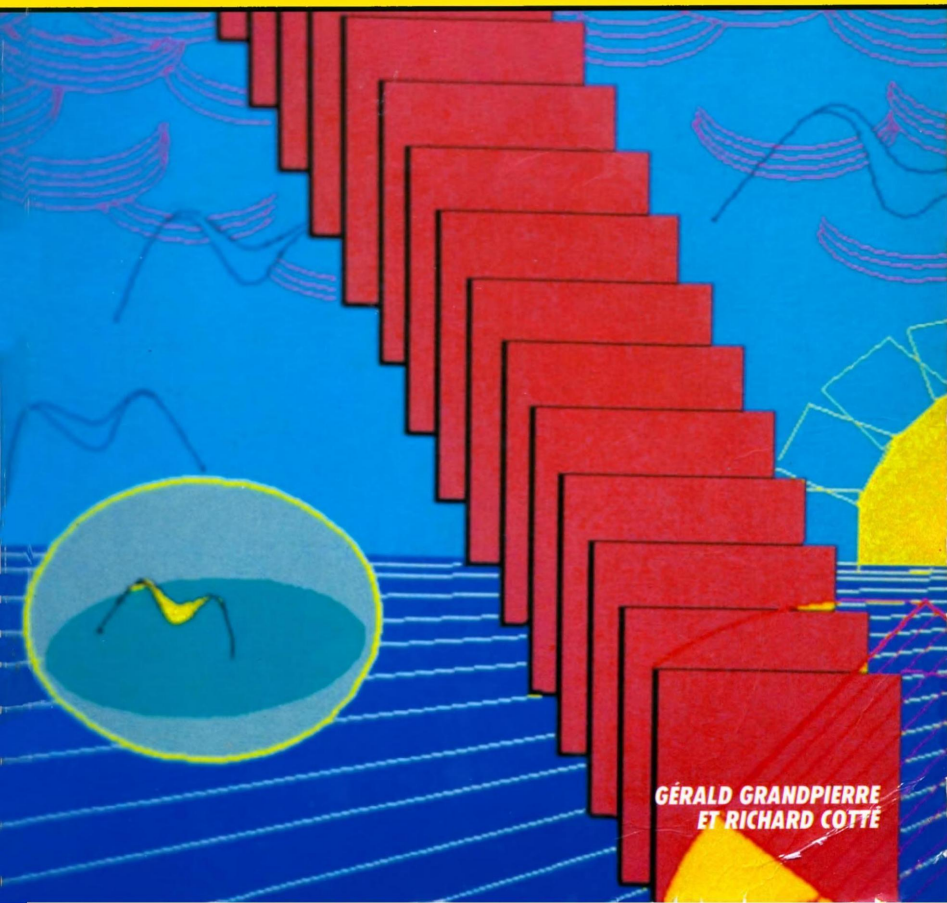
MICRO POUR L'ÉCOLE

MATHEMATIQUES ET GRAPHISMES



GÉRALD GRANDPIERRE
ET RICHARD COTTÉ

MATHÉMATIQUES ET GRAPHISMES

Autres ouvrages d'application scolaire

— Physique en poche, tomes 1 et 2 - Yves Dao-Léna.
— Bac-Basic, tome 1 - Yves Dao-Léna.
— Méca-Basic - Thierry Tacquet et Francis Legroux.
— Électro-Basic - Claude Nowakowski.

A paraître

— Bac-Basic, tomes 2 et 3 - Yves Dao-Léna.
— Compta-Basic - Gaston Miclot.

Autres ouvrages sur le même sujet

Certains de ces ouvrages disposent d'une disquette d'accompagnement *.
— * L'Apple et ses fichiers - Jacques Boisgontier.
— * Modèles d'expression graphique - Jean-Pierre Blanger.
— * Pangraphe - Dessins en 3 dimensions - Jean-Pierre Petit.

Ouvrages d'initiation sur Apple

— Apple pour tous - Jacques Boisgontier et Sophie Brébion.
— 36 programmes Apple II pour tous - Jacques Boisgontier.
— Exercices pour Apple II - Frédéric Lévy.
— La découverte de l'Apple II - Dominique Schraen et Frédéric Lévy.
— La pratique de l'Apple II, tomes 1 et 2 - Nicole Bréaud-Pouliquen.
— La pratique de l'Apple II, tome 3 - Nicole Bréaud-Pouliquen et Daniel-Jean David.
— 102 programmes pour Apple II - Jacques Deconchat.

Ouvrages d'initiation sur MO5 et TO7

— MO5 et TO7/70 pour tous - Jacques Boisgontier.
— Exercices pour MO5, Dominique Schraen et Maurice Charbit.
— Exercices pour TO7/70, Maurice Charbit et Dominique Schraen.
— La découverte du MO5, Dominique Schraen et Maurice Charbit.
— La découverte du TO7, Dominique Schraen et Maurice Charbit.

© *Éditions du P.S.I., B.P. 86, 77402 Lagny Cedex, 1985*
ISBN : 2-86595-209-6

GÉRALD GRANDPIERRE
ET RICHARD COTTÉ

MATHÉMATIQUES ET GRAPHISMES

ÉDITIONS DU P.S.I.
1985

Professeurs de Mathématiques et passionnés d'informatique individuelle depuis l'avènement des micros, **Richard Cotté** et **Gérald Grandpierre** aiment mieux calculer la dix millième décimale de π que gérer leur budget familial, contempler des courbes intégrales que faire du traitement de texte.

Pédagogues, ils s'intéressent à tous les aspects de l'E.A.O., à l'introduction de l'informatique dans la pédagogie et ce, à tous les niveaux : des classes de maternelle, pour lesquelles Gérald Grandpierre a réalisé des logiciels éducatifs, aux classes de terminale, où l'informatique permet de renouveler l'enseignement des Mathématiques.

A Christine et Martine

micro-martyres

Sommaire général

TROISIEME PARTIE - ESPACE

Avant-propos

Ce livre s'adresse à tous les possesseurs d'un micro-ordinateur doté de possibilités graphiques et qui veulent en tirer le meilleur parti.

Nous avons délibérément choisi :

- de ne pas privilégier un matériel particulier.
  Chaque ordinateur a son dialecte "Basic". Dans le domaine graphique, il s'agit d'un patois. Refusant l'esprit de chapelle, nous avons rédigé des programmes facilement adaptables à tout ordinateur capable de visualiser des points et des droites (pour le graphisme, c'est tout de même le minimum !).
- d'expliquer simplement les méthodes utilisées.
  Nous sommes convaincus que tout lecteur connaissant un peu de mathématiques - quelques connaissances de niveau BAC sont requises - peut comprendre les méthodes utilisées.

Ces choix ont des conséquences :

- tous les programmes de ce livre sont à adapter à votre matériel. Cela représente, de votre part, cher lecteur, un certain effort qui sera, nous l'espérons, récompensé. Le terme "certain effort" est impropre. Il s'agit, lorsque nous écrivons :

  ```
  REM EFFACAGE D'ECRAN
  ```

  de remplacer cette remarque par votre HOME, CLS, HCLS numéro de couleur ou PRINT CHR$( )... enfin, d'écrire l'instruction D'EFFACAGE D'ECRAN de votre ordinateur. Le détail de ces adaptations est donné page 13 ;

- souhaitant rester simples et accessibles à un large public, nous n'avons pas toujours choisi les meilleures méthodes possibles. Ainsi, dans le chapitre "Systèmes différentiels", la méthode dite de RUNGE-KUTTA fournit de meilleurs résultats que celle d'EULER. Mais elle nous a semblé trop délicate à exposer. De même, dans le chapitre "Surfaces", nous avons axé notre étude sur l'algorithme partie vue, partie cachée. Nous aurions pu choisir de placer l'observateur en n'importe quel point de l'espace. Cela aurait entraîné une difficulté supplémentaire (maniement d'angles...) : nous avons préféré écarter un tel point de vue.

En un sens, ce livre n'est qu'un point de départ :

- nous avons choisi d'illustrer les méthodes par de nombreux exemples. Malgré le choix de méthodes que nous jugeons simples, nous sommes

conscients que des lecteurs ne connaissent pas certains outils mathématiques (méthode du pivot, séries de Fourier, méthode d'Euler, produit vectoriel...). Nous les avons redéfinis... mais cela risque de ne pas suffire.

Aussi, toutes les applications étudiées sont illustrées par de nombreux exemples. Si l'aspect mathématique vous rebute un peu, ce livre vous donnera, nous l'espérons, beaucoup à voir.

Les fractals, les courbes, leurs approximations, les surfaces, les cubes, dodécaèdres, icosaèdres qui tournent ... échappent au strict domaine des mathématiques et touchent à l'esthétisme.

En ce sens, ce livre est une introduction à un univers étrange et merveilleux, trop souvent réservé aux seuls matheux ... Laissez-vous guider et admirez.

Lexique

Les Basics possédant des instructions graphiques assez différentes les unes des autres, voici un petit lexique qui doit vous permettre d'adapter facilement les programmes de ce livre à votre ordinateur :

REM EFFACAGE D'ECRAN : c'est clair !
REM MODE TEXTE : écriture d'un texte.
REM MODE GRAPHIQUE : passage au mode graphique.

Vous utiliserez ces instructions si votre ordinateur ne permet pas l'affichage simultané du texte et du graphisme. S'il l'autorise, n'en tenez pas compte. Veillez à ce que le texte ne détruise pas le graphisme.

POINT X,Y : affiche le point de coordonnées X,Y sur l'écran.
DROITE X1,Y1 A X2,Y2 : trace le segment joignant les points de coordonnées (X1,Y1) et (X2,Y2) sur l'écran.

Dans un chapitre (surfaces en Z^2), nous aurons besoin d'effacer un segment. Nous avons noté :

DROITE ... A ... EFFACE

Cela nécessite assez souvent le tracé du segment dans la couleur de fond de l'écran.

Enfin, certaines variables ont un sens très précis dans tous les programmes :

LG : longueur de votre écran graphique (abscisses des points de Ø à LG compris, orientation vers la droite).

HT : hauteur de votre écran graphique (ordonnées des points de Ø à HT compris, orientation vers le bas).

XC : abscisse du centre de l'écran. Prendre XC=LG/2 ou INT(LG/2).

YC : ordonnée du centre de l'écran. Prendre YC=HT/2 ou INT(HT/2).

PI : le nombre célèbre 3.1415926535...

KE : facteur important (voir première partie, chapitre 2) permettant de tenir compte de la différence d'unités entre les deux axes de votre écran (voir son calcul en page 50).

Vous le voyez, voici un "certain effort" d'adaptation qui doit être à votre portée.

Adaptations tous matériels

Sur les matériels les plus répandus, voici comment doivent être réalisées ces adaptations.

APPLE II

Les instructions TEXT et HGR permettent d'accéder soit au mode TEXTE, soit au mode GRAPHIQUE Haute résolution. On aura intérêt à utiliser la page graphique 280 x 160 qui permet d'utiliser les quatre dernières lignes de l'écran pour du texte.

```
LG=279  HT=159 ou 191 : selon la page utilisée.
HCOLOR                : trace dans la couleur choisie.
HPLOT X,Y             : affiche un point.
HPLOT X1,Y1 TO X2,Y2  : trace une droite.
HOME                  : efface l'écran.
```

COMMODORE 64

En version de base, les ordres graphiques n'existent pas. La cartouche TOOL permet d'en disposer.

Les instructions GRAPHIC et TEXT permettent d'accéder au graphisme ou au texte. DISPLAY autorise l'affichage de textes sur l'écran graphique.

Avec cette cartouche, l'origine de l'écran graphique est en bas, à gauche de l'écran : des règles de trois sont à reprendre, des signes - tenant compte de l'orientation de l'axe Y vers le bas - sont à modifier...

```
LG=319  HT=199
MOVE X,Y              : déplace le curseur graphique.
PLOT X,Y,1 ou 0       : trace ou efface un point.
DRAW X,Y,1 ou 0       : trace ou efface une droite de la position du
                        curseur au point X,Y.
SCLEAR                : efface l'écran.
```

DRAGON

Le texte et le graphisme ne sont pas mixables.

```
SCREEN 0,0            : donne le mode texte.
CLS                   : efface le texte.
```

On accède à la haute résolution par PMODE 4,1:SCREEN 1,1 (entre autres).

```
LG=255  HT=191
PCLS                    : efface l'écran graphique.
PSET (X,Y)              : affiche un point.
PRESET (X,Y)            : éteint un point.
LINE (X1,Y1)-(X2,Y2),PSET   : trace une droite.
LINE (X1,Y1)-(X2,Y2),PRESET : efface une droite.
```

ORIC

TEXT permet d'accéder au mode TEXTE et HIRES au mode HAUTE-RESOLUTION. En haute résolution, il reste trois lignes, en bas de l'écran, pour du texte : c'est en général suffisant.

```
LG=239  HT=199
CURSET X,Y,Ø ou 1       : place le curseur ou point X,Y avec ou sans
                          affichage.
DRAW X,Y,Ø ou 1         : permet de tracer une droite de la position
                          du curseur à la position augmentée de X et Y.
```

Notre instruction droite X1,Y1 à X2,Y2 est donc à remplacer par :

```
CURSET X1,Y1,1:DRAW X2-X1,Y2-Y1,1
```

TO7-MO5

TEXTE et GRAPHISME sont mixables. L'instruction CONSOLE permet de définir une fenêtre de travail. Cela peut être utile afin de réserver quelques lignes de texte.

```
CLS                     : efface l'écran.
LG=319  HT=199          : si on utilise tout l'écran.
PSET (X,Y),COULEUR      : affiche ou efface un point.
LINE (X1,Y1)-(X2,Y2),COULEUR : trace ou efface une droite.
```

TRS8Ø

Dans la version de base, TEXTE et GRAPHISME sont mixables.

```
CLS                     : efface l'écran.
LG=127  HT=47
SET(X,Y)                : affiche un point.
RESET(X,Y)              : efface un point.
```

Le Basic ne comporte pas d'instruction de tracé de droite mais, dans la littérature consacrée à ce matériel, on trouve facilement comment pallier ce manque.

Il existe des hautes résolutions adaptables à ce matériel. Les instructions graphiques et les dimensions de l'écran sont trop variables pour que nous puissions entrer dans le détail. Mieux vaut consulter la documentation.

Nous ne pouvons pas donner d'indication sur la valeur du coefficient KE. Pour un même matériel, il dépend du moniteur utilisé : à vous de le mesurer.

Que les possesseurs de matériels non cités ne nous en veuillent pas. Votre ordinateur est excellent, c'est même sans doute le meilleur. Jamais de sa vie, on ne l'oubliera, le premier ordinateur qu'on a pris dans ses bras ! Nous n'avons ni la vocation, ni le temps de tester tous les matériels disponibles. Comme de plus, vous connaissez bien mieux votre ordinateur que nous, vous êtes le plus qualifié pour y adapter nos programmes.

Première partie
DROITES

Chapitre I
Déformations, enveloppes

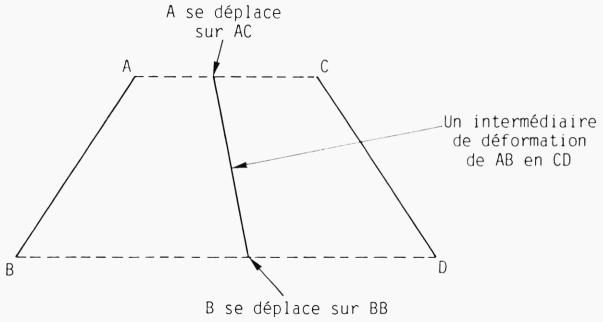
DEFORMATIONS

Si vous êtes un peu comme nous - et nous avons déjà en commun un certain goût pour l'informatique et le graphisme - vous vous êtes sans doute demandé comment passer imperceptiblement d'une figure à une autre... Si imperceptiblement même qu'entre les figures, initiale et finale, il n'y a presqu'aucun rapport !

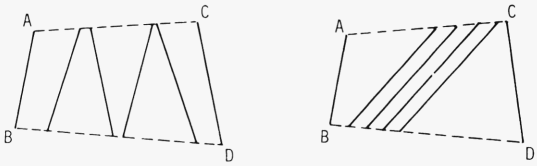
Entre différentes méthodes possibles, nous en avons choisi une, très simple, donnant d'assez bons résultats. Elle repose sur la possibilité de déformer un segment en un autre.

La méthode utilisée

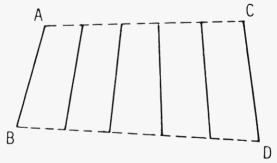
Comment donc déformer un segment AB en un segment CD ? A devant être déformé en C et B en D, le plus simple est qu'au cours des différentes étapes de déformation, A se déplace sur AC et B sur BD :



Mais où placer les "intermédiaires de déformation" ? Sur AC et BD ? Avec quatre intermédiaires, les déformations ci-dessous risquent de paraître un peu bizarres :



Il est souhaitable que les positions successives soient régulièrement espacées sur AC et BD. Toujours avec quatre déformations, on obtient la figure :



qui produira l'effet désiré car AC et BD ont été divisés en cinq segments égaux : la déformation est progressive et régulière.

Récapitulons : pour déformer un segment AB en un segment CD, avec N déformations intermédiaires, il suffit de diviser les segments AC et BD en N+1 segments égaux (A A_1, A_1A_2... A_NC et BB_1, B_1B_2..., B_ND). Les déformations successives de AB en CD sont les segments A_1 B_1, A_2 B_2..., A_N B_N.

Si A, B, C, D ont pour coordonnées (XA,YA),(XB,YB),(XC,YC),(XD,YD) :

A_i a pour coordonnées $XA + i \times \frac{XC - XA}{N+1}$, $YA + i \times \frac{YC - YA}{N+1}$

B_i a pour coordonnées $XB + i \times \frac{XD - XB}{N+1}$, $YB + i \times \frac{YD - YB}{N+1}$

A et B correspondent donc à i = Ø, C et D à i = N+1 ; on peut écrire A = AØ, B = BØ, C = A_{N+1}, D = B_{N+1}.

Le *petit programme ci-dessous* vous permettra de déformer AB en CD sur votre matériel favori.

```
10 REM DEFORMATION D'UN SEGMENT EN UN AUTRE
20 LG=   :HT=
30 REM MODE TEXTE
40 INPUT"COORDONNEES DU POINT A ";XA,YA
50 IF XA<0 OR XA>LG OR YA<0 OR YA>HT THEN 40
60 INPUT"COORDONNEES DU POINT B ";XB,YB
70 IF XB<0 OR XB>LG OR YB<0 OR YB>HT THEN 60
80 INPUT"COORDONNEES DU POINT C ";XC,YC
90 IF XC<0 OR XC>LG OR YC<0 OR YC>HT THEN 80
100 INPUT"COORDONNEES DU POINT D ";XD,YD
110 IF XD<0 OR XD>LG OR YD<0 OR YD>HT THEN 100
120 INPUT"NOMBRE DE DEFORMATIONS";N
130 IF N<=0 THEN 120
```

→

```
140 REM MODE GRAPHIQUE
150 REM EFFACAGE D'ECRAN
160 FOR I=0 TO N+1
170 DROITE XA+I*(XC-XA)/(N+1),YA+I*(YC-YA)/(N+1)
     A   XB+I*(XD-XB)/(N+1),YB+I*(YD-YB)/(N+1)
180 NEXT I
```

Les "matheux" reconnaîtront une propriété bien connue lorsque a varie de 0 à 1, le barycentre des points A,1-a et C,a décrit le segment AC. Il le décrit même continûment, ce qui assure une déformation régulière.

Ici, $a = \frac{N+1-i}{N+1}, 1-a = \frac{i}{N+1}$.

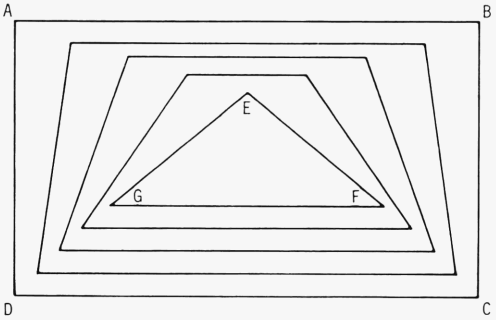
Elémentaire cette méthode, non ? Mais cela vous laisse peut-être sur votre faim : on annonce des déformations de figures et on ne vous donne que des segments. Patience, les déformations de figures vont arriver ; encore faut-il définir une "figure".

Introduction d'une figure

Nous allons maintenant définir les figures à déformer. Il nous faut deux figures. Baptisons-les de départ et d'arrivée. Elles seraient toutes deux formées de segments. Pourquoi uniquement de segments ? Pour deux raisons :

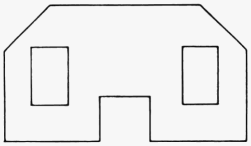
- déformer un segment en un autre est facile à réaliser : nous venons de le voir ;
- toute courbe, aussi compliquée soit-elle, peut être considérée comme formée de segments assez petits.

Les figures de départ et d'arrivée seront en plus définies par le même nombre S de segments. Cela peut sembler être une contrainte assez gênante, trop restrictive... A première vue, seulement. Comme un segment peut être déformé en un point (on peut très bien prendre C=D dans le paragraphe expliquant la méthode utilisée), on pourra déformer un rectangle ABCD en un triangle EFG en déformant AB en EE, BC en EF, CD en FG et DA en GE :

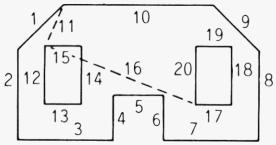


Il importe de bien comprendre que si, a priori, les motifs de départ et d'arrivée n'ont pas le même nombre de segments, certains segments seront réduits à des points.

De plus, parmi les segments définissant les figures, certains seraient vus, d'autres cachés. Supposons qu'on veuille déformer une maison ainsi tracée...



en n'importe quoi... cela n'a pas d'importance. Si nous définissons un segment par les coordonnées de ses deux extrémités, comme chaque point appartient à deux segments, ses coordonnées seraient comptées deux fois. Cette façon de procéder est assez maladroite. Il est préférable de définir cette maison par des segments consécutifs, vus ou cachés :



—— segments vus
(1 à 10, 12 à 15, 17 à 20)

---- segments cachés
(11, 16)

Il convient de bien remarquer que les segments vus sur la figure de départ se déforment en segments vus sur la figure d'arrivée ... éventuellement réduits à un point, mais vus. Imaginez un instant qu'un segment caché se déforme en un segment vu, ou l'inverse : nous n'aurions pas appelé ce chapitre Déformations, mais réincarnation ou désintégration !

Compte tenu de tout cela, nous devrons introduire :

- S **nombre total de segments**, vus ou cachés, éventuellement réduits à un point, des figures de départ et d'arrivée ;

- **la nature du segment** : vu ou caché. Nous l'appellerons "couleur" du segment. Ne vous inquiétez pas si vous n'avez pas de couleur sur votre matériel. Nous utilisons ce terme uniquement parce que, sur la majeure partie des ordinateurs, effacer un segment c'est le tracer de la couleur du fond de l'écran. Nous conviendrons que :

  couleur = 1 si le segment est vu ;
  couleur = 0 si le segment est caché.

Les "couleurs" des segments seront gérées dans un tableau COU( ).

- **les coordonnées** X(,),Y(,) des segments qui seront stockées dans des tableaux à deux indices :

	Motif de départ	*Motif d'arrivée*
Segment n° 1	X(Ø,Ø),Y(Ø,Ø) X(1,Ø),Y(1,Ø)	X(Ø,1),Y(Ø,1) X(1,1),Y(1,1)
Segment n° 2	X(1,Ø),Y(1,Ø) X(2,Ø),Y(2,Ø)	X(1,1),Y(1,1) X(2,1),Y(2,1)
Segment n° i	X(i-1,Ø),Y(i-1,Ø) X(i,Ø),Y(i,Ø)	X(i-1,1),Y(i-1,1) X(i,1),Y(i,1)
Segment n° S	X(S-1,Ø),Y(S-1,Ø) X(S,Ø),Y(S,Ø)	X(S-1,1),Y(S-1,1) X(S,1),Y(S,1)

Il s'agit, bien entendu, de coordonnées-écran (les X entre Ø et LG, les Y entre Ø et HT).

Une fois définie la façon dont les figures à déformer seront gérées par l'ordinateur, il nous faut préciser la façon dont ces figures seront introduites.

Bien sûr, on peut mettre S, les coordonnées, les "couleurs" en data et les faire lire... Cela ne permettra pas une grande variété de figures.

Nous allons imposer deux contraintes permettant une introduction facile et interactive des figures :

- tout segment vu est immédiatement visualisé à l'écran ;
- tout segment peut être modifié si l'utilisateur juge qu'il l'a mal placé. Par exemple, introduire un négatif pour X et Y autorisera la modification des coordonnées introduites juste avant. La modification sera visualisée aussitôt.

