2-3 mètres (1). Le premier calcul du programme vous donnera le vrai nombre-guide ; de même, si vous voulez utiliser votre flash dans des conditions spéciales, par exemple avec un réflecteur gonflable pour adoucir les ombres (2) :

Sensibilité ISO du film d'essai ? 100
Valeur du diaphragme correct ? 11
Distance flash-sujet ? 2
La réponse apparaît : 22 pour 100 ISO.

Si maintenant vous vous posez la question de savoir ce que vous pouvez faire avec le même flash et un film de 1600 ISO, commencez par calculer le nouveau nombre-guide :

(Programme D)
Sensibilité ISO de départ ? 100
Nombre-guide correspondant ? 22
Nouvelle sensibilité ? 1600
Réponse donnée : NG 88 pour 1600 ISO.

Nous pouvons utiliser ce dernier résultat avec le programme C, sachant que l'ouverture maximale de notre objectif est 2.8, pour trouver la portée théorique du flash avec ce film et cet objectif :

(Programme C)
Nombre-guide ? 88
Diaphragme désiré ? 2.8
Distance flash-sujet : 31.4 m

(1) La prise de vue en question sera faite de préférence avec un film inversible (diapositives) dont la faible latitude d'exposition permettra de mieux repérer le bon diaphragme.
(2) Le nombre-guide d'un flash n'est établi que pour son réflecteur d'origine dans des conditions de réflexion ambiante précises et pour des distances supérieures à 1 m. En dessous de cette distance, pour des raisons de focalisation de l'éclair, le nombre-guide baisse. Il sera donc intéressant de calculer, avec ce programme, les nombres-guides applicables en « proxiphotographie » (photographie rapprochée).

LOGO

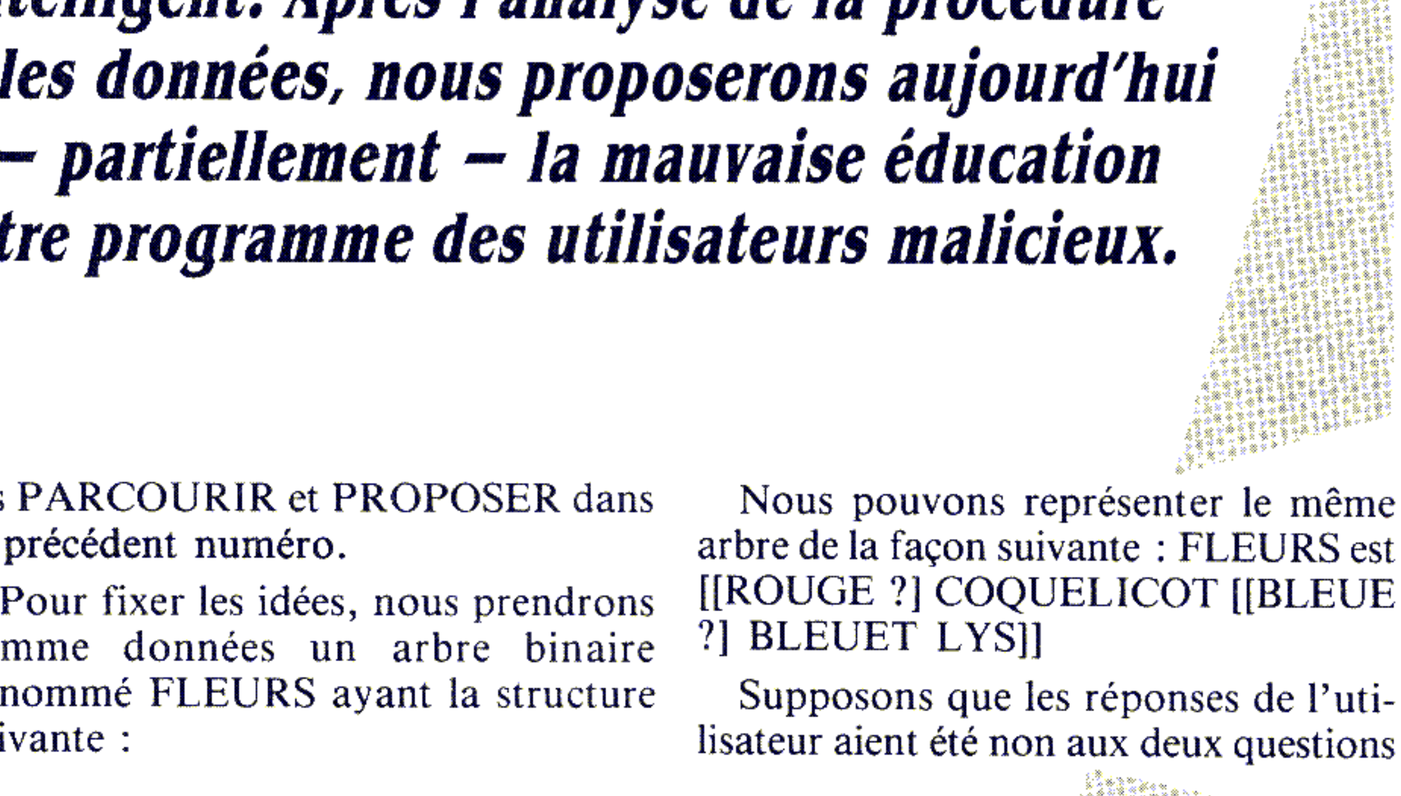
ÉVOLUER DANS LES ARBRES BINAIRES

LA construction et l'exploration des arbres binaires permettent d'écrire des procédures du type de celles qui servent en "intelligence artificielle". Chaque utilisateur enrichit le programme qui s'éduque, comme s'il était vraiment intelligent. Après l'analyse de la procédure qui permet de faire évoluer les données, nous proposerons aujourd'hui des solutions pour déjouer – partiellement – la mauvaise éducation que pourraient donner à notre programme des utilisateurs malicieux.

Reprenons le jeu des questions et réponses que nous avions abordé dans *LIST* 7. On se souvient que tout arbre binaire est essentiellement composé de nœuds (qui sont autant de questions) et de feuilles (qui sont autant de réponses et constituent les limites de l'arbre). Si une feuille n'est suivie de rien, un nœud en revanche est toujours suivi de deux éléments, soit deux autres nœuds, soit deux feuilles, soit un nœud et une feuille. Le premier de tous les nœuds, qui n'est précédé par rien, est appelé racine.

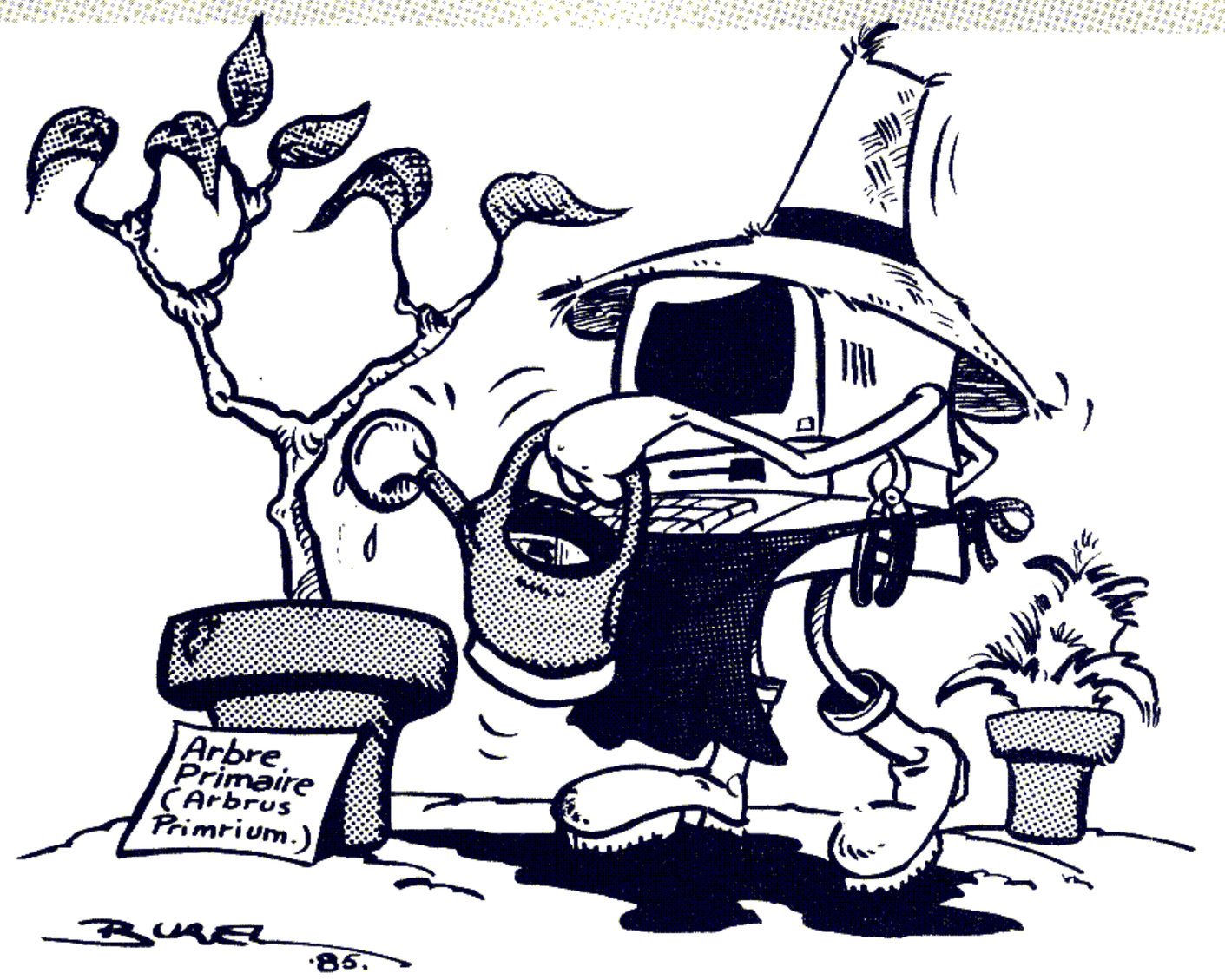
Dans l'exemple que nous avions retenu, le but de l'exploration de l'arbre binaire est de faire deviner par la machine à quelle fleur l'utilisateur pense. Si cette fleur n'est pas encore inscrite dans l'arbre, le joueur fournit les éléments nécessaires pour compléter les connaissances de la machine. A cette fin, nous avons déjà écrit les procédures PARCOURIR et PROPOSER dans le précédent numéro.

Pour fixer les idées, nous prendrons comme données un arbre binaire dénommé FLEURS ayant la structure suivante :



Nous pouvons représenter le même arbre de la façon suivante : FLEURS est [[ROUGE ?] COQUELICOT [[BLEUE ?] BLEUET LYS]]

Supposons que les réponses de l'utilisateur aient été non aux deux questions



(“rouge ?” et “bleue ?”) et que, par conséquent, la réponse LYS ait été proposée. Supposons encore que la fleur à laquelle il pense est celle du pissenlit. La bonne question pour distinguer le lys du pissenlit est ici “jaune ?”. Il s’agit donc, dans l’arbre binaire, de remplacer la feuille LYS par le nœud [[JAUNE ?] PI SSENLIT LYS].

Reconstituer les branches

Notre arbre devient alors : FLEURS est [[ROUGE ?] COQUELICOT [[BL EUE ?] BLEUET [[JAUNE ?] PISSEN LIT LYS]]].

Exprimons le problème en français. Pour compléter l’arbre, il suffit de le parcourir jusqu’à une feuille. Chaque feuille fait remonter au nœud précédent en reconstituant la branche qui aboutissait à cette feuille. Cependant, lorsque nous tombons sur la feuille à remplacer, la branche est reconstituée avec le nouveau nœud et ses deux feuilles dont l’une est nouvelle.

Passons en Logo maintenant. Pour simplifier les écritures, nous noterons A l’arbre, F la feuille à remplacer, et B la nouvelle branche à ajouter.

```
POUR COMPLETER :A :F :B
SI :A = :F [RETOURNE :B]
SI MOTP :A [RETOURNE :A]
RETOURNE (LISTE PREMIER :A COM
   PLETER PREMIER SP :A :F :B COM
   PLETER DERNIER :A :F :B)
FIN
```

Exécutons maintenant pas à pas la procédure en poursuivant notre exemple.

```
COMPLETER :FLEURS “LYS [[JAUNE
   ?] PISSENLIT LYS]
:FLEURS est [[ROUGE ?] COQUELICOT
   [[BLEUE ?] BLEUET LYS]]
:FLEURS n’est pas “LYS
:FLEURS n’est pas un mot
```

Résultat : [[ROUGE ?] COMPLETER “CO QUELICOT “LYS [[JAUNE ?] PISSEN LIT LYS] COMPLETER [[BLEUE ?] BLEUET LYS] “LYS [[JAUNE ?] PIS SENLIT LYS]

```
COMPLETER “COQUELICOT “LYS
   [[JAUNE ?] PISSENLIT LYS]
:FLEURS n’est pas “LYS
:FLEURS est le mot COQUELICOT
```

Résultat : COQUELICOT, qui est retourné, termine la première procédure COMPLETER et fait passer au niveau supérieur.

```
COMPLETER [[BLEUE ?] BLEUET LYS]
   “LYS [[JAUNE ?] PISSENLIT LYS]
:FLEURS n’est pas “LYS
:FLEURS n’est pas un mot
```

Résultat : [[BLEUE ?] COMPLETER “BLEUET “LYS [[JAUNE ?] PISSEN LIT LYS] COMPLETER “LYS “LYS [[JAUNE ?] PISSENLIT LYS]

```
COMPLETER “BLEUET “LYS
   [[JAUNE ?] PISSENLIT LYS]
```

Résultat : “BLEUET et passage au niveau supérieur.

```
COMPLETER “LYS “LYS [[JAUNE
   ?] PISSENLIT LYS]
:FLEURS est “LYS
```

Résultat [[JAUNE ?] PISSENLIT LYS] et passage au niveau supérieur, c’est-à-dire fin de l’exploration dans notre cas. Attention, la procédure est terminée parce que l’on a fini d’explorer l’arbre, et non pas parce que LYS a été trouvé.

Résultat final :
[[ROUGE ?] COQUELICOT [[BLEUE ?] BLEUET [[JAUNE ?] PISSENLIT LYS]]].
Conforme à ce qui était prévu.

Ainsi, lorsque l’arbre binaire a été défini, notre problème initial se résoud en quatre procédures principales qui ont été reproduites dans l’encadré ci-dessous.

Pour améliorer ces procédures, on peut, bien entendu, adapter les dialogues, mais les véritables perfectionnements ont trait aux réponses que peut

La solution en quatre procédures

```
POUR JOUER
EC [PENSE A UNE FLEUR, JE
VAIS LA DEVINER]
PARCOURIR :FLEURS
FIN

POUR PARCOURIR :A
SI MOTP :A [PROPOSER :A STOP]
EC PREMIER :A
SI LISLISTE=[OUI] [PARCOURIR
   PREMIER SP :A] [PARCOURIR DE
   RNIER :A]
FIN

POUR PROPOSER :F
(EC [JE PROPOSE :] :F [EST-CE EXA
   CT ?])
SI LISLISTE=[OUI] [EC [JE SUIS
   GENIAL] STOP]
EC [A QUELLE FLEUR PENSAIS-TU
   ?]
DONNE “REP.OUI DERNIER LIS
   LISTE
(EC [QUELLE QUESTION AURAIS-
   TU POSEE POUR QUE LA RE
   PONSE SOIT OUI POUR] :REP.
   OUI [ET NON POUR] :F)
DONNE “NQ LISLISTE
DONNE “FLEURS COMPLETER :FL
   EURS :REP.OUI (LISTE :NQ :REP.
   OUI :F)
FIN

POUR COMPLETER :A :F :B
SI :A = :F [RETOURNE :B]
SI MOTP :A [RETOURNE :A]
RETOURNE (LISTE PREMIER :A
   COMPLETER PREMIER SP :A :F
   :B COMPLETER DERNIER :A :F
   :B)
FIN
```

ÉVOLUER DANS LES ARBRES BINAIRES

donner l'utilisateur. Pour les réponses par OUI ou NON, on remplacera SI LIS LISTE = [OUI] par SI REPONSE = [OUI]

```
POUR REPONSE
DONNE "REP LISLISTE
SI :REP = [OUI] [RETOURNE [OUI]]
SI :REP = [NON] [RETOURNE [NON]]
EC [DONNEZ-MOI UNE REPONSE PAR
   OUI OU PAR NON]
RETOURNE REPONSE
FIN
```

Les réactions des joueurs

Lorsque l'utilisateur doit donner un nom de fleur, il faut éventuellement éliminer l'article et transformer les mots restants en un seul. Par exemple UNE GUEULE DE LOUP deviendra GUEULE__DE__LOUP.

```
DONNE "REP.OUI DERNIER LIS
   LISTE deviendra DONNE "REP.
   OUI NOUV.FLEUR

POUR NOUV.FLEUR
DONNE "REP LISLISTE
VERIFIE :REP
REPETE 3 [SI MEMBREP PREMIER
   :REP :ARTICLES [DONNE "REP
   SP :REP VERIFIE :REP]
DONNE "REP TRANSFORME :REP
FIN

POUR VERIFIE :R
SI  :R = [] [EC [DONNEZ-MOI UN
   NOM DE FLEUR] NOUV.FLEUR]
FIN

POUR TRANSFORME :L
SI :L =  [] [RETOURNE "]
RETOURNE SD (MOT PREMIER :L
   "__ TRANSFORME SP :L)
FIN
```

On n'oubliera pas d'initialiser quelque part la liste "ARTICLES DONNE "ARTICLES [L' LE LA LES UN UNE DES C' C'EST EST A AU]. Ainsi la réponse "c'est un iris des prés" donnera le mot IRIS__DES__PRES. Bien sûr, si la réponse est "c'est à une rose que je pense", le programme extraira la réponse ROSE__QUE__JE__PENSE. C'est pratiquement inévitable, sauf si vous êtes présent lors de l'utilisation de votre programme pour guider le joueur. Vous n'éviterez pas non plus le mot LOCOMOTIVE comme nom de fleur. Il en ira de même pour la nouvelle question que vous ne pourrez pas empêcher d'être farfelue. Vous pourrez simplement vérifier par programme qu'elle n'est pas vide. DONNE "NQ LISLISTE deviendra DONNE "NQ NOUV.QUESTION :

```
POUR NOUV.QUESTION
DONNE "NQ LISLISTE
SI NON :NQ = [] [RETOURNE :NQ]
EC [DONNEZ-MOI UNE QUESTION]
RETOURNE NOUV.QUESTION
FIN
```

Une dernière amélioration :REJOUER.

```
POUR REJOUER
EC [VEUX-TU RECOMMENCER]
SI REPONSE = [OUI] [JOUER] [DE
   TRUIS "JEU SAUVE "JEU]
FIN
```

JEU est supposé être le fichier contenant le jeu, que vous avez ramené en début de partie et qui se réécrit modifié après chaque fin de jeu.

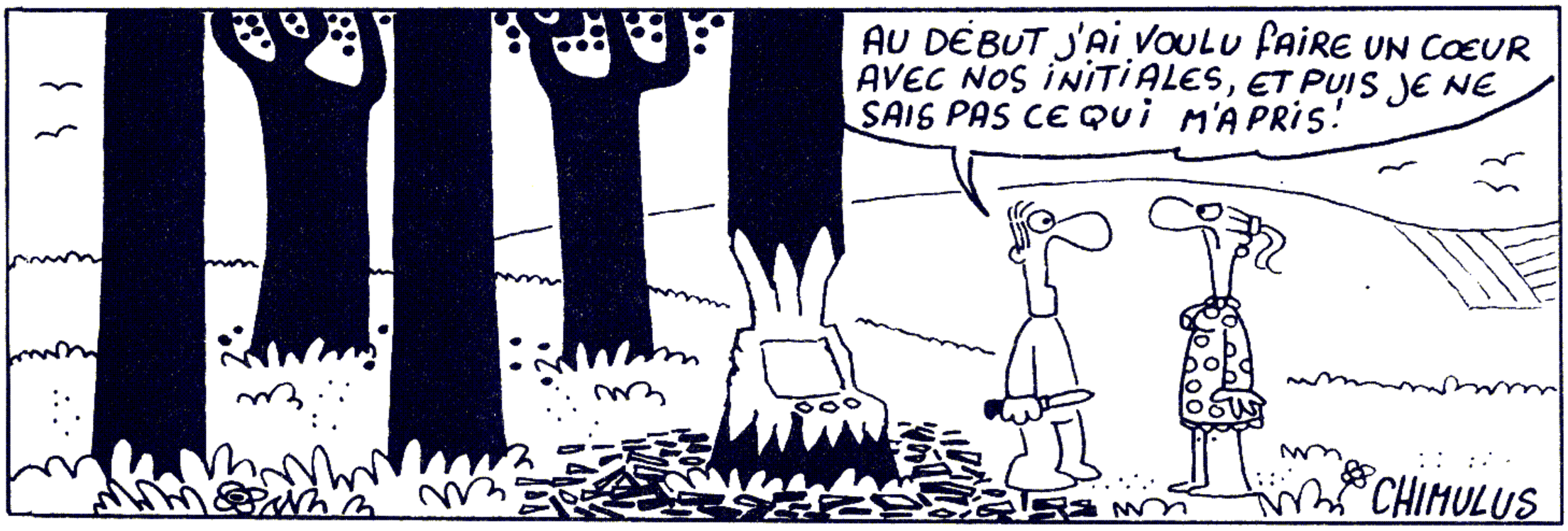
Les utilisateurs ont, vous le constaterez, envie de se mesurer avec la machine. La première fois, ils répondent tous correctement, mais dans le même temps, ils apprennent que l'enrichissement du programme dépend d'eux. La moitié change alors d'attitude. Un sur deux, en effet, aimerait alors répondre n'importe quoi. Et c'est d'ailleurs ce qu'il fait si vous lui laissez le loisir d'effectuer une deuxième tentative.

