PEINTRE ET MUSICIEN sur AMSTRAD

Daniel-Jean DAVID

Edimicro

PEINTRE ET MUSICIEN SUR AMSTRAD

CPC 464
CPC 664
CPC 6128

Couverture : Thierry Duval
Préparation éditoriale : Bénédicte Dieras

Tome 1. ISBN : 2-904457-63-1

EDIMICRO

DÉPARTEMENT ÉDITIONS DE F.D.S. SARL

121-127, avenue d'Italie, 75013 Paris

PEINTRE ET MUSICIEN SUR AMSTRAD

CPC 464
CPC 664
CPC 6128

Daniel-Jean DAVID

Edimicro

Daniel-Jean DAVID enseigne l'informatique de gestion à l'Université de Paris-1 Panthéon Sorbonne.

Par ailleurs, il enseigne l'utilisation des microprocesseurs à l'E.N.S.A.M.

Ses sujets de recherche vont de l'informatique graphique aux techniques d'interface des microprocesseurs et des systèmes multi-processeurs.

Spécialiste des microprocesseurs, il donne régulièrement de nombreux séminaires, tant pour débutants qu'au niveau le plus élevé, et il a écrit de nombreux articles dans les revues spécialisées.

SOMMAIRE

PRESENTATION

Peintre et musicien, c'est ce que vous deviendrez grâce à votre AMSTRAD après avoir lu ce livre. Ce livre suppose que vous avez acquis une connaissance préalable du langage Basic. Pour cela nous vous recommandons de lire "La Découverte de l'AMSTRAD".

A vrai dire, le sujet de ce livre est un approfondissement de l'étude du BASIC de l'AMSTRAD, insistant plus particulièrement sur les extensions que présente le Basic AMSTRAD par rapport à d'autres: des instructions de structuration et d'aide à la mise au point des programmes plus commodes, des instructions graphiques faciles à utiliser, des instructions sonores synthétiques. Sa caractéristique essentielle est de ne faire appel qu'à BASIC, même pour les traitements graphiques ou sonores les plus élaborés; on peut ainsi mesurer les excellentes capacités du Basic AMSTRAD dans ces domaines.

Le livre est valable pour les trois modèles AMSTRAD CPC 464, 664 et 6128; les restrictions (peu nombreuses) du 464 sont signalées et, si possible, on donne une méthode pour les contourner. Le plan du livre est alors tout tracé : le premier chapitre donne un rappel succinct de Basic (qui s'appuie fortement sur "La Découverte de l'AMSTRAD") et examine les extensions apportées par l'AMSTRAD dans les instructions de structuration et les aides à la mise au point. Le deuxième chapitre introduit les différents modes d'affichage de l'AMSTRAD. Le chapitre 3 examine les graphiques haute résolution les plus simples et donne leurs applications en gestion : diagrammes en bâtons et camemberts.

Le chapitre 4 utilise les graphiques haute résolution sous leur forme la plus élaborée avec dessins en 3 dimensions et algorithmes de parties cachées; ce chapitre se termine par un programme de définition de formes qu'on peut faire tourner à volonté dans toutes les directions et visualiser avec faces cachées. Vous trouverez aussi un programme très esthétique de visualisation de surfaces. Le Basic AMSTRAD ne possède pas d'instructions spéciales pour définir et manipuler des lutins (objets graphiques déplaçables); le chapitre 5 esquisse des méthodes permettant de les simuler. Enfin, on peut dire que le chapitre 6 *fera du bruit*, ou plutôt en fera faire à votre AMSTRAD, puisqu'il étudie les possibilités sonores et musicales! Là aussi, elles sont faciles à mettre en oeuvre grâce aux instructions évoluées offertes par l'AMSTRAD.

Le livre est complété par des annexes qui regroupent les informations de référence indispensables, et donnent la solution des exercices (il y en a plus de 60).

Il va sans dire que tous les listings donnés ici sont faits sur l'ordinateur après test du programme. Pour une exécution correcte, vous devez les taper avec soin.

CHAPITRE 1

BASIC

1.1 Courte révision des instructions Basic fondamentales

Cette révision sera la plus brève possible. En cas de problème, nous vous conseillons de consulter les chapitres 1 à 5 de "La Découverte de l'AMSTRAD".

Les instructions de Basic peuvent être réparties en un certain nombre de catégories. La distinction la plus importante nous semble être entre instructions **séquentielles**, c'est-à-dire instructions telles que, une fois achevées, on passe immédiatement à la suivante, et instructions **de structuration** qui permettent d'arranger l'ordre d'exécution des différentes séquences, de donner -autrement dit- sa structure au programme.

Instructions séquentielles

Les trois principales instructions séquentielles correspondent aux trois principales opérations qu'on trouve dans tout traitement informatique.

A l'**acquisition de données** correspond l'instruction **INPUT** de la

forme :

INPUT "texte";liste de variables

qui, après avoir affiché le texte spécifié à l'écran, attend qu'on fournisse au clavier des valeurs qui seront entrées dans les variables indiquées.

Ex. INPUT "RAYON,HAUTEUR";R,H

Au **calcul**, élaboration d'informations nouvelles à partir de données de départ, correspond l'**instruction arithmétique** de la forme :

variable = expression arithmétique

qui calcule l'expression arithmétique à droite du signe = et range le résultat dans la variable indiquée à gauche.

```
Ex. A=2*EXP(-T^2/2)
    C$="BONJOUR "+X$
```

Les exemples ci-dessus mettent en évidence la possibilité pour Basic de manipuler des données **numériques** (nombres) ou **alpha-numériques** (chaînes de caractères ou textes). Les variables chaînes ont leur nom terminé par un $. Il existe de nombreuses fonctions sur les chaînes de caractères. Basic permet aussi de manipuler des collections de données rassemblées dans des variables tableaux ou variables indicées.

L'opération qui est peut-être la plus importante, en tous cas celle sans laquelle on n'achèterait pas d'ordinateurs est la **sortie des résultats**. Il lui correspond l'instruction **PRINT** de la forme :

PRINT élément ; élément , élément ...

où les éléments peuvent être des constantes, des variables ou des expressions numériques ou alphanumériques. Lorsque le séparateur est ';' les éléments sont imprimés jointifs, il y a une tabulation si l'on emploie ','.

Ex. PRINT "RESULTAT = ";A

Instructions de structuration

Le Basic élémentaire comporte trois instructions de structuration (ou rupture de séquence) :

GOTO numéro de ligne qui fait aller inconditionnellement à la ligne indiquée

Ex. GOTO 50

IF condition **THEN** instructions qui fait exécuter les instructions indiquées seulement si la condition spécifiée est vraie

Ex. IF A>B THEN PRINT "SUPERIEUR"

et le couple FOR variable=départ TO fin STEP pas ... NEXT qui fait répéter les instructions placées entre FOR et NEXT pour toutes les valeurs de la variable comprises entre début et fin, avec un intervalle d'incrémentation égal à 'pas'.

Ex. FOR I=1 TO N STEP 2

Un autre couple d'instructions de structuration très important permet de définir des sous-programmes. L'instruction d'appel est **GOSUB** numéro de ligne : elle fait sauter au début du sous-programme à la ligne indiquée. Le sous-programme se termine par **RETURN** qui fait revenir juste après l'appel.

Autres instructions

Basic a bien entendu beaucoup d'autres instructions. Certaines manipulent des données, d'autres agissent sur l'affichage, ou créent des effets graphiques ou sonores (c'est cette catégorie qui est le principal sujet de ce livre). Une autre catégorie importante est celle des instructions de manipulation de fichiers. Enfin, il y a des instructions

qui font le lien entre Basic et le langage machine : PEEK, POKE, CALL, MEMORY, DI, EI, IN, OUT et WAIT.

Les commandes

A côté des instructions, Basic renferme des **commandes** qui aident à l'écriture, à la mise au point et à l'exécution des programmes. Elles s'utilisent généralement en mode direct.

La plus importante est **RUN** qui fait exécuter le programme. Presqu'aussi importante est **LIST** qui le fait lister.

Une catégorie de commandes fondamentales est celle de **SAVE** et **LOAD** qui permettent de sauvegarder puis de retrouver un programme sur cassette ou disquette.

1.2 Instructions élaborées du Basic AMSTRAD

Le Basic AMSTRAD est un des plus perfectionnés qui soient: il est à peu près dépourvu de lacunes. Outre les graphiques et les sons, les domaines dans lesquels il apporte les instructions - ou les formes - élaborées qui faisaient cruellement défaut dans les versions élémentaires, sont les manipulations disques, les aides à l'écriture et à la mise au point des programmes et les instructions de structuration.

Instructions disques (ou cassettes)

Nous passons assez rapidement sur ce domaine qui n'est pas essentiel pour nos graphiques : les seules instructions qui nous intéressent sont celles qui permettent de sauver un programme ou un dessin et quelques instructions annexes. Nous ne parlons pas des instructions de gestion de fichiers : ceci est la matière d'un autre ouvrage.

LOAD

LOAD "nom" charge en mémoire le programme "nom" à partir du

périphérique par défaut, disque ou cassette. Vous pouvez imposer le périphérique utilisé par |TAPE ou |DISC. Le nom peut être spécifié sous forme de variable chaîne de caractères (précédée de @ sur 464) : LOAD NOM$ ou LOAD @NOM$

Pour un module en langage machine ou une table en mémoire, il peut y avoir un second paramètre qui spécifie l'adresse de chargement lorsqu'elle est différente de celle utilisée lors de la sauvegarde.

MERGE est équivalent à LOAD, mais n'efface pas le programme déjà en mémoire : le nouveau programme se rajoute au premier, ce qui permet de constituer des bibliothèques de sous-programmes.

SAVE

Cette instruction sauvegarde le programme sur disque ou cassette avec le nom spécifié. Ex. SAVE "ZOZO"

La forme ci-dessus est la plus simple : elle correspond à la simple sauvegarde d'un programme. Les formes SAVE "nom",P et SAVE "nom",A sauvent le programme respectivement sous forme "protégée" et sous forme ASCII (=texte, seule forme utilisable en vue de MERGE).

La dernière forme permet de sauvegarder le contenu d'une zone mémoire. On dit que c'est une sauvegarde binaire. Les applications sont nombreuses : on peut ainsi sauver un programme en langage machine, un tableau de données etc... Dans ce livre nous l'utiliserons plus spécialement lorsque la zone mémoire sauvegardée est la mémoire d'écran, ce qui fournit la sauvegarde d'une image.

L'instruction s'écrit : SAVE "nom",B,ad,no

où 'nom' est le nom du fichier à créer, ad est l'adresse de départ de la zone à sauver, no est le nombre d'octets à sauver. Ces paramètres sont commodément spécifiés en hexadécimal (&xxxx).

Ex : pour sauver la mémoire d'écran, on fait : SAVE "ECRAN",B,&C000,&4000

Exercice 1.1

Sauver la zone mémoire qui va des adresses 1024 à 2047.

CAT ou |DIR

Cette instruction fait afficher la table des matières de la cassette ou disquette.

La commande AMSDOS |DIR ne porte que sur disquette mais elle a l'avantage de permettre de spécifier un groupe de fichiers à l'aide de caractères joker :

Ex. |DIR "P*" ne fait lister que les fichiers dont le nom commence par un P.

Instructions de structuration

Cette catégorie reçoit sur AMSTRAD de substantielles améliorations par rapport au Basic élémentaire. On a enfin la structure alternative complète avec IF ... THEN ... ELSE

et WHILE ... WEND procure une structure de boucle "jusqu'à" ou "tant que".

IF ... THEN ... ELSE ...

Une alternative complète s'écrit :

IF condition THEN instructions 1 ELSE instructions 2

S'il y a plusieurs instructions dans les suites instructions 1 ou instructions 2, elles doivent être séparées par des ':' Si la condition spécifiée est vraie, on exécute les instructions 1 puis on passe à la ligne suivante; si elle est fausse, on exécute les instructions 2 puis on passe à la ligne suivante (sous réserve qu'il n'y ait pas de GOTO dans les instructions).

Ex. 100 IF A>B THEN GOSUB 1000 ELSE GOSUB 2000

Un problème qui se pose est que tout l'ensemble doit tenir dans une ligne Basic (255 caractères). L'utilisation de GOSUB apporte une solution.

Exercice 1.2

Dans l'exemple :

```
10 i01
20 IF c THEN i11 : i12 else i21 : i22
30 i31
```

quelles instructions sont exécutées si la condition est vraie ? Si elle est fausse ? Si i12 est un GOTO 1000 ?

WHILE...WEND

Ces instructions font répéter toutes les instructions qui se trouvent entre elles tant qu'une condition de continuation est vraie :

```
WHILE condition
... instructions
WEND
```

Si la condition spécifiée est toujours vraie (WHILE 1 ... WEND), on boucle indéfiniment.

Ex. A$="" : WHILE A$="" : A$=INKEY$: WEND attend qu'on appuie sur une touche quelconque au clavier.

Exercice 1.3

Attendre qu'on appuie sur la touche 'A' au clavier.

On peut sortir de la boucle à partir de l'intérieur à l'aide d'une

instruction IF :

```
Ex.  100 I=0 : WHILE I<=NC
     110 I=I+1
     120 IF NOM$(I)="DUPONT" THEN 140
     130 WEND
     140 ...
```

recherche le client DUPONT parmi le tableau des noms des NC clients.

Exercice 1.4

Pourrait-on supprimer la clause WHILE à la ligne 100 ? Pourrait-on supprimer la ligne 120 et écrire :

```
100 I=0: WHILE NOM$(I)<>"DUPONT"
```

L'exercice qui précède pose le problème de la possibilité d'évaluer la condition, notamment à l'entrée de la boucle.

Un autre problème est celui-ci : il faut évidemment que la condition d'arrêt puisse être atteinte, sinon on bouclerait indéfiniment. Il faut en particulier que les calculs effectués dans la boucle fassent évoluer la valeur de la condition.

Instructions d'aide à la programmation

Le Basic AMSTRAD apporte de nouvelles instructions qui aident considérablement à programmer. On peut distinguer des instructions d'aide à l'écriture des programmes, des instructions d'aide à la recherche des erreurs et enfin des instructions diverses qui facilitent la programmation.

Aide à l'écriture

Cette catégorie comporte les instructions de numérotation automatique, de renumérotation et de suppression d'un bloc

d'instructions qui manquaient aux précédentes versions de Basic. Il s'y ajoute l'instruction de programmation des touches de fonction. Ces instructions s'utilisent presque exclusivement en mode direct.

AUTO

Cette commande lance la numérotation automatique. Elle est de la forme :

AUTO n1,n2

où n1 est le numéro de départ et n2 est l'intervalle entre numéros. Les deux paramètres ont pour valeur par défaut 10.

DELETE

Cette commande supprime une série d'instructions. Les numéros de lignes à supprimer sont spécifiés comme dans LIST :

DELETE n1-n2

DELETE tout court est dangereux car cela revient à supprimer la totalité du programme.

RENUMBER

De la forme :

RENUMBER ad,nd,i

cette commande renumérote le programme à partir de la ligne ad. Les nouveaux numéros commencent à nd (qui remplace ad) et l'intervalle est i.

Cette commande est assez longue à exécuter.

KEY

Cette instruction permet de programmer n'importe quelle touche du clavier, mais elle est surtout utilisée pour les touches du pavé numérique : on dit que ce sont les touches de fonction. Elles ont un f sur 664 ou 6128.

La seule combinaison préprogrammée à part les chiffres est *CTRL RETURN*, qui vaut RUN"*RETURN*.

Pour changer le rôle d'une de ces touches, on écrit :

KEY numéro logique,chaîne

où chaîne représente une expression chaîne de caractères dont la valeur apparaîtra dès qu'on appuiera sur la touche f'numéro'. Le plus souvent, l'expression se réduit à un texte entre guillemets, par exemple KEY 128,"BONJOUR", mais si l'on veut incorporer à la chaîne des caractères comme *ESC*, *RETURN/ENTER* ou des guillemets, il faut utiliser respectivement CHR$(27), CHR$(13) ou CHR$(34).

Ex. KEY 129,"LIST"+CHR$(13) permet d'obtenir un listing dès qu'on appuie sur la touche f1.

Les numéros logiques vont de 128 à 159. Pour les touches chiffres du pavé numérique, on a numéro=128+chiffre.

KEY DEF

Pour pouvoir programmer n'importe quelle touche, il faut utiliser KEY DEF qui associe un numéro logique au numéro physique d'une touche. Le numéro physique d'une touche est immuablement lié à l'emplacement physique de la touche sur le clavier. Le manuel constructeur fournit un schéma du clavier avec les numéros physiques; ce schéma est rappelé sur l'unité de disques de certains modèles.

Le numéro logique est le plus souvent par défaut le code ASCII de la touche. En fait, il y a trois numéros logiques selon que la touche est

appuyée seule, avec *SHIFT* ou avec *CTRL*. L'instruction s'écrit :

KEY DEF np,r,nl,nls,nlc

où np (de 1 à 79) est le numéro physique, r indique si la touche aura (r=1) ou non (r=0) la répétition automatique, nl est le numéro logique attribué à la touche seule. Les paramètres suivants sont facultatifs et ils attribuent un numéro logique à la touche utilisée respectivement avec *SHIFT* et avec *CTRL*. Rappelons que l'espace entre KEY et DEF est obligatoire.

Une application évidente de KEY DEF est d'échanger les rôles de certaines touches : cela permettrait par exemple de simuler un clavier AZERTY. Ainsi, pour mettre le A à la place de la touche Q (numéro physique 67), il suffit de :

KEY DEF 67,1,97,65,1

(97 est le code ASCII de a, 65 est celui de A et 1 correspond à *CTRL* A). Attention, pour obtenir un clavier AZERTY complet, il faut faire les échanges complets : avec la seule instruction ci-dessus, vous avez deux touches A sur votre clavier. Pour les lettres accentuées, il faut en outre créer leur dessin, ce qui se fait avec SYMBOL comme on le verra plus loin.

Traitements d'erreurs

Les instructions TRON et TROFF aident au dépistage des erreurs tandis que le couple ON ERROR ... /RESUME permet de laisser le contrôle au programme lorsqu'une erreur s'est produite.

TRON/TROFF

L'instruction TRON active le mode trace, l'instruction TROFF le désactive. En mode trace, on imprime les numéros de ligne à mesure qu'on passe dessus. C'est très efficace pour déceler une boucle indéfinie ou un test qui ne se fait pas comme prévu.

ON ERROR GOTO

Si après exécution d'une instruction ON ERROR GOTO n , il se produit une erreur, le programme ne sera pas stoppé avec impression du message d'erreur approprié : le programme passera à l'instruction de numéro n spécifié.

A cette instruction, vous démarrez votre propre routine d'erreur. Vous aurez probablement besoin de connaître le numéro de la ligne où l'erreur s'est produite : il est contenu dans la variable système ERL; le numéro d'erreur (ex. 2 : syntaxe, 11 : division par 0 , etc., dont vous trouverez la liste complète dans le manuel constructeur) est, lui, dans la variable ERR. Vous pouvez le tester pour isoler un type d'erreur auquel vous réservez un traitement particulier.

ON ERROR GOTO 0 désactive votre traitement d'erreur; on revient donc à l'action habituelle du système : message et arrêt.

RESUME

Votre routine d'erreur doit se terminer par une instruction RESUME qui fait continuer l'exécution.

Avec RESUME tout court, on reprend à l'instruction qui avait causé l'erreur, c'est-à-dire qu'on essaie de la ré-exécuter; il faut bien sûr que votre traitement d'erreur ait corrigé la condition qui causait l'erreur.

Avec RESUME NEXT, on reprend à l'instruction qui suit l'instruction où l'erreur s'était produite. RESUME n fait reprendre à l'instruction numéro n.

Exercice 1.5

Isoler l'erreur "DATA exhausted" (ERR=4) : dans ce cas, on doit reprendre au début des données. Pour les autres erreurs, on veut le comportement habituel du système.

Exercice 1.6

Esquissez un moyen d'avoir les messages d'erreur en français.

Enfin, il existe une instruction, ERROR e qui crée artificiellement l'erreur no e, e pouvant aller jusqu'à 255 alors que les erreurs connues par Basic vont jusqu'à 32. Vous pouvez donc implanter votre propre gestion des erreurs particulières à votre traitement.

Améliorations diverses

Nous regroupons dans cette section soit diverses instructions nouvelles qui facilitent la tâche du programmeur, soit des extensions d'instructions qui existaient déjà dans les versions élémentaires de Basic.

RESTORE n

RESTORE peut être accompagné d'un numéro de ligne : on reprend aux DATA de cette ligne.

PRINT USING

Cette forme facilite beaucoup le cadrage et la présentation des impressions. On écrit :

PRINT USING format;liste

où 'liste' est la liste des données à imprimer; elles sont séparées par des virgules alors que c'est un ';' qui sépare du format. 'format' est une chaîne de caractères, spécifiée entre guillemets ou sous forme de variable alphanumérique.

La chaîne de format est une suite de spécifications qui se rapporteront à chacune des données à imprimer. S'il y a moins de spécifications que de données, le format est reparcouru (= réutilisé). Les spécifications sont séparées par des espaces (qui apparaissent à l'impression).

Maintenant, chaque spécification est une **image** de la donnée qui sera imprimée, c'est-à-dire que chaque caractère de la spécification correspond à un caractère qui sera imprimé.

Ex. Si A vaut 5, PRINT "###";A fera imprimer ■■5 (les ■ figurent des espaces) : il y a 3 caractères imprimés puisque le format a 3 caractères.

Pour une donnée numérique, les caractères possibles sont # + - . , $$, ** et, en fin de format, une suite de 4 ↑ (flèche en haut). Voici leurs significations.

représente un chiffre, avant ou après le point décimal.

+ et - représentent le signe. Ils forcent l'impression du signe même si le nombre est positif : + fait imprimer un '+', - fait imprimer un espace. Bien sûr, si le nombre est négatif, c'est un '-' qui est imprimé; on a le moyen de le faire imprimer en fin du nombre, ce qui est parfois utilisé en comptabilité. - ne peut figurer qu'en fin de format.

. représente le point décimal. Le nombre de '#'qui le suivent détermine le nombre de décimales qui seront imprimées (avec éventuellement un arrondi) tandis que le nombre de '#' qui la précèdent fixe le nombre de chiffres ou espaces de la partie entière. Le problème de l'alignement des impressions de nombres est résolu. Bien sûr, il ne peut y avoir qu'un '.' dans un format; s'il n'y en a pas, on impose l'impression sous forme d'entier.

, placée devant le point fait imprimer des virgules selon la méthode anglaise pour séparer les groupes de trois chiffres dans un grand nombre.

$$ en tête du format fait imprimer un '$' (signe monétaire, utile pour représenter un prix). En France, on peut employer FF au lieu de $$.

** en tête de format fait remplacer les espaces en tête du nombre par des *. Cela peut servir, par exemple, à protéger des chèques. ** peut se combiner avec le signe monétaire : **$ ou **F .

^^^^ placés en fin de format imposent l'impression du nombre en notation scientifique (de la forme x.yyE±ee).

Si un nombre est trop grand pour tenir dans le format demandé, on imprime un signe % et le nombre avec autant de caractères que nécessaire.

Placés avant ou après le format, les caractères autres que ceux vus ci-dessus sont purement et simplement recopiés dans l'impression.

Ex. Si A vaut 45.678, PRINT "* ###.## *"; A fait imprimer :
* 45.68 *

Exercice 1.7

Complètez le tableau :

format	nombre	impression
####	-1000	
* $$##.## *	5.948	
* $$##.## *	54.95	
* $$##.## *	545.95	
* $$##.## *	5435.95	
* $$##.## *	-5.95	
A= ### M	35.4	

Pour une donnée alphanumérique, les caractères de format autorisés sont & ! \ et espace .

& permet d'imprimer à cheval sur deux lignes. En effet, si compte tenu de la position actuelle sur la ligne et de la largeur de ce qu'il faut imprimer, Basic détermine que cela ne tiendra pas, il fait aller à la ligne avant de commencer l'impression. Le format & supprime cet effet.

! fait imprimer uniquement le premier caractère de la chaîne.

\ \ (avec n espaces) fait imprimer la chaîne dans une zone de n+2 caractères. Si la chaîne est plus longue, on imprime les n+2 premiers caractères; si elle est plus courte, on termine par des blancs.

DEC$

DEC$(n,f) fournit une représentation décimale de l'expression arithmétique n, suivant le format f.

F se présente comme un format de PRINT USING.

INSTR

Cette fonction de la forme INSTR(D,A$,B$) renvoie 0 si la chaîne B$ n'est pas comprise dans A$ et, si elle y est, on obtient la position dans A$ où commence la copie de B$. D est la position de départ de la recherche; si D n'est pas spécifié, on commence au début comme si D=1.

Ex. INSTR("BONJOUR","JOUR") vaut 4.

Les trois fonctions qui suivent sont surtout utiles en relation avec le langage machine.

HEX$

HEX$(X,N) donne la représentation hexadécimale du nombre X avec N chiffres hexadécimaux.

BIN$

BIN$(X,N) donne la représentation binaire du nombre X (qui doit

être compris entre -32768 et 65536) avec N chiffres binaires. Comme pour HEX$, N doit être <= 16.

@

@variable fournit l'adresse de la variable spécifiée. Sur 464, les arguments de certaines instructions doivent être fournis sous la forme @X lorsqu'ils ne sont pas des constantes.

Gestion d'horloges

L'AMSTRAD possède des "chronos" que l'on peut lancer et qui, après épuisement du temps vont créer une interruption. En attendant le délai, l'ordinateur peut effectuer d'autres opérations. Il y a quatre chronos numérotés de 0 à 3.

AFTER

Cette instruction est de la forme AFTER n,c GOSUB l . Après un délai de n cinquantièmes de seconde passés sur le chrono c, on appelle le sous-programme qui commence en ligne l. Entre temps l'AMSTRAD exécute les instructions qui suivent AFTER. Si c n'est pas spécifié, on prend le chrono 0 par défaut.

EVERY

EVERY n,c GOSUB l est très analogue à AFTER et les paramètres jouent le même rôle. La différence est que l'interruption est périodique : tous les n/50 seconde, le sous-programme l ... est exécuté, pendant que l'AMSTRAD exécute d'autres instructions.

Ex. (simplet mais démonstratif) :

On imprime BONJOUR de façon répétée et toutes les secondes, on

fait un bip :

```
10 EVERY 50 GOSUB 100
20 PRINT "BONJOUR" : GOTO 20
100 SOUND 1,142 : RETURN
```

(on voit la forme où on prend le chrono 0 par défaut)

Les appels cessent lorsque l'exécution du programme est terminée par STOP, END ou double *ESC*. Dans le cas de AFTER, si la fin a lieu avant épuisement du délai, le sous-programme événement n'aura jamais lieu.

CHAPITRE **2**

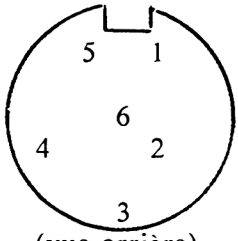
LES MODES GRAPHIQUES

2.1 Modes d'affichage de l'AMSTRAD

L'AMSTRAD possède trois modes d'affichage numérotés 0, 1 et 2, qui diffèrent par la résolution et les choix de couleurs possibles. Chaque mode est à la fois *graphique et texte* et les textes se mélangent facilement aux dessins. On peut aussi définir huit fenêtres sur l'écran. Le mode d'affichage est valable pour toutes les fenêtres. L'instruction de *changement de mode* vide toutes les fenêtres. Seule la fenêtre 0 peut avoir des dessins graphiques.

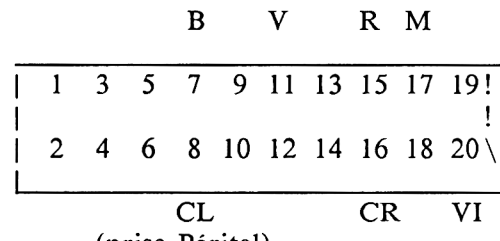
Les moniteurs et téléviseurs

L'AMSTRAD est livré avec un moniteur. Bien entendu, avec le moniteur noir et blanc (ou plutôt vert), les couleurs sont remplacées par des nuances d'intensité. Vous pouvez aussi utiliser un téléviseur, soit avec l'interface Péritel vendue par AMSTRAD, soit avec un montage fait par vous-même et dont nous rappelons le schéma ci-dessous. Dans ce dernier cas, vous devez alimenter l'AMSTRAD avec le moniteur, ce qui est encombrant, ou alors "bricoler" aussi une alimentation 5V (plus 12V pour le 664/6128).



(vue arrière)

1 Rouge
2 Vert
3 Bleu
4 Synchro
5 Masse
6 Luminance



(prise Péritel)

Câble Péritel

AMSTRAD		Péritel
1	---------	15
2	---------	11
3	---------	7
4	---------	16
5	---------	17
6	---------	20

(Les autres broches ne sont pas connectées. Selon le téléviseur, il peut être nécessaire de relier une pile 9 V avec le + à la broche 8 de la prise Péritel).

Possibilités des différents modes

Le mode qui a la plus basse résolution est le mode 0. Il permet d'afficher 25 lignes de 20 caractères de texte. En graphique, il offre une grille de 200 (vertical) par 160 (horizontal). En mode 1, on a 25 lignes de 40 caractères et une résolution graphique de 200 par 320 : dans chaque mode, la résolution horizontale est doublée par rapport au mode précédent; la résolution verticale reste la même. En mode 2, on a 25 lignes de 80 caractères et une résolution graphique de 200 sur 640 pixels. Rappelons qu'on appelle pixel (abréviation de picture element = élément d'image) le point élémentaire de la résolution graphique.

Le gain en résolution, lorsqu'on passe d'un mode à l'autre, est compensé par une perte au niveau du choix des couleurs : en mode 0, on peut avoir 16 couleurs différentes simultanément sur l'écran; en

mode 1, il n'y en a plus que 4 et en mode 2, il n'y a plus que deux couleurs possibles, la couleur de fond et celle de tracé.

Passage d'un mode à l'autre

L'instruction fondamentale pour changer de mode d'affichage est MODE. Elle est de la forme :

MODE m

où m est le numéro du mode d'affichage désiré 0, 1 ou 2.

Attention, toute exécution de l'instruction MODE vide l'écran, et même toutes les fenêtres de l'écran.

Exercice 2.1

Activez le mode moyenne résolution, 4 couleurs.

Nous pouvons maintenant passer les modes d'affichage en revue de façon plus détaillée.

2.2 Graphiques en forme de texte

Malgré le titre de cette section, il ne s'agit pas ici du problème du mélange texte-dessin qui sera abordé plus loin. Il s'agit de l'utilisation des caractères semi-graphiques qui permettent d'obtenir certains dessins avec les instructions classiques d'impression de texte. Bien sûr, nous ne disons rien ici de l'impression de textes ordinaires : elle relève du Basic le plus élémentaire. Le mode texte permet un peu de graphiques grâce au jeu de caractères semi-graphiques que présente l'AMSTRAD. Les caractères semi-graphiques s'obtiennent par des PRINT CHR$ (code). Les caractères semi-graphiques sont décrits avec leurs codes dans le manuel constructeur et dans "La découverte de l'AMSTRAD".

Exercice 2.2

"Dessine-moi un mouton".

Le dessin peut se faire par de simples PRINT des caractères choisis. Vous pouvez d'ailleurs enrichir le jeu de caractères semi-graphiques en redéfinissant vous-même les caractères dont vous avez besoin à l'aide de l'instruction SYMBOL. Nous n'insisterons pas plus car cette méthode a été complètement décrite dans "La découverte de l'AMSTRAD".

Il est plus judicieux d'entrer la suite de caractères voulus dans une variable chaîne de caractères (y compris les mouvements de curseur) : un simple PRINT de cette variable fera afficher le dessin. Maintenant, pour définir la variable, le mieux est de mettre la suite des codes dans une instruction DATA :

```
10 DATA 214,10,143,143,143,215,10,8,8,8,8,8,204,204,32,32,
   205,205,0
20 READ A : MOUT$=""
30 WHILE A<>0 : MOUT$=MOUT$+CHR$(A) : WEND
40 PRINT MOUT$
```

Les caractères de contrôle intéressants ici sont : 8 (curseur gauche), 9 (curseur droite), 10 (curseur bas), 11 (curseur haut), 12 (vidage écran), 13 (retour chariot) et 30 (curseur au coin supérieur gauche). Nous verrons bientôt qu'on a aussi des caractères de changement de couleurs.

Exercice 2.3

Pourrait-on appeler la variable MOUTON$?

Comment positionner le dessin sur l'écran ? On dispose de l'instruction LOCATE.

Pour se positionner ligne L colonne K, il suffit de faire LOCATE

K,L; essayez par exemple :

```
LOCATE 15,10 : PRINT MOUT$
```

2.3 Effets de couleurs

En mode texte, il y a deux manières de commander la couleur des caractères qu'on imprime. En mode graphique, seule la première méthode est applicable. La première méthode contrôle aussi la couleur de fond de l'écran et celle de la bordure autour de l'écran.

Pour comprendre le fonctionnement des instructions de gestion de la couleur, il faut savoir que les couleurs sont gérées de manière indirecte. Tout se passe comme s'il y avait un catalogue de peintures et des pots choisis dans le catalogue. Ensuite, on a un stylo et du papier qui peuvent être trempés dans les pots. Le catalogue comprend 27 couleurs possibles. Selon le mode, on a à disposition 16 pots (mode 0), 4 pots (mode 1) ou seulement deux pots (mode 2). Sur 664/6128, on a en fait deux papiers et deux stylos car on distingue le papier et le stylo graphiques de leurs homologues texte. En fait, cette distinction n'a pas un intérêt très grand et si on ne fait rien, l'AMSTRAD les confond.

Un peu paradoxalement, le remplissage d'un pot a un effet rétroactif : si l'on change la couleur d'un pot, tout ce qui avait été coloré avec ce pot change de couleur. Au contraire, le trempage par exemple du stylo dans un pot n'a d'effet que pour les tracés futurs.

Maintenant, il faut remplir un pot avec une peinture. C'est le rôle de l'instruction INK de la forme :

```
INK n1,n2
```

qui "remplit le pot n1 avec la peinture n2" ou en d'autres termes associe la couleur n2 à l'encre n1.

Il y a aussi la forme INK n1,n2,n3 qui attribue à l'encre n1 une couleur qui clignote (ou oscille) entre les couleurs n2 et n3. La vitesse de clignotement est fixée par l'instruction SPEED INK t2,t3 où t2 et t3 sont les temps d'activité des couleurs n2 et n3 respectivement (en

Il y a une autre instruction qui, elle, détermine la couleur du cadre de l'écran. C'est BORDER c où c est un code de couleur de 0 à 26.

Exercice 2.4

On veut la bordure rose, le fond bleu et les caractères blancs.

Exercice 2.5 (macabre)

Préparez un faire-part de décès.

Ces propriétés ont pour conséquence que les caractères et les fonds consécutifs peuvent changer de couleurs à volonté puisqu'une couleur de caractères n'est valable que pour les caractères imprimés après le PEN ... qui la détermine et elle n'est valable que jusqu'au prochain PEN ... qui la change.

Exemple : PEN 3:PRINT "BONJOUR ";:PEN 2:PRINT "MADAME"

imprime BONJOUR en rouge et MADAME en bleu.

Exercice 2.6

Comment faire pour avoir dans la première moitié de l'écran du texte en rouge sur fond vert, et dans la deuxième moitié du texte en bleu sur fond gris clair ?

Ceci implique une "résolution" en couleurs égale à la maille caractères sur l'écran puisque l'on peut changer de couleurs d'une maille caractères à l'autre. Nous verrons que pour les graphiques, cette résolution est à l'échelle du pixel.

Exercice 2.7

Recouvrir l'écran d'un pavage aléatoire de couleurs.

Caractères de contrôle

La seconde méthode consiste à imprimer un caractère de contrôle (code < 32). Il se trouve que certains caractères de contrôle ont reçu pour rôle de simuler les instructions PAPER, PEN, INK et BORDER. En fait il faut imprimer une séquence de caractères puisqu'il faut fournir les paramètres.

Ainsi, la séquence CHR$(14);CHR$(n) simule PAPER n

CHR$(15);CHR$(n) simule PEN n
CHR$(24) échange les couleurs de PEN et PAPER
CHR$(28);CHR$(n1);CHR$(n2);CHR$(n3) simule INK n1,n2,n3

Exemple

Si nous voulons que notre mouton ait la tête rouge et le reste du corps jaune, le début de la chaîne MOUT$ s'écrira (on suppose les options de INK par défaut en mode 1) :

MOUT$=CHR$(15)+CHR$(3)+CHR$(214)+CHR$(15)+CHR$(1)+ ....

ce qui revient à PEN 3, impression de la tête et retour à PEN 1.

Les fenêtres

Un moyen de définir des zones sur l'écran ayant des couleurs différentes ou des textes bien délimités est d'employer les fenêtres. Une fenêtre se définit par :

WINDOW #f,g,d,h,b

où f est le numéro de fenêtre (de 0 à 7),g est la colonne (de 1 à 20, 40 ou 80) du coin supérieur gauche, h (de 1 à 25) est la ligne du coin supérieur gauche, d et b déterminent de même le coin inférieur droit. La valeur limite pour une colonne dépend du mode; attention on compte à partir de 1.

On imprime dans une fenêtre par PRINT #f, ... où f est le numéro. Cette instruction a pour cas particulier PRINT qui considère

la fenêtre 0 par défaut. Lorsqu'on imprime dans une fenêtre, ce qui est dans les autres fenêtres est en principe inchangé. En particulier si l'impression entraîne un défilement, celui-ci ne se produit que dans la fenêtre.

Il peut y avoir jusqu'à huit fenêtres distinctes. Elles peuvent se recouvrir en partie; mais attention, la gestion de l'AMSTRAD n'est pas aussi perfectionnée que certains logiciels professionnels. Si, en imprimant dans la fenêtre 1 qui recouvre la fenêtre 2, vous effacez un caractère de la fenêtre 2, celui-ci ne réapparaîtra pas lorsque vous effacerez la fenêtre 1. Le mieux est souvent de créer des fenêtres jointives mais non recouvrantes.