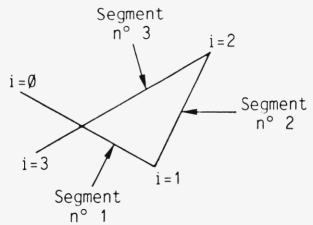
Voilà un programme répondant à tous ces critères. Une seule différence : il ne permet que l'introduction d'une seule figure, les tableaux X( ) et Y( ) n'ont donc qu'une seule dimension. Il pourra être utilisé comme sous-programme dans de nombreux programmes. Tapez-le, utilisez-le pour vous familiariser avec ce mode d'introduction d'une figure. En le tapant, ne vous étonnez pas de quelques "bizarreries" (tableaux dimensionnés à S+1 ...). Il ne s'agit pas d'erreurs d'impression : voyez les commentaires qui suivent ce programme.

```
50 LG=  :HT=
100 REM MODE TEXTE
110 INPUT"NOMBRE DE SEGMENTS ";S
120 IF S<=0 OR S<>INT(S) THEN 110
130 DIM X(S+1),Y(S+1),COU(S+1)
140 COU(0)=0
200 REM INTRODUCTION DES POINTS
210 FOR I=0 TO S+1
220 IF I=0 THEN GOSUB 400:GOTO 310
230 REM MODE TEXTE
240 PRINT"POINT D'ARRIVEE,SEGMENT NUMERO ";I
250 IF COU(I-1)=1 AND I<S THEN GOSUB 500 :
    GOTO 270
260 COU(I)=1
270 INPUT"COORDONNEES (X,Y)";X,Y
280 IF X>LG OR Y>HT THEN 270
290 IF X<0 OR Y<0 THEN GOSUB 800:GOTO 220
300 IF I<>S+1 THEN GOSUB 600
310 NEXT I
320 END
400 REM PREMIER POINT
410 REM MODE TEXTE
420 PRINT"POINT DE DEPART"
430 INPUT"COORDONNEES (X,Y)";X,Y
440 IF X>LG OR Y>HT OR X<0 OR Y<0 THEN 430
450 X(0)=X:Y(0)=Y
460 REM MODE GRAPHIQUE
470 POINT X(0),Y(0)
480 RETURN
500 REM COULEUR DU SEGMENT
510 INPUT"COULEUR (1=VU,0=CACHE)";COU(I)
520 IF COU(I)<>0 AND COU(I)<>1 THEN 510
530 RETURN
600 REM MISE A JOUR
610 X(I)=X:Y(I)=Y                                    →
```

```
620 D=I-1 GOSUB 900
630 RETURN
800 REM RECTIFICATION DE SEGMENTS
810 IF I=0 THEN RETURN
820 I=I-1
830 REM EFFACAGE D'ECRAN
840 IF I=0 THEN RETURN
850 REM MODE GRAPHIQUE
860 IF I=1 THEN POINT X(0),Y(0):RETURN
870 D=I-2:GOSUB 900
880 RETURN
900 REM TRACE DES SEGMENTS
910 REM MODE GRAPHIQUE
920 FOR J=0 TO D
930 IF COU(J+1)=0 THEN POINT X(J+1),Y(J+1) :
    GOTO 950
940 DROITE X(J),Y(J) A X(J+1),Y(J+1)
950 NEXT J
960 RETURN
```

Un certain nombre de points de ce programme nécessitent des commentaires :

- pourquoi les tableaux sont-ils dimensionnés à **S+1** et non pas à **S**, ce qui semblerait plus normal ?
  Pour une raison assez simple : un point peut être modifié lorsqu'on introduit un négatif pour l'une des coordonnées du point suivant... Pour pouvoir modifier le dernier point introduit, d'indice S, il faut considérer le "der des ders", d'indice S+1, qui ne sert qu'à une éventuelle modification ;
- le premier et le dernier segment sont forcément vus, sinon il faut modifier **S**. De même, un segment qui suit un segment caché est vu. Les lignes 250 et 260 ne demandent la couleur que dans les cas utiles et le **COU(Ø)=Ø** va imposer au premier segment d'être vu ;
- si l'une des coordonnées introduites pour un segment est négative, le segment qui vient d'être introduit doit être effacé... Mais cet effaçage peut affecter un segment que l'on souhaite garder et qu'il faut donc réafficher :



Si pour i=4 on introduit un négatif, le segment 3 est effacé ainsi qu'un point du segment 1.

Dans ce cas, nous avons choisi une solution radicale : tout l'écran est effacé et les segments conservés sont réaffichés. Ainsi, les "couleurs", en tant que couleurs, ne jouent aucun rôle : seuls les segments vus sont affichés, les autres ne sont pas pris en compte ...ce qui évite de les effacer en les traçant dans la couleur de fond de l'écran.

Une dernière remarque au niveau de la programmation. Dans le sous-programme "Rectification" (lignes 800 à 880), l'indice **i** de la boucle principale est modifié : cela est fortement déconseillé en programmation. Malgré les risques, il nous est apparu assez normal de modifier cet indice

puisque le nombre de segments introduits varie selon les choix de l'utilisateur.

Nous avons maintenant tous les outils qui vont nous permettre de déformer une figure en une autre.

Déformation d'une figure en une autre

Nous devons :

- introduire les motifs de départ et d'arrivée, comme nous venons de le voir ;
- indiquer le nombre de déformations voulues et déformer chaque segment du motif de départ en un segment correspondant du motif d'arrivée. Les XA, YA, XB, YB du paragraphe "méthode utilisée" seront remplacés par des quantités plus compliquées, des X(i,J), Y(i,J) et il y aura une boucle sur l'indice **i**. Ces modifications sont normales et peu compliquées.

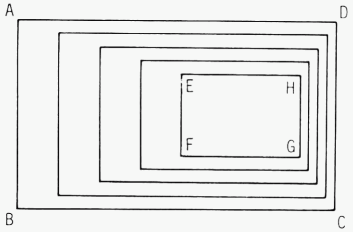
Le programme principal est donc constitué de deux sous-programmes : l'introduction des motifs et la déformation proprement dite.

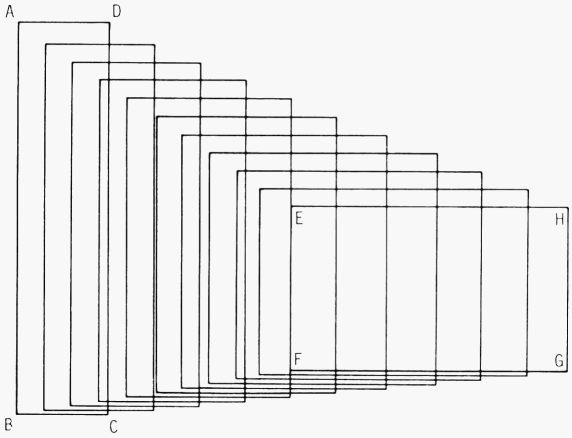
```
40 LG=    :HT=
100 GOSUB 1000
110 GOSUB 2000
120 END
1000 REM INTRODUCTION D'UNE FIGURE
1010 REM MODE TEXTE,EFFACAGE D'ECRAN
1020 INPUT"NOMBRE DE SEGMENTS";S
1030 IF S<=0 OR S<>INT(S) THEN 1020
1040 DIM X(S+1,1),Y(S+1,1),COU(S+1)
1050 COU(0)=0:D=0
1100 REM INTRODUCTION DES POINTS
1110 FOR J=0 TO 1
1120 REM EFFACAGE D'ECRAN
1130 FOR I=0 TO S+1
1140 REM MODE TEXTE
1150 IF J=0 THEN PRINT"FIGURE DE DEPART  ";
     GOTO 1170
1160 PRINT"FIGURE D'ARRIVEE  ";
1170 IF I=0 THEN GOSUB 1300:GOTO 1260
1180 PRINT"POINT D'ARRIVEE,SEGMENT NUMERO";I
1190 IF J=1 THEN 1220
1200 IF COU(I-1)=1 AND I<S THEN GOSUB 1400 :
     GOTO 1220
1210 COU(I)=1
1220 INPUT"COORDONNEES (X,Y)";X,Y
1230 IF X>LG OR Y>HT THEN 1220
1240 IF X<0 OR Y<0 THEN GOSUB 1600:GOTO 1140
1250 IF I<>S+1 THEN GOSUB 1500
1260 NEXT I
1270 NEXT J
1280 RETURN
1300 REM PREMIER POINT
1310 REM CLS
1320 PRINT"POINT DE DEPART"
1330 INPUT"COORDONNEES (X,Y)";X,Y
1340 IF X<0 OR Y<0 OR X>LG OR Y>HT THEN 1330
1350 X(0,J)=X:Y(0,J)=Y
1360 REM MODE GRAPHIQUE
1370 POINT X(0,J),Y(0,J)
1380 RETURN                                    →
```

```
1400 REM COULEUR
1410 INPUT"COULEUR (1=VU,0=CACHE)";COU(I)
1420 IF COU(I)<>0 AND COU(I)<>1 THEN 1410
1430 RETURN
1500 REM MISE A JOUR
1430 RETURN
1500 REM MISE A JOUR
1510 X(I,J)=X:Y(I,J)=Y
1520 D=I-1:GOSUB 1700
1530 RETURN
1600 REM RECTIFICATION
1610 IF I=0 THEN RETURN
1620 I=I-1
1630 REM EFFACAGE D'ECRAN
1640 IF I=0 THEN RETURN
1650 REM MODE GRAPHIQUE
1660 IF I=1 THEN POINT X(0,J),Y(0,J):RETURN
1670 D=I-2:GOSUB 1700
1680 RETURN
1700 REM TRACE DES SEGMENTS
1710 REM MODE GRAPHIQUE
1720 FOR K=0 TO D
1730 IF COU(K+1)=0 THEN POINT X(K+1,J),Y(K+1,J) :
     GOTO 1750
1740 DROITE X(K,J),Y(K,J) A X(K+1,J),Y(K+1,J)
1750 NEXT K
1760 RETURN
2000 REM DEFORMATIONS
2010 REM MODE TEXTE
2020 INPUT"NOMBRE DE DEFORMATIONS";N
2030 IF N<0 THEN RETURN
2035 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
2040 FOR I=0 TO N+1
2050 FOR J=1 TO S
2060 IF COU(J)=0 THEN 2080
2070 DROITE X(J-1,0)+I*(X(J-1,1)-X(J-1,0))/(N+1),
     Y(J-1,0)+I*(Y(J-1,1)-Y(J-1,0))/(N+1) A
     X(J,0)+I*(X(J,1)-X(J,0))/(N+1) ,
     Y(J,0)+I*(Y(J,1)-Y(J,0))/(N+1)
2080 NEXT J
2090 NEXT I
2100 GOTO 2020
```

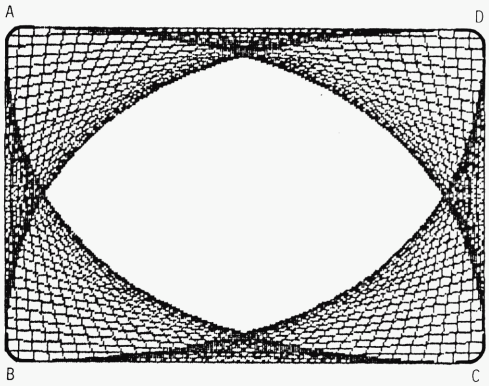
Pour une bonne utilisation de ce programme, dans un premier temps, cherchez combien de déformations vous pouvez visualiser. Cela dépend de la résolution de votre ordinateur. Il faut au minimum une quinzaine de déformations (en-dessous de ce nombre, les déformations ne sont pas assez progressives) et au maximum une centaine (les déformations sont progressives, mais l'écran est alors rempli et illisible !). Entre ces bornes, à vous de trouver "le bon choix".

Au début aussi, déformez des choses simples ; par exemple, le rectangle ABCD en rectangle EFGH. Faites varier les dimensions et positions de chacun des rectangles ci-contre.

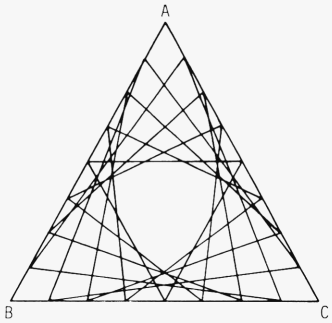




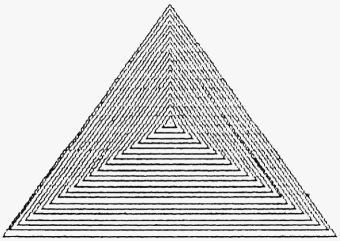
Puis, n'hésitez pas à donner du mouvement. Déformez un rectangle ABCD en lui-même, mais selon BCDA ... ou un triangle ABC en un triangle BCA :



RECTANGLE - RECTANGLE
3Ø DEFORMATIONS

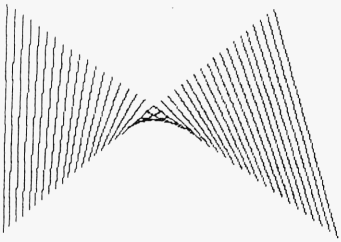


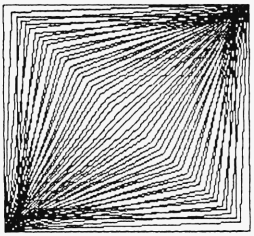
Puis, faites courir votre imagination. La méthode utilisée permet de déformer une figure formée de parties disjointes en une autre. Essayez donc le célèbre sigle  en "OIE".



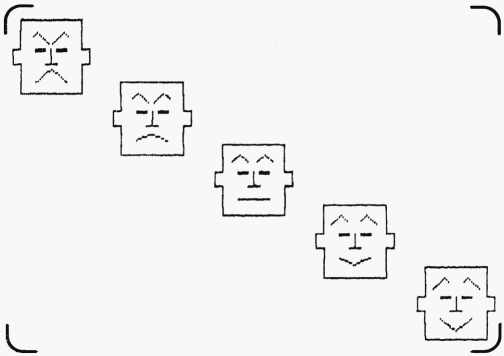
TRIANGLE - POINT
2Ø DEFORMATIONS

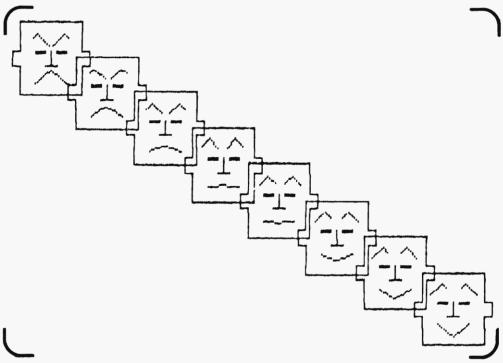
SEGMENT - SEGMENT
4Ø DEFORMATIONS





CARRE - CARRE
25 DEFORMATIONS





Le programme donné est, comme tous ceux de ce livre, à adapter à votre matériel et à vos goûts. Voici quelques suggestions :

- si vous pouvez avoir simultanément texte et graphisme, utilisez cette possibilité dans l'introduction des motifs : c'est beaucoup plus commode que de faire alterner texte puis graphisme ;
- si vous avez des couleurs, vous pouvez demander à l'utilisateur de préciser la couleur de chaque segment ;
- le programme donné visualise toutes les déformations en même temps. Si vous préférez du "vue à vue", ajoutez un effaçage d'écran avant l'affichage de chaque nouvelle déformation ;
- si vous disposez d'une instruction du type FILL (remplissage d'une zone), ne vous privez pas de tester l'effet produit ;
- ajoutez à ce programme un confort supplémentaire : en partant d'un nombre fixé de déformations (que vous aurez choisi), faites augmenter ce nombre en appuyant sur la touche "+" ou faites-le diminuer en appuyant sur la touche "-". Vous pouvez cumuler ce "confort" avec la possibilité de faire ou non du "vue à vue" (par appui sur la touche "v", par exemple).

Si vous les jugez intéressantes, rédigez certaines des adaptations proposées. Surtout, tirez le meilleur parti de votre matériel. Fermez ce livre un moment et déformez tout en n'importe quoi, c'est un vaste domaine ! Ne fermez ce livre qu'un moment, la suite en vaut la peine.

ENVELOPPES

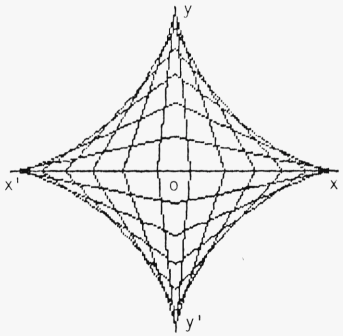
Après ces déformations, étudions d'autres graphismes engendrés par des droites qui varient. Vous avez sûrement déjà vu des tableaux faits à partir de fils et de clous. Nous vous proposons d'en réaliser avec beaucoup moins de risques : pas de marteau à manier pour placer les clous, pas de fils à emmêler.

De quoi s'agit-il dans de tels tableaux ? Des formes (visages, bateaux, papillons ...) sont esquissées par des droites, mais non tracées. Dans un langage plus mathématique, on dit que la forme est enveloppée par les droites au sens suivant : toutes les droites tracées (les fils tendus) sont tangentes (visualisent) à une courbe (une forme). De ce point de vue mathématique, il s'agit tout simplement de faire varier une famille de droites. Ces droites devront tout de même être assez voisines pour bien esquisser la forme.

Avant de passer au cas général, étudions quelques exemples.

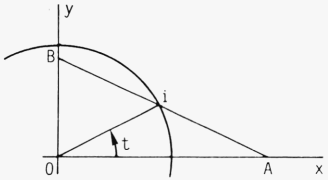
Exemples d'enveloppes

Considérons un segment AB de longueur fixe ℓ. Imposons à A de varier sur un axe x'ox et à B de varier sur y'oy (x'ox et y'oy étant deux axes perpendiculaires). On obtient la figure :



Ici, peu de droites AB suffisent à esquisser la courbe enveloppée. Cette courbe a la forme du carreau ◇ d'un jeu de carte. Elle est nommée "astroïde".

Pour programmer très simplement cela, il suffit de remarquer que AB ayant une longueur fixe ℓ, le milieu i de AB est à la distance $\ell/2$ de O : il varie donc sur un cercle de centre O, de rayon $\ell/2$.



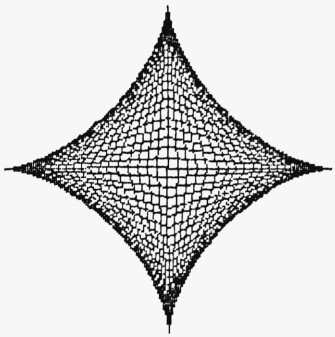
Ses coordonnées sont donc i($\ell/2$ cos t, $\ell/2$ sin t). D'où celles de A et B :

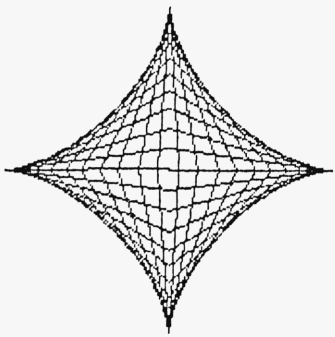
$$A \begin{pmatrix} \ell \cos t \\ 0 \end{pmatrix} \qquad B \begin{pmatrix} 0 \\ \ell \sin t \end{pmatrix}$$

Toutes les droites AB seront obtenues en faisant varier T de 0 à 2π. D'où le programme suivant qui tracera les droites AB correspondant à $T = \emptyset, \frac{2\pi}{N}, \frac{4\pi}{N}, \frac{6\pi}{N} \ldots, 2\pi$ ce qui fournira N droites AB ($T=\emptyset$ et $T=2\pi$ donnent la même droite) :

```
10 REM ASTROIDE
20 LG=    :HT=
30 XC=    :YC=
40 PI=3.14159
50 INPUT"LONGUEUR L=";L
60 INPUT"NOMBRE DE DROITES N=";N
70 IF N<=0 THEN 60
80 REM EFFACAGE D'ECRAN,MODE GRAPHIQUE
90 YA=YC:XB=XC
100 FOR I=0 TO N-1
110 T=2*PI*I/N
120 XA=XC+L*COS(T)
130 YB=YC+L*SIN(T)
140 DROITE XA,YA A XB,YB
150 NEXT I
```

Vous aurez peut-être intérêt à rajouter un test sur L qui doit être inférieur à HT/2. Cela évitera tout appel illégal de fonction lors du tracé des droites.

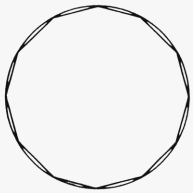


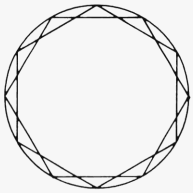


Si vous avez la curiosité de mesurer les différents segments tracés AB, vous avez toute chance de ne pas trouver une longueur fixe... Et, outre le fait que votre écran n'est pas plat ou que vos mesures sont entachées d'erreurs..., il y a une raison plus profonde : le repère de votre écran n'est sans doute pas orthonormé et manier correctement distances et angles est un peu compliqué... Mais, n'anticipons pas : ce sera l'objet du chapitre suivant.

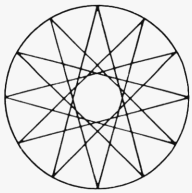
Prenons d'autres exemples simples. Plaçons, sur un cercle, N points formant un polygone régulier et joignons-les de toutes les façons possibles. Pour N entre 15 et 30, on obtient des motifs rappelant des vitraux ou de la dentelle. Comme quoi l'ordinateur peut avoir un côté rétro...

Dans ce fouillis assez esthétique de droites, qu'y-a-t-il ? Principalement des droites enveloppant des cercles et formant parfois des étoiles :





Points joints de 2 en 2



Points joints de 5 en 5

Là encore, la programmation est très simple. Les coordonnées des points du cercle sont du type :

$$R \cos \frac{2iPi}{N} \qquad (0 \leqslant i \leqslant N-1)$$

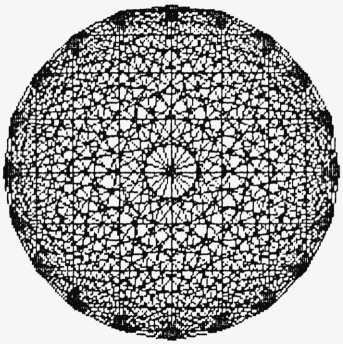
$$R \sin \frac{2iPi}{N}$$

En numérotant les points de Ø à N-1 (ce qui correspond à l'indice i), il suffit d'opérer ainsi pour tracer tous les segments :

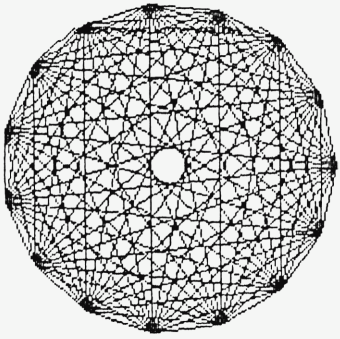
Point numéro	*Joint aux points numéro*	
Ø	1,2,..., N-1	
1	2,..., N-1	
2		
.		
i	i+1,..., N-1	
N-2	N-2	(peu d'intérêt)

Cela s'effectue par deux boucles imbriquées, la principale portant sur le numéro du point (i de Ø à N-2), l'autre sur les indices des points auxquels il est joint (J de i+1 à N-1).

```
10 REM NAPERONS
20 LG=    :HT=
30 XC=    :YC=
40 PI=3.14149
50 REM MODE TEXTE
60 INPUT"RAYON DU CERCLE";R
70 IF R<0 THEN 60
80 INPUT"NOMBRE DE POINTS";N
90 IF N=0 OR N<>INT(N) THEN 80
100 REM EFFACAGE D'ECRAN,MODE GRAPHIQUE
110 FOR I=0 TO N-2
120 TI=2*PI*I/N
130 FOR J=I+1 TO N-1
140 TJ=2*PI*J/N
150 DROITE XC+R*COS(TI),YC+R*SIN(TI)  A
    XC+R*COS(TJ),YC+R*SIN(TJ)
160 NEXT J
170 NEXT I
```



N= 2Ø



N= 15

N'introduisez ni N trop grand (au-delà de 30, on ne voit plus rien), ni N trop petit (au-dessous de 15, on voit des étoiles, il n'y a pas assez de droites pour esquisser les cercles enveloppés).

Modifions un tout petit peu la règle de jonction des points. Q étant un nombre entier fixé dont la signification sera expliquée plus tard, joignons les points d'indices :

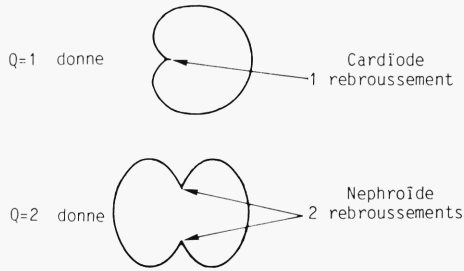
```
Ø  et  Ø       (peu d'intérêt)
1  et  Q+1
2  et  2(Q+1)
:
:
i  et  i(Q+1)
:
:
N-1 et  (N-1)(Q+1)
```

Dès qu'une quantité du type i(Q+1) dépasse N-1, on considère son reste dans la division par N qui est un nombre compris entre 0 et N-1.