En règle générale, on observe que le joueur n'essaie jamais d'induire le programme en erreur dès la première tentative parce qu'il a envie de voir si sa question a bien été enregistrée. Mais pour un joueur sur deux, il faut intervenir dès la deuxième tentative.

Au troisième essai, c'est 99 % des joueurs qui ont envie de faire apprendre n'importe quoi à la machine. Ne faites donc jamais jouer trois fois la même personne. Le libre service est très déconseillé. A moins bien sûr de vouloir étudier l'évolution du délire chez le joueur.

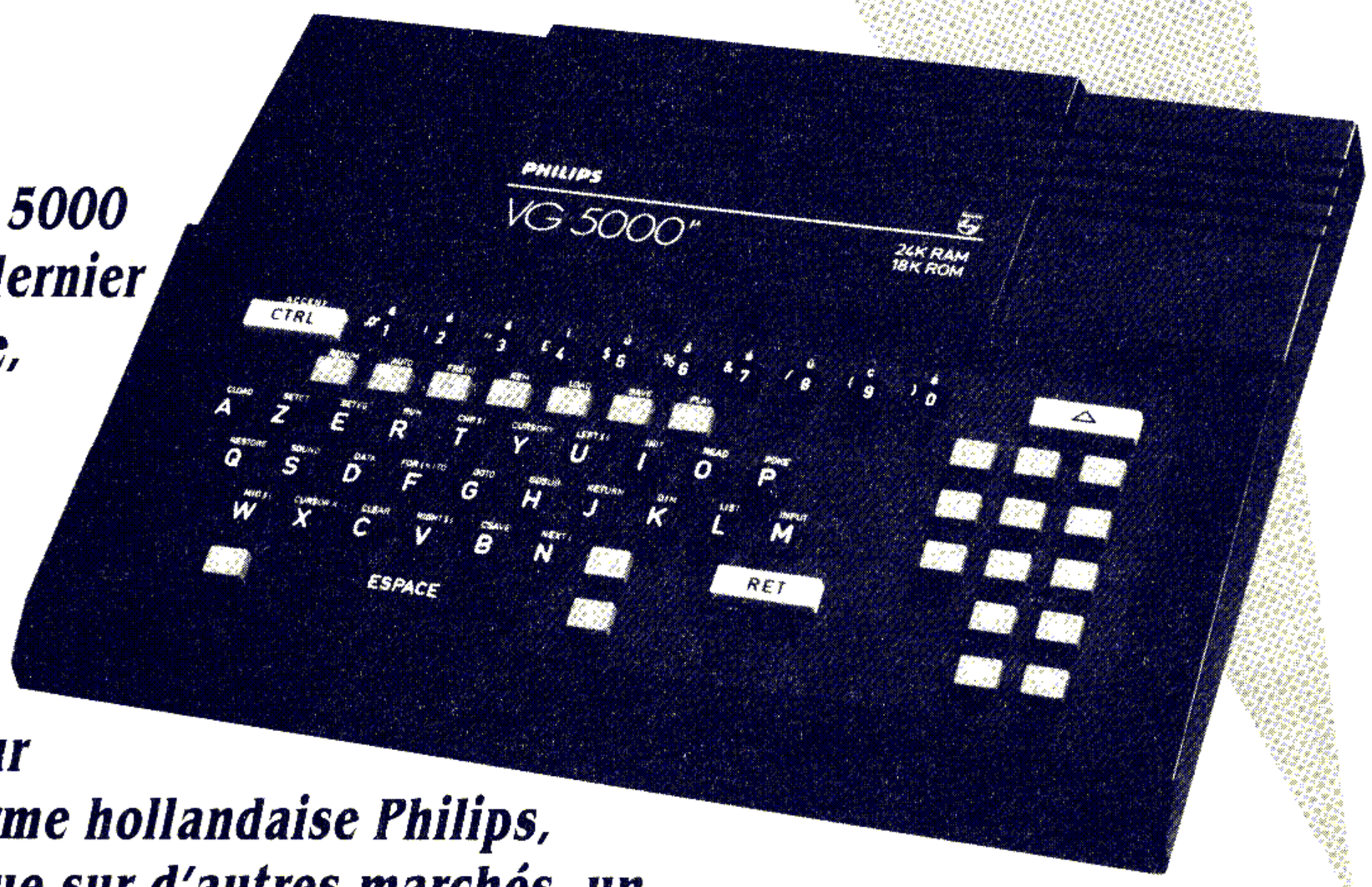
Après avoir abordé des procédures qui faisaient évoluer des données, nous étudierons une prochaine fois des procédures qui se modifient elles-mêmes en fonction de conditions extérieures et sur des structures de données plus complexes que les arbres binaires. N'oublions pas que Logo est un langage d'Intelligence Artificielle qui ne distingue pas les procédures des données.

Robert DAGUESSE



LIST A TESTÉ

LE BASIC DU VG 5000



L'ANNONCE du VG 5000 en septembre dernier fut, à plus d'un titre, un petit événement dans le monde de la micro-informatique. C'était à la fois le premier ordinateur domestique de la firme hollandaise Philips, ô combien dynamique sur d'autres marchés, un nouvel ordinateur français (mais oui !) car conçu et fabriqué en France, et enfin un bon rapport qualité-prix dans le bas de gamme.

Le VG 5000 est livré en version de base avec un gros boîtier d'alimentation à découpage, un câble permettant d'utiliser un magnétophone à cassettes et l'indispensable câble péritélévision qui autorise la connexion à tout téléviseur du commerce. Le clavier se veut de type « minitel », et ce n'est pas un compliment.

En effet, ce clavier ne permet pas la frappe rapide : les touches sont toutes petites, proches les unes des autres et ne restent pas horizontales lorsqu'on les enfonce. Mise à part l'absence de pavé numérique, la disposition est assez intelligente bien qu'elle s'écarte notablement du standard, et que l'on soit obligé (au début) de chercher tel ou tel caractère. Les quatre opérations sont ainsi accessibles dans le pavé d'édition situé à droite du clavier principal. La partie alphabétique est disposée selon le standard français AZERTY, et les minuscules accentuées sont présentes. Comme le veut une tendance actuelle sur les machines bon marché, la plupart des mots clés sont accessibles directement au clavier.

Le Basic est d'origine Microsoft, et on retrouve donc un noyau d'instructions présent sur d'autres matériels.

Un petit raffinement

L'éditeur est plein écran et son emploi particulièrement facile. Les touches associées à cet éditeur sont à répétition (comme le reste du clavier) et permettent l'insertion et la suppression d'une ligne, la suppression d'un caractère, et l'insertion d'un « blanc ». Il n'y a pas de véritable mode insertion : l'insertion d'un texte nécessitera celle du nombre de blancs correspondant. Les autres touches du pavé d'édition sont celles d'effacement de l'écran et de déplacement du curseur dans les quatre directions. Les précieuses commandes AUTO et RENUM sont présentes. Ajoutons que la longueur maximum d'une ligne est de 144 caractères. Le VG 5000 offre un petit raffinement : les messages d'erreurs, assez nombreux et bien conçus, s'affichent en français et en clair (entendez par là, sans abréviation).

Le Basic ne possède qu'un type de variables numériques. Les valeurs sont stockées sur quatre octets. La précision est donc assez mauvaise : six chiffres significatifs. Les noms de variables peuvent être de longueur quelconque, mais seuls les deux premiers caractères sont significatifs et les mots réservés sont bien sûr proscrits.

Le contrôle des structures s'effectue

grâce à une poignée d'instructions maintenant parfaitement classiques. Seul le strict nécessaire est présent. Ainsi IF... THEN ne peut être suivi de ELSE, et ne rêvez pas à quelque WHILE.... WEND et autres procédures : ce n'est pas sur cette machine de bas de gamme que vous les trouverez.

L'affichage s'effectue dans une matrice de 24 lignes de 40 caractères. Il est possible de positionner le curseur à la position désirée par CURSORX et CURSORY. Il n'existe pas de véritable mode haute définition, c'est-à-dire que chaque point de l'écran n'est pas accessible individuellement. C'est une déception, car la création de dessins devient très délicate. Il faut en effet définir les caractères graphiques nécessaires à la composition de la figure à dessiner, puis les afficher au moyen de PRINT aux positions désirées.

Cette définition concerne bien entendu la forme du caractère, mais aussi sa couleur (huit couleurs sont disponibles) et la couleur du fond. Lors d'un affichage, la couleur du fond du caractère se propage jusqu'au bout de la ligne courante.

Des images, des sons et des poignées

Trois instructions d'affichage, originales et très pratiques, sont là : DELIM fait ressortir un texte lors d'un PRINT ; DISPLAY offre la possibilité de régler la vitesse d'affichage ; et enfin, PAGE interdit tout défilement de l'écran, SCROLL rétablissant bien sûr la situation initiale.

Dans le domaine des sons, et même, osons le dire, de la musique, le VG 5000 ne se débrouille pas trop mal. Son Basic est doté à cette fin de deux instructions : SOUND suivi de trois paramètres émet un son réglable en fréquence et en durée. Le troisième paramètre modifie « le rapport cyclique de l'onde produite» (selon l'expression employée dans la documentation). La deuxième instruction est PLAY « *chaîne de caractères* » ; son fonctionnement est similaire au PLAY du Basic MSX (en moins puissant) et permet donc de jouer une mélodie composée d'une suite de notes (avec éventuellement dièses et bémols), selon le tempo désiré.

Des images, des sons. Il ne manquerait plus que quelques instructions pour gérer le fonctionnement des poignées, et à nous les petits jeux d'action ; ces instructions existent, et permettent de tester la position de deux poignées avec leur bouton. En fait, les quatre flèches de direction peuvent être utilisées comme une poignée de jeu.

Dans le domaine des chaînes de caractères, rien de nouveau sous le soleil, le conservatisme de Microsoft s'exprime ici pleinement, et à bon escient, dans la mesure où la petite panoplie de fonctions de chaînes alphanumériques suffit à tout faire simplement. On regrettera juste l'absence de INSTR, très pratique pour extraire une sous-chaîne d'une chaîne.

Les fonctions mathématiques sont peu nombreuses. Comme souvent en Basic, la seule fonction trigonométrique inverse est ATN. Heureusement, DEFFN permettra de combler cette lacune. Les fonctions logiques se présentent sous forme de mot (et non pas de signe) : AND, OR et NOT. Mais la fonction XOR (*ou exclusif*) fait défaut.

Il est possible de sauvegarder et de relire des programmes Basic, des tableaux ou des chaînes de caractères grâce au « duo » : CSAVE-CLOAD. Un deuxième duo, SAVE-LOAD vient compléter le premier en permettant la sauvegarde et la lecture de programmes sous forme ASCII. En fait, il en existe quatre autres du même type permettant la sauvegarde et la lecture de tableaux, de chaînes de caractères, de zones de la mémoire en code machine, de la zone écran (image vidéo). Quant à CSAVEL, il saute l'amorce de la bande magnétique. CLOADA remplit le rôle du classique MERGE en permettant la fusion de deux programmes, et enfin CLOAD? permet de vérifier la sauvegarde d'un programme.

Les habitués du chapitre « accès au langage-machine » sont là aussi : le couple PEEK et POKE ; CALL qui lance un programme en langage-machine à partir de l'adresse spécifiée par l'unique argument ; USR qui effectue la même opération, à ceci près que l'adresse doit être définie au préalable et qu'il est possible de passer un argument qui ira se loger dans un registre du micro-processeur.

Le Basic du VG 5000 est un des plus rudimentaires qu'ait produit Microsoft. Cela dit, le public visé est celui des débutants, et il est vrai que ce Basic est particulièrement facile à apprendre. Il faut d'ailleurs ajouter que le prix de la machine est intéressant. Vu sous cet angle, le tableau s'éclaircit : dans cette gamme de prix, la concurrence est quasiment inexistante. Mais, répétons-le, et contrairement à une machine MSX ou un Amstrad, le passage à la vitesse supérieure nécessitera, de la part de l'amateur l'achat d'une machine plus puissante.

Thierry LÉVY-ABÉGNOLI

Liste des mots clés du Basic du VG 5000

ABS
ACTION
AND
ASC
ATN
AUTO
CALL
CHR$
CLEAR
CLOAD
CONT
COS
CSAVE
CURSORX
CURSORY
DATA
DEFFN
DELIM
DIM
DISPLAY
EG
END
ET
EXP
FOR... STEP... NEXT
FRE
GOSUB
GOTO
GR
IF... GOTO
IF... THEN
INIT
INPUT
INT
KEY
LEFT$
LEN
LET
LIST
LLIST
LOAD
LOG
LPRINT
LPOS
MID$
NEW
ON... GOSUB
ON... GOTO
OR
PAGE
PEEK
PLAY
POKE
POS
PRINT
READ
REM
RENUM
RESTORE
RETURN
RIGHT$
RND
RUN
SAVE
SCREEN
SCROLL
SETEG
SETET
SGN
SIN
SOUND
SPC
SQR
STICKX
STICKY
STOP
STORE
STR$
TAB
TAN
TX
USR
VAL
& "xxxx"

Fiche technique du VG 5000

Constructeur : *Philips (SA Philips IC)*
Distributeur : *Philips*
Prix public : *1 590 FF (environ)*
Processeur : *Z80 A*
Mémoire morte : *18 Ko*
Mémoire vive : *24 Ko*
Mémoire vive disponible : *13758 octets (13,4 Ko)*
Langage : *Basic résident*
Affichage : *25 lignes de 40 caractères*
Résolution graphique : *400 x 200 points (non adressables individuellement)*
Précision : *7 chiffres pour les calculs, 6 chiffres stockés et affichés*
Nombre de mots clés du Basic : *85*

LIST - PAGE 58

LA SUITE A SUIVRE

TROUVER le « suivant » d'une suite de nombres donnés n'est pas simple, si l'on ne connaît pas la loi qui régit cette suite. Il faut donc trouver des formules qui permettent d'extrapoler. Elles existent, mais attention aux résultats.

Pour extrapoler une suite de données, on commence par considérer ces données comme les ordonnées d'une fonction correspondant à des abscisses équidistantes. Puis, on utilise une formule d'interpolation, par exemple celle de Lagrange. Elle remplace la fonction inconnue par le polynôme le plus simple, celui dont la courbe représentative passe par tous les points correspondants aux données.

De cette manière, il paraît naturel de chercher à résoudre les tests de suites numériques, comme les fameux tests qui servent prétendument à mesurer le *quotient intellectuel.*

Mais on constate très vite que les résultats sont franchement mauvais. Les relations de récurrence déterminant ces suites aboutissent rarement à une représentation algébrique simple en fonction du rang du nombre.

En analysant un peu plus à fond, on observe que ces relations restent généralement du premier degré (équations linéaires), et ne font appel qu'à un ou deux des termes précédents. Mais il y a souvent deux relations différentes, parfois trois, et les critères de simplicité retenus par les auteurs des tests semblent échapper à toute définition exhaustive ou systématique. Il n'est donc pas facile de trouver des solutions sur ordinateur, et vain d'espérer une infaillibilité dans ce domaine.

Le programme, quant à lui, explore successivement les possibilités suivantes :

$A_n = p\,A_{n-1} + q$
$A_n = p\,A_{n-2} + q$
$A_n = p\,A_{n-1} + q\,A_{n-2}$
$A_n = p\,A_{n-2} + q$, si n est pair et
$A_n = r\,A_{n-2} + s$, si n est impair
(relations polynômiales avec interpolation de Lagrange).

Le nombre des réponses est égal à dix moins le nombre des données. Le programme refuse les coefficients p qui ne seraient pas entiers ou inverses d'entiers. Il donnera des réponses exactes dans la plupart des cas. Les aberrations proviennent généralement des relations polynômiales essayées en désespoir de cause dans les cas les plus complexes. Par exemple, si l'on a utilisé trois lois de récurrence ou plus, ou encore une suite de nombres premiers. En pareil cas, le programme souligne le caractère douteux des réponses par un petit bip.

Écrit en Basic standard sur un PX-8 (Epson), on pourra le transposer sans trop de mal sur d'autres matériels. Si vous n'avez pas de ON ERROR GOTO, un message d'erreur sans conséquence suivra la dernière réponse. En revanche, avec un Basic qui explore toutes les boucles au moins une fois avant d'en tester les bornes (comme le Canon X-07), il faudra agir à coup de IF...THEN.

Un dernir truc pour parfaire votre entraînement : si aucun bip n'a dénoncé une interpolation de Lagrange, essayez donc de deviner la (ou les) relations(s) utilisée(s) et contrôlez les valeurs de K, p, q ; mais aussi, si K = 4, celles de r et s.

Suite à suivre
Programme en Basic
Auteur Pierre Barnouin
Copyright *LIST* et l'auteur

Explications du programme

Lignes	Commentaires
1 à 3	Initialisation, entrée des données
4 à 6 et 20	a (n) = pa (n − 1) + q
7 à 9 et 21	a (n) = pa (n − 2) + q
10 à 12 et 22	a (n) = pa (n − 1) + q a (n − 2)
13 à 18, 23 et 24	a (n) = pa (n − 2) + q (n pair) a (n) = ra (n − 2) + s (n impair)
19	Tri des solutions
25 à 27	Relation polynômiale

```
1 PRINT"Entrez les données puis un RETURN supplémentaire"
2 DEFINT A:ON ERROR GOTO 28
3 INPUT A(N):IF A(N) THEN N=N+1:GOTO 3
4 P=(A(2)-A(1))/(A(1)-A(0)):Q=A(1)-P*A(0)
5 FOR I=3 TO N-1:IF A(I)<>P*A(I-1)+Q THEN I=90
6 NEXT:K=1:IF I=N THEN 19
7 P=(A(3)-A(1))/(A(2)-A(0)):Q=A(2)-P*A(0)
8 FOR I=4 TO N-1:IF A(I)<>P*A(I-2)+Q THEN I=90
9 NEXT:K=2:IF I=N THEN 19
10 P=(A(3)*A(0)-A(2)*A(1))/(A(2)*A(0)-A(1)*A(1)):Q=(A(3)-P*A(2))/A(1)
11 FOR I=4 TO N-1:IF A(I)<>P*A(I-1)+Q*A(I-2) THEN I=90
12 NEXT:K=3:IF I=N THEN 19
13 P=(A(4)-A(2))/(A(2)-A(0)):Q=A(2)-P*A(0)
14 R=P:IF N>4 THEN R=(A(5)-A(3))/(A(3)-A(1))
15 S=A(3)-R*A(1)
16 FOR I=5 TO N-1 STEP 2:IF A(I)<>R*A(I-2)+S THEN I=90
17 NEXT:FOR J=6 TO N-1 STEP 2:IF A(J)<>P*A(J-2)+Q THEN J=90
18 NEXT:K=4:IF I+J>90 THEN 25
19 IF P=INT(P) OR 1/P=INT(1/P) THEN ON K GOTO 20,21,22,23 ELSE 25
20 A(N)=P*A(N-1)+Q:PRINT A(N);:N=N+1:GOTO 20
21 A(N)=P*A(N-2)+Q:PRINT A(N);:N=N+1:GOTO 21
22 A(N)=P*A(N-1)+Q*A(N-2):PRINT A(N);:N=N+1:GOTO 22
23 IF N MOD 2 THEN A(N)=R*A(N-2)+S:PRINT A(N);:N=N+1
24 A(N)=P*A(N-2)+Q:PRINT A(N);:N=N+1:GOTO 23
25 BEEP 9:FOR I=0 TO N-1:Z=A(I):FOR J=0 TO N-1
26 IF I-J THEN Z=Z*(N-J)/(I-J)
27 NEXT:Y=Y+Z:NEXT:A(N)=Y:PRINT A(N);:Y=0:N=N+1:GOTO 25
28 END
```

Pierre BARNOUIN

UN PETIT PIANO DANS LA POCHE

NOUS savons que, grâce à une courte routine en langage-machine, le PC-1251 peut générer des sons sans utiliser son bip (1). En faisant varier fréquence et durée et en attribuant les notes ainsi obtenues aux différentes touches du clavier, l'ordinateur devient un minuscule instrument de musique.

Mises à part CL, ENTER et les sept premières touches de la ligne supérieure du clavier, toutes les autres seront réquisitionnées pour les besoins de la musique. Nous allons en effet pouvoir jouer sur deux octaves en augmentant ou diminuant éventuellement les notes d'un demi-ton (dièse et bémol), en choisissant leur durée (de la double croche à la blanche en passant par la noire pointée) et leur timbre.

Pour vous en convaincre, tapez soigneusement le programme et lancez-le.

Si le menu défile un peu lentement à votre goût, sachez que, par la suite, vous pourrez appeler directement, à partir de DEF, le sous-programme qui vous intéresse.

Vous avez le choix entre six options :

• C, comme Clavier, affecte à chaque touche une note. Pour connaître leur position et utiliser votre PC-1251 comme un (petit) piano, reportez-vous à la figure de la page ci-contre. La première gamme commence à la lettre A et se termine en K, la seconde part de K et s'arrête sur la barre de fraction de la ligne supérieure.

• M, comme Mémorisation, vous permet d'entrer en mémoire un air que la machine vous restituera ensuite deux fois, sur un rythme plus rapide.

• A, comme Automatique, rend la main à la machine qui compose elle-même une mélodie à partir des éléments que vous lui donnez : le nombre de notes, la possibilité d'inclure ou non des dièses et celle de choisir les sonorités sur une ou deux octaves.

• J, comme « je Joue », fera exécuter à la machine les compositions mises en mémoire.

• K, comme K7, déclenche la sauvegarde des morceaux sur cassette.

• L, comme Load, permet de les récupérer pour les écouter de nouveau.

Pour sortir d'un sous-programme et en appeler un autre, tapez sur SPC.

A partir de la routine de génération de sons proposée par Marc Leygnac dans le n° 5 de *LIST*, j'ai déterminé les valeurs correspondant aux fréquences des notes de la gamme et j'ai trouvé un moyen d'obtenir des durées égales quelle que soit la hauteur des sons.

Il ne restait plus, dès lors, qu'à affecter les différentes notes de la gamme aux touches du clavier ; et pour mémoriser une mélodie, de créer les tableaux enregistrant les paramètres des différentes sonorités.