Les instructions locales à une fenêtre, c'est-à-dire qui admettent une forme portant sur une fenêtre sont CLS, LOCATE, PAPER, PEN, PRINT, LIST et INPUT; elles s'écrivent :

CLS #f (vidage de la fenêtre f)
LOCATE #f,k,l (les coordonnées sont comptées à partir du coin supérieur gauche *de la fenêtre*)
PAPER #f,n (p. ex. si une fenêtre 1 est définie, la séquence PAPER #1,3 : CLS #1 fait apparaître un rectangle rouge)
PEN #f,n (change la couleur d'écriture pour les PRINT #f pas pour les PRINT)
PRINT #f,expressions
LIST lignes,#f (listing dans une fenêtre; permet de visualiser simultanément deux parties du programme)
INPUT#f,"texte";variables (acquisition dans une fenêtre)

Une application spécialement intéressante d'INPUT dans une fenêtre est le contrôle d'une saisie au clavier-écran : vous définissez une fenêtre qui encadre exactement le nombre attendu et ainsi la frappe de l'utilisateur est guidée.

Au contraire, les autres instructions sont globales vis-à-vis des fenêtres. C'est notamment le cas de MODE : la taille des caractères et le nombre de couleurs sont les mêmes dans toutes les fenêtres, et de INK : le choix des couleurs est le même pour tout l'écran.

2.4 Les modes graphiques

Il ne s'agit pas à proprement parler de modes graphiques puisque, dans chacun des trois modes, les tracés graphiques coexistent avec le texte. Il faut noter qu'il ne peut y avoir de graphiques que dans la fenêtre 0.

Il peut être judicieux de se réserver la fenêtre 0 pour les dessins et de se définir une fenêtre texte de 4 ou 5 lignes en bas de l'écran pour une légende ou une explication. Cette fenêtre texte peut être utile aussi pour examiner le listing du programme.

Les trois modes diffèrent entre eux par la résolution horizontale, la résolution verticale restant de 200 dans tous les cas. Les résolutions horizontales sont comme nous l'avons déjà dit de 160, 320 et 640 pixels dans les modes 0, 1 et 2. Le pixel double de largeur lorsqu'on descend d'un mode, mais, pour compenser, on a un plus grand choix de couleurs. On a le même choix de couleurs que pour les textes : en mode 2, deux couleurs possibles, 4 en mode 1 et 16 en mode 0. Le mode 1 représente un excellent compromis.

La résolution en couleurs est la même que la résolution spatiale, c'est-à-dire que, quelle que soit la direction, deux pixels consécutifs peuvent être de couleurs différentes. C'est un avantage de l'AMSTRAD car ce n'est pas le cas pour tous les micro-ordinateurs familiaux.

Les coordonnées sont X en horizontale, de gauche à droite et Y en verticale, de bas en haut. Y va de 0 à 399, X va de 0 à 639 quel que soit le mode. Les coordonnées sont les mêmes dans tous les modes, ce qui permet une compatibilité des programmes de tracé, mais, bien sûr, les dessins seront plus grossiers en modes 1 et surtout 0. Les coordonnées verticales de 0 à 400 laisseraient supposer une résolution double de celle que nous avons annoncée : il n'en est rien, simplement les pixels ayant Y=2*K et Y=2*K+1 forment en fait le même pixel. C'est le même principe qui joue pour permettre d'avoir les mêmes coordonnées X dans tous les modes.

Un point très important est que vous pouvez spécifier des tracés hors des limites : bien sûr les points seront invisibles et les lignes seront tronquées à la frontière de l'écran. Tout se passe comme si

l'écran était une fenêtre dans un écran plus vaste. Cela vous évite de tester à tout moment si vos points sont bien dans l'écran. En fait, on peut changer l'origine des coordonnées et spécifier une fenêtre graphique.

Exemple : Faites (en mode direct) MODE 1 : CLS (résolution moyenne et vidage).

Faites ensuite PLOT 50,50 : DRAW 800,200

On obtient une droite sur l'écran, des points 50,50 à 800,200. La droite est arrêtée au bord droit de l'écran car la coordonnée X du point d'arrivée (800) est supérieure à la limite. Vous voyez que la droite est tracée quand même. Une droite tracée par DRAW a pour point d'arrivée le point spécifié et pour point de départ le dernier point obtenu; ici, on l'initialise par PLOT.

La droite est jaune : si l'on ne précise pas de couleur, le tracé est fait dans la couleur de PEN. Faites maintenant :

DRAW 400,400 : on obtient une droite entre les points 800,200 (qui est hors du dessin) et 400,400.

Faites maintenant DRAW 300,0,3 : on obtient une droite qui aboutit au point 300,0 mais, surtout, qui est rouge (couleur d'encre numéro 3). Si nous avions fait DRAW x,y,0 nous n'aurions rien vu car le tracé aurait été de la même couleur que le fond; c'est la méthode idéale pour effacer un tracé.

On voit qu'on peut avoir des tracés de différentes couleurs : pour changer de couleur entre deux tracés, il suffit de spécifier l'encre dans l'instruction de tracé ou de faire un PEN ...

Il est intéressant de voir ce qui se passe au croisement des deux droites : c'est la couleur de la dernière droite tracée qui domine au croisement. En fait, ce comportement peut être modifié (plus facilement sur 664/6128) avec ce qu'on appelle le mode de tracé. Nous le verrons plus loin.

Nous retrouverons plus en détail l'instruction DRAW au chapitre suivant qui décrit toutes les instructions graphiques.

CHAPITRE 3

GRAPHIQUES HAUTE RESOLUTION : GESTION

Dans ce chapitre, nous passons en revue les principales instructions graphiques du Basic de l'AMSTRAD et nous en faisons l'application dans le domaine de la gestion. Le chapitre suivant verra, lui, les instructions et les méthodes les plus spécialisées, avec leur application au dessin artistique et scientifique.

3.1 Les instructions graphiques

Les instructions graphiques se répartissent en trois catégories : les instructions de dessin effectif, les instructions de mode qui définissent les paramètres du dessin futur et une série de fonctions qui "lisent" des données concernant les dessins.

INSTRUCTIONS DE MODE

Ces instructions définissent les paramètres qui vont influer sur les dessins produits par les instructions de dessin qui suivent. Nous avons déjà vu les principales instructions de cette catégorie : MODE, INK, PAPER et PEN. Nous les rappelons pour mémoire.

3.2 MODE

C'est l'instruction de mode la plus importante puisqu'elle détermine le mode d'affichage. Le mode détermine simultanément la taille des caractères de texte et la résolution graphique. Textes et dessins peuvent être mélangés sur l'écran. La création d'une fenêtre texte en bas de l'écran ne dépend pas du mode : vous pouvez la créer par WINDOW. Rappelons que les graphiques ne peuvent apparaître que dans la fenêtre 0.

Le passage d'un mode à un autre est toujours associé à un vidage écran ce qui est un peu gênant.

INK

Cette instruction associe à un numéro d'encre une couleur véritable. Si on spécifie deux couleurs, on obtient un effet de clignotement. Seuls les deux premiers numéros sont utilisables en mode 2; on a droit à 4 numéros en mode 1 et à 16 en mode 0. On ne peut jamais avoir les 27 couleurs différentes possibles sur l'écran.

PAPER - PEN

Ces instructions attribuent au fond de l'écran (PAPER) ou au tracé (PEN) un numéro d'encre, la couleur véritable qui en résulte étant déterminée par INK. Le 664 et le 6128 possèdent en plus les instructions GRAPHICS PAPER et GRAPHICS PEN qui permettent de différencier la couleur des tracés graphiques. Sur le 464 qui ne dispose pas de GRAPHICS PAPER, le fond pour les graphiques a toujours l'encre n° 0.

CLS - CLG

Cette instruction a pour rôle de vider l'écran. CLS vide l'écran texte ou plutôt la fenêtre texte spécifiée (CLS #f). CLG vide l'écran

graphique ou plutôt la fenêtre graphique. Lorsqu'on n'a pas défini de fenêtres, on voit peu de différence, et pourtant il y en a.

1- CLS ramène le curseur texte en haut de l'écran et laisse le curseur graphique en place. CLG ramène le curseur graphique à l'origine et laisse le curseur texte en place.

2- CLS met la fenêtre spécifiée dans la couleur de fond correspondant au dernier PAPER. CLG met la fenêtre graphique dans la couleur de fond correspondant au dernier GRAPHICS PAPER; sur 464, c'est la couleur de l'INK 0 qui intervient.

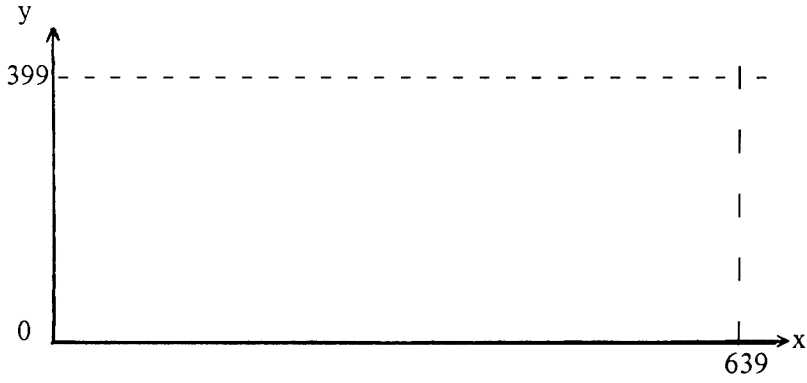
Curseur texte et curseur graphique

Puisque textes et graphiques coexistent dans la fenêtre 0, il y a deux curseurs, le curseur texte et le curseur graphique. Le curseur graphique marque la dernière position atteinte par un tracé. Il n'est pas matérialisé sur l'écran. Le curseur texte marque la prochaine position où apparaîtra un caractère; il est matérialisé par un rectangle clignotant.

Outre les déplacements automatiques, on peut déplacer le curseur graphique par MOVE, le curseur texte par LOCATE.

Les coordonnées

Dans tous les modes, l'écran graphique obéit au schéma :



Le fait que les coordonnées soient les mêmes dans tous les modes exige une explication. En fait, les coordonnées employées sont celles du mode le plus fin, le 2. Dans ce mode, deux valeurs de x consécutives correspondent à deux points (pixels) voisins mais distincts. Dans le mode 1, les pixels sont deux fois plus larges ou encore un pixel du mode 1 couvre deux pixels du mode 2, de sorte que les points de coordonnées x=2k et x=2k+1 sont confondus. Le phénomène s'accentue dans le mode 0 où un pixel est quatre fois plus large qu'en mode 2 de sorte que les points de coordonnées $2k \leq x \leq 2k+3$ sont confondus.

C'est le même principe qui explique que la résolution verticale est bien de 200 et non de 400 comme les coordonnées pourraient le faire croire : les points de coordonnées y=2l et y=2l+1 sont confondus.

Normalement, on spécifie des coordonnées conformes à ces schémas donc des ordonnées comprises entre 0 et 399 et des abscisses comprises entre 0 et 639.

Mais ce n'est pas obligatoire en vertu de deux phénomènes, le fenêtrage et le décalage d'origine. L'écran est considéré comme une fenêtre dans un espace plus grand et seuls les points qui tombent dans cette fenêtre sont tracés. Vous pouvez en fait spécifier des coordonnées comprises entre 32767 et -32767. Vous pouvez aussi spécifier les dimensions de la fenêtre graphique et décaler l'origine. C'est l'objet de l'instruction ORIGIN que nous voyons maintenant.

ORIGIN

Cette instruction effectue deux choses : elle fixe les coordonnées de la nouvelle origine des coordonnées et elle définit la fenêtre graphique (dans la "fenêtre WINDOW #0" puisqu'il n'y a qu'elle qui peut avoir des graphiques). Elle est de la forme :

ORIGIN X0,Y0,G,D,H,B

où X0,Y0 sont les coordonnées (exprimées par rapport à l'origine standard, c'est-à-dire le coin inférieur gauche de l'écran) du centre des nouvelles coordonnées : à partir de ce moment les coordonnées intervenant dans toutes les instructions de tracé seront exprimées par

rapport à la nouvelle origine. Les autres paramètres sont facultatifs : ils représentent respectivement les abscisses des bords gauche et droit et les ordonnées des bords haut et bas de la fenêtre graphique si l'on désire en créer une. S'ils sont absents, la fenêtre graphique s'étend sur tout l'écran.

Les paramètres de fenêtre sont exprimés en unités pixels. Si un paramètre implique une fenêtre dépassant un bord de l'écran, on n'en tient pas compte et la fenêtre s'arrêtera à ce bord.

Pour vous rendre compte, faites en mode direct la suite d'instructions :

```
ORIGIN 0,0,200,400,300,100
PAPER 3 : CLS
CLG
```

vous obtiendrez un rectangle bleu sur fond rouge.

Exercice 3.1

Expliquez cette suite d'instructions. Pourquoi les deux premiers paramètres de ORIGIN sont-ils 0 ?

Exercice 3.2

On vient de faire ORIGIN 320,200. Quelles sont maintenant les coordonnées du coin supérieur droit de l'écran ?

Une chose est à noter : dans l'instruction ORIGIN, ce sont toujours les coordonnées par rapport à l'origine standard qui interviennent : ainsi, pour rétablir l'origine au point bas gauche de l'écran, on fait ORIGIN 0,0 .

Coordonnées absolues et relatives

On peut spécifier les coordonnées d'un point qu'on veut désigner soit par rapport à l'origine - on dit que les coordonnées sont absolues,

soit par rapport à la position actuelle du curseur graphique - on dit que les coordonnées sont relatives. En coordonnées relatives, en somme, on spécifie les composantes du vecteur déplacement du curseur graphique. Les instructions graphiques ont en principe deux versions (p. ex. MOVE et MOVER, PLOT et PLOTR etc.) : la forme avec un R final utilise des coordonnées relatives.

Instructions de positionnement du curseur graphique
MOVE et MOVER

De la forme MOVE X,Y dans la version la plus simple, cette instruction positionne le curseur graphique au point de coordonnées absolues X et Y, tandis que MOVER DX,DY **déplace** le curseur graphique de DX,DY par rapport à sa position au départ de l'instruction (coordonnées relatives).

Sur 664 et 6128, les deux instructions admettent deux paramètres supplémentaires facultatifs :

MOVE X,Y,N,M

où N est un numéro d'encre : les tracés futurs seront dans la couleur correspondante, et M est le **mode** de tracé qui deviendra actif pour les prochains tracés (voir plus loin la question des modes).

Sur 464, il faut faire appel à DRAW ou PLOT pour changer la couleur de tracé.

XPOS,YPOS

Ces fonctions sans argument permettent de connaître à tout moment les coordonnées du curseur graphique. Ainsi, après un MOVE 100,200 , PRINT XPOS,YPOS fait imprimer : 100 200 .

INSTRUCTIONS DE TRACÉ

Ces instructions permettent de dessiner des points isolés ou des traits (segments de droite).Toutes admettent un couple de coordonnées

X,Y et un paramètre de couleur qui spécifie le numéro d'encre à employer.

La couleur effective dépend de l'association encre-couleur spécifiée par la dernière instruction INK exécutée. En mode 2, on n'a que deux choix possibles : 0 = couleur du fond, donc tracé invisible ou effacement et 1 = tracé visible (du moins si l'on a fait PAPER 0 et PEN 1, ce qui est le cas le plus fréquent). En mode 1, on a 4 choix possibles et en mode 0, on en a 16.

Sur 664/6128, on a en plus la possibilité de spécifier un paramètre de mode de tracé.

Les instructions existent en deux versions selon que l'on fournit des coordonnées absolues ou relatives. Pour un tracé, le point de départ est la position actuelle du curseur graphique. Ce sont les coordonnées du point d'arrivée qu'on fournit, et, après l'instruction, ce point devient la nouvelle position du curseur graphique.

PLOT - PLOTR

C'est l'instruction qui trace (ou efface) un point. Sa forme la plus simple est :

PLOT X,Y ou PLOTR DX,DY

qui trace le point de coordonnées X,Y (PLOT) ou le point de déplacements DX,DY par rapport au curseur graphique (PLOTR). Après l'instruction, le curseur graphique vient sur le point tracé.

Sous cette forme, le tracé se fait dans la couleur déterminée par GRAPHICS PEN sur 664/6128 et dans la couleur de l'encre 1 sur 464.

La seconde forme ajoute deux paramètres facultatifs permettant de déterminer la couleur de tracé et le mode. Le 464 n'admet que le premier paramètre (le n° d'encre) :

PLOT X,Y,N,M (même chose pour PLOTR)

où N est le n° d'encre (de 0 à 1, 3 ou 15 selon le mode graphique) qui

sera utilisé dorénavant pour les tracés et M (de 0 à 3) est le mode de tracé qu'on impose.

Il est temps d'expliquer ce que sont ces modes de tracé.

Les modes de tracé

Deux premiers éléments sont à noter tout d'abord :

1 - Il ne faut pas confondre le mode de tracé, qui détermine l'effet véritable d'un tracé, avec le mode graphique qui détermine la résolution du dessin et le choix de couleurs. D'ailleurs, le mode de tracé peut avoir les valeurs 0 à 3 alors que le mode graphique peut avoir les valeurs 0, 1 et 2.

2 - Le mode de tracé ne peut être imposé que sur 664/6128. Sur 464, les tracés sont faits en mode de tracé 0. Les effets correspondant aux autres modes ne sont accessibles que de façon très compliquée en Basic.

Pour comprendre comment fonctionnent les modes de tracé, nous raisonnons d'abord en mode graphique 2 dans lequel chaque pixel de l'écran correspond à un bit (pouvant valoir 0 ou 1) dans la mémoire d'écran et où les seuls numéros d'encre possibles sont 0 (point éteint = de la couleur du fond) et 1 (point allumé). Tracer un point consiste précisément à écrire la valeur voulue sur le bit correspondant de la mémoire d'écran. Si on veut allumer le point, on écrira la valeur 1 sur le bit, si on veut l'éteindre on écrira la valeur 0.

En mode de tracé 0, on impose d'autorité la valeur allumé ou éteint au point tracé quelque soit son état antérieur. Après PLOT X,Y,1 le point est toujours allumé; après PLOT X,Y,0 il est toujours éteint.

Dans les autres modes de tracé, l'effet obtenu est une combinaison du n° d'encre écrit et de l'**état antérieur** du point.

En mode de tracé 1, on fait l'opération OU EXCLUSIF entre les

deux. Sa table de vérité est :

		on écrit	on avait	on obtient
0 0 --> 0		éteint	éteint	éteint
0 1 --> 1	ce qui donne	éteint	allumé	allumé
1 0 --> 1		allumé	éteint	allumé
1 1 --> 0		allumé	allumé	éteint

En mode de tracé 2, on fait l'opération ET entre les deux. Sa table de vérité est :

		on écrit	on avait	on obtient
0 0 --> 0		éteint	éteint	éteint
0 1 --> 0	ce qui donne	éteint	allumé	éteint
1 0 --> 0		allumé	éteint	éteint
1 1 --> 1		allumé	allumé	allumé

En mode de tracé 3, on fait l'opération OU entre les deux. Sa table de vérité est :

		on écrit	on avait	on obtient
0 0 --> 0		éteint	éteint	éteint
0 1 --> 1	ce qui donne	éteint	allumé	allumé
1 0 --> 1		allumé	éteint	allumé
1 1 --> 1		allumé	allumé	allumé

On remarquera que si l'on vient de faire un tracé, il suffit pour l'effacer de le retracer en mode 1.

Dans les autres modes graphiques, les choses sont plus compliquées car les numéros de couleur sont sur plusieurs bits et on effectue bit à bit la combinaison des deux numéros de couleur (celui qu'on écrit et celui qui est déjà présent) et c'est donc un troisième numéro qui est obtenu en définitive.

Le tableau ci-dessous montre ce qui se passe pour le mode graphique 1 où on a les numéros de couleur 0 à 3 :

	on avait			
on écrit	0	1	2	3
0	0 0 0	1 0 1	2 0 2	3 0 3
1	1 0 1	0 1 1	3 0 3	2 1 3
2	2 0 2	3 0 3	0 2 2	1 2 3
3	3 0 3	2 1 3	1 2 3	0 3 3

Chaque case de ce tableau contient un triplet qui donne les n°s de couleur obtenus respectivement en mode OU EXCLUSIF, ET et OU. Par exemple, en mode ou exclusif, si sur un pixel de couleur 1 (jaune), on écrit en encre 2 (bleu clair), on obtiendra du rouge (couleur par défaut de l'encre 3)!!!

Pour obtenir ces effets sur 464, il vous suffit d'imprimer une séquence de caractères de contrôle. Ici, vous faites : PRINT CHR$(23);CHR$(MT) où MT est le mode de tracé que vous souhaitez obtenir.

DRAW - DRAWR

C'est l'instruction fondamentale de dessin. Elle permet de dessiner un segment de droite. Elle est de la forme :

DRAW X,Y,N,M ou DRAWR DX,DY,N,M

où N et M sont facultatifs et jouent le même rôle que dans PLOT et, bien sûr, seul N est spécifiable sur 464. Cette instruction fait tracer le segment de droite qui va de la position actuelle du curseur graphique au point de coordonnées spécifiées (de façon absolue ou relative selon qu'on emploie DRAW ou DRAWR). Dans le cas relatif, DX et DY

sont les composantes du vecteur tracé. Le vecteur peut être effacé si la couleur spécifiée est celle du fond.

A l'issue de l'instruction le point d'arrivée est la nouvelle position du curseur graphique.

Avec cette instruction, nous avons notre outil principal de dessin puisque toute courbe, aussi compliquée soit-elle, sera obtenue par juxtaposition de petits segments.

Exercice 3.3

Dessinez un triangle isoscèle d'axe vertical.

Pointillés

Les traits que nous obtenons avec DRAW ou DRAWR sont des traits pleins. Si nous voulions des traits pointillés, il faudrait alterner les DRAW et les MOVE, par exemple :

```
100 PLOT 100,100 : FOR I=1 TO 20
110 MOVER 5,7 : DRAWR 5,7
120 NEXT
```

Pour 664/6128, il y a un moyen encore plus simple et qui permet en plus de faire varier le type de pointillé obtenu. Il utilise l'instruction MASK qui n'existe pas sur 464 et qui est de la forme :

```
MASK motif,premier
```

où motif est un motif binaire sur 8 bits qui va définir le pointillé. Premier (paramètre facultatif) vaut 0 ou 1 selon que l'on veuille le premier point éteint ou allumé.

Chaque ligne va être considérée comme formée de suites de 8 pixels et les pixels correspondant à un bit 1 dans le motif seront allumés.

```
MASK &X00001111 définit un pointillé régulier,
MASK &X00110011 définit un pointillé plus fin,
MASK &X11000000 définit : - - - - ...
```

Dessin de rectangles

Dès que l'on sait tracer des segments, il est facile de tracer des rectangles, surtout si les côtés sont parallèles aux axes de coordonnées. Le sous-programme suivant dessine le rectangle qui part du curseur graphique et a pour côtés DX et DY :

```
10000 REM  RECTANGLE VIDE
10010 DRAWR DX,0
10020 DRAWR 0,DY
10030 DRAWR -DX,0
10040 DRAWR 0,-DY
10050 RETURN
```

Un appel possible de ce sous-programme serait :

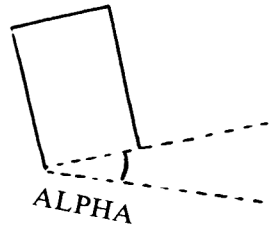
```
MOVE X,Y : DX= ... : DY= ... : GOSUB 10000
```

Exercice 3.4

Quelle instruction -la plus simple possible- ajouter à ce sous-programme pour qu'il admette en plus le paramètre N, numéro d'encre du tracé ?

Exercice 3.5

Dessinez le même rectangle mais ayant subi une rotation de l'angle ALPHA.



Exercice 3.6

Dessinez un carré qui ait une diagonale horizontale.

Rectangles pleins

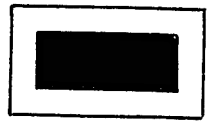
Sur 664/6128, pour dessiner un rectangle plein, il suffit d'en dessiner un vide puis de le remplir avec l'instruction FILL. Comme cette instruction n'existe pas sur 464, nous donnons ci-dessous un sous-programme valable pour tous les modèles. Il a pour paramètres d'entrée les dimensions DX et DY et, pour l'appeler, vous fixez les coordonnées du premier sommet et la couleur par PLOT X,Y,N : GOSUB 12000.

```
12000 REM  RECTANGLE PLEIN
12010 ORIGIN XPOS,YPOS
12020 FOR V=0 TO DY STEP 2*SGN(DY)
12030 MOVE 0,V : DRAW DX,V
12040 NEXT
12050 ORIGIN 0,0
12060 RETURN
```

L'élément qui nécessite peut-être le plus d'explications est le STEP en 12020. Il fait 2 unités puisque Y varie de 0 à 400 pour une résolution verticale de seulement 200 : si on prenait une unité, le tracé durerait deux fois plus longtemps sans aucun bénéfice. Ensuite il doit avoir le signe de DY d'où l'emploi de la fonction SGN; en effet, les DX et DY peuvent être négatifs selon l'orientation que vous voulez donner à votre rectangle par rapport à son premier sommet.

Exercice 3.7

Dessinez un rectangle plein à l'intérieur d'un vide (a). Faites aussi l'inverse (b).

(a)  (b) 

Tracé de cercles

Le Basic de l'AMSTRAD n'a pas d'instruction spéciale pour tracer

des cercles. Nous fournissons ci-dessous un sous-programme qui permet de tracer des cercles, des ellipses, des arcs de cercles et des polygones réguliers. Il a besoin des paramètres NE,XC,YC,R,F,AD,AF,AR et IA

où NE est le numéro d'encre de tracé

XC,YC sont les coordonnées du centre

R est le rayon horizontal, F est le facteur d'ellipticité : le rayon vertical est égal à F*R de sorte que, en fait, on trace une ellipse de demi-axes R et F*R.

AD et AF sont les angles de départ et de fin de l'arc tracé (valeurs par défaut 0 et 360 degrés)

AR est l'angle de rotation de la figure, ce qui permet d'avoir des ellipses d'axes non horizontaux

IA est l'incrément angulaire : en fait, on trace le polygone régulier à 360/IA côtés. La valeur conseillée est 2 degrés ce qui donne une bonne approximation du cercle; la valeur minimale 1 degré ne donne pas grande amélioration. Des valeurs sous-multiples de 360 donnent des polygones réguliers. Des valeurs supérieures à 90 donnent des polygones étoilés. Un polygone étoilé sera fermé à condition de spécifier comme angle de fin AF=IA*nombre de côtés à dessiner.

```
20000 REM  CERCLE
20010 REM
20020 ORIGIN XC,YC
20030 CC=COS(AR) : SS=SIN(AR)
20040 X=R*COS(AD) : Y=F*R*SIN(AD)
20050 XX=X*CC-Y*SS : YY=X*SS+Y*CC
20060 PLOT XX,YY,NE
20070 FOR I=AD TO AF+IA/10 STEP IA
20080 X=R*COS(I) : Y=F*R*SIN(I)
20090 XX=X*CC-Y*SS : YY=X*SS+Y*CC
20100 DRAW XX,YY
20110 NEXT
20120 ORIGIN 0,0
20130 NE=1 : F=1 : AD=0 : AR=0
20140 AF=8*ATN(1) : IA=AF/180
20150 RETURN
```

Avant une série d'appels à ce sous-programme, il est conseillé de donner des valeurs par défaut aux paramètres, ce qui permettra de ne donner que les valeurs spéciales au moment de l'appel. Ces valeurs par

défaut sont rétablies par le sous-programme au moment du retour :

```
NE=1 : F=1 : AD=0 : AR=0 : AF=8*ATN(1) : IA=AF/180
```

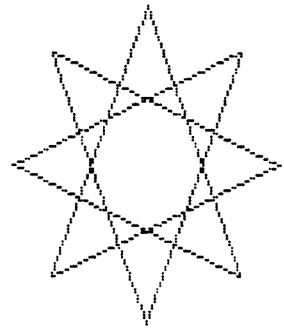
Quelques exemples

Mettez-vous en mode 1 avant de taper les exemples : les dessins obtenus sont déjà assez fins, mais vous pouvez aussi vous mettre en mode 2.

N.B. Nous reproduisons ci-après des copies d'écran sur imprimante; elles présentent une légère déformation par rapport à l'écran.

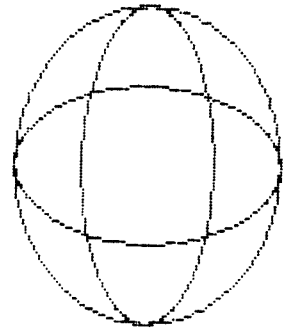
Polygone étoilé :

```
XC=200 : YC=200 : R=100 : DEG : AF=8*135 : IA=135 : CLG :
GOSUB 20000
```



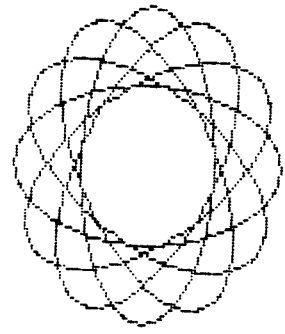
Cercle, ellipse de grand axe vertical, ellipse de grand axe horizontal :

```
10 CLS : R=100 : GOSUB 20000
20 F=2 : R=50 : GOSUB 20000
30 F=0.5 : R=100 : GOSUB 20000 : END
```



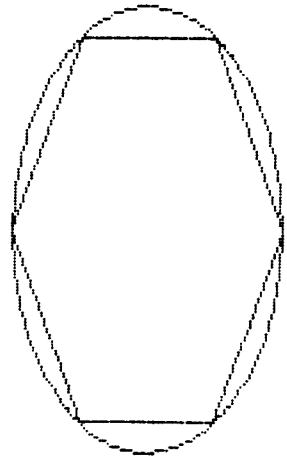
Ellipses avec rotations :

```
10 CLS : R=100 : DEG
20 FOR AA=0 TO 180 STEP 30
30 AR=AA : F=0.5 : GOSUB 20000
40 NEXT : END
```



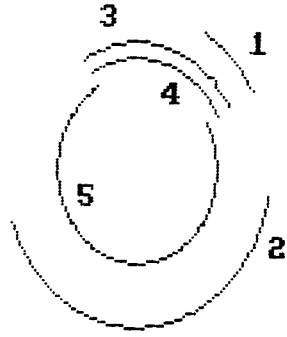
Hexagone inscrit dans une ellipse :

```
CLS : DEG : IA=2 : F=1.4 : GOSUB 20000
IA=60 : F=1.4 : GOSUB 20000
```



Quelques arcs :

```
CLS : DEG : R=100
AD=30 : AF=60 : GOSUB 20000                          (1)
AD=200 : AF=350 : GOSUB 20000                        (2)
AD=30 : AF=120 : R=80 : GOSUB 20000                  (3)
AD=20 : AF=110 : AR=10 : R=70 : GOSUB 20000          (4)
AD=120 : AF=390 : R=60 : GOSUB 20000                 (5)
```



Il faut penser à fournir tous les paramètres qui risquent d'avoir été changés par un tracé précédent. Si vous fournissez F=1, vous obtiendrez un cercle à peu près rond (il peut y avoir une légère déformation due au téléviseur ou au moniteur). Sur l'imprimante, la

déformation est plus importante. Avec la DMP 2000 et le programme de copie décrit plus loin, il faudrait F=0.83 pour avoir un cercle parfait.

Les angles sont indiqués en degrés ou en radians selon que l'on est en mode DEG ou RAD, dans le sens des aiguilles d'une montre. Le 0 est horizontal vers la droite. Les arcs sont supposés tracés dans le sens des aiguilles d'une montre.

Pour un polygone, vous pouvez changer son orientation soit par rotation générale, soit en donnant un angle de départ (à ce moment, il faut donner un angle final égal à 360+rotation si vous voulez que le polygone reste fermé). Pour des polygones inscrits dans des ellipses, cela fait une différence.

Exercice 3.8

Dessiner un secteur circulaire entre les angles A et B (un secteur circulaire est formé d'un arc et des rayons extrêmes).

Exercice 3.9

Dessiner une couronne circulaire pleine.

FILL

Voilà une instruction très intéressante dont le besoin s'est fait sentir à l'exercice précédent. Avec FILL il nous aurait suffi de dessiner les deux cercles extrêmes. En effet, FILL a pour résultat de remplir de couleur une zone limitée par des courbes. L'ennui est qu'elle n'existe pas sur 464, donc les exemples qui suivent ne s'appliquent qu'au 664/6128. Pour la simuler sur 464, il faut obligatoirement recourir au langage machine car un remplissage est toujours long. L'instruction s'écrit :

FILL NE

où NE est le numéro d'encre qu'on utilisera pour le remplissage (0 à

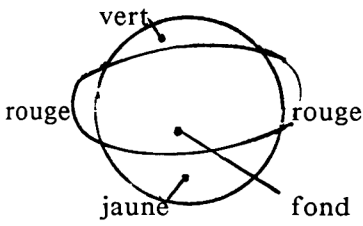
1, 3 ou 15 selon le mode graphique).

Les coordonnées du point de départ qui doit être à l'intérieur de la zone à remplir sont celles du curseur graphique.

Ex. GOSUB 20000 : MOVE XC,YC : FILL 1 dessine un disque (cercle plein)

Exercice 3.10

Produire la figure :



Exercice 3.11

Dessinez Saturne avec un anneau.

Mélange texte-dessin

On peut parfaitement mélanger textes et dessins avec les instructions que nous connaissons déjà : LOCATE pour se positionner où on veut sur l'écran et PRINT pour imprimer. Le seul inconvénient est que LOCATE procure un positionnement en mailles caractères ce qui n'est pas aussi précis que la résolution graphique. Le couple TAG / TAGOFF offre une solution : après TAG, les impressions (par PRINT) sont positionnées de sorte que le coin supérieur gauche du premier caractère à imprimer coïncide avec le curseur graphique.

A l'issue de l'impression de chaque caractère, le curseur graphique est déplacé de 8 pixels vers la droite, de sorte qu'on est prêt à imprimer le prochain caractère. Les instructions PRINT doivent se

terminer par un ';' car, en mode TAG, les caractères de contrôle apparaissent sous forme d'un graphique : ce serait le cas du retour-chariot qui apparaît si le PRINT n'est pas terminé par ';'.

Un autre effet important de TAG est que les textes écrits en mode TAG sont dans la couleur de l'encre graphique (spécifiée par GRAPHICS PEN sur 664/6128 ou par le paramètre d'encre de PLOT ou DRAW) et non dans la couleur de texte (spécifiée par PEN).

TAGOFF annule les effets de TAG : les PRINT sont à nouveau positionnés au curseur texte et non plus au curseur graphique et on revient à l'encre texte.

Exercice 3.12

Des cosmonautes ont marqué le nom SATURNE en lettres géantes sur la Planète Saturne de l'exercice précédent.

FONCTIONS DE LECTURE

Ces fonctions permettent au programme de prendre connaissance de tous les paramètres qui influencent l'affichage. De fait, le Basic de l'AMSTRAD ne possède que les fonctions XPOS et YPOS déjà vues et TEST qui permet de connaître l'état d'un point de l'écran. Pour prendre connaissance par programme des autres paramètres (ex. le mode d'affichage actuel), il faudra faire des PEEK aux adresses appropriées.

TEST - TESTR

Cette fonction est la plus importante : elle donne l'état d'un point de l'écran :

TEST(X,Y) a pour résultat le numéro de l'encre actuellement présente sur le point de coordonnées X,Y. En particulier, on obtient 0 si le point est éteint.

Un effet annexe de l'appel de la fonction TEST est d'amener le

curseur graphique en X,Y. Si vous voulez éviter cet effet, c'est-à-dire tester l'état d'un point tout en laissant le curseur graphique là où il est, il faut procéder par sauvegarde : XX=XPOS : YY=YPOS : T=TEST(X,Y) : MOVE XX,YY.

Il existe aussi la forme TESTR(DX,DY) qui déplace le curseur graphique de DX,DY et fournit l'état du point atteint.

Autres fonctions

Il peut sembler paradoxal d'avoir besoin de connaître, par exemple, le mode graphique actuel : le programmeur devrait le connaître puisque c'est lui qui l'a imposé et un programmeur organisé utilise une variable pour cela, faisant par exemple M= ... : MODE M à chaque changement de mode. Mais il y a un cas où ce n'est pas possible : lorsque vous écrivez un sous-programme que vous voulez général, sachant que c'est le programme appelant qui peut changer les paramètres.

Le Basic de l'AMSTRAD n'a pas de fonctions pour déterminer ces paramètres, mais vous pouvez les connaître par PEEK à certaines adresses, par exemple PEEK(&B1C8) donne le mode d'affichage, 0, 1 ou 2. Voici quelques adresses :

adresse 464	adresse 664/6128	rôle
B1C8	B7C3	Mode graphique
B1DA	B7D4	Table des couleurs des encres (32 octets : 2 pour chaque encre)
B285	B726	Ligne curseur texte
B286	B727	Colonne curseur texte
B28F	B72F	Encre caractères
B290	B730	Encre papier texte
B328,29	B693,94	Abscisse de l'origine des coordonnées
B32A,2B	B695,96	Ordonnée de l'origine des coordonnées
B338	B6A3	Encre de tracé graphique
B339	B6A4	Encre du papier graphique
B8F7	B113	0=RAD / 255=DEG

Nous avons maintenant passé en revue les instructions graphiques élémentaires. Certaines d'entre elles sont très élaborées et très puissantes. Il nous reste à les appliquer.

3.3 Tracé de courbes - remplissage - mélange avec du texte

Bien entendu, l'AMSTRAD peut tracer toutes les courbes possibles et imaginables. Nous verrons au chapitre suivant le tracé de courbes définies mathématiquement par une équation.