Voir le programme de la page suivante.

```
10 REM EPICYCLOIDES ENVELOPPEES
20 LG=    :HT=
30 XC=    :YC=
40 PI=3.14149
50 REM MODE TEXTE
60 INPUT"RAYON DU CERCLE";R
70 IF R<0 THEN 60
80 INPUT"NOMBRE DE POINTS";N
90 IF N=0 OR N<>INT(N) THEN 80
100 INPUT"QUANTITE Q=";Q
110 Q=Q+1
120 REM EFFACAGE D'ECRAN,MODE GRAPHIQUE
130 FOR I=0 TO N-1
140 TI=2*PI*I/N
150 J=I*Q-N*INT(I*Q/N)
160 TJ=2*PI*J/N
170 DROITE XC+R*COS(TI),YC+R*SIN(TI)
    A XC+R*COS(TJ),YC+R*SIN(TJ)
180 NEXT I
```

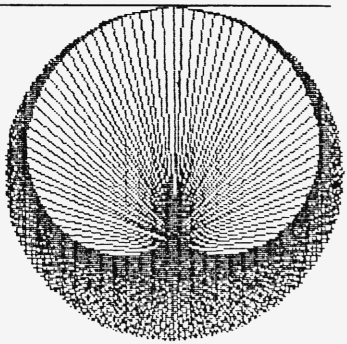
Pour bien utiliser ce programme, n'hésitez pas à choisir N assez grand (100 à 200), contrairement au programme précédent !



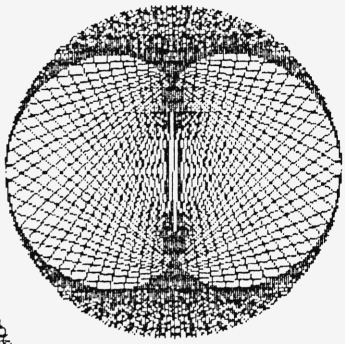
De façon générale, on obtient une courbe fermée, intérieure à un cercle, baptisée du nom barbare d'épicycloïde à Q rebroussements. Voilà la signification de Q : nombre de rebroussements de la courbe ! Vous pouvez faire défiler toute la série en partant de Q=1 et en faisant boucler le programme sur Q (voir figures à la page suivante).

Vous constaterez ainsi que plus Q s'accroît, plus l'épicycloïde s'aplatit sur le cercle au sens où :

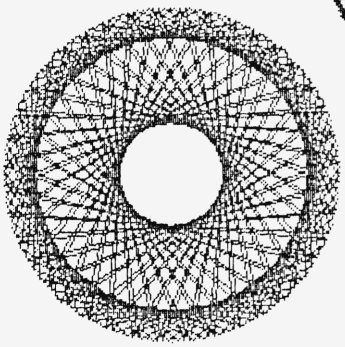
Pour des valeurs de Q trop grandes (testez sur votre machine le sens de "trop grande" !), le dessin devient confus..., sauf lorsque Q divise N ou qu'il se forme à l'intérieur du cercle, un moiré n'ayant rien à voir avec l'enveloppe étudiée.



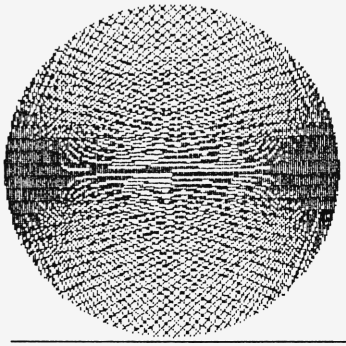
N= 200 N= 200 : Q= 1



N= 200 : Q= 2



N= 200 : Q= 40



N= 200 : Q= -101

Il y aura, sans doute, des lecteurs pressés qui auront introduit des valeurs négatives de Q. Après tout, rien ne l'interdit... A la rigueur, Q = -1, Q = -2 fournissent des graphismes interprétables. Mais, ensuite ? Rien que des segments plus ou moins réguliers à l'intérieur d'un cercle. Peu intéressant, sauf si l'on prolonge ces segments à l'extérieur du cercle. On constate qu'alors ils enveloppent bien des courbes, appelées hypocycloïdes. Que le lecteur ignorant tout de ces êtres barbares se rassure : ils réapparaîtront beaucoup plus clairement d'ici peu de temps.

Lors de l'étude de l'astroïde, nous avons précisé qu'à l'écran les différents segments n'ont pas la même longueur. Pour la même raison (repère-écran non orthonormé), les cercles des deux derniers exemples ressemblent sûrement plus à des ellipses. Le chapitre suivant indiquera les modifications à apporter pour obtenir de "bons" cercles.

Ces exemples, choisis pour leur simplicité et leur caractère esthétique, avaient pour seul but d'illustrer la notion d'enveloppe d'une famille de droites : c'est la courbe à laquelle toutes les droites de la famille sont tangentes.

Enveloppe d'une famille de droites

Dans tous les exemples vus, les droites considérées étaient tracées à partir de deux points. Il existe une autre façon mathématique de considérer des droites : par leur équation. Toute droite admet en effet une équation du type :

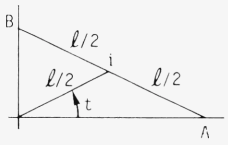
$$ax + by + c = 0$$

a, b, c étant a priori des nombres. Pour étudier une famille de droites, il suffit de faire varier ces nombres a, b, c, c'est-à-dire de les considérer comme fonctions d'une variable que nous appellerons **t**.

Pour chaque valeur de t, l'équation

$$a(t)\,x + b(t)\,y + c(t) = 0$$

fournit une droite. Lorsque t varie, cette équation fournit donc une famille de droites. Ainsi, dans notre premier exemple :



$A \begin{pmatrix} \ell \cos t \\ 0 \end{pmatrix}$ $B \begin{pmatrix} 0 \\ \ell \sin t \end{pmatrix}$

la droite AB a pour équation

$x \sin t + y \cos t - \ell \sin t \cos t = 0$

Ici, $a(t) = \sin t$, $b(t) = \cos t$, $c(t) = -\ell \sin t \cos t$

Les fonctions a(t), b(t), c(t) étant supposées connues (les lecteurs impatients trouveront au paragraphe suivant un catalogue d'exemples qui nous semblent particulièrement esthétiques), cherchons à faire tracer, par l'ordinateur, une telle famille de droites. De quoi avons-nous besoin ? Laissant les fonctions a, b, c qui occuperont chacune une ligne du programme, il nous faudra :

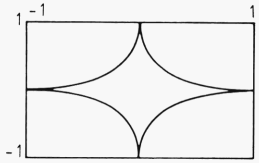
- introduire des valeurs définissant ce qu'on appelle la "fenêtre-écran". Ce terme signifie que, sur l'écran de visualisation, seul un rectangle (la fenêtre) peut être affiché. C'est à l'utilisateur de préciser quel rectangle il choisit.

Reprenons l'exemple de l'astroïde, en supposant que $\ell=1$. Si l'on décide de visualiser

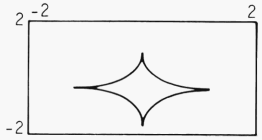
ce qui correspond à :

$-1 \leqslant X \leqslant 1$
$-1 \leqslant Y \leqslant 1$

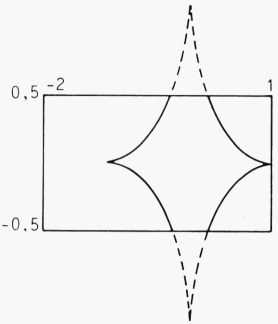
on a à l'écran :



$-2 \leqslant X \leqslant 2$
$-2 \leqslant Y \leqslant 2$



$-2 \leqslant X \leqslant 1$
$-0.5 \leqslant Y \leqslant 0.5$



La fenêtre-écran est donc définie par quatre nombres :

Xi : valeur inférieure de X (correspond à Ø)
XS : valeur supérieure de X (correspond à LG)
Yi : valeur inférieure de Y (correspond à HT)
YS : valeur supérieure de Y (correspond à Ø)

Nous prendrons toujours Yi en bas de l'écran et YS en haut, conformément à l'usage mathématique (axe de Y orienté vers le haut) et contrairement à l'usage informatique.

De simples règles de trois permettent de passer des coordonnées mathématiques (abscisse X entre Xi et XS, ordonnée Y entre Yi et YS) aux coordonnées écran (abscisse i entre Ø et LG, ordonnée J entre Ø et HT) :

$$\frac{X-Xi}{XS-Xi} = \frac{i-Ø}{LG-Ø} \quad \text{d'où} \quad i = LG * \frac{X-Xi}{XS-Xi}$$

$$\frac{YS-Y}{YS-Yi} = \frac{J-Ø}{HT-Ø} \quad \text{d'où} \quad J = HT * \frac{YS-Y}{YS-Yi}$$

Dans les programmes, nous prendrons pour i et J :

INT (LG*(X-Xi)/(XS-Xi)+.5) et INT (HT*(YS-Y)/(YS-Yi)+.5)

afin que i et J soient des "entiers bien arrondis". Cette façon de procéder est classique.

Cette fenêtre-écran sera évidemment introduite par des instructions INPUT. En plus de cette fenêtre, il nous faut :

- introduire les droites que nous voulons visualiser. Il suffit d'indiquer entre quelles valeurs le paramètre T doit varier. Appelons ces valeurs :

  Ti : valeur inférieure de T.
  TS : valeur supérieure de T.

  Il faut ensuite préciser le nombre N de droites à tracer. En faisant varier T de Ti à TS par pas TH=(TS-TI)/(N-1), on fera tracer les N droites voulues.

Là encore, les valeurs de TS, Ti, N seront introduites par des instructions INPUT. Reste à réaliser :

- un sous-programme traçant, sur la fenêtre-écran, $Xi \leqslant x \leqslant XS$ $Yi \leqslant Y \leqslant YS$ la droite d'équation :

  AX+BY+C=0

  Nous n'écrirons pas a(t), b(t), c(t), mais a, b, c, uniquement pour simplifier les écritures.

La méthode choisie est simple et facile à comprendre : nous allons chercher les points en lesquels une droite coupe le bord de l'écran (si elle le coupe, car certaines droites pourraient être extérieures à la fenêtre-écran).

Les bords de l'écran correspondent à :

$$\left.\begin{array}{ll} X = Xi & \text{d'où } Y = -\dfrac{A*Xi+C}{B} \\ X = XS & \text{d'où } Y = -\dfrac{A*XS+C}{B} \end{array}\right\} \text{ si } B \neq \emptyset$$

$$\left.\begin{array}{ll} Y = Yi & \text{d'où } X = -\dfrac{B*Yi+C}{A} \\ Y = YS & \text{d'où } X = -\dfrac{B*YS+C}{A} \end{array}\right\} \text{ si } A \neq \emptyset$$

mais dès que nous aurons deux points, nous pourrons tracer la droite : ces quatre essais ne seront pas tous utiles.

Cela conduit à l'algorithme :

si $A = \emptyset$ et $B = \emptyset$: inutile de poursuivre, l'équation $AX + BY + C = \emptyset$ n'est pas une équation de droite.

si $B \neq \emptyset$: on prend X = Xi - Si $Y = -\dfrac{A*Xi+C}{B}$ est entre Yi et YS, le point $\left(\begin{array}{c} Xi \\ -\dfrac{A*Xi+C}{B} \end{array}\right)$ convient.

on prend X = XS - Si $Y = -\dfrac{A*XS+C}{B}$ est entre Yi et YS, le point $\left(\begin{array}{c} XS \\ -\dfrac{A*XS+C}{B} \end{array}\right)$ convient.

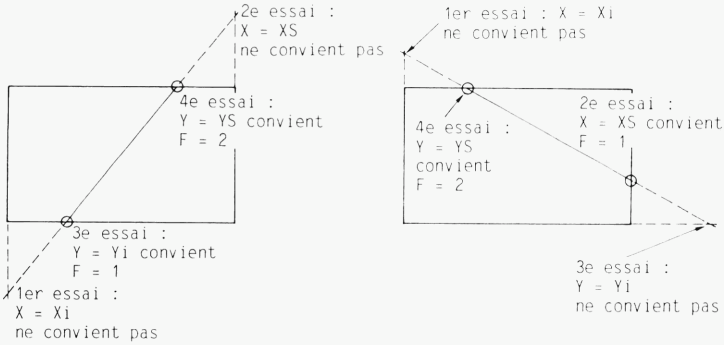
si $A \neq \emptyset$: on prend $Y = Yi$ - Si $X = -\frac{B * Yi + C}{A}$ est entre Xi et XS, le point $\left(-\frac{B * Yi + C}{A} \; Yi\right)$ convient.

on prend $Y = YS$ - Si $X = -\frac{B * YS + C}{A}$ est entre Xi et XS, le point $\left(-\frac{B * YS + C}{A} \; YS\right)$ convient.

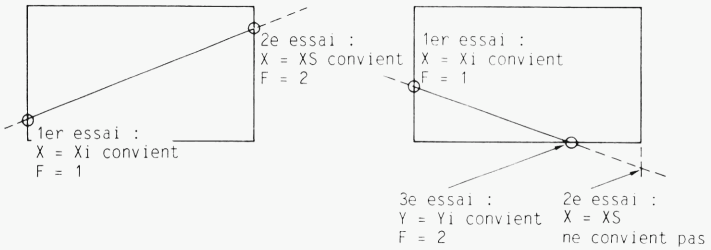
"convient" signifie ici : point de la droite situé au bord de l'écran.

Dès qu'un point convient, une variable F, sorte de "drapeau", mise à Ø au début de chaque étude de droite, est augmentée de 1, et l'on stocke les coordonnées - écran du point qui convient dans X(F) et Y(F).

Illustrons cet algorithme par des exemples :



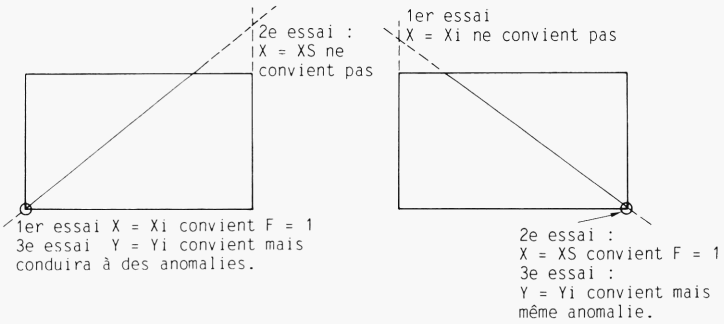
Sur ces deux exemples, il faut aller jusqu'au dernier essai pour tracer la droite désirée. Ce n'est pas toujours le cas :



Pour éviter les calculs inutiles, on va donc ajouter à l'algorithme un test supplémentaire : la comparaison de F à 2, à partir du second essai (X=XS). Si F vaut 2, c'est que la droite à tracer est définie...

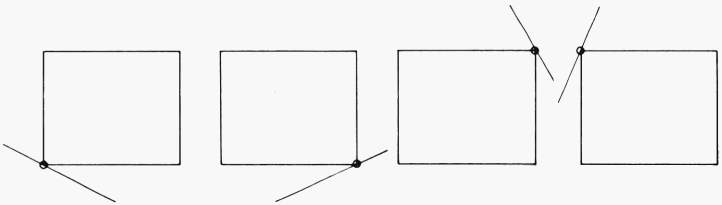
Arrivé à ce point, vous pouvez penser, cher lecteur, que tant de pages pour un simple tracé de droites, c'est trop... Peut-être, mais ce n'est

pas tout. Dans certains cas, ce test de F par rapport à 2, effectué sans précaution, peut conduire à des difficultés.



Pour éviter cette difficulté, lorsque F=2, on testera si les points de coordonnées (X(1),Y(1)) et (X(2),Y(2)) sont confondus ou non. S'ils le sont, F restera à 1 et on continuera les essais.

En procédant ainsi, il est évident que les droites qui ne coupent la fenêtre-écran qu'en un seul point ne seront pas tracées. Ces droites sont les suivantes :



Ne pas tracer ces "droites", réduites à un point pour la fenêtre considérée, n'a aucune importance pour la question qui nous préoccupe, l'enveloppe d'une famille. De toute façon, entre plusieurs maux, nous avons choisi le moindre.

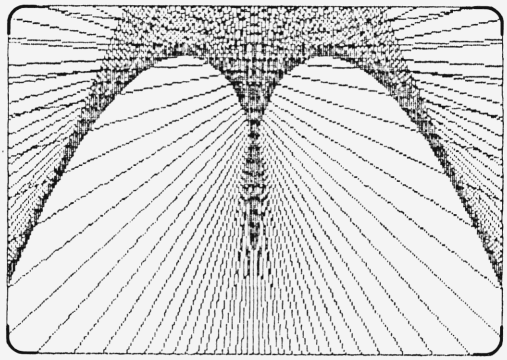
Voilà maintenant défini un algorithme de tracé d'une droite d'équation $ax + by + c = 0$. Si nous avons tant détaillé, c'est que, sous une apparence simple, un tel algorithme n'est pas immédiat.

Nous disposons maintenant des trois éléments de base qu'il nous fallait :

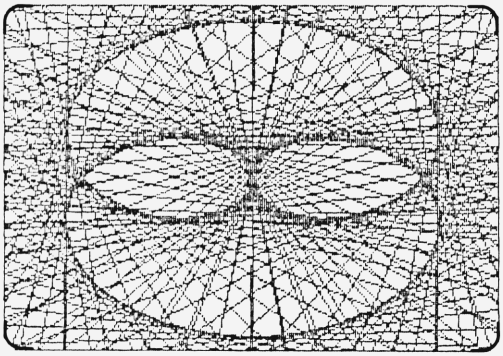
- définition de la fenêtre-écran (Xi,XS,Yi,YS) ;
- définition de la famille de droites (Ti,TS,N) ;
- tracé d'une droite d'équation $ax + by + c = 0$.

Le programme de visualisation de l'enveloppe d'une famille de droites en découle facilement :

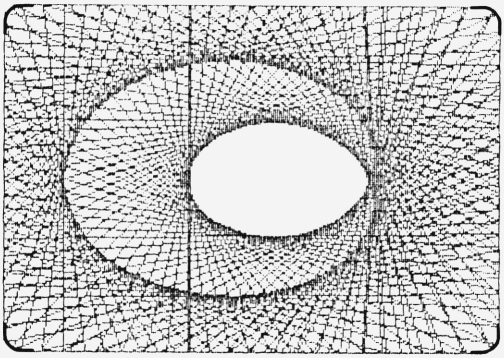
```
30 REM ENVELOPPE D'UNE FAMILLE DE DROITES
40 DIM X(2),Y(2)
50 LG=   :HT=
100 GOSUB 1000
110 GOSUB 1500
120 END
1000 REM INTRODUCTION DES DONNEES
1010 REM MODE TEXTE,EFFACAGE D'ECRAN
1020 INPUT"ABSCISSE MINIMUM";XI
1030 INPUT"ABSCISSE MAXIMUM";XS
1040 IF XI>=XS THEN 1020
1050 PRINT
1060 INPUT"ORDONNEE MINIMUM";YI
1070 INPUT"ORDONNEE MAXIMUM";YS
1080 IF YI>=YS THEN 1060
1090 PRINT
1100 INPUT"PARAMETRE MINIMUM";TI
1110 INPUT"PARAMETRE MAXIMUM";TS
1120 IF TI>=TS THEN 1100
1130 PRINT
1140 INPUT"NOMBRE DE DROITES";N
1150 IF N<=1 AND N<>INT(N) THEN 1140
1160 TH=(TS-TI)/(N-1)
1170 RETURN
1500 REM BOUCLE PRINCIPALE
1510 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
1520 FOR T=TI TO TS STEP TH
1530 A=
1540 B=
1550 C=
1560 GOSUB 2000
1570 NEXT T
1580 RETURN
2000 REM DETERMINATION D'UNE DROITE
2010 IF A=0 AND B=0 THEN RETURN
2020 F=0
2030 IF B=0 THEN 2090
2040 X=XI:Y=-(C+A*X)/B
2050 GOSUB 2200
2060 X=XS:Y=-(C+A*X)/B
2070 GOSUB 2200:IF F=2 THEN 2130
2080 IF A=0 THEN RETURN
2090 Y=YI:X=-(C+B*Y)/A
2100 GOSUB 2200:IF F=2 THEN 2130
2110 Y=YS:X=-(C+B*Y)/A
2120 GOSUB 2200:IF F<>2 THEN RETURN
2130 DROITE X(1),Y(1) A X(2),Y(2)
2140 RETURN
2200 REM TABLEAU POINTS UTILES
2210 IF X<XI OR X>XS OR Y<YI OR Y>YS
     THEN RETURN
2220 F=F+1
2230 X(F)=INT(LG*(X-XI)/(XS-XI)+.5)
2240 Y(F)=INT(HT*(YS-Y)/(YS-YI)+.5)
2250 IF F=2 AND X(1)=X(2) AND Y(1)=Y(2)
     THEN F=1
2260 RETURN
```



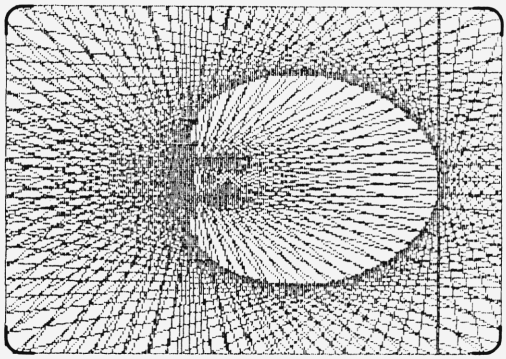
```
A=2-T*T
B=-2*T
C=T*T*T
XI=-5 : XS= 5
YI=-2 : YS= 1.5
TI=-3 : TS= 3
 1ØØ DROITES
```



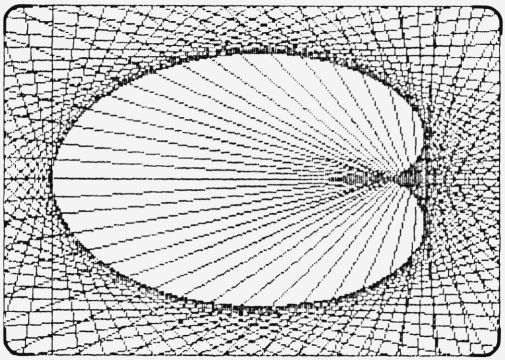
```
A=COS(3*T)
B=SIN(3*T)
C=SIN(T)^2
XI=-1 : XS= 1
YI=-1.1 : YS= 1.1
TI= Ø : TS= 6.28
 15Ø DROITES
```



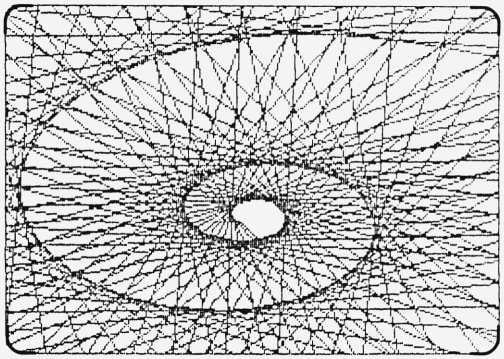
```
A=COS(T)
B=SIN(T)
C=2+COS(T/2)
XI=-4 : XS= 4
YI=-4 : YS= 4
TI=-6.28 : TS= 6.28
 15Ø DROITES
```



```
A=COS(T)
B=SIN(T)
C=-.5+COS(T/2)
XI=-2 : XS= 2
YI=-2 : YS= 2
TI=-6.28 : TS= 6.28
 15Ø DROITES
```



```
A=COS(3*T)
B=SIN(3*T)
C=COS(T)
XI=-1.2 : XS= .8
YI=-1.2 : YS= 1.2
TI=-1.57Ø8 : TS= 1.57Ø8
 1ØØ DROITES
```



```
A=COS(T)
B=SIN(T)
C=-EXP(-T/5)
XI=-2 : XS= 2
YI=-2 : YS= 3
TI=-5 : TS= 1Ø
 1ØØ DROITES
```

a(t)	b(t)	c(t)	Xi	XS	Yi	YS	Ti	TS
$2-t^2$	-2 t	t^3	-5	5	-2	1,5	-3	3
cos 3 t	sin 3 t	$\sin^2 t$	-1	1	-1,1	1,1	0	6,28
cos 3 t	sin 3 t	cos t	-1,2	0,8	-1,2	1,2	-1,5708	1,5708
cos t	sin t	$e(-e^{-t/5})$	-2	2	-2	3	-5	10
cos t	sin t	$-\frac{\sin t}{t}$	-0,3	1,2	-0,9	0,9	-5	5
cos t	sin t	-1-INT (T/2Pi)	-6	6	-6	6	0	31,415
cos t	sin t	t	-10	10	-11	11	-10	10
cos t	sin t	1/t	-2	2	-1	0,5	-10	10
cos t	sin t	logt	-4	4	-4	4	0,2	20
cos t	sin t	$\frac{t^2}{1+t^2}$	-1	1	-1	1	-6,28	6,28
cos t	sin t	$\frac{t}{1+t^2}$	-0,5	0,5	-1,2	0,5	-9,4248	9,4248
cos t	sin t	$\frac{1-t^2}{1+t^2}$	-1,5	1,5	-1,5	1,5	-6,28	6,28
cos t	sin t	$\sqrt{\lvert t \rvert}$	-4	4	-4	4	-10	10
cos t	sin t	$\frac{t}{1+t}$	-3	3	-3	3	-10	10
cos t	sin t	2+cos(t/2)	-4	4	-4	4	-6,28	6,28
cos t	sin t	-0,5+cos (t/2)	-2	2	-2	2	-6,28	6,28
cos t	sin t	$t^2 \sin t$	-4	4	-10	3	-3,1416	3,1416
cos t	sin t	$\frac{t^2}{1-t^2}$	-1,5	2	-2	2	-6,28	6,28
cos t	sin t	$\sin t + \frac{1}{2}\sin^2 t + \frac{1}{3}\sin^3 t$	-2	2	-3	0	-3,1416	3,1416
cos t	sin t	$\frac{1}{\sqrt{\lvert t \rvert}}$	-2	1	-1,5	1,5	-6,281	6,28
sin t	cos t	t sin t	-10	10	-0,5	2,5	-10	10
sin t	cos t	sin t logt	-3	1	-0,2	1	0,2	10

Petite bibliothèque d'enveloppes

Pour vous éviter de perdre trop de temps dans la recherche de bonnes fonctions a(t), b(t), c(t), de bonnes bornes Xi, XS, Yi, YS, Ti, TS, nous vous livrons le fruit de certains de nos essais. Nous espérons que vous aurez le même plaisir que le nôtre à regarder ces enveloppes.