Faisons, si vous le voulez, un essai. Prenons un air simple, connu de tous, « Au clair de la lune » : DO DO DO RE MI RE DO MI RE RE DO.

Tapez DEF M. La machine vous demande de choisir une base-temps et un timbre. A la première sollicitation, répondez par un nombre compris entre 0 et 255 (par exemple 200), à la seconde par un chiffre pris en 0 et 9. Indiquez enfin le nombre de notes de la mélodie (11 dans notre cas), puis introduisez celle-ci au clavier. L'ordinateur mémorise alors l'air et vous le joue deux fois de suite.

Pas de difficulté particulière si l'on se contente de ne jouer qu'en noires (valeur : un temps), mais il est tout à fait possible de faire intervenir des notes de valeur différente. Il faut alors indiquer, après avoir tapé la note, s'il s'agit d'une noire pointée (un temps et demi), d'une blanche (deux temps), d'une croche (un demi-temps) ou d'une double croche (un quart de temps). Frappez respectivement sur les touches 0, 9, 8 et 7.

Vous en savez maintenant suffisamment pour reproduire quelques refrains connus ou vous lancer dans l'improvisation. Un conseil toutefois, si vous possédez l'interface-cassette (CE-125), il vaut mieux, pendant l'emploi du programme, se servir de l'alimentation secteur pour faire fonctionner le PC-1251. Il semble bien en effet que, lors d'une utilisation prolongée du haut-parleur, les piles se déchargent plus rapidement qu'en temps normal.

(1) Une petite musique de poche de Marc Leygnac, LIST *5, page 64.*

Daniel BRIANT

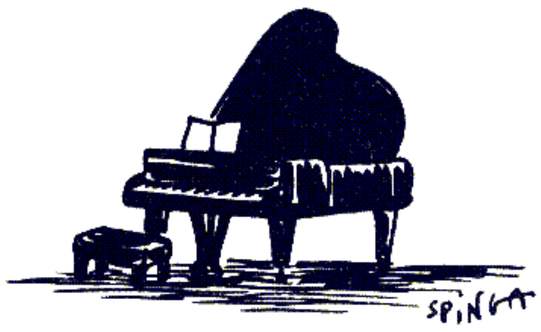
Piano de poche

Programme pour PC-1251

Auteur Daniel Briant

Copyright *LIST* et l'auteur

```
 1:REM MENU
 2:DIM X$(0)*65: WAIT 9
   0: PRINT " * MENU ..
   . DEF ?": WAIT 20
 3:X$(0)="          C/CLA
   VIER, M/MEMORIS., A/
   IMPROVIS., K/ENREG.,
    L/LECT.K7"
 4:FOR I=1 TO 45: PRINT
   MID$ (X$(0),I,24):
   NEXT I
 5:POKE 63548,1: WAIT :
   PRINT "CHOISIS : C M
    A K  OU L"

10:REM FREQUENCES
20:RESTORE
30:DATA 20,5,0,1,15,15,
   2,55,41,30,47,35,25,
   .15,.3,3,0,0,0,61,0,
   0,0
40:DATA 255,136,190,190
   ,207,176,149,125,84,
   105,95,77,95,113,69,
   69,232,162,221,136
50:DATA 113,162,207,221
   ,136,255
60:CLEAR : DIM F(48):
   FOR I=0 TO 48: READ
   F(I): NEXT I
70:REM PARAMETRES
80:INPUT "BASE-TEMPS ";
   B,"TIMBRE ";R
90:Z=256: IF B>Z-1 OR R
   >9 THEN 80
100:Y= INT (Z*B/(Z-B)):D
   = INT (B*(Y+Z))
110:DIM U(R): FOR I=0 TO
   R: READ U(R): NEXT I
   :R=U(R)
120:DATA 16,34,50,80,98,
   114,146,162,178,242
130:REM OCTETS
140:POKE &C000,&03,&00,&
   02,R,&FF,&B1,&4E,&00
   ,&02,&00,&FF,&B1,&C3
   ,&29,&0C,&37
150:RETURN

160:REM IMPULSIONS
170:T(I)=F(J):N(I)= INT
   (D/(Y+T(I)))
180:REM  S O N
190:POKE &C001,N(I):
   POKE &C007,T(I):
   CALL &C000: RETURN

200:REM TABLEAUX NOTES
210:INPUT "NOMBRE DE NOT
   ES ";M
220:DIM N(M),T(M):
   RETURN

240:REM TOUCHE
250:A$= INKEY$ : IF A$="
   " THEN 250
260:J= ASC (A$)-42:
   RETURN
300:"C": REM CLAVIER
310:GOSUB 20:I=0: DIM N(
   I),T(I): BEEP 2
320:GOSUB 250: IF J=-10
   THEN END
330:GOSUB 170: GOTO 320

350:"M": REM MEMOIRE
360:GOSUB 20: GOSUB 210
370:BEEP 2: PAUSE "JOUEZ
    ! J ENREGISTRE !"
380:FOR I=0 TO M
390:GOSUB 250: IF J=-10
   BEEP 2: GOTO 450
400:IF J=6 OR J=13 OR J=
   14 OR J=15 LET N(I-1
   )= INT (F(J)*B):
   GOTO 390
410:GOSUB 170: NEXT I:
   BEEP 2: GOTO 450

450:"J": REM EXECUTION
460:GOSUB 140: PAUSE "
   JE  JOUE  !"
470:FOR K=1 TO 3
480:FOR I=0 TO M-1: IF N
   (I)=0 THEN 500
490:GOSUB 190: NEXT I
500:FOR L=0 TO 50: NEXT
   L: NEXT K: BEEP 2:
   END

540:REM IMPROVISATION
550:"A": CLEAR : RESTORE
   120: GOSUB 80: GOSUB
   210
560:INPUT "OCTAVE 1/2 ";
   O,"AVEC #  O/N ";G$
570:E=8: IF G$="O" LET E
   =13
580:DIM F(E): RESTORE 58
   0+(10*O): FOR I=0 TO
   E: READ F(I): NEXT I
590:DATA 225,221,190,176
   ,149,125,105,95,232,
   207,162,136,113
600:DATA 95,77,61,55,41,
   30,20,15,84,69,47,35
   ,25
610:PAUSE "  J IMPROVISE
    !": RANDOM
620:FOR I=0 TO M-1:J=
   RND E: GOSUB 170
630:IF RND 0<.5 LET N(I)
   = INT (.25*B)
640:NEXT I: GOTO 450

650:"K": REM ENREGISTRER
660:INPUT "K7 PRETE ";A$
670:PRINT #M: PRINT #R,N
   (*),T(*)
680:BEEP 2: END

700:"L": REM LECTURE K7
710:CLEAR : INPUT "K7 PR
   ETE ";A$
720:INPUT #M: DIM N(M),T
   (M)
730:INPUT #R,N(*),T(*)
740:GOTO 450
```



L'organisation du clavier

Q DO # 232	W RE # 207	E MI b 207	R FA # 162	T SOL # 136	Y LA b 136	U SI b 113	I DO # 84	O RE # 69	P MI b 69	4 FA #	5 SOL #	6 SI b	/ DO 15
A DO 255	S RE 221	D MI 190	F FA 176	G SOL 149	H LA 125	J SI 105	K DO 95	L RE 77	= MI	1 FA	2 SOL	3 LA	* SI 20
Z DO 255	X RE 221	C MI 190	V FA # 162	B SOL # 136	N LA # 113	M DO 95	SPC FIN	ENTER		0 𝅗𝅥.	. DO 15	+ RE 5	− MI b 1

7 : 𝅘𝅥𝅯 — 8 : 𝅘𝅥 — 9 : 𝅗𝅥

Deux octaves

Plus 1 ton et demi

Octave à 6 notes séparées par 1 ton

Les valeurs à ne pas dépasser
— Base-temps : de 0 à 255 (nombre entier)
— Timbre : de 0 à 9
— Touches de durée : 7 et 8 (base-temps < 256)
0 (base-temps < 128)
9 (base-temps < 86)

UTILISEZ VOTRE MAGNÉTOPHONE AVEC ADRESSE

TÉLÉCOMMANDER le lecteur de cassettes, savoir si une de ses touches est enfoncée ou non, arrêter le moteur, les ordinateurs Commodore font ça très bien. Mais pourquoi le système d'exploitation serait-il seul à le faire ? Voici quelques adresses et comment les utiliser.

Quiconque a déjà utilisé un de ces (... mais en existe-t-il plusieurs ?) ordinateurs qui ne savent pas arrêter un magnétophone sait qu'une machine honnête *doit* avoir quelque part dans son système d'exploitation de quoi contrôler le fonctionnement du lecteur de cassettes. Chez Commodore, deux adresses sont d'une importance méééééétaphysiquement strrratégique, dirait Dali. L'une peut télécommander le moteur, l'autre sait si une touche du magnétophone a été enfoncée. Une touche, n'importe laquelle (sauf bien sûr STOP), car l'unité centrale est bien incapable de savoir si la touche enfoncée est le bobinage rapide, la lecture, ou l'enregistrement et la lecture ensemble : il faut bien faire un peu confiance à l'utilisateur.

Quatre instructions de base seront donc à considérer.

• Attendre qu'une touche du magnétophone soit enfoncée.
Sur CBM : WAIT 59408, 16, 16
Sur Vic : WAIT 37151, 64, 64
Sur C.64 : WAIT, 1, 16, 16

• Attendre l'arrêt du magnétophone (toutes les touches relevées).
Sur CBM : WAIT 59408, 16
Sur Vic : WAIT 37151, 64
Sur C.64 : WAIT 1, 16

• Arrêter le moteur du magnétophone.
Sur CBM : POKE 59411, 60
Sur Vic : POKE 37148, 244
Sur C.64 : POKE 192, 7 : POKE 1, PEEK (1) AND 223 OR 32

• Relancer le moteur du magnétophone.
Sur CBM : POKE 59411, 61
Sur Vic : POKE 37148, 254
Sur C.64 : POKE 192, 0 : POKE 1, PEEK (1) AND 223

Rappelons que WAIT bloque l'exécution d'un programme jusqu'à ce qu'un octet particulier soit à l'état voulu (et sans tester la touche STOP, ce qui en fait une instruction redoutable ; voir encadré page suivante).

L'anglais vous donne de l'urticaire ?

Le bon usage de ces adresses peut singulièrement simplifier la vie de ceux qui utilisent des fichiers séquentiels sur cassette, et même des autres.

Le premier exemple sera la francisation des messages relatifs au magnétophone. Ce n'est pas la peine de faire de jolis programmes dans la langue de Molière (« Ma chère cassette ... »), si c'est pour voir apparaître sur l'écran, dès qu'on veut faire des entrées/sorties sur icelle (la cassette), des messages rédigés dans la langue de Shakespeare : "PRESS RECORD AND PLAY ON TAPE" (HamPet Acte 7, scène 255). C'est même franchement rebutant pour les gens qui ne demanderaient qu'à se servir d'un ordinateur, mais à qui la seule vue d'une phrase anglaise colle de l'urticaire aussi sec.

Or, dès qu'on fait un OPEN, 1, 1, 0, "ZOZO" (ouverture en lecture d'un

fichier séquentiel sur cassette), apparaît un laid "PRESS PLAY ON TAPE", qu'on le veuille ou non, à moins que, sentant venir le vent, l'utilisateur ne se soit précipité, index frémissant, pour enfoncer la touche lecture avant qu'on ne le lui demande.

L'astuce va donc consister à demander à l'utilisateur *avant* l'OPEN d'appuyer sur la touche ; et pendant qu'on y est, à le lui demander *en français* :

```
5000 PRINT "APPUYEZ SUR LA TOUCHE PLAY DU MAGNETOPHONE"
5010 WAIT 1, 16, 16
5020 PRINT "D'ACCORD"
5030 OPEN ...
```

et la vieille dame du Quai Conti (Non ! pas Yourcenar, les quarante !) d'applaudir bien fort.

Etalonner le programme

Une autre utilisation de cette astuce consiste, par exemple dans un programme qui gère un seul fichier sur cassette, à mettre le fichier sur la même cassette que le programme, à distance respectueuse. Les choses se passent ainsi : le programme principal, dans la partie consacrée à l'écriture du fichier, demande de rembobiner la cassette jusqu'à son début, puis d'appuyer sur STOP. Il demande ensuite d'appuyer sur la touche « marche avant rapide ». Dès que cette touche est enfoncée, il lance un chrono et, lorsque ce chrono atteint le temps nécessaire pour que le programme principal soit dépassé sur la cassette, il coupe le moteur et demande d'appuyer sur la touche STOP du magnétophone avant de procéder à l'écriture.

```
49999 REM *** POSITIONNEMENT DE LA CASSETTE ***
50000 PRINT "METTEZ LA CASSETTE A SON DEBUT"
50010 PRINT "PUIS FRAPPEZ RETURN."
50020 GET R$ : IF R$ <  > CHR$(13) THEN 50020
50030 PRINT "ENFONCEZ LA TOUCHE 'AVANCE RAPIDE'"
50040 PRINT "DU MAGNETOPHONE."
50050 WAIT 1, 16, 16
50060 PRINT "D'ACCORD"
50070 TI$ = "000000"
50080 IF TI <  XX THEN 50080
50090 POKE 192, 7 : POKE 1, PEEK (1) AND 223 OR 32
50100 PRINT "ENFONCEZ LA TOUCHE 'STOP'"
50110 PRINT "DU MAGNETOPHONE.""
50120 WAIT 1, 16
50130 PRINT "D'ACCORD"
50140 REM *** ECRITURE DU FICHIER ***
50150 PRINT "ENFONCEZ LES TOUCHES ENREGISTREMENT"
50160 PRINT "ET LECTURE DU MAGNETOPHONE."
50170 WAIT 1, 16, 16
50180 PRINT "D'ACCORD"
50190 OPEN 1, 1, 1, "ZOZO"
etc.
```

WAIT : attendez qu'on s'explique !

WAIT n'est pas une instruction présente sur toutes les machines, et si elle est présente dans le Basic de Commodore, elle n'est pas souvent utilisée. Il est vrai que c'est une instruction booléenne en diaaaable, quoâ !

Elle fonctionne selon la syntaxe WAIT A,B,C où A représente une adresse et B et C deux nombres compris entre 0 et 255, C étant facultatif et valant 0 par défaut. En fait, l'ordinateur comprend : IF (PEEK(A) XOR C) AND B est différent de 0 THEN je continue, sinon, je boucle et j'attends obstinément qu'il cesse d'être nul comme ça.

XOR, c'est l'opération de OU exclusif ; AND, c'est le ET logique.

A	B	A XOR B	A AND B
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

Sur le Commodore 64, le bit 4 de l'octet 1 est consacré à la détection d'une touche enfoncée sur le magnétophone, et le bit 5 au contrôle du moteur.
Si on veut attendre que toutes les touches du magnétophone soient relevées, il faut que le bit 4 de l'octet contenu dans l'adresse 1 soit à 1.

Essayez d'enfoncer la touche PLAY de votre lecteur de cassette et de faire PRINT PEEK(1) : vous obtenez 7 en temps normal. Si PEEK(1) vaut 7, comme la représentation de 7 sur un octet est 00000111 et celle de 16 est 00010000, alors 7 AND 16 devient sur un octet :

7	00000111
16	00010000
7 AND 16	00000000

Le résultat de 7 AND 16 est nul : WAIT 1, 16 fait boucler le programme.
Appuyez sur la touche STOP et refaites PRINT PEEK(1) : vous obtenez 55.
Si PEEK(1) vaut 55, on a sur un octet :

55	00110111
16	00010000
55 AND 16	00010000

Le résultat de 55 AND 16 est non-nul, le programme peut continuer.
Si, en l'état actuel des choses, on tombe sur un WAIT 1, 16, 16 on a :

55	00110111
16	00010000
55 XOR 16	00100111
(55 XOR 16) AND 16	00000000

Le résultat est nul et le programme boucle. Mais si on enfonce une touche, les bits 4 et 5 passent à zéro (l'octet étant formé de huit bits numérotés : 76543210) et l'on a :

7	00000111
16	00010000
7 XOR 16	00010111
(7 XOR 16) AND 16	00010000

Le résultat est non-nul et le programme peut continuer.

L'important dans l'histoire est donc d'étalonner le programme ainsi modifié pour fixer la valeur représentée par XX à la ligne 50080. Cette valeur est comparée à celle de TI. Rappelons que chez Commodore, deux variables réservées sont consacrées à l'horloge interne. L'une, TI, représente l'heure exprimée en *jiffies* ou 1/60e de seconde. On peut la lire, mais pas lui affecter de valeur : essayez de faire en mode direct TI=0 ou TI=12, vous aurez du SYNTAX ERROR autant que vous voudrez. Pour remettre TI à zéro, il faut passer par la

seconde variable, TI$, une chaîne de caractères comme son nom l'indique, qui représente l'heure au format HHMMSS, c'est-à-dire "101254" pour 10 heures 12 minutes 54 secondes. C'est ce qui est fait à la ligne 50070. Or donc, pour étalonner le programme principal, on peut se rentrer vite fait un petit programme du genre :

```
100 PRINT "APPUYER SUR LA
    TOUCHE 'AVANCE RAPIDE'"
110 WAIT 1, 16, 16
120 PRINT "D'ACCORD"
130 TI$ = "000000"
140 WAIT 1, 16
150 PRINT TI
```

Pour l'utiliser, il faut d'abord enregistrer le programme principal au début d'une cassette, puis rembobiner complètement la bande (amorce comprise), mettre le compteur du magnétophone à zéro et faire un LOAD ou un VERIFY. On note le numéro du compteur, et on rembobine. On peut alors lancer l'étalonnage.

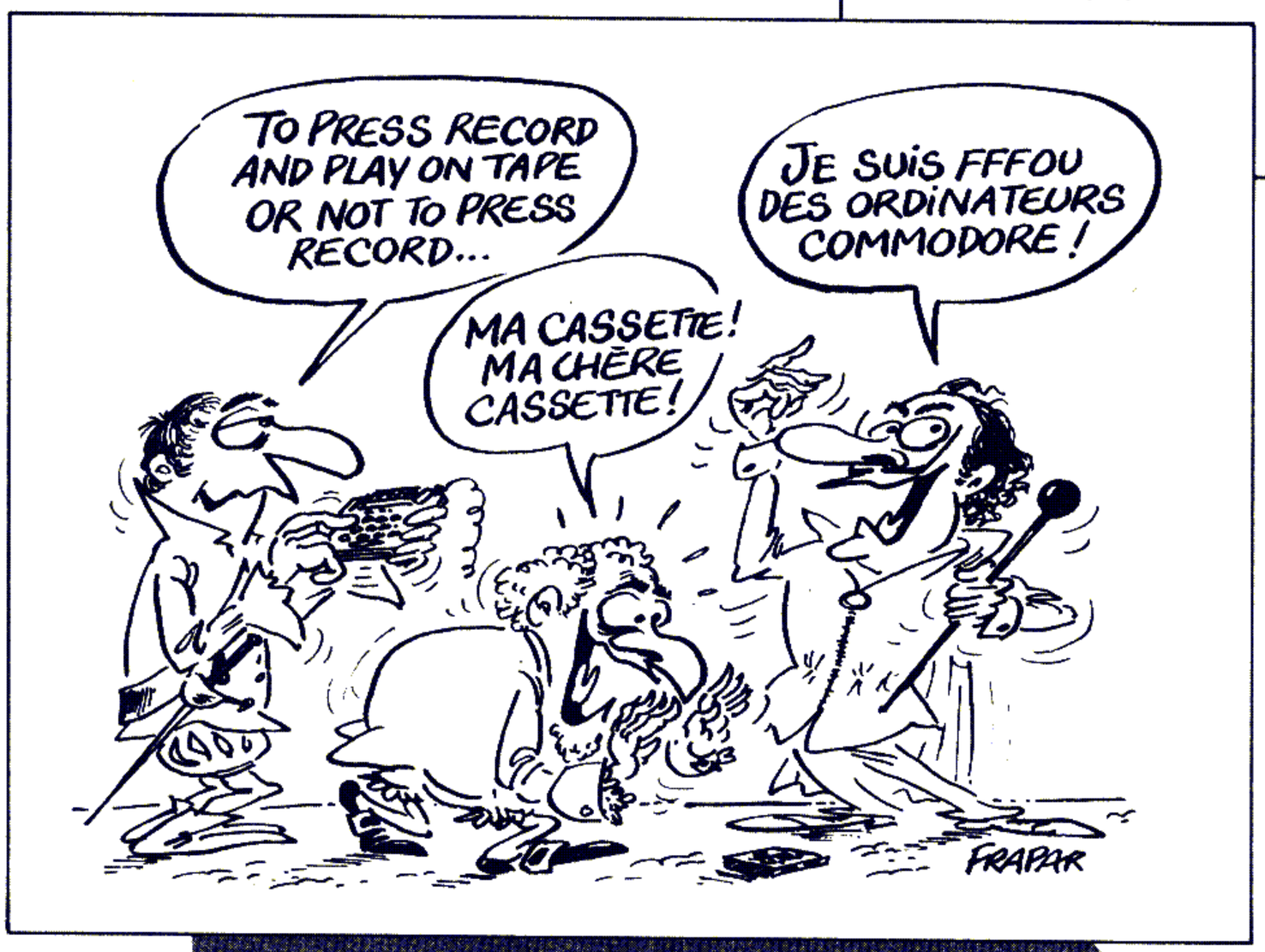
Attention aux noyaux des cassettes

Le programme attend qu'une touche soit enfoncée. Dès que c'est fait, il remet à zéro la variable réservée TI$ et, en même temps, la variable réservée TI. Lorsque le compteur atteint le numéro que vous avez noté, appuyez sur la touche STOP du magnétophone. Le temps (en jiffies) nécessaire au bobinage s'affiche alors à l'écran, et vous le notez précieusement : c'est ce nombre qui figurera à la place de XX ligne 50080.