Ici, nous allons tracer une courbe définie par des données. Supposons que nous ayons les cours de deux actions pendant une dizaine d'années, de 1975 à 1985, trimestre par trimestre. Pour le premier trimestre 75, on a fait un changement d'échelle qui donne aux deux actions la valeur 100. Les cours sont dans des instructions DATA et on va tracer le diagramme en dents de scie correspondant. Voilà le programme :

```
1 REM DIAGRAMME EN DENTS DE SCIE
2 REM
10 MODE 1 : CLG
20 READ A,B : A=2*A : B=2*B
30 X=0 : FOR I=1 TO 43
40 XX=X+10 : READ AA,BB
50 AA=2*AA : BB=2*BB
60 PLOT X,A,1 : DRAW XX,AA,1
70 PLOT X,B,2 : DRAW XX,BB,2
80 X=XX : A=AA : B=BB
90 NEXT
100 END
110 DATA 100,100,105,98,107,92,110,85
120 DATA 112,80,120,83,122,84,124,87
130 DATA 126,87,130,89,132,90,133,92
140 DATA 135,95,137,98,137,101,139,104
150 DATA 142,107,148,110,144,111,142,112
160 DATA 145,114,149,117,154,122,159,125
170 DATA 165,136,168,146,172,149,130,125
```

```
180 DATA 130,126,131,126,132,128,134,129
190 DATA 134,131,132,129,131,128,129,128
200 DATA 130,129,135,133,138,137,140,139
210 DATA 144,142,145,144,147,146,147,143
```

Comment mettre en évidence la différence entre les deux actions ? En colorant la zone comprise entre les deux courbes, bien sûr. Rajoutons :

```
95 MOVE 50,200:FILL 1:MOVE 290,255:FILL 1
```

Ce qu'il faudrait maintenant, c'est marquer les années. Vous avez tout ce qu'il faut pour le faire.

Exercice 3.13

Faites-le. Vous pouvez aussi graduer l'axe des Y et donner un titre au diagramme.

3.4 Report sur imprimante

Voici une fonction tout à fait nécessaire. En effet, c'est bien beau d'avoir un magnifique dessin sur l'écran, mais il faut pouvoir en garder une copie durable pour les applications qui nécessitent de l'examiner à loisir. Bien sûr, il y a la possibilité de stocker la mémoire d'écran sur disque, mais ce n'est pas vraiment l'image qui est conservée : c'est seulement une possibilité de la revoir.

Donc la véritable copie durable, c'est la copie sur imprimante. Les imprimantes à matrice de points qui ont un mode graphique (c'est le cas des AMSTRAD) le permettent. Un problème supplémentaire est celui de la couleur : il y a bien sûr des imprimantes couleur, mais, de toutes façons la copie couleur serait trop longue à obtenir; en noir et blanc c'est déjà bien long. Pour un dessin en mode 2, il n'y a pas de problème : on fait la correspondance point allumé - point noir et fond - papier. En mode multicolore, on peut noircir tous les points colorés ou noircir seulement ceux d'une couleur; on peut même faire trois copies (à condition d'être patient!) : c'est ainsi que les imprimeurs

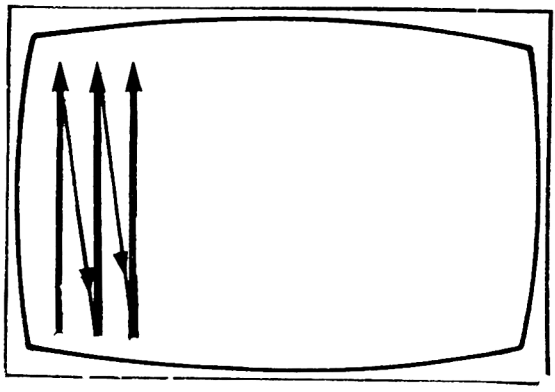
procèdent pour imprimer des pages en couleurs; il y a un film pour chaque couleur de base.

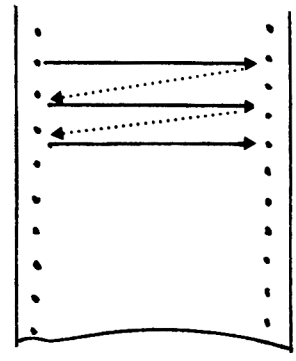
Pourquoi le programme de copie est-il si lent ? C'est parce qu'il faut parcourir l'écran point par point et, pour chaque point décider si on noircit le point correspondant sur la page. On utilise la fonction TEST pour cela. La correspondance entre le balayage écran et le balayage imprimante dépend du modèle d'imprimante. Nous donnons une version valable pour la DMP 2000.

Chaque colonne graphique se spécifie en envoyant CHR$(N), N étant calculé d'après le schéma :

	colonne	poids	
points	o	64	
	o	32	
	o	16	
	o	8	N=somme des poids des points
	o	4	à noircir
	o	2	
	o	1	

On va donc parcourir l'écran de gauche à droite et de bas en haut, parallèlement au balayage de l'imprimante suivant le schéma ci-dessous.





Un programme effectuant une telle copie est fourni dans

"Périphériques et fichiers sur AMSTRAD". Une copie prend une cinquantaine de minutes!! Nous donnons ci-dessous une version plus rapide.

La première amélioration à proposer est de sauter les valeurs impaires de Y puisque la résolution en Y est de 200 et non 400. Nous allons faire la même chose pour les X, pour une raison de conservation d'échelle entre les deux coordonnées : cela limite la déformation à environ 15%. Mais à ce moment, le sous-programme n'est plus valable en toute rigueur pour le mode 2 où il risque de perdre des pixels. Voici ce sous-programme, valable en toute rigueur pour les modes 0 et 1 :

```
60000 REM Copie Ecran H-R DMP 2000 mode 1
60001 REM
60010 WIDTH 255 :PRINT#8,CHR$(27);CHR$(49): G
$=CHR$(27)+CHR$(75)+CHR$(100)+CHR$(0)
60020 FOR X=0 TO 639 STEP 14
60030 A$="" : : FOR Y=0 TO 399 STEP 2
60040 N=0 : FOR I=0 TO 6
60050 N=N-(2^I)*(TEST(X+12-I-1,Y)<>0)
60060 NEXT : A$=A$+CHR$(N)
60070 NEXT
60080 PRINT#8,G$;LEFT$(A$,100);G$;RIGHT$(A$,1
00)
60090 NEXT
60100 PRINT#8,CHR$(15)
60110 RETURN
```

On obtient une copie en faisant GOSUB 60000.

On reconnaît en 60010 l'initialisation de l'imprimante à l'interligne 7/72è de pouce et la préparation de la chaîne G$ qui signifie "passage en mode graphique pour les 100 prochaines colonnes élémentaires". En 60020, boucle sur les X, avec STEP 14=7 points élémentaires * 2 puisqu'on saute un X sur 2.

En 60030, préparation de la chaîne A$ qui contiendra les données graphiques et boucle sur les Y. En 60040, préparation de la colonne élémentaire N et boucle sur les aiguilles. C'est en 60050 que tout se joue avec l'examen par TEST de l'état de chaque point élémentaire. En 60080, envoi de la chaîne graphique A$. Cela se fait en deux fois, car

on ne peut spécifier une valeur 200 (=nb. de caractères de A$) avec l'interface imprimante de l'AMSTRAD qui n'a que 7 bits. On partage donc la chaîne A$ en deux.

L'exécution prend environ 14 minutes. Le seul moyen véritable d'accélérer est de passer au langage machine. Si votre page écran n'est pas pleine, vous pouvez essayer de diminuer le 639 en 60020, le 3999 en 60030 et le 100 en 60010 et 60080 pour ne pas parcourir tout l'écran; vous pouvez d'ailleurs paramètrer ces quantités.

Exercice 3.14

Que faire pour que ce programme ne perde pas de pixels en mode 2 ?

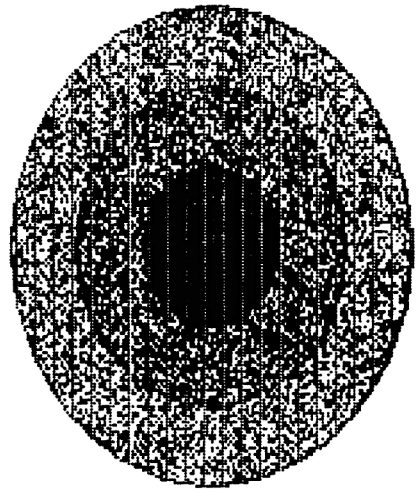
La version qui suit est réservée aux modes 0 et 1. Elle fait un grisé différent pour les numéros d'encre 0, 1(noir complet), 2 (gris foncé) et 3 et plus (gris clair). On procède ainsi : lorsque pour un groupe de 7 aiguilles c'est la première fois qu'on rencontre un pixel d'encre 2 ou 3 et plus, on noircit (cela assure de toujours noircir à une frontière). Si ce n'est pas la première fois, on noircit avec la probabilité 75% pour l'encre 2 et avec la probabilité 55% pour l'encre 3 ou plus. Les calculs de probabilité sont effectués dans le sous-programme 61000 et les variables A2 et A3 mémorisent l'arrivée pour la première fois ou non dans une couleur. On a aussi utilisé la variable K pour éviter les calculs de 2^I, ce qui gagne un peu de temps.

Ci-dessous, le résultat du programme pour la cocarde obtenue par les instructions 10 à 50 puis le sous-programme de copie avec grisés.

```
60000 REM Copie Ecran H-R DMP 2000 mode 1  avec
grises
60001 REM
60010 WIDTH 255 :PRINT#8,CHR$(27);CHR$(49): G$=C
HR$(27)+CHR$(75)+CHR$(100)+CHR$(0)
60020 FOR X=0 TO 639 STEP 14
60030 A$="" : : FOR Y=0 TO 399 STEP 2
60040 A2=0 : A3=0 : K=1 : N=0 : FOR I=0 TO 6
60045 T=TEST(X+12-I-I,Y)
60050 GOSUB 61000
60055 N=N+K*U : K=K+K
```

```
60060 NEXT : A$=A$+CHR$(N)
60070 NEXT
60080 PRINT#8,G$;LEFT$(A$,100);G$;RIGHT$(A$,100)
60090 NEXT
60100 PRINT#8,CHR$(15)
60110 RETURN
61000 REM  CALCULS DE GRISES
61010 IF T<2 THEN U=T : A2=0 : A3=0 : RETURN
61020 IF T=2 GOTO 61050
61030 A2=0 : IF A3=0 THEN A3=1 : U=1 : RETURN
61040 U=-(RND(1)<0.55) : RETURN
61050 A3=0 : IF A2=0 THEN A2=1 : U=1 : RETURN
61060 U=-(RND(1)<0.75) : RETURN
```

```
10 NE=1:DEG:AD=0:AF=360:AR=0:IA=2:F=1:XC=200:YC=
200
20 R=50:CLG:GOSUB 20000:MOVE XC,YC:FILL 1
30 R=100:GOSUB 20000:MOVE XC+60,YC:FILL 2
40 R=150:GOSUB 20000:MOVE XC+110,YC:FILL 3
50 END
```



3.5 Applications

Nous avons maintenant tout ce qu'il faut pour traiter les principales applications de gestion. Celles-ci consistent pour la plupart à tracer des diagrammes en bâtons ou en camemberts représentatifs d'une série de données. Il y a aussi les diagrammes en étoile. Nous commençons par eux car ce sont les plus simples.

Un diagramme en étoile est utilisé le plus souvent pour dessiner le profil d'un individu en fonction d'un certain nombre de qualités. A chaque qualité on attribue une direction dans le plan et l'individu concerné a un certain nombre de points relatifs à cette qualité. On dessinera un sommet dans la direction considérée, d'autant plus éloigné que l'individu possède plus fortement cette qualité. On obtient alors un polygone assez régulier si l'individu a des qualités équilibrées. Si l'individu a une lacune sur une qualité, il y a un angle rentrant; il y a une pointe s'il est extraordinaire dans un domaine.

Exemple

Essayez le programme suivant :

```
1 REM DIAGRAMME EN ETOILE
2 REM
10 ORIGIN 200,200:CLG:TAG:RAD
20 READ N$,N:A=2*PI/N
30 FOR I=0 TO N-1
40 READ X$
50 X=-20+100*COS(I*A):Y=100*SIN(I*A)
60 MOVE X,Y:PRINT STR$(I+1);
70 MOVE 160,150-20*I:PRINT STR$(I+1);"  ";X$;

80 NEXT
90 READ R:XA=R:YA=0
100 FOR I=1 TO N
110 PLOT 0,0:DRAW XA,YA
120 READ R:X=R*COS(I*A):Y=R*SIN(I*A)
130 DRAW X,Y
140 XA=X:YA=Y : NEXT
```

```
150 MOVE -50,-150:PRINT N$;
160 END
200 DATA ALFRED,5
210 DATA INTELLIGENCE,AUTORITE,COURAGE
220 DATA ADRESSE,CALME
230 DATA 65,30,80,72,24,65
```

On voit que l'individu Alfred manque d'autorité et de calme, ce qui est un handicap pour commander. Le programme ci-dessus est parfaitement général : il suffit de fournir les DATA convenables soit le nom de l'individu et le nombre N de qualités à envisager, puis les N libellés des qualités et enfin les valeurs. Les valeurs doivent être mises à l'échelle pour couvrir l'intervalle 0 à 90 et la première valeur doit être répétée à la fin pour que le polygone soit fermé.

Exercice 3.15

Ce programme marche très bien aussi pour visualiser des résultats scolaires. Quelles sont les données à fournir pour les résultats suivants :

mathématiques	15
physique	18
chimie	16
sciences nat	14
français	9
philosophie	13
hist-géo	11

3.6 Diagrammes en bâtons

Improprement appelés histogrammes (ce terme a en fait un sens bien spécial en Statistique), ces diagrammes permettent de représenter un certain nombre de données sous forme de rectangles de hauteurs proportionnelles aux valeurs. Les pyramides des âges sont une variante de ce type de diagrammes.

On peut dessiner des rectangles à plat et alors le sous-programme 10000 vu ci-dessus est bien adapté. On peut faire des rectangles de couleurs différentes ou non, pleins ou non. On peut aussi faire des

parallélépipèdes en perspective. On peut aussi utiliser deux couleurs pour représenter deux séries de données en parallèle.

Le programme ci-dessous lit une série de données en DATA et il les représente sous forme perspective. Les données sont mises à l'échelle afin d'être inférieures à 200. La première valeur lue est le nombre de données.

```
1 REM DIAGRAMME EN BATONS
2 REM
10 MODE 1:CLG:TAG
20 READ N:X1=16:Y1=100:Y4=116
30 FOR I=1 TO N
40 READ D:X2=X1+16:X3=X1+32
50 Y2=Y1+D:Y3=Y2+16
60 PLOT X1,Y1,1:DRAW X2,Y1:DRAW X2,Y2:DRAW X1,Y2

70 DRAW X1,Y1:DRAW X1,Y2:DRAW X2,Y3:DRAW X3,Y3
80 DRAW X2,Y2:DRAW X3,Y3:DRAW X3,Y4:DRAW X2,Y1
90 MOVE X1-8,80:PRINT STR$(I);
100 X1=X1+48
110 NEXT
120 DATA 10,50,64,110,180,134,160,150,90,74,42
```

Exercice 3.16

Graduer l'axe vertical.

Exercice 3.17

Une série de données qui vont 2 par 2 représente la consommation d'énergie en 1970, 1975, 1980 et 1985, respectivement sous forme pétrolière et sous forme non pétrolière. Représentez ces données sous forme de doubles bâtons : un bâton bleu pour l'énergie pétrolière, un bâton rouge pour l'autre.

3.7 Diagrammes camemberts

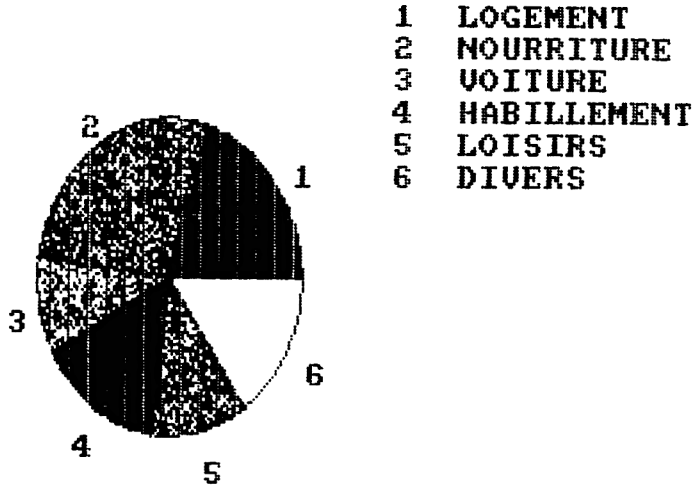
Ces diagrammes, que vous pouvez aussi appeler "parts de gâteau" si

vous préférez les pâtisseries, servent à représenter une répartition ou des pourcentages. Le total sera représenté par un cercle complet et chaque donnée est un pourcentage de ce total : elle sera représentée par un secteur de ce cercle d'angle proportionnel à sa valeur. Les secteurs seront en principe colorés de différentes couleurs.

Le programme suivant n'est pas très différent de celui du diagramme en étoile. Il est très général puisqu'il suffit de changer les DATA pour traiter n'importe quel problème.

```
1 REM DIAGRAMME CAMEMBERT
2 REM
5 DEF FNX(A)=200+100*COS(A)
7 DEF FNY(A)=200+100*SIN(A)
10 MODE 1:TAG:RAD:F=1:XC=200:YC=200
15 R=100:NE=1:AR=0
20 READ N$,N:AX=PI/180:IA=2*AX:AK=3.6
25 DIM AD(N+1),AF(N) : AD(1)=0
30 FOR I=1 TO N
40 READ X$,AN
45 AF(I)=AD(I)+AN*AK*AX:AD(I+1)=AF(I)
47 A=(AD(I)+AF(I))/2
50 X=180+120*COS(A)
60 Y=200+120*SIN(A)
70 PLOT X,Y,1:PRINT STR$(I);
75 PLOT 350,390-20*I:PRINT STR$(I);"  ";X$;
80 NEXT
90 C=1
100 FOR II=1 TO N
105 AD=AD(II):AF=AF(II):GOSUB 20000
110 X1=FNX(AD(II)):X2=FNX(AF(II))
115 Y1=FNY(AD(II)):Y2=FNY(AF(II))
120 PLOT 200,200:DRAW X1,Y1 :PLOT 200,200:DRAW X
2,Y2
125 IF II<>N THEN MOVE (X1+X2)/2,(Y1+Y2)/2:FILL
C
130 C=C+1:IF C=4 THEN C=1
140 NEXT
150 PLOT 150,50,1:PRINT N$;
160 END
```

```
200 DATA DEPENSES,6
210 DATA LOGEMENT,20,NOURRITURE,28
220 DATA VOITURE,10,HABILLEMENT,15
230 DATA LOISIRS,12,DIVERS,15
```



Ce programme utilise FILL. Sur 464, vous devrez l'adapter en vous inspirant de l'exercice 3.9.

Parmi les variantes possibles, on peut excentrer une part pour la mettre en évidence, on peut laisser les parts non colorées (nous l'avons fait pour le "divers" ci-dessus), on peut hachurer les parts. Un dessin un peu plus difficile est de représenter le gâteau comme un cylindre plat vu en perspective.

CHAPITRE **4**

GRAPHIQUES HAUTE RESOLUTION ARTS ET SCIENCES

Voici le chapitre des belles courbes, que leur beauté procède de la froide beauté des mathématiques ou de l'émotion artistique. Nous n'apprendrons pas d'instructions nouvelles dans ce chapitre, mais nous emploirons celles du chapitre précédent pour mettre en oeuvre des méthodes sophistiquées d'expression graphique comme celles qui sont nécessaires au dessin 3 dimensions (Ex. parties cachées).

4.1 Les systèmes de coordonnées

Ce terme recouvre en fait deux problèmes. Il y a d'abord un problème géométrique : le choix d'une représentation d'un point dans le plan ou l'espace. Ce choix revient à choisir entre coordonnées *cartésiennes* (X,Y ou X,Y,Z dans l'espace) et coordonnées *polaires* (dans le plan, un point est défini par l'angle Θ qui détermine la direction de la droite qui le relie à l'origine et par sa distance R à l'origine). Dans l'espace, on parle de coordonnées *sphériques*, la direction étant définie par deux angles.

Les coordonnées cartésiennes sont plus souvent utilisées pour trois raisons :

- les coordonnées polaires ne sont utiles que lorsque la symétrie particulière du problème à traiter rend les équations plus simples en coordonnées polaires

- ayant les coordonnées polaires, on a immédiatement les coordonnées cartésiennes. Supposant l'axe des Y orienté de bas en haut (c'est l'habitude de l'AMSTRAD) et l'origine des angles étant l'horizontale de gauche à droite, on a :

```
X=R*COS(TETA)
Y=R*SIN(TETA)
```

- les instructions graphiques de l'AMSTRAD ou des autres ordinateurs s'expriment en coordonnées cartésiennes.

On emploie parfois des coordonnées cartésiennes *homogènes* qui consistent à ajouter une 4è coordonnée W (dans l'espace : dans le plan, W est la 3è). Le point X,Y,Z a pour coordonnées homogènes X,Y,Z,1 ou tout quadruplet proportionnel. Les coordonnées homogènes ont l'avantage de traiter facilement les translations sous forme matricielle et de permettre de considérer le point à l'infini.

Le second problème est un simple problème d'unités sur les axes (coordonnées cartésiennes). On a déjà vu les plages de variation des coordonnées sur l'écran graphique : X de 0 à 639 et Y de 0 à 399. Il s'agit là de coordonnées "machine" ou "objet". Mais vous pouvez aussi travailler en coordonnées "sujet", c'est-à-dire en coordonnées exprimées dans les unités adaptées à votre problème. Vous pouvez aussi rester en coordonnées "objet" et employer à chaque fois la formule de transformation de coordonnées adéquate.

On emploie souvent cette dernière technique car les coordonnées objet sont les seules qui conservent l'isotropie du plan de l'écran et donc permettent de gérer avec sûreté des distances et des angles.

4.2 Tracé de courbes dans le plan

Une courbe peut être définie soit par son équation Y=F(X) soit par un couple d'équations paramètriques X=F1(T)
Y=F2(T)

Il y a deux tracés possibles : point par point, mais la courbe résultante peut s'avérer trop pointillée, ou par petits segments. Les deux programmes ci-dessous tracent une courbe Y=F(X) ou une

courbe paramétrique. Leur particularité est que l'on fournit au dernier moment, sous instructions imprimées par la première partie du programme, la (les) fonction(s) concernées en tapant les instructions DEF FN nécessaires puis GOTO l'instruction de suite du programme.

```
1 REM  TRACE DE COURBE Y=F(X)
2 REM
3 PRINT "Pour entrer la fonction a tracer"
4 PRINT "tapez 10 DEF FNF(X)= votre fonction Ret
urn"
5 PRINT "puis GOTO 10 Return"
6 END
10 DEF FNF(X)=EXP(X)
20 INPUT "BORNES,T/P";X1,X2,T$
30 H=(X2-X1)/300:XX=-2
40 CLG:PLOT 0,FNF(X1),1
50 FOR X=X1 TO X2 STEP H
60 XX=XX+2
70 Y=FNF(X)
80 IF T$="P" THEN PLOT XX,Y ELSE DRAW XX,Y
90 NEXT
```

```
1 REM  TRACE DE COURBE PARAMETRIQUE
2 REM
3 PRINT "Pour entrer les  fonctions a tracer"
4 PRINT "tapez 10 DEF FNF(T)= votre 1ere fonctio
n : DEF FNG(T)= votre 2e fonction Return"
5 PRINT "puis GOTO 10 Return"
6 END
10 DEF FNF(T)=200+100*COS(T) : DEF FNG(T)=200+10
0*SIN(T)
20 INPUT "BORNES,T/P";T1,T2,T$
30 H=(T2-T1)/300
40 CLG:PLOT FNF(T1),FNG(T1),1
50 FOR T=T1 TO T2 STEP H
60 X=FNF(T)
70 Y=FNG(T)
80 IF T$="P" THEN PLOT X,Y ELSE DRAW X,Y
90 NEXT
```

Ces programmes prennent systématiquement 300 points, mais vous pouvez changer ce paramètre. Ils n'ont aucune mise à l'échelle, ce qui suppose que les formules données pour les fonctions soient adaptées en conséquence. Les exercices suivants suppriment cet inconvénient. Ensuite, les programmes vous demandent d'entrer les bornes de variation de X ou de T et une lettre T ou P selon que vous désirez un tracé continu ou pointillé.

Exercice 4.1

Ramener à l'échelle une courbe fournie sous la forme Y=F(X). Il faut ramener à 0-600 l'intervalle X1-X2 et calculer le maximum et le minimum de Y pour ramener à 0-400 leur intervalle.

Exercice 4.2

Faire de même pour une courbe paramètrique. Faire attention à conserver l'isotropie.

Exercice 4.3

Il faut aussi dessiner et graduer les axes.

Ces programmes donnent une bonne idée de l'allure d'une courbe. Il peut être utile de joindre à ces programmes le sous-programme de copie sur imprimante que nous avons vu précédemment. Un cas particulier de courbe paramètrique est :

```
X=XC+R*COS(T)
Y=YC+R*SIN(T)
```

Le résultat n'est autre qu'un cercle. Nous avons plus simple pour le tracer, mais pas en pointillé.

Exercice 4.4

Quoi de plus simple pour tracer un cercle ?

Paramétrisation de la fonction à tracer

Les programmes que nous avons fournis permettent une certaine paramètrisation de la fonction à tracer puisque l'utilisateur tape celle-ci au dernier moment. Le procédé n'est toutefois pas très pratique puisqu'il faut taper l'instruction DEF FN complète et un GOTO 10 pour faire redémarrer le programme.

Nous donnons ci-dessous une autre version des premières instructions du programme pour les courbes Y=F(X) (l'adaptation aux courbes paramétriques est facile). Dans cette version, vous fournissez uniquement la fonction sous forme d'une chaîne de caractères et le programme implante lui-même le DEF FN. Pour cela, il écrit l'instruction sur un petit fichier ASCII et il **fusionne** ce fichier avec le programme à l'aide de CHAIN MERGE. Pour des détails sur cette instruction, voyez "Périphériques et fichiers sur AMSTRAD". L'utilisation est bien sûr plus commode sur disque que sur cassette (qui nécessite un rembobinage).

```
3 INPUT "Fonction a tracer";F$
4 OPENOUT "ZOZO"
5 PRINT#9,"10 DEF FNF(X)="+F$
6 CLOSEOUT
7 CHAIN MERGE "ZOZO",10
```

Familles de courbes

Une modification très simple des programmes ci-dessus permettrait de tracer plusieurs courbes sans effacer l'écran entre deux tracés. On pourrait donc avoir une famille de courbes.

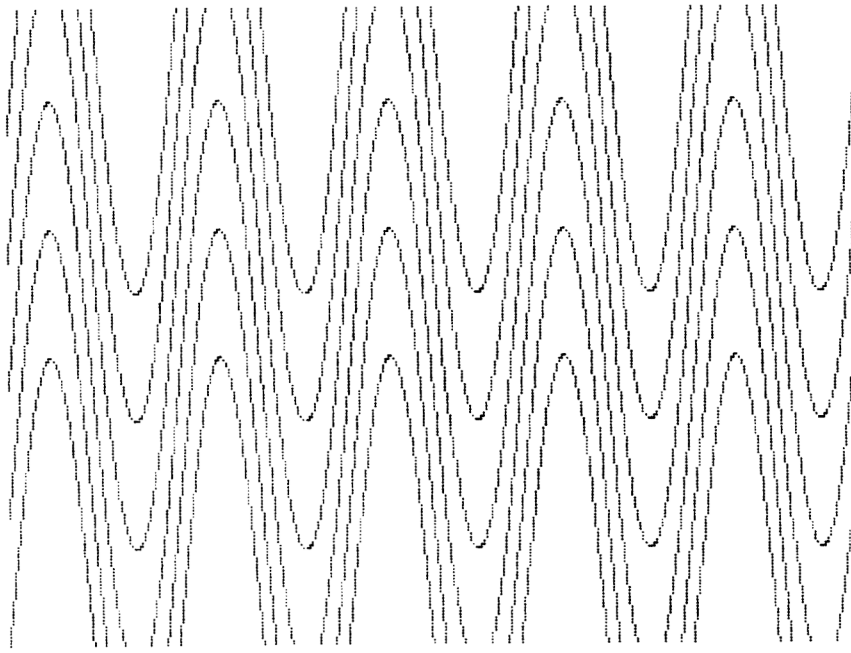
Exercice 4.5

Faites la modification voulue.

Nous allons examiner quelques exemples de courbes et de familles de courbes dont l'intérêt esthétique et mathématique ne vous décevra pas.

Nous commençons par une série de sinusoïdes déduites l'une de l'autre par translation.

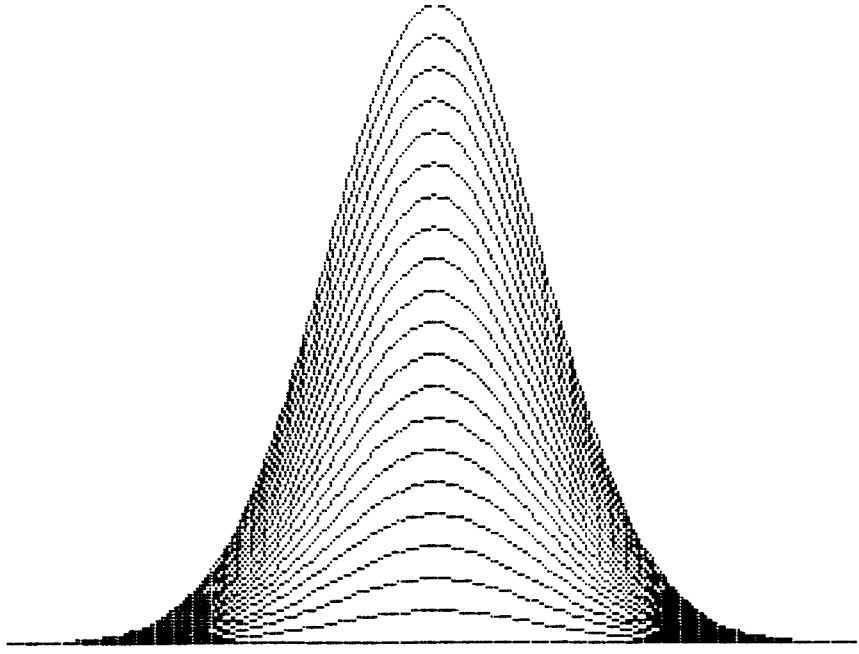
```
1 REM  SINUSOIDES
2 REM
5 CLG
10 FOR Y0=0 TO 400 STEP 80
20 DEF FNF(X)=Y0+180*SIN(PI*X/64)
30 X=0:PLOT X,FNF(X),1
40 FOR X=0 TO 639 : Y=FNF(X)
50 DRAW X,Y
60 NEXT
70 NEXT
```



On remarque qu'il n'y a aucune précaution particulière pour les points qui sont hors de l'écran : ils ne sont simplement pas dessinés. Cela s'appelle le "fenêtrage" : l'écran n'est qu'une fenêtre sur un espace plus vaste et l'AMSTRAD s'en occupe automatiquement.

Dans l'exemple suivant, on passe d'une courbe à l'autre par affinité. Voici une série de gaussiennes :

```
1 REM  GAUSSIENNES
2 REM
5 CLG
10 FOR K=0 TO 400 STEP 20
20 DEF FNF(X)=K*EXP(-(X-320)^2/12800)
30 X=0:PLOT X,FNF(X),1
40 FOR X=0 TO 639 STEP 5: Y=FNF(X)
50 DRAW X,Y
60 NEXT
70 NEXT
```

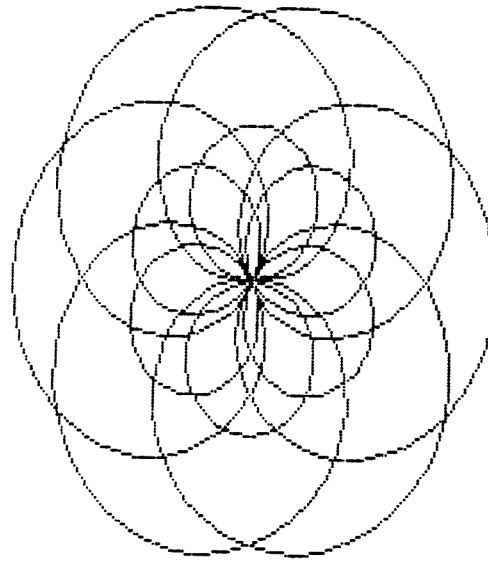


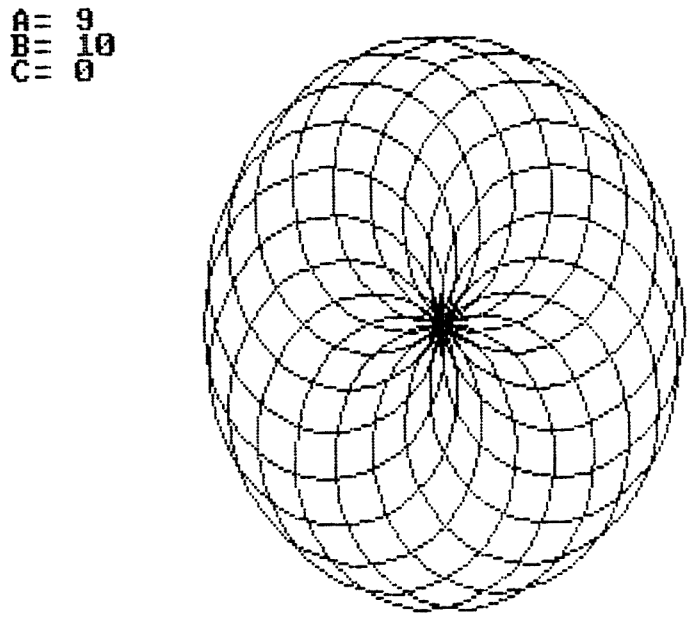
Courbes polaires

Voici quelques courbes en coordonnées polaires. En prenant l'angle comme paramètre, on a immédiatement les coordonnées paramètriques. Les fonctions de la forme R=COS ou SIN(A*T/B) avec A et B premiers entre eux donnent des courbes assez spectaculaires lorsque T varie de 0 à 2*B*PI. Lorsqu'on ajoute un terme constant, C, les résultats sont encore plus spectaculaires. Le programme suivant demande A,B et C à l'utilisateur. Essayez diverses combinaisons.

```
1 REM  COURBES POLAIRES R=(COS(A*T/B)+C)/(1+C)
2 REM
5 CLG
10 INPUT "A,B,C";A,B,C
20 DEF FNF(T)=(COS(A*T/B)+C)/(1+C)
30 DEF FNG(T)=320+180*FNF(T)*COS(T)
40 DEF FNH(T)=200+180*FNF(T)*SIN(T)
50 CLS
60 PRINT "A=";A
70 PRINT "B=";B
80 PRINT "C=";C
90 T=0:X=FNG(T):Y=FNH(T):PLOT X,Y,1
100 FOR T=PI/30 TO 2*B*PI+PI/30 STEP PI/30
110 X=FNG(T):Y=FNH(T)
120 DRAW X,Y
140 NEXT
```

```
A= 6
B= 7
C= 0.3
```



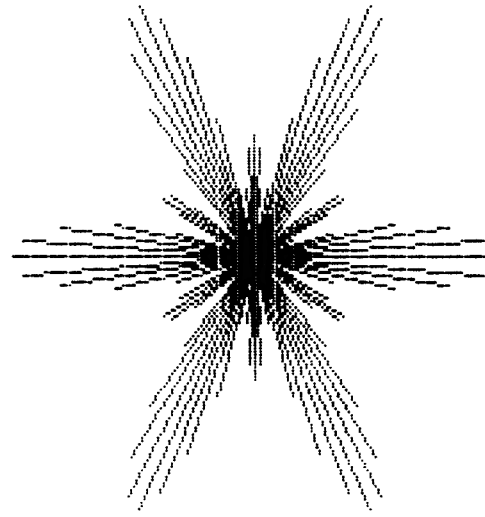


Tracés en rayons

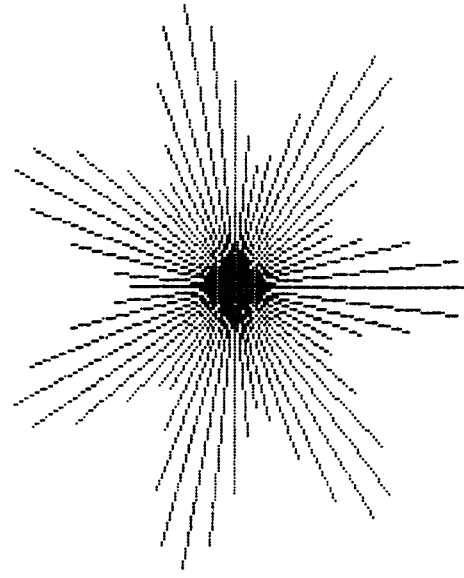
Un dessin spectaculaire s'obtient en traçant non pas la courbe mais les rayons qui joignent le centre aux points de la courbe. Il vaut mieux, cette fois prendre B=1. Il s'ajoute le paramètre N, demi-nombre de rayons à tracer.

```
1 REM  COURBES POLAIRES RAYONS
2 REM
5 CLG
10 INPUT "A,B,C,N";A,B,C,N
20 DEF FNF(T)=(COS(A*T/B)+C)/(1+C)
30 DEF FNG(T)=320+180*FNF(T)*COS(T)
40 DEF FNH(T)=200+180*FNF(T)*SIN(T)
50 CLS
60 PRINT "A=";A
70 PRINT "B=";B
80 PRINT "C=";C
90 PRINT "N=";N
100 FOR T=0 TO 2*B*PI+PI/N STEP PI/N
110 X=FNG(T):Y=FNH(T)
120 PLOT 320,200,1:DRAW X,Y
140 NEXT
```

```
A= 6
B= 1
C= 0.4
N= 50
```



```
A= 7
B= 1
C= 2.5
N= 30
```



Cycloïdes

Les épicycloïdes et hypocycloïdes ont des aspects assez semblables. Elles sont engendrées par un point d'un cercle de rayon R2 qui roule sans glisser sur un cercle de rayon R1. Leur équation générale est :

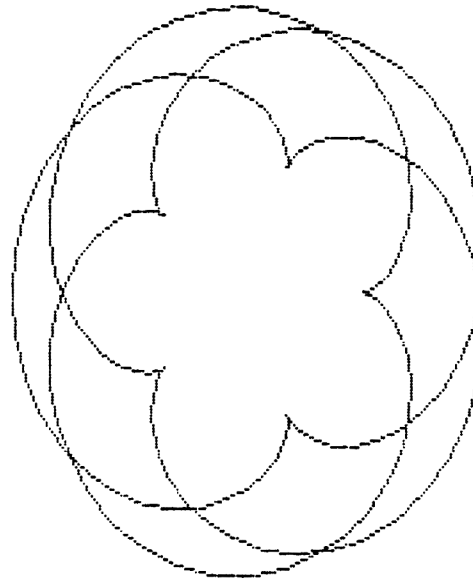
x=(R1+R2)cost -R2cos((R1+R2)t/R2)
y=(R1+R2)sint -R2sin((R1+R2)t/R2)

(R2<0 si hypocycloïde, >0 si épicycloïde)

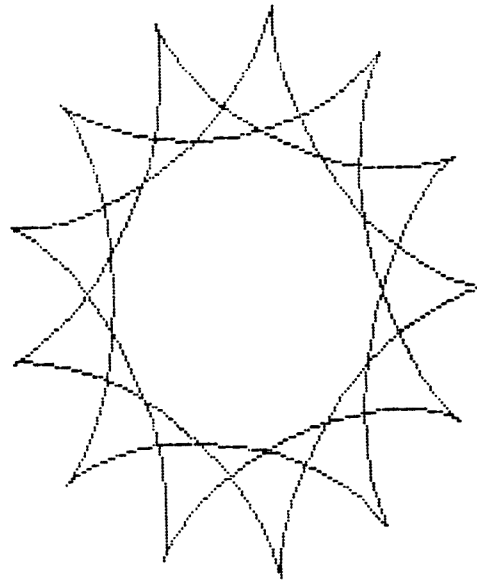
Le programme suivant demande les rayons (prenez des entiers pas trop grands et donnez R2 <0 pour une hypocycloïde). Les instructions 80 à 85 calculent combien de tours doivent être accomplis pour que la courbe se referme (on limite à 20).