Ne figurent pas dans le tableau (page 44) deux cas particulièrement intéressants :

$a(t) = \cos t \quad b(t) = \sin t \quad c(t) = (K+2) * \sin \frac{Kt}{K+2}$

avec X et Y compris entre -(K+2) et K+2
et T compris entre Ø et π (K+2).

Testez ces enveloppes pour certaines valeurs de l'entier K. Cela ne vous rappelle rien ? Vraiment ? Non ? Alors, faites un petit retour en arrière, reprenez les exemples d'enveloppes. Mais oui, il s'agit des épicycloïdes à K rebroussements ! Les revoilà tracées d'une façon un peu différente.

De même, ne figure pas le cas :

$a(t) = \cos t \quad b(t) = \sin t \quad c(t) = (K-2) * \sin \frac{Kt}{K-2}$

avec X et Y compris entre -K et K
T compris entre Ø et π (K-2).

Là encore, testez certaines valeurs de K (entier). Si ces jolies courbes vous rappellent quelque chose c'est, ou bien que vous avez une certaine culture, ou bien que vous êtes un lecteur pressé ayant introduit une valeur de Q négative dans le dernier exemple d'enveloppe ! Mais là, vous obtenez l'enveloppe des droites et non plus des segments, ce qui était insuffisant pour visualiser l'enveloppe. Les courbes ainsi obtenues sont des hypocycloïdes à K rebroussements.

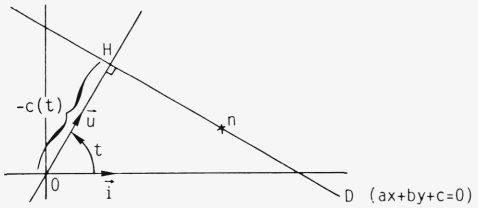
Rien ne vous empêche, bien sûr, de prendre pour K des valeurs non entières... Mais alors, parler de K rebroussements n'a plus de sens, les bornes indiquées sur T n'étant plus valables. Le dessin obtenu est illisible. Mieux vaut tracer les courbes enveloppées point par point. Ce sera fait au chapitre I de la deuxième partie.

Il peut être intéressant de tenter de prévoir quel type d'enveloppe on obtiendra. Tout cours de mathématiques supérieures ou mathématiques spéciales comporte un chapitre consacré aux enveloppes. La théorie n'est pas très compliquée mais dépasse le cadre de cet ouvrage.

Vous aurez sans doute remarqué que, dans la plupart des exemples, on choisit :

$a(t) = \cos t \quad b(t) = \sin t$

c'est que, pour un tel choix, la fonction c(t) s'interprète géométriquement :



$\vec{u}$ étant un vecteur unitaire tel que $(\vec{i},\vec{u}) = t$, on a :

$$\overline{OH} = -c(t)$$

sur la droite passant par O perpendiculaire à D d'équation $ax + by + c = 0$.

Pour s'en convaincre, il suffit de remarquer que $\vec{u}\begin{pmatrix}\cos t\\ \sin t\end{pmatrix}$ et que $M\begin{pmatrix}x\\ y\end{pmatrix}$ appartenant à D :

$$\overline{OH} = \overline{OM} \cdot \vec{u} = x \cos t + y \sin t = -c(t)$$

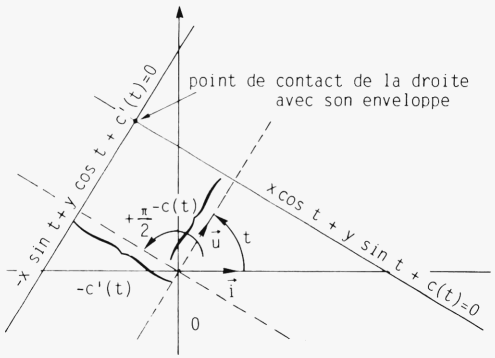
Cela n'est valable qu'en repère orthonormé (axes perpendiculaires et de même sens), ce qui n'est pas le cas du repère de l'écran de visualisation... Cela permet tout de même d'avoir une certaine idée de la façon dont les droites varient.

De façon plus précise, on montre (voir cours de maths sups pour les démonstrations rigoureuses) que les coordonnées du point de contact de D avec la courbe enveloppée vérifient le système :

$$\begin{cases} x\cos t + y\sin t + c(t) = 0 \\ -x\sin t + y\cos t + c'(t) = 0 \end{cases} \quad \text{d'où} \quad \begin{cases} x = -c(t)\cos t + c'(t)\sin t \\ y = -c(t)\sin t - c'(t)\cos t \end{cases}$$

(c'(t) : fonction dérivée de c(t)).

Ou encore, géométriquement :



car l'équation $-x \sin t + y \cos t + c'(t) = 0$ s'écrit aussi $x \cos (t+\frac{\pi}{2}) + y \sin (t+\frac{\pi}{2}) + c'(t) = 0$: elle est du même type que celle de D, t étant remplacé par $t+\frac{\pi}{2}$ et c(t) par c'(t). Le point de contact de la droite avec son enveloppe est à une distance $\sqrt{c(t)^2 + c'(t)^2}$ de O.

Toutes ces précisions risquant de paraître bien théoriques, dans la pratique on aura intérêt à procéder de la façon suivante : on choisira une fenêtre assez grande (-10 à 10 pour X et Y ou -5 à 5), puis on la réduira s'il s'avère que l'enveloppe est trop réduite sur l'écran, sinon on l'augmentera. En quelques essais, on parvient à une bonne fenêtre. Pour des essais, on aura intérêt à prendre N = 100. Autant le choix de la fenêtre est indépendant du matériel (il dépend plutôt du goût de l'utilisateur quant au cadrage), autant celui du nombre de droites N dépend de la résolution du matériel utilisé : à vous donc de trouver ce qui convient.

En plus du catalogue et de ces remarques, voici quelques autres idées qui vous permettront d'étudier de nouvelles enveloppes.

Il existe trois types classiques d'équation pour des courbes. Pour de plus amples précisions et de beaux exemples, voyez la deuxième partie de ce livre. Ces types sont :

- *Equation cartésienne* Y=F(X).

L'équation de la tangente au point d'abscisse XØ est :

Y-F(XØ) = F'(XØ)(X-XØ)
ou F'(XØ)X-Y+F(XØ) - XØ F'(XØ) =0

(cela figure dans tout cours de première). Il suffit donc de prendre :

A=F'(T) B=-1 C=F(T)-TF'(T) (F': dérivée de F)

Bien entendu, F et F' sont à remplacer par leur expression ou à définir par des instructions de définition de fonction DEFFN...

- *Equations paramétriques* X=F(T) Y=G(T).

Soit l'équation de la tangente au point d'abscisse TØ (si F'(TØ) et G'(TØ) ne sont pas tous deux nuls) :

$$\begin{vmatrix} X-F(T\emptyset) & F'(T\emptyset) \\ Y-G(T\emptyset) & G'(T\emptyset) \end{vmatrix} = 0$$

Pardon pour ce déterminant, maintenant banni de l'enseignement secondaire... Mais il paraît qu'il existe encore de vieux professeurs, réactionnaires par définition, qui, non seulement connaissent cette notion, mais encore l'enseignent, au mépris des directives ministérielles... S'ils ne sont pas trop "réactionnaires", demandez-leur...

Nous disions donc que, pour des équations paramétriques, il suffit de prendre :

A=G'(T) B=-F'(T) C=G(T) F'(T) - G'(T) F(T)

(F',G' : dérivées de F et G).

- *Equation polaire* R=F(T).

Ce cas se ramène au second en posant :

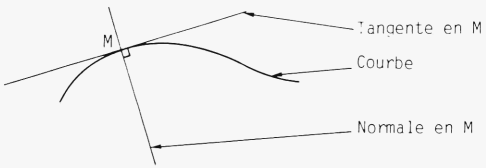
X=R cos(T) = F(T) cos(T) Y=R sin(T) = F(T) sin(T)

ce qui donne :

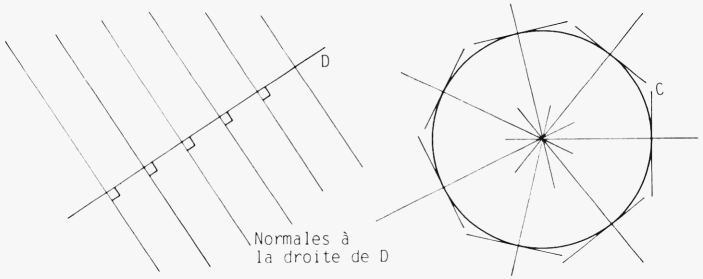
A=F'(T) sin(T)+F(T) cos(T) B=-F'(T) cos(T)+F(T) sin(T) C=-F(T)

Ces valeurs de A, B, C vous permettront de tracer toute courbe, non pas point par point, mais comme enveloppe de ses tangentes.

Lorsqu'on définit une courbe, on s'intéresse souvent à deux familles de droites : les tangentes et les droites appelées normales, perpendiculaires aux tangentes au point de contact :



Ainsi, toutes les normales à une droite sont parallèles et toutes les normales à un cercle se coupent en son centre :



Les normales à une courbe enveloppent elles aussi une autre courbe appelée développée. Pour chacun des trois cas vus ci-avant, voici les valeurs de A, B, C qui fournissent les développées :

- *équation cartésienne* Y=F(X)

  A=1 B=F'(T) C=-T-F(T)F'(T)

- *équations paramétriques* X=F(T) Y=G(T)

  A=F'(T) B=G'(T) C=-F(T)F'(T)-G(T)G'(T)

- *équation polaire* R=F(T)

  A=-F'(T)cos(T)+F(T)sin(T) B=-F'(T)sin(T)-F(T)cos(T)
  C=+F(T)F'(T)

Un cas simple et intéressant est celui où une droite d'équation :

$$x \cos t + y \sin t + c = 0$$

enveloppe une courbe. Nous avons vu que la normale a pour équation :

$$-x \sin t + y \cos t + c' = 0$$

Dans tous les exemples donnés où a=cos t, b=sin t, il suffit de prendre a=-sin t, b=cos t et de remplacer c par sa dérivée pour passer d'une courbe à sa développée. Attention : vous pouvez prendre pour T les mêmes bornes mais, pour Xi, XS, Yi, YS, il y aura sûrement quelques modifications...

Vous avez maintenant du pain sur la planche. Testez les exemples donnés. Puis, cherchez bien : vous connaissez sûrement quelques courbes. Allez : paraboles, ellipses, hyperboles... et toutes les autres que n'importe quel livre de maths vous donnera. Enveloppez-les, développez-les. Les résultats ne sont pas garantis à tous les coups, mais vous trouverez sûrement d'aussi beaux exemples que les nôtres. Nous espérons que vous éprouverez un plaisir analogue au nôtre devant certaines réalisations. Cela vous réconciliera peut-être avec un aspect esthétique des mathématiques, malheureusement souvent oublié dans le domaine scolaire...

Un dernier avertissement : si vous essayez de tracer, en même temps, une courbe comme enveloppe de ses tangentes et sa développée comme enveloppe de ses normales, vous constaterez que tangentes et normales ne sont pas perpendiculaires... Cela est, encore une fois, dû au fait que le repère utilisé n'est pas orthonormé. Mais peut-être êtes-vous las de nous entendre, pour la troisième fois (nous avons compté !), vous signaler des difficultés sur tout ce qui touche aux longueurs, aux perpendiculaires... Eh bien, le moment est venu de clarifier tout cela.

Chapitre II
Rotations, angles

ANGLES ET DISTANCES

Rotation d'un segment

Pour commencer, regardons la **figure 1.** Si un segment [OA] de longueur ℓ fait un angle t avec l'axe des x, les coordonnées du point A sont (ℓ cos t, ℓ sin t).

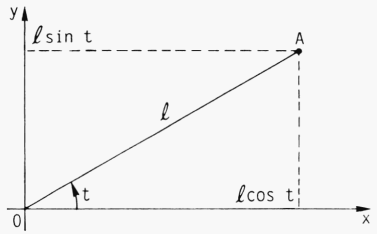


Figure 1

Nous en savons maintenant assez pour faire tourner un segment autour du centre de l'écran. Il suffit de taper le programme suivant :

```
10 XC=   :YC=   :PI=3.14159265
20 REM MODE TEXTE
30 INPUT"ANGLE EN DEGRES";A
40 INPUT"LONGUEUR DU SEGMENT";L
50 A=A*PI/180
60 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
70 DROITE XC,YC A XC+L,YC
80 DROITE XC,YC A XC+L*COS(A),YC+L*SIN(A)
90 END
```

Les ordinateurs calculant COS et SIN avec un argument en radian (180° = π radians), il faut taper les lignes 10 et 50 pour faire la conversion.

En faisant tourner ce programme, vous n'obtiendrez certainement pas les résultats escomptés. Il faut bien sûr ne pas donner de trop grandes valeurs

à L, sinon un message d'erreur apparaît ; mais il y a plus grave. Si vous avez voulu obtenir la **figure 2-a**, vous avez obtenu la **figure 2-b** :

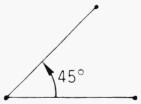


Figure 2a

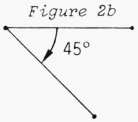
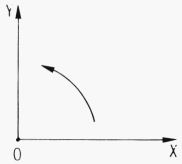
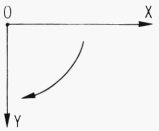


Figure 2b

Ceci est dû au fait que nous avons raisonné dans un repère comme celui-ci :



Malheureusement, pour la plupart des écrans d'ordinateurs, l'orientation est ainsi :

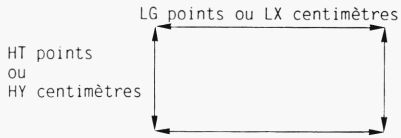


Elle est donc contraire à l'orientation classique. Pour rectifier, il suffit, à la ligne 8Ø, de remplacer YC+L*SIN(A) par YC-L*SIN(A).

A ce stade, tout devrait bien marcher mais, sur beaucoup d'écrans, vous n'obtiendrez pas des segments ayant parfaitement la même longueur (mesurez-les, vous verrez !). Pour obtenir des figures parfaites, des cercles qui soient vraiment des cercles, il nous faudra, dans tous les programmes, ajuster les distances horizontales et verticales selon un certain coefficient que nous noterons KE.

Pour déterminer ce coefficient KE, procédez de la manière suivante :

- faites tracer le plus grand rectangle possible sur votre écran, mesurez sa longueur LX et sa hauteur HY.

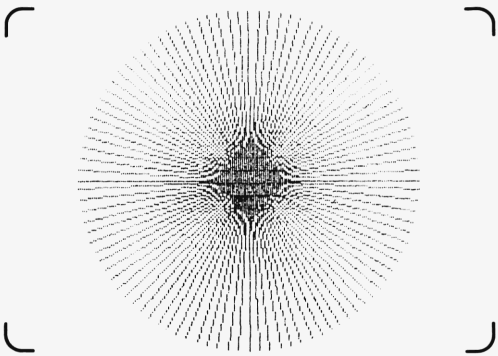


$$KE = \frac{HY}{LX} \times \frac{LG}{HT}$$

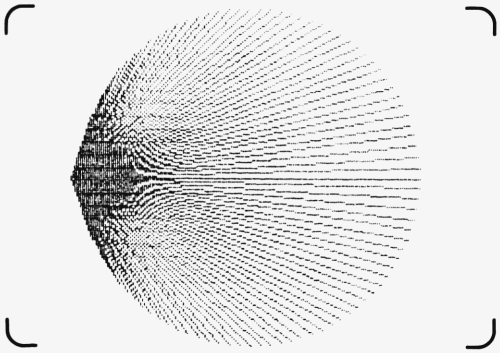
L'idéal est évidemment d'obtenir K=1, mais peu d'écrans donnent cette valeur. Pour obtenir un résultat parfait, tapez maintenant YC-L/KE*SIN(A) à la ligne 8Ø.

Nous allons utiliser nos nouvelles connaissances à des applications simples mais, nous l'espérons, esthétiques.

```
10 KE=    :XC=    :YC=    :PI=3.14159265
20 L=HT/2:REM MOITIE DE LA HAUTEUR DE L'ECRAN
30 REM MODE TEXTE
40 INPUT"NOMBRE DE RAYONS";N
50 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
60 FOR I=0 TO N-1
70 DROITE XC,YC
   A XC+L*KE*COS(2*PI*I/N),YC-L*SIN(2*PI*I/N)
80 NEXT I
```



100 RAYONS

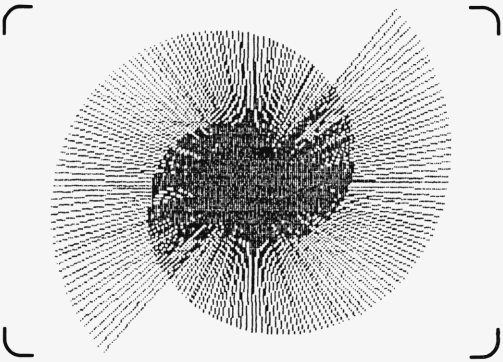


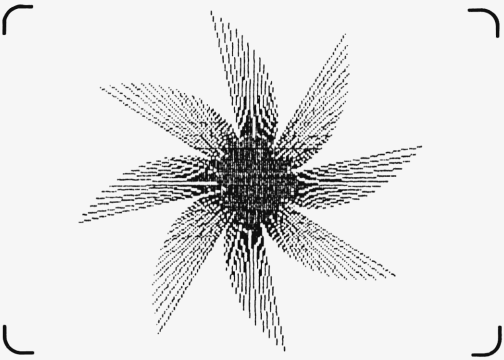
On ajuste ici la longueur en X, la longueur de référence étant verticale. Variante intéressante : au lieu de faire partir les rayons du centre de l'écran, partez de n'importe quel point.

Après avoir vu ce que l'on pouvait obtenir en faisant tourner un segment de longueur constante, nous pouvons faire changer la longueur à chaque pas :

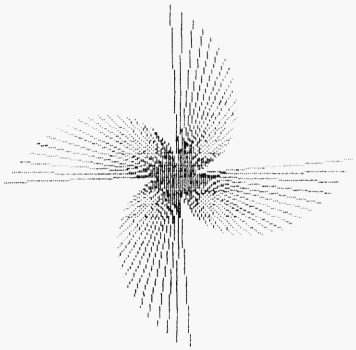
```
10 LG=    :HT=
20 XC=    :YC=
30 KE=    :PI=3.14159265
40 REM MODE TEXTE
50 INPUT"ANGLE EN DEGRES";A
60 INPUT"LONGUEUR DU SEGMENT";L
70 INPUT"ACCROISSEMENT DE LA LONGUEUR";DL
80 A=A*PI/180
90 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
100 X=XC+L*KE*COS(N*A)
110 Y=YC-L*SIN(N*A)
120 IF X<0 OR X>LG OR Y<0 OR Y>HT THEN 150
130 DROITE XC,YC A X,Y
140 L=L+DL :N=N+1 :GOTO 100
150 GOTO 150
```

L=Ø : DL=.3 : A=181

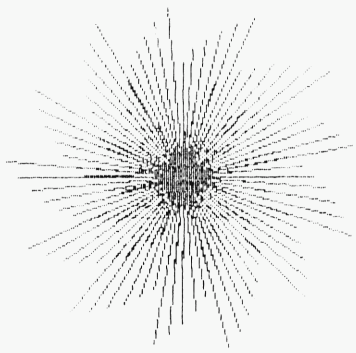




L=Ø : DL=.5 : A=45.3



L=1 : DL= 1 : A=91



L=Ø : DL=.5 : A=55

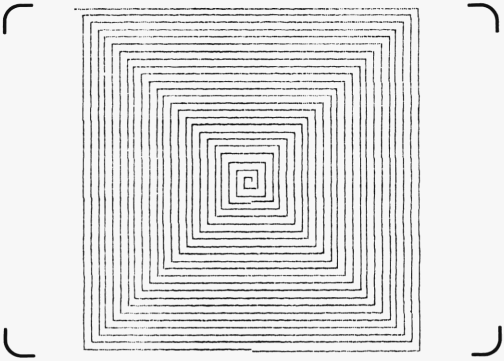
La longueur L augmente de DL à chaque pas. Il faut donc prévoir un test (ligne 12Ø) pour éviter un message d'erreur lorsqu'on sort des limites de l'écran. Pour obtenir des figures intéressantes, donnez à L et DL des valeurs faibles (1 et 1 par exemple) et ne prenez pas pour A un diviseur de 180°, sinon les segments se superposent. Les valeurs suivantes donnent d'assez jolis résultats :

A	L	DL
31	1	1
7	1	1
3	1	0,5
55	1	1
91	1	1
47	1	0,5
179	1	1
182	1	0,5
89	100	-1
44	100	-1

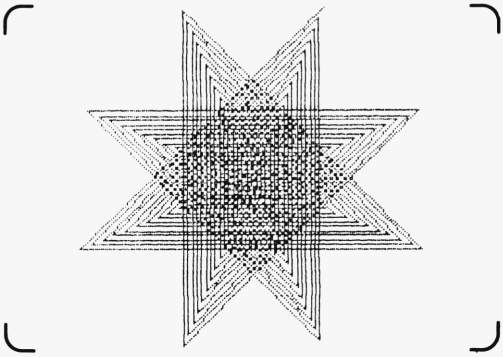
Les derniers exemples ne correspondent pas précisément aux indications ci-dessus, mais ils sont assez beaux et il est permis de s'y essayer (diminuez L si votre résolution n'est pas très fine).

Il est facile de modifier ce programme pour que les segments soient mis bout à bout, au lieu de partir tous du même point. Il suffit de rajouter ou de modifier les lignes suivantes :

```
95 C=XO:D=YO
100 X=C+L#*E#COS(N#A)
110 Y=D-L#SIN(N#A)
130 DROITE C,D A X,Y
135 C=X:D=Y
```

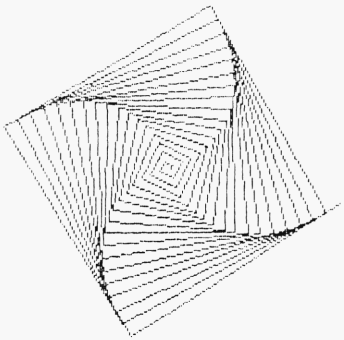


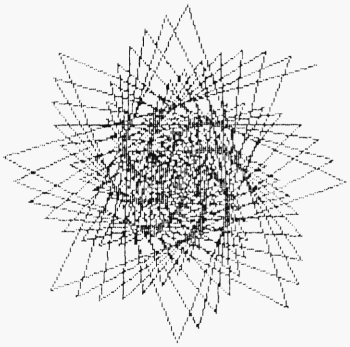
L= Ø : DL= 2 : A= 9Ø



L= Ø : DL= 1.5 : A= 225

L= Ø : DL= 2 : A= 91





L= Ø : DL= 1.5 : A= 224

Ce programme peut maintenant vous donner des spirales, des étoiles et bien d'autres choses. Essayez, par exemple :

A	L	DL
90	2	0
211	0	1
225	0	1,5
210	0	1
10	0	0,1
181	0	1
55	1	1
89	100	-1
89	0	1
61	0	0,5
45	0	1
90	0	1

et bien d'autres...

Vous êtes maintenant en mesure de modifier les programmes du premier chapitre qui ne donnaient pas des segments de même longueur. Pour chacun d'eux, il suffit de rajouter :

```
15  KE=
```

Pour le programme "**astroïde**", prendre :

```
12Ø  XA=XC+L*KE*COS(T)
```

Pour le programme "**napperons**", prendre :

```
15Ø  DROITE XC+R*KE*COS(Ti),YC+R*SIN(Ti)
           A XC+R*KE*COS(TJ),YC+R*SIN(TJ)
```

Pour le programme "**épicycloïdes**", prendre pour ligne 18Ø la modification qui vient d'être écrite pour le programme "napperons".

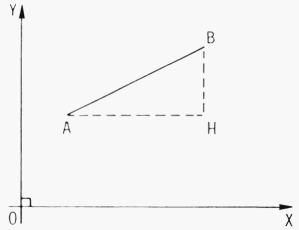
Distances

Dans les exemples précédents, nous avons agi sur un segment dont la longueur était donnée. Dans bien des cas, on connaît les coordonnées des extrémités, mais pas leur longueur.

Pour calculer la distance entre deux points, nous pouvons nous servir du bon vieux théorème de Pythagore :

$$AB^2 = AH^2 + BH^2$$

$$AB = \sqrt{AH^2 + BH^2}$$



En se servant des coordonnées des points $A(x_A,y_A)$ et $B(x_B,y_B)$, la formule devient :

$$AB = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$

ou en posant :

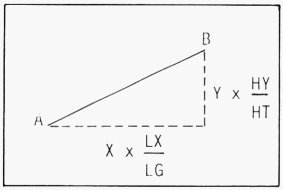
$$X = x_B - x_A \quad , \quad Y = y_B - y_A$$

$$AB = \sqrt{X^2 + Y^2}$$

Malheureusement, sur l'écran, l'axe des X et l'axe des Y n'ont pas forcément la même unité.