Mais attention ! Quechvouzesplique un machin important : les cassettes ont beau être standard, seules les dimensions extérieures sont fixes. A l'intérieur, chacun fait ce qu'il veut, et notamment dans le domaine du diamètre des noyaux. Quatre mètres de bande, ça ne met pas le même temps à se bobiner sur un tout petit noyau de 12 mm de diamètre et sur un noyau confort de 25. Si vous copiez votre programme sur des cassettes différentes, il faudra procéder chaque fois à un étalonnage. Qu'on se le tienne pour dit.

La même remarque vaut pour l'utilisation suivante. Car ce qu'on vient de faire pour un fichier, rien n'empêche de le faire pour plusieurs programmes. On peut mettre en début de cassette un programme de ce style :

```
10 DATA 100, 200, 300, 400, 500, 600
20 DIM T(6)
30 FOR I=1 TO 6:READ T(I):NEXT I
100 PRINT CHR$(147); TAB(18); "JEUX": PRINT
110 PRINT "1. FLIGHT SIM"
120 PRINT "2. BATTLE OF NEW CALEDONIA"
130 PRINT "3. SORCERER OF LEVALLOIS"
140 PRINT "4. MORPIO ATTACK"
150 PRINT "5. RETURN OF THE SON OF DONKEY KONG JR."
160 PRINT "6. PANICK AT CHATEAUVALLON"
180 PRINT
190 PRINT "VOTRE CHOIX ?" ;
200 GET R$:R=VAL(R$): IF R <1 AND R > 6 THEN 200
210 XX=T(R): PRINT R
220 PRINT "ENFONCEZ LA TOUCHE 'AVANCE RAPIDE'"
230 PRINT "DU MAGNETOPHONE."
240 WAIT 1, 16, 16
250 PRINT "D'ACCORD"
260 TI$="000000"
270 IF TI < XX THEN 270
280 POKE 192, 7:POKE, 1, PEEK(1) AND 223 OR 32
290 PRINT "ENFONCEZ LA TOUCHE 'STOP'"
300 PRINT "DU MAGNETOPHONE."
310 WAIT 1, 16
320 PRINT "D'ACCORD. MAINTENANT, ENFONCEZ"
330 PRINT "LA TOUCHE PLAY."
340 WAIT 1, 16, 16
350 LOAD "", 1, 1
```



Les DATA du début correspondent à des valeurs étalonnées comme précédemment et qui sont fonction du temps mis pour atteindre le programme voulu depuis la fin du programme de catalogue.

Si vous faites partie de ces gens méticuleux qui sont capables de conserver avec chaque cassette une fiche où son contenu est répertorié avec les numéros au compteur, ce programme ne vous servira strictement à *rien* ! Si vous avez les moyens de ne mettre qu'un programme par cassette, et si vous avez suffisamment de place chez vous, ou suffisamment peu de cassettes, il sera également dérisoire et inutile. Bonne chance à tous les autres !

François J. BAYARD

PICOLOGO

UN PROGRAMME A PETITS PAS

VOUS possédez depuis peu un Amstrad, et vous n'êtes pas encore un programmeur chevronné ? Parfait, nous allons nous entendre... Voici donc une sorte de suite à Amstrad-graphe, sous la forme d'un programme qui aura le double avantage de vous faire connaître (très superficiellement !) le langage Logo et de vous apporter de nouvelles connaissances sur le Basic de votre machine préférée.

Nous avons décidé de tirer parti de la rapidité du Basic et des remarquables possibilités graphiques du CPC 464 en écrivant en Basic un mini-interpréteur Logo. Si vous l'ignorez, Logo est en partie un langage artistique. Une « tortue », matérialisée sous la forme d'une flèche, est susceptible de laisser sur l'écran les traces de ses déplacements, en fonction des ordres simples que vous lui donnerez : avance, recule, tourne...

Répétons-le : notre version de Logo est évidemment très simplifiée, mais notre but est aussi de vous fournir une base de travail sur laquelle vous pourrez greffer vos propres idées : ajouter par exemple la possibilité de définir des procédures, de traiter des variables, etc. Nous serons heureux de connaître le résultat de vos travaux, et d'en faire profiter nos lecteurs.

Décortiquons le programme

Avant d'aborder le mode d'emploi du programme, commençons tout de suite par en analyser le contenu. Nous vous conseillons de le taper au clavier tout en suivant nos explications.

Ligne 100 : lors de la mise au point d'un programme, il est très utile de prévoir une programmation de touche qui permettra, sur une simple pression, de « reprendre la main » sans complication. Cette ligne pourra être éliminée quand le programme sera parfaitement au point. Celle que nous avons redéfinie est la touche ENTER du pavé numérique.

Lignes 160-190 : préparation de l'écran et des couleurs. Nous utiliserons le mode standard (40 colonnes) dont la résolution est excellente pour notre application. La fenêtre 1 contiendra le texte, tandis que la fenêtre 0 sera le terrain des manœuvres graphiques de la tortue.

Ligne 200 : nul n'est à l'abri des erreurs ; il faut donc tout prévoir !

Lignes 220-240 : nous utiliserons deux tableaux. Le premier contiendra la suite des ordres donnés à la tortue, et le second mémorisera les 15 ordres auxquels elle sait obéir (les primitives). Ce nombre pourra être augmenté à volonté, en modifiant simplement la valeur de la variable NP.

Lignes 260-270 : voici les 15 primitives, que nous avons abrégées en deux ou trois lettres pour des raisons d'économie (AV = AVance). Vous pouvez les écrire en entier si vous le voulez : vous en découvrirez l'inconvénient à l'usage. La signification des ordres est expliquée à la suite.

Ligne 300 : pour éviter des complica-

PICOLOGO UN PROGRAMME A PETITS PAS

tions inutiles, nous sélectionnons le mode de calcul en DEGrés, car nous utiliserons les SINus et COSinus. En effet, le mode normal est en RADians, et nous n'y sommes guère habitués. Enfin, les coordonnées de départ de la tortue sont fixées au centre de la fenêtre graphique.

Ligne 310 : divers paramètres sont fixés, et la tortue est sommée d'apparaître à l'emplacement prévu, grâce au sous-programme qui débute en ligne 1520.

Lignes 330-610 : voici la partie principale du programme, la plus utilisée. C'est ici que se font l'entrée des commandes destinées à la tortue, et leur traitement.

Ligne 350 : la machine attend vos ordres, et elle les stocke temporairement dans une chaîne nommée L$. Le LINE INPUT est indispensable, car il accepte tout ce qui est tapé au clavier, ce qui ne serait pas le cas avec un simple INPUT. Si vous appuyez sur ENTER sans donner de réponse, un bouclage se produit sur cette même ligne.

Ligne 360 : cette ligne est destinée au confort de l'utilisateur. Quand il frappe "↑", la ligne précédemment tapée réapparaît, et les ordres qu'elle contenait sont réexécutés par la tortue.

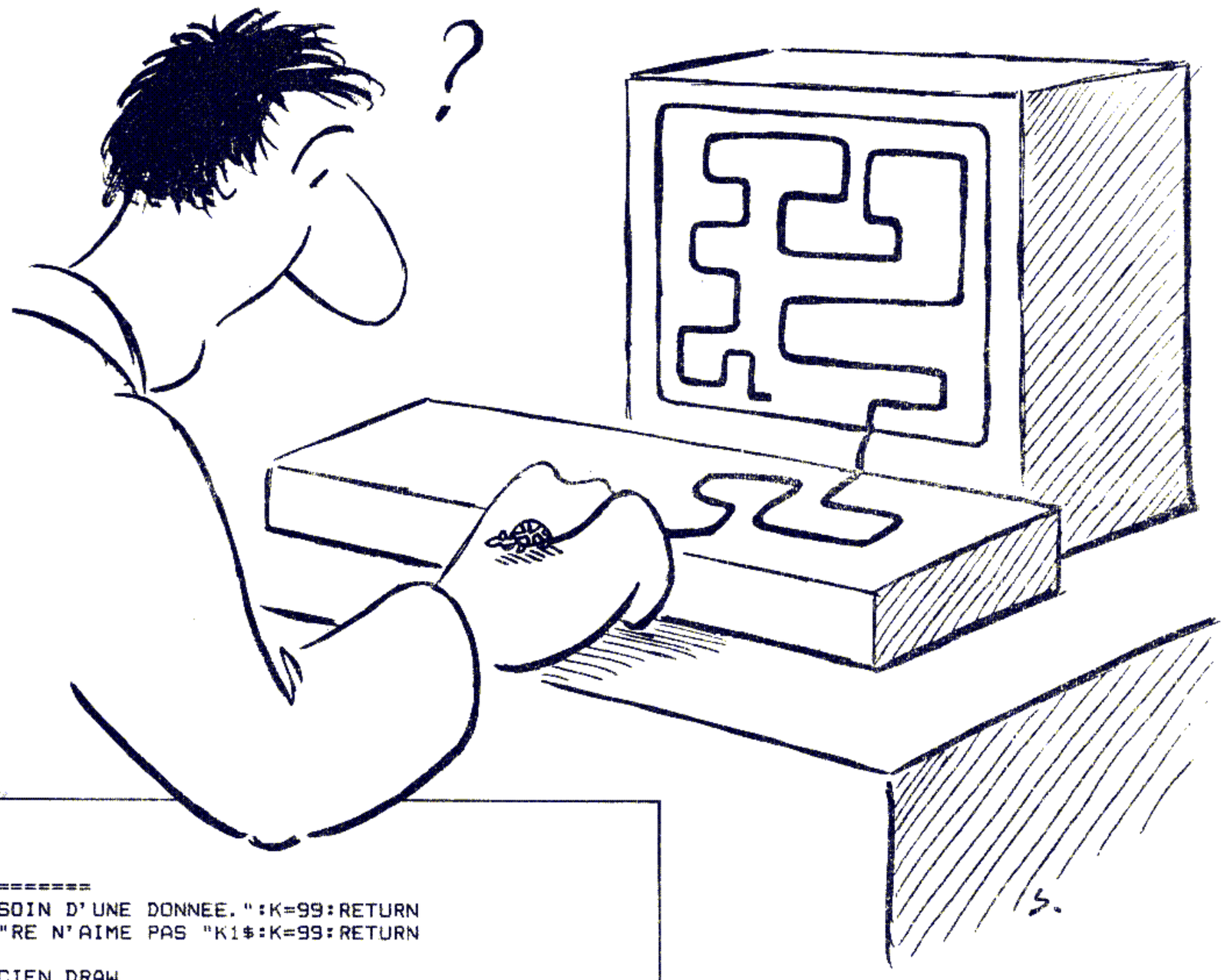
Lignes 380-490 : si l'utilisateur n'a pas tapé "↑", les ordres contenus dans L$ sont découpés en éléments qui sont ensuite rangés dans le tableau C$. Le découpage se fait en fonction des espaces détectés dans L$. Ainsi, AV 10 DR 90 occupe quatre cases dans le tableau, tandis que AV 10 DR90 (qui est une erreur) en occuperait trois. N'omettez donc pas de **séparer toutes vos instructions** et leur(s) paramètre(s) éventuel(s) par un espace au moins ! S'il y a plusieurs espaces, ce n'est pas un problème, grâce à la routine de traitement qui les élimine à la ligne 440. A la sortie, NC représente le nombre de cases occupées par la suite de commandes.

Lignes 510-610 : ces commandes sont maintenant exécutées l'une après l'autre par l'appel des sous-programmes correspondants. Si une commande n'est pas reconnue, c'est le sous-programme qui débute en 1830 qui est appelé, et un sympathique message d'erreur coupe court à l'exécution. La variable K représente le nombre de cases à sauter pour passer à la commande suivante située dans C$. En cas d'erreur, K prend pour valeur 99, ce qui provoque la fin de la

```
100 KEY 139,"MODE 1:PAPER 0:INK 1,21:PEN 1:LIST"+CHR$(13)
110 '*************************************
120 '*      PICOLOGO V.1 * CPC 464       *
130 '*      (C) JP LALEVEE ET LIST       *
140 '*************************************
150 :
160 MODE 1:INK 0,1:INK 1,24:INK 2,20:INK 3,7:BORDER 0
170 WINDOW 1,40,21,25:PEN 0:PAPER 3:CLS
180 WINDOW#1,1,40,1,20:PAPER#1,0:CLS#1
190 :
200 ON ERROR GOTO 1930
210 :
220 DIM C$(99):REM COMMANDES
230 NP=15:REM NOMBRE DE PRIMITIVES
240 DIM PR$(NP):REM PRIMITIVES
250 :
260 FOR I=1 TO NP:READ PR$(I):NEXT
270 DATA DR,GA,AV,RE,FPOS,FCC,LC,BC,CT,MT,VE,CTR,ORI,REP,ADIEU
280 :
290 C=1:AC=C:REM COULEUR D'ORIGINE
300 DEG:X=320:Y=240:X1=X:Y1=Y
310 MT=1:BC=1:A=90:GOSUB 1520:REM DESSINE TORTUE
320 :
330 '====== ENTREE DES COMMANDES ======
340 WHILE R$()"#"
350 L$="":WHILE L$="":PRINT"?";:LINE INPUT L$:WEND
360 BIS=0:IF L$="↑"THEN BIS=1:PRINT"*";:FOR I=0 TO N:PRINT C$(I);" ";:NEXT:PRINT
370 :
380 '----------- DECOUPAGE -----------
390 WHILE BIS=0
400 R$=UPPER$(L$):L1$=L$:WHILE LEFT$(R$,1)=" ":R$=MID$(R$,2):WEND
410 :
420 NC=0:D=INSTR(1,R$," "):WHILE D
430 C$(NC)=LEFT$(R$,D-1):R$=MID$(R$,D+1):NC=NC+1
440 WHILE LEFT$(R$,1)=" ":R$=MID$(R$,2):WEND
450 D=INSTR(1,R$," ")
460 WEND
470 IF R$)" " THEN C$(NC)=R$:ELSE NC=NC-1
480 REM NC=NOMBRE DE COMMANDES
490 BIS=1:WEND
500 :
510 '====== TRAITEMENT COMMANDES ======
520 N=0:K=0
530 WHILE N(=NC
540 K0$=C$(N):NM=NP+1
550 FOR I=1 TO NP:IF K0$=PR$(I) THEN NM=I
560 NEXT
570 K1$=C$(N+1):K2$=C$(N+2)
580 ON NM GOSUB 980,980,630,750,1120,880,940,940,1080,1080,1760,1690,1800,1250,1
860,1830
590 N=N+K:WEND
600 WEND
610 PRINT"AU REVOIR !":END
620 :
630 '============ AVANCE ============
640 IF K1$="" THEN PRINT"AV A BESOIN D'UNE DONNEE.":K=99:RETURN
650 L=VAL(K1$):IF L=0 THEN PRINT"AV N'AIME PAS ";K1$:K=99:RETURN
660 GOSUB 1600:REM EFFACE TORTUE
670 IF BC THEN GOSUB 1650:REM DERNIERE LIGNE
680 ZX=L*COS(A)+X:ZY=L*SIN(A)+Y
690 IF ZX(10 OR ZX)630 OR ZY(90 OR ZY)390 THEN PRINT"LA TORTUE VA SORTIR !":GOSU
B 1520:K=99:RETURN
700 X1=X:Y1=Y:X=ZX:Y=ZY
710 MOVE X1,Y1:IF BC THEN DRAW X,Y,C:ELSE X1=X:Y1=Y
720 GOSUB 1520:REM DESSINE TORTUE
730 K=2:RETURN
740 :
```

LIST - PAGE 66

Picologo
Programme pour
Amstrad CPC 464
Auteur Jean-Pierre Lalevée
Copyright *LIST* et l'auteur



```
750 '============ RECULE ==============
760 IF K1$="" THEN PRINT"RE A BESOIN D'UNE DONNEE.":K=99:RETURN
770 L=VAL(K1$):IF L=0 THEN PRINT"RE N'AIME PAS "K1$:K=99:RETURN
780 GOSUB 1600:REM EFFACE TORTUE
790 IF BC THEN GOSUB 1650:REM ANCIEN DRAW
800 ZX=L*COS(180+A)+X:ZY=L*SIN(180+A)+Y
810 IF ZX<10 OR ZX>630 OR ZY<90 OR ZY>390 THEN PRINT"LA TORTUE VA SORTIR !":GOSU
B 1520:K=99:RETURN
820 IF BC THEN GOSUB 1650:REM ANCIEN DRAW
830 X1=X:Y1=Y:X=ZX:Y=ZY
840 MOVE X1,Y1:IF BC THEN DRAW X,Y,C:ELSE X1=X:Y1=Y
850 GOSUB 1520:REM DESSINE TORTUE
860 K=2:RETURN
870 :
880 '============== FCC ===============
890 IF K1$="" THEN PRINT"FCC A BESOIN D'UNE DONNEE.":K=99:RETURN
900 IF K1$<"0" OR K1$>"3" THEN PRINT"FCC N'AIME PAS "K1$:K=99:RETURN
910 AC=C:C=VAL(K1$):GOSUB 1520
920 K=2:RETURN
930 :
940 '============= LC / BC ============
950 BC=ABS(K0$="BC")
960 K=1:RETURN
970 :
980 '============= DR / GA ============
990 IF K1$="" THEN PRINT K0$" A BESOIN D'UNE DONNEE.":K=99:RETURN
1000 IF VAL(K1$)=0 THEN PRINT K0$" N'AIME PAS "K1$:K=99:RETURN
1010 ANG=VAL(K1$):IF K0$="DR"THEN ANG=-ANG
1020 A=A+ANG:IF A>360 THEN A=A-360
1030 GOSUB 1600:REM EFFACE TORTUE
1040 GOSUB 1650:REM ANCIEN DRAW
1050 GOSUB 1520:REM DESSINE TORTUE
1060 K=2:RETURN
1070 :
1080 '============ CT / MT ============
1090 IF K0$="MT" THEN MT=1:GOSUB 1520:ELSE MT=0:GOSUB 1600
1100 K=1:RETURN
1110 :
1120 '=========== FPOS (- -) ==========
1130 IF LEFT$(K1$,1)<>"(" THEN PRINT"FPOS VEUT DES ().":K=99:RETURN
1140 IF K2$="" THEN PRINT"FPOS VEUT 2 DONNEES.":K=99:RETURN
1150 XP=VAL(MID$(K1$,2)):IF ABS(XP)>319 THEN PRINT"FPOS N'AIME PAS":XP:K=99:RETU
RN
1160 IF RIGHT$(K2$,1)<>")" THEN PRINT"FPOS VEUT DES ().":K=99:RETURN
1170 YP=VAL(LEFT$(K2$,LEN(K2$)-1))
1180 IF ABS(YP)>157 THEN PRINT"FPOS N'AIME PAS "YP:K=99:RETURN
1190 GOSUB 1600:REM EFFACE TORTUE
1200 GOSUB 1650:REM ANCIEN DRAW
1210 X1=X:Y1=Y:MOVE X,Y:X=320+XP:Y=240+YP:IF BC THEN DRAW X,Y,C
1220 GOSUB 1520:REM DESSINE TORTUE
1230 K=3:RETURN
1240 :
1250 '============== REP ==============
1260 IF K1$="" THEN PRINT"REPETE A BESOIN ENCORE DE DONNEES.":K=99:RETURN
1270 REP=VAL(K1$):IF REP<1 THEN PRINT"REPETE N'AIME PAS "K1$:K=99:RETURN
1280 :
1290 REM ::::: RECHERCHE DU { :::::::::
1300 IF LEFT$(K2$,1)<>"{" THEN PRINT"IL MANQUE UN {":K=99:RETURN
1310 C$(N+2)=MID$(K2$,2)
1320 :
1330 REM ::::: RECHERCHE DU } :::::::::
1340 I=N+2:WHILE I<=NC AND RIGHT$(C$(I),1)<>"}":I=I+1:WEND
1350 IF I>NC THEN PRINT"IL MANQUE UN }":K=99:RETURN
1360 LR=I:C$(I)=LEFT$(C$(I),LEN(C$(I))-1)
1370 :
```

▶

boucle WHILE en ligne 590, et le retour à la ligne 390, pour une nouvelle acquisition d'instructions.

Lignes 630-1960 : ici commencent toutes les routines nécessaires, qui sont de deux types, celles qui correspondent à un ordre donné à la tortue (AV, RE, FCC...) et d'autres, plus élémentaires, qui sont elles-mêmes appelées par les précédentes (dessin du trait, représentation de la tortue...).

La liste des primitives

Chaque sous-programme contient ses propres traitements d'erreurs : contrôle de la syntaxe, contrôle des paramètres accompagnant la commande, qui sont contenus dans K1$ et K2$. La routine REP (lignes 1250-1500) est de loin la plus complexe. A ce sujet, signalons que les [] ont été remplacés sur notre liste par des accolades, à cause d'une imprimante inadaptée. A vous de restituer les crochets carrés.