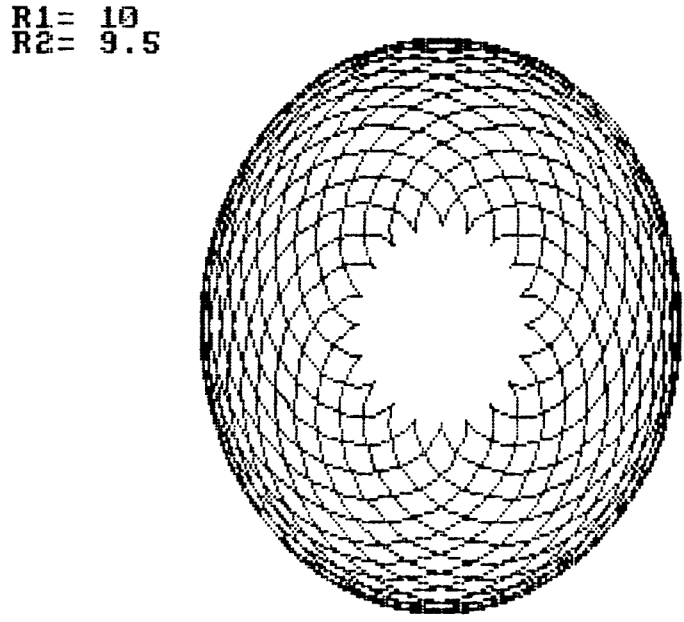
```
1 REM  HYPO ET EPICYCLOIDES
2 REM
10 INPUT "R1,R2";R1,R2:RR=R1+R2:RT=R1-2*R2*(R2>0
)
20 DEF FNF(T)=RR*COS(T)-R2*COS(RR*T/R2)
25 DEF FNE(T)=RR*SIN(T)-R2*SIN(RR*T/R2)
30 DEF FNG(T)=320+180*FNF(T)/RT
40 DEF FNH(T)=200+180*FNE(T)/RT
50 CLS
60 PRINT "R1=";R1
70 PRINT "R2=";R2
80 Q=R1/ABS(R2):FOR B=1 TO 20
83 IF B*Q=INT(B*Q) THEN 90
85 NEXT
90 T=0:X=FNG(T):Y=FNH(T):PLOT X,Y,1
100 FOR T=PI/30 TO 2*B*PI+PI/30 STEP PI/30
110 X=FNG(T):Y=FNH(T)
120 DRAW X,Y
140 NEXT
```

```
R1= 10
R2= 6
```



```
R1= 13
R2=-3
```





Courbes de Lissajous

Nous retrouverons des napperons un peu plus loin. En attendant, nous voyons les courbes de Lissajous qui fournissent des formes très variées. Leurs équations paramètriques sont de la forme:

x=sinat
y=sinct

La diversité des formes vient de la diversité des rapports entre a et c.

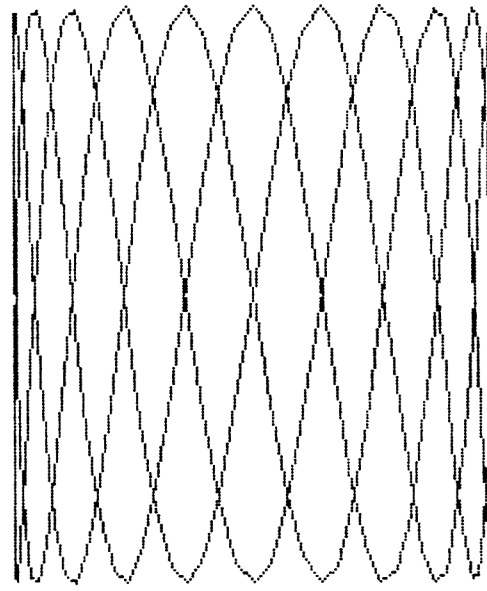
En fait, nous allons étudier des courbes plus générales de la forme:

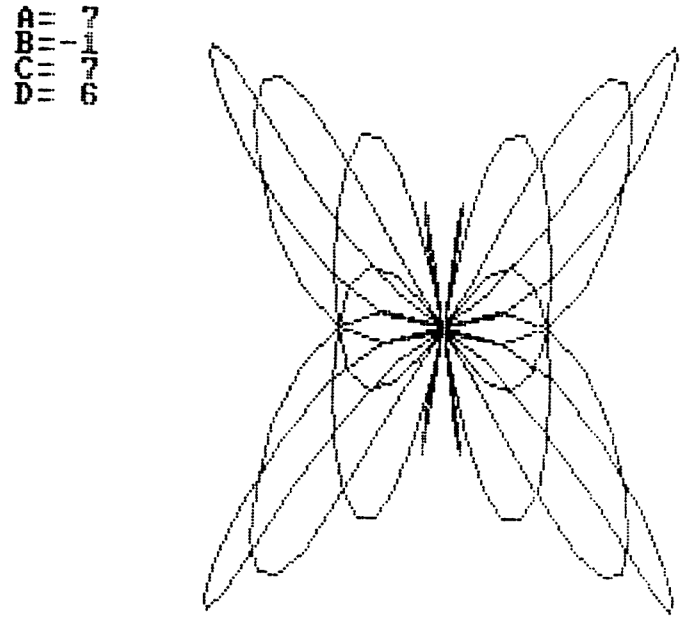
x=sinat*sinbt
y=sinct*sindt

qui se réduisent à la forme de Lissajous si l'on fournit b et d nuls. Aussi, si l'on fournit a,b,c ou d négatif, le système prendra la fonction cosinus au lieu de sinus.

```
1 REM  LISSAJOUS ET ANALOGUES
2 REM
10 INPUT "A,B,C,D";A,B,C,D
20 DEF FNF(T)=SIN(A*T+AA)*(SIN(B*T+BB)-(B=0))
25 DEF FNE(T)=SIN(C*T+CC)*(SIN(D*T+DD)-(D=0))
30 DEF FNG(T)=320+180*FNF(T)
40 DEF FNH(T)=200+180*FNE(T)
50 CLS
60 PRINT "A=";A
65 PRINT "B=";B
70 PRINT "C=";C
75 PRINT "D=";D
80 IF A<0 THEN A=-A:AA=-PI/2
82 IF B<0 THEN B=-B:BB=-PI/2
84 IF C<0 THEN C=-C:CC=-PI/2
86 IF D<0 THEN D=-D:DD=-PI/2
90 T=0:X=FNG(T):Y=FNH(T):PLOT X,Y,1
100 FOR T=PI/90 TO 2*PI+PI/90 STEP PI/90
110 X=FNG(T):Y=FNH(T)
120 DRAW X,Y
140 NEXT
```

```
A= 2
B= 0
C= 11
D= 0
```

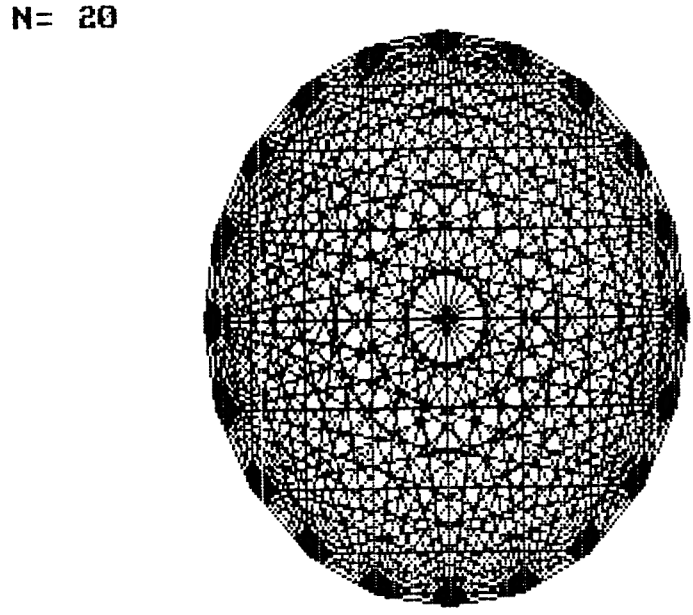




Napperons

Les napperons s'obtiennent en considérant les sommets d'un polygone régulier inscrit dans un cercle et en reliant chaque sommet à tous les autres. Le programme demande le nombre de sommets.

```
1 REM  NAPPERON
2 REM
10 INPUT "NB DE SOMMETS";N
20 DEF FNF(T)=320+180*COS(T)
30 DEF FNG(T)=200+180*SIN(T)
40 D=2*PI/N
50 CLS
60 PRINT "N=";N
90 FOR TA=0 TO 2*PI STEP D
100 XA=FNF(TA):YA=FNG(TA)
110 FOR T=TA+D TO 2*PI STEP D
120 X=FNF(T):Y=FNG(T)
130 PLOT XA,YA:DRAW X,Y
140 NEXT:NEXT
```



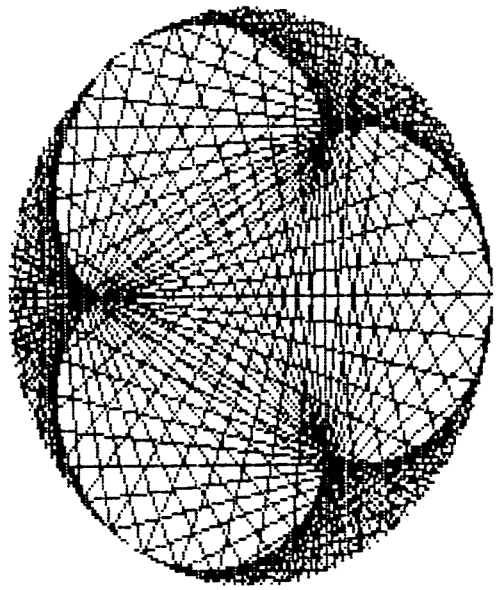
Enveloppes

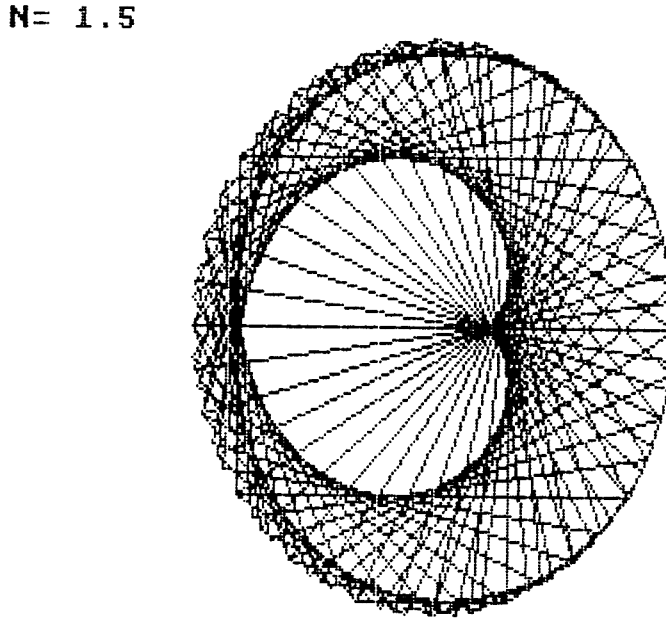
Dans l'exemple précédent, on voit se dessiner des cercles concentriques : ils ne sont pas dessinés intentionnellement, ils se dessinent parce qu'il y a un ensemble de droites qui leur sont tangentes. Ces cercles sont les *enveloppes* de ces droites et souvent, lorsqu'on dessine une famille de droites, leur enveloppe peut apparaître. Nous allons en voir quelques exemples du plus bel effet.

Commençons par des cordes de cercles. On trace la corde qui relie le point d'angle T au point d'angle N*T, pour toutes les valeurs de T de 0 à 2*PI par pas de PI. La figure obtenue s'appelle "bolygone" d'ordre N. Le programme demande l'ordre à l'utilisateur. Vous pouvez donner un ordre non entier, auquel cas le programme fera plusieurs tours pour que la courbe se referme (voir instructions 80-85). Voyez aussi l'instruction 40 qui calcule le pas en fonction de l'ordre.

```
1 REM  BOLYGONES
2 REM
10 INPUT "ORDRE";N
20 DEF FNF(T)=320+180*COS(T)
30 DEF FNG(T)=200+180*SIN(T)
40 D=2*PI/(40*N)
50 CLS
60 PRINT "N=";N
80 FOR B=1 TO 10
82 IF B*N=INT(B*N) THEN 90
85 NEXT
90 FOR TA=0 TO 2*PI*B STEP D
100 XA=FNF(TA):YA=FNG(TA)
110 T=N*TA
120 X=FNF(T):Y=FNG(T)
130 PLOT XA,YA:DRAW X,Y
140 NEXT
```

N= 4



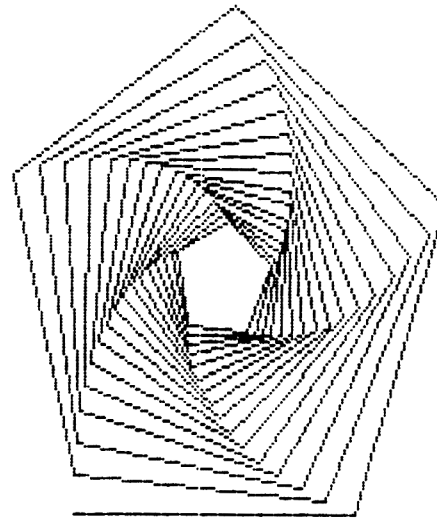


Polygones spiralés

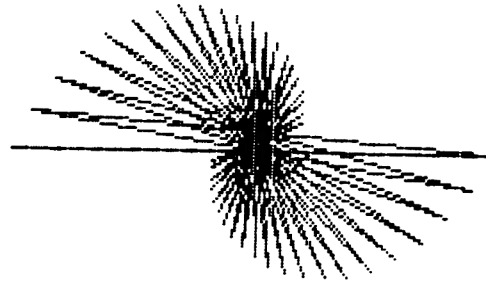
Notre prochaine famille est celle des *polygones en spirale.* On passe d'un côté au suivant en tournant d'un angle T et en multipliant la longueur par U (<1). Ces paramètres sont fournis par l'utilisateur ainsi que N, le nombre de segments à dessiner. Le programme commence par déterminer le segment de départ en fonction de l'angle T.

```
1 REM  POLYGONES SPIRALES
2 REM
10 INPUT "ANGLE,U,N";T,U,N
20 TR=T*PI/180
30 X=320-180*SIN(TR/2):Y=200-180*COS(TR/2)
40 L=2*180*SIN(TR/2):XA=X+L:YA=Y
50 CLS
60 PRINT "T=";T
70 PRINT "U=";U
75 PRINT "N=";N
80 PLOT X,Y,1:DRAW XA,YA:X=XA
90 FOR K=1 TO N
100 L=U*L
110 X=X+L*COS(K*TR):Y=Y+L*SIN(K*TR)
120 DRAW X,Y
130 NEXT
```

```
T= 71
U= 0.98
N= 80
```



```
T= 176
U= 0.96
N= 40
```



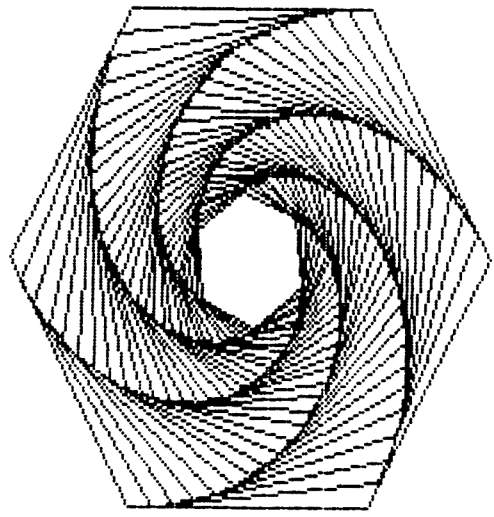
Polygones en diaphragme

Pour les polygones en diaphragme, on part d'un polygone régulier (il pourrait ne pas l'être) de N côtés. On passe d'un polygone au suivant de la manière que voici : chaque nouveau sommet -X- est sur le côté formé par deux anciens -soit A et B- de sorte que X divise AB dans un rapport fixe R. On obtient une apparence de rotation qui donne une forme analogue à celle des diaphragmes d'appareils photo. On choisit aussi le nombre P de polygones à tracer.

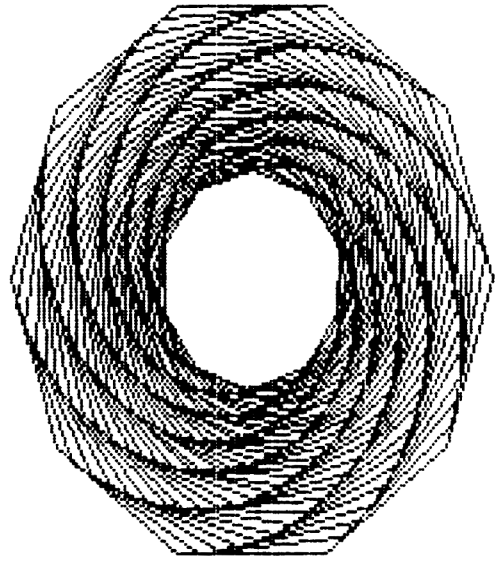
Les coordonnées des sommets sont dans deux tableaux, X et Y dimensionnés à N+1 : X(N+1) est la copie de X(1) utilisée parce que pour calculer le nouvel X(N) on a besoin de l'ancien X(1) et que celui-ci est déjà modifié.

```
1 REM  POLYGONES EN DIAPHRAGME
2 REM
10 INPUT "N,R,P";N,R,P
20 DIM X(N+1),Y(N+1)
30 DEF FNF(T)=320+180*COS(T)
40 DEF FNG(T)=200+180*SIN(T)
50 FOR I=1 TO N:T=2*PI*(I-1)/N
60 X(I)=FNF(T):Y(I)=FNG(T):NEXT
70 X(N+1)=X(1):Y(N+1)=Y(1)
80 CLS
90 PRINT "N=";N
100 PRINT "R=";R
110 PRINT "P=";P
120 GOSUB 200 : REM TRACE POLYGONE
130 FOR K=1 TO P : FOR I=1 TO N
140 X(I)=X(I)+(X(I+1)-X(I))/R
150 Y(I)=Y(I)+(Y(I+1)-Y(I))/R : NEXT
160 X(N+1)=X(1):Y(N+1)=Y(1)
170 GOSUB 200 : REM TRACE POLYGONE
180 NEXT
199 END
200 PLOT X(1),Y(1),1:FOR I=2 TO N+1
210 DRAW X(I),Y(I):NEXT:RETURN
```

N= 6
R= 10
P= 30



N= 10
R= 7
P= 40



L'enveloppe qui se dessine est une spirale logarithmique.

Exercice 4.6

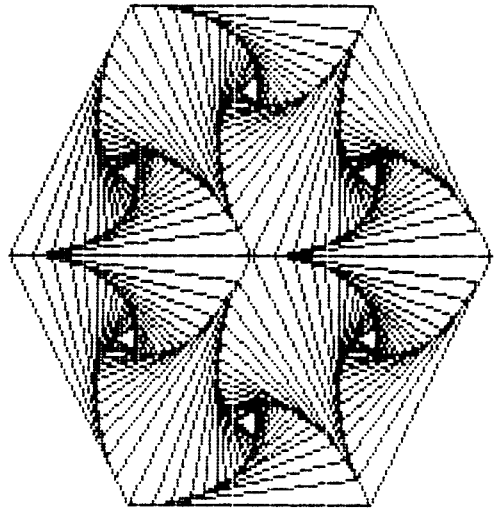
Comment changer le sens de la rotation apparente ?

Apparence 3-d

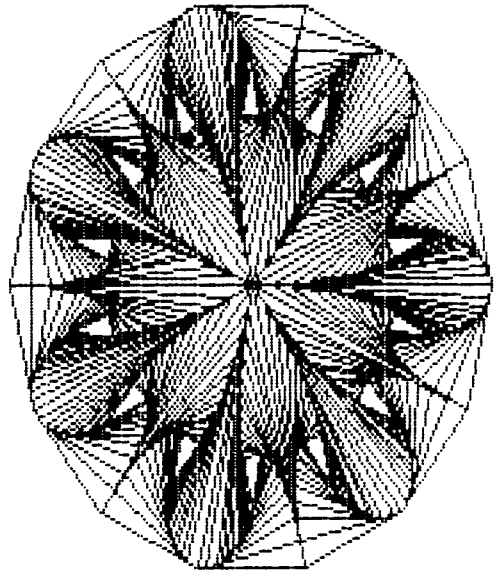
Soit maintenant un polygone régulier d'ordre N pair. On applique le traitement qui précède à chaque triangle isoscèle formé par le centre et un côté, à ceci près qu'on alterne le sens de rotation. On obtient des résultats saisissants qui ont l'apparence d'être en trois dimensions.

```
1 REM  APPARENCE 3 D
2 REM
10 INPUT "N,R,P";N,R,P :N2=N/2
20 DIM X(4),Y(4)
30 DEF FNF(T)=320+180*COS(T)
40 DEF FNG(T)=200+180*SIN(T)
50 CLS
60 PRINT "N=";N
70 PRINT "R=";R
80 PRINT "P=";P
90 FOR J=1 TO N2:TA=2*PI*(J-1)/N2
100 X(1)=320:Y(1)=200:X(4)=320:Y(4)=200
110 X(2)=FNF(TA):Y(2)=FNG(TA)
120 TB=TA-2*PI/N:X(3)=FNF(TB):Y(3)=FNG(TB)
130 GOSUB 300 : REM TRACE D'UNE SERIE
140 X(1)=320:Y(1)=200:X(4)=320:Y(4)=200
150 X(2)=FNF(TA):Y(2)=FNG(TA)
160 TB=TA+2*PI/N:X(3)=FNF(TB):Y(3)=FNG(TB)
170 GOSUB 300 : REM TRACE D'UNE SERIE
180 NEXT
199 END
200 PLOT X(1),Y(1),1:FOR I=2 TO 4
210 DRAW X(I),Y(I):NEXT:RETURN
300 GOSUB 200 : REM TRACE POLYGONE
310 FOR K=1 TO P
320 FOR I=1 TO 3
330 X(I)=X(I)+(X(I+1)-X(I))/R
340 Y(I)=Y(I)+(Y(I+1)-Y(I))/R : NEXT
350 X(4)=X(1):Y(4)=Y(1)
360 GOSUB 200 : REM TRACE POLYGONE
370 NEXT : RETURN
```

N= 6
R= 10
P= 15



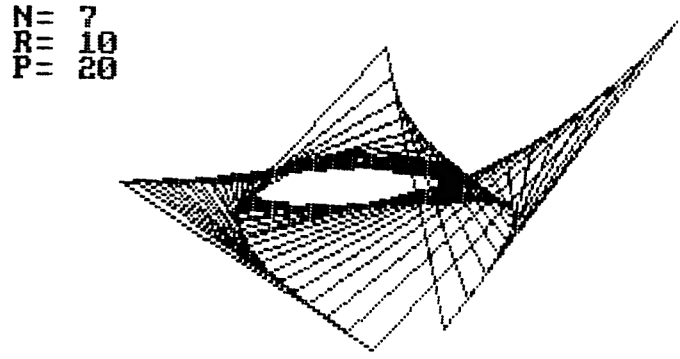
N= 14
R= 11
P= 12

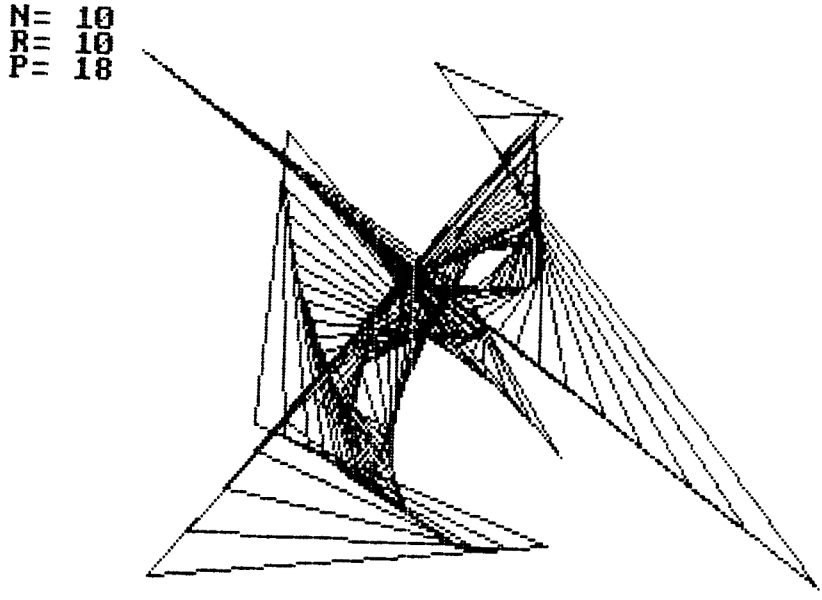


Polygones aléatoires

Nous pouvons appliquer le même traitement à un polygone de départ dont les coordonnées ont été "tirées au hasard" grâce à la fonction RND. C'est ce que fait le programme suivant qui demande à l'utilisateur le nombre de côtés N, le rapport de réduction R et le nombre de pas à effectuer P.

```
1 REM  POLYGONES ALEATOIRES
2 REM
10 INPUT "N,R,P";N,R,P
20 DIM X(N+1),Y(N+1)
30 FOR I=1 TO N
40 X(I)=100+INT(540*RND(1))
50 Y(I)=INT(400*RND(1))
60 NEXT
70 X(N+1)=X(1):Y(N+1)=Y(1)
80 CLS
90 PRINT "N=";N
100 PRINT "R=";R
110 PRINT "P=";P
120 GOSUB 200 : REM TRACE POLYGONE
130 FOR K=1 TO P : FOR I=1 TO N
140 X(I)=X(I)+(X(I+1)-X(I))/R
150 Y(I)=Y(I)+(Y(I+1)-Y(I))/R : NEXT
160 X(N+1)=X(1):Y(N+1)=Y(1)
170 GOSUB 200 : REM TRACE POLYGONE
180 NEXT
199 END
200 PLOT X(1),Y(1),1:FOR I=2 TO N+1
210 DRAW X(I),Y(I):NEXT:RETURN
```





Les sorties d'écran que nous avons reproduites sur imprimante étaient en mode 1. Nous vous conseillons d'examiner aussi ce que l'on obtient en mode 2 : vous utilisez à chaque fois exactement le même programme en ayant préalablement imposé le mode 2.

Un peu de couleur

Pour le moment, nous avons fait nos dessins en noir et blanc, uniquement pour des raisons de reproduction sur imprimante. Que se passerait-il si, avant de tracer chaque segment, on tirait sa couleur au hasard ?

Exercice 4.7

Faites-le.

Le résultat n'est pas désagréable, mais il y a une limitation : on ne peut avoir que 4 couleurs en mode 1. On peut en avoir 16 en mode 0, mais les dessins seront plus grossiers.

Exercice 4.8

Faites-le.

4.3 Dessins en trois dimensions

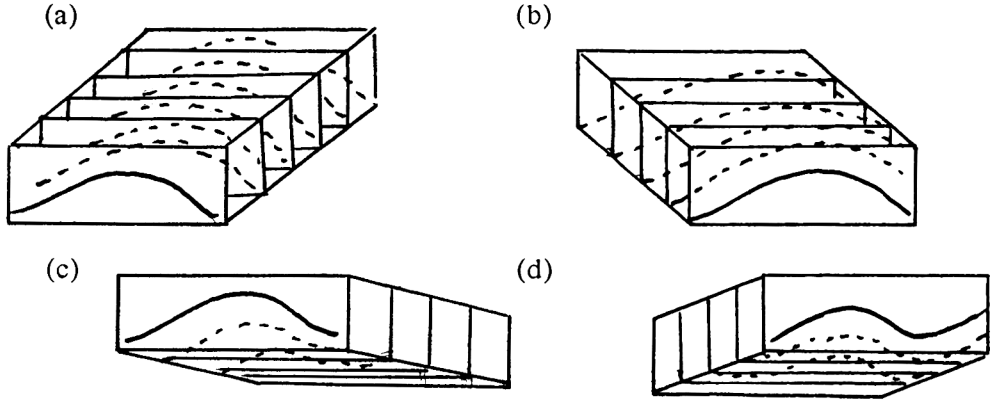
Les dessins précédents nous ont allèchés, mais nous allons devoir nous borner à quelques questions très simplifiées touchant au domaine du dessin en trois dimensions. Il y a en effet des algorithmes mathématiques très délicats qui interviennent et nous ne pourrons ici les aborder tous : nous pourrons juste en effleurer quelques uns.

Représentation d'une surface par une série de courbes

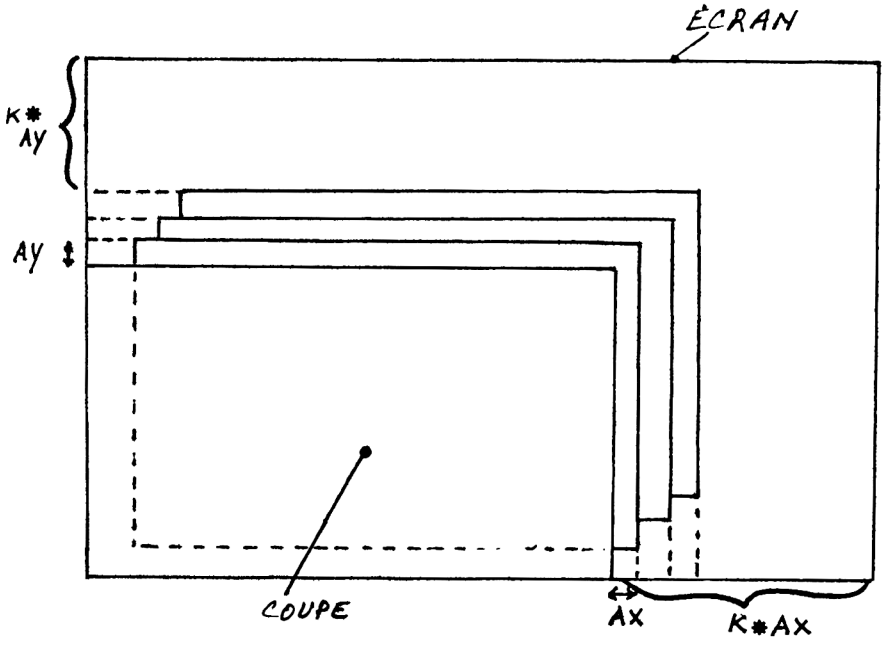
Soit une surface d'équation z=f(t,u) (ces coordonnées n'ont rien à voir avec les coordonnées X et Y sur l'écran). Pour la représenter, on imagine qu'on a effectué K coupes par des plans verticaux parallèles, par exemple des plans t=constante ou u=constante.

Comme on étudie la surface pour u compris entre U1 et U2 et t entre T1 et T2, on a en somme un parallélépipède à représenter. On a

alors le choix entre quatre représentations possibles :



On choisit le plus souvent (a) ou (b). Nous choisirons (a) mais l'adaptation à (b) est très facile. Alors, pour obtenir un effet de perspective cavalière, chaque rectangle qui représente une coupe est décalé de AX et AY par rapport à la coupe précédente :

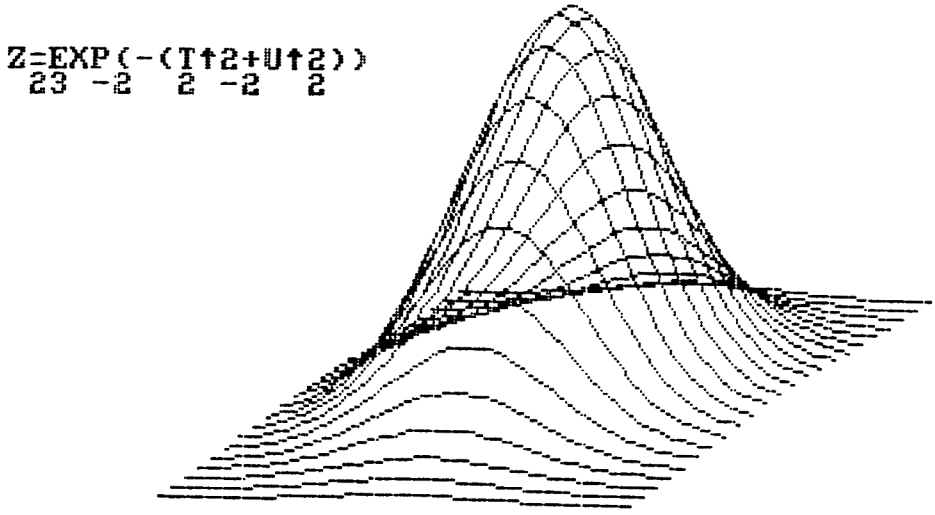


Maintenant, nous décidons que chaque rectangle fera 360 sur 240 (un peu moins des 2/3 de 640 sur 400) et que AX et AY vaudront respectivement 6 et 10. Cela permet un maximum de 25 coupes. Si vous voulez faire plus de coupes, ces paramètres sont faciles à changer. Vous fournissez la formule de la fonction en réponse à un INPUT (le programme applique la technique à base de CHAIN MERGE vue plus haut). Les variables sont T et U puisqu'on ne peut prendre X et Y (réservées à l'écran). Les coupes sont faites à U constant dans l'intervalle U1-U2; T varie de T1 à T2. Le U de la coupe n° I est U=U1+(U2-U1)*(I-1)/K donc vous devez fournir U1, U2 dans l'ordre 1ère coupe / dernière coupe (ce n'est pas forcément l'ordre croissant). Vous devez fournir T1 et T2 dans l'ordre gauche droite; dans la coupe n° I, le X qui correspond à un T est X=360*(T-T1)/(T2-T1)+20+(I-1)*AX; le Y qui correspond à un Z est Y=Z+(I-1)*AY.

Le programme commence par chercher le maximum et le minimum de la fonction donnée pour la mettre à l'échelle. Cette opération est très longue.

```
1 REM  COUPES 1
2 REM
10 INPUT "FONCTION";F$
11 OPENOUT "ZOZO"
12 PRINT#9,"20 DEF FNF(T)="+F$
13 CLOSEOUT
14 CHAIN MERGE "ZOZO",20
20 DEF FNF(T)=EXP(-(T^2+U^2))
30 INPUT "K,T1,T2,U1,U2";K,T1,T2,U1,U2
40 CLS : PRINT "Z="+F$ : PRINT K;T1;T2;U1;U2
45 HH=(U2-U1)/K:H=(T2-T1)/50:MA=-1E+20:MI=1E+20
50 FOR U=U1 TO U2 STEP HH : FOR T=T1 TO T2 STEP
H
60 Z=FNF(T) : IF Z>MA THEN MA=Z
70 IF Z<MI THEN MI=Z
80 NEXT : NEXT
90 DEF FNG(T)=BY+MI+240*(FNF(T)-MI)/(MA-M1)
100 DEF FNH(T)=360*(T-T1)/(T2-T1)+BX
110 AX=10 : AY=6
120 FOR I=1 TO K : BX=(I-1)*AX+20 : BY=(1-1)*AY
130 U=U1+(U2-U1)*(I-1)/K
```

```
140 PLOT FNH(T1),FNG(T1),1
150 FOR T=T1+H TO T2 STEP H
160 X=FNH(T):Y=FNG(T)
170 DRAW X,Y
180 NEXT
190 NEXT
```



Le défaut apparaît immédiatement sur la figure ci-dessus qui présente l'exemple de la surface en cloche d'équation Z=EXP(-(T^2+U^2)) étudiée de -2 à +2 sur les deux axes avec 23 coupes. Des courbes qui devraient être cachées sont affichées ce qui diminue la visibilité.