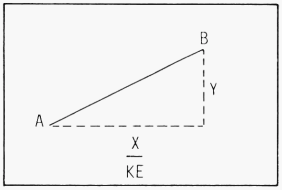
Pour trois unités simples (le centimètre, celle des X et celle des Y), nous donnons les coordonnées du vecteur $\vec{AB}$ avec cette unité, et la distance correspondante.

- *Unité : le centimètre*



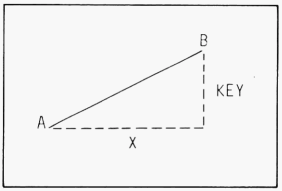
$$AB = \sqrt{X^2 \frac{LX^2}{LG^2} + Y^2 \frac{HY^2}{HT^2}}$$

- *Unité : l'unité en Y*



$$AB = \sqrt{\frac{X^2}{KE^2} + Y^2}$$

- *Unité : l'unité en X*

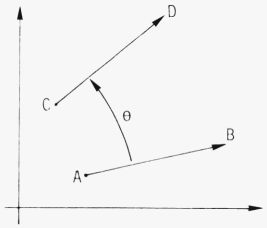


$$AB = \sqrt{X^2 + KE^2\ Y^2}$$

Angles

Les micro-ordinateurs disposent en général de quatre fonctions trigonométriques : COS, SIN, TAN et ATN, abréviations de cosinus, sinus, tangente et arctangente. Nous avons déjà utilisé les deux premières ; pour TAN, rappelons que TAN(X)=SIN(X)/COS(X). ATN est l'inverse de TAN : ATN(A) donne l'angle dont la tangente est A.

Toutes ces fonctions sont très utiles, mais il est indispensable, dans certaines applications, de calculer un cosinus ou un sinus sans que l'angle soit fourni. Nous allons donner les formules à utiliser, d'abord dans un repère orthonormé (axes perpendiculaires et même unité) à l'orientation classique, puis dans le repère de l'écran où les unités sont différentes suivant les deux axes et où l'orientation est contraire.



Les vecteurs $\vec{AB}$ et $\vec{CD}$ ayant pour coordonnées (X,Y) et (X',Y'), les formules sont :

$$\cos\theta = \frac{XX' + YY'}{\sqrt{X^2 + Y^2}\sqrt{X'^2 + Y'^2}} \qquad \sin\theta = \frac{XY' - X'Y}{\sqrt{X^2 + Y^2}\sqrt{X'^2 + Y'^2}}$$

Ici, θ est l'angle entre $\vec{AB}$ et $\vec{CD}$, l'angle entre $\vec{CD}$ et $\vec{AB}$ est (-θ).

Dans le repère de l'écran, si l'on choisit l'unité de l'axe des X, il faut remplacer Y par KEY et l'orientation étant contraire, on mettra -KEY. On obtient donc :

$$\cos\theta = \frac{XX' + KE^2\,YY'}{\sqrt{X^2 + KE^2Y^2}\sqrt{X'^2 + KE^2Y'^2}}$$

$$\sin\theta = \frac{KE\,(X'Y - XY')}{\sqrt{X^2 + KE^2Y^2}\sqrt{X'^2 + KE^2Y'^2}}$$

et pour la tangente :

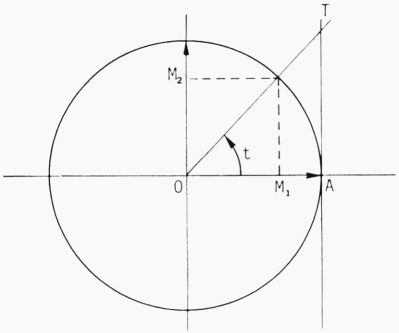
$$tg\ \theta = \frac{KE(X'Y - XY')}{XX' + KE^2\ YY'}$$

C'est la formule en définitive la plus importante car, sur la plupart des micro-ordinateurs, la seule façon de déterminer un angle est d'en calculer la tangente, puis d'utiliser la fonction ATN.

Cette fonction donne les angles en radians, il est facile de les convertir en degrés en les multipliant par 180/π. Cela étant fait, vous pourrez constater que les résultats sont toujours compris entre 90° et -90°. On obtient notamment 45° pour ATN(1) ; comme l'on sait que 45°=π/4 radians, on tapera désormais PI = 4*ATN(1) dans les programmes nécessitant cette quantité.

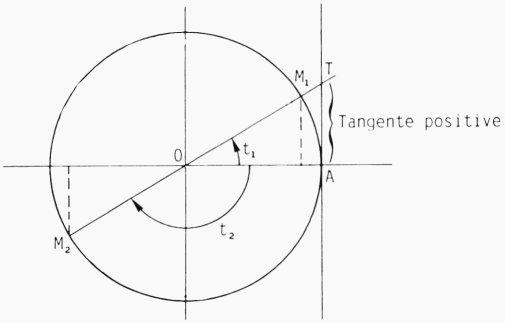
Maintenant que nous savons obtenir un angle, il reste à résoudre le problème suivant : de quelle sorte d'angle a-t-on besoin ? Si l'on parle des angles d'un triangle, il nous faut des valeurs comprises entre 0° et 180° ; si l'on parle de longitude, il nous faut des valeurs comprises entre -180° et +180° (180° est et 180° ouest) ; si l'on parle d'angle associé à un arc de cercle, il nous faut des valeurs comprises entre 0° et 360°. La fonction ATN donnant des valeurs comprises entre -90° et 90°, cela ne convient à aucun des cas précédents. Il faut donc adapter les résultats, ce qui est réalisé dans les figures de la page suivante.

Regardons d'abord le cercle trigonométrique :

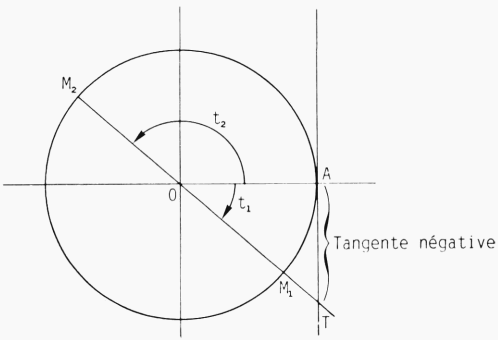


Rappelons que : $\overline{OM_1}$ = cos t, $\overline{OM_2}$ = sin t, $\overline{AT}$ = tg t.

Nous pouvons maintenant voir comment trouver un angle entre -180° et 180° :



Si la tangente est positive, on prend l'arc tangente si le cosinus est positif, l'arc tangente diminué de 180° si le cosinus est négatif.



Si la tangente est négative, on prend l'arc tangente si le cosinus est positif, l'arc tangente augmenté de 180° si le cosinus est négatif.

Si la tangente n'est pas définie (cosinus nul), on prend 90° multiplié par le signe du sinus.

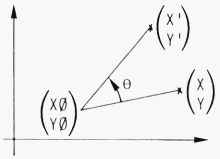
On pourra donc utiliser les formules suivantes :

- si COS=0 ATN=90*SIGN(SIN)
- si COS≠0 ATN=ATN-(1-SGN(COS))*90*SGN(ATN)

Pour obtenir un angle entre 0° et 180°, on peut utiliser ce qui précède et prendre la valeur absolue du résultat. Autre méthode possible : on remplace, dans le calcul, le sinus par sa valeur absolue (cette quantité doit être positive) et on ajoute 180° dans le cas où le cosinus est négatif.

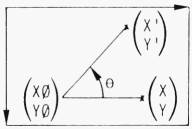
Une dernière chose avant de passer à quelques applications simples. Comment faire tourner un point autour d'un autre point d'un angle donné T ?

Dans le cas classique (repère orthonormé, orientation classique), faire tourner le point de coordonnées (X,Y) autour du point de coordonnées (XØ,YØ) se traduit par :



$$X'-X\emptyset = (X-X\emptyset)\cos T - (Y-Y\emptyset)\sin T$$
$$Y'-Y\emptyset = (X-X\emptyset)\sin T + (Y-Y\emptyset)\cos T$$

Compte tenu du repère dans lequel nous travaillons, qui n'a pas la bonne orientation et qui impose un facteur KE, ces formules deviennent :



$$X'-X\emptyset = (X-X\emptyset)\cos T + KE\,(Y-Y\emptyset)\sin T$$
$$Y'-Y\emptyset = -(X-X\emptyset)\frac{\sin T}{KE} + (Y-Y\emptyset)\cos T$$

Cela nous suffira dans la pratique : autant passer à quelques applications simples.

QUELQUES APPLICATIONS SIMPLES

La première application - angles d'un triangle - est une application assez immédiate. Les deux autres - flocons et polygones emboîtés - ont été choisies pour leur côté esthétique.

Angles d'un triangle

Problème : tracer un triangle au hasard à l'écran de votre ordinateur et afficher ses trois angles en degré (on laissera les minutes et les secondes).

La programmation de ce petit problème est très simple. Trois sous-programmes principaux sont utilisés.

- *Choix des sommets du triangle*
  (les coordonnées de ces sommets sont mises dans les tableaux X(i) et Y(i), avec $0 \leqslant i \leqslant 2$.)
- *Affichage du triangle*
  Le sous-programme utilise la fonction FNR(X) = X-3 INT(X/3) qui donne le reste de X dans la division par 3. Cette fonction sert à connaître les indices des sommets du triangle autres que le sommet d'indice i. A priori, ce sont i-1 et i+1... mais il y a problème pour i=Ø (à cause de i-1) et i=2 (à cause de i+3). La considération de FNR(i-1) et FNR(i+1) évite tout ennui.
- *Calcul et affichage des angles*
  La méthode théorique vue plus haut pour déterminer un angle entre 0 et 180 est programmée. C'est volontairement que nous n'avons pas mis les dénominateurs dans les variables CO (comme cosinus) et SI (comme sinus) : ils se simplifient dans le calcul de la tangente SI/CO.

```
10 REM ANGLES D'UN TRIANGLE
20 DEFFNR(X)=X-3*INT(X/3):REM R(X) EST LE
   RESTE,DANS LA DIVISION PAR 3 DE X
30 LG=    :HT=
40 KE=    :PI=4*ATN(1)
50 DIM X(2),Y(2)
100 REM PROGRAMME PRINCIPAL
110 GOSUB 1000:REM CHOIX DES SOMMETS
    DU TRIANGLE
120 GOSUB 2000:REM AFFICHAGE DU TRIANGLE
130 GOSUB 3000:REM CALCUL DES ANGLES
140 END
1000 REM CHOIX DES SOMMETS DU TRIANGLE
     RND(0) EST UN NOMBRE ALEATOIRE COMPRIS
     ENTRE 0 ET 1.C'EST RND(1) SUR CERTAINS
     BASICS
1010 FOR I=0 TO 2
1020 X(I)=INT(LG*RND(0))
1030 Y(I)=INT(HT*RND(0))
1040 IF I=0 THEN 1090
1050 REM REJET DES POINTS CONFONDUS
1060 FOR J=0 TO I-1
1070 IF X(I)=X(J) AND Y(I)=Y(J) THEN 1020
1080 NEXT J
1090 NEXT I
1100 RETURN
2000 REM AFFICHAGE DU TRIANGLE
2010 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
2020 FOR I=0 TO 2
2030 DROITE X(I),Y(I) A X(FNR(I+1)),Y(FNR(I+1))
2040 NEXT I
2050 RETURN
3000 REM ANGLES
3010 REM MODE TEXTE
3020 FOR I=0 TO 2
3030 X=X(FNR(I-1))-X(I):Y=Y(FNR(I-1))-Y(I)  →
```

```
3040 X1=XC(FNR(I+1))-XC(I):Y1=YC(FNR(I+1))-YC(I)
3050 GOSUB 3500:REM ANGLE ENTRE 0 ET 180
3060 A=INT(A+.5)
3070 PRINT"ANGLE";I;"=";A
3080 NEXT I
3090 RETURN
3500 REM ANGLE ENTRE 0 ET 180 DEGRES
3510 CO=X*X1+KE*Y*Y1:IF CO=0 THEN A=90:RETURN
3520 SI=ABS(KE*(X1*Y-X*Y1))
3530 A=ATN(SI/CO)
3540 IF A<0 THEN A=A+PI
3550 A=A*180/PI
3560 RETURN
```

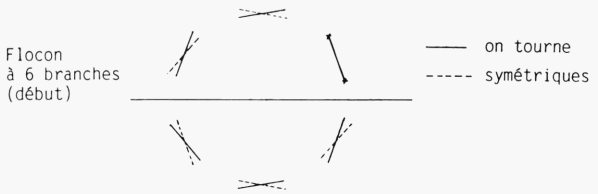
Les deux autres applications que nous vous proposons utilisent la rotation d'un point autour du centre de l'écran dont nous noterons les coordonnées XC et YC.

Flocons

Problème : faire tracer à l'écran des flocons à B branches au hasard. Pour cela, faire N fois de suite :

- choix d'un segment au hasard ;
- affichage du segment obtenu en tournant de $0, \frac{180}{B}, 2 \times \frac{180}{B}$ ..., $(B-1) \times \frac{180}{B}$ et affichage du segment symétrique par rapport à une horizontale passant par le centre de l'écran.

(N et B seront laissés au choix de l'utilisateur. Pour B=6, on obtiendra des flocons de neige classiques.)



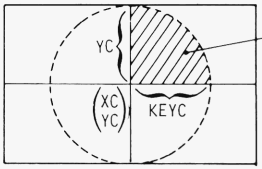
Trois sous-programmes principaux sont utilisés :

- *Choix du type de flocon* (N et B)

  Rien à dire en programmation. Dans la pratique, prendre $3 \leqslant B \leqslant 10$ et N pas trop grand... sinon le flocon se transforme en une boule épaisse.

- *Choix du segment*

  Il faut choisir le segment à l'intérieur du plus grand cercle traçable à l'écran (cercle ayant pour centre le centre de l'écran). Ce cercle est en général ainsi :



Le segment est choisi à l'intérieur du quart de cercle hachuré. Le test :

$$X(J)^2 + KE^2\,Y(J)^2 > KE^2\,YC^2$$

garantit cela. Il faut bien comprendre qu'ayant choisi l'unité sur l'axe des X, le rayon de ce cercle est KEXC ! Pour ceux que cela choque, le test peut s'écrire :

$$\frac{X(J)^2}{KE^2} + Y(J)^2 > YC^2$$

et là, le rayon du cercle est YC. Unité : celle de CY et le $\frac{X(J)^2}{KE^2} + Y(J)^2$ est le carré d'une distance vue au début du chapitre où l'unité est celle des Y. Ouf ! Tout cela est bien cohérent.)

- *Affichage des segments* déduits par rotation et par symétrie

  Pour la détermination de XØ, YØ, X1, Y1, le lecteur reconnaîtra aisément les formules vues plus haut : rotation du point de coordonnées X(i), Y(i) autour du point de coordonnées XC,YC d'un angle T.

Dans la pratique, on aura intérêt à choisir un cercle un tout petit peu plus petit que celui décrit plus haut dans des cas limites (points sur le cercle), les rotations effectuées peuvent conduire à des coordonnées inacceptables (appel illégal de fonction pour tracer les droites).

Pour éviter ces inconvénients, il vaut mieux prendre :

```
X(J) = INT((KE*R)*RND(Ø))
Y(J) = INT(R*RND(Ø))
```

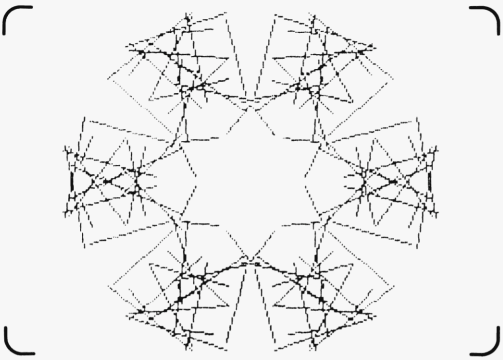
à la place de :

```
X(J) = INT((KE*YC)*RND(Ø))
Y(J) = INT(YC*RND(Ø))
```

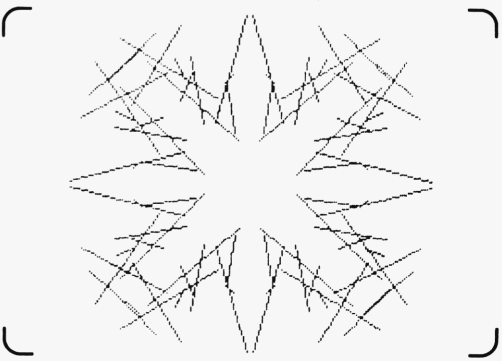
avec R plus petit que YC.

```
10 REM FLOCONS ALEATOIRES
20 XC=    :YC=
30 KE=    :PI=4*ATN(1)
40 DIM X(1),Y(1)
100 REM PROGRAMME PRINCIPAL
110 GOSUB 1000:REM TYPE DE FLOCON
120 FOR I=1 TO N
130 GOSUB 2000:REM CHOIX ALEATOIRE DU SEGMENT
140 GOSUB 3000:REM ROTATIONS ET SYMETRIES
150 NEXT I
160 END
1000 REM TYPE DE FLOCON
1010 REM MODE TEXTE,EFFACAGE D'ECRAN
1020 INPUT"NOMBRE DE SEGMENTS";N
1030 IF N<=1 THEN 1020
1040 INPUT"NOMBRE DE BRANCHES";B
1050 IF B=0 OR B=1 OR B<>INT(B) THEN 1040
1060 RETURN                                    →
```

```
2000 REM CHOIX DU SEGMENT.RND(0) OU RND(1)
     SELON VOTRE BASIC...
2010 FOR J=0 TO 1
2020 XC(J)=INT((KE*YC)*RND(0))
2030 YC(J)=INT(YC*RND(0))
2040 IF XC(J)*XC(J)+KE*KE*YC(J)*YC(J)>KE*KE*YC*YC
     THEN 2020
2050 NEXT J
2060 RETURN
3000 REM ROTATIONS ET SYMETRIES
3010 REM MODE GRAPHIQUE
3020 FOR J=0 TO B-1
3030 T=-2*PI*J/B
3040 X0=XC(0)*COS(T)+KE*YC(0)*SIN(T)
3050 Y0=-XC(0)*SIN(T)/KE+YC(0)*COS(T)
3060 X1=XC(1)*COS(T)+KE*YC(1)*SIN(T)
3070 Y1=-XC(1)*SIN(T)/KE+YC(1)*COS(T)
3080 DROITE XC+X0,YC+Y0 A XC+X1,YC+Y1 :
     REM ON TOURNE
3090 DROITE XC+X0,YC-Y0 A XC+X1,YC-Y1 :
     REM SEGMENT SYMETRIQUE
3100 NEXT J
3110 RETURN
```



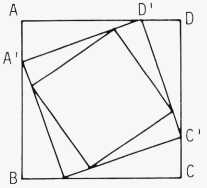
6 BRANCHES : 1Ø DROITES



4 BRANCHES : 1Ø DROITES

Polygones emboîtés

Problème : faire tracer des polygones réguliers de N côtés, emboîtés les uns dans les autres, au sens suivant :



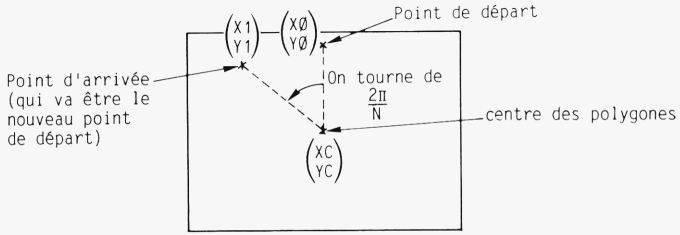
A', B', C', D' sont tels que : $\vec{AA'}=D\times\vec{AB}$, $\vec{BB'}=D\times\vec{BC}$, $\vec{CC'}=D\times\vec{CD}$, $\vec{DD'}=D\times\vec{DA}$, et ainsi de suite ($0<D<1$).

N et D seront choisis par l'utilisateur.

La programmation est simple, à condition de ne pas employer de méthodes trop lourdes (tableaux pour les coordonnées des sommets du polygone de départ...). Il suffit de se rendre compte que seules sont utiles :

- les coordonnées du centre commun à tous les polygones qui sera le centre de l'écran (XC,YC) ;
- les coordonnées d'un des sommets du polygone de départ (XØ,YØ). On peut prendre XØ=XC, YØ=Ø, mais YØ=Ø risque de conduire, par suite d'erreurs de calcul, à un appel illégal de fonction au cours d'un tracé de droite. Mieux vaut prendre YØ non nul et petit.

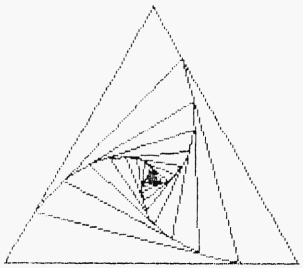
A partir de ces coordonnées, on trouve facilement, en faisant une rotation de $2\pi/N$, les coordonnées (X1,Y1) de l'autre sommet du côté du polygone. On recommence cela N fois de suite en remplaçant XØ et YØ par X1 et Y1.

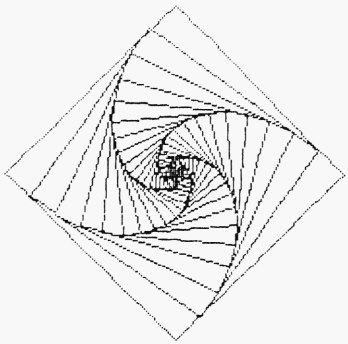



```
10 REM POLYGONES EMBOITES
20 XC=    :YC=
30 KE=    :PI=4*ATN(1)
40 XØ=XC:YØ=Ø:REM YØ EST PEUT ETRE A MODIFIER
   ...SELON LA PRECISION DE VOTRE BASIC,IL Y A
   RISQUE D'APPEL ILLEGAL DE FONCTION !!
100 REM PROGRAMME PRINCIPAL
110 GOSUB 1000:REM CHOIX DU TYPE DE DESSIN
120 GOSUB 2000:REM TRACE DU POLYGONE
130 GOSUB 3000:REM DECALAGE
140 GOTO 120
1000 REM CHOIX DU TYPE DE DESSIN                    →
```

```
1010 REM MODE TEXTE,EFFACAGE D'ECRAN
1020 INPUT"NOMBRE DE COTES DU POLYGONE";N
1030 IF N<=2 OR N<>INT(N) THEN 1020
1040 INPUT"DECALAGE ENTRE LES POLYGONES
     (ENTRE 0 ET 1)";D
1050 IF D<0 OR D>1 THEN 1040
1060 RETURN
2000 REM TRACE D'UN POLYGONE
2010 REM MODE GRAPHIQUE
2020 T=2*PI/N
2030 F=N
2040 X1=XC+(X0-XC)*COS(T)+KE*(Y0-YC)*SIN(T)
2050 Y1=YC-(X0-XC)*SIN(T)/KE+(Y0-YC)*COS(T)
2060 DROITE X0,Y0 A X1,Y1
2070 F=F-1
2080 IF F<>0 THEN X0=X1 :Y0=Y1 :GOTO 2040
2090 RETURN
3000 REM DECALAGE POUR LE POLYGONE SUIVANT
3010 X0=X0+D*(X1-X0)
3020 Y0=Y0+D*(Y1-Y0)
3030 RETURN
```

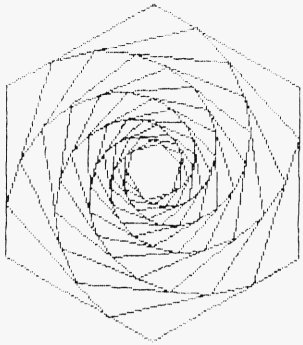
N= 3 : D= .8

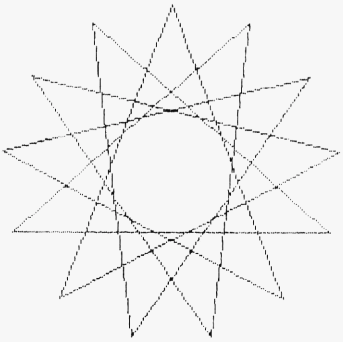




N= 4 : D= .15

N= 6 : D= .7





N= 2.6 : D= 1

Ce remplacement de XØ et YØ par X1 et Y1 n'est effectué que N -1 fois (test F < > 1). Ainsi, pour le dernier côté, on garde les coordonnées de ses deux extrémités (XØ,YØ) et (X1,Y1). Tout cela est traité par le sous-programme "polygone".

Avoir conservé les coordonnées des deux extrémités du dernier côté du polygone permet d'effectuer très facilement le décalage pour le polygone suivant : il suffit de partir du point de coordonnées :

```
XØ+D(X1-XØ)
YØ+D(Y1,YØ)
```

(sous-programme "décalage").

A l'exécution, ce petit programme montre qu'on peut tracer de jolies spirales à partir de polygones !

Il montre aussi qu'à partir de N=15 (ou un peu plus si la résolution de votre ordinateur est grande) un polygone ne se distingue pas d'un cercle. C'est un moyen pratique et simple de simuler une instruction CIRCLE si votre Basic n'en dispose pas. Pour de telles valeurs de N, tous les polygones emboîtés sont confondus avec le polygone de départ et le programme n'a que peu d'intérêt !