Pour terminer, voici la liste en bon français des primitives : DRoite, GAuche, AVance, REcule, FixePOSition, FixeCouleurCrayon, Lève Crayon, Baisse Crayon, Cache Tortue, Montre Tortue, Vide Écran, CenTRe, ORIgine, REPète, ADIEU. Quant au mode d'emploi, il tient en quelques lignes :

• LC, BC, CT, MT, VE, CTR, ORI et

PICOLOGO UN PROGRAMME A PETITS PAS

Suite du programme

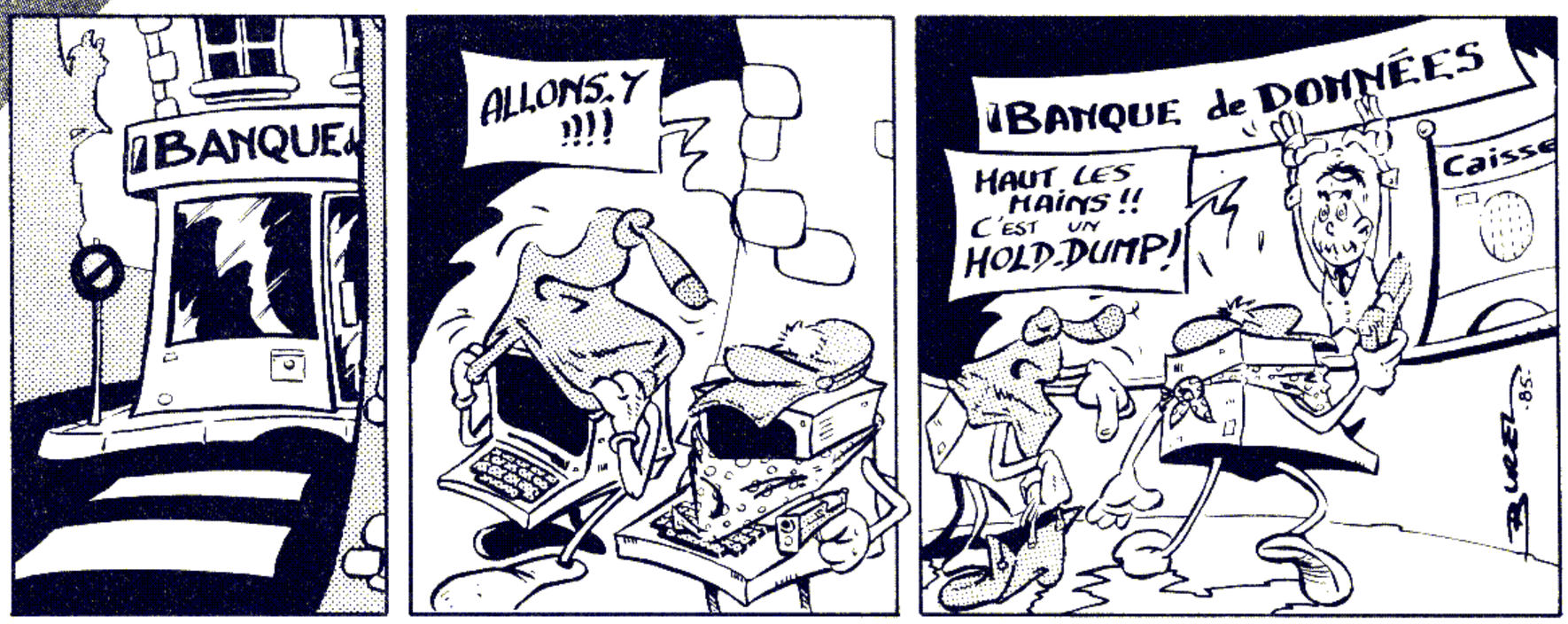
```
1380 REM ::::::: TRAITEMENT ::::::::::
1390 FOR J=1 TO REP
1400 I=N+2:WHILE I(=LR
1410 K@$=C$(I):NM=NP+1
1420 FOR H=1 TO NP:IF K@$=PR$(H) THEN NM=H
1430 NEXT
1440 IF NM=14 THEN PRINT"REP NE SAIT PAS SE REPETER !":K=99:I=LR:J=REP:GOTO 1470
1450 K1$=C$(I+1):K2$=C$(I+2)
1460 ON NM GOSUB 980,980,630,750,1120,880,940,940,1080,1080,1760,1690,1800,1250,
1860,1830
1470 I=I+K:WEND
1480 NEXT J
1490 C$(N+2)="{"+C$(N+2):C$(LR)=C$(LR)+"}":REM RAJOUTE LES {}
1500 K=LR+1:RETURN
1510 :
1520 '======= DESSINE LA TORTUE =======
1530 IF MT=@ THEN PLOT X,Y,C:RETURN
1540 TX1=10*COS(A)+X:TY1=10*SIN(A)+Y
1550 TX2=10*COS(A+90)+X:TY2=10*SIN(A+90)+Y
1560 TX3=10*COS(A-90)+X:TY3=10*SIN(A-90)+Y
1570 MOVE TX2,TY2:DRAW TX1,TY1,C:DRAW TX3,TY3,C:PLOT X,Y,C
1580 RETURN
1590 :
1600 '======== EFFACE LA TORTUE =======
1610 IF CT THEN PLOT X,Y,@:RETURN
1620 MOVE TX2,TY2:DRAW TX1,TY1,@:DRAW TX3,TY3,@:PLOT X,Y,@
1630 RETURN
1640 :
1650 '==== REPASSE LIGNE PRECEDENTE ===
1660 IF BC THEN MOVE X1,Y1:DRAW X,Y,AC:AC=C
1670 RETURN
1680 :
1690 '============ CENTRE ============
1700 GOSUB 1600:REM EFFACE TORTUE
1710 GOSUB 1650:REM ANCIEN DRAW
1720 X=320:Y=240:X1=X:Y1=Y:A=90
1730 PLOT X,Y,C:IF BC THEN GOSUB 1520:REM DESSINE TORTUE
1740 K=1:RETURN
1750 :
1760 '========= VIDE ECRAN ===========
1770 CLS#1:GOSUB 1520:X1=X:Y1=Y
1780 K=1:RETURN
1790 :
1800 '============ ORIGINE ==========
1810 BC=1:CLS#1:GOSUB 1720:K=1:RETURN
1820 :
1830 '======== COMMANDE INCONNUE ======
1840 PRINT"JE NE SAIS PAS FAIRE ";K@$:K=99:RETURN
1850 :
1860 '=== ========= ADIEU ? ==========
1870 PRINT"VOUS VOULEZ ME QUITTER (O/N) ?"
1880 R$="":WHILE R$("N" OR R$)"O"
1890 R$=UPPER$(INKEY$):WEND
1900 IF R$="O" THEN R$="#":ELSE PRINT"ON CONTINUE !"
1910 K=99:RETURN
1920 :
1930 '============= BUG ==============
1940 PRINT"AIE !... LA TORTUE N'AIME PAS LES BUGS."
1950 PRINT"L'INSECTE EST DANS LA LIGNE";ERL
1960 END
```

ADIEU s'utilisent sans paramètre.

- AV, RE exigent une valeur en nombre de points-écran (les pas de la tortue).
- DR, GA demandent une valeur angulaire de rotation en degrés.
- FCC nécessite le numéro de la couleur choisie (PEN du Basic, de 0 à 3).
- FPOS exige deux paramètres à placer entre parenthèses, qui sont des coordonnées absolues par rapport au centre de l'écran graphique. Exemple : FPOS (– 50 90).
- REP est plus complexe. Pour dessiner un cercle, la syntaxe serait : REP 360 [AV 1 DR 1]. Ce qui est long à dessiner, mais on peut faire le même cercle plus rapidement : à vous de chercher !
- ORI efface tout l'écran et remet la tortue en position de départ, tandis que VE efface, mais laisse la tortue à l'endroit où elle se trouve.

Achevons ce petit tour avec une remarque ayant trait à l'art de la programmation. Certains puristes considèrent que la valeur d'un programmeur se mesure à l'inverse du nombre de GOTO que contiennent ses programmes. Ils n'ont pas toujours tort puisque certains programmes deviennent impossibles à modifier à cause de GOTO inconsidérés. Aussi, puisque le Basic Amstrad est un peu structuré (pas beaucoup, hélas !), nous nous sommes amusé à limiter les GOTO de notre programme. Mais il n'empêche, allez, c'est vrai que parfois un bon GOTO est plus simple qu'un WHILE torturé. Essayons tout de même de ne pas en abuser !

Jean-Pierre LALEVÉE



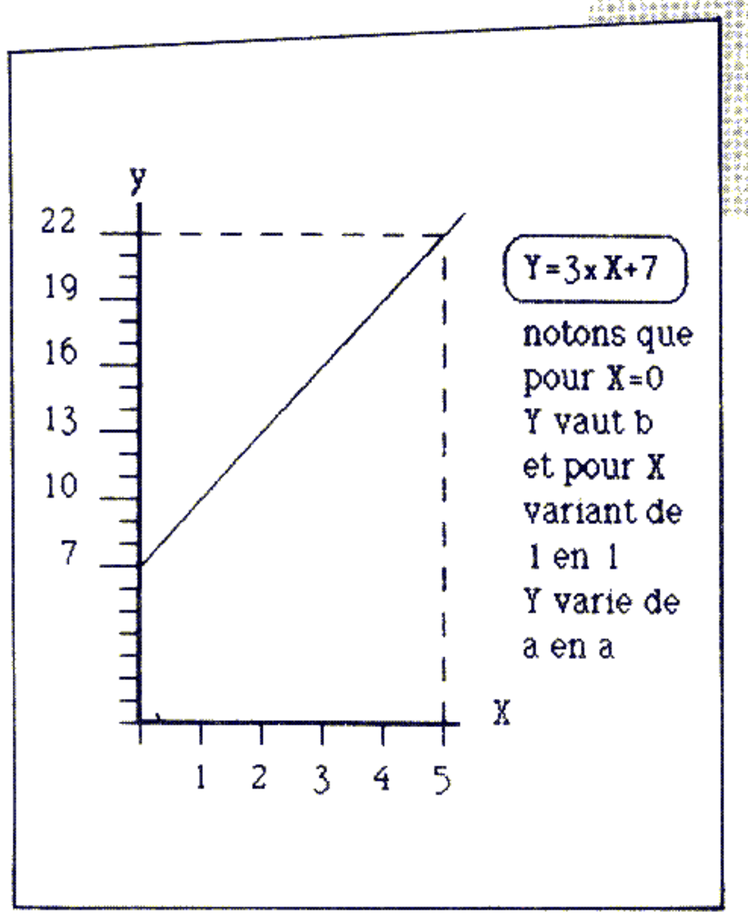
MISEZ P'TIT OPTIMISEZ

LA LIGNE DROITE ET LA BONNE PENTE

Si jongler avec la pile opérationnelle de votre HP-41C, traquer la milliseconde perdue et rogner le moindre octet est votre pain quotidien... Ou si, à l'inverse, vous échappe parfois un peu de la subtile recherche des programmes en Notation Polonaise Inverse... Alors voici qui doit vous intéresser. En matière de programmation, est-on jamais certain d'avoir fait aussi bien que possible ? Dans cette rubrique, les défis – vos défis – se succèdent : des programmes toujours plus courts, plus rapides... Et les records vivent !

Le plus court chemin entre deux points est encore, c'est bien connu, la droite. Dans un plan (O, X, Y), un point se définit par ses deux coordonnées X et Y. Connaissant les coordonnées de deux points du plan, calculer l'équation de la seule droite passant par ces deux points. Tel était le défi publié dans *LIST* n° 6.

Une droite se définit quant à elle par deux coefficients a et b de telle manière que tous les points alignés qui la composent vérifient l'équation $Y = aX + b$. Le graphe ci-contre représente une telle droite dont l'équation est : $Y = 3X + 7$. Ainsi, quelle que soit la valeur choisie sur l'axe des X, on calcule l'autre coordonnée Y du point par $3X + 7$. Les débutants le vérifieront aisément et



reconnaîtront (c'est l'occasion) que le coefficient a de l'équation est la pente de la droite tandis que b est la valeur à l'origine O (à zéro).

Le programme de référence occupait 12 octets et opérait en 28/100e de seconde pour trouver les coefficients a et b de la droite passant par les points (13,17) et (3,11). Beaucoup ont fait aussi bien, seuls deux lecteurs ont fait mieux : Eric Levenez et Didier Renevot. En 11 octets (sans compter ni le LBL de tête ni le END final) ils obtiennent la solution en... 2 centièmes de seconde de

moins (26/100e). Les plus belles économies sont celles des ultimes bouts de chandelles...

La routine tient en 10 pas de programme : introduire pour les deux points leurs coordonnées Y et X (par analogie avec les noms des registres de la pile) : Y1 ENTER X1 ENTER Y2 ENTER X2 XEQ "ED". Vingt-six centièmes de seconde après, les coefficients a et b de l'équation de la droite sont, respectivement, dans les registres Y et X de la pile opérationnelle.

L'algorithme de calcul employé est évidemment très simple puisque le programme tient en si peu de lignes. Suivons le cheminement. On a, à la fois, $Y1 = aX1 + b$ et $Y2 = aX2 + b$, soit deux équations pour deux inconnues. De celles-ci, on tire : $b = Y1 - aX1 = Y2 - aX2$ et, donc, $Y1 - Y2 = a(X1 - X2)$.

Ainsi avons-nous trouvé l'expression de a en fonction des coordonnées des deux points : $a = (Y1 - Y2)/(X1 - X2)$. On reconnaît évidemment l'expression de la pente de la droite (rapport des accroissements respectifs de Y et X).

Pour trouver l'expression de b, il suffit de remplacer dans une des deux équations de départ le a par sa formule : $Y2 = aX2 + b$ devient $Y2 = X2((Y1 - Y2)/(X1 - X2)) + b$ et, donc $b = Y2 - X2((Y1 - Y2)/(X1 - X2))$.

Droite
Programme pour HP-41C
Auteurs E. Levenez et D. Renevot
Copyright *LIST* et les auteurs

```
01 LBL "ED"
02 ST - Z
03 RDN
04 ST - Z
05 RDN
06 /
07 STO T
08 *
09 -
10 END
```

QUI DIT MIEUX ?

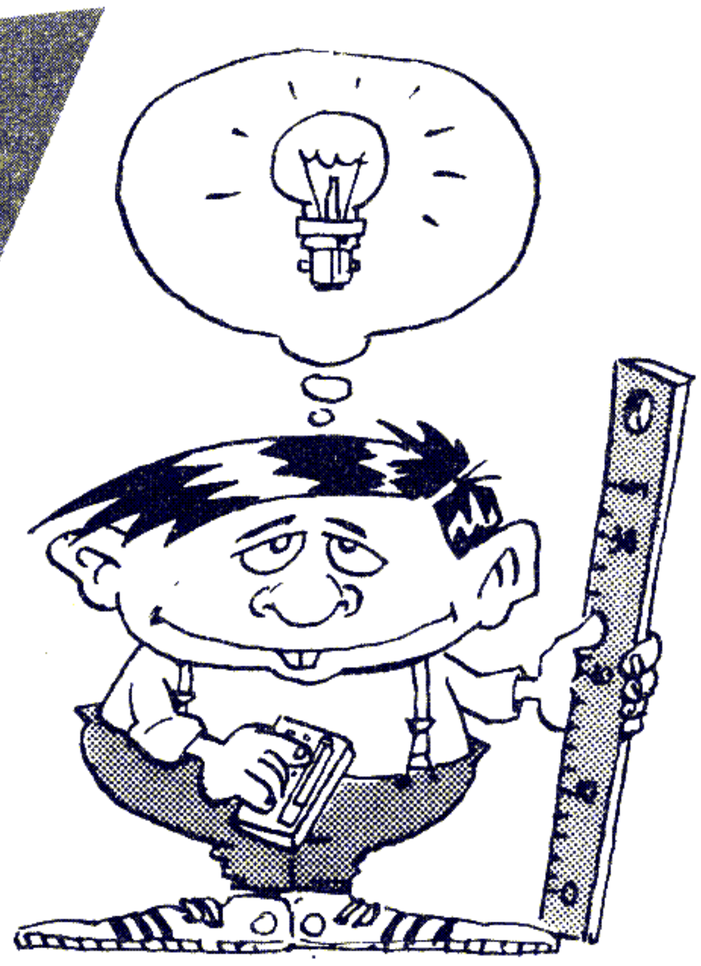
L'ÉQUATION du second degré $Y = aX^2 + bX + c$, on connaît. Sa résolution fut d'ailleurs le tout premier défi de *Misez p'tit* (*l'Op* n° 15). Résoudre une équation, c'est en trouver les racines (les zéros, c'est-à-dire les valeurs de X telles que $Y = 0$). Mais sauriez-vous trouver la valeur de X qui maximise ou minimise le résultat Y ?

Il est fortement recommandé de prêter attention au calcul de la fonction dérivée (c'est l'occasion ou jamais de réviser).

Mon programme occupe 28 octets pour 19 pas de programme (sans compter ni le LBL de tête ni le END final). On introduit en pile, dans l'ordre, les coefficients de l'équation a, b et c. Le résultat se trouve en X en fin de calcul (et Y, l'extremum, se trouve en Y). Pour l'équation $Y = 3X^2 - 4X + 5$ le résultat 0,666.. (soit 2/3) est trouvé en 63 centièmes de seconde.

Christophe FAUTRES

NDLR : si vous trouvez cet extremum, pouvez-vous déterminer s'il s'agit d'un maximum ou bien d'un minimum ? Ce sera le sujet d'un prochain défi...



Ces deux expressions de a et b sont calculées par le programme. Il suffit d'en suivre le cheminement opération par opération pour le constater. Difficile d'optimiser plus encore le programme des deux vainqueurs du défi. Mais, il ne faut jurer de rien.

Jean-Christophe KRUST

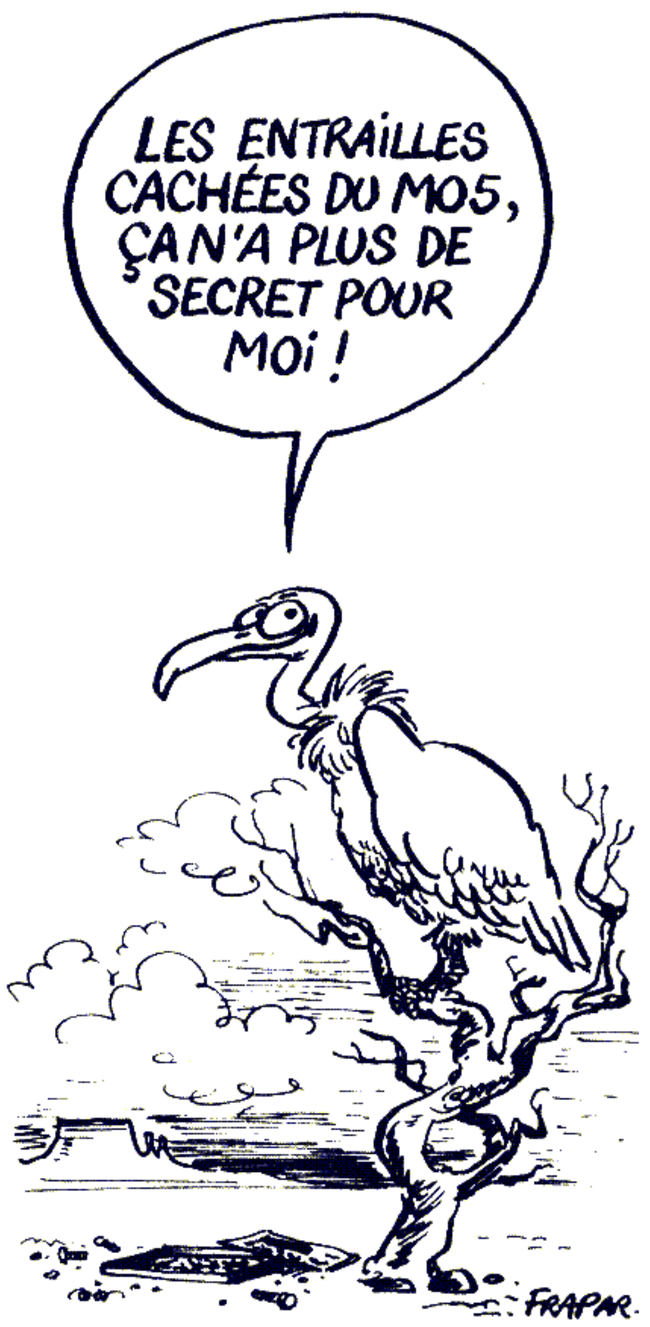
LA BOÎTE A MALICES...

PRENEZ un programme et ôtez-en très soigneusement toutes les astuces, des plus élémentaires aux plus subtiles. Vous êtes certain de n'en avoir laissé passer aucune ? Bien. Que reste-t-il ? Rien, ou peut-être une bogue ou deux (tout le monde peut se tromper). En fait, tout programme n'est qu'une suite d'astuces. Dans les pages qui suivent, vous en trouverez un grand nombre. Certaines sont de portée très générale. D'autres ne valent que pour un matériel particulier. Mais dans tous les cas, vous aurez intérêt à être curieux, à fouiner dans la boîte à malices. Même s'il ne s'agit pas de votre machine, vous trouverez souvent des idées à reprendre. Par ailleurs, vous avez sans doute vos propres recettes, vos façons de faire... Si ce ne sont pas des secrets que vous cherchez à conserver jalousement, faites-en part au journal. Celles qui nous paraîtront les plus intéressantes enrichiront à leur tour la boîte à malices. Tous les lecteurs pourront ainsi en profiter.

LIST

INCURSIONS DANS LE BASIC

Le Basic du MO5 réside dans une EPROM (Erasable Programmable Read Only Memory, mémoire morte effaçable et programmable avec un outillage spécialisé) de 16 Koctets, implantée tout en haut de la mémoire, entre les adresses $C000 et $FFFF. L'étude du premier Koctet peut se révéler pleine d'enseignements pour le curieux.



Les adresses $C0A0 à $C25E renferment la liste des mots clés acceptés — du moins en théorie — par le Basic (voir le tableau des premiers octets, page suivante).