Il nous faut donc un algorithme de lignes cachées. Nous allons utiliser celui dit "des lignes de crête".

A chaque instant, on a parmi les lignes déjà tracées, la ligne la plus haute et la ligne la plus basse. On les appelle ligne de crête supérieure et ligne de crête inférieure. Soit maintenant un segment de la nouvelle ligne en cours de tracé : il ne sera tracé effectivement que s'il est au dessus de la ligne de crête supérieure ou au dessous de la ligne de crête inférieure. Les lignes de crête doivent être mémorisées et mises à jour à chaque tracé.

Comment mémoriser ? C'est facile : on constitue un tableau qui garde la valeur de l'ordonnée pour chaque abscisse. Il vient alors une difficulté : à cause du décalage AX, on ne connaît pas les valeurs aux mêmes abscisses lorsqu'on passe d'une coupe à la suivante. Il faudrait obtenir que l'écart entre les points soit égal à AX : dans le programme ci-dessus ce n'est pas le cas, l'écart est de 360/50=7.2 alors que DX=10. Mais de toutes façons avoir des segments sur pixels non consécutifs pose d'autres problèmes. Nous adopterons alors la solution la plus brutale qui consiste à prendre des pixels consécutifs et donc à ajouter l'instruction H=(T2-T1)/360 en 110. Cela ralentit beaucoup le tracé sur écran. On n'ajoute l'instruction qu'en 110, ce qui garde une vitesse raisonnable pour la recherche du maximum et du minimum.

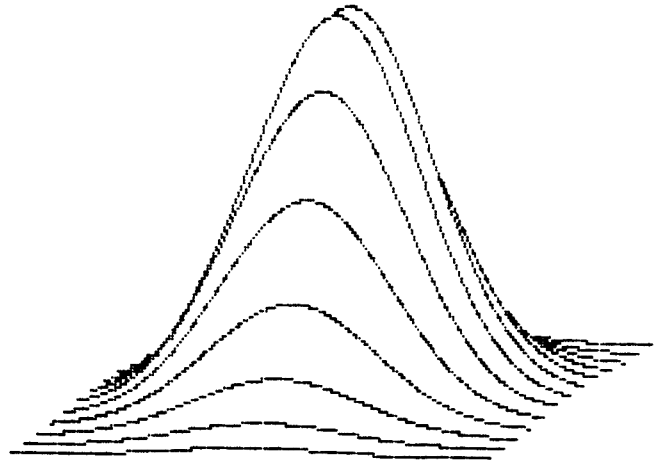
Les lignes de crête sont gardées dans les tableaux CS et CI dimensionnés à 639. Il reste le problème de leur initialisation. Il est simple : la ligne inférieure est initialisée pour être la plus haute possible, soit Y=400 et la ligne supérieure est initialisée pour être la plus basse possible, soit Y=0. Ainsi, la première ligne sera toujours tracée.

Voici le programme avec deux exemples de la même surface que ci-dessus, différant par le nombre de coupes : 13 et 23. Il faut vous armer de patience pour obtenir ces dessins : c'est aussi long que la copie sur imprimante. Le plus gênant est que la surface est parcourue deux fois pour trouver le maximum et le minimum pour l'échelle. Vous pourriez réaliser une version où l'on fournirait la fonction déjà à l'échelle. Sinon la mise en langage machine de certaines parties ne serait pas du luxe!

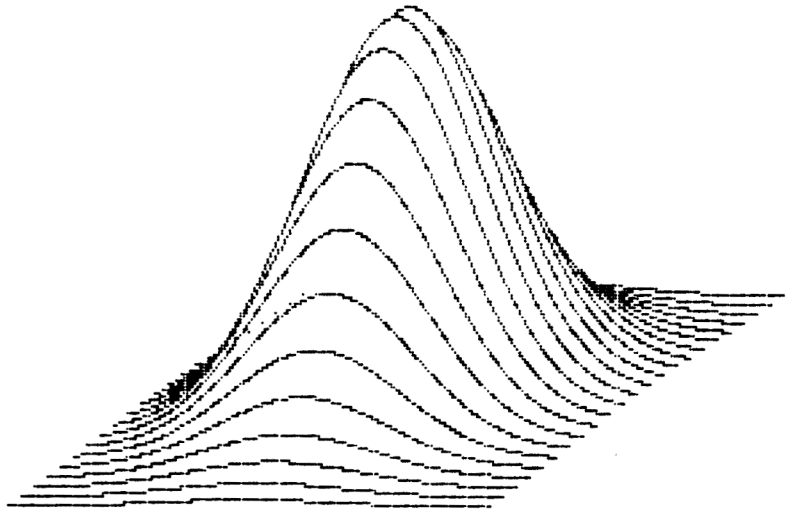
```
1 REM  COUPES AVEC LIGNES CACHEES
2 REM
10 INPUT "FONCTION";F$
11 OPENOUT "ZOZO"
12 PRINT#9,"20 DEF FNF(T)="+F$
13 CLOSEOUT
14 CHAIN MERGE "ZOZO",20
20 DEF FNF(T)=EXP(-(T^2+U^2))
22 DIM CS(639),CI(639)
24 FOR I=0 TO 639
26 CS(I)=0:CI(I)=400
28 NEXT
30 INPUT "K,T1,T2,U1,U2";K,T1,T2,U1,U2
```

```
40 CLS : PRINT "Z="+F$ : PRINT K;T1;T2;U1;U2
45 HH=(U2-U1)/K:H=(T2-T1)/50:MA=-1E+20:MI=1E+20
50 FOR U=U1 TO U2 STEP HH : FOR T=T1 TO T2 STEP
H
60 Z=FNF(T) : IF Z>MA THEN MA=Z
70 IF Z<MI THEN MI=Z
80 NEXT : NEXT
90 DEF FNG(T)=BY+MI+240*(FNF(T)-MI)/(MA-MI)
100 DEF FNH(T)=360*(T-T1)/(T2-T1)+BX
110 AX=10 : AY=6 : H=(T2-T1)/360
120 FOR I=1 TO K : BX=(I-1)*AX+20 : BY=(I-1)*AY
130 U=U1+(U2-U1)*(I-1)/K
140 PLOT FNH(T1),FNG(T1),1
150 FOR T=T1+H TO T2 STEP H
160 X=FNH(T):Y=FNG(T)
165 TR=0:IF Y>CS(X) THEN CS(X)=Y:TR=1
167 IF Y<CI(X) THEN CI(X)=Y:TR=1
170 IF TR=1 THEN DRAW X,Y ELSE MOVE X,Y
180 NEXT
190 NEXT
```

```
Z=EXP(-(T↑2+U↑2))
 13 -2  2 -2  2
```



```
Z=EXP(-(T↑2+U↑2))
  23 -2  2 -2  2
```

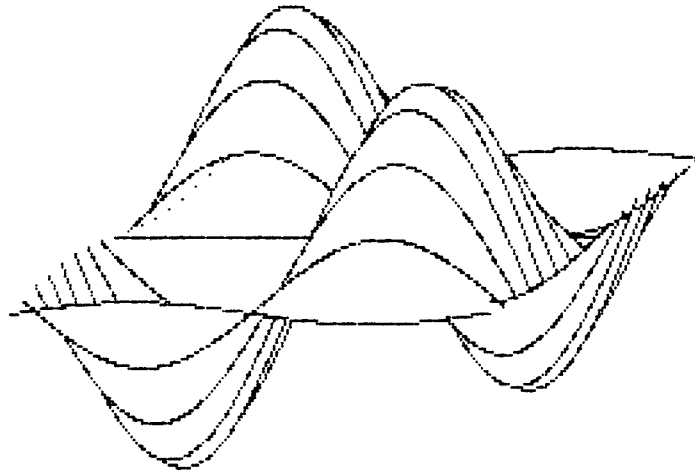


Exercice 4.9

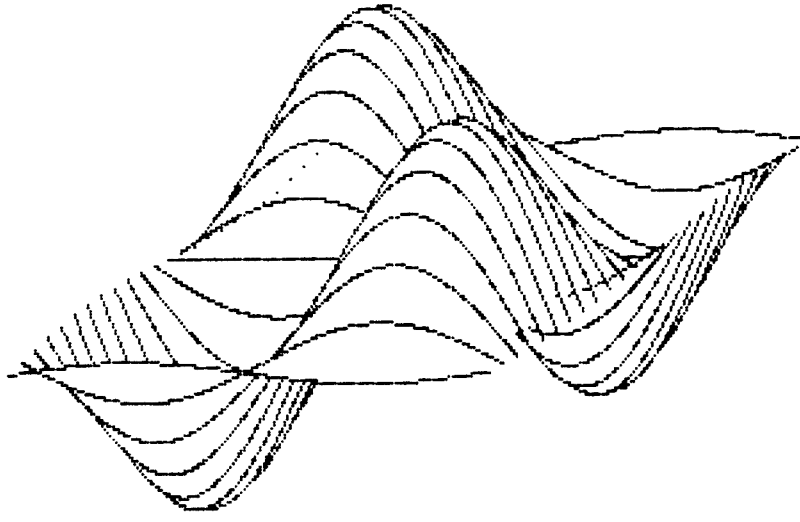
Il y a une légère erreur dans le programme ci-dessus (et aussi dans celui sans lignes cachées). Elle n'affecte pas le rendu apparent de la surface mais corrigez-la tout de même.

Une fois cette erreur corrigée, nous pouvons tracer quelques autres exemples.

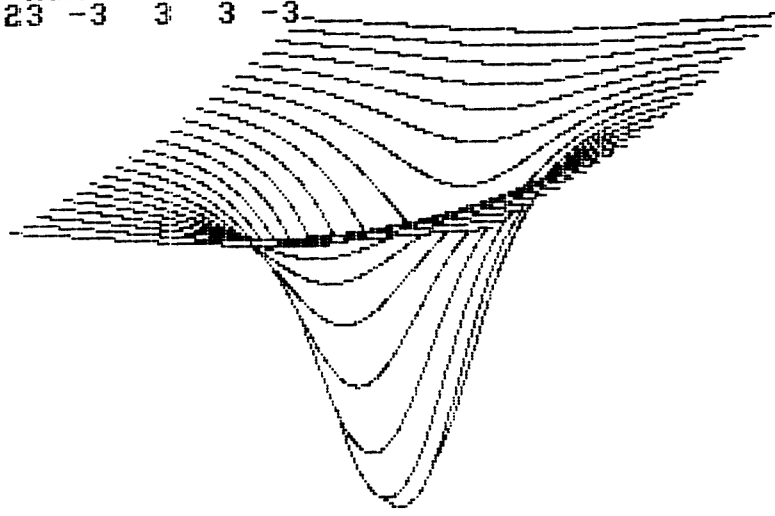
```
Z=SIN(T)*SIN(U)
 17 -3.2  3.2  3.2 -3.2
```



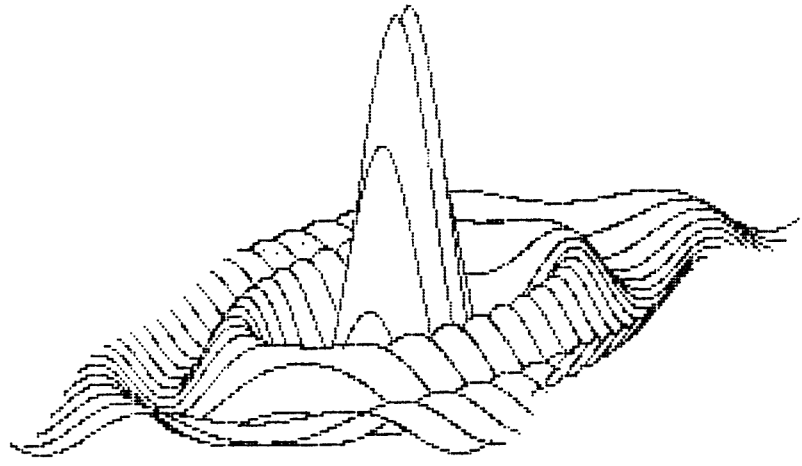
```
Z=SIN(T)*SIN(U)
 25 -3.2  3.2  3.2 -3.2
```

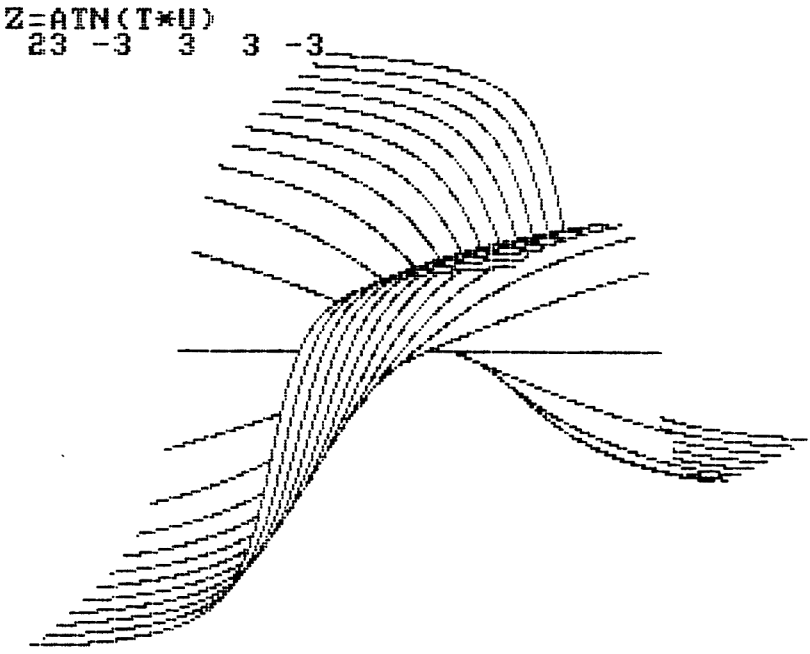


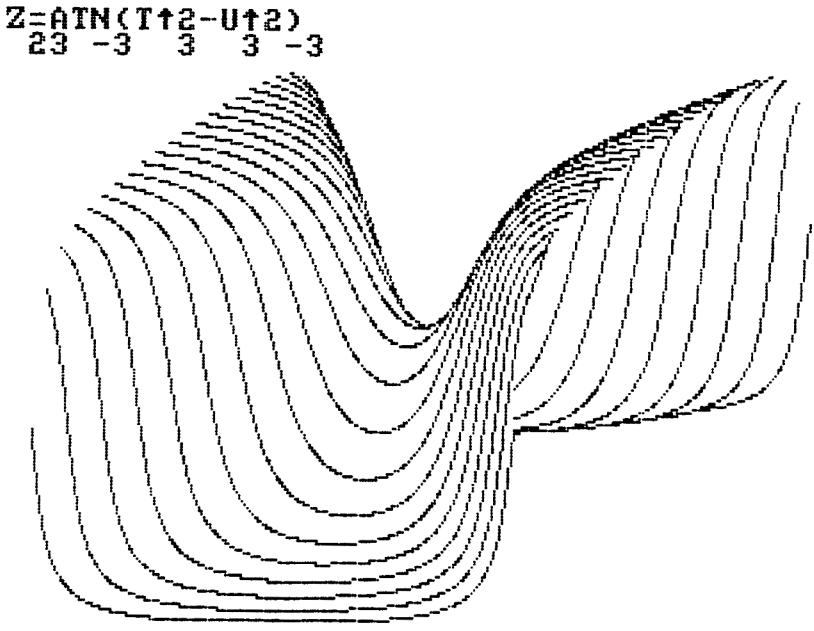
```
Z=ATN(T↑2+U↑2)
 23 -3  3  3 -3
```



```
Z=SIN(SQR(T↑2+U↑2))/SQR(T↑2+U↑2)
 24 -13  13  13 -13
```



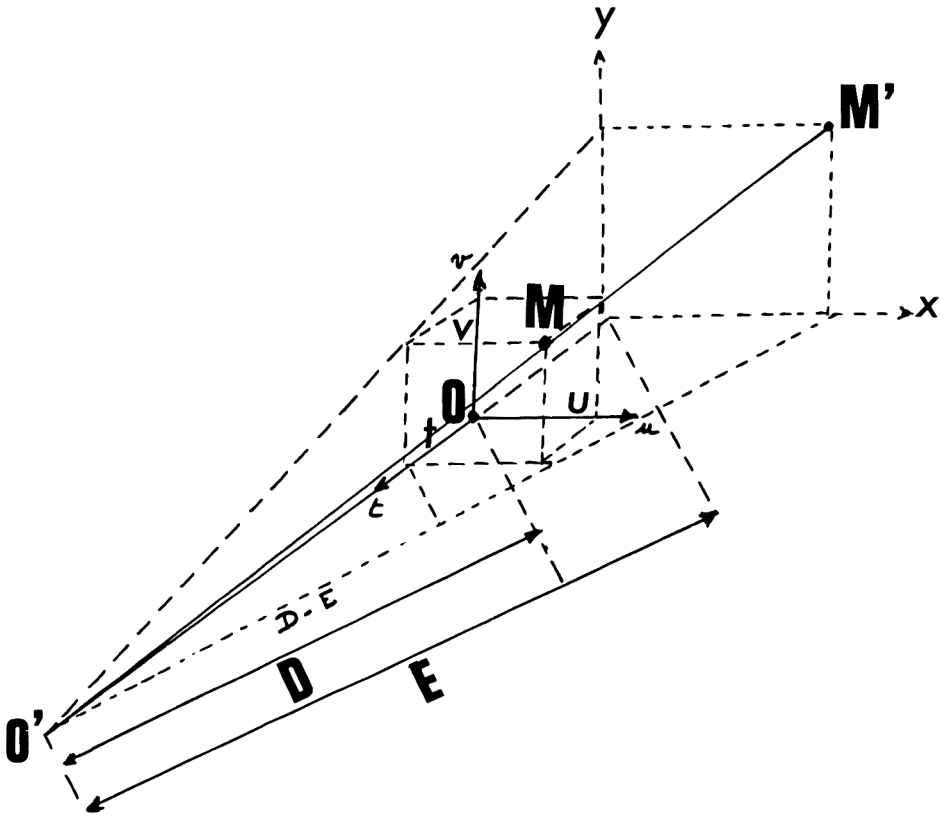

Z=ATN(T*U)
23 -3 3 3 -3


Z=ATN(T↑2-U↑2)
23 -3 3 3 -3

On pourrait varier ces exemples à l'infini. Il faut nous limiter. Parmi les variantes possibles, le changement de type de perspective s'obtient facilement : il suffit de changer les signes de AX et/ou AY. On peut effectuer de même le dessin de surfaces de révolution autour de l'axe vertical en prenant pour coupes des plans passant par l'axe de révolution.

4.4 Représentation d'un objet en perspective - Rotations

Cette fois, nous définissons un objet (polyèdre) par les coordonnées de ses sommets (en DATA) rapportées à un système de coordonnées liées au centre de l'objet Otuv. On a un observateur O' à la distance D de O sur l'axe Ot et un écran perpendiculaire à OO't à la distance E de O'. Nous nous excusons de ce fatras mathématique, mais il n'y a pas moyen de s'en passer. Le plus souvent on aura E=D, c'est-à-dire l'écran dans le plan uOv : la variation de E change la taille du dessin tandis que celle de D change le point de vue. En particulier, si l'on prend D petit, on crée un effet de grand angle qui exagère les perspectives.



Le point M' est la représentation perspective du point M de l'objet si M' est sur l'écran (qu'on appelle aussi plan de projection) et aligné avec O' et M. Ayant les coordonnées t, u et v de M, on a facilement celles de M' sur l'écran :

x=u.E/(D-t) y=v.E/(D-t)

d'où les véritables coordonnées écran pour l'AMSTRAD :

X=320+U*E/(D-T)
Y=200+V*E/(D-T)

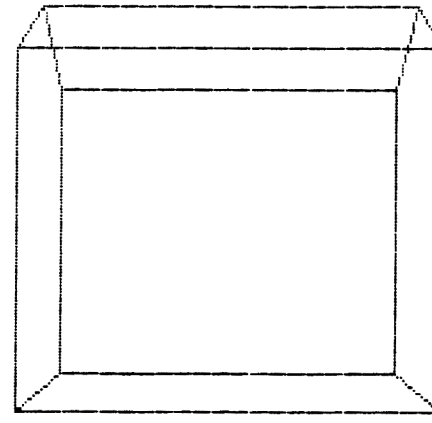
Le programme ci-dessous reçoit en DATA les coordonnées des NS sommets d'un polyèdre. La première donnée est NS. On donne ensuite les données des faces sous la forme nombre de faces puis, pour chaque face, nombre de sommets de cette face puis les numéros des sommets de cette face. Le programme gère pour cela trois tableaux de coordonnées de sommets (T, U et V) et le tableau des faces : F(I,J) est le numéro du sommet de la face I. F(I,0) est le nombre de sommets de cette face. Il faut noter que le nombre de sommets d'une face est automatiquement augmenté de 1 par le programme et le numéro du premier sommet de la face est répété à la fin pour assurer que la face soit dessinée fermée.

Le programme dessine ensuite l'objet, en prenant pour valeurs standard D=E=1000. Vous devez donner les coordonnées des sommets de sorte que leur dessin tienne dans l'écran. Ensuite, le programme attend des commandes de la forme E valeur ou D valeur et il recalcule les coordonnées en fonction des nouvelles valeurs au reçu de la touche *RETURN*. Il redessine l'objet au reçu de la touche G. La commande I fait copier le dessin sur l'imprimante. Rappelons que la routine d'impression est devenue un sous-programme grâce à la ligne 60110 RETURN .

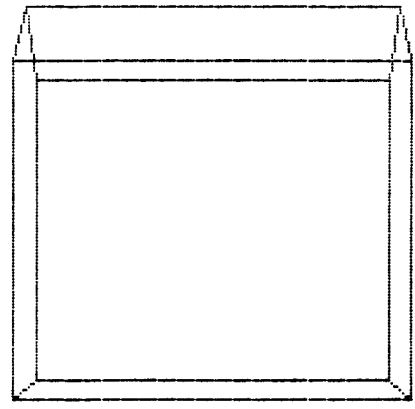
```
1 REM  FORME 1
2 REM
10 READ NS:DIM T(NS),U(NS),V(NS)
20 FOR I=1 TO NS:READ T(I),U(I),V(I):NEXT
30 READ NF:DIM F(NF,11):FOR I=1 TO NF
40 READ K:FOR J=1 TO K:READ F(I,J):NEXT
50 F(I,0)=K:F(I,K+1)=F(I,1):NEXT
60 D=1000:E=1000
70 DEF FNF(T)=320+U*E/(D-T):DEF FNG(T)=200+V*E/(
D-T)
80 GOSUB 600
90 FOR I=1 TO NF
100 K=F(I,0)
120 GOSUB 500 ' TRACE POLYGONE
130 NEXT
140 A$=INKEY$:IF A$="" GOTO 140
150 IF A$="E" THEN INPUT E:GOSUB 600:GOTO 140
160 IF A$="D" THEN INPUT D:GOSUB 600:GOTO 140
170 IF A$="I" THEN GOSUB 60000:GOTO 140
180 IF A$="G" GOTO 80
190 GOTO 140
```

```
500 FOR J=1 TO K:N1=F(I,J):N2=F(I,J+1)
510 T=T(N1):U=U(N1):V=V(N1):X1=FNF(T):Y1=FNG(T)
520 T=T(N2):U=U(N2):V=V(N2):X2=FNF(T):Y2=FNG(T)
530 PLOT X1,Y1,1:DRAW X2,Y2
540 NEXT: RETURN
600 CLS
610 PRINT "D E";D;E
620 RETURN
1000 DATA 10
1010 DATA 0,-140,140
1020 DATA 0,140,140
1030 DATA 120,-140,100
1040 DATA 120,140,100
1050 DATA -120,140,100
1060 DATA -120,-140,100
1070 DATA 120,-140,-100
1080 DATA 120,140,-100
1090 DATA -120,140,-100
1100 DATA -120,-140,-100
1110 DATA 7
1120 DATA 4,1,3,4,2
1130 DATA 4,1,6,5,2
1140 DATA 5,1,3,7,10,6
1150 DATA 5,2,4,8,9,5
1160 DATA 4,3,7,8,4
1170 DATA 4,6,10,9,5
1180 DATA 4,7,8,9,10
60000 REM Copie Ecran H-R DMP 2000 mode 1
```

```
D E 1000  1000
```



```
D E 2000  2000
```



Les DATA correspondent à la petite maison dessinée.

Rotations

Nous ajoutons maintenant une commande intéressante : la lettre T, U ou V suivie d'un angle en degrés (et *RETURN*) et l'objet tourne de l'angle indiqué autour de l'axe correspondant à la lettre!

En fait, c'est facile : les trois coordonnées d'un point après rotation s'obtiennent à partir des coordonnées avant rotation par multiplication par une matrice dite matrice de rotation. Par exemple, si R est l'angle, la matrice de rotation autour de l'axe V est :

$$\begin{matrix} \cos R & -\sin R & 0 \\ \sin R & \cos R & 0 \\ 0 & 0 & 1 \end{matrix}$$

et les autres sont à l'avenant.

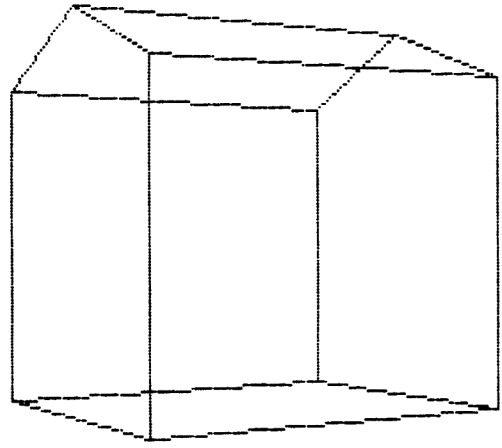
Le programme qui suit admet les commandes de rotation. Là encore, pour voir le résultat, il faut envoyer une commande G. On voit ci-dessous le résultat d'une rotation de 45° autour de l'axe vertical (on tape V45). Notez que si vous n'êtes pas satisfait d'une rotation, vous revenez à l'état antérieur en demandant la rotation opposée de même axe (Ex. V-45). Notez que l'angle doit être suivi de *RETURN*.

```
1 REM  FORME AVEC ROTATIONS
2 REM
10 READ NS:DIM T(NS),U(NS),V(NS)
20 FOR I=1 TO NS:READ T(I),U(I),V(I):NEXT
30 READ NF:DIM F(NF,11):FOR I=1 TO NF
40 READ K:FOR J=1 TO K:READ F(I,J):NEXT
50 F(I,0)=K:F(I,K+1)=F(I,1):NEXT
60 D=1000:E=1000
70 DEF FNF(T)=320+U*E/(D-T):DEF FNG(T)=200+V*E/(
D-T)
80 GOSUB 600
90 FOR I=1 TO NF
100 K=F(I,0)
120 GOSUB 500 ' TRACE POLYGONE
130 NEXT
140 A$=INKEY$:IF A$="" GOTO 140
150 IF A$="E" THEN INPUT E:GOSUB 600:GOTO 140
160 IF A$="D" THEN INPUT D:GOSUB 600:GOTO 140
170 IF A$="I" THEN GOSUB 60000:GOTO 140
180 IF A$="G" GOTO 80
190 IF A$="T" THEN GOSUB 350:GOTO 230
200 IF A$="U" THEN GOSUB 400:GOTO 230
210 IF A$="V" THEN GOSUB 450:GOTO 230
220 GOTO 140
```

```
230 GOSUB 300 : GOTO 140
300 REM ROTATION
310 INPUT R:R=PI*R/180
320 FOR I=1 TO NS:T=T(I):U=U(I):V=V(I)
330 T(I)=FNA(T):U(I)=FNB(T):V(I)=FNC(T)
340 NEXT : RETURN
350 REM MATRICE ROTATION T
360 DEF FNA(T)=T
370 DEF FNB(T)=U*COS(R)-V*SIN(R)
380 DEF FNC(T)=U*SIN(R)+V*COS(R)
390 RETURN
400 REM MATRICE ROTATION U
410 DEF FNA(T)=T*COS(R)+V*SIN(R)
420 DEF FNB(T)=U
430 DEF FNC(T)=-T*SIN(R)+V*COS(R)
440 RETURN
450 REM MATRICE ROTATION V
460 DEF FNA(T)=T*COS(R)-U*SIN(R)
470 DEF FNB(T)=T*SIN(R)+U*COS(R)
480 DEF FNC(T)=V
490 RETURN
500 FOR J=1 TO K:N1=F(I,J):N2=F(I,J+1)
510 T=T(N1):U=U(N1):V=V(N1):X1=FNF(T):Y1=FNG(T)
520 T=T(N2):U=U(N2):V=V(N2):X2=FNF(T):Y2=FNG(T)
530 PLOT X1,Y1,1:DRAW X2,Y2
540 NEXT: RETURN
600 CLS
610 PRINT "D E";D;E
620 RETURN
1000 DATA 10
1010 DATA 0,-140,140
1020 DATA 0,140,140
1030 DATA 120,-140,100
1040 DATA 120,140,100
1050 DATA -120,140,100
1060 DATA -120,-140,100
1070 DATA 120,-140,-100
1080 DATA 120,140,-100
1090 DATA -120,140,-100
1100 DATA -120,-140,-100
1110 DATA 7
1120 DATA 4,1,3,4,2
```

```
1130 DATA 4,1,6,5,2
1140 DATA 5,1,3,7,10,6
1150 DATA 5,2,4,8,9,5
1160 DATA 4,3,7,8,4
1170 DATA 4,6,10,9,5
1180 DATA 4,7,8,9,10
```

```
D E 1000  1000
```



Surfaces cachées

On aurait une meilleure compréhension du solide (surtout sous certains angles) si seules les faces visibles étaient dessinées. Les algorithmes sont plutôt complexes. Nous n'allons traiter le problème **que dans le cas d'un solide convexe**. C'est très restrictif, mais c'est le seul cas accessible dans le cadre d'un tel livre.

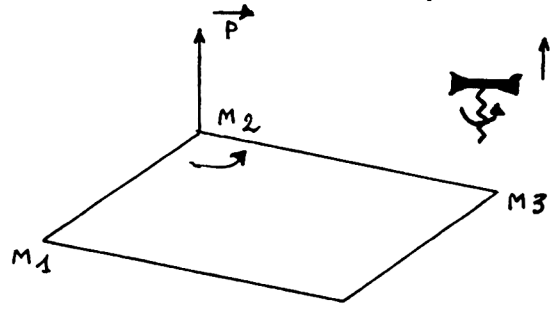
Pour un solide convexe, le problème se pose au niveau de chaque face : une face est visible ou non auquel cas elle sera tracée ou non. C'est une simplification : si le solide n'est pas convexe une même face peut avoir des morceaux visibles et d'autres cachés.

Soient les coordonnées tuv des sommets 1, 2 et 3 d'une face. Le vecteur P de composantes :

(u1-u2)(v3-v2)-(v1-v2)(u3-u2); (v1-v2)(t3-t2)-(t1-t2)(v3-v2);

(t1-t2)(u3-u2)-(t1-t2)(u3-u2);

est perpendiculaire à la face. Dans quel sens est-il orienté ? Considérez la rotation qui amène le vecteur M2M1 sur M2M3. Imaginez que c'est un tire-bouchon qui accomplit cette rotation : le vecteur P est orienté dans le sens où se déplacerait le tire-bouchon.



Maintenant, pour que ce vecteur P nous serve, il faut qu'il soit orienté vers l'extérieur du solide. Vous devez donc bien choisir les trois premiers sommets de chaque face.

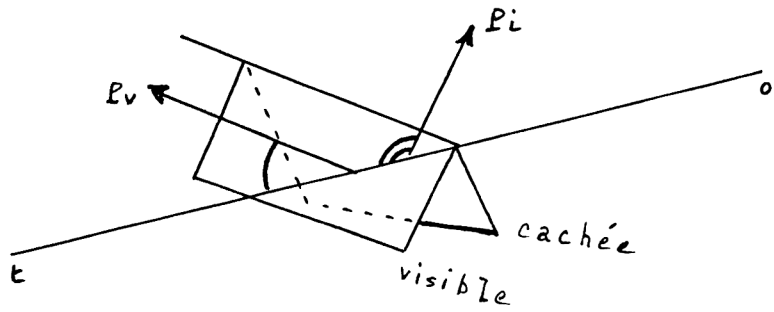
Exercice 4.10

Quels changements faire aux données du programme précédent pour assurer cette orientation.

Exercice 4.11

Définir de même les données d'un cube.

Ceci fait, regardons la figure suivante qui compare l'angle fait entre ce vecteur perpendiculaire extérieur et la direction Ot, qui est, rappelons-le, la direction du regard (au moins si D est assez grand, ce que nous supposerons).



On voit que si l'angle est aigu (composante sur t du vecteur normal positive), la face est visible. Si cette composante est négative, l'angle est obtus et la face est cachée. On a donc un critère de visibilité et, sous sa forme simplifiée, il ne nécessite que la première composante du vecteur.

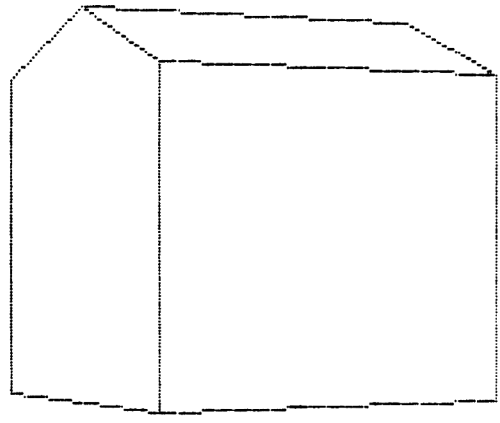
Le programme qui suit exploite ce critère.

```
1 REM  FORME AVEC ROTATIONS
2 REM
10 READ NS:DIM T(NS),U(NS),V(NS)
20 FOR I=1 TO NS:READ T(I),U(I),V(I):NEXT
30 READ NF:DIM F(NF,11):FOR I=1 TO NF
40 READ K:FOR J=1 TO K:READ F(I,J):NEXT
50 F(I,0)=K:F(I,K+1)=F(I,1):NEXT
60 D=1000:E=1000
70 DEF FNF(T)=320+U*E/(D-T):DEF FNG(T)=200+V*E/(
D-T)
80 GOSUB 600
90 FOR I=1 TO NF
100 K=F(I,0)
120 GOSUB 500 ' TRACE POLYGONE
130 NEXT
140 A$=INKEY$:IF A$="" GOTO 140
150 IF A$="E" THEN INPUT E:GOSUB 600:GOTO 140
160 IF A$="D" THEN INPUT D:GOSUB 600:GOTO 140
170 IF A$="I" THEN GOSUB 60000:GOTO 140
180 IF A$="G" GOTO 80
190 IF A$="T" THEN GOSUB 350:GOTO 230
200 IF A$="U" THEN GOSUB 400:GOTO 230
210 IF A$="V" THEN GOSUB 450:GOTO 230
220 GOTO 140
```

```
230 GOSUB 300 : GOTO 140
300 REM ROTATION
310 INPUT R:R=PI*R/180
320 FOR I=1 TO NS:T=T(I):U=U(I):V=V(I)
330 T(I)=FNA(T):U(I)=FNB(T):V(I)=FNC(T)
340 NEXT : RETURN
350 REM MATRICE ROTATION T
360 DEF FNA(T)=T
370 DEF FNB(T)=U*COS(R)-V*SIN(R)
380 DEF FNC(T)=U*SIN(R)+V*COS(R)
390 RETURN
400 REM MATRICE ROTATION U
410 DEF FNA(T)=T*COS(R)+V*SIN(R)
420 DEF FNB(T)=U
430 DEF FNC(T)=-T*SIN(R)+V*COS(R)
440 RETURN
450 REM MATRICE ROTATION V
460 DEF FNA(T)=T*COS(R)-U*SIN(R)
470 DEF FNB(T)=T*SIN(R)+U*COS(R)
480 DEF FNC(T)=V
490 RETURN
500 GOSUB 550:IF VI=0 THEN 540
505 FOR J=1 TO K:N1=F(I,J):N2=F(I,J+1)
510 T=T(N1):U=U(N1):V=V(N1):X1=FNF(T):Y1=FNG(T)
520 T=T(N2):U=U(N2):V=V(N2):X2=FNF(T):Y2=FNG(T)
530 PLOT X1,Y1,1:DRAW X2,Y2 :NEXT
540 RETURN
550 REM VISIBLE ?
560 N1=F(I,1):N2=F(I,2):N3=F(I,3)
570 U=U(N1)-U(N2):UU=U(N3)-U(N2)
580 V=V(N1)-V(N2):VV=V(N3)-V(N2)
590 UV=U*VV-V*UU:VI=UV>0:RETURN
600 CLS
610 PRINT "D E";D;E
620 RETURN
1000 DATA 10
1010 DATA 0,-140,140
1020 DATA 0,140,140
1030 DATA 120,-140,100
1040 DATA 120,140,100
1050 DATA -120,140,100
1060 DATA -120,-140,100
```

```
1070 DATA 120,-140,-100
1080 DATA 120,140,-100
1090 DATA -120,140,-100
1100 DATA -120,-140,-100
1110 DATA 7
1120 DATA 4,1,2,4,3
1130 DATA 4,1,6,5,2
1140 DATA 5,1,3,7,10,6
1150 DATA 5,2,5,9,8,4
1160 DATA 4,3,4,8,7
1170 DATA 4,6,10,9,5
1180 DATA 4,7,8,9,10
```

```
D E 2000  2000
```



Nous voilà au bout de notre périple graphique. Nous avons été obligés de laisser de côté beaucoup de détails et de simplifier au maximum les algorithmes utilisés. Toutefois, nous sommes arrivés à un ensemble de possibilités intéressantes. Bien sûr, on pourrait perfectionner : faire intervenir la couleur, hachurer les faces des solides, dessiner des éléments sur ces faces (ex. portes et fenêtres pour la maison). Nous laissons libre cours à votre imagination pour le faire. Autrement plus difficile serait de transformer notre programme de rotations de solides (déjà spectaculaire, vous le reconnaîtrez) en programme d'animation; cela nécessiterait sans doute d'accélérer les affichages et rendrait impératif le recours au langage machine.