Pour des valeurs de N plus petites, le programme boucle sur lui-même. Vous pourrez facilement le faire arrêter en ajoutant un test sur le côté du polygone.

L'ETRANGE UNIVERS DES FRACTALS

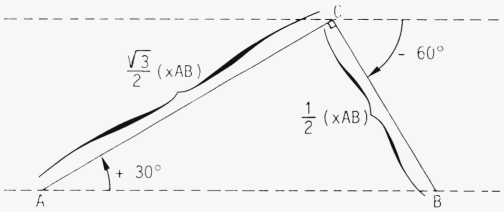
La méthode utilisée

Pour ceux qui ignorent tout ou presque des "fractals", mieux vaut d'abord regarder les exemples des pages 68-69 et 73 à 78 (courbes de Van Koch, de Peano). Ces exemples furent historiquement les premiers objets fractals considérés par les mathématiciens. Nous nous intéressons surtout à eux parce qu'ils sont constitués de segments de droites bizarrement agencés. Bizarrement agencés c'est sûr... mais tout de même avec une certaine logique. Depuis, on a fait mieux, ou plus exactement on a fait plus "design", en remplaçant la trop vieille courbe de Peano par des colliers péaniens, en remplaçant des segments trop monotones par des arcs de cercle (voir "Penser les mathématiques").

Toujours est-il que nous considèrerons des objets fractals comme étant définis par :

- un motif de base formé de N segments de droites (donc N+1 points) ;
- la construction (à une certaine échelle) répétée indéfiniment de ce motif de base sur chacun des segments. Dans la pratique, le "indéfiniment" sera limité à un certain nombre d'étapes F.

Pour expliquer la méthode suivie pour fractaliser n'importe quel motif, prenons un exemple peu esthétique mais simple :

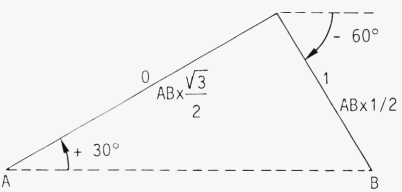


Des quantités importantes pour la suite sont notées sur cette figure :

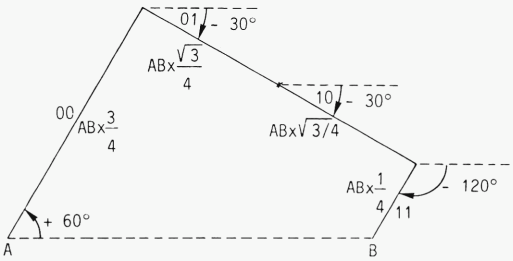
- les angles des segments du motif par rapport à AB (pour être très précis : $(\vec{AB},\vec{AC})$ et $(\vec{AB},\vec{CB})$) ;
- les distances AC et CB, en prenant AB pour unité (pour être très précis : $\frac{AC}{AB}$ et $\frac{CB}{AB}$)

A l'étape F, la courbe comprend N^F segments puisque à chaque étape, le nombre initial de segments est multiplié par N. Numérotons ces segments, dans l'ordre où ils sont construits, par un nombre de F chiffres en base N. F chiffres est important : s'il y a des zéros, il faut les faire figurer. Là, les choses se corsent, alors reprenons l'exemple vu plus haut.

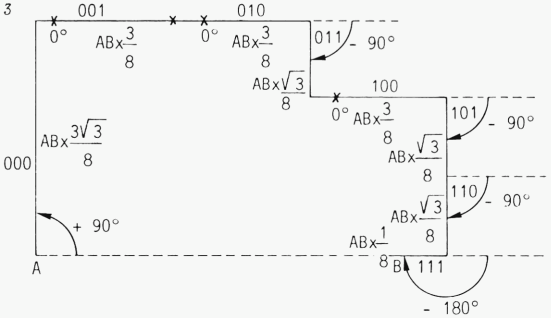
Etape n° 1



Etape n° 2



Etape n° 3



Appelons maintenant A(Ø)(=30), A(1)(=-60) les angles et D(Ø)(=$\sqrt{3}/2$) et D(1)(=1/2) les distances que nous avons qualifiés d'importants. Dressons un tableau où figurent : le numéro du segment, l'angle qu'il fait avec AB et sa longueur (en prenant AB pour unité) :

Numéro du segment	*Angle avec AB*	*Longueur par rapport à AB*
0	30 = A(Ø)	$\sqrt{3}/2$ = D(Ø)
1	- 60 = A(1)	1/2 = D(1)
00	60 = A(Ø) + A(Ø)	3/4 = D(Ø) x D(Ø)
01	- 30 = A(Ø) + A(1)	$\sqrt{3}/4$ = D(Ø) x D(1)
10	- 30 = A(1) + A(Ø)	$\sqrt{3}/4$ = D(Ø) x D(1)
11	-120 = A(1) + A(1)	1/4 = D(1) x D(1)
000	90 = A(Ø) + A(Ø) + A(Ø)	$3\sqrt{3}/8$ = D(Ø) x D(Ø) x D(Ø)
001	0 = A(Ø) + A(Ø) + A(1)	3/8 = D(Ø) x D(Ø) x D(1)
010	0 = A(Ø) + A(1) + A(Ø)	3/8 = D(Ø) x D(Ø) x D(1)
011	- 90 = A(Ø) + A(1) + A(1)	$\sqrt{3}/8$ = D(Ø) x D(1) x D(1)
100	0 = A(1) + A(Ø) + A(Ø)	3/8 = D(1) x D(Ø) x D(1)
101	- 90 = A(1) + A(Ø) + A(1)	$\sqrt{3}/8$ = D(1) x D(Ø) x D(1)
110	- 90 = A(1) + A(1) + A(Ø)	$\sqrt{3}/8$ = D(1) x D(1) x D(Ø)
111	-180 = A(1) + A(1) + A(1)	1/8 = D(1) x D(1) x D(1)

Une règle simple (et oui !) se dégage : si les F chiffres du numéro du segment, en base N, sont : $c_F\ c_{F-1} \ldots c_1$ (à l'étape E, il y a F chiffres y compris les éventuels zéros), l'angle que ce segment fait avec AB est :

$$A(c_F) + A(c_{F-1}) + \ldots + A(c_1)$$

et sa longueur, par rapport à AB est :

$$D(c_F) \times D(c_{F-1}) \times \ldots \times D(c_1)$$

ça y est : l'essentiel de la méthode est dit !

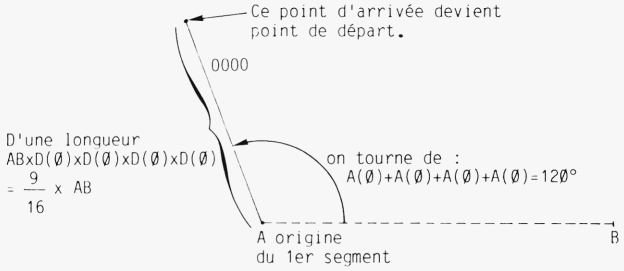
On vérifie, pas toujours très facilement, ces règles sur les courbes de Van Koch, de Peano... Une réflexion un peu plus approfondie permet de justifier ces règles, nous en ferons grâce au lecteur qui a fait l'effort de nous lire jusque-là.

De façon un peu plus algorithmique, la fractalisation d'un motif de N segments, donc de N+1 points, à l'étape F, s'opérera ainsi :

- on détermine les angles et les longueurs des segments du motif par rapport au segment joignant les points de départ et d'arrivée du motif ;
- à l'étape F, pour tracer le énième segment ($0 \leqslant i \leqslant N^F-1$) :
  - on décompose i en base N (avec ses F chiffres) ;
  - on trace un segment de longueur $D(c_F) \times D(c_{F-1}) \times \ldots \times D(c_1)$ (par rapport à AB), faisant un angle $A(c_F) + A(c_{F-1}) + \ldots + A(c_1)$ avec AB.

Ce tracé de segments se fait de proche en proche. Le point de départ du premier segment est le point de départ du motif. Ce qui précède permet de déterminer le point d'arrivée du premier segment qui sert de point de départ du second segment ... et ainsi de suite.

Pour *l'étape n° 4* de l'exemple choisi, cela donne :



Longueur = $AB \times D(Ø) \times D(Ø) \times D(Ø) \times D(1) = AB \times \frac{3\sqrt{3}}{16}$.

Evidemment, les segments tracés doivent être mis bout à bout, mais ce n'est pas un problème difficile pour un ordinateur !

Programmation

Maintenant, nous pouvons passer à la programmation de la "fractalisation" de n'importe quel motif. Le programme principal comportera trois sous-programmes :

- *définition d'un motif* par dialogue interactif (à partir de la ligne 1000) ;
- *analyse du motif* (à partir de la ligne 2000) : les angles et les distances sont mis dans les tableaux A(N-1) et D(N-1) ;
- *"fractalisation"* proprement dite (à partir de la ligne 3000). Les étapes successives sont visualisées. On passe d'une étape à une autre en appuyant sur n'importe quelle touche. Compte tenu du temps de calcul, des résolutions graphiques, nous avons limité le nombre d'étapes NF à 5 (ligne 1070) : vous pouvez modifier cette limite facilement !

Grosso modo, tous les sous-programmes intervenant ici ont déjà été étudiés. Bornons-nous à quelques précisions :

- par rapport au sous-programme de dialogue interactif vu au chapitre Déformations, pour une fractalisation, on ne peut pas accepter que les points de départ A et d'arrivée B du motif soient confondus (les angles et distances par rapport à AB n'auraient pas de sens). De plus, les "couleurs" des segments définissant le motif ne sont pas demandées : tout segment est supposé vu.
- le sous-programme analyse du motif (détermination des angles et des distances) utilisera les méthodes vues au début du chapitre (problème de détermination d'angles d'un triangle)... mais là, les angles devront être considérés entre -180° et +180° (et non pas entre 0 et 180° seulement). Quant aux distances, les formules ont été vues.

On ajoutera, à ce sous-programme, la détermination d'un angle dont nous n'avons, pour l'instant, pas parlé pour ne pas trop compliquer des choses parfois délicates. Rien, en effet, n'indique que les points A et B, introduits par l'utilisateur, définiront une droite horizontale : dans l'exemple choisi, AB est horizontale dans un souci de simplification... mais regardez la courbe de Peano. L'angle dont on doit tourner est dans tous les cas :

$$A(c_F) + A(c_{F-1}) + \ldots + A(c_1) + \underline{A},$$

angle que fait AB avec une horizontale !

Nous ajouterons donc la détermination de A à ce sous-programme.

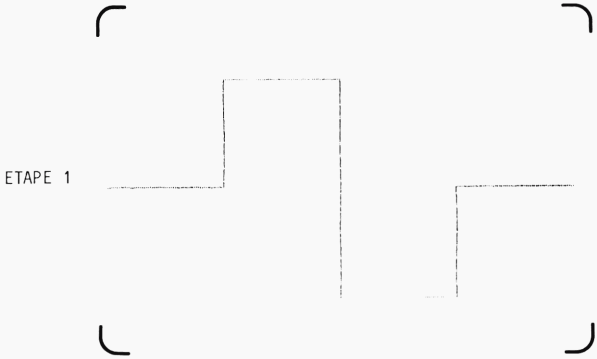
- le sous-programme visualisation à l'étape F comprend en plus d'une décomposition en base N (pas difficile à rédiger) le tracé d'un segment obtenu par rotation : cela a déjà été vu (programme flocons, polygones emboîtés...). La seule précaution à prendre est d'ajouter un test pour savoir si les points considérés restent encore sur l'écran avant de tracer le segment, certains fractals, comme sur l'exemple choisi, s'étendent indéfiniment.

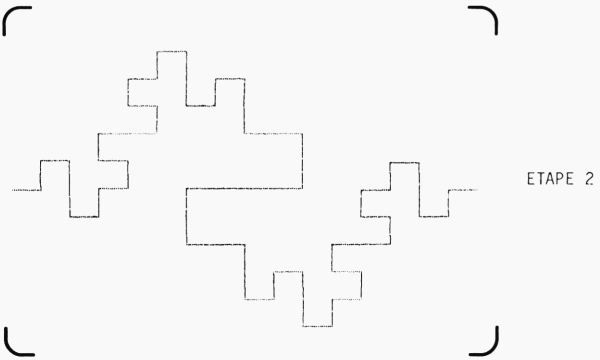
Tout ce dont nous avons besoin a déjà été traité : les "fractals" constituent un exercice d'application peu évident, mais très formateur...

```
10 REM OBJETS FRACTALS
20 LG=    :HT=
30 KE=    :PI=4*ATN(1)
100 REM PROGRAMME PRINCIPAL
110 GOSUB 1000:REM CHOIX DU MOTIF
120 GOSUB 2000:REM ANGLES ET DISTANCES
130 GOSUB 3000:REM FRACTALISATION
140 END
1000 REM CHOIX DU MOTIF
1010 REM MODE TEXTE,EFFACAGE D'ECRAN        →
```

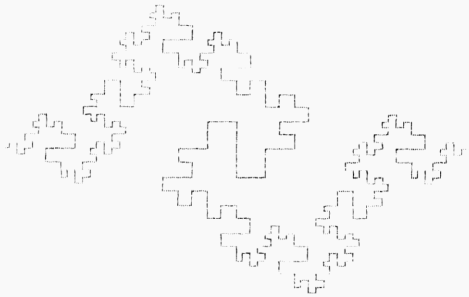
```
1020 INPUT"NOMBRE DE POINTS";N
1030 IF N<=2 OR N<>INT(N) THEN 1020
1040 N=N-1
1050 DIM X(N+1),Y(N+1),A(N-1),D(N-1)
1060 INPUT"NOMBRE D'ITERATIONS (2 A 5)";NF
1070 IF NF<2 OR NF>5 THEN 1060
1080 REM EFFACAGE D'ECRAN
1090 FOR I=0 TO N+1
1100 REM MODE TEXTE,EFFACAGE D'ECRAN
1110 PRINT"POINT" I+1
1120 INPUT"ABSCISSE";X(I)
1130 INPUT"ORDONNEE";Y(I)
1140 IF X(I)>LG OR Y(I)>HT THEN 1100
1150 IF X(I)<0 OR Y(I)<0 THEN GOSUB 1700 :
     GOTO 1100
1160 IF I=N AND X(I)=X(0) AND Y(I)=Y(0)
     THEN I=I-1 GOTO 1180
1170 IF I<>N+1 THEN GOSUB 1500
1180 NEXT I
1190 RETURN
1500 REM AFFICHAGE
1510 REM MODE GRAPHIQUE
1520 IF I=0 THEN POINT X(0),Y(0):RETURN
1530 DROITE X(I-1),Y(I-1) A X(I),Y(I)
1540 RETURN
1700 REM RECTIFICATION
1710 REM MODE GRAPHIQUE
1720 IF I=0 THEN RETURN
1730 I=I-1
1740 REM EFFACAGE D'ECRAN
1750 IF I=0 THEN RETURN
1760 IF I=1 THEN POINT X(0),Y(0):RETURN
1770 REM TRACE DU MOTIF
1780 FOR J=0 TO I-2
1790 DROITE X(J),Y(J) A X(J+1),Y(J+1)
1800 NEXT J
1810 RETURN
2000 REM ANGLES ET DISTANCES
2010 DX=X(N)-X(0)
2020 DY=Y(N)-Y(0)
2030 D=SQR(DX*DX+KE*KE*DY*DY)
2040 IF DX=0 THEN A=-PI/2*SGN(DY):GOTO 2060
2050 A=ATN(-KE*DY/DX):IF DX<0 THEN A=A+PI
2060 FOR I=0 TO N-1
2070 X=X(I+1)-X(I)
2080 Y=Y(I+1)-Y(I)
2090 D(I)=SQR(X*X+KE*KE*Y*Y)/D
2100 SI=KE*(Y*DX-X*DY)
2110 CO=X*DX+KE*KE*Y*DY
2120 IF CO=0 THEN A(I)=-PI/2*SGN(SI):GOTO 2140
2130 A(I)=ATN(-SI/CO):IF CO<0 THEN A(I)=A(I)+PI
2140 NEXT I
2150 RETURN
3000 REM FRACTALISATION
3010 FOR F=1 TO NF
3020 REM MODE TEXTE,EFFACAGE D'ECRAN
3030 PRINT"ETAPE NUMERO ";F
3040 Z1=X(0):Z2=Y(0)
3050 X0=X(0):Y0=Y(0)
3060 FOR I=0 TO NF -1:REM N ELEVE A LA
     PUISSANCE F MOINS 1
3070 REM MODE GRAPHIQUE
3080 R=I:A1=A:D1=D                                    →
```

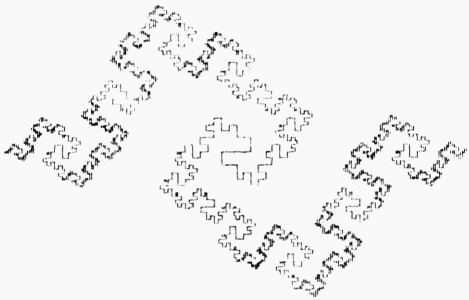
```
3090 GOSUB 3500:REM DECOMPOSITION DE R EN BASE N
3100 Z3=Z1+D1*COS(A1)
3110 X1=INT(Z3+.5)
3120 Z4=Z2-D1*SIN(A1)/KE
3130 Y1=INT(Z4+.5)
3140 IF X0<0 OR X0>LG OR Y0<0 OR Y0>HT OR X1<0
     OR X1>LG OR Y1<0 OR Y1>HT THEN 3180
3150 DROITE X0,Y0 A X1,Y1
3160 X0=X1:Y0=Y1
3170 Z1=Z3:Z2=Z4
3180 NEXT I
3190 A$=INKEY$:REM OU GET A$
3200 IF A$="" THEN 3190
3210 NEXT F
3220 RETURN
3500 REM DECOMPOSITION EN BASE N
3510 FOR J=1 TO F
3520 Q=INT(R/N)
3530 R1=R-Q*N
3540 A1=A1+A(R1)
3550 D1=D1*D(R1)
3560 R=Q
3570 NEXT J
3580 RETURN
```





ETAPE 3



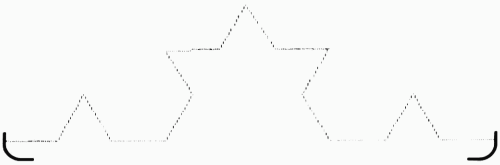


ETAPE 4

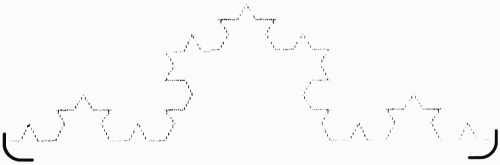
Courbe de Van Koch

ETAPE 1

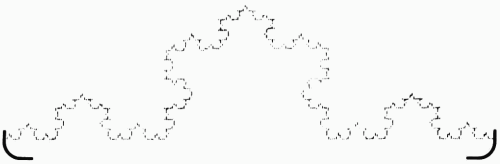
ETAPE 2

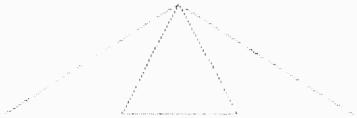


ETAPE 3



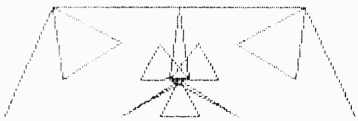
ETAPE 4

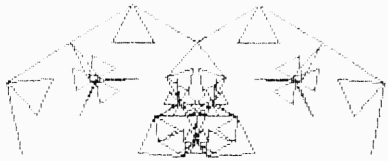




ETAPE 1

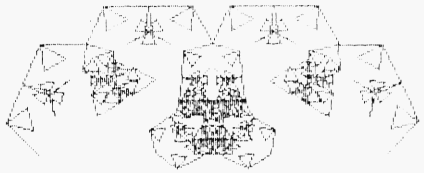
ETAPE 2





ETAPE 3

ETAPE 4



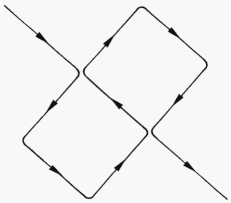
Il est difficile de ne pas dire pourquoi des objets fractals ont acquis une certaine célébrité... au moins chez les mathématiciens. Ces petits monstres ont été domestiqués mais, au départ, ils soulevèrent des questions difficiles.

Regardez bien la courbe de Van Koch (qui est la limite de toutes les étapes de fractalisation) : sans jamais repasser par le même point, elle a une longueur infinie, tout en délimitant une surface finie ! Maintenant, vous avez une réponse mais si, a priori, on vous demandait, avec une ficelle infinie, et sans repasser par le même point, de délimiter une surface finie..., votre intuition serait sans doute déroutée !

Regardez aussi la courbe de Peano. On comprend assez facilement qu'elle finisse par remplir un carré. Voilà une courbe, disons une ligne, sans épaisseur qui se comporte comme une surface ! Là encore, notre intuition peut être déroutée. Cela a contribué à donner aux fractals une dimension comprise entre 1 (celle d'une droite) et 2 (celle d'un plan)...

Courbe de Peano

ETAPE 1



Certaines parties ont été arrondies pour mieux voir le chemin parcouru.

ETAPE 2

Si vous passez sans difficulté à l'étape n° 3, vous avez droit à toutes nos félicitations.

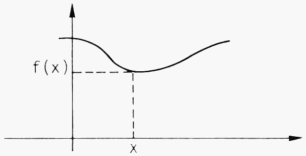
Deuxième partie
COURBES

Chapitre I
Courbes

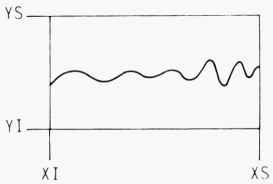
COURBES REPRESENTATIVES D'UNE FONCTION

Tracé

Nous allons tracer, dans ce paragraphe, des courbes d'équation y=f(x), f étant une fonction définie par différents moyens.



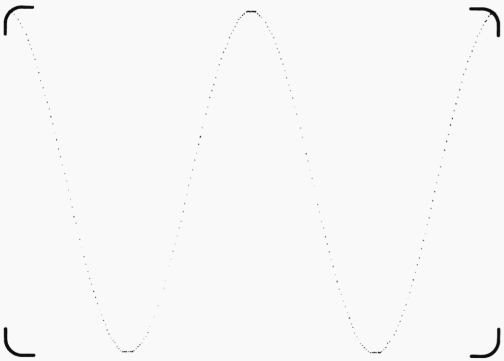
Il nous faut d'abord définir notre repère, c'est-à-dire savoir quelle partie de la courbe nous allons regarder et avec quelle échelle elle sera tracée.



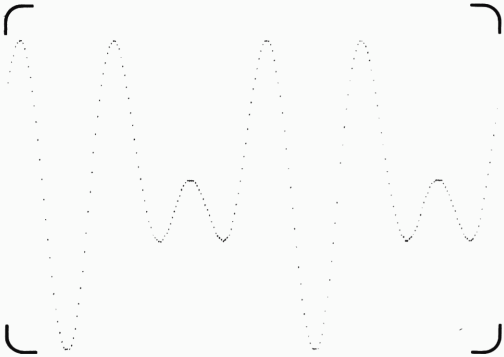
Le choix des deux bornes XI et XS (abscisse inférieure et abscisse supérieure) sera évidemment laissé à l'utilisateur. On peut également choisir au clavier YI et YS (ordonnée minimum et ordonnée maximum), mais cela peut présenter des inconvénients : la mise en page peut être mal choisie, avec une échelle trop grande qui écrase la courbe, ou un intervalle trop petit qui occasionne un message d'erreur (la courbe se traçant en dehors de l'écran). Sinon, on peut très facilement calculer le minimum et le maximum de la fonction pour assurer une belle mise en page. Malheureusement, le

temps de calcul peut être un peu long. Il vaut mieux laisser le choix entre les deux méthodes, ce que fait le programme suivant :

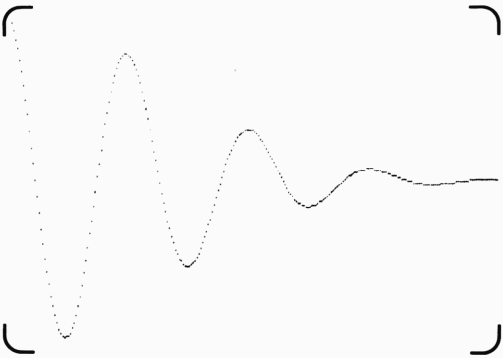
```
10 REM COURBE D'EQUATION Y=F(X)
20 DEFFNF(X)=
30 LG=    HT=
100 REM INTERVALLE D'ETUDE
110 REM MODE TEXTE,EFFACAGE D'ECRAN
120 INPUT"BORNES EN X";XI,XS
130 IF XS<=XI THEN 120
140 INPUT"CALCUL DE MINIMUM ET DE MAXIMUM";A$
150 IF LEFT$(A$,1)<>"N" THEN 210
160 INPUT"BORNES EN Y";YI,YS
170 IF YS<=YI THEN 160
180 GOTO 300
200 REM MINIMUM ET MAXIMUM
210 YI=1E30
220 YS=-1E30
230 FOR I=0 TO LG
240 X=XI+I*(XS-XI)/LG
250 Y=FNF(X)
260 IF Y<YI THEN YI=Y
270 IF Y>YS THEN YS=Y
280 NEXT I
300 REM TRACE
310 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
320 FOR I=0 TO LG
330 X=XI+I*(XS-XI)/LG
340 Y=FNF(X)
350 C=HT*(YS-Y)/(YS-YI)
360 POINT I,C
370 NEXT I
380 END
```



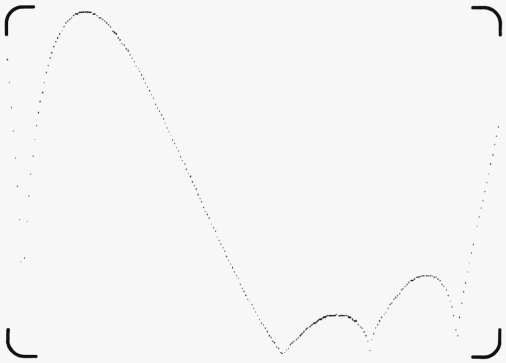
FNF(X)=COS(X)
XI=-6.28 : XS=6.28
YI=-1 : YS=1



```
FNF(X)=COS(2*X)+SIN(3*X)
XI=-6.28 : XS=6.28
YI=-2    : YS=2
```



```
FNF(X)=EXP(-X*X/8)*COS(4*X)
XI= Ø : XS=6.28
YI=-1 : YS=1
```



```
FNF(X)=SQR(ABS(X*X*(X-1)*(X-2)*(X+3)))
XI=-3.2 : XS=2.5
YI= Ø   : YS=7.3
```

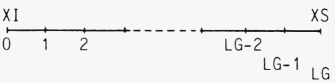
On a utilisé ici l'instruction **DEFFN.** Pour les Basics qui en sont dépourvus, on pourra utiliser des sous-programmes. Ceci est d'ailleurs indispensable dans certains cas : fonctions définies par intervalles, fonctions non définies en un point. On peut, par exemple, modifier le programme précédent en supprimant la ligne 2Ø, en remplaçant les lignes 25Ø et 34Ø (appels de la fonction) par GOSUB 1ØØØ, et en rajoutant les lignes suivantes :

```
1000 REM SI VOUS N'AVEZ PAS DE DEFFN(X)
1010 IF X=0 THEN Y=1:RETURN
1020 Y=SIN(X)/X:RETURN
```

La fonction $f(x) = \frac{\sin x}{x}$ n'est évidemment pas définie pour x=0 (qui n'a jamais vu son ordinateur afficher DIVISION BY ZERO ERROR ?). On fixe sa valeur à 1 pour x=0, ce qui nécessite un test dans le programme. Le même procédé pourra être employé chaque fois qu'une fonction n'est pas définie pour une valeur donnée.