A leur sujet, il est possible de faire quelques remarques :

- l'absence de quelques mots clés est un phénomène tout à fait normal si l'on prend en compte le fait que l'espace mémoire occupé par ce Basic est

```
.: C000 D5 D9 D6 08 D5 9E CA 6A D6 AB D6 F1 D7 34 D7 AE  UYV.U.J*V+V1W4W.
.: C010 D7 B5 D8 0B CC D4 CC 11 CB 64 CB 9F CC 0E CC 5B  W5X.LTL.K$K.L.L[
.: C020 E3 BE D6 27 21 E9 21 F9 D5 E0 22 3C 21 F9 E8 9F  #>V'!)!9U "<!9<.
.: C030 E8 B1 CC 52 CC 25 CC 1E CC 67 CC E7 D0 D9 DB B6  <1LRL%L.L'L'PY[6
.: C040 E4 6E E3 DE E4 69 E7 14 E7 1B E4 36 E8 AD 22 DB  $.#↑$)'.'.$6<-"[
.: C050 26 EB 28 23 2E EB C1 D4 C2 D0 C5 19 C7 E5 C8 00  &+<#.+ATBPE.G%H.
.: C060 FF AE 04 00 00 03 02 79 D2 AC 79 D2 9F 7C D2 EE  ?......9R,9R.<R.
.: C070 7C D5 1C 7F D6 B4 50 C8 86 46 C8 8C 3C C8 92 32  <U.?V4PH.FH.<H.2
.: C080 C8 9D 28 C8 98 7A D3 77 7B D3 4A 64 C8 7F 64 C8  H.<H.:S7;SJ$H?$H
.: C090 65 64 C8 6D D3 D4 C8 79 D5 D4 C8 73 D5 D3 C8 7C  %$H-STH9UTH3USH<
.: C0A0 45 4E C4 46 4F D2 4E 45 58 D4 44 41 54 C1 44 49  ENDFORNEXTDATADI
.: C0B0 CD 52 45 41 C4 FF 47 CF 52 55 CE 49 C6 52 45 53  MREAD?GORUNIFRES
.: C0C0 54 4F 52 C5 52 45 54 55 52 CE 52 45 CD A7 53 54  TORERETURNREM'ST
.: C0D0 4F D0 45 4C 53 C5 54 52 4F CE 54 52 4F 46 C6 44  OPELSETRONTROFFD
.: C0E0 45 46 53 54 D2 44 45 46 49 4E D4 44 45 46 53 4E  EFSTRDEFINTDEFSN
.: C0F0 C7 FF 4F CE 54 55 4E C5 45 52 52 4F D2 52 45 53  G?ONTUNEERRORRES
.: C100 55 4D C5 41 55 54 CF 44 45 4C 45 54 C5 4C 4F 43  UMEAUTODELETELOC
.: C110 41 54 C5 43 4C D3 43 4F 4E 53 4F 4C C5 50 53 45  ATECLSCONSOLEPSE
.: C120 D4 4D 4F 54 4F D2 53 4B 49 50 C6 45 58 45 C3 42  TMOTORSKIPFEXECB
.: C130 45 45 D0 43 4F 4C 4F D2 4C 49 4E C5 42 4F D8 FF  EEPCOLORLINEBOX?
.: C140 41 54 54 52 C2 44 45 C6 50 4F 4B C5 50 52 49 4E  ATTRBDEFPOKEPRIN
.: C150 D4 43 4F 4E D4 4C 49 53 D4 43 4C 45 41 D2 44 4F  TCONTLISTCLEARDO
.: C160 D3 FF 4E 45 D7 53 41 56 C5 4C 4F 41 C4 4D 45 52  S?NEWSAVELOADMER
.: C170 47 C5 4F 50 45 CE 43 4C 4F 53 C5 49 4E 50 45 CE  GEOPENCLOSEINPEN
.: C180 50 45 CE 50 4C 41 D9 54 41 42 A8 54 CF 53 55 C2  PENPLAYTAB<TOSUB
.: C190 46 CE 53 50 43 A8 55 53 49 4E C7 55 53 D2 45 52  FNSPC<USINGUSRER
.: C1A0 CC 45 52 D2 4F 46 C6 54 48 45 CE 4E 4F D4 53 54  LERROFFTHENNOTST
.: C1B0 45 D0 AB AD AA AF DE 41 4E C4 4F D2 58 4F D2 45  EP+-*/↑ANDORXORE
.: C1C0 51 D6 49 4D D0 4D 4F C4 C0 BE BD BC 53 47 CE 49  QVIMPMOD@>=<SGNI
.: C1D0 4E D4 41 42 D3 46 52 C5 53 51 D2 4C 4F C7 45 58  NTABSFRESQRLOGEX
.: C1E0 D0 43 4F D3 53 49 CE 54 41 CE 50 45 45 CB 4C 45  PCOSSINTANPEEKLE
.: C1F0 CE 53 54 52 A4 56 41 CC 41 53 C3 43 48 52 A4 45  NSTR$VALASCCHR$E
.: C200 4F C6 43 49 4E D4 FF FF 46 49 D8 48 45 58 A4 FF  OFCINT??FIXHEX$?
.: C210 53 54 49 43 CB 53 54 52 49 C7 47 52 A4 4C 45 46  STICKSTRIGGR$LEF
.: C220 54 A4 52 49 47 48 54 A4 4D 49 44 A4 49 4E 53 54  T$RIGHT$MID$INST
.: C230 D2 56 41 52 50 54 D2 52 4E C4 49 4E 4B 45 59 A4  RVARPTRRNDINKEY$
.: C240 49 4E 50 55 D4 43 53 52 4C 49 CE 50 4F 49 4E D4  INPUTCSRLINPOINT
.: C250 53 43 52 45 45 CE 50 4F D3 50 54 52 49 C7 00 C4  SCREENPOSPTRIG.D
.: C260 9E CF 97 D0 5C C5 E2 CA 61 DD 3B 21 F9 C5 5F C5  .O.P\E"J!];!9E←E
.: C270 4B C5 EA C4 8A C5 C1 C5 E5 C5 E5 C4 A7 C5 E5 CF  KE*D.EAE%E%D'E%O
.: C280 1F CF 20 CE EC CE EF CE E9 21 F9 C6 1F E9 00 CF  .O N,N/N)!9F.).O
.: C290 24 CF 2D 22 36 CE C5 E5 FE E8 FC E6 40 E7 93 EC  $O-"6NE%)<<&@',,
.: C2A0 6C EB CE E8 E9 E8 FA E6 B3 E8 1B E8 2C 21 F9 E6  ,+N()(:&3(.(,!9&
.: C2B0 1C E8 68 CC DD CD 7A C4 CF E0 6A C4 DE F0 00 21  .((L]M:DO *D↑0.!
.: C2C0 F9 C4 22 E0 A3 E2 8C E2 58 E1 E9 E1 B8 E8 C3 21  9D" #"."X!)!8<C!
```

quelque peu réduit par rapport à celui du « grand frère » TO7. Dommage pour les puristes. Le LET a fait les frais de cette opération d'élagage, accompagné en cela par les OCT$, CDBL, CSNG, DEFFN, et d'autres dont l'absence est regrettable ;

• des mots clés apparaissent dans la liste (allez savoir pourquoi !) alors que leur emploi se solde par un message d'erreur : c'est le cas, par exemple, de AUTO.

Enfin, il est permis de s'interroger sur la présence de DEFSNG, alors que DEFDBL a disparu corps et biens.

Les adresses $C25F à $C2D1 contiennent une première table des adresses d'exécution des routines correspondant aux différents mots clés. En voici donc quelques-unes, que vous pourrez essayer :

Instruction	Exec
END	&HC49E
RUN	&HC54B
TRON	&HCF1F
TROFF	&HCF20
CLS	&HE8FC
BEEP	&HE8FA
CONT	&HC4CF
LIST	&HE06A

Chacun pourra compléter la liste, la

Les premiers octets du Basic MO5

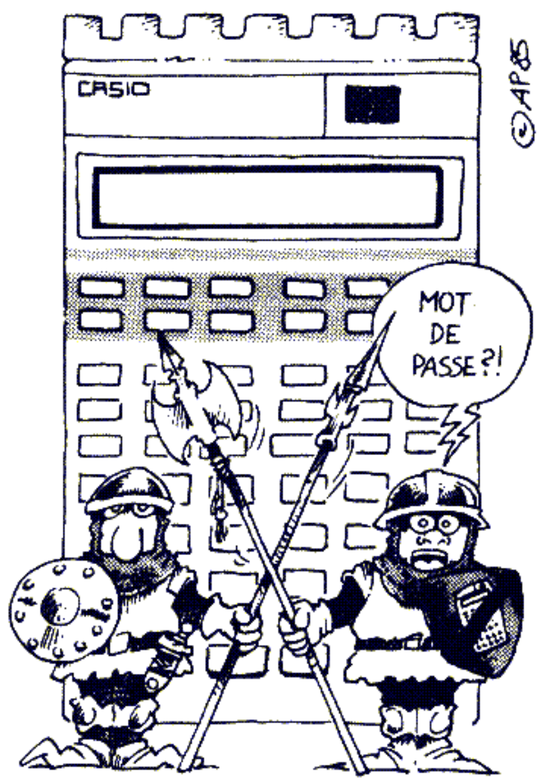
nôtre ne concernant que les routines les plus simples, celles qui n'ont en principe besoin d'aucun paramètre pour pouvoir fonctionner. Vous remplacerez ainsi dans vos propres programmes le END par un EXEC 50334, ce qui fait tout de même plus sérieux. Et d'ailleurs, pourquoi faire simple si on peut faire compliqué, n'est-ce pas ?

Il est amusant de constater que REM et ELSE commencent à la même adresse. Ce qui a pour effet secondaire d'autoriser le remplacement du premier par le second, tandis que l'inverse n'est pas forcément vrai : *100 ELSE ceci est une remarque* fonctionne donc parfaitement. Tandis que : 200 IF A > 10 THEN PRINT A*2 REM PRINT A/2 n'est pas apprécié par l'interpréteur !

Vous pouvez poursuivre les investigations dans les entrailles cachées de votre machine et faire ainsi d'intéressantes découvertes... que vous nous ferez partager sans doute.

Robin BOIS

FX-602P



UNE MEILLEURE PROTECTION

Sur la FX-602P, un mot de passe de quatre caractères protège l'entrée dans la liste d'un programme. Mais on peut très bien alors « faire tourner » le programme.

```
*** P0
MAC
"PASSE ?"
PAUSE
PAUSE
PAUSE MinF
23456 EXP 5 x=F
GOTO1
GOTO0
LBL1
"Programme "
HLT
LBL0
AC
          ...041steps
```

Pour empêcher cela, il fallait un petit utilitaire. Le mot de passe de celui qui est présenté ici est : 23456 EXP 5. Il ne faut manquer ni de réflexe ni de mémoire pour l'introduire au moment où il est demandé. La présence de trois PAUSE ne suffit pas toujours.

Et n'oubliez pas de protéger aussi ce petit utilitaire par un mot de passe. On n'est jamais trop prudent.

Alain CHAUVAUX

APPLESOFT

UNE NOUVELLE INSTRUCTION

Voici un court programme qui vous permettra d'initialiser une disquette comportant une nouvelle instruction du système d'exploitation des disquettes (SED), à la place de la fonction VERIFY. Cette instruction est sollicitée par FREE et indique à l'utilisateur le nombre de secteurs restant libres sur la disquette. Elle peut être employée soit dans un programme :

10 PRINT CHR$(4); "FREE"

soit comme une commande, en tapant directement FREE au clavier. On aura, auparavant, effectué un CATALOG.

Pour modifier le SED et installer ce que les Anglais appellent un "patch", le programme utilise une routine dite de S.H. Lam qui mérite quelques explications. A ce sujet, je tiens à remercier Jean-François Duvivier pour son article « S.H. Lam, une routine bien pratique » publié par la revue *Pom's*.

La technique consiste à placer dans une chaîne de caractères des commandes à destination du moniteur. On peut ainsi installer en mémoire un court programme en langage-machine avec une relative facilité et sans un nombre invraisemblable de POKEs ou de READ/DATAs. L'installation en mémoire s'effectue en trois étapes.

La chaîne est d'abord scindée en caractères qui sont placés dans le tampon du clavier les uns à la suite des autres. Cette opération occupe le tampon à partir de l'adresse $200 (512 en décimal). Dans un deuxième temps, le CALL-144 appelle un sous-programme résident qui met en œuvre les routines examinant le tampon-clavier et vérifiant l'exécution des commandes inscrites dans la chaîne de caractères. En dernier lieu, il faut revenir au Basic. Pour ce faire, il suffit de se brancher à l'adresse $D9C6, ce qui correspond à un retour au mode RUN.

Pour inclure FREE dans le SED, il faut, bien sûr, que les adresses $B6B3 à $B6FD n'aient pas été déjà utilisées par un "patch". On lance le programme puis on initialise la disquette en tapant "INIT *nom du programme*". A noter que la fonction FREE, justement, ne supprime pas INIT, mais VERIFY.

Marcel COTTINI

Secteurs libres
Patch pour Apple II, II+ ou IIe
Auteur Marcel Cottini
Copyright *LIST* et l'auteur

```
10 HEX$ = "B6B3:A9 BD 20 F0 FD A9 00 85 40 85 41 A2 8C BD F2 B3 0A 90 06
    E6 40 D0 02 E6 41 D0 F5 CA D0 EF A6 40 A5 41 AC 00 E0 C0 4C F0 03 4C
    1B E5 4C 24 ED NA902:46 52 45 C5 00 N9D54:B2 B6 NA93F:40 00"
20 HEX$ = HEX$ + " ND9C6G"
30 FOR I = 1 TO  LEN (HEX$)
40 POKE 511 + I, ASC ( MID$ (HEX$,I,1)) + 128
50 NEXT : POKE 72,0: CALL  - 144
```

X-07

UN MOUCHARD DANS LE CANON

Lors de chaque extinction, le X-07 mémorise l'heure qu'il est dans trois octets situés en fin de mémoire vive. Les heures, minutes et secondes sont conservées aux adresses RAMEND + 2, RAMEND + 3 et RAMEND + 4, RAMEND étant le pointeur de fin de mémoire. Le programme proposé ici permet d'afficher l'heure à laquelle votre machine s'est éteinte et dans certains cas (qui sait ?) l'heure à laquelle votre fils ou votre frère s'est couché. Plus sérieusement, ce programme peut aider à calculer la durée moyenne de fonctionnement de votre X-07 ainsi qu'à calculer avec précision la durée de vie de vos piles. Mais vous trouverez certainement d'autres applications.

Patrick LECLERCQ

```
10 REM HEURE DE LA COUPURE DU SYSTEME
20 DEFFNP(X)=PEEK(X)+PEEK(X+1)*256:'LECT
URE D'UN POINTEUR
30 AD=FNP(&H212):'CALCUL DE L'ADRESSE DE
 RAMEND
40 H=PEEK(AD+2):'LECTURE DES HEURES
50 M=PEEK(AD+3):'LECTURE DES MINUTES
60 S=PEEK(AD+4):'LECTURE DES SECONDES
70 PRINT"Le systeme a ete coupe a :"
80 PRINTH;"h";M;"mn";S"s"
90 END
```

DAI

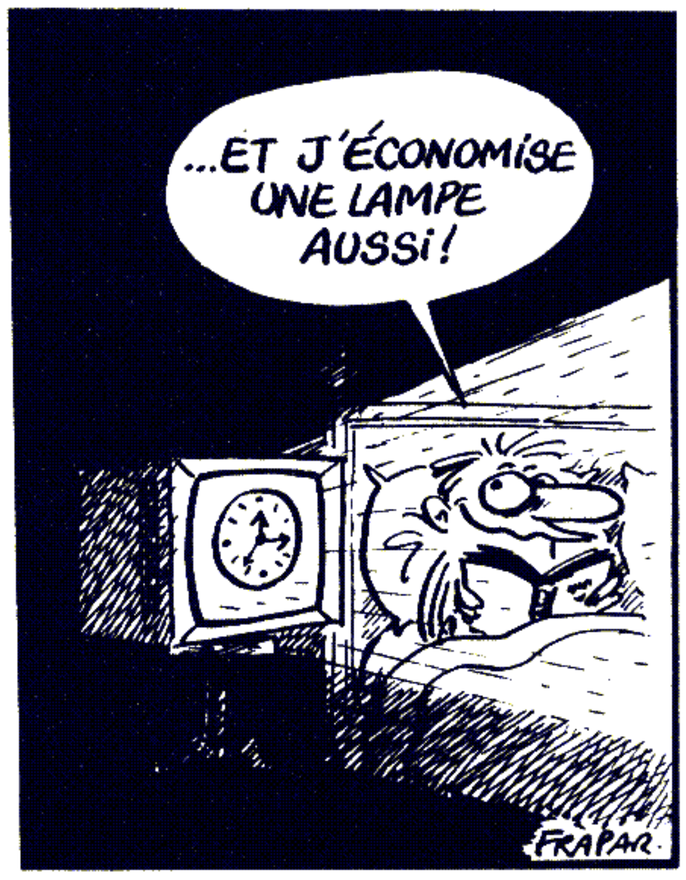
HORLOGE GRAPHIQUE

Vous pourrez désormais placer votre téléviseur sur votre table de chevet !

Ce programme simule le fonctionnement d'une pendule à cadran à trois aiguilles. Il fonctionne sous tous les modes graphiques ou demi-graphiques pairs (4 couleurs). Totalement indépendant du Basic, grâce à sa gestion par l'interruption 7, il pourra aisément

s'intégrer dans des jeux de société ou de simulation.

Le chargeur Basic réserve un espace suffisant à la routine machine dans le tableau MLP(110). Si ce programme est intégré dans un autre logiciel, il faudra veiller à ce que ce tableau soit déclaré en premier, afin qu'il commence effectivement en #2F0. Il est possible d'accéder directement aux variables de l'horloge, pour une utilisation quelconque. Ainsi, PEEK (#2F5) donne le nombre de secondes, PEEK (#2F6) le nombre de minutes, et INT(PEEK(#2F7)/3) MOD 12 le nombre d'heures.



L'horloge peut également servir de chronomètre au dixième de seconde (évidemment, pas pratique d'emporter le Dai sur un terrain de sport !). Pour cela, faire POKE #30A,5. Attention : un passage direct en mode 0 (LIST, par exemple) peut parfois « planter » le Dai, si l'interruption 7 n'est pas restaurée. Avant toute manipulation, il vaut donc mieux faire RUN 80, qui remet les choses dans l'état initial.

Alain NORMAND

Horloge graphique
Programme pour Dai
Auteur Alain Normand
Copyright *LIST* et l'auteur

```
10    REM
20    REM ***  HORLOGE GRAPHIQUE ***
30    REM
40    CLEAR 1000:DIM MLP(110)
50    RESTORE:FOR I=1 TO 24:READ DUMMY:NEXT
60    FOR I=#2F0 TO #4A1:READ DA:POKE I,DA:NEXT
70    GOTO 90
80    POKE #70,#A9:POKE #71,#D9:END
90    MODE 0:COLORT 4 13 4 4
100   PRINT CHR$(12):PRINT :PRINT
110   INPUT "COORDONNEES DU CENTRE DE L'HORLOGE (X,Y) ";X,Y
120   POKE #2F0,Y:POKE #2F2,X MOD 256:POKE #2F3,INT(X/256)
130   PRINT :PRINT :INPUT "HEURE (H,M,S) ";H,M,S
140   IF H<0 OR M<0 OR S<0 GOTO 120
150   H=(H MOD 12)*3:M=M MOD 60:S=S MOD 60
160   IF M>19 THEN H=H+1:IF M>39 THEN H=H+1
170   POKE #2F5,S:POKE #2F6,M:POKE #2F7,H
180   MODE 6A:COLORG 0 5 10 15:PRINT CHR$(12);
190   POKE #70,0:POKE #71,3
200   RESTORE:FOR I=1 TO 12:READ C,D:C=X+C:D=Y+D
210   FILL C,D C+2,D+2 23:NEXT I
220   DATA -1,31,15,27,26,16,30,-1,26,-18,15,-29
230   DATA -1,-33,-17,-29,-28,-18,-32,-1,-28,16,-17,27
235   REM
240   DATA #00,#00,#00,#00,#00,#00,#00,#00,#00,#00,#00,#00,#00,#00,#00,#00
250   DATA #E5,#D5,#C5,#F5,#21,#F4,#02,#34,#7E,#FE,#32,#C2,#9B,#03,#36,#00
260   DATA #21,#F5,#02,#7E,#87,#5F,#CD,#A2,#03,#21,#F5,#02,#34,#7E,#FE,#3C
270   DATA #C2,#5A,#03,#36,#00,#23,#7E,#87,#5F,#CD,#A2,#03,#21,#F6,#02,#34
280   DATA #7E,#FE,#14,#CA,#42,#03,#FE,#28,#CA,#42,#03,#FE,#3C,#C2,#5A,#03
290   DATA #36,#00,#23,#7E,#87,#5F,#21,#E0,#03,#3E,#14,#CD,#A7,#03,#21,#F7
300   DATA #02,#34,#7E,#FE,#24,#C2,#5E,#03,#36,#00,#21,#F7,#02,#7E,#87,#5F
310   DATA #21,#E0,#03,#3E,#15,#CD,#A7,#03,#21,#F6,#02,#7E,#87,#5F,#3E,#15
320   DATA #CD,#A4,#03,#21,#F5,#02,#7E,#87,#5F,#3E,#16,#CD,#A4,#03,#1E,#78
330   DATA #3E,#24,#32,#CE,#03,#3E,#16,#21,#2B,#0D,#22,#CB,#03,#CD,#A4,#03
340   DATA #3E,#21,#32,#CE,#03,#21,#00,#00,#22,#CB,#03,#F1,#C1,#D1,#E1,#C3
350   DATA #A9,#D9,#3E,#14,#21,#28,#04,#F5,#AF,#47,#57,#19,#5E,#23,#4E,#B1
360   DATA #F2,#B5,#03,#06,#FF,#2A,#F0,#02,#7D,#09,#45,#4F,#AF,#B3,#F2,#C3
370   DATA #03,#16,#FF,#2A,#F2,#02,#E5,#19,#EB,#E1,#F1,#00,#00,#EF,#21,#D0
380   DATA #21,#00,#00,#22,#17,#01,#21,#A9,#D9,#22,#70,#00,#C3,#02,#E6,#00
390   DATA #00,#11,#03,#11,#06,#10,#09,#0F,#0B,#0D,#0D,#0B,#0F,#09,#10,#06
400   DATA #11,#03,#11,#00,#11,#FD,#10,#FA,#0F,#F7,#0D,#F5,#0B,#F3,#09,#F1
410   DATA #06,#F0,#03,#EF,#00,#EF,#FD,#EF,#FA,#F0,#F7,#F1,#F5,#F3,#F3,#F5
420   DATA #F1,#F7,#F0,#FA,#EF,#FD,#EF,#00,#EF,#03,#F0,#06,#F1,#09,#F3,#0B
430   DATA #F5,#0D,#F7,#0F,#FA,#10,#FD,#11,#00,#1D,#03,#1D,#06,#1C,#09,#1B
440   DATA #0C,#1A,#0E,#19,#10,#17,#12,#15,#14,#13,#16,#11,#18,#0F,#19,#0C
450   DATA #1A,#09,#1B,#06,#1C,#03,#1C,#00,#1C,#FD,#1B,#FA,#1A,#F7,#19,#F4
460   DATA #18,#F1,#16,#EF,#14,#ED,#12,#EB,#10,#E9,#0E,#E7,#0C,#E6,#09,#E5
470   DATA #06,#E4,#03,#E3,#00,#E3,#FD,#E3,#FA,#E4,#F7,#E5,#F4,#E6,#F2,#E7
480   DATA #F0,#E9,#EE,#EB,#EC,#ED,#EA,#EF,#E8,#F1,#E7,#F4,#E6,#F7,#E5,#FA
490   DATA #E4,#FD,#E4,#00,#E4,#03,#E5,#06,#E6,#09,#E7,#0C,#E8,#0F,#EA,#11
500   DATA #EC,#13,#EE,#15,#F0,#17,#F2,#19,#F4,#1A,#F7,#1B,#FA,#1C,#FD,#1D
510   DATA #01,#01
```

CRIC, CRAC

Cric-crac
Programme pour PC-1500
Auteur Frédéric Charles
Copyright *LIST* et l'auteur

Encore un tour... Cric, crac. Ça y est, le coffre est ouvert ! Filons !