CHAPITRE **5**

OBJETS GRAPHIQUES DÉPLAÇABLES OU LUTINS

Appelés en anglais "SPRITES", c'est-à-dire esprits ou fantômes, ces objets permettent des effets d'animation intéressants. On les appelle à peu près universellement "LUTINS" en français, ce qui insiste sur le caractère ludique de l'essentiel de leurs applications.

Un lutin est donc un petit dessin qu'il est très facile de déplacer à volonté sur l'écran. Mais ce qui est intéressant surtout, c'est que les lutins sont déplaçables **rapidement**, même en Basic.

Le problème est que, à la différence de quelques autres micros, le Basic de l'AMSTRAD ne possède pas d'instructions spécifiques pour gérer des lutins. Nous allons donc devoir les **simuler**. Cela ne devrait pas créer trop de difficultés car l'affichage d'un dessin sur l'écran n'est pas sorcier. Il y aura peut-être plus à réfléchir si on veut que le déplacement soit automatique...

5.1 Simulation de lutins

La base de notre simulation, nous l'avons déjà vue au chapitre 2 : l'affichage d'un petit dessin est rapide lorsque celui-ci est constitué d'une chaîne de caractères graphiques.

Nous examinerons successivement les problèmes suivants :

- définition (forme et couleurs),
- positionnement et déplacements,
- mouvements automatiques,
- déplacements devant ou derrière un décor,
- détection de collisions.

Définition de la forme

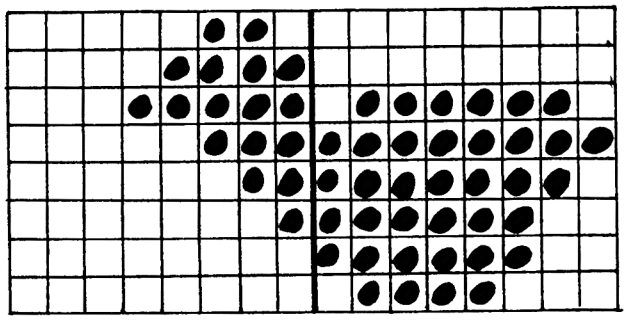
Pour la définition de la figure formée par le lutin souhaité, on n'est pas limité aux caractères semi-graphiques fournis par l'AMSTRAD : on peut, avec le couple SYMBOL / SYMBOL AFTER, redéfinir les caractères que l'on veut et leur donner toute forme que l'on peut souhaiter. Nous renvoyons le Lecteur à "La Découverte de l'AMSTRAD" pour une étude détaillée de ces instructions.

En outre, on a vu au chapitre 2 que le lutin peut être formé de plusieurs caractères juxtaposés. La juxtaposition sur une ligne ne pose pas de problème. Pour un lutin formé de plusieurs lignes, on sait qu'on peut incorporer à la chaîne de caractères les caractères de contrôle qui effectuent les mouvements de curseur voulus. Mais il y a un problème en liaison avec le positionnement : si l'on se contente d'un positionnement en mailles caractères (avec LOCATE), pas de problème, on l'a vu pour le mouton du chapitre 2. En revanche, si l'on veut un positionnement au pixel près (par TAG) - et c'est nécessaire pour avoir des mouvements continus - alors il faut que le dessin soit uniligne car, avec TAG, les caractères de contrôle font afficher des dessins de flèches au lieu d'être exécutés.

De même, certains caractères de contrôle permettent de changer de couleur : ils permettraient donc d'avoir des lutins multicolores, mais il y a la même limitation vis-à-vis du positionnement par TAG.

Exercice 5.1

Définir une variable A$ qui permette d'imprimer où on veut un dessin en forme de pigeon.



Positionnement du lutin

Pour afficher un lutin dont la définition est contenue dans la chaîne A$, il suffit de faire PRINT A$. Mais où sera-t'il affiché ? Il y a un choix très simple.

- Ou bien on se contente d'un positionnement en mailles caractères, et il suffit de LOCATE K,L puis PRINT pour avoir le coin supérieur gauche en ligne L, colonne K. Mais la résolution est alors de 25 lignes sur 20,40 ou 80 colonnes selon le mode.

- On peut avoir un positionnement pixel par pixel, donc 200 sur 160, 320 ou 640. Il faut, pour cela, utiliser TAG, puis MOVE X,Y et PRINT pour avoir le coin supérieur gauche du dessin en X,Y.

Cette dernière possibilité a malheureusement un inconvénient qu'elle hérite du fonctionnement de TAG : comme en mode TAG, les caractères de contrôle apparaissent sous forme de dessins, la chaîne de caractères de définition du lutin ne doit pas en contenir, ce qui exclut les changements de couleur ou les mouvements de curseur (un lutin de plusieurs lignes). Si cette contrainte est acceptable, le positionnement avec TAG sera préféré car il permet des déplacements beaucoup moins saccadés, encore que les saccades soient peu gênantes si les déplacements sont rapides. LOCATE permet des mouvements plus rapides.

5.2 Déplacements

Déplacer un lutin, c'est tout simplement :

1- l'afficher au point de départ,
2- attendre un certain délai,
3- effacer le lutin de sa position de départ,
4- le réafficher à la position d'arrivée.

Toutes ces opérations sont faciles; c'est peut-être l'**effacement** qui est le plus délicat, et encore... Sinon, les points 1 et 4, nous les avons déjà vus. Le délai en 2 a pour utilité de permettre de voir le lutin. Il peut en partie être mis à profit pour calculer les nouvelles coordonnées. La combinaison de la valeur du délai et de la distance de déplacement sert à règler la vitesse du mouvement.

Avec LOCATE, il faut spécifier les nouvelles coordonnées en absolu. En mode TAG, on a le choix entre coordonnées absolues ou relatives (c'est donc les coordonnées du vecteur déplacement qu'on donnerait) selon qu'on utilise MOVE ou MOVER.

Pour le délai, on peut utiliser bêtement une petite boucle vide : FOR T=1 TO N : NEXT où N est le nombre de 1/2000è de seconde qu'on veut laisser passer.

Une autre technique est d'utiliser TIME, fonction sans argument qui augmente de 1 tous les 1/300 èmes de seconde :

```
100 T=TIME
110 IF (TIME-T)<NN THEN 110
```

où NN est le nombre de trois-centièmes de seconde à attendre.

Enfin, on peut utiliser AFTER :

AFTER N,C GOSUB llll attend N cinquantièmes de seconde au chrono n° C et appelle le sous-programme ligne llll.

(Remarquons qu'il y a aussi EVERY qui nous servira pour les déplacements automatiques).

L'effacement

L'effacement ne pose pas de problèmes si le lutin est à déplacer sur un simple fond d'écran : pour l'effacer, il suffit de le redessiner dans l'encre n° 0. Donc, sur 664/6128, on fait un GRAPHICS PEN 0 et, sur 464, on fait un PLOT X,Y,0.

Mais il y a une technique encore plus spectaculaire : c'est d'afficher le lutin en mode de tracé **ou exclusif**. A ce moment, il suffit de réafficher le lutin sur place pour qu'il soit effacé. Pour imposer le mode, c'est facile sur 664/6128 : il suffit de mettre un 4ème paramètre dans un MOVE ou PLOT. Sur 464, c'est plus délicat : il faut faire PRINT CHR$(23)+CHR$(1) mais alors il ne faut pas être en mode TAG. Donc, ou bien on impose le mode tout au début, ou bien on fait TAGOFF, PRINT... et enfin TAG.

Exercice 5.2

Définissez un lutin en forme de pigeon comme à l'exercice 5.1. Faites-le se déplacer le long d'une horizontale dans la partie supérieure de l'écran.

Déplacement automatique

Mais l'AMSTRAD nous permet de faire mieux : il nous permet de faire déplacer un lutin (ou plusieurs) automatiquement, c'est-à-dire sans que le programme ait à s'en occuper, ce qui lui permet de faire autre chose en même temps.

On utilise pour cela les instructions de gestion d'événements. On peut, pour certains événements, envoyer l'AMSTRAD à un sous-programme de traitement de l'événement. Les événements concernés peuvent être l'appui sur certaines touches (Ex. ON BREAK GOSUB ... qui capte les appuis sur la touche *ESC*) ou l'épuisement de certains délais. L'instruction EVERY que nous allons utiliser appartient à cette catégorie.

Elle est de la forme EVERY N,C GOSUB llll où C (de 0 à 3) est le numéro de chrono utilisé (le chrono est d'autant plus prioritaire que son n° est plus grand). Lorsque cette instruction a été exécutée, tous

les N cinquantièmes de seconde, on appelle le sous-programme qui commence en ligne 1111. Le retour se fait à l'instruction qu'exécutait l'AMSTRAD lorsqu'il a été interrompu.

Dans l'exemple ci-dessous, le pigeon se déplace sur l'écran et il est stoppé si on appuie sur une touche : on assure l'indépendance entre la partie du programme qui fait déplacer le pigeon (le sous-programme de traitement de EVERY) et la boucle d'attente d'une touche.

Un autre appui de touche fait reprendre le mouvement. On remarquera en 80 l'appel de la fonction REMAIN(C) qui fournit le temps restant à décompter dans le chrono C, mais surtout arrête le chrono et donc, ici, arrête le déplacement.

```
1 REM  DEPLACEMENT AUTOMATIQUE
2 REM
10 SYMBOL 241,6,15,31,7,3,1,0,0
20 SYMBOL 242,0,0,126,255,254,252,252,56
30 A$=CHR$(241)+CHR$(242)
40 PRINT CHR$(23)+CHR$(1) : TAG :CLG
50 X=0:Y=390:PLOT X,Y,1:PRINT A$;
60 EVERY 2,1 GOSUB 500
70 X$=INKEY$ : IF X$="" GOTO 70
80 Z=REMAIN(1)
90 X$=INKEY$ : IF X$="" GOTO 90
100 GOTO 60
500 REM S/P DE DEPLACEMENT
501 REM
510 MOVE X,Y:PRINT A$;
520 X=X+7:IF X>=600 THEN X=0
530 MOVE X,Y:PRINT A$;
540 RETURN
```

Exercice 5.3

On fait déplacer un lutin de gauche à droite sur l'écran et quand il a atteint le bord droit, on inverse le mouvement; on inverse à nouveau lorsqu'il atteint le bord gauche.

En règlant bien la vitesse, vous pouvez avoir un véritable

hypnotiseur. Si vous définissez un second lutin, vous pouvez faire que votre mobile se retourne lorsqu'il change de sens de déplacement.

Exercice 5.4

Faites-le en prenant toujours notre lutin en forme de pigeon.

Déplacement dans un décor

Que se passe-t'il lorsque le lutin se déplace sur un écran où il y a déjà des dessins ? On dit que l'on a un décor. Notre astuce d'utiliser le mode de tracé "ou exclusif" résout automatiquement une bonne partie des problèmes : le deuxième affichage n'efface pas seulement le lutin lorsque celui-ci était affiché sur le fond; en fait, il résulte des propriétés de l'opérateur ou exclusif que tout objet qui était dessiné à l'emplacement du lutin se trouvera rétabli.

On peut dans beaucoup de cas se contenter de cela, en particulier pour des mouvements rapides. Dans d'autres cas, ce n'est pas suffisant. Lorsque le lutin se superpose à un objet, il y a combinaison de couleurs. Ce qu'on souhaiterait plutôt, c'est de créer des effets de passage du lutin soit **devant**, soit **derrière** l'objet.

Disons tout de suite que, pour être fait proprement, ceci exige le recours au langage machine et, donc, sort du cadre de ce livre.

Détection de collisions

Une autre possibilité offerte par les Basic qui ont des instructions de lutins, c'est le déclenchement automatique d'événement lorsque l'on décèle que deux lutins se sont heurtés ou qu'un lutin a heurté un objet sur l'écran. C'est facile à simuler dès lors que l'on conserve dans des variables les positions des objets sur l'écran. Il suffit d'incorporer à chaque mouvement (donc dans le sous-programme de déplacement) des tests sur les coordonnées.

Nous sommes maintenant en mesure de traiter très simplement un exemple assez spectaculaire, le tir au pigeon. On définit un lutin en forme de pigeon qui se déplacera de gauche à droite en haut de l'écran

et un lutin en forme de projectile qui se déplacera de bas en haut au milieu de l'écran : c'est la balle. Le tir est déclenché par appui sur une touche du clavier. Le score est affiché en permanence.

```
1 REM  TIR AU PIGEON
2 REM
10 SYMBOL 241,6,15,31,7,3,1,0,0
15 SYMBOL 243,0,0,126,255,127,63,63,30
20 SYMBOL 242,0,0,126,255,254,252,252,56
25 SYMBOL 244,96,240,248,224,192,128,0,0
30 A1$=CHR$(241)+CHR$(242):A2$=CHR$(243)+CHR$(24
4)
40 PRINT CHR$(23)+CHR$(1) : TAG :CLG
50 A$=A2$:D=7:X=0:Y=390:PLOT X,Y,1:PRINT A$;
60 EVERY 3,1 GOSUB 500
70 B$=CHR$(231):XB=300:YB=10:MOVE XB,YB:PRINT B$
;
80 SC=0:MOVE 50,200:PRINT SC;
90 X$=INKEY$ : IF X$="" GOTO 90
100 B=1
110 GOTO 90
500 REM S/P DE DEPLACEMENT
501 REM
510 MOVE X,Y:PRINT A$;
520 X=X+D:IF X>=600 THEN D=-D:A$=A1$
525 IF X<ABS(D) THEN D=-D:A$=A2$
530 MOVE X,Y:PRINT A$;
540 IF B=0 THEN 650
600 REM MVT BALLE
610 MOVE XB,YB:PRINT B$;
620 YB=YB+15:MOVE XB,YB:PRINT B$;:IF YB<390 GOTO
 650
630 B=0:MOVE XB,YB:PRINT B$;:YB=10:MOVE XB,YB:PR
INT B$;
640 IF X>=300 AND X<320 THEN MOVE 50,200:PRINT S
C;:SC=SC+1:MOVE 50,200:PRINT SC;
650 RETURN
```

CHAPITRE **6**

LES SONS

Nous allons maintenant placer notre AMSTRAD sous le patronage d'Euterpe. L'AMSTRAD a en effet des possibilités sonores très étendues produites par le même circuit intégré sonore que les MSX et l'ORIC. Rappelons qu'elles consistent d'une part dans la possibilité de faire retentir simultanément trois voix sur une étendue de 8 octaves mais aussi dans la possibilité de définir la variation de l'intensité du son au cours du temps, ce qui permet de varier les attaques et les résonances, donc permet d'imiter certains instruments. On a aussi la possibilité de définir la variation de la hauteur du son au cours du temps, ce qui permet des effets tels que le vibrato.

Nous allons tout d'abord examiner quelques propriétés générales des sons, afin de mieux comprendre les possibilités offertes par l'AMSTRAD. Puis nous étudierons les instructions de gestion des sons que possède l'AMSTRAD et, enfin, nous regarderons plus particulièrement les problèmes de traduction d'une partition.

6.1 Propriétés des sons

A part la durée, les deux propriétés les plus fondamentales d'un son - et aussi les plus familières à tout un chacun - sont l'intensité (ou volume) et la hauteur (la note jouée). Comme on le sait, le son correspond à une vibration transmise par l'air ambiant. L'intensité correspond à l'**amplitude** de cette vibration; elle est liée à l'énergie mise en jeu. La hauteur correspond à la **fréquence** (=nombre de

périodes par seconde) de cette vibration; si le son est haut ou aigu, la fréquence est grande; si le son est bas ou grave, la fréquence est petite.

Mais ces deux propriétés ne suffisent pas à caractériser complètement un son; en particulier, elles ne permettent pas du tout de dire quel instrument est en train de jouer; or, vous êtes capables de reconnaître les instruments (au moins quelques-uns) à l'oreille, c'est donc que le son véhicule l'information correspondante. Cette information est portée concurremment par deux paramètres, le timbre et l'enveloppe.

Le timbre

On obtient rarement un son pur (son dont la vibration ne présente qu'une seule fréquence) mais presque toujours un son formé d'un ensemble de fréquences. Cet assortiment de fréquences détermine la "couleur" du son ou, comme on dit, son timbre. La fréquence présente la plus basse s'appelle le fondamental : elle détermine la fréquence apparente du son entendu puisqu'elle détermine la période du son. Si les autres fréquences présentes sont des multiples du fondamental, on obtient un son musical; si les autres fréquences n'ont pas de rapport simple avec le fondamental, on obtient un *bruit*.

Le timbre définit la forme du signal tel qu'on pourrait par exemple l'observer sur un oscilloscope.

A part le choix bruit/musique, **le timbre n'est pas commandable sur l'AMSTRAD**. L'AMSTRAD produit un timbre un peu passe-partout qui s'accorde avec la plupart des instruments usuels.

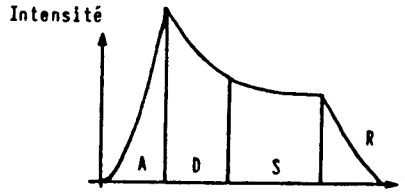
L'enveloppe

Mais le timbre ne suffit pas à lui seul pour la reconnaissance d'un instrument de musique. Par exemple, et cela peut sembler paradoxal, le violon, la guitare et le piano n'ont pas des timbres très différents : dans tous les cas ce sont des cordes qui vibrent. Il y a donc un élément supplémentaire qui intervient, l'enveloppe.

Lorsqu'on joue une note, son intensité varie au cours du temps : en

gros, elle monte au commencement de la note et elle décroît à la fin. L'enveloppe est la courbe qui résume l'historique de l'intensité d'une note tout au long de sa durée.

Toute enveloppe comprend quatre phases, certaines pouvant être inexistantes parce que trop brèves. Elles sont résumées par la figure ci-dessous :



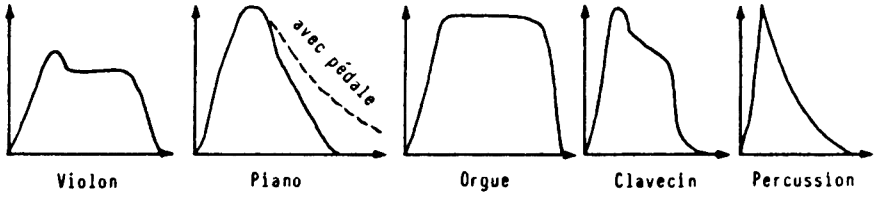
A est l'**attaque** ou montée du son : le paramètre qu'on spécifie est la durée de cette montée, plus ou moins brusque. La hauteur du sommet est, elle, déterminée par le volume.

D est la **descente**, exprimée elle aussi sous forme de durée.

S est le **soutien**. Le son reste le plus longtemps sur un palier dont le paramètre spécifie la hauteur.

R est la **relaxation**, qui marque le retour du son à zéro. Ici aussi, c'est la durée qu'on spécifie.

Voici quelques exemples d'enveloppes d'instruments familiers :



Nous avons maintenant vu les propriétés de base des sons. L'AMSTRAD permet d'autres effets en combinant différents sons. D'abord, il peut faire retentir trois sons simultanément : il a en effet trois voix sonores programmables indépendamment. Mais même sur une voix donnée, on peut combiner les effets de sons successifs pour produire des vibratos ou des effets de sirène.

6.2 Les instructions sonores de l'AMSTRAD

Nous voyons maintenant les instructions de manipulation des sons offertes par le Basic de l'AMSTRAD. Avant cela, nous rappelons que l'AMSTRAD produit les sons sur son haut-parleur incorporé. Le volume est commandé par la molette située à côté de l'interrupteur de mise sous tension. Mais, pour une meilleure qualité sonore, vous pouvez le connecter à une chaîne Hi-Fi; vous aurez alors des effets stéréo, le canal gauche recevant un mélange des voix 1 et 2, le canal de droite recevant un mélange des voix 3 et 2. (N.B. Vous pouvez vous contenter de connecter un couple de petites enceintes amplifiées du type utilisé couramment avec les "Walkman" : vous aurez déjà une amélioration substantielle des sons obtenus).

Nous citons pour mémoire le fait que PRINT CHR$(7) ou PRINT " 'CTRL' G " produit un bip.

SOUND

L'instruction principale de production d'un son sur l'AMSTRAD est SOUND. Comme les paramètres sont nombreux et ont des options par défaut, elle revêt des formes diverses. Voici la forme minimale :

SOUND voix,hauteur

où voix détermine le numéro de l'une des trois voies (ou voix!! ou canaux) que l'on veut faire retentir et hauteur est un nombre compris entre 1 et 4095 proportionnel à la période de la note à jouer. La période est l'inverse de la fréquence donc plus la note est haute, plus la valeur est petite. On peut spécifier 0 pour les effets de bruits.

Les valeurs des notes sont précisées par le tableau suivant. Par exemple SOUND 1,142 fait retentir un la.

Tableau 6.1 Valeurs des notes

Note	Valeur	Note	Valeur	Note	Valeur	Note	Valeur
do -1	3822	do 0	1911	do 1	956	do 2	478
do#	3608	do#	1804	do#	902	do#	451
ré	3405	ré	1703	ré	851	ré	426
ré#	3214	ré#	1607	ré#	804	ré#	402
mi	3034	mi	1517	mi	758	mi	379
fa	2863	fa	1432	fa	716	fa	358
fa#	2703	fa#	1351	fa#	676	fa#	338
sol	2551	sol	1276	sol	638	sol	319
sol#	2408	sol#	1204	sol#	602	sol#	301
la	2273	la	1136	la	568	la	284
la#	2145	la#	1073	la#	536	la#	268
si	2025	si	1012	si	506	si	253
do 3	239	do 4	119	do 5	60	do 6	30
do#	225	do#	113	do#	56	do#	28
ré	213	ré	106	ré	53	ré	27
ré#	201	ré#	100	ré#	50	ré#	25
mi	190	mi	95	mi	47	mi	24
fa	179	fa	89	fa	45	fa	22
fa#	169	fa#	84	fa#	42	fa#	21
sol	159	sol	80	sol	40	sol	20
sol#	150	sol#	75	sol#	38	sol#	19
la	142	la	71	la	36	la	18
la#	134	la#	67	la#	34	la#	17
si	127	si	63	si	32	si	16

On a indiqué les numéros des octaves. Le do du milieu du piano et le la du diapason sont dans l'octave 3. Bien sûr, les valeurs sont approximatives surtout dans l'octave la plus aiguë.

On trouvera à l'annexe III la représentation des principales notes sur partition. Notons aussi que les valeurs de période à indiquer dans SOUND correspondent approximativement à la formule :

hauteur(SOUND) = 62500/fréquence

Une forme plus complète est :

SOUND voix,hauteur,durée,volume

où durée indique la durée de la note en centièmes de seconde (valeur par défaut 20 soit 0,2 s). Une durée 0 signifie durée déterminée par l'enveloppe. Une valeur négative (-n) signifie "répéter l'enveloppe n fois". Volume (de 0 - silence - à 15 - le plus fort -) définit le volume du son. La valeur par défaut est 12.

La forme complète est :

SOUND voix,hauteur,durée,volume,enveloppe,modulation,bruit

où enveloppe est un numéro d'enveloppe qui définira la variation d'amplitude du son au cours de son déroulement. Une enveloppe est définie par une instruction ENV. Modulation est aussi un numéro d'enveloppe, mais faisant varier la hauteur du son pour produire, par exemple, du vibrato. Une modulation est définie par une instruction ENT. SOUND fait référence à un numéro d'enveloppe ou de modulation et on peut se constituer un jeu d'enveloppes ou de modulations par autant d'instructions ENV et ENT (au plus 15 de chaque). Bruit est une période qui permet d'ajouter des effets de bruit. Si hauteur<>0 et bruit=0, on a un son pur; si hauteur=0 et bruit<>0, on a un bruit pur et si les deux sont non nuls on a un mélange son-bruit.

Les files d'attente - RELEASE

Une propriété essentielle de l'instruction SOUND est qu'elle lance le son et redonne immédiatement le contrôle à l'utilisateur. Cela permet de lancer une autre note sur une autre voix et, ainsi, de produire un accord.

Bien entendu, si vous envoyez une deuxième instruction SOUND sur la même voix, le son attendra que le premier son soit terminé pour démarrer. Mais le contrôle vous sera tout de même rendu : les sons envoyés à une voix sont mis en file d'attente: il peut y avoir jusqu'à 5 sons en attente sur une voix et, bien sûr, si la file d'attente est pleine, une nouvelle instruction SOUND sur la voix considérée met le programme en attente.

Deux autres phénomènes d'attente peuvent intervenir. Lorsqu'on spécifie un son sur un canal (voix), on peut spécifier un "rendez-vous" avec un autre canal, c'est-à-dire qu'il démarre lorsque le prochain son du canal de rendez-vous démarrera. C'est ce qui permet des accords bien simultanés.

Enfin, on peut spécifier que le son considéré soit en attente et qu'il ne démarre que lorsque l'on exécutera une instruction RELEASE voix sur la combinaison de canaux indiquée.

Le paramètre 'voix'

Il nous faut voir maintenant comment on spécifie le canal. Le paramètre 'voix' spécifie en fait une combinaison de canaux. Le paramètre 'voix' est un octet dont chaque bit a une signification précise obéissant au schéma suivant :

Tableau 6.2 Le paramètre 'voix'

bit	valeur	fonction
0	1	jouer sur voix 1
1	2	jouer sur voix 2
2	4	jouer sur voix 3
3	8	rendez-vous avec voix 1
4	16	rendez-vous avec voix 2
5	32	rendez-vous avec voix 3
6	64	mise en attente
7	128	vidage de la file d'attente de la voix

Les valeurs de la 2è colonne sont intéressantes. Pour activer un certain ensemble de fonctions, il suffit de spécifier la somme des valeurs correspondantes.

Exemples :
Pour envoyer la même note sur les canaux 1 et 3, comme 1+4=5, il suffit de faire SOUND 5, ...
Pour envoyer une note sur la voix 1 avec rendez-vous avec la voix 2,

faire SOUND 17, ...
Note sur la voix 2, avec attente : SOUND 66, ...
Libération des attentes sur les trois canaux (ce qui permet de démarrer un accord) : RELEASE 7.

La fonction 128 vide la(les) file(s) d'attente de la (des) voix spécifiée(s) et donc SOUND 135, ... arrête tous les sons. On peut le faire aussi par PRINT " 'CTRL' G ".

Bien sûr, les paramètres peuvent être fournis sous forme d'expressions arithmétiques ou de valeurs lues en DATA ou de toute autre manière favorable.

Lorsque l'on spécifie deux notes successives identiques sur une voix, l'AMSTRAD les enchaîne de sorte qu'elles n'en forment qu'une seule, de durée égale à la somme des durées :

SOUND 1,142,100 : SOUND 1,142,100 est équivalent à :

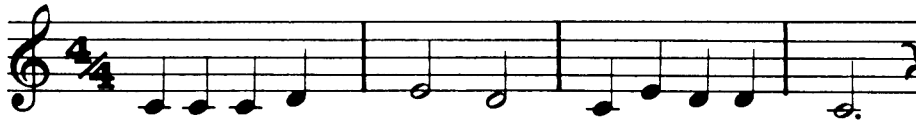
SOUND 1,142,200 . Si on veut deux sons séparés, il faut intercaler un court silence, obtenu, par exemple, avec un volume nul :

SOUND 1,142,100 : SOUND 1,142,10,0 : SOUND 1,142,100 .

Une autre façon serait d'utiliser une enveloppe se terminant par une petite portion silencieuse.

Exercice 6.1

Jouer "Au clair de la Lune".



(dododorémi,ré,domirérédo)

(nous anticipons un peu sur la section "traduction d'une partition").

Exercice 6.2

Ecrivez un programme qui vous permette d'accorder votre guitare (mi 660, si 495, sol 393, ré 294, la 220, mi 165).

Exercice 6.3

Produire l'accord DO MI SOL.

Les enveloppes : ENV

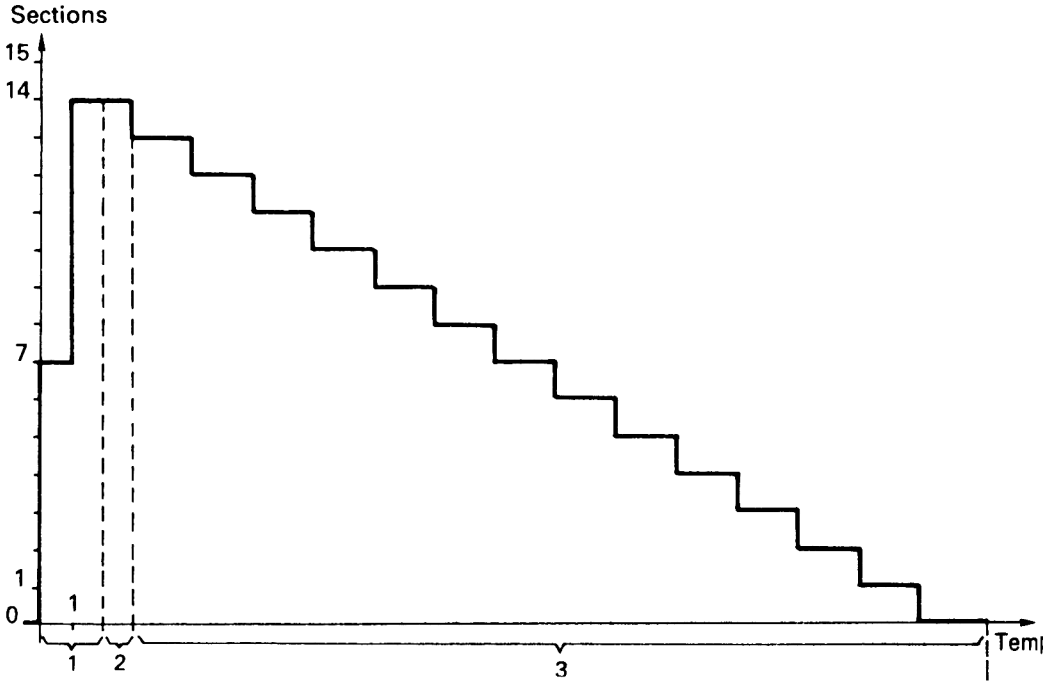
On a vu comment on spécifiait l'usage d'une enveloppe dans l'instruction SOUND. L'enveloppe doit préalablement avoir été définie par une instruction ENV de la forme :

ENV enveloppe,sections

où enveloppe est le numéro d'enveloppe tel qu'il est référencé dans une instruction SOUND. Il peut y avoir au plus 5 sections de variation, qui sont chacune de la forme :

nombre de pas,valeur du pas,temps

La valeur du pas indique de combien d'unités de volume on augmente ou on diminue. Le temps de chaque pas est en centièmes de secondes. Voici une enveloppe de clavecin :



Elle est produite par ENV 1,2,7,1,1,0,1,14,-1,2. Pour l'expérimenter, faites SOUND 1,142,0,0,1. SOUND 1,142,-3,0,1 fait retentir la note 3 fois.

Cette enveloppe a trois sections : une montée rapide, un court palier (1 pas à variation nulle) et une descente plus lente. Si vous changez le dernier 2 en 1, vous avez le mode étouffé du clavecin. Si vous le changez en 4, le son ressemble à celui du piano.

Le volume spécifié dans SOUND sert d'amplitude de départ. Si dans une section, le volume est amené en dessous de 0, il reprend à 15 et s'il est amené au dessus de 15, il reprend à 0; on obtient donc des dents de scie.

La relation avec la durée spécifiée dans SOUND est complexe. De toutes façons, l'enveloppe a une durée bien déterminée, somme des durées des sections. La durée de chaque section est:nombre de pas x temps de chaque pas. Si la durée spécifiée est 0, alors la durée du son sera celle de l'enveloppe. Si la durée spécifiée est négative (-n),

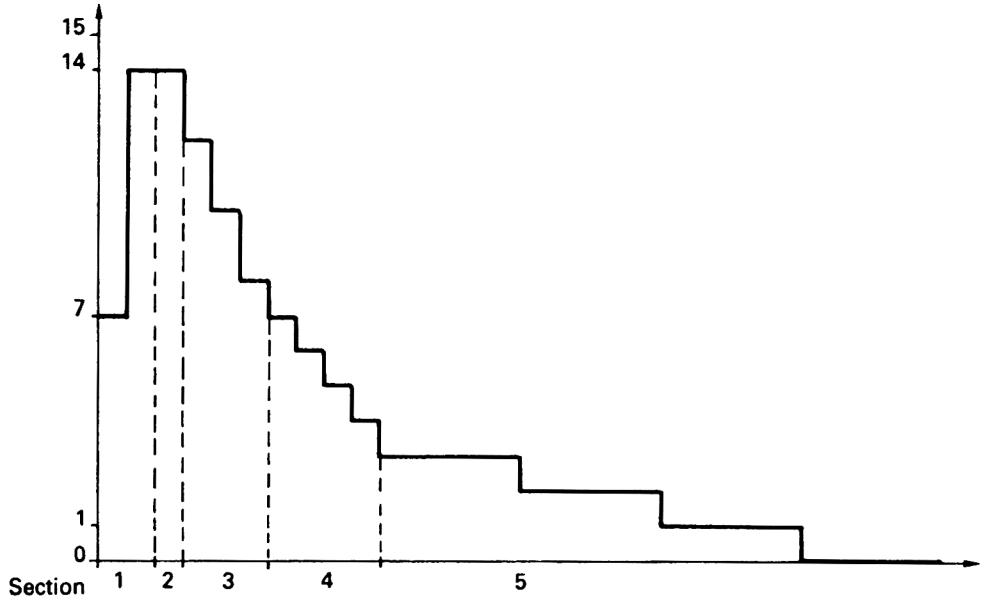
l'enveloppe sera répétée n fois, donc la durée du son sera n fois la durée de l'enveloppe. Si la durée spécifiée est positive, soit d1, d2 étant la durée de l'enveloppe, il y a deux cas :

1- d1<d2 La durée du son sera d1, c'est-à-dire que l'enveloppe sera interrompue en cours de déroulement;

2- d1>d2 La durée du son sera d1, les d2 premiers centièmes de seconde suivant l'enveloppe. Le son se terminera par d1-d2 centièmes de seconde faits avec une amplitude constante égale à la valeur atteinte en fin d'enveloppe.

Exercice 6.4

Produire l'enveloppe figure ci-dessous (essai de simuler une descente exponentielle).



ENT

Les modulations sont à la hauteur ce que les enveloppes sont à

l'amplitude. D'ailleurs, AMSTRAD les appelle "enveloppes de ton". L'instruction de spécification de modulation ENT a exactement la même forme que ENV :

ENT modulation,sections

où le premier paramètre est le numéro de modulation qui sera référencé dans SOUND. Comme pour ENV, on a au plus 5 sections de la forme : nombre de pas, variation, temps. Le temps est en centièmes de seconde et la variation (positive ou négative) s'exprime en unités de période.

La modulation a elle aussi une durée, mais sa relation avec la durée du son est différente de celle de l'enveloppe. Soit d la durée résultant de l'enveloppe et de la durée spécifiée dans SOUND, et d3 la durée de la modulation.

Si d3>d, la durée du son sera d et il y aura une partie de la modulation inutilisée.

Si d3<d, la durée du son sera d avec une partie modulée de durée d3 puis d-d3 centièmes de seconde avec hauteur constante égale à la valeur atteinte en fin de modulation.

Il y a un cas particulier lorsque d3<d : si, dans ENT, on a spécifié le numéro de modulation négatif (-n, attention, il doit quand-même être spécifié positif : n, dans SOUND), alors la modulation est périodique, c'est-à-dire qu'elle est répétée jusqu'à écoulement de la durée d.

Ce dernier cas est idéal pour le vibrato; essayez :

```
ENT -1,4,1,1,8,-1,1,4,1,1
SOUND 1,142,1000,15,,1
```

Remarquez les virgules consécutives dans SOUND qui marquent l'absence de spécification d'enveloppe. Naturellement, l'amplitude de variation doit être adaptée aux notes jouées : ici, à partir du la, on va à mi-chemin du sol dièse et du la dièse. En outre, la vitesse de vibrato doit être adaptée au tempo du morceau.

Exercice d'entraînement 6.5

Reconstituez le schéma du vibrato précédent. Modifiez l'instruction pour qu'il se fasse deux fois plus lentement.

Exercice 6.6

Faire retentir une sirène.

LES BRUITS

Le dernier paramètre de l'instruction SOUND permet de produire des bruits, c'est-à-dire des mélanges aléatoires de fréquences situées autour d'une fréquence centrale dont le dernier paramètre (qui doit être compris entre 1 et 31) est proportionnel à la période.

Le programme ci-dessous imite un coup de feu.

```
1 REM  Coup de feu
2 REM
10 ENV 1,2,7,1,1,0,1,3,-2,1,4,-1,1,4,-1,4
20 SOUND 1,0,0,0,1,,10
```

On a adopté l'enveloppe "clavecin à décroissance exponentielle".

Nous vous conseillons d'expérimenter : changez l'enveloppe; changez la fréquence centrale du bruit. On peut tabler sur les valeurs suivantes (elles doivent être comprises entre 1 et 31) :

1	sifflements
2, 3, 4	réacteurs, fuites de gaz
5 - 8	tempête, lance-flammes
9 - 16	vent
17 - 22	machines, explosion
23 - 31	avions, grondements.