Commentons maintenant le programme :

- les **lignes 1ØØ à 18Ø** permettent d'introduire les bornes XI et XS, puis les bornes YI et YS si l'utilisateur le désire, sinon on passe au calcul de minimax ;
- les **lignes 2ØØ à 28Ø** calculent le minimum et le maximum de la fonction sur l'intervalle [XI,XS]. On commence par fixer le minimum YI à une valeur très forte (1E30), puis on le remplace par toute valeur inférieure rencontrée. Le procédé est symétrique pour le maximum. On fait le calcul pour tous les points qui seront affichés (l'indice I correspond à l'abscisse-écran), soit LG+1 points. Nous devons donc diviser l'intervalle [XI,XS] en LG parties de longueur (XS-XI)/LG.



- les **lignes 3ØØ à 37Ø** assurent le tracé de la fonction. On trouve, à la ligne 35Ø, la règle de trois calculant l'ordonnée-écran de

chaque point. Il ne faut pas s'étonner de voir -Y dans cette formule. N'oublions pas que l'orientation du repère de l'écran est contraire à l'orientation classique.

Nous verrons plus loin les limites de ce programme et les améliorations que l'on peut y apporter. Mais auparavant, nous allons donner un petit catalogue de fonctions intéressantes, soit parce qu'elles sont bien utiles à connaître, soit parce qu'elles donnent de jolies courbes. Vous pourrez également utiliser toute la bibliothèque de fonctions de votre ordinateur pour tracer des courbes d'équation extrêmement compliquée (en employant +, -, *, /, ↑, ABS, ATN, TAN, COS, SIN, LOG, EXP, INT, SQR, que ne peut-on faire !).

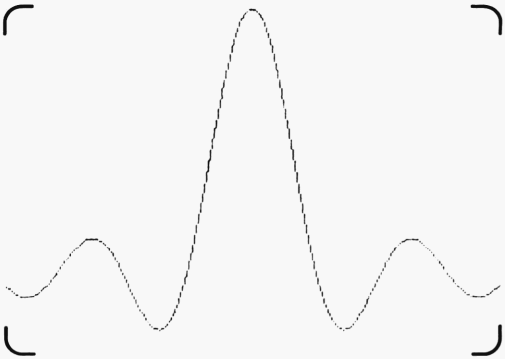
Catalogue de fonctions y=f(x)

FNF(X)=	XI, XS	XI, YS
X	-5, 5	-5, 5
COS(X)	-6.28, 6.28	-1, 1
COS(2*X)+SIN(3*X)	-6.28, 6.28	-2, 2
ABS(X)	-2, 2	0, 2
ABS(SIN(X))	-6.28, 6.28	0, 1
SQR(X)	0, 4	0, 2
SQR(ABS(X/2+SIN(2*X)))	-6.28, 6.28	0.24, 1.77
INT(X)	0, 6.5	0, 6
X-INT(X)	0, 5	0, 1
X↑2	-2, 2	0, 4
ATN(X)	-10, 10	-1.5, 1.5
EXP(-X*X)	-3, 3	0, 1
EXP(-X*X/8)*COS(4*X)	0, 6.28	-1, 1
SQR(ABS(X*X*(X-1)*(X-2)*(X+3)))	-3.2, 2.5	0, 7.3

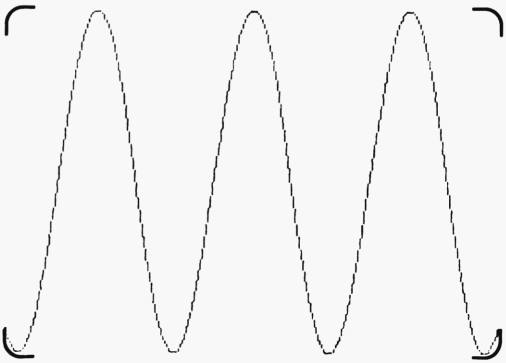
Modifications

L'utilisateur de ce programme pourra trouver certaines courbes peu esthétiques (essayez par exemple COS(X) entre -10 et +10). En effet, lorsque les variations de la fonction sont importantes, on obtient des points nettement séparés sur l'écran. On peut facilement remédier à cela en reliant les points entre eux, il suffit de remplacer les lignes 3ØØ à 38Ø (tracé de la fonction point par point) par les suivantes :

```
300 REM TRACE
310 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
320 Y=FNF(XI)
330 A=0
340 B=HT*(YS-Y)/(YS-YI)
350 FOR I=1 TO LG
360 X=XI+I*(XS-XI)/LG
370 Y=FNF(X)
380 C=HT*(YS-Y)/(YS-YI)
390 DROITE A,B  A  I,C
400 A=I
410 B=C
420 NEXT I
430 END
```



```
FNF(X)=SIN(X)/X
XI=-12 : XS=12
YI=-.3 : YS= 1
```



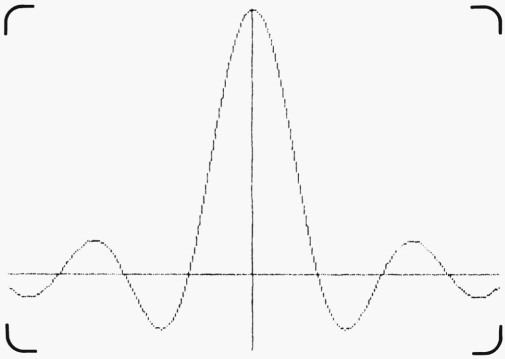
```
FNF(X)=COS(X)
XI=-1Ø : XS=1Ø
YI=-1  : YS= 1
```

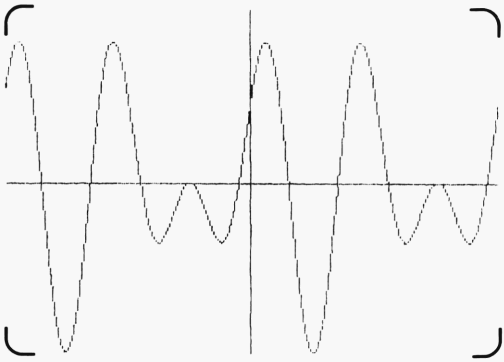
On obtient maintenant des tracés plus continus, mais il convient évidemment de ne pas employer cette méthode pour représenter des fonctions non continues (INT(X), par exemple).

On peut aussi aisément rajouter le tracé des axes du repère (c'est toujours par là que les potaches commencent leurs constructions de courbes), dans le cas bien sûr où l'origine (coordonnées Ø,Ø) est comprise dans la fenêtre apparaissant à l'écran. Il faut rajouter les lignes suivantes :

```
115 INPUT"TRACE DES AXES (O/N)";B$
315 IF B$="O" THEN GOSUB 2000
2000 REM TRACE DES AXES
2010 U=LG*(-XI)/(XS-XI)
2020 V=HT*YS/(YS-YI)
2030 IF XI*XS<=0 THEN DROITE Ø,V  A  LG,V :
     REM AXE OX
2040 IF YI*YS<=0 THEN DROITE U,0  A  U,HT :
     REM AXE OY
2050 RETURN
```

```
FNF(X)=SIN(X)/X
XI=-12 : XS=12
YI=-.3 : YS= 1
```





```
FNF(X)=COS(2*X)+SIN(3*X)
XI=-6.28 : XS=6.28
YI=-2    : YS=2
```

Nous avons pour l'instant corrigé des défauts mineurs de ce programme. Tel qu'il est, on pourrait croire que dans tous les cas, on obtient bien le minimum et le maximum de la fonction étudiée et que l'allure de la courbe est satisfaisante. Hélas, il n'en est rien, et il est très facile de le faire se tromper gravement.

Essayez de faire tracer la fonction F(X)=1/X avec XI= -3, XS= +3. Impossible, direz-vous, puisque l'on va passer par Ø et que la fonction n'est pas définie pour cette valeur. Pourtant, cela ne gêne absolument pas l'ordinateur qui va tracer deux branches d'hyperbole sans se poser de questions (si, par miracle, il refuse, prenez XI= -3, XS= +3.1, et tout passera). Si, après avoir fait tourner le programme, vous l'interrogez en mode direct sur la valeur du minimum YI et du maximum YS, vous obtiendrez approximativement les valeurs suivantes :

```
YI= -LG/3
YS=  LG/3
```

En recommençant l'étude avec XI= -10, XS= +10, on aura :

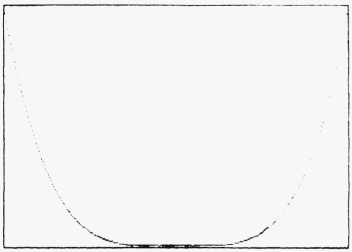
YI= -LG/10
YS= LG/10

Voilà des résultats bien inquiétants quand on sait que la fonction étudiée n'est pas définie pour X=Ø et qu'elle prend des valeurs infinies du voisinage de Ø. Or, on n'obtient le message DIVISION BY ZERO ERROR que si l'on prend l'une des bornes XI ou XS égale à Ø, et on ne voit jamais OVERFLOW ERROR (dépassement de capacité).

On comprend facilement ce qui se passe : la machine ne calcule la valeur de la fonction que pour LG+1 valeurs de l'intervalle d'étude, et ne tombe pas forcément juste sur la valeur Ø. Après cet exemple, on voit qu'il ne faut pas demander à ce programme plus qu'il ne peut faire : calculer avec précision un maximum et un minimum demande d'autres méthodes et voir l'allure générale d'une fonction ne dispense pas d'une étude sérieuse.

Il faut également se méfier de l'allure de la courbe. Si l'on fait tracer la fonction FNF(X)=(X-1)↑2*(X+1)↑2, on peut obtenir des résultats différents suivant l'intervalle d'étude choisi.

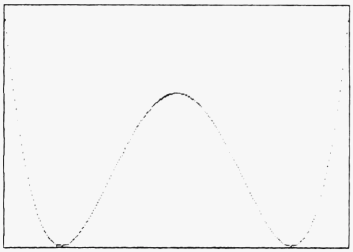
En prenant XI= -1Ø, XS= 1Ø, on obtient l'allure suivante :



On a approximativement :

YI = Ø
YS = 1ØØØØ

En prenant XI= -1.5, XS= 1.5, on obtient :



YI = Ø
YS = 1.56

Les deux tracés ne semblent pas représenter la même fonction. Les variations apparaissant dans le deuxième cas sont écrasées dans le premier. Le maximum local de la deuxième figure a une ordonnée de 1. On comprend qu'il disparaisse dans le premier cas : les ordonnées varient de Ø à 1ØØØØ ; si l'écran a une résolution verticale de 200 points, il faudrait une ordonnée de 1ØØØØ/2ØØ, soit 50 pour pouvoir le distinguer. Ce phénomène arrive assez fréquemment et, dès que la fonction prend des valeurs de grande valeur absolue, une partie de la courbe se retrouve écrasée.

VARIANTES ORNEMENTALES

Les courbes d'équation y=f(x) ne donnent qu'un point pour une abscisse donnée. Les résultats peuvent ne pas être déplaisants, mais ils sont rarement très spectaculaires. Comment améliorer les choses ? La réponse est évidente, c'est en traçant plusieurs courbes.

Translations

La première idée qui vient à l'esprit est d'étager plusieurs courbes les unes au-dessus des autres, ce qui est réalisé dans le programme ci-dessous :

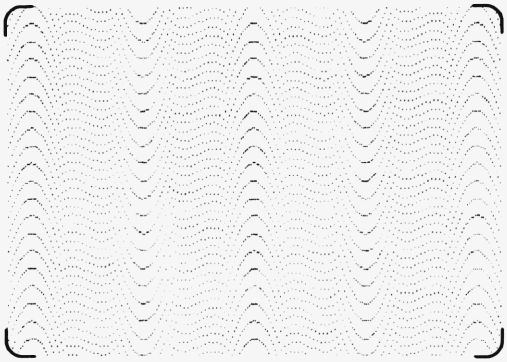
```
10 REM COURBES D'EQUATION Y=F(X),VARIANTE
   ORNEMENTALE
20 DEFFNF(X)=
30 LG=    :HT=
100 REM INTERVALLE D'ETUDE
110 REM MODE TEXTE,EFFACAGE D'ECRAN
120 INPUT"BORNES EN X";XI,XS
130 IF XS<=XI THEN 120
140 INPUT"BORNES EN Y";YI,YS
150 IF YS<=YI THEN 140
160 INPUT"NOMBRE DE COURBES";N
170 IF N<=0 OR N<>INT(N) THEN 160
180 K=HT/N
300 REM TRACE
310 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
320 FOR I=0 TO LG
330 X=XI+I*(XS-XI)/LG
340 Y=FNF(X)
350 C=HT*(YS-Y)/(YS-YI)
360 FOR J=0 TO N
370 D=C+K*J
380 IF D>HT THEN 410
390 POINT I,D
400 NEXT J
410 FOR J=0 TO N
420 D=C-K*J
430 IF D<0 THEN 460
440 POINT I,D
450 NEXT J
460 NEXT I
470 END
```

Nous avons gardé les mêmes notations que pour les programmes précédents. Au départ, l'utilisateur doit donner les bornes XI, XS, YI et YS, ainsi que le nombre de courbes qu'il désire voir apparaître. Le nombre K calculé à la ligne 180 représente la distance-écran verticale dont seront séparées les différentes courbes. Pendant le tracé, on calcule d'abord l'ordonnée-écran C de la courbe de départ. On décale vers le bas en ajoutant K tant que l'on peut (lignes 360 à 400), puis on décale vers le haut (lignes 410 à 450).

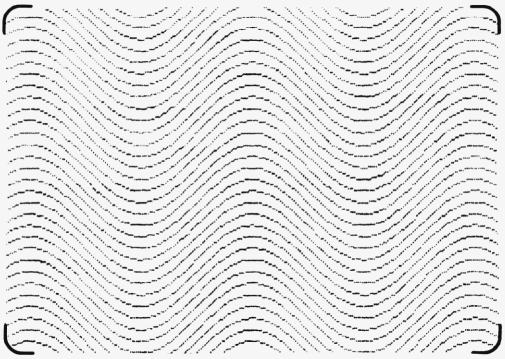
En faisant tourner ce programme, il faut s'assurer que la fonction sera bien comprise entre les bornes YI et YS, ce qui ne devrait pas être trop difficile avec le catalogue de fonctions dont vous disposez. Le nombre N de courbes à tracer pour obtenir un effet satisfaisant dépend bien sûr de la résolution de votre système, mais surtout de l'échelle choisie. Pour COS(X) par exemple, si l'on prend XI= -7, XS= +7, YI= -1, YS= +1, N=10 sera suffisant. A partir de N=20, on obtiendra des effets de moirés parfois

fort intéressants, mais on ne reconnaîtra plus la courbe de départ. Si l'on réduit l'échelle en prenant YI=-5, YS=+5, on pourra tracer davantage de courbes.

Pour les systèmes disposant de plusieurs couleurs, on pourra faire tracer chaque courbe d'une couleur différente. Si, par exemple, vous disposez de huit couleurs numérotées de Ø à 7, vous pouvez prendre, aux lignes 39Ø et 44Ø, le reste de la division de l'indice J par 8 comme numéro de couleur (R=J-8*INT(J/8)). L'effet produit est alors très spectaculaire.



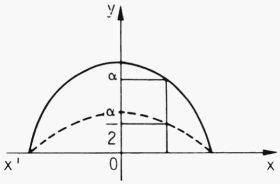
```
FNF(X)=COS(X)
XI=-7 : XS=7
YI=-1 : YS=1
N =2Ø
```



```
FNF(X)=COS(X)
XI=-7 : XS=7
YI=-5 : YS=5
N =3Ø
```

Affinités

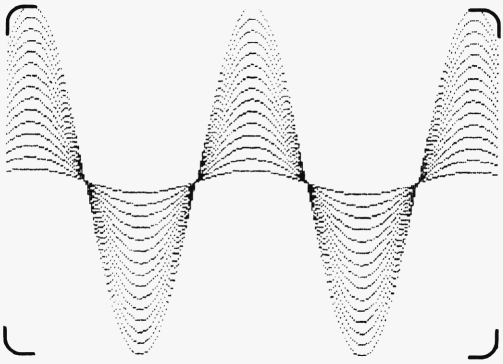
Un autre moyen d'obtenir plusieurs courbes est de faire subir, à la courbe de départ, une transformation que les matheux appellent "affinité".



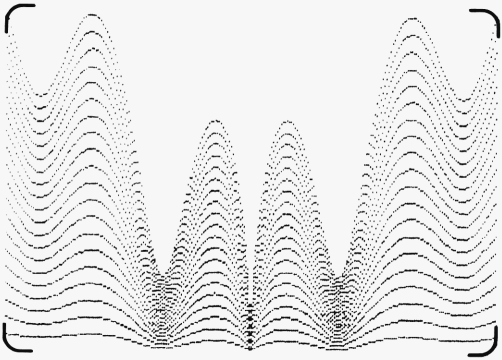
Si l'on divise les ordonnées des points de la courbe par 2, on dit que l'on a appliqué une affinité d'axe x'Ox et de rapport 1/2.

Pour tracer N courbes, on pourra prendre des affinités de rapports 1/N, 2/N, jusqu'à N/N=1 qui redonnera la courbe de départ.

```
10 REM COURBES D'EQUATION Y=F(X)/VARIANTE
   AFFINITE
20 DEFFNF(X)=
30 LG=    :HT=
100 REM INTERVALLE D'ETUDE
110 REM MODE TEXTE,EFFACAGE D'ECRAN
120 INPUT"BORNES EN X";XI,XS
130 IF XS<=XI THEN 120
140 INPUT"BORNES EN Y";YI,YS
150 IF YS<=YI THEN 140
160 INPUT"NOMBRE DE COURBES";N
170 IF N<=0 OR N<>INT(N) THEN 160
300 REM TRACE
310 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
320 FOR I=0 TO LG
330 X=XI+I*(XS-XI)/LG
340 Y=FNF(X)
350 FOR J=1 TO N
360 D=Y*J/N:REM ORDONNEE MULTIPLIEE PAR LE
    RAPPORT
370 C=HT*(YS-D)/(YS-YI)
380 IF C>=0 AND C<=HT THEN POINT I,C
390 NEXT J
500 NEXT I
510 END
```



```
FNF(X)=COS(X)
XI=-7 : XS=7
YI=-1 : YS=1
N =15
```

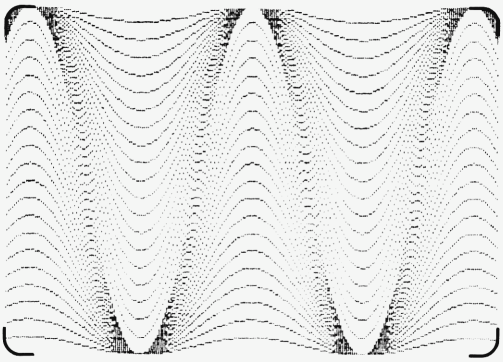


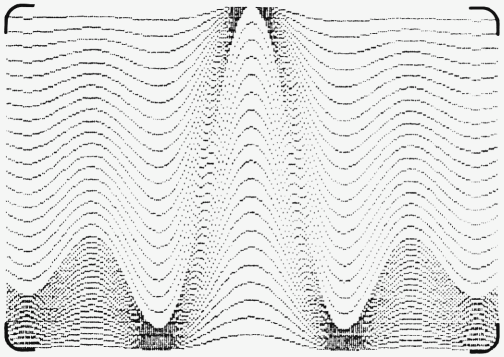
```
FNF(X)=SQR(ABS(X/2+SIN(2*X)))
XI=-6.28 : XS=6.28
YI= Ø    : YS=1.78
N =2Ø
```

En faisant tourner ce programme avec des fonctions de notre petit catalogue, voir certaines zones vides nous donnera peut-être envie de les remplir par des affinités. Pour cela, remplacez les lignes 35Ø à 39Ø par les suivantes :

```
350 C=HT*(YS-Y)/(YS-YI)
360 FOR J=1 TO N
370 D=C*J/N
380 IF D>=Ø AND D<=HT THEN POINT I,D
390 E=HT-(HT-C)*J/N
400 IF E>=Ø AND E<=HT THEN POINT I,E
410 NEXT J
```

```
FNF(X)=COS(X)
XI=-7 : XS=7
YI=-1 : YS=1
N =2Ø
```





```
FNF(X)=SIN(X)/X
XI=-12 : XS=12
YI=-.3 : YS= 1
N =2Ø
```

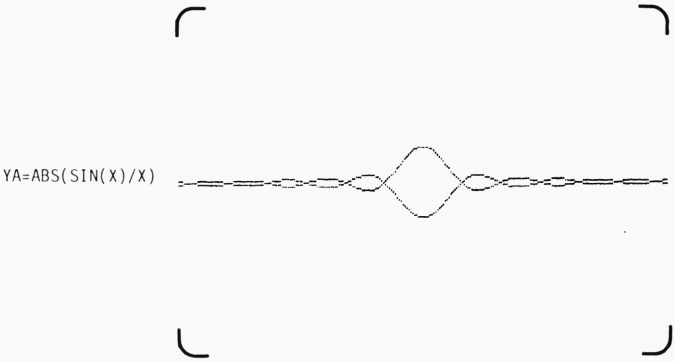
Maintenant, le programme opère deux affinités : l'une par rapport au haut de l'écran (ligne 37Ø), l'autre par rapport au bas de l'écran (ligne 39Ø).