L'aurez-vous deviné ? Le PC-1500 est devenu un véritable coffre-fort dont **vous** devez faire sauter la serrure. Facile ? Non seulement il vous faudra de l'oreille et de la jugeotte pour trouver la clef mais vous devrez aussi découvrir la nature du système d'alarme.

Ce coffre possède une combinaison à quatre chiffres. Pour la trouver, il faut faire tourner un à un les quatre barillets jusqu'à l'affichage de la bonne solution. A chaque tour de barillet, l'ordinateur émet un petit clic sonore. Avec une bonne oreille, on parvient à trouver la combinaison. Ce sont les touches de fonction situées au-dessous de chaque barillet affiché qu'il faut presser pour les faire tourner.

Attention : le coffre est protégé par un système d'alarme connecté au commissariat voisin et commandant une sirène. De plus, un système de brouillage a aussi été connecté sur la serrure ! Le mode exact de fonctionnement de cet attirail de protection vous est inconnu : découvrez-le vite, car le temps vous est compté, vous pouvez être arrêté.

Si la sirène d'alarme se déclenche, vous êtes pris. Pour rejouer, dix années plus tard, faites BREAK puis RUN. Lorsque vous aurez enfin réussi à ouvrir le coffre, une mélodie se fera entendre saluant votre victoire.

Le programme ne pose aucun problème d'introduction pour ce qui concerne sa partie Basic. En revanche, faire très attention pour introduire les codes du langage-machine rangés en lignes de DATA. Une seule petite erreur et il faudra tout recommencer.

A vos barillets, donc, et bonne chance ! Solution dans le prochain numéro de *LIST*.

Frédéric CHARLES

```
 1:CLEAR :RESTORE
   "CODE":CO=PEEK
   &7863*256+&C5:
   FOR I=CO TO I+5
   3:READ J:POKE
   I,J:NEXT I
 5:WAIT 0:RANDOM
   :BEEP ON :BEEP
   3:PRINT "    *
   ** COFFRE FORT
   ":J$="    "
10:FOR I=1 TO 4:
   CALL CO+29:FOR
   J=1 TO 15:NEXT
   J:NEXT I
15:FOR I=6 TO 9:@$
   (I)=STR$ (RND
   10-1):NEXT I
20:FOR I=1 TO 4:@$
   (I)=STR$ (RND
   10-1):NEXT I
25:TIME =0:CLS
30:E$=A$+J$+B$+J$
   +C$+J$+D$:
   CURSOR 0:PRINT
   "  "+E$
32:IF F$=A$IF B$=
   G$IF C$=H$IF D
   $=I$CURSOR 18:
   PRINT "CLIC!!!
   ":GOTO 300
35:E=ASC INKEY$ :
   IF E=0THEN 65
40:IF E<17OR E>20
   THEN 65
45:E=E-16:IF F=E
   LET U=U+1:GOTO
   55
50:U=0
55:F=E:A=VAL @$(E
   )+1:IF A=10LET
   A=0
60:@$(E)=STR$ A:
   BEEP 1,ABS (A-
   VAL @$(E+5))*2
   +1,100
65:IF TIME >15E-4
   CALL CO+29:FOR
   I=1 TO 9:NEXT I
   :CALL CO+29:U=
   U+1:GOTO (U=5)
   *290+20
70:IF U>9THEN 310
75:GOTO 30
300:FOR I=1 TO 60:
   BEEP 1,RND 25,
   100:NEXT I
305:GOTO 305
310:CURSOR 18:
   PRINT "VROUIN!
   !
315:CALL CO:GOTO 3
   15
400:"CODE"DATA 72,
   1,74,5,106,75,
   90,1,190,230,1
   11,98,110,0,15
   3,8,190,230,11
   1,96
410:DATA 110,100,1
   45,8,82,94,0,1
   53,21,72,112,7
   4,0,106,0,5,18
   9,255,65,78,78
   ,153,8
420:DATA 72,113,74
   ,0,96,110,2,15
   3,17,154,0,0,0
```

DES TRUCS EN VRAC

Dans *LIST* 4, page 73, on avait déjà trouvé quelques trucs destinés à faciliter la vie des programmeurs sur C. 64. En voici d'autres, concernant principalement l'affichage, que je vous livre pêle-mêle.

Avec POKE 204,0 (clignotement du curseur pendant un GET), il est possible qu'en sortie le curseur reste visible selon qu'il est allumé ou non. Pour remédier à ce défaut, il faut prendre le caractère sous le curseur **avant** le GET et le réafficher en sortie. C'est facile à réaliser quand on sait que :

• l'octet 206 contient le caractère sous le curseur ;

• les octets 209 et 210 contiennent l'adresse de la ligne où se situe le curseur ;

• l'octet 211 contient la position du curseur dans la ligne.

Ainsi POKE (PEEK (211) + PEEK (209) + PEEK (210) * 256), PEEK (206) imprime ce qu'il y avait sous le curseur. Sur ce point précis comme sur la plupart de ceux qui vont suivre, on pourra se reporter au programme ci-dessous.

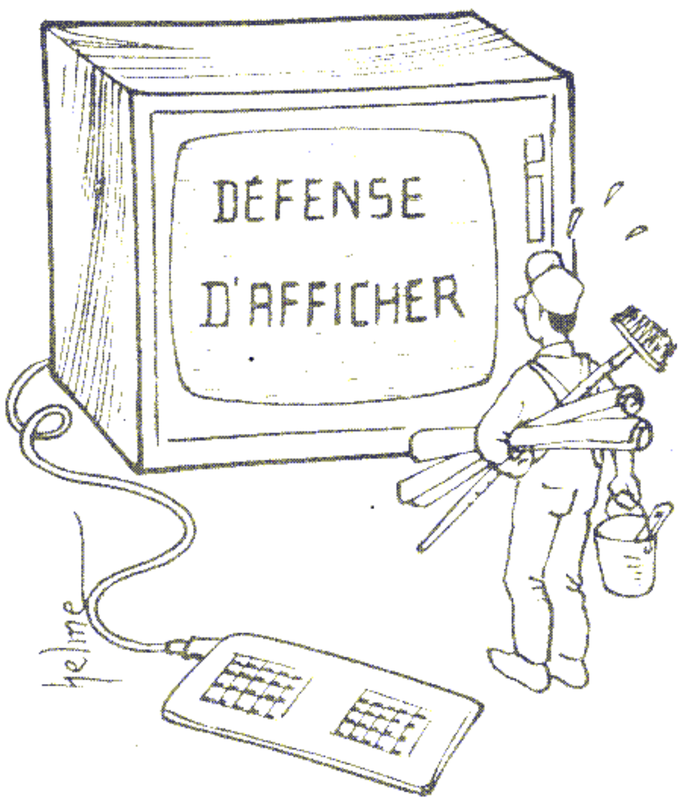
La séquence POKE 214, 12 : PRINT : PRINT « TEXTE » n'affiche pas le texte sur la douzième ligne d'écran, mais sous la quatorzième. En effet, la première ligne de l'écran porte le numéro 0, et l'instruction PRINT fait effectuer un saut de ligne. Le programme donné ci-dessous numérote les lignes de 1 à 25 et les colonnes de 1 à 40.

Pour choisir de faire sortir les informations sur l'écran ou sur l'imprimante (voir *Sur la piste directory,* dans *LIST* 4, page 54), il suffit d'utiliser le mode « CMD », et les informations en sortie seront dirigées soit vers l'écran soit vers l'imprimante, et cela sans modification du programme.

Les octets 53270 et 53265 recèlent

Du neuf sur les écrans
Utilitaires pour C. 64
Auteur Gérard Foucault
Copyright *LIST* et l'auteur

```
10 REM ****************************
20 REM *                          *
30 REM *       UTILITAIRES        *
40 REM *                          *
50 REM *   AUTEUR G. FOUCAULT     *
60 REM *                          *
70 REM ****************************
80 :
90 POKE 53280,6:POKE 53281,6
100 PV=10:PH=6:GOSUB 9190
110 :
120 PRINT"DEMONSTRATION  D'UTILITAIRES"
130 :
140 GOSUB 820
150 PV=5:PH=5:GOSUB 9190
160 PRINT"ESSAI DE LA FONCTION GET
170 PV=10:PH=0:GOSUB9200
180 GOSUB 9050
190 A=25:IF IMPRIME THEN A=5
200 FOR I = 1 TO A
210 PRINT TAB(10)"BONJOUR LIST"
220 NEXT I
230 GOSUB 9130:
240 GOSUB 9340
250 PV=5:PH=2:GOSUB 9190
260 :
270 PRINT"DEROULEMENT DE 1 PIXEL VERS LA GAUCHE
280 :
290 PV=7:PH=6:GOSUB 9200
300 PRINT"D'UNE CHAINE DE 40 CARACTERES"
310 GOSUB 820
320 POKE 53270,PEEK(53270)AND 240
330 A$=" TSIL RUOJNOB TSIL RUOJNOB TSIL RUOJNOB"
340 PV=10:PH=1:GOSUB 9190
350 FOR I=LEN(A$) TO 1 STEP -1
360 B$=B$+MID$(A$,I,1)
370 PH=I:GOSUB 9200:
380 POKE 53270,(PEEK(53270)AND 240)+7:PRINT B$
390 FOR D=6 TO 0 STEP -1
400 POKE 53270,(PEEK(53270)AND 240)+D:
410 GOSUB 860:NEXT D:NEXT I
420 POKE 53270,(PEEK(53270)AND 240)OR 8
430 GOSUB 9340
440 PV=5:PH=2:GOSUB 9190
450 :
460 PRINT"DEROULEMENT DE 1 PIXEL VERS LA DROITE
470 :
480 PV=7:PH=8:GOSUB 9200
490 PRINT"D'UNE CHAINE DE CARACTERES"
500 GOSUB 820
510 B$="":POKE 53270,PEEK(53270)AND 240
520 A$="BONJOUR LIST BONJOUR LIST BONJOUR LIST  BONJOUR LIST
    BONJOUR LIST"
530 PV=10:PH=1:GOSUB 9190
540 FOR I=LEN(A$) TO 1 STEP -1
550 B$=MID$(A$,I,1)+B$
560 IF LEN(B$)> 40 THEN B$=LEFT$(B$,40)
570 GOSUB 9200
580 POKE 53270,(PEEK(53270)AND 248):PRINT B$
590 FOR D=0 TO 6
600 POKE 53270,(PEEK(53270)AND 248)+D
610 GOSUB 860:NEXT D:NEXT I
620 POKE 53270,(PEEK(53270)AND 240)OR 0
630 GOSUB 9340
640 PV=5:PH=2:GOSUB 9190
650 :
660 PRINT"DEROULEMENT DE 1 PIXEL VERS LE HAUT
670 :
680 PV=7:PH=12:GOSUB 9200
690 PRINT"DE L'ECRAN"
700 GOSUB 820
710 PV=25:PH=10:GOSUB 9190
720 FORI=1 TO25
730 POKE53265,(PEEK(53265)AND240)OR7
740 PRINT:GOSUB9230
750 PRINT "BONJOUR LIST";
760 FOR D=6 TO 0 STEP -1
770 POKE53265,(PEEK(53265)AND240)ORD
780 GOSUB 860:NEXT D:NEXT I
790 POKE53265,27
800 GOSUB 9340:PRINTCHR$(147):END
810 :
820 PV=15:PH=2:GOSUB 9200
830 PRINT"APPUYEZ SUR UNE TOUCHE POUR COMMENCER
840 GOSUB9290:RETURN
850 :
860 FORK=0TO 50:NEXT K:RETURN
870 :
8960 END
8970 :
8980 REM *** SOUS PROGRAMME GET A$
8990 :
9000 POKE204,0:POKE198,0:WAIT198,1:GET A$:POKE204,1
9010 POKE(PEEK(211)+PEEK(209)+PEEK(210)*256),PEEK(206):RETURN
9020 :
9030 REM *** SOUS PROGRAMME IMPRIMANTE
9040 :
9050 PRINT "SORTIE SUR IMPRIMANTE  O/N ==> ";
9060 GOSUB 9000:PRINT A$
9070 IF A$<>"O" THEN IMPRIME=0:RETURN
9080 PRINT "    TEXTE OU GRAPHIQUE T/G ==> ";
9090 GOSUB 9000:PRINT A$:IMPRIME=1
9100 IF A$<>"M" THEN OPEN9,4:CMD9:RETURN
9110 OPEN9,4,7:CMD9:RETURN
9120 :
9130 IF IMPRIME THEN PRINT#9:CLOSE9
9140 RETURN
9150 :
9160 REM *** SOUS PROGRAMME CURSEUR
9170 :
9180 REM PV=POSITION VERTICALE:PH=POSITION HORIZONTALE
9190 PRINTCHR$(147)
9200 IF PV > 25 THEN PV = 25
9210 PV=PV-2:IF PV < 0 THEN PV=-1:PRINT CHR$(19);:
9220 IF PV > -1 THEN POKE214,PV:PRINT
9230 PH=PH-1:IF PH<0 THEN PH=0
9240 IF PH>39 THEN PH=39
9250 POKE211,PH:PV=PV+2:PH=PH+1:RETURN
9260 :
9270 REM *** SOUS PROGRAMME CLIGNOTEMENT ECRAN
9280 :
9290 FOR ZZ=0 TO 1:ZZ=0
9300 POKE53265,PEEK(53265) OR 16:GOSUB 9340
9310 GET A$:IF A$ THEN ZZ=2:NEXT:RETURN
9320 POKE53265,PEEK(53265) AND 239:GOSUB 9340
9330 NEXT ZZ
9340 FOR IZ=0 TO 300:NEXT IZ:RETURN
```



quelques possibilités très intéressantes. Ainsi, à la mise sous tension, l'octet 53270 a pour valeur 200. Les bits 8, 7 et 4 sont à 1, ce qui nous fait bien 128 + 64 + 8 = 200. Pour allouer 40 colonnes à l'écran, il suffit de mettre le bit 4 à 1. Mis à 0, il n'en alloue plus que 38, auquel cas les colonnes 1 et 40 sont occultées ; le contenu n'en est pas perdu et l'affichage (masqué) est possible.

Par ailleurs, toujours dans l'octet 53270, les bits 1, 2 et 3 positionnent l'image sur l'écran. Le premier d'entre eux, mis à 1, décale l'image d'un pixel vers la droite. Le bit 2, mis à 1, décale l'image de 2 pixels, et les bits 1, 2 et 3 mis à 1 décalent l'image de 7 pixels vers la droite. Enfin, le bit 4 mis à 0 et les bits 1, 2 et 3 mis à 1 allouent 38 colonnes à l'écran, les colonnes 39 et 40 étant alors occultées.

En combinant ces différentes actions, on parvient à faire se dérouler des chaînes de caractères à l'écran, pixel par pixel, de la gauche vers la droite ou inversement.

A la mise sous tension, l'octet 53265 vaut, quant à lui, 27 : les bits 5, 4, 2 et 1 sont donc à 1. Si l'on met le bit 6 à 1, on passe en mode graphique. Le bit 5 à 0 occulte complètement l'écran. Toute la couleur du cadre est alors visible, mais les informations affichées à l'écran ne sont pas pour autant perdues.

Si l'on met le bit 4 à 1, on obtient un écran de 25 lignes (24 lignes quand il est à zéro). Comme dans l'octet 53270, les bits 1, 2 et 3 positionnent l'image sur l'écran. Les bits 1 et 2 mis à 0 descendent l'image de 3 pixels, et le bit 3 à 1 la décale vers le haut de 4 pixels. On peut ainsi faire défiler tout l'écran vers le haut pixel par pixel.

Cette dernière possibilité est illustrée, comme les précédentes, par le programme de démonstration.

Gérard FOUCAULT

TO7

GÉNÉRATION AUTOMATIQUE DE DATA

Lorsqu'un programme utilise des données en grand nombre sous forme de lignes de DATA, rien n'est plus fastidieux que de taper ces lignes. Il faut être sûr de la nature des DATA, valeurs numériques ou chaînes de caractères et de leur nombre.

Ce petit programme vous facilitera la tâche en numérotant et en tapant DATA au début de chaque ligne de façon automatique. Il vous permettra d'entrer vos données sans erreur conformément à un masque de saisie que vous aurez préalablement défini. Enfin, chose très importante, il mettra automatiquement les virgules entre chaque DATA.

Jean-Paul CARRÉ

```
100 '*** GENERATEUR DE LIGNES DE DATA ***
120 CLS
130 CLEAR1500
140 DIMLIGNE$(100)
150 INPUT"Numero de la 1ere ligne de Data a generer (Superieur au No de la
derniere ligne de votre programme) ";N
160 IF N<1000 THEN 150
170 INPUT"Increment entre lignes ";NA
180 IF NA<1 THEN 170
190 INPUT "Nombre de lignes (max 20 a la fois.)";NL
200 PRINT"Definissez un masque "
210 GOSUB 470
220 CLS
230 FOR I=0 TO (NL-1)*NA STEP NA
240 L=I+1000
250 LIGNE$(I)= STR$(L)+" DATA "
260 FOR H=1 TO NND
270 COLOR1,7:PRINT L$(H);:COLOR 7,0:INPUT D$
280 LIGNE$(I)=LIGNE$(I)+D$+","
290 NEXT H
300 LIGNE$(I)=LEFT$(LIGNE$(I),LEN(LIGNE$(I))-1)
310 NEXT I
350 CLS
360 COLOR 1,3
370 PRINT"Validation des lignes Data"
380 COLOR 7,0
390 PRINT :PRINT
400 FOR I=0 TO (NL-1)*NA STEP NA
410 PRINT LIGNE$(I)
420 NEXT I
430 LOCATE0,19:PRINT"Validez ces lignes en amenant le curseur sous chacune
d'elles puis en frappant";:COLOR0,3:LOCATE 20,21:PRINT"ENTREE ":COLOR 7,0
440 PRINT"faire DELETE-999";
450 PRINT" puis SAVE''DATAS'',A"
460 END
470 ' *** GENERATION DU MASQUE DE SAISIE
480 INPUT"Nombre de donnees par ligne ";NND
490 DIM L$(NND)
500 FOR H=1 TO NND
510 PRINT "Libelle de la donnee No";H;
520 INPUT L$(H)
530 NEXT H
540 RETURN
```

AMSTRAD

POKER AVEC ADRESSE SUR L'ÉCRAN

La gestion de l'écran par le système d'exploitation de l'Amstrad est remarquable sur bien des points. Il suffit pour s'en convaincre de manipuler un peu les fenêtres, ce qui est rendu très facile par l'intermédiaire du Basic. De même, la redéfinition des caractères est d'une étonnante simplicité, pour autant qu'on veuille se contenter de caractères monochromes.

Mais si vous envisagez des applications très élaborées (images en haute résolution multicolores, par exemple), vous devrez peut-être tourner vos regards du côté de la mémoire d'écran.

Celle-ci occupe le quart haut de la mémoire vive, entre les adresses &C000 et &FFFF. Ce qui représente tout de même la taille respectable de 16 Koctets ; cette zone étant modifiable...

Si ces adresses peuvent être gérées depuis le Basic à l'aide de simples

POKE, ou mieux encore en langage-machine, il est préférable d'étudier la zone mémoire-écran de plus près, car l'adressage des points sur l'écran y est tout à fait inhabituel.

L'origine écran (en haut à gauche) a pour adresse théorique &C000, ce qu'un POKE &C000, 255 pourra vous indiquer. En mode standard, ce POKE allume quatre pixels sur l'écran (8 bits pour 4 pixels, voilà l'indice d'une organisation spéciale des octets !). Il faudra donc deux POKE pour couvrir la largeur d'un caractère.

Pour allumer les quatre points d'à côté, il n'y a aucun problème avec un POKE &C001,255. Mais si vous voulez allumer les quatre points de gauche sur la seconde ligne de pixels, les choses se corsent : vous pouvez essayer d'ajouter 80 (en décimal) à l'adresse d'origine. Ce serait logique, mais faux : essayez POKE &C000+80,255... Les points allumés seront ceux de la huitième ligne de pixels, c'est-à-dire au sommet du caractère éventuellement situé sur la deuxième ligne de l'écran-texte.

L'allumage des pixels corrects doit se faire avec POKE &C000+&800, 255. Le décalage entre chaque ligne est donc de 2048 ! Le tableau ci-dessous vous en dira plus qu'un long discours sur l'enchaînement des adresses d'écran.

En bref, pour allumer un demi-caractère à l'origine-écran, il faudra taper : FOR A=0 TO 7:POKE &C000+A*&800,255:NEXT A

Et il y a pire... Lorsqu'un déroulement d'écran a lieu, provoqué par exemple par la descente du curseur, les adresses sont elles aussi décalées. Faites donc l'essai suivant : MODE 1, puis POKE &C000,255

Descendez le curseur jusqu'au bas de l'écran pour provoquer un scroll, puis tapez à nouveau POKE &C000,255. Intéressant, non ?