Les enveloppes périodiques peuvent donner de bons effets, essayez

le programme ci-dessous :

```
1 REM  Moteur de rafiot
2 REM
10 ENV 1,1,15,1,3,-5,2,50,0,50
20 SOUND 1,0,10,0,1,,,10
30 GOTO 20
```

Si besoin est, pour arrêter le bruit, faites :

PRINT " 'CTRL' G ".

SQ

La fonction SQ (voix) fournit l'état du canal indiqué, les bits ayant les rôles suivants :

bits 0 à 2	nombre de places libres dans la file d'attente
bits 3, 4 ou 5	rendez-vous attendu avec la voix 1, 2 ou 3
bit 6	ce canal est en attente
bit 7	ce canal est en train de jouer.

Par exemple, IF SQ(1)<128 THEN SOUND 1, ... renvoie un son sur la voix 1 dès qu'elle ne joue plus.

La voix se spécifie comme dans les autres instructions, donc 1=voix 1, 2=voix 2 et 4=voix 3.

6.3 Imitation d'instruments

Voici un petit catalogue d'enveloppes permettant d'imiter quelques instruments usuels. Ce catalogue résulte de nos essais et de notre appréciation personnelle et il va sans dire que vous pouvez varier les paramètres si votre appréciation est différente.

Piano

ENV 1,1,0,1,2,7,1,1,0,1,14,-1,5

Spécifiez un volume de départ nul dans SOUND (ex. pour avoir 3 la : SOUND 1,142,-3,0,1)

Clavecin

ENV 2,1,0,1,2,7,1,1,0,1,14,-1,2

Clavecin étouffé

ENV 3,1,0,1,2,7,1,1,0,1,14,-1,1,1,0,10

(remarquer la plage de silence à la fin pour normaliser la durée)

Guitare

ENV 4,1,0,1,2,7,1,1,0,1,3,-2,1,8,-1,4

Banjo (ou violon pizzicato)

ENV 5,1,0,1,2,7,1,1,0,2,4,-2,1,6,-1,4

Violon (alto, violoncelle, contrebasse)

Pour ces instruments, il faut aussi du vibrato, dont l'amplitude doit être adaptée à la note jouée et la vitesse doit être adaptée au tempo.

```
ENT -1,1,1,4,2,-1,4,1,1,4
ENV 6,1,0,1,2,7,1,1,-1,1,10,0,5
```

Clarinette

Cet instrument peut être imité sans enveloppe, ce qui revient à une enveloppe plate :

ENV 7,1,0,1,2,7,1,10,0,5

Flûte

ENV 8,1,0,1,4,3,3,10,0,10,4,-3,3

En ajoutant une modulation en début de note, on obtient un effet de flûte indienne :

ENT 2,2,1,4,4,-1,4,2,1,4

Trompette

ENV 9,1,15,7,1,0,15,15,-1,12

(le résultat n'est pas très ressemblant, mais c'est un problème de timbre. Il est encore plus impossible d'imiter le hautbois!)

Accordéon (bandonéon, harmonica)

Prendre l'enveloppe du type flûte et faire retentir un accord de quinte ou de quarte :

SOUND 1,142,100,0,8:SOUND 2,106,100,0,8

Tambour

ENV 10,1,0,1,1,15,1,1,0,5,1,-15,1,1,0,20

en prenant une note assez grave : SOUND 1,350,-3,0,10

Maracas

On utilise l'enveloppe du clavecin étouffé, mais on mélange avec du bruit :

SOUND 1,300,-3,0,3,,4

Bien entendu, ces chiffres ne sont donnés que comme point de

départ pour vos propres recherches. Nous n'avons pas cité l'orgue car cet instrument a une telle diversité que l'ordinateur peut difficilement l'imiter : c'est du côté des jeux de flûte et accordéon qu'il faudrait chercher.

Nous avons, en tous cas, un bon petit répertoire d'instruments; il s'agit maintenant de leur faire jouer des partitions.

6.4 Codage d'une partition

Cette section suppose que le Lecteur sait lire une partition en notation musicale habituelle. Ce livre ne peut, faute de place, contenir un cours de solfège. Toutefois les partitions abordées ici seront assez simples.

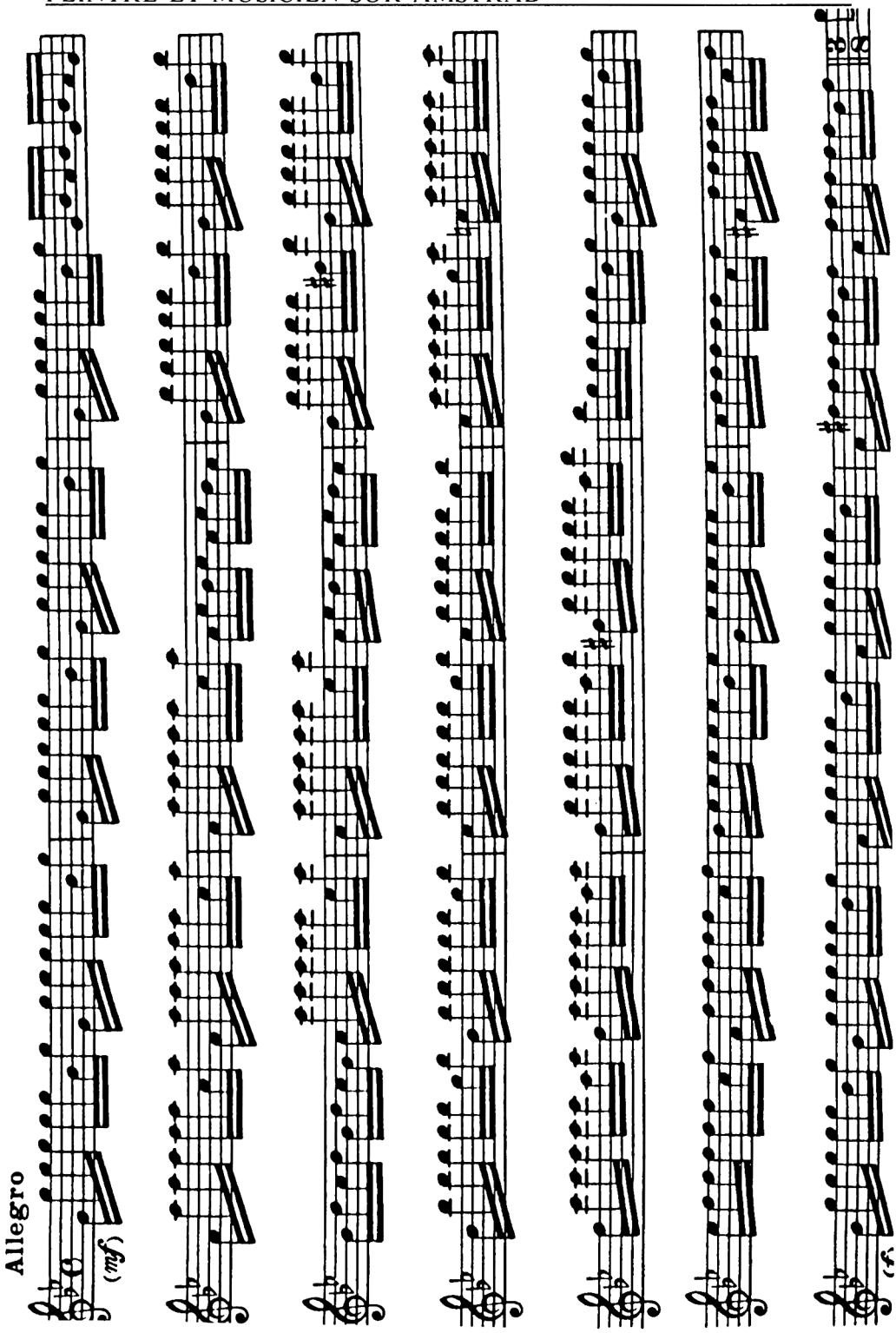
En effet, ne possèdant que trois voix, l'AMSTRAD est désarmé devant les partitions polyphoniques complexes. Une simple partition de piano peut le laisser coi : il y a souvent 5 à 10 notes à jouer en même temps. On est alors amené à simplifier la partition, ce qui est un exercice qui exige de grandes connaissances en harmonie. Une solution de ce problème consiste à faire des essais empiriques jusqu'à obtention d'un résultat satisfaisant.

Nous allons passer en revue quelques exemples, chacun s'adressant à un type donné de difficulté. Nous avons pris soin de les choisir dans différents genres musicaux : nous espérons que votre genre favori est représenté, mais nous avons été obligé de nous limiter faute de place.

Morceaux monodiques

"Le coucou"

Nous commençons par un morceau monodique, dont toutes les notes sont égales, le thème du coucou dans "l'Eté" des "Quatre Saisons" de Vivaldi. (Voir la partition ci-contre)


Allegro
(mf)

Nous prenons l'enveloppe 6 définie ci-dessus ("violon"). Il faut faire attention aux changements d'octaves et aux altérations : tous les si et mi sont bémolisés. Voici une traduction possible où nous utilisons la même méthode qu'à l'exercice 6.1 : les données sont mises en DATA, et comme les notes sont égales, c'est encore plus simple.

```
1 REM LE COUCOU DE VIVALDI
2 REM
10 ENV 6,1,0,1,2,7,1,1,-1,1,10,0,5
20 ENT -1,1,1,4,2,-1,4,1,1,4
30 READ H
40 WHILE H<>0
50 SOUND 1,H,12,0,6,1
60 READ H
70 WEND
100 DATA 159,80,80,80,80,80,134,80
110 DATA 159,80,80,80,80,80,134,80
120 DATA 159,80,80,80,80,80,134,80
130 DATA 159,80,80,80,80,80,134,80
140 DATA 159,80,80,80,80,80,134,80
150 DATA 159,134,142,134,159,134,142,159
160 DATA 142,71,71,71,71,71,119,71
170 DATA 142,71,71,71,71,71,119,71
180 DATA 142,71,71,71,71,71,119,71
190 DATA 142,119,134,119,142,119,134,142
200 DATA 134,67,67,67,67,67,106,67
210 DATA 134,67,67,67,67,67,106,67
220 DATA 134,106,119,106,134,106,119,134
230 DATA 119,60,60,60,60,60,106,60
240 DATA 119,60,60,60,60,60,106,60
250 DATA 119,100,106,100,119,100,106,119
260 DATA 106,53,53,53,53,53,84,53
270 DATA 106,53,53,53,53,53,84,53
280 DATA 106,67,67,67,67,67,80,67
290 DATA 106,67,67,67,67,67,80,67
300 DATA 106,67,67,67,67,67,80,67
310 DATA 106,67,67,67,67,67,80,67
320 DATA 100,60,60,60,60,60,80,60
330 DATA 95,60,60,60,60,60,80,60
340 DATA 89,60,60,60,60,60,71,60
350 DATA 89,60,60,60,60,60,71,60
360 DATA 89,53,53,53,53,53,71,53
```

```
370 DATA 84,53,53,53,53,53,71,53
380 DATA 67,80,80,80,80,80,106,80
390 DATA 134,80,80,80,80,80,106,80
400 DATA 100,89,89,89,89,89,119,89
410 DATA 142,89,89,89,89,89,119,89
420 DATA 106,100,100,100,100,100,134,100
430 DATA 159,100,100,100,100,100,134,100
440 DATA 119,106,106,106,106,106,142,106
450 DATA 169,106,106,106,106,106,142,106
460 DATA 134,80,80,80,80,80,106,80
470 DATA 134,80,80,80,80,80,106,80
480 DATA 142,80,80,80,80,80,106,80
490 DATA 142,80,80,80,80,80,106,80
500 DATA 142,84,84,84,84,84,106,84
510 DATA 142,84,84,84,84,84,106,84
520 DATA 80,0
```

En fait, ce n'est qu'à moitié satisfaisant : les DATA sont très fastidieuses. Ce qu'il nous faudrait, c'est un petit langage de traduction de partitions, et, bien sûr, le programme qui interprète ce langage. Nous allons nous doter d'un tel langage.

```
500 REM  INTERPRETE
501 REM
505 D=33:H=239:M=180:O=3:MM=1:OO=1
510 VV$="RBNCDTQncd":DIM V(10)
520 FOR I=1 TO 7:V(I)=2^(8-I):NEXT
530 V(8)=24:V(9)=12:V(10)=6
540 NN$="DO RE MI FA SOLLA SI DO#RE#FA#LA#SOL#"
550 DIM N(12):N(1)=239:N(2)=213:N(3)=190:N(4)=179
560 N(5)=159:N(6)=142:N(7)=127:N(8)=225:N(9)=201
570 N(10)=169:N(11)=134:N(12)=150
580 READ C$
590 WHILE C$<>"FIN"
600 IF LEN(C$)=1 GOTO 650
610 I=(INSTR(NN$,C$)-1)/3+1
620 H=CINT(N(I)*OO)
630 SOUND 1,H,D,VOL,EV,ET,BR
640 READ C$
645 WEND : RETURN
650 IF C$="+" THEN O=O+1:OO=OO/2:GOTO 640
660 IF C$="-" THEN O=O-1:OO=2*OO:GOTO 640
670 IF C$="M" THEN READ M:MM=180/M:GOTO 640
680 IF C$="L" THEN READ D:GOTO 640
690 I=INSTR(VV$,C$):D=CINT(MM*V(I)):GOTO 640
```

Le programme ci-dessus réalise un tel interprète. Voici les règles du langage correspondant. Les commandes sont à la suite les unes des autres, dans des instructions DATA, séparées par des virgules.

Il y a trois sortes de commandes :

- Les commandes de hauteur de notes. On donne une note à la fois, sous forme de son nom en français, ex. SOL ou RE#. Pour les altérations, on n'a que le dièse, donc pour un si bémol, vous devez écrire LA#. Est illégal tout nom qui ne correspond pas à une touche noire du piano (ex. SI# : écrivez DO).

- Les commandes de valeur de notes : R, B, N, C, D, T, Q pour ronde, blanche, noire, croche, double, triple et quadruple croche. n, c et d correspondent aux mêmes figures, mais dans un triolet. Une valeur reste la même pour les notes successives tant qu'on ne la change pas.

- Les commandes diverses; M,valeur impose le tempo : la valeur à mettre est l'indication métronomique correspondant à la noire. L'option par défaut est noire=180 (allegro vivace). L,valeur impose la valeur du paramètre durée. + et - montent et descendent d'une octave; l'octave par défaut au départ est 3.

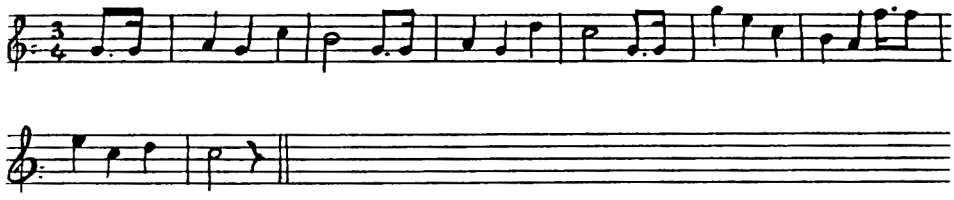
L'interprète a été mis sous forme d'un sous-programme : pour l'appeler, vous définissez enveloppe et modulation et vous imposez les valeurs respectives de EV (enveloppe), ET (modulation), VOL (volume) et BR (bruit); vous faites RESTORE au numéro de ligne où commencent vos DATA et GOSUB 500.

Ci-dessous, nouvelle version du Coucou de VIVALDI qui y fait appel.

```
1 REM  LE COUCOU VERSION 2
2 REM
10 ENV 6,1,0,1,2,7,1,1,-1,1,10,0,5
20 ENT -1,1,1,4,2,-1,4,1,1,4
30 EV=6:ET=1:BR=0:VOL=0
40 RESTORE 100
50 GOSUB 500
70 END
100 DATA M,120,D
110 DATA SOL,+,SOL,SOL,SOL,SOL,SOL,-,LA#,+,SOL,-
120 DATA SOL,+,SOL,SOL,SOL,SOL,SOL,-,LA#,+,SOL,-
130 DATA SOL,+,SOL,SOL,SOL,SOL,SOL,-,LA#,+,SOL,-
140 DATA SOL,+,SOL,SOL,SOL,SOL,SOL,-,LA#,+,SOL,-
150 DATA SOL,+,SOL,SOL,SOL,SOL,SOL,-,LA#,+,SOL,-
160 DATA SOL,LA#,LA,LA#,SOL,LA#,LA,SOL
170 DATA LA,+,LA,LA,LA,LA,LA,DO,LA,-
180 DATA LA,+,LA,LA,LA,LA,LA,DO,LA,-
190 DATA LA,+,LA,LA,LA,LA,LA,DO,LA,-
200 DATA LA,+,DO,-,LA#,+,DO,-,LA,+,DO,-,LA#,LA
210 DATA LA#,+,LA#,LA#,LA#,LA#,LA#,RE,LA#,-
220 DATA LA#,+,LA#,LA#,LA#,LA#,LA#,RE,LA#,-
230 DATA LA#,+,RE,DO,RE,-,LA#,+,RE,DO,-,LA#,+
240 DATA DO,+,DO,DO,DO,DO,DO,-,RE#,+,DO,-
250 DATA DO,+,DO,DO,DO,DO,DO,-,RE#,+,DO,-
260 DATA DO,RE#,RE,RE#,DO,RE#,RE,DO
270 DATA RE,+,RE,RE,RE,RE,RE,-,FA#,+,RE,-
280 DATA RE,+,RE,RE,RE,RE,RE,-,FA#,+,RE,-
290 DATA RE,LA#,LA#,LA#,LA#,LA#,SOL,LA#
300 DATA RE,LA#,LA#,LA#,LA#,LA#,SOL,LA#
310 DATA RE#,LA#,LA#,LA#,LA#,LA#,SOL,LA#
320 DATA RE#,LA#,LA#,LA#,LA#,LA#,SOL,LA#
330 DATA RE#,+,DO,DO,DO,DO,DO,-,SOL,+,DO,-
340 DATA MI,+,DO,DO,DO,DO,DO,-,SOL,+,DO,-
350 DATA FA,+,DO,DO,DO,DO,DO,-,LA,+,DO,-
360 DATA FA,+,DO,DO,DO,DO,DO,-,LA,+,DO,-
370 DATA FA,+,RE,RE,RE,RE,RE,-,LA,+,RE,-
380 DATA FA#,+,RE,RE,RE,RE,RE,-,LA,+,RE,-
390 DATA LA#,SOL,SOL,SOL,SOL,SOL,RE,SOL,-
400 DATA LA#,+,SOL,SOL,SOL,SOL,SOL,RE,SOL
410 DATA RE#,FA,FA,FA,FA,FA,DO,FA,-
415 DATA LA,+,FA,FA,FA,FA,FA,DO,FA
420 DATA RE,RE#,RE#,RE#,RE#,RE#,-,LA#,+,RE#,-
425 DATA SOL,+,RE#,RE#,RE#,RE#,RE#,-,LA#,+,RE#
430 DATA DO,RE,RE,RE,RE,RE,-,LA,+,RE,-
```

```
435 DATA FA#,+,RE,RE,RE,RE,RE,-,LA,+,RE,-
440 DATA LA#,+,SOL,SOL,SOL,SOL,SOL,RE,SOL,-
445 DATA LA#,+,SOL,SOL,SOL,SOL,SOL,RE,SOL,-
450 DATA LA,+,SOL,SOL,SOL,SOL,SOL,RE,SOL,-
455 DATA LA,+,SOL,SOL,SOL,SOL,SOL,RE,SOL,-
460 DATA LA,+,FA#,FA#,FA#,FA#,FA#,RE,FA#,-
465 DATA LA,+,FA#,FA#,FA#,FA#,FA#,RE,FA#,C,SOL
499 DATA FIN
```

Notes inégales : "Happy Birthday"



Aucune difficulté dans cette partition (nous nous bornons à un court extrait). Remplacez les DATA de l'exemple précédent par :

```
100 DATA M,120
110 DATA n,SOL,D,SOL,N,LA,SOL,+,DO,-,B,SI
120 DATA n,SOL,D,SOL,N,LA,SOL,+,RE,B,DO,-
130 DATA n,SOL,D,SOL,+,N,SOL,MI,DO,-,SI,LA
140 DATA +,n,FA,D,FA,N,MI,DO,RE,B,DO
499 DATA FIN
```

Exercice 6.7

En musique, DC (Da Capo) veut dire qu'il faut rejouer tout le morceau. Faites-le.

Accords et polyphonie

Il nous faut tout d'abord enrichir notre interprète pour qu'il

admette une spécification de voix. La commande de voix sera formée d'un numéro (1, 2 ou 3) qui indique la voix à laquelle s'adressent les prochaines commandes. D'autre part, comme les accords sont gérés par attentes et libération, les commandes de notes des trois voix sont mises en files d'attente, et la commande J ("joue!") fait retentir l'accord. En fait, vous devrez répartir votre partition en petits segments simultanés, déclenchés par des commandes J. Il faudra veiller à ce que chaque segment ne contienne pas plus de cinq notes pour chaque voix, puisque c'est la limite des files d'attente.

Une difficulté supplémentaire dans la préparation des données est que, avec cet enchaînement de données s'adressant à des voix différentes, vous devez faire attention à quelle octave vous êtes : si, venant de spécifier une note de l'octave 2 pour la voix 1, vous revenez à l'octave 3 pour la voix 2, n'oubliez pas le +; c'est la note qu'on vient de spécifier qui compte et non pas la note qui précède dans la même voix.

Voici notre interprète version 2. On en a profité pour inclure la commande '.' : elle suit une commande de durée de note et, comme en notation musicale classique, elle ajoute 50% de durée.

```
500 REM  INTERPRETE 2
501 REM
505 D=33:H=239:M=180:O=3:MM=1:OO=1
510 VV$="RBNCDTOncd":DIM V(10)
520 FOR I=1 TO 7:V(I)=2^(8-I):NEXT
530 V(8)=24:V(9)=12:V(10)=6
540 NN$="DO RE MI FA SOLLA SI DO#RE#FA#LA#SOL#"
550 DIM N(12):N(1)=239:N(2)=213:N(3)=190:N(4)=17
9
560 N(5)=159:N(6)=142:N(7)=127:N(8)=225:N(9)=201
570 N(10)=169:N(11)=134:N(12)=150
575 P(1)=65:P(2)=66:P(3)=68:K=1
580 READ C$
590 WHILE C$<>"FIN"
600 IF LEN(C$)=1 GOTO 650
610 I=(INSTR(NN$,C$)-1)/3+1
620 H=CINT(N(I)*OO)
630 K=P(J):IF SQ(K AND 191)>=128 THEN 630
635 SOUND K,H,D,VOL(J),EV(J),ET(J),BR(J):IF P(J)
>64 THEN P(J)=P(J)-64
640 READ C$
```

```
645 WEND : RETURN
650 IF C$="+" THEN O=O+1:OO=OO/2:GOTO 640
660 IF C$="-" THEN O=O-1:OO=2*OO:GOTO 640
670 IF C$="M" THEN READ M:MM=180/M:GOTO 640
680 IF C$="L" THEN READ D:GOTO 640
690 IF C$="." THEN D=CINT(1.5*D):GOTO 640
700 IF C$="J" THEN RELEASE 7:P(1)=65:P(2)=66:P(3
)=68:K=1:GOTO 640
710 JJ=VAL(C$):IF JJ=0 THEN I=INSTR(VV$,C$):D=CI
NT(MM*V(I)):GOTO 640
720 J=JJ:GOTO 640
```

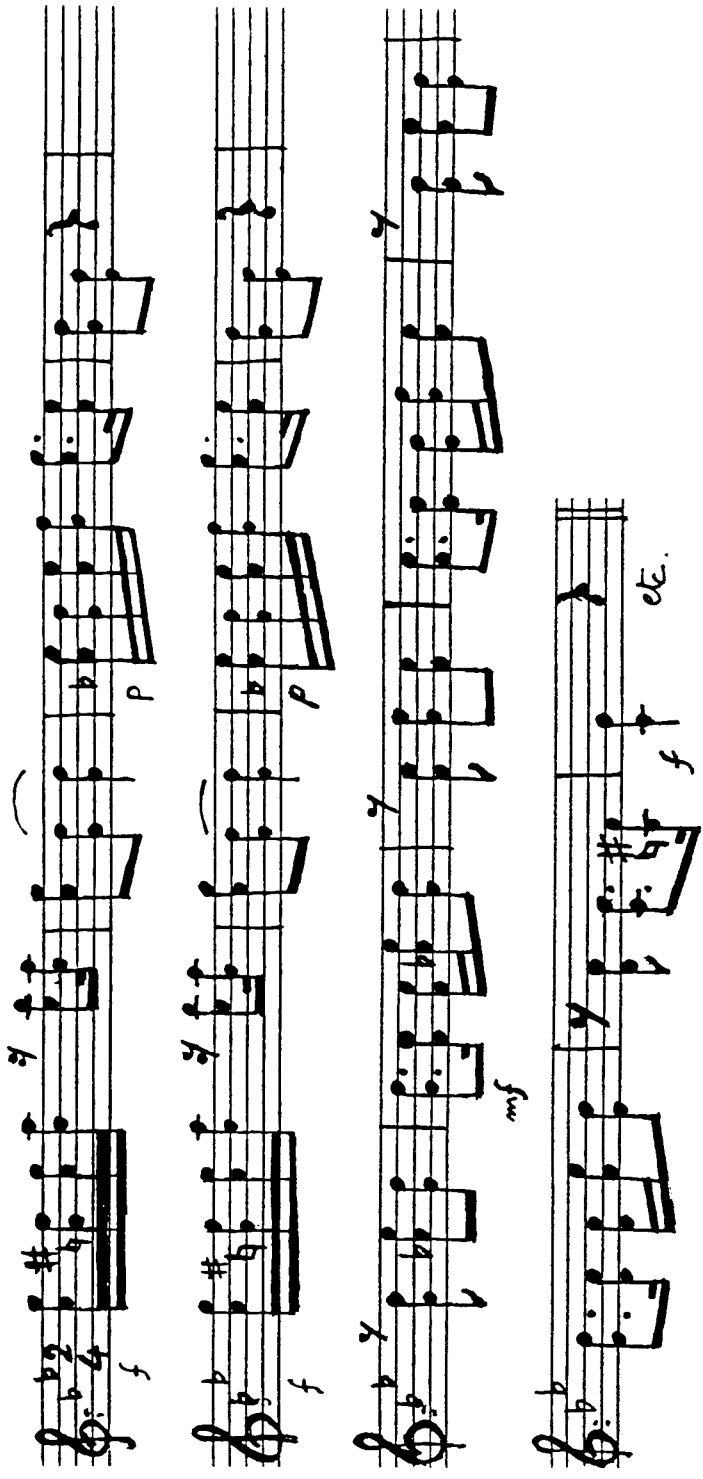
Bien voir en 630, 635 comment on évite de surcharger la file d'attente en testant la fonction SQ.

Exercice 6.8

Ajoutez une commande P qui crée un silence de durée égale à la durée qui résulte de la dernière commande de durée : la séquence N,P produira un soupir (silence égal à une noire).

"La Cumparsita"

Nous donnons ci-dessous un extrait du célèbre tango de Matos Rodriguez. Nous avons remplacé chaque note par un accord de quinte ou de quarte pour mieux imiter le bandonéon. (Voir la partition ci-contre)


p
f
p
f
mf
etc.
f

Par rapport au programme qui précède, nous changeons bien sûr le tempo et l'enveloppe. Les accords devraient se faire sans problèmes puisque ce sont des accords simples, de notes de même durée. On utilise la commande P (pause) ajoutée à l'exercice 6.8.

```
1 REM  LA CUMPARSITA
2 REM
10 ENV 8,1,0,1,4,3,3,10,0,10,4,-3,3
20 FOR I=1 TO 3
30 EV(I)=8:ET(I)=0:VOL(I)=0:BR(I)=0:NEXT
40 RESTORE 100
50 GOSUB 500
60 END
100 DATA M,100
110 DATA 1,+,D,SOL,FA#,SOL,LA
120 DATA 2,DO,-,SI,+,DO,RE,J,P
130 DATA 1,C,LA#,D,LA,C,SOL,N,.,RE
140 DATA 2,C,RE#,D,RE,C,DO,-,N,.,SOL,J
150 DATA 1,+,D,RE#,RE,RE#,FA
160 DATA 2,-,SOL#,SOL,SOL#,LA#,J
170 DATA 1,+,C,.,SOL,D,RE#,C,RE,-,LA#
180 DATA 2,+,C,.,DO,D,-,LA,C,SOL,RE#,J
190 DATA 1,+,D,SOL,FA#,SOL,LA
200 DATA 2,DO,-,SI,+,DO,RE,J,P
210 DATA 1,C,LA#,D,LA,C,SOL,N,.,RE
220 DATA 2,C,RE#,D,RE,C,DO,-,N,.,SOL,J
230 DATA 1,+,D,RE#,RE,RE#,FA
240 DATA 2,-,SOL#,SOL,SOL#,LA#,J
250 DATA 1,+,C,.,SOL,D,RE#,C,RE,-,LA#
260 DATA 2,+,C,.,DO,D,-,LA,C,SOL,RE#,J
270 DATA C,P,1,+,RE,RE#,RE,C,.,RE
280 DATA 2,-,C,SOL,SOL#,SOL,C,.,SOL,J
290 DATA 1,+,D,DO,DO,RE#,C,RE
300 DATA 2,-,D,FA,FA,SOL#,C,SOL,J,P
310 DATA 1,+,DO,RE,DO,C,.,DO
320 DATA 2,-,C,FA,SOL,FA,C,.,FA,J
330 DATA 1,D,LA#,LA#,+,RE,C,DO
340 DATA 2,D,-,RE#,RE#,SOL,C,FA,J,P
350 DATA 1,LA#,+,DO,-,LA#,C,.,LA#
360 DATA 2,C,RE#,FA,RE#,C,.,RE#,J
370 DATA 1,D,LA,LA,+,DO,-,C,LA#
```

```
380 DATA 2,D,RE,RE,FA,C,LA#,J,P
390 DATA 1,LA,C,.,SOL,D,FA#,N,SOL
400 DATA 2,C,RE,C,.,DO,D,-,SI,+,N,DO,J
499 DATA FIN
```

On pourrait traduire des partitions plus compliquées, mais deux difficultés apparaissent :

- Les accords dépassent vite trois notes et donc sortent des possibilités de l'AMSTRAD. Une solution est de simplifier ces accords en supprimant certaines notes, mais il faut les sélectionner avec soin.

- Si la partition est trop complexe, on arrive à dépasser les possibilités de notre interprète au point de vue vitesse, c'est-à-dire qu'on a une exécution de la partition ralentie par moments. Ceci est dû à la lenteur de Basic et une solution serait d'écrire un interprète analogue en langage machine. Une autre solution, très fastidieuse, serait de renoncer aux avantages de notre interprète et de revenir à des DATA préparées comme au début de ce chapitre.

  Une autre solution possible aussi serait de faire une sorte de compilation : on remplacerait l'interprète par un programme en deux parties. La première partie lirait les données, en déduirait les valeurs des paramètres et les mettrait dans des tableaux. La seconde partie jouerait alors la musique des tableaux. Elle pourrait être gérée par événements à l'aide ON SQ() GOSUB pour gagner encore du temps.

Dans tous les cas, les principes à appliquer ont maintenant été vus et vous avez pu vous rendre compte des possibilités de l'AMSTRAD en matière musicale.

ANNEXE I

RÉSUMÉ DES ORDRES GRAPHIQUES DE L'AMSTRAD

(Un + signale une instruction qui n'existe pas ou qui a des restrictions sur 464)

mot-clé	définition **exemples**	page
BORDER	Définit la couleur du bord de l'écran **BORDER 4** **BORDER 4,6**	36
CLG	Efface l'écran graphique et fixe la couleur de fond si elle est spécifiée **CLG** **CLG 4**	42
CLS	Vide l'écran (la fenêtre indiquée) **CLS**	42
CURSOR +	Rend le curseur visible (1) ou non (0) **CURSOR 1**	
DRAW	Trace une ligne jusqu'au point indiqué de la couleur spécifiée **DRAW 100,200** **DRAW 100,200,5**	50

DRAWR	Trace une ligne jusqu'au point indiqué de la couleur spécifiée, mais en coordonnées relatives **DRAWR 100,200 DRAWR 100,200,5**	50
FILL +	Remplit une aire fermée dans l'écran graphique avec l'encre spécifiée **FILL 3**	58
FRAME	Synchronise l'action au balayage écran ce qui peut rendre l'affichage moins saccadé	
GRAPHICS PAPER +	Etablit la couleur de fond de l'écran graphique **GRAPHICS PAPER 3**	42
GRAPHICS PEN +	Etablit la couleur de tracé graphique et la transparence (2è param = 1) **GRAPHICS PEN 1 GRAPHICS PEN 1,1**	42
INK	Associe une couleur à un n° d'encre Si on spécifie deux couleurs, il y a clignotement **INK 1,0,13**	42
LOCATE	Positionne le curseur dans la fenêtre **LOCATE #5,20,10 LOCATE 20,10**	32
MASK +	Définit le pointillé utilisé pour les tracés **MASK &X00001111,0**	51
MODE	Fixe le mode d'affichage et vide l'écran **MODE 2**	42
MOVE	Déplace (sans tracé) le curseur graphique à l'emplacement indiqué **MOVE 100,200**	46
MOVER	Comme MOVE, mais en coordonnées relatives **MOVER 10,10**	46

ORIGIN	Fixe l'origine des coordonnées sur l'écran graphique, et éventuellement définit une fenêtre **ORIGIN 100,200** **ORIGIN X,Y,G,D,H,B**	44
PAPER	Fixe la couleur de fond dans la fenêtre **PAPER 3** **PAPER #5,3**	42
PEN +	Fixe la couleur d'écriture dans la fenêtre **PEN 1** **PEN #5,1** **PEN #5,1,1** (664 : mode transparent)	42
PLOT	PLOT x,y,c,m trace le point (x,y) dans la couleur c et (664) avec le mode m m=0 (normal), 1 (XOR), 2 (AND) ou 3 (OR) **PLOT 100,200** **PLOT 100,300,2** (664) **PLOT 100,300,2,2**	47
PLOTR	Comme PLOT, mais les coordonnées du point sont relatives **PLOTR 20,20**	47
SPEED INK	Règle la vitesse de clignotement des couleurs **SPEED INK 50,30**	33
SYMBOL	Permet de redéfinir un dessin de caractère **SYMBOL 200,0,0,0,0,1,1,1,1**	32
SYMBOL AFTER	Doit précéder une série de SYMBOL pour préciser à partir de quel code on fait des redéfinitions **SYMBOL AFTER 200**	
TAG	Envoie le texte à la position du curseur graphique	59
TAGOFF	Annule l'effet de TAG	59
WIDTH	Ajuste la largeur d'imprimante	
WINDOW	WINDOW #n,g,d,h,b définit la fenêtre n° n dans l'ordre gauche, droite, haut, bas **WINDOW #3,1,20,1,5**	37

WINDOW SWAP	Echange le contenu de deux fenêtres **WINDOW SWAP 1,2**	
	FONCTIONS	
COPYCHR$ +	Copie le caractère sous le curseur dans la fenêtre spécifiée **X$=COPYCHR$(#0)**	
TEST	Couleur d'encre du point indiqué **C=TEST(X,Y)**	60
TESTR	Comme TEST mais en coordonnées relatives	60
VPOS	Ligne où se trouve le curseur texte dans la fenêtre indiquée **L=VPOS(#0)**	
XPOS	Abscisse du curseur graphique	46
YPOS	Ordonnée du curseur graphique	46

ANNEXE **II**

RESUME DES INSTRUCTIONS SONORES

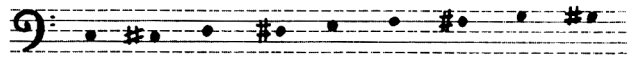
mot-clé	définition **exemples**	page
ENT	Définit une modulation sonore (voir chapitre 6) **ENT 1,10,2,2**	139
ENV	Définit une enveloppe sonore (voir chapitre 6) **ENV 1,10,2,2,10,0,2,20,-1,2**	137
ON SQ(n) GOSUB	Appelle le sous-programme indiqué dès qu'il y a une place libre dans la file d'attente du canal sonore spécifié **ON SQ(2) GOSUB 500**	156
RELEASE	Libère une ou plusieurs voix **RELEASE** 7 (libère les trois voix)	134
SOUND	Fait retentir un son (cf chapitre 6) **SOUND 1,142,1000**	132

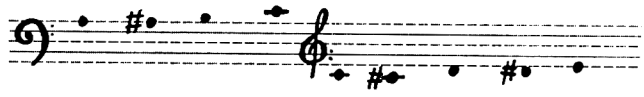
	FONCTION	
SQ	Indique l'état de la voix spécifiée L'argument vaut 1, 2 ou 4 (2^voix) Le résultat s'interprète bit par bit : bits 0-2 nombre de places libres dans la file d'attente bits 3-5 état du rendez-vous bit 6 =1 si la tête de file est bloquée bit 7 =1 si la voix est en activité **IF SQ(1) AND 64 THEN ...**	142

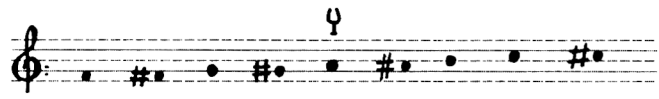
ANNEXE III

TABLEAU DES NOTES

Pour chaque note (de ut-2 à fa#-5), on donne son nom en français, sa fréquence et la valeur correspondante pour SOUND.