Robin BOIS

Le coin supérieur gauche de l'écran : adresses des premiers caractères

C000	C001	C002	C003	C004
C800	C801	C802	C803	
D000	D001	D002		
D800	D801	D802		
E000	E001			
E800	E801			
F000				
F800				
C050				
C850				
D050				
D850				
E050				
E850				
F050				
F850				
C0A0				
C8A0				

HECTOR HRX

VIDAGE EN FORTH

Si, pour une raison ou pour une autre, vous êtes curieux de savoir ce qui se trouve exactement dans la mémoire de votre ordinateur, vous avez besoin d'un petit programme de vidage (en anglais "dump"). En voici un, rédigé en Forth et destiné aux utilisateurs de l'Hector HRX.

On le lance de deux façons : soit en indiquant l'adresse du départ en décimal puis en tapant HEX et DUMP, soit avec HEX, l'adresse de départ en hexadécimal et DUMP. L'affichage s'effectue sur 7 colonnes. Les deux premières contiennent l'adresse en hexadécimal et en décimal. Les quatre suivantes correspondent à l'octet : valeur en hexa, traduction en ASCII, valeur en décimal et en binaire. La dernière colonne enfin indique la valeur sur deux octets.

En appuyant sur la touche 1, on augmente l'adresse de 10 (hexadécimal), soit 16 en décimal. En pressant 2, on avance de 100 (h), soit 256 (d) ; et avec 3, on avance de 1000 (h), soit 4096 (d). Inversement, on diminue l'adresse de 10, 100, 1000 (h), soit 16, 256 et 4096 (d) en pressant respectivement sur A, Z et E. L'appui sur toute autre touche fait passer à l'adresse immédiatement suivante. Enfin, on interrompt le vidage avec R/C.

Michel BROCHAND

Vidage en Forth
Programme pour Hector HRX
Auteur Michel Brochand
Copyright *LIST* et l'auteur

```
ECRAN N. 1 DUMP MEMOIRE
 HEX  : ADH DUP 0 <# # # # # #> TYPE
 SPACE  ; STANDARD
: ADDE DUP  DECIMAL 0 <# # # # # # #>
 TYPE SPACE 3A EMIT SPACE HEX ;
: AFOH DUP C@ 0 <# # # #> TYPE SPACE
 : AFODE DECIMAL DUP C@ 0 <# # # # #>
   TYPE SPACE ;
: AF?  C@  DUP 1F > SWAP 7F < AND ;
: AASCI DUP DUP AF? IF C EMIT SPACE
  ELSE  DROP 2E EMIT SPACE THEN ;
: AFBIN DUP 2 BASE ! C@ 0 <# # # # # #
 # # # #> TYPE SPACE HEX ;
: AV2OH DUP  @  DECIMAL 0 <# # # # # #
 #> TYPE HEX ;
: DUMP CR     BEGIN CR ADH ADDE AFOH
   AASCI AFODE AFBIN  AV2OH  KEY DUP
0D = NOT WHILE CASE 31 OF 10 + ENDOF
41 OF 10 - ENDOF 32 OF 100 + ENDOF 5A
OF 100 - ENDOF 33 OF 1000 + ENDOF 45
OF 1000 - ENDOF  DROP 1+ DUP ENDCASE
 REPEAT DROP DROP CR DECIMAL ; DECIMAL
```

TRS-80

AMÉLIORER LE TRI

Le TRS-80 Modèle III est équipé d'un logiciel intégré d'une puissance surprenante : il trie un tableau de 50 éléments en une seconde alors qu'il en faut une trentaine avec un tri de Hoare. (D'autres modèles l'ont aussi, tout dépend de leur système d'exploitation.)

Mais ce logiciel ne trie que les chaînes et l'utilisateur n'a pas accès au programme. L'édition triée d'un fichier sur disque est donc interdite par ce tri. Chaque argument doit en effet être suivi du numéro de sa fiche pour pouvoir récupérer celle-ci après le tri.

Il existe cependant un moyen simple d'utiliser ce « super tri » : ajouter, à chaque argument, l'élément chaîne du numéro de la fiche, au moment de la création du tableau à trier. Par exemple, si un fichier de noms N$ doit être trié alphabétiquement, on créera le tableau T$ ainsi :

```
10 FOR I=1 TO LOF(1)
20 GET 1,I
30 T$(I)=N$+STR$(I) 'si le fichier commence à 1
40 NEXT
```

Le tableau ainsi créé, on le trie par :

```
50 CMD "O", LOF(1),T$(1)
```

LOF(1) représente le nombre d'éléments de T$, et T$(1) indique le premier du tableau à trier. Il est alors utilisé en récupérant pour chaque élément le numéro de la fiche qu'on y a précédemment accroché :

```
60 FOR I=1 TO LOF(1)
70 GET 1,VAL(MID$(T$(I),L+1)) 'L est la longeur du secteur Field de N$
80 PRINT N$;A$;B$; etc. 'secteurs Field
90 NEXT:GOTO...
```

La récupération du numéro de fiche se fait par un MID$ commençant au premier caractère après la fin du secteur Field de N$, et allant jusqu'à la fin de l'élément du tableau. Ceci rend impossible toute emprise sur un caractère de N$ qui viendrait annuler le VAL.

Il ne faut pas oublier de prévoir DIM T$(LOF(1)) et CLEAR n, où n couvre le nombre total de caractères, par exemple LOF(1)*(L+3). Pour un tri dégressif, il n'y a qu'à utiliser le tableau trié à l'envers par :

```
FOR I=LOF(1) TO 1 STEP -1
```

Il ne vous reste plus qu'à essayer, avec un vrai fichier.

Francis GOUDARD

PB-700

CALCUL D'INTÉGRALES

Si vous êtes comme moi élève de terminale, vous savez que dans très peu de temps, nous aurons à "plancher" sur une étude de fonction (logarithme, exponentielle,...). Et que, très vite, on aboutira au calcul de son intégrale.

Mon programme ne fera pas le travail à votre place, mais il vous permettra de vérifier votre résultat. En effet, il ne démontre rien, il calcule !

Après avoir rentré le programme dans votre PB-700, faites RUN. Il vous demande alors les deux bornes d'intégration (les rentrer une par une). Environ cinq secondes après, il affiche une magnifique intégrale. Seulement voilà, comme vous avez pu le remarquer, la ligne 30 contient la fonction choisie pour *mon* exemple : Y(X)=1/X. Pour une autre fonction, il suffira de faire EDIT 30 et de remplacer 1/X (à droite du signe "=") par la nouvelle fonction, pour obtenir l'intégrale correspondante.

```
  5 REM INTEGRALES-CHRSTOPHE MAURIN
 10 CLS :ANGLE 1
 15 INPUT "BORNES D INTEGRATION";A1,A2
:DIM Y(A2*10)
 20 W=(A2-A1)/8
 25 FOR X=A1 TO A2 STEP W
 30 Y(X)=1/X
 35 IF X=A1 THEN I=1
 40 IF X=A2 THEN I=1
 45 Y(X)=I*Y(X)
 50 IF I=4 THEN I=2 ELSE I=4
 55 S=S+Y(X)
 60 NEXT X:Z=W*S/3
 65 CLS
 70 PRINT CHR$(156);A2,CHR$(150);"=";U
SING"###.####";Z;CHR$(159);USING"";A1
 75 IF INKEY$="" THEN 75 ELSE CLEAR
```

Calcul d'intrégrales
Programme pour PB-700
Auteur Christophe Maurin
Copyright *LIST* et l'auteur

Une remarque très importante : les bornes d'intégration doivent être rentrées dans un ordre croissant. Par exemple, pour le calcul de $\int_1^4 1/t\ dt$, il faut rentrer 1 d'abord, puis 4. De même, si les bornes avaient été inversées (il suffirait, pour avoir le bon résultat, d'en prendre le signe opposé).

Enfin, si vous souhaitez une plus grande précision, remplacez le "8" de la ligne 20 par une plus grande valeur, ou bien modifiez le format d'impression de la ligne 70.

Christophe MAURIN

LES JEUX ET CASSE-TÊTE INFORMATIQUES

de Thierry CHAMORET

LES jeux et casse-tête qui vous sont proposés dans cette rubrique ont plusieurs aspects. Tout d'abord, ils peuvent être pris sous l'angle ludique, c'est-à-dire qu'il s'agit de jeux, de petits problèmes plus ou moins faciles à résoudre. Ils ont également un aspect pratique. Ils permettent en effet à chacun d'exercer son agilité logique. Et il n'est pas nécessaire, pour trouver la solution, d'avoir un ordinateur sous la main...

23

Un programme anticonvivial

Un programmeur, à qui l'on a demandé d'écrire un logiciel permettant d'effectuer la moyenne des valeurs comprises entre 0 et 20 a écrit le programme suivant :

```
10 DIM Valeur (100)
20 INPUT Nombre
30 FOR I = 1 TO Nombre
40    INPUT Valeur (I)
50 NEXT I
60 Total = 0
70 FOR I = 1 TO Nombre
80    IF (Valeur (I) < 0) OR (Valeur (I)
      > 20) THEN GOTO 130
90    Total = Total + Valeur (I)
100 NEXT I
110 PRINT Total/Nombre
120 STOP
130 PRINT « Valeur erronée »
140 STOP
150 END
```

Sans qu'il soit besoin de mettre en œuvre ce programme, nous constatons immédiatement qu'il n'est pas très convivial, pas très agréable à utiliser. Quelles remarques ne manquerait pas de faire un utilisateur de ce programme ? Vous pourrez en trouver au moins dix, d'autant que vous n'aurez aucun regret à critiquer un programme que vous n'avez pas écrit.

24

La parité des mots

Lors des transferts d'informations, que ce soit à l'intérieur même de l'ordinateur ou en direction des périphériques, les erreurs sont redoutables. Différentes méthodes ont été élaborées pour les détecter. La plus simple d'entre elles consiste à ajouter un bit de parité à chaque octet transmis.

Prenons par exemple un octet représenté par la suite de ses huit bits : 0 1 0 0 0 1 1 0.

Il a pour valeur décimale 70 ($2^6 + 2^2 + 2^1 = 64 + 4 + 2$). Il possède un nombre impair de bits à 1. Pour rétablir la parité, on ajoute un neuvième bit que l'on met à 1. Ce neuvième bit sera appelé *bit de parité :* 0 1 0 0 0 1 1 0 .1

* *Les solutions des jeux proposés ici paraîtront dans le prochain numéro de* LIST

*_ J'AI UNE INFORMATION POUR VOUS!
— SEULEMENT UN PETIT BIT! (OU: UN PETIT PEU, "BIT" EN ANGLAIS SIGNIFIE "UN BRIN").

Bien entendu, si le nombre de bits à 1 est pair, le bit de parité sera égal à 0.

Ce bit n'intervient pas dans la signification de l'octet, mais constitue un moyen de contrôle. Ainsi, avec cette technique, si une erreur de transmission sur un bit survient, elle est immédiatement détectée.

Le problème est de calculer ce bit de parité. Sur les ordinateurs possédant un dispositif de contrôle de parité, un circuit spécialisé se charge en général de ce calcul. Sinon, il est nécessaire d'écrire une routine en Assembleur chargée de compter le nombre de bits à 1. Si ce nombre est pair, le bit de parité est égal à 0. Dans le cas contraire, il est égal à 1.

Dans un langage évolué (Basic, par exemple), nous n'avons pas le moyen de tester si un bit est égal à 0 ou à 1 dans un octet. Mais vous trouverez certainement le moyen de vous en sortir avec les opérations classiques de votre ordinateur ou de votre calculatrice programmable.

25

En toute logique

A et B sont deux variables logiques pouvant chacune prendre l'une des deux valeurs *faux* (0) ou *vrai* (1). Simplifiez les équations logiques suivantes, bien entendu sans en calculer les valeurs :

1. non (non B ou A)
2. (A et B) ou (A et (non B))
3. (A ou B) et (A et B)
4. non ((A ou B) et B)
5. non ((non A) et (non B))
6. ((non A) et (non B)) ou B
7. ((non A) et B) ou (A et B)
8. ((non A) et (non B)) ou ((non A) et B)
9. ((non A) et (non B)) ou (A et (non B)) ou (A et B)
10. A et (A ou B) et (non B) et (A ou (non B))

26

Lequel des trois

Avez-vous une idée de l'ordre de grandeur de certaines valeurs relatives à l'informatique ? Les cinq questions ci-dessous vous proposent trois réponses possibles, dont l'une est exacte. A vous de la déterminer.

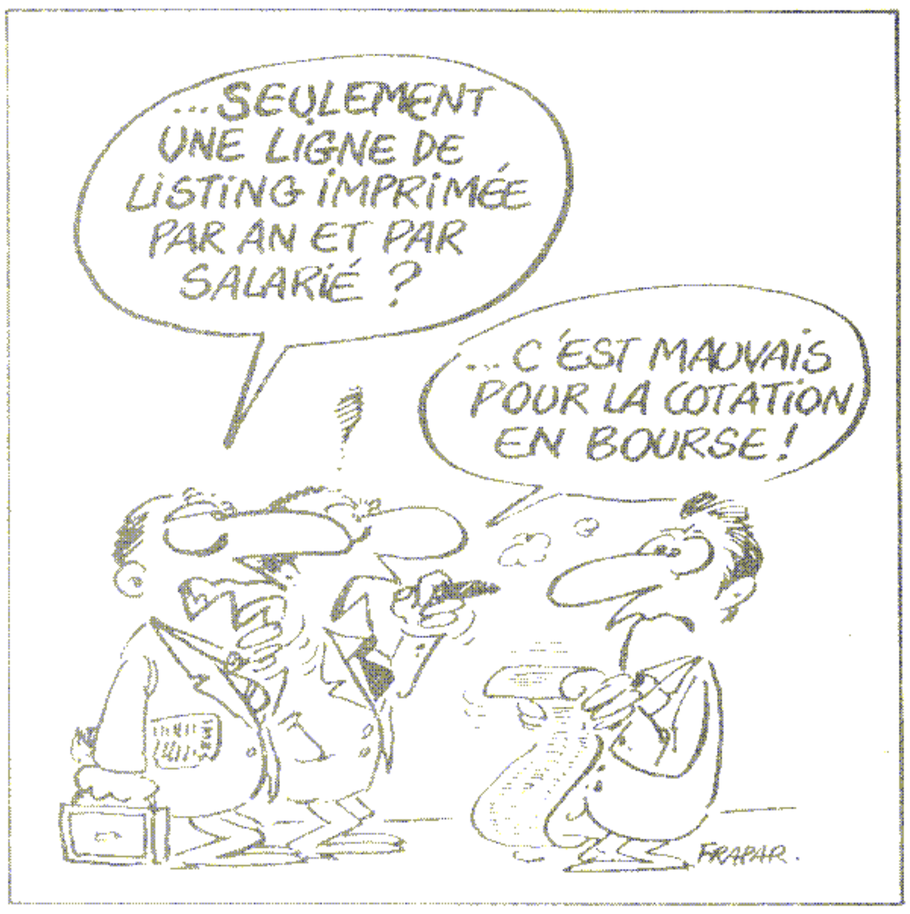
1. Quelle est généralement la vitesse de rotation d'un disque souple ?
a - 30 tours par minute
b - 300 tours par minute
c - 3000 tours par minute

2. Mark 1 a été un des premiers ordinateurs construits dans le monde. Quelle était sa vitesse de calcul ?
a - 1200 multiplications à la seconde
b - 12 multiplications à la seconde
c - 12 multiplications à la minute

3. Quel était, en 1981, le nombre moyen de lignes de listes imprimées dans les entreprises utilisant un ordinateur ?
a - 555 lignes par an et par salarié
b - 5550 lignes par an et par salarié
c - 55500 lignes par an et par salarié

4. Quelle est la productivité moyenne d'un programmeur professionnel travaillant sur un grand projet ?
a - 5 instructions par jour
b - 50 instructions par jour
c - 500 instructions par jour

5. En 1987, il y aura approximativement 76 millions de micro-ordinateurs installés dans le monde. Combien y en avait-il dix ans plus tôt, en 1977 ?
a - 5000 micro-ordinateurs
b - 50000 micro-ordinateurs
c - 500000 micro-ordinateurs



27

Recherche du DU

Les langages de programmation ne disposent pas toujours de la fonction appelée quelquefois POS, INSTR, ou STRPOS, dont l'objet est de retourner la position d'une sous-chaîne dans une chaîne. Dans ce cas, il est nécessaire d'écrire un sous-programme réalisant un tel traitement. Cela semble simple au premier abord, mais cette opération est en fait plus complexe qu'elle ne paraît. Ainsi, pour rechercher la séquence « DU » dans une chaîne, un programmeur a écrit le petit programme suivant :

```
10 Pos = 0
21 I = 0
30 I = I + 1
40 IF I > = LEN (Chaine) THEN
   GOTO 90
50 IF Chaine (I) < > "D" OR Chaine
   (I) < > "d" THEN GOTO 30
60 I = I + 1
70 IF Chaine (I) < > "U" OR Chaine
   (I) < > "u" THEN GOTO 30
80 Pos = I
90 PRINT "Position :" ; Pos
```

Dans ce petit programme, la variable CHAINE représente la chaîne dans laquelle il faut rechercher la première occurrence de « DU ». Le langage utilisé permet ici d'indexer une chaîne comme un tableau de caractères, l'indice indiquant la position du symbole de la chaîne que l'on désire obtenir.

Ce petit programme paraît simple. Et pourtant il donne un résultat faux dans certains cas. Arriverez-vous à trouver dans quelles conditions il est erroné, et comment on peut le corriger ?

SOLUTIONS DU NUMÉRO PRÉCÉDENT

Les solutions présentées ici répondent aux problèmes posés dans le précédent numéro de LIST. Ce ne sont pas forcément les meilleures !

19

Fonctions logiques

Les fonctions logiques comme le ou exclusif (ox), l'équivalence (equ), l'implication (imp) et la réciprocité (rpc) sont rarement implantées dans les langages de programmation courants. Ils peuvent être facilement remplacés par des expressions analogues :

- A < > B pour A ox B
- A = B pour A equ B
- A < = B pour A imp B
- A > = B pour A rpc B

Bien entendu, il existe d'autres solutions qui utilisent en particulier les opérateurs AND, OR et NOT.

20

Interdit de sauter

1. La seconde version de programme fournit le même résultat que la première seulement dans le cas où le tableau Arrete ne contient qu'une seule valeur égale à 0. Dans les autres cas, la seconde version indique, par l'intermédiaire de Prear, l'indice du dernier (et non du premier) élément égal à 0. Elle peut être facilement corrigée en inversant la boucle qui doit se faire de 24 à 1 au lieu de 1 à 24 :

```
POUR I : = 24 JUSQU__A 1 PAS = − 1
    SI Arrete (I) = 0 ALORS Prear : = I
FIN__POUR
```

2. Cette dernière version n'est toutefois pas satisfaisante car elle explore la totalité du tableau Arrete. Comme seul l'indice du premier élément nul nous intéresse, nous pouvons l'écrire ainsi :

```
Prear : = 1
TANT__QUE Arrete (Prear) < > 0 ET Prear < 24
    Prear : = Prear + 1
FIN__TANT__QUE
```

21

Max et les tableaux

1. Pour supprimer le GOTO présent après THEN, on inverse le sens de l'inégalité du test. Ce qui donne le programme.

```
10 M = 0
20 FOR I = 1 TO 10
30    IF T (I) > M THEN M = T (I)
40 NEXT I
50 PRINT M
```

2. Le programme affiche toutefois un résultat faux lorsque toutes les valeurs du tableau T sont négatives. La valeur affichée est alors égale à 0.

3. La correction du programme s'effectue en initialisant la variable M non pas à 1, mais à T (1). Ainsi, la boucle en ligne 20, peut commencer à 2 au lieu de 1.

```
10 M = T (1)
20 FOR I = 2 TO 10
30    IF T (I) > M THEN M = T (I)
40 NEXT I
50 PRINT M
```

22

Les clones informatiques

Le programme ci-dessous est un exemple de clone informatique : son exécution entraîne sa reproduction exacte, et avec quelle élégance !

Bien entendu, quelques modifications devront être apportées selon les Basic. La seule particularité de celui qui est présenté ici est peut-être la ligne 100 : elle dimensionne un tableau de 14 chaînes pouvant mémoriser jusqu'à 80 caractères. Un tel programme pourrait être écrit dans d'autres langages. Nous attendons toutes vos suggestions en Pascal, Fortran, Forth, Logo ou tout autre « dialecte ».

```
100 DIM A$(14)[80]
110 A$(1)="DIM A$(14)[80]"
120 A$(2)="PRINT ' 100 ';A$(1)"
130 A$(3)="FOR I=1 TO 14"
140 A$(4)="PRINT 100+10*I;'A$(';VAL$(I);')='CHR$(34);A$(I);CHR$(34)"
150 A$(5)="NEXT I"
160 A$(6)="FOR I=1 TO 14"
170 A$(7)="FOR J=1 TO LEN(A$(I))"
180 A$(8)="IF A$(I)[J;1]=CHR$(39) THEN A$(I)[J;1]=CHR$(34)"
190 A$(9)="NEXT J"
200 A$(10)="NEXT I"
210 A$(11)="FOR I=2 TO 14"
220 A$(12)="PRINT 230+10*I;A$(I)"
230 A$(13)="NEXT I"
240 A$(14)="STOP"
250 PRINT " 100 ";A$(1)
260 FOR I=1 TO 14
270 PRINT 100+10*I;"A$(";VAL$(I);")=";CHR$(34);A$(I);CHR$(34)
280 NEXT I
290 FOR I=1 TO 14
300 FOR J=1 TO LEN(A$(I))
310 IF A$(I)[J;1]=CHR$(39) THEN A$(I)[J;1]=CHR$(34)
320 NEXT J
330 NEXT I
340 FOR I=2 TO 14
350 PRINT 230+10*I;A$(I)
360 NEXT I
370 STOP
```

Liste du programme clone, ou résultat de son exécution. Qui sait ?

Imprimerie SIMA, rue des Epinettes, ZI de Torcy, 77200 Torcy. Printed in France. Directeur de la publication : Jean-Luc Verhoye.
Dépôt légal mars 1985. Diffusion NMPP. Numéro de commission paritaire : 63617