NOTE									
NOM	DO-2		RE		MI	FA		SOL	
fréquence	131	139	147	155	165	175	185	196	207
F SOUND	478	451	426	402	379	358	338	319	301

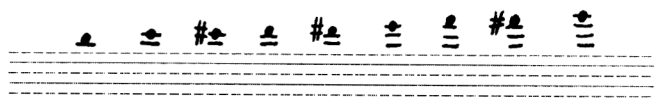
NOTE								
NOM	LA		SI	DO-3		RE		MI
fréquence	220	233	247	262	277	293	311	330
F SOUND	284	268	253	239	225	213	201	190

NOTE 

NOM	FA		SOL	LA		SI	DO-4		
fréquence	349	370	392	415	440	466	494	523	554
F SOUND	179	169	159	150	142	134	127	119	113

NOTE 

NOM	RE		MI	FA		SOL		LA	
fréquence	587	622	659	698	740	784	830	880	932
F SOUND	106	100	95	89	84	80	75	71	67

NOTE 

NOM	SI	DO-5		RE		MI	FA	FA#	
fréquence	987	1046	1108	1174	1243	1318	1397	1479	1567
F SOUND	63	60	56	53	50	47	45	42	40

ANNEXE **IV**

SOLUTION DES EXERCICES

Exercice 1.1

SAVE "ZONE",B,1024,1024

Exercice 1.2

Si la condition est vraie : i01, i11, i12 et i31 (ou 1000 et les suivantes si i12 est un GOTO 1000).
Si elle est fausse : i01, i21, i22 et i31.

Exercice 1.3

A$="" : WHILE A$<>"A" : A$=INKEY$: WEND

Exercice 1.4

Non, car s'il n'y a pas de client de nom DUPONT, on ne s'arrêtera que sur un message d'erreur (I trop grand par rapport à la dimension du tableau).

On ne peut démarrer la boucle par :

100 I=0: WHILE NOM$(I)<>"DUPONT"

que si on a mis un nom différent de DUPONT comme élément 0 du tableau.

Exercice 1.5

```
10 ON ERROR GOTO 1000
:
:
1000 IF ERR=4 THEN RESTORE : RESUME
1010 ON ERROR GOTO 0 : RESUME
```

Exercice 1.6

On forme un tableau ERF$ tel que ERF$(I) contienne le message en français convenant à l'erreur n° I. C'est assez fastidieux à constituer et cela consomme de la place mémoire vu le nombre d'erreurs possibles.

Une fois ceci fait, la routine d'erreur s'écrit (on suppose qu'on fait un ON ERROR GOTO 10000) :

```
10000 PRINT ERF$(ERR) : STOP
```

Exercice 1.7

format	nombre	impression
####	-1000	%-1000
* $$##.## *	5.948	* $5.95 *
* $$##.## *	54.95	* $54.95 *
* $$##.## *	545.95	* $545.95 *
* $$##.## *	5435.95	* %$5435.95 *
* $$##.## *	-5.95	* -$5.95 *
A= ### M	35.4	A= 35 M

Exercice 2.1

MODE 1

Exercice 2.2

```
10 PRINT CHR$(214)
20 PRINT " ";CHR$(143);CHR$(143);CHR$(143);CHR$(215)
30 PRINT CHR$(204);CHR$(204);"  ";CHR$(205);CHR$(205)
```

Exercice 2.3

Oui. (Ce n'est pas le cas dans tous les Basic, mais c'est possible en Basic AMSTRAD).

Exercice 2.4

BORDER 16 : INK 0,1 : INK 1,26

en supposant que PEN et PAPER ont gardé leurs valeurs par défaut. Sinon faire aussi PAPER 0 : PEN 1 .

Pour que le fond devienne effectivement bleu s'il ne l'était pas, faire aussi CLS .

Exercice 2.5

Il faut la bordure noire, le fond blanc, les caractères noirs, d'où :

BORDER 0 : INK 0,26 : INK 1,0 : CLS

Exercice 2.6

Tout dépend du mode dans lequel on se trouve. En mode 2, il n'y a pas moyen puisqu'on ne peut avoir que deux couleurs. En modes 0 et 1, on fera :

10 INK 0,9 : INK 1,3 : INK 2,13 : INK 3,1

20 PAPER 0 : PEN 1 : CLS
30 écriture dans la première moitié de l'écran
....p. ex. 40 LOCATE 1,2 : PRINT " ... "
....
60 LOCATE 1,13 : PAPER 2 : PEN 3
70 FOR I=1 TO 13 : PRINT " 20 ou 40 espaces ";:NEXT
80 écriture dans la 2è moitié
.... p. ex. 80 LOCATE 1,15 : PRINT " ... "

(70 a pour but d'imposer la 2è couleur de fond dans la 2è moitié de l'écran. Ce serait plus facile avec une fenêtre).

Exercice 2.7

```
1 REM  Ex 2-7
2 REM
10 MODE 0
20 FOR I=1 TO 500
30 K=INT(14*RND(1))
40 PAPER K : PRINT" ";
50 NEXT
60 A$="":WHILE A$="" : A$=INKEY$ : WEND
```

Exercice 3.1

L'instruction ORIGIN définit la fenêtre graphique. Les deux premiers paramètres sont nuls car on ne déplace pas l'origine des coordonnées. La suite PAPER 3 puis CLS définit la couleur de fond de l'écran texte comme rouge et le CLS recouvre effectivement l'écran de cette couleur. Le CLG vide la fenêtre graphique, ce qui la fait revenir au bleu, couleur de l'encre 0. Notez que l'écran reste entouré de sa bordure bleue puisque rien n'a été fait pour la changer.

Exercice 3.2

320,200

Exercice 3.3

MOVE 300,200 : DRAWR -50,-100 : DRAWR 100,0 : DRAWR -50,100

Exercice 3.4

```
10005 PLOTR 0,0,N
```

Exercice 3.5

```
11000 REM  RECTANGLE TOURNE
11010 HX=DX*COS(ALPHA) : HY=DX*SIN(ALPHA)
11020 VX=-DY*SIN(ALPHA) : VY=DY*COS(ALPHA)
11030 DRAWR HX,HY : DRAWR VX,VY
11040 DRAWR -HX,-HY : DRAWR -VY,-VY
11050 RETURN
```

Exercice 3.6

DRAWR Z,-Z : DRAWR Z,Z : DRAWR -Z,Z : DRAWR -Z,-Z

Z est la longueur de la demi-diagonale. Ces instructions sont à exécuter après s'être positionné sur le premier sommet et après avoir, s'il y a lieu, fixé la couleur de tracé.

Exercice 3.7

(a)
```
10 PLOT 100,100 : DX=200 : DY=100 : GOSUB 10000
20 PLOT 120,120 : DX=160 : DY=60 : GOSUB 12000
```

(b)
```
10 PLOT 100,100 : DX=200 : DY=100 : GOSUB 12000
20 PLOT 120,120,0 : DX=160 : DY=60 : GOSUB 12000
```

On utilise les sous-programmes de tracé de rectangles fournis dans le texte.

Exercice 3.8

```
1 REM  EXERCICE 3.8
2 REM
10 NE=1:F=1:AR=0:DEG:XC=200:YC=200:IA=2
20 CLG:AD=30:AF=80:R=100
30 PLOT XC,YC,NE:AAF=AF
40 DRAW XC+R*COS(AD),YC+R*SIN(AD)
50 GOSUB 20000
60 PLOT XC,YC:AF=AAF
70 DRAW XC+R*COS(AF),YC+R*SIN(AF)
80 END
```

En 20000 commence, bien sûr, la routine de tracé de cercles vue dans le texte.

Exercice 3.9

```
1 REM  EXERCICE 3.9
2 REM
10 NE=1:F=1:AR=0:DEG:XC=200:YC=200:IA=2
20 CLG:AD=0:AF=360
30 FOR R=80 TO 150
40 GOSUB 20000
50 NEXT
60 END
```

Ceci est très lent à l'exécution alors que le programme suivant qui utilise FILL (attention, pas sur 464!) est plus rapide :

```
1 REM  EXERCICE 3.9 - 2
2 REM
10 NE=1:F=1:AR=0:DEG:XC=200:YC=200:IA=2
20 CLG:AD=0:AF=360
30 R=80 : GOSUB 20000
40 R=150 : GOSUB 20000
50 MOVE XC+100,YC : FILL 1
60 END
```

Exercice 3.10

```
1 REM  EXERCICE 3.10
2 REM
10 MODE 1 : INK 2,18 : NE=1
20 F=1:DEG:AD=0:AR=0:AF=360:IA=2
30 XC=200:YC=200:R=100:CLG
40 GOSUB 20000
50 R=150:F=0.3:GOSUB 20000
60 MOVE 60,200 :FILL 3
70 MOVE 310,200 :FILL 3
80 MOVE 200,290 :FILL 2
90 MOVE 200,110 :FILL 1
100 END
```

Exercice 3.11

```
1 REM  EXERCICE 3.11
2 REM
10 MODE 1 : INK 2,15 : NE=1
20 F=1:DEG:AD=0:AR=0:AF=360:IA=2
30 XC=200:YC=200:R=100:CLG
40 GOSUB 20000
50 R=170:F=0.3:GOSUB 20000
60 R=135:F=0.3:GOSUB 20000
70 MOVE 200,200 : FILL 1
80 MOVE 200,242 : FILL 1
90 MOVE 200,260 : FILL 1
100 MOVE 200,110 : FILL 1
110 MOVE 200,155 : FILL 2
120 MOVE 50,200 : FILL 2
130 MOVE 340,200 : FILL 2
140 END
```

Exercice 3.12

```
1 REM  EXERCICE 3.12
2 REM
10 MODE 1 : INK 2,24 : NE=1
20 F=1:DEG:AD=0:AR=0:AF=360:IA=2
30 XC=200:YC=200:R=100:CLG
```

```
40 F=0.83 : GOSUB 20000
50 R=170:F=0.3:GOSUB 20000
60 R=135:F=0.3:GOSUB 20000
70 MOVE 200,200 : FILL 2
80 MOVE 200,242 : FILL 2
90 MOVE 200,260 : FILL 2
100 MOVE 200,120 : FILL 2
110 MOVE 200,155 : FILL 3
120 MOVE 50,200 : FILL 3
130 MOVE 340,200 : FILL 3
140 PLOT 130,210,1 : TAG
150 PRINT "SATURNE"; : TAGOFF
160 END
```

On a effectué quelques ajustements de couleurs et donné le facteur d'ellipticité voulu pour que ce soit rond sur l'imprimante. Voici une reproduction du résultat :

Exercice 3.13

La fin devient :

```
90 NEXT
95 MOVE 50,200:FILL 1:MOVE 290,255:FILL 1
100 TAG : FOR I=0 TO 400 STEP 40
101 PLOT I,100,1 : PRINT RIGHT$(STR$(75+I/40),2)
;
102 NEXT
105 END
110 DATA 100,100,105,98,107,92,110,85
```

Exercice 3.14

On peut remplacer 60050 par :

```
60045 XX=X+12-I-I : T=TEST(XX)+TEST(XX+1)
60050 N=N-(2^I)*(T<>0)
```

Exercice 3.15

```
1 REM  EXERCICE 3.15
2 REM
200 DATA DUPONT,7
210 DATA MATHS,PHYSIQUE,CHIMIE
220 DATA SC NAT,FRANCAIS,PHILO,HIST-GEO
230 DATA 60,72,64,56,36,52,44,60
```

Exercice 3.16

On change l'initialisation de X1 en 20 : elle devient X1=60 et on ajoute à la fin :

```
130 PLOT 0,300:DRAW 0,100
140 FOR Y=0 TO 100 STEP 10
150 YY=100+2*Y : Y$=RIGHT$(STR$(Y),3)
160 PLOT 0,YY:DRAWR 5,0
170 MOVER 0,12:PRINT Y$;
180 NEXT
```

Exercice 3.17

```
1 REM EXERCICE 3.17
2 REM
10 CLG : X=16
20 Y=100 : TAG : DX=16
30 FOR I=1 TO 4
40 READ P,Q
50 DY=P:PLOT X,Y,2:GOSUB 12000
60 DY=Q:PLOT X+16,Y,3:GOSUB 12000
```

```
70 PLOT X-20,70,1:PRINT RIGHT$(STR$(1965+5*I),4)
;
80 X=X+80
90 NEXT
100 END
110 REM DONNEES ARBITRAIRES
120 DATA 70,4,90,10,120,30,160,80
12000 REM RECTANGLE PLEIN
12010 ORIGIN XPOS,YPOS
12020 FOR V=0 TO DY STEP 2*SGN(DY)
12030 MOVE 0,V : DRAW DX,V
12040 NEXT
12050 ORIGIN 0,0
12060 RETURN
```

Exercice 4.1

```
1 REM  EXERCICE 4.1
2 REM
3 PRINT "Pour entrer la fonction a tracer"
4 PRINT "tapez 10 DEF FNF(X)= votre fonction Ret
urn"
5 PRINT "puis GOTO 10 Return"
6 END
10 DEF FNF(X)=SIN(X)
20 INPUT "BORNES,T/P";X1,X2,T$
30 H=(X2-X1)/300:XX=-2
40 MY=FNF(X1):YM=MY
50 FOR X=X1 TO X2 STEP H
60 Y=FNF(X)
70 IF Y<MY THEN MY=Y
80 IF Y>YM THEN YM=Y
90 NEXT
100 DEF FND(X)=MY+10+350*(FNF(X)-MY)/(YM-MY)
110 CLG:PLOT 0,FND(X1),1
120 FOR X=X1 TO X2 STEP H
130 XX=XX+2
140 Y=FND(X)
150 IF T$="P" THEN PLOT XX,Y ELSE DRAW XX,Y
160 NEXT
```

Exercice 4.2

```
1 REM  EXERCICE 4.2
2 REM
3 PRINT "Pour entrer les  fonctions a tracer"
4 PRINT "tapez 10 DEF FNF(T)= votre 1ere fonctio
n : DEF FNG(T)= votre 2e fonction Return"
5 PRINT "puis GOTO 10 Return"
6 END
10 DEF FNF(T)=COS(T):DEF FNG(T)=0.5*SIN(T)
20 INPUT "BORNES,T/P";T1,T2,T$
30 H=(T2-T1)/300
40 MX=FNF(T1):XM=MX
45 MY=FNG(T1):YM=MY
50 FOR T=T1 TO T2 STEP H
60 X=FNF(T):Y=FNG(T)
65 IF X<MX THEN MX=X
70 IF Y<MY THEN MY=Y
75 IF X>XM THEN XM=X
80 IF Y>YM THEN YM=Y
90 NEXT
95 PX=600/(XM-MX):PY=350/(YM-MY)
100 P=MIN(PX,PY)
105 DEF FND(T)=MX+10+P*(FNF(T)-MX)
110 DEF FNE(T)=MY+10+P*(FNG(T)-MY)
115 CLG:PLOT FND(T1),FNE(T1),1
120 FOR T=T1 TO T2 STEP H
130 X=FND(T)
140 Y=FNE(T)
150 IF T$="P" THEN PLOT X,Y ELSE DRAW X,Y
160 NEXT
```

Pour garder l'isotropie, il faut appliquer le même facteur d'échelle P dans les deux directions : on applique donc le plus petit des deux facteurs trouvés PX et PY.

Exercice 4.3

Un premier problème est de savoir si les axes X=0 et Y=0 sont dans la figure, auquel cas c'est eux qu'on trace. Sinon, on trace des axes proches du bord supérieur et du bord gauche de l'écran.

Les instructions qui suivent résolvent ce problème :

```
170 REM  EXERCICE 4.3
171 REM
180 AX=MX+10-P*MX:AY=MY+10-P*MY
190 IF AX<0 OR AX>640 THEN AX=5
200 IF AY<0 OR AY>360 THEN AY=5
210 PLOT AX,0:DRAW AX,360
220 PLOT 0,AY:DRAW 640,AY
```

Pour la graduation, le plus simple est d'inscrire les valeurs des maxima et minima de X et Y.

Exercice 4.4

L'emploi du sous-programme défini au chapitre 3. Mais il est incapable de fournir des pointillés.

Exercice 4.5

Il suffit de supprimer l'option de vidage écran dans l'instruction 40 qui deviendrait 40 PLOT .... Avant la première exécution, on ferait un CLG en mode direct.

Exercice 4.6

On peut soit changer l'ordre des sommets, soit remplacer R par R/(R-1).

Exercice 4.7

Le sous-programme de tracé de polygone est remplacé par :

```
200 REM EX 4.7 POLYGONE MULTICOLORE
204 C=1+INT(3*RND(1))
205 PLOT X(1),Y(1),C:FOR I=2 TO N+1
207 C=1+INT(3*RND(1))
210 DRAW X(I),Y(I),C:NEXT:RETURN
```

Exercice 4.8

```
25 MODE 0
200 REM EX 4.8 POLYGONE MULTICOLORE
204 C=1+INT(15*RND(1))
205 PLOT X(1),Y(1),C:FOR I=2 TO N+1
207 C=1+INT(15*RND(1))
210 DRAW X(I),Y(I),C:NEXT:RETURN
```

Exercice 4.9

S'il y a K plans de coupe, il y a K-1 intervalles donc en 45 et en 130, la division doit être par (K-1) et non par K.

Exercice 4.10

Il faut que lorsqu'on regarde la face depuis l'extérieur, la succession M1, M2, M3 semble tourner dans le sens des aiguilles d'une montre. Cela nous fait changer les DATA en 1120, 1150 et 1160.

Voir les changements dans le programme "rotation avec lignes cachées" dans le texte.

Exercice 4.11

Les données seraient :

```
998 REM  EX 4-11
999 REM
1000 DATA 8
1010 DATA 60,60,60
1020 DATA -60,60,60
1030 DATA -60,-60,60
1040 DATA 60,-60,60
1050 DATA 60,60,-60
1060 DATA -60,60,-60
1070 DATA -60,-60,-60
1080 DATA 60,-60,-60
1090 DATA 6
```

```
1100 DATA 4,1,4,3,2
1110 DATA 4,1,2,6,5
1120 DATA 4,2,3,7,6
1130 DATA 4,3,4,8,7
1140 DATA 4,1,5,8,4
1150 DATA 4,5,6,7,8
```

Exercice 5.1

```
1 REM  EX 5.1
2 REM
10 SYMBOL 241,6,15,31,7,3,1,0,0
20 SYMBOL 242,0,0,126,255,254,252,252,56
30 A$=CHR$(241)+CHR$(242)
40 PRINT A$
```

Exercice 5.2

```
1 REM  EX 5.2
2 REM
10 SYMBOL 241,6,15,31,7,3,1,0,0
20 SYMBOL 242,0,0,126,255,254,252,252,56
30 A$=CHR$(241)+CHR$(242)
40 PRINT CHR$(23)+CHR$(1) : TAG :CLG
50 X=0:Y=390:PLOT X,Y,1:PRINT A$;
60 WHILE X<600
70 FOR I=1 TO 2:NEXT
80 MOVE X,Y:PRINT A$;
90 X=X+2:MOVE X,Y:PRINT A$;
100 WEND
```

Exercice 5.3

```
1 REM  EX 5.3
2 REM
10 SYMBOL 241,6,15,31,7,3,1,0,0
20 SYMBOL 242,0,0,126,255,254,252,252,56
30 A$=CHR$(241)+CHR$(242)
40 PRINT CHR$(23)+CHR$(1) : TAG :CLG
50 D=7:X=0:Y=390:PLOT X,Y,1:PRINT A$;
```

```
60 EVERY 2,1 GOSUB 500
70 X$=INKEY$ : IF X$="" GOTO 70
80 Z=REMAIN(1)
90 X$=INKEY$ : IF X$="" GOTO 90
100 GOTO 60
500 REM S/P DE DEPLACEMENT
501 REM
510 MOVE X,Y:PRINT A$;
520 X=X+D:IF X>=600 THEN D=-D
525 IF X<ABS(D) THEN D=-D
530 MOVE X,Y:PRINT A$;
540 RETURN
```

Exercice 5.4

```
1 REM  EX 5.4
2 REM
10 SYMBOL 241,6,15,31,7,3,1,0,0
15 SYMBOL 243,0,0,126,255,127,63,63,30
20 SYMBOL 242,0,0,126,255,254,252,252,56
25 SYMBOL 244,96,240,248,224,192,128,0,0
30 A1$=CHR$(241)+CHR$(242):A2$=CHR$(243)+CHR$(244)
40 PRINT CHR$(23)+CHR$(1) : TAG :CLG
50 A$=A2$:D=7:X=0:Y=390:PLOT X,Y,1:PRINT A$;
60 EVERY 2,1 GOSUB 500
70 X$=INKEY$ : IF X$="" GOTO 70
80 Z=REMAIN(1)
90 X$=INKEY$ : IF X$="" GOTO 90
100 GOTO 60
500 REM S/P DE DEPLACEMENT
501 REM
510 MOVE X,Y:PRINT A$;
520 X=X+D:IF X>=600 THEN D=-D:A$=A1$
525 IF X<ABS(D) THEN D=-D:A$=A2$
530 MOVE X,Y:PRINT A$;
540 RETURN
```

Exercice 6.1

```
1 REM EX 6.1
2 REM
10 READ H,D
20 WHILE H<>0
30 SOUND 1,H,D
40 SOUND 1,H,1,0
50 READ H,D
60 WEND
100 DATA 239,33,239,33,239,33,213,33
110 DATA 190,66,213,66,239,33,190,33
120 DATA 213,33,213,33,239,66
130 DATA 0,0
```

Le procédé est simple : les données qui seront successivement nécessaires sont mises dans un groupe de data. Remarquer aussi le couple de 0 pour marquer la fin.

Exercice 6.2

```
1 REM EX 6.2
2 REM
10 PRINT "Appuyez sur une touche quelconque"
15 PRINT "pour passer a la corde suivante"
20 FOR CORDE=1 TO 6
30 READ H
40 SOUND 1,H,32000
50 A$=INKEY$:IF A$="" THEN 50
60 SOUND 135,H
70 NEXT
100 DATA 95,127,159,213,284,379
```

On spécifie une durée très grande pour chaque note : en fait, l'utilisateur l'interrompra (en appuyant sur une touche) pour passer à la suivante lorsqu'il sera prêt.

Exercice 6.3

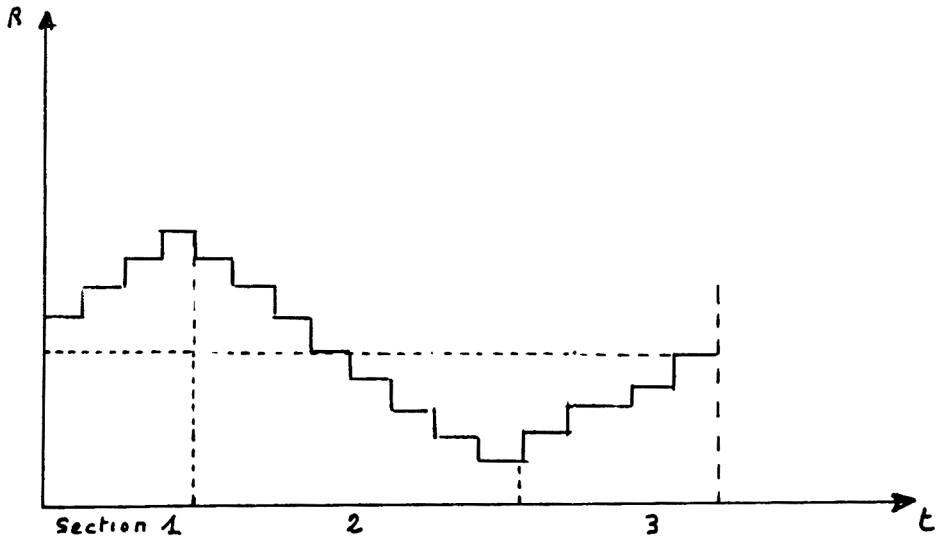
```
1 REM EX 6.3
2 REM
10 SOUND 65,239,130
20 SOUND 66,190,130
30 SOUND 68,159,130
40 RELEASE 7
```

On prépare l'accord en fournissant les données voulues pour chaque voix, avec attente. On déclenche l'accord en libérant les trois voix par RELEASE.

Exercice 6.4

ENV 1,2,7,1,1,0,1,3,-2,1,4,-1,1,4,-1,5

Exercice 6.5



ENT -1,4,1,2,8,-1,2,4,1,2

Exercice 6.6

```
1 REM EX 6.6
2 REM
10 ENT -1,20,2,8,1,-40,1
20 SOUND 1,200,30000,,,,1
```

Exercice 6.7

Il suffit d'appeler deux fois de suite l'interprète, en faisant un RESTORE entre les deux. On peut d'ailleurs effectuer des répétitions de structure bien plus complexe.

Exercice 6.8

On ajoute :

```
695 IF C$="P" THEN FOR T=1 TO 10*D : NEXT : GOTO 640
```

ANNEXE **V**

BIBLIOGRAPHIE

● Sur les instructions de base de Basic :

L'AMSTRAD avec plaisir : comment faire de bons programmes
par R.A. et J.W. PENFOLD ..... EDIMICRO

Périphériques et fichiers sur AMSTRAD
par D-J DAVID ..... PSI

● Sur la musique :

Musique sur Amstrad ..... EDIMICRO

● Sur l'informatique graphique :

Mathématiques et graphismes
par G. GRANPIERRE et R. COTTE ..... PSI

Graphisme scientifique sur Micro-ordinateur
par R. DONY .....Masson

ANNEXE **VI**

INDEX DES PROGRAMMES

Extraits musicaux

LA BIBLIOTHÈQUE EDIMICRO

- ***Collection « Guides professionnels »***

MACINTOSH

Gaucherand ..... **Macintosh : outils, progiciels, applications**

Joutel, Tenin ..... **Bases de données sur Macintosh**

APPLE II, IIe, IIc

de Merly ..... **GUIDE DE L'APPLE**
Tome 1 : l'Apple standard
Tome 2 : les extensions
Tome 3 : les applications

SINCLAIR QL

Tenin, Van Thong ..... **Guide pratique du Sinclair QL**

K et S Brain ..... **Intelligence artificielle sur QL**

- ***Collection « Progiciels »***

IBM PC et compatibles

Bonnet, Dinh ..... **Lotus 1-2-3 à votre portée**

Bonnet, Dinh ..... **Multiplan sur IBM PC : exercices de gestion**

Hœnig ..... **Framework versions 1 et 2 à la portée de tous**

Commandeur ..... **Framework sur IBM PC : exercices de gestion**

Tenin ..... **Archive : bases de données***

Devisscher ..... **Abacus, Easel et Quill : calcul, graphisme et texte***

MACINTOSH

Bouilloux, Galassi ..... **Excel sur Macintosh : exercices de gestion**

Bouilloux ..... **Multiplan et Chart sur Macintosh**

Hornn ..... **Jazz sur Macintosh : exercices de gestion**

Tenin ..... **Bases de données sur Macintosh**

Longevialle ..... **Pascal sur Macintosh**

APPLE II, IIe, IIc

Bonnet, Dinh ..... **Multiplan sur Apple : exercices de gestion**

- ***Collection « Mémentos Progiciels »***

Hornn ..... **Mémento Multiplan**

CMS-Formation ..... **Mémento dBASE III**

Chigard ..... **Mémento R:BASE 5000**

CMS ..... **Mémento MultiMate**

Djama ..... **Mémento Textor**

Hornn ..... **Mémento Symphony 2**

Hornn ..... **Mémento Lotus 1.2.3**

*** Ces ouvrages, qui décrivent les modules d'XCHANGE sur IBM PC, sont également destinés aux utilisateurs du Sinclair QL.**

- *Collection « Micro-ordinateurs et votre profession : applications pratiques »*

Falef ..... **Micro-ordinateurs et professions juridiques**
Anthon, Hornn ..... **Micro-ordinateurs et comptabilité avec dBase III**
Tenin ..... **Micro-ordinateurs et marketing**

- *Collection « Ouvrages de base »*

Lamoitier ..... **Le Traducteur Micro**
Darnis, Van Thong ..... **Graphisme et CAO**
Avon, Rolet ..... **Courbes de maths en Basic**

- *Collection « Ordinateurs familiaux et éducatifs »*

VG 5000
Amsler, Bardon ..... **Guide du VG 5000 Philips**
Amsler, Villemaud ..... **Jeux sur VG 5000 Philips**

MO5
Pitey ..... **Secrets pour MO5, TO7, TO7-70**
Bieber, Perbost, Renucci ..... **Tout sur le MO5**
Perbost, Renucci ..... **Jeux sur MO5**
Bacconnier, Cettier ..... **L'Informatique aide les Maths**

AMIGA
Lawrence, England ..... **L'AMIGA : Fonctionnement et utilisation**

AMSTRAD
Penfold ..... **L'Amstrad avec plaisir**
Waugh ..... **Musique sur Amstrad**

ATARI
Lawrence, England ..... **L'ATARI ST en action**

ATMOS/ORIC 1
Kosniowsky ..... **Nouveaux Jeux sur ATMOS**
Chane-Hune, Darbois ..... **Jeux Graphiques sur ATMOS**
Viguier ..... **Premiers Pas en Programmation**

SPECTRUM
Bridge, Carnell ..... **Aventures sur Spectrum**
Hurley ..... **Jeux Graphiques sur Spectrum**

COMMODORE 64
Fleurier, Meiller ..... **Jeux d'adresse et de hasard**
Fleurier, Meiller ..... **Jeux d'action et de réflexion**

PHILIPS C7420 VIDEOPAC +
Bardon, de Merly ..... **Jeux sur Philips C7420 Vidéopac +**

TO7/TO7-70
— ouvrages
Bieber, Perbost, Renucci ..... **Guide complet et pratique du TO7-70**
Perbost, Renucci ..... **Jeux prêts à l'emploi sur TO7-70**

MSX

Ly ..... **Applications familiales en Basic MSX**
Perbost, Berthet ..... **Introduction à MSX**

GENERAL

Djama ..... **Je construis mon premier robot**
Penfold ..... **Introduction aux périphériques**

- ***Collection « Logiciels éducatifs »***

Série de sept logiciels sur cassette pour l'enseignement primaire. Les programmes ont été développés par l'Association pour la promotion de l'informatique didactique (APID) et par une équipe pluridisciplinaire réunissant des spécialistes du matériel didactique, des instituteurs, et des analystes-programmeurs.
Chaque cassette comporte une face TO7-70 et une face MO5.

ALPHABET
Chaque lettre de l'alphabet s'affiche agrandie sur une grille : reconstituer correctement cette lettre, en pointant au crayon optique les cases à remplir. Référence : AL 1. Une cassette.

BRIC A BRAC
Jeu de logique où l'élève doit assembler les éléments qui déterminent la bonne solution : formes, couleurs, ... Référence BR 1. Une cassette.

COLOR GRILLE
Création d'images simples, à l'aide du crayon optique, en utilisant des grilles de dessin et de couleur. Référence CO 2. Deux cassettes.

CACHE-MOTS
Logiciel de mots croisés évolutifs, sur des grilles préétablies ou créées par l'enseignant : retrouver des mots de même type, exercices grammaticaux. Référence MT 2. Deux cassettes.

MOSAIQUE
Programme de créativité : dessiner des mosaïques, à partir de formes géométriques simples, grâce au crayon optique. Réf. MQ 1. Une cassette.

PUZZLE
Le menu propose un choix sur le nombre de pièces qui composent le puzzle : sept niveaux de difficulté. Référence PU 2. Deux cassettes.

QUESTION
Chaque élément grammatical est affecté d'une couleur (vert pour article, rouge pour sujet, bleu pour COD...). L'élève doit donner la bonne couleur, l'ordinateur affiche en fin d'exercice le nom de l'élève et le nombre de fautes. Référence QU 2. Deux cassettes.

Nombreux autres titres à paraître. Catalogue sur simple demande.

EDIMICRO 121-127, avenue d'Italie, 75013 Paris

EXTRAITS DE PRESSE

EXCEL SUR MACINTOSH

« Bien stucturé, complète parfaitement le manuel d'Excel. Ce qui ressort de ce livre est sa facilité de lecture... il se déroule comme le scénario d'un film. »

L'Echo des Apple.

FRAMEWORK SUR IBM PC : Exercices de Gestion

« Des explications claires et détaillées qui vous permettront de maîtriser rapidement Framework... »

Le Journal de la Commande Electronique.

PROGRAMMEZ EN PASCAL SUR MACINTOSH

« L'auteur a réussi là une véritable 'exploration' du Macintosh... »

01 Informatique hebdo.

« Voilà un livre dense à la fois technique et pédagogue... »

L'Echo des Apple.

« Le livre est toujours clair et écrit dans un style agréable. »

Education et Informatique.

LES MOTS ESSENTIELS DE LA MICRO-INFORMATIQUE

« Un précieux document..., obligatoire pour tout professionnel qui ne veut pas avoir l'air d'une 'pomme' devant un client curieux... »

Vente Informatique.

JAZZ SUR MACINTOSH

« Un livre à la fois clair, pédagogique, technique, et truffé d'exemples d'application. »

Science et Vie Micro.

« Ce livre est conseillé à tous ceux qui voudraient apprendre rapidement à se servir de Jazz... »

L'Echo des Apple.

ARCHIVE, Progiciel de bases de Données

« L'un des meilleurs ouvrages existant sur ce type de logiciel... »

Science et Vie Micro.

COURBES DE MATHS EN BASIC

« Constitue une excellente introduction au graphisme..., pour réaliser de superbes courbes mathématiques. »

Théophile.

MULTIPLAN ET CHART SUR MACINTOSH

« Présente l'incomparable avantage de lier Microsoft Chart et Microsoft Multiplan... »

Microsoft Europe.

« Un outil exemplaire... Les explications sont lumineuses et précises. »

Micro-VO.

BASES DE DONNÉES SUR MACINTOSH

« Un bon guide avant l'achat... »

Science et Vie Micro.

MULTIPLAN SUR IBM PC

« Ce livre rendra de nombreux services. »

OPC.

« Des modèles de qualité professionnelle. »

L'Ordinateur Personnel.

FICHIERS EN BASIC PAR L'EXEMPLE

« Cet ouvrage est recommandé aux personnes amenées à entreprendre la programmation des fichiers du Basic, pour lesquels il constitue une étude pratique, sérieuse et en profondeur. »

Ordi Magazine.

GUIDE DE L'APPLE

« Un des meilleurs sur la place. »

Le Figaro.

« Le livre que nous attendions : complet, clair et pratique. »

Apple France.

Nombreux autres titres à paraître. Catalogue sur simple demande.

EDIMICRO 121-127, avenue d'Italie, 75013 Paris

EXTRAITS DE PRESSE

INTRODUCTION AUX PÉRIPHÉRIQUES D'ORDINATEURS

« Explique avec clarté en quoi consistent les périphériques pour micro-ordinateurs... »

01 Informatique.

« Un guide pratique particulièrement concis et précis... très facile à lire »

Théophile.

JE CONSTRUIS MON PREMIER ROBOT

« Ce genre de livre est trop rare... Croyez-moi, c'est exceptionnel et 'à saisir'. »

L'Ordinateur individuel.

APPLICATIONS FAMILIALES EN BASIC MSX

« Une bonne initiation professionnelle aux micros. »

Electronique Technique et Industrie.

« Un véritable cours de programmation en Basic... à retenir lorsque l'on est un 'débutant éclairé' et que l'on souhaite comprendre plus à fond. »

MSX-magazine.

LE TRADUCTEUR MICRO

« Un très bon ouvrage... quant au contenu et à la présentation. »

Ordimagazine.

JEUX SUR VG 5000 PHILIPS

« Ouvrage bien écrit, structuré et clair... »

Microfer.

TOUT SUR LE MO5

« Un livre consistant, original et rassurant dans sa démarche... »

Science et Vie Micro.

INTRODUCTION A MSX

« Un bon ouvrage de base, agrémenté d'exemples utiles. »

Interface-Microfer.

GUIDE PRATIQUE DU SINCLAIR QL

« Une somme impressionnante d'informations, des programmes nombreux et intéressants... Un bon outil d'initiation à la pratique de notre Sinclair QL. »

Direco International-Sinclair.

« Cet ouvrage comblera tous les possesseurs de l'un des meilleurs ordinateurs du marché. »

Science et Vie Micro.

GUIDE DU TO7

« Ce guide a le privilège d'être présenté de façon claire, dans un style parfaitement accessible à tous. »

Micro 7.

« Un manuel de référence absolue. »

Le Figaro.

JEUX SUR TO7

« De nombreux conseils et 'trucs' pour une programmation rapide et efficace. »

Décision Informatique.

« Un excellent livre d'approche, plein de renseignements utiles. »

Micro 7.

JEUX SUR COMMODORE 64

« Cet ouvrage permet aux débutants de s'initier au langage Basic en s'amusant... Un livre d'un bon niveau. »

Tilt.

PREMIERS PAS EN PROGRAMMATION

« Une approche originale dans la littérature consacrée aux micros. »

Décision Informatique.

Nombreux autres titres à paraître. Catalogue sur simple demande.

EDIMICRO 121-127, avenue d'Italie, 75013 Paris

Achevé d'imprimer en octobre 1986
sur les presses de l'imprimerie Laballery — 58500 Clamecy
Dépôt légal : octobre 1986
Numéro d'impression : 609061

PEINTRE ET MUSICIEN sur AMSTRAD

Daniel-Jean DAVID

CPC 464
CPC 664
CPC 6128

Pour tous les passionnés, qui veulent utiliser pleinement les ressources graphiques et sonores de leur AMSTRAD : des programmes, des exemples précis, de nombreuses illustrations et figures, des explications claires.

- **Les modes graphiques**
- **Les graphiques de gestion :** courbes, diagrammes en bâtons, *« camemberts »*.
- **Les graphiques artistiques et scientifiques :** dessin en trois dimensions, perspectives et rotations.
- **Les lutins :** objets graphiques déplaçables.
- **Les sons :** bruits, imitation d'instruments, partitions...

Dans la même série :

Peintre et musicien sur C128, Référence 64-X,
Prix indicatif : 148 F.

ISBN : 2-904457-63-1

148 F

9 782904 457630

Edimicro
121-127, Avenue d'Italie 75013 Paris

PEINTRE ET MUSICIEN SUR AMSTRAD

D. J. DAVID

Document numérisé
avec amour par :

AMSTRAD
CPC

MÉMOIRE ÉCRITE

http://amstradcpc.fredisland.net/