Construction d'un motif

On peut facilement utiliser les fonctions que l'on connaît pour fabriquer différents motifs. Nous allons montrer comment faire tracer un collier de perles à l'écran, en utilisant des fonctions simples :

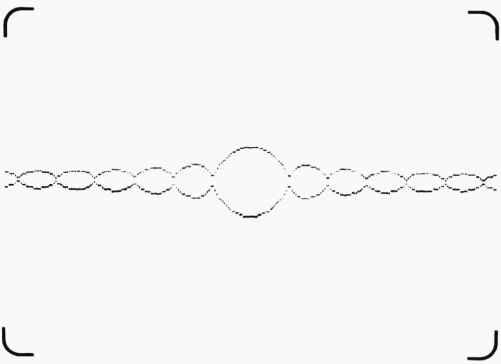
- un collier de perles ondule : on prendra une fonction périodique : SIN(X) ;
- la taille des perles va en décroissant à partir du centre : divisons par X:SIN(X)/X .
- pour représenter le collier, nous avons besoin de deux points pour une abscisse donnée : on prendra +Y et -Y pour avoir une symétrie.

```
10 REM COLLIER
30 LG=    :HT=
100 REM INTERVALLE D'ETUDE
110 REM MODE TEXTE,EFFACAGE D'ECRAN
120 INPUT"BORNES EN X";XI,XS
130 IF XS<=XI THEN 120
140 INPUT"BORNES EN Y";YI,YS
150 IF YS<=YI THEN 140
300 REM TRACE
310 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
320 FOR I=0 TO LG
330 X=XI+I*(XS-XI)/LG
340 YA=ABS(SIN(X)/X)
350 YB=-YA
360 C=HT*(YS-YA)/(YS-YI)
370 D=HT*(YS-YB)/(YS-YI)
380 IF C>=0 AND C<=HT THEN POINT I,C
390 IF D>=0 AND D<=HT THEN POINT I,D
400 NEXT I
410 END
```



Maintenant, en prenant XI=-2Ø, XS=+2Ø, YI=-5, YS=+5, nous pouvons contempler notre oeuvre : le résultat est assez joli, mais la perle centrale est trop grasse par rapport aux autres : on va corriger cela en tapant :

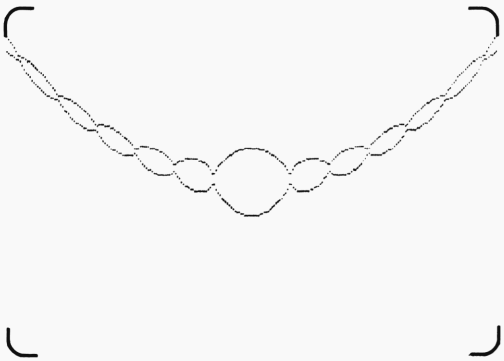
```
340 YA=SQR(ABS(SIN(X)/X))
350 YB=-YA
```



YA=SQR(ABS(SIN(X)/X))

Avec les mêmes valeurs pour XI, XS, YI et YS, le résultat est assez satisfaisant, mais le collier paraît bien plat. Qu'à cela ne tienne, en se souvenant que X*X donne une parabole, rajoutons +.Ø1*X*X à la fin de la ligne 34Ø pour avoir un collier de bonne facture.

YA=SQR(ABS(SIN(X)/X))+.Ø1*X*X



COURBES PARAMETRIQUES

Cas général

Avec les courbes d'équation y=f(x), on ne peut obtenir qu'un point pour une abscisse donnée. Même si les résultats sont parfois intéressants, ils sont rarement très jolis. Si l'on fait dépendre chacune des coordonnées x et y d'une fonction différente, tout va bouger dans tous les sens et souvent donner des résultats fort plaisants. Nous allons poser :

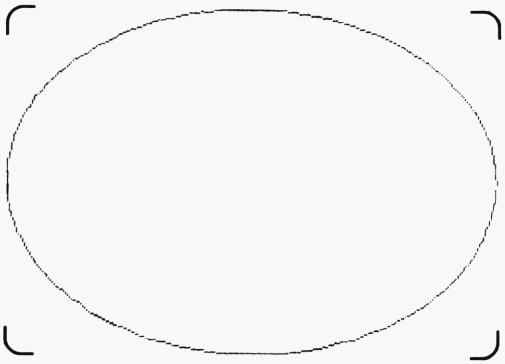
x=f(t)
y=g(t)

t est une variable appelée paramètre, d'où le nom de courbes paramétriques.

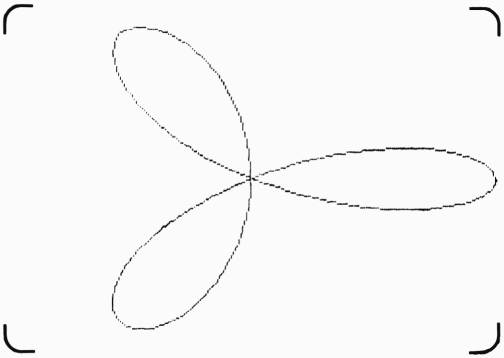
Pour représenter de telles courbes, il nous faudra d'abord choisir la fenêtre d'écran envisagée, que nous rentrerons avec les variables habituelles XI, XS, YI, YS. Mais il faudra également choisir entre quelles valeurs nous examinerons le paramètre et avec quel pas nous le ferons varier.

```
10 REM COURBES EN PARAMETRIQUES
20 DEFFNX(T)=
30 DEFFNY(T)=
40 LG=    :HT=
100 REM BORNES D'ETUDE
110 REM MODE TEXTE,EFFACAGE D'ECRAN
120 INPUT"BORNES EN X";XI,XS
130 IF XS<=XI THEN 120
140 INPUT"BORNES SUR Y";YI,YS
150 IF YS<=YI THEN 140
160 INPUT"BORNES SUR LE PARAMETRE";TI,TS
170 IF TS<=TI THEN 160
180 INPUT"PAS D'ETUDE",P
190 IF P<=0 THEN 180
200 REM TRACE
210 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
220 X=FNX(TI)
230 Y=FNY(TI)
240 A=LG*(X-XI)/(XS-XI)
250 B=HT*(YS-Y)/(YS-YI)
260 FOR T=TI+P TO TS STEP P
270 X=FNX(T)
280 Y=FNY(T)
290 C=LG*(X-XI)/(XS-XI)
300 D=HT*(YS-Y)/(YS-YI)
310 IF A<0 OR A>LG OR B<0 OR B>HT THEN 340
320 IF C<0 OR C>LG OR D<0 OR D>HT THEN 340
330 DROITE A,B  A  C,D
340 A=C
350 B=D
360 NEXT T
370 END
```

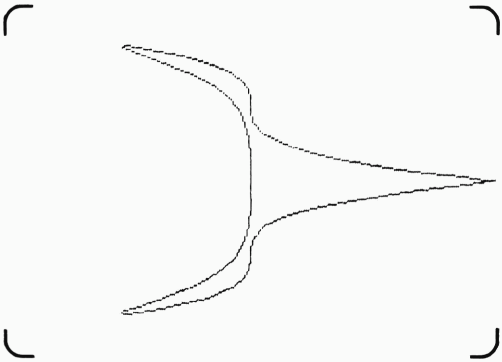
FNX(T)	FNY(T)	XI,XS	YI,YS	TI,TS	P
SIN(T)	COS(T)↑2/(2-COS(T))	-1, 1	0, 1	0, 6.28	.05
COS(T)	SIN(T)	-1, 1	-1, 1	0, 6.28	.05
COS(T)+COS(2*T)	SIN(T)-SIN(2*T)	-2, 2	-2, 2	0, 6.28	.1
COS(T)↑2*(COS(T)+COS(2*T))	SIN(T)↑2*(SIN(T)+SIN(2*T))	-.5, 2	-1.5, 1.5	0, 6.28	.1
3*SIN(T)-SIN(3*T)	3*COS(T)-COS(3*T)	-4, 4	-4, 4	0, 6.28	.05
SIN(9*T)	SIN(10*T)	-1, 1	-1, 1	0, 6.28	.01
1-SIN(T)	1-COS(T)	0, 20	0, 6	0, 21	.1
1-2*SIN(T)	1-2*COS(T)	0, 50	-5, 5	0, 50	.1
COS(T)-T*SIN(T)	SIN(T)+T*COS(T)	-20, 20	-20, 20	-20,20	.1
COS(T)*COS(2*T)	SIN(T)*COS(3*T)	-1, 1	-1, 1	0, 6.28	.05
COS(T)*COS(T/2)	SIN(T)*COS(3*T)	-1, 1	-1, 1	0, 12.7	.05
COS(T)*(COS(3*T))↑3	SIN(T)*SIN(2*T)	-1, 1	-1, 1	0, 3.14	.05
COS(T)+.3*COS(8*T)	SIN(T)	-1.3, 1.3	-1, 1	0, 6.28	.05
COS(T)+.3*COS(8*T)	SIN(T)+.3*SIN(7*T)	-1.3, 1.3	-1.3, 1.3	0, 6.28	.02
COS(T)↑3	SIN(T)↑3	-1, 1	-1, 1	0, 6.28	.05
COS(T)*COS(5*T)	SIN(T)*COS(5*T)	-1, 1	-1, 1	0, 6.28	.02
COS(T)+.5*SIN(7*T)	SIN(T)+.5*COS(3*T)	-1.5, 1.5	-1.5, 1.5	0, 6.28	.02
1-COS(T)	T*SIN(T)	-50, 50	-50, 50	-50,50	.1
COS(T)*COS(2T/3)	SIN(T)*SIN(3*T/2)	-1, 1	-1, 1	0, 18.9	.05
COS(T)/(3-COS(3*T))	SIN(2*T)	-.5, .5	-1, 1	0, 6.28	.05



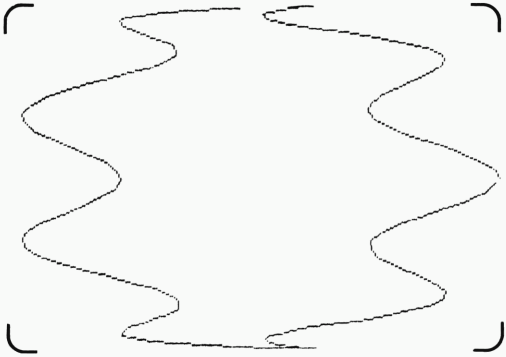
```
FNX(T)=COS(T)
FNY(T)=SIN(T)
XI=-1 : XS=1
YI=-1 : YS=1
TI= Ø : TS=6.3
P= .Ø5
```



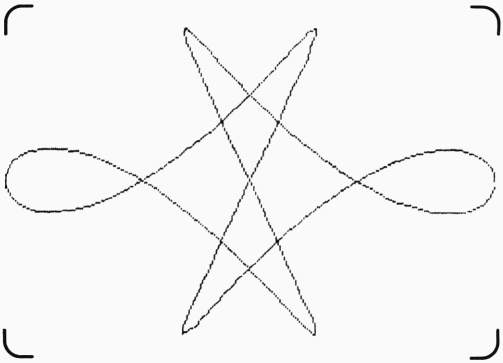
```
FNX(T)=COS(T)+COS(2*T)
FNY(T)=SIN(T)-SIN(2*T)
XI=-2 : XS=2
YI=-2 : YS=2
TI= Ø : TS=6.3
P =.1
```



```
FNX(T)=COS(T)*(COS(3*T))^3
FNY(T)=SIN(T)*SIN(2*T)
XI=-1 : XS=1
YI=-1 : YS=1
TI= Ø : TS=3.15
P =.Ø5
```

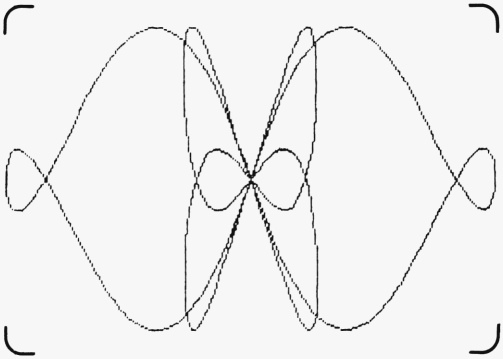


```
FNX(T)=COS(T)+.3*COS(8*T)
FNY(T)=SIN(T)
XI=-1.3: XS=1.3
YI=-1  : YS=1
TI= Ø  : TS=6.29
P =.Ø5
```



```
FNX(T)=COS(T)*COS(2*T)
FNY(T)=SIN(T)*COS(3*T)
XI=-1 : XS=1
YI=-1 : YS=1
TI= Ø : TS=6.3
P =.Ø5
```

```
FNX(T)=COS(T)*COS(T/2)
FNY(T)=SIN(T)*COS(3*T)
XI=-1 : XS=1
YI=-1 : YS=1
TI= Ø : TS=12.7
P =.Ø5
```



Nous avons choisi ici de représenter la courbe par une succession de segments. Ce tracé est plus rapide que le tracé point par point et donne d'excellents résultats si le pas est suffisamment petit. Un segment ne sera bien sûr tracé que si ses deux extrémités sont dans le cadre choisi.

Nous donnons un petit catalogue d'exemples montrant de jolis résultats avec ce programme. Le lecteur notera sans aucun doute l'omniprésence des fonctions COS et SIN. Ces deux fonctions ont leurs valeurs comprises entre -1 et +1, on peut donc prévoir facilement les limites entre lesquelles varient les fonctions construites à partir de COS et SIN. De plus, si les variables X ou Y prennent de grandes valeurs, toutes les ondulations de la courbe se trouvent écrasées, pour des questions d'échelle déjà expliquées (notre programme permet mal l'étude des branches infinies), et il est plus pratique de se limiter à de faibles valeurs.

On pourra être surpris de l'aspect obtenu avec FNX(T)=COS(T), FNY(T)=SIN(T). On devrait obtenir un cercle et l'on n'obtient qu'une ellipse : c'est parce que nous n'avons pas pris la même unité pour les abscisses et les ordonnées (nous travaillons dans un repère non orthonormé). Nous verrons, dans les exemples suivants, comment résoudre cette difficulté.

Courbes de Lissajous

Nous allons maintenant étudier une famille de courbes intéressante : les courbes de Lissajous (utiles dans l'étude des mouvements vibratoires). Elles sont définies paramétriquement par :

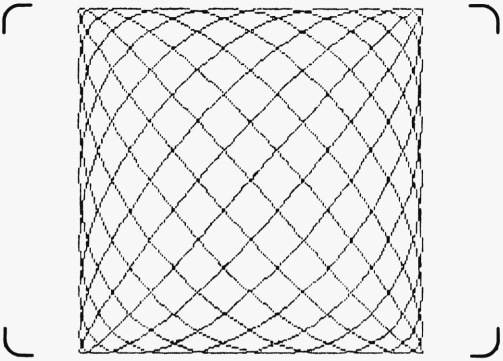
$$x = a \sin m\,t$$
$$y = b \sin n\,t$$

L'abscisse oscille le long d'un segment de longueur 2a, l'ordonnée oscille le long d'un segment de longueur 2b. Nous prendrons, sur notre ordinateur, a=b=1. En écrivant :

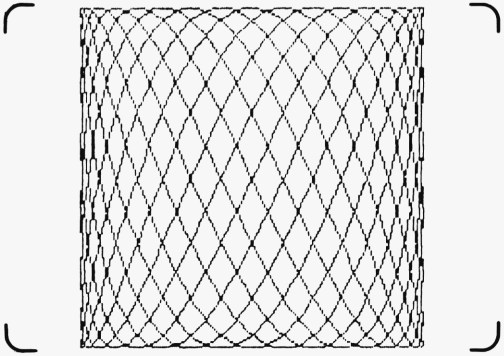
$$x = \sin m\,t$$
$$y = \sin n\,t$$

On voit que x et y sont tous deux compris entre -1 et +1 : la courbe sera comprise à l'intérieur d'un carré, ce que nous ferons apparaître en prenant soigneusement un repère orthonormé.

```
10 REM COURBES DE LISSAJOUS
20 LG=    :HT=
30 XC=    :YC=
40 Z=INT(HT/2)
100 REM BORNES D'ETUDE
110 REM MODE TEXTE,EFFACAGE D'ECRAN
120 INPUT"VALEURS DE M ET N";M,N
130 INPUT"BORNE SUR LE PARAMETRE";TS
140 IF TS<=0 THEN 130
150 INPUT"PAS SUR LE PARAMETRE";P
160 IF P<=0 THEN 150
200 REM TRACE
210 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
220 A=XC
230 B=YC
240 FOR T=P TO TS STEP P
250 C=XC+Z*SIN(M*T)
260 D=YC-Z*SIN(N*T)
270 DROITE A,B  A C,D
280 A=C
290 B=D
300 NEXT T
310 A$=INKEY$:REM OU GET A$
320 IF A$="" THEN 310
330 RUN
```



M =9   : N= 10
TS=6.29 : P=.01



M =8   : N=17
TS=6.29 : P=5E-03

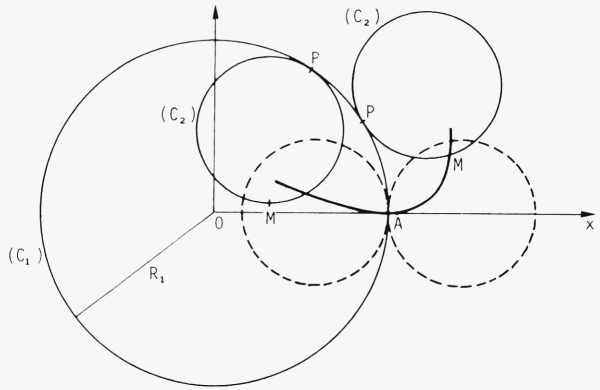
Comme abscisse et ordonnée varient entre -1 et +1, nous prendrons l'origine au centre de l'écran, ce qui est fait dans les variables XC et YC. On calcule ensuite dans Z la plus grande longueur disponible à partir du centre : c'est la moitié de la hauteur puisque l'écran est toujours plus large que haut. Ces calculs étant faits, le programme demande des valeurs pour M et N, la borne supérieure TS du paramètre (nous sommes convenus de partir de T=Ø) et le pas d'étude. Après le tracé, on contemple le résultat jusqu'à ce qu'une touche soit pressée, et l'on recommence.

D'après les calculs des lignes 25Ø et 26Ø, le repère a l'air bien orthonormé, mais c'est compter sans la distorsion du téléviseur ou du moniteur dont nous vous avons déjà parlé dans le chapitre sur les rotations : il faut adapter la ligne 25Ø si vous n'êtes pas tout à fait content du résultat, avec le coefficient KE déjà calculé, en tapant :

```
50 KE=
250 C=XC+Z#KE#SIN(M#T)
```

Après cette modification, vous pourrez pleinement profiter de ce programme en essayant quelques valeurs. Démarrez par exemple avec M=1 et N=2,3,4, etc. Vous vous rendrez compte que plus M et N sont grands, plus le pas P doit être petit. On obtient de jolies courbes fermées avec M=9, N=10, ou M=15, N=16 (et TS=6.28), mais parfois la courbe ne se ferme pas (M=1.01, N=2.13...) et on doit prendre une grande valeur pour TS. Avec beaucoup de patience, vous pourrez alors voir le carré se remplir entièrement.

Epicycloïdes et hypocycloïdes



Quand un cercle (C_2) roule sans glisser sur un cercle fixe (C_1), chacun de ses points décrit une courbe appelée épicycloïde, si (C_2) est extérieur à (C_1), et hypocycloïde, si (C_2) est intérieur à (C_1).

Au départ, le cercle (C_2) a la position tracée en pointillé. Dire que (C_2) roule sans glisser, c'est dire que, pour tout point M de la courbe envisagée, l'arc $\overset{\frown}{AP}$ du cercle fixe est égal à l'arc $\overset{\frown}{PM}$ du cercle mobile.

On montre qu'un point M d'une épicycloïde a pour coordonnées :

$$\begin{cases} x = (R_1 + R_2)\cos t - R_2 \cos\left[\left(1 + \frac{R_1}{R_2}\right) t\right] \\ y = (R_1 + R_2)\sin t - R_2 \sin\left[\left(1 + \frac{R_1}{R_2}\right) t\right] \end{cases}$$

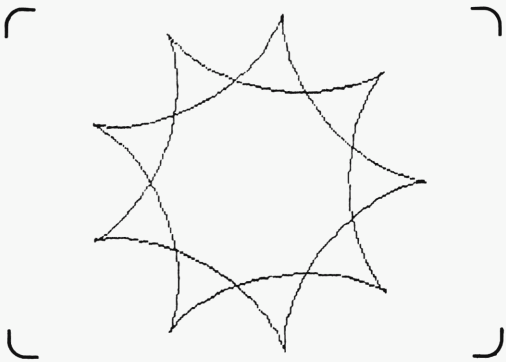
Pour une hypocycloïde, par contre :

$$\begin{cases} x = (R_1 - R_2)\cos t + R_2 \cos\left[\left(1 - \frac{R_1}{R_2}\right) t\right] \\ y = (R_1 - R_2)\sin t + R_2 \sin\left[\left(1 - \frac{R_1}{R_2}\right) t\right] \end{cases}$$

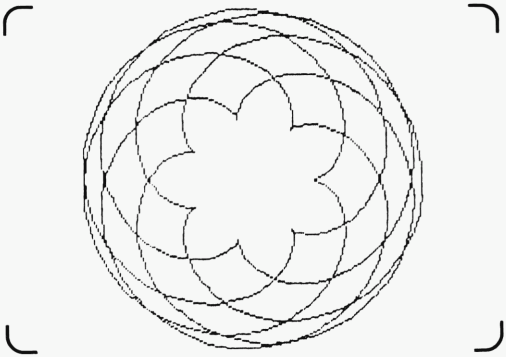
On voit, sur ces formules, que le seul changement concerne le signe de R_2.

Le programme tient compte de cela : il demande les rayons positifs, le type de courbe (épicycloïde : 1 ou hypocycloïde : 2) ; dans le cas d'une hypocycloïde, R_2 est changé en son opposé :

```
50 LG=    :R1=
60 XC=    :YC=
70 Z=INT(R1/2)
100 REM MODE TEXTE:CLEAR:HOME:CLS
110 INPUT"RAYON DU CERCLE FIXE";R1
120 INPUT"RAYON DU CERCLE ROULE";R2
130 INPUT"EPICYCLOIDE =1 OU HYPO =2";V
140 IF V<>1 AND V<>2 THEN 130
150 IF V=2 THEN R2=-R2
160 M=1:R1=R2
170 RM=R1+2*R2
180 IF V=2 THEN RM=ABS(R2*(R2<R1))
200 INPUT"BORNE SUP DU PARAMETRE";TS
210 IF TS<=0 THEN 200
220 INPUT"PAS D'ETUDE";P
230 IF P<=0 THEN 220
300 REM MODE GRAPHIQUE:CLEAR:CLS
310 A=R2:A=-(A*Z-RM*X
320 B=YC
330 FOR T=P TO TS STEP P
340 C=XC+R2*Z*RM*COS(T)-COS(M*T)
350 D=YC-R2*Z*RM*SIN(T)-SIN(M*T)
360 DRAW"BM A,B M C,D
370 A=C
380 B=D
390 NEXT T
400 END
```



HYPOCYCLOIDE
R1= 9 : R2= 2
TS=13 : P =.1



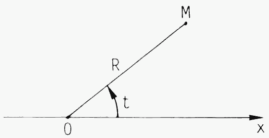
EPICYCLOIDE
R1= 7 : R2= 6
TS=38 : P =.1

Il vaut mieux prendre TS très grand pour ne pas obtenir une courbe tronquée. Commencez par explorer les épicycloïdes et les hypocycloïdes avec des rayons entiers et, ensuite, lancez-vous dans l'irrationnel.

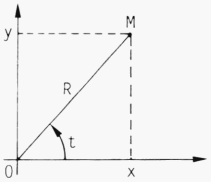
COORDONNEES POLAIRES

Cas général

Nous avons utilisé jusqu'ici les coordonnées cartésiennes pour repérer un point dans le plan. Il existe une autre méthode :



On peut repérer un point M par sa distance R au point O (le pôle) et par l'angle t formé par les droites Ox et OM. Il est facile de passer des coordonnées polaires aux coordonnées cartésiennes :

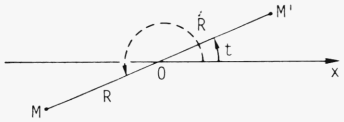


$x = R \cos t$

$y = R \sin t$

Si la longueur R reste constante, on obtient la représentation paramétrique d'un cercle, comme nous l'avons déjà noté. Mais si cette longueur varie en fonction de l'angle t, on aura d'autres courbes dont certaines seront très, très jolies (vous pourrez le constater avec notre catalogue). Il y a un point qui pourra inquiéter les lecteurs peu habitués à cette représentation. Les fonctions R(t) que nous donnerons peuvent prendre des valeurs

négatives et R doit représenter une longueur, qui est quelque chose de bien positif. Rassurez-vous, la courbe n'a pas disparu dans un espace-temps parallèle, comme on peut le constater avec cette figure :



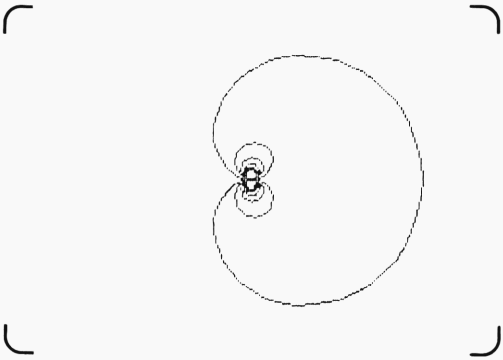
Au lieu d'obtenir le point M' de coordonnées (R,t), on obtient son symétrique par rapport au point O qui a pour coordonnées (R,t+¶) : on ajoute 180° à l'angle t, soit ¶ radians.

Après ce préambule mathématique, nous allons donner le programme permettant de construire les courbes en polaires :

```
10 REM COURBES EN POLAIRES
20 DEFFNR(T)=    :REM EQUATION DE LA COURBE
30 LG=    :HT=
40 XC=    :YC=
50 Z=HT/2
100 REM BORNES D'ETUDE
110 REM MODE TEXTE,EFFACAGE D'ECRAN
120 INPUT"RAYON MAXIMUM",RM
130 IF RM<=0 THEN 120
140 INPUT"BORNES SUR LE PARAMETRE",TI,TS
150 IF TS<=TI THEN 140
160 INPUT"PAS D'ETUDE",P
170 IF P<=0 THEN 160
200 REM TRACE
210 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
220 R=FNR(TI)
230 A=XC+R*Z/RM*COS(TI)
240 B=YC-R*Z*SIN(TI)
250 FOR T=TI TO TS STEP P
260 R=FNR(T)
270 C=XC+R*Z/RM*COS(T)
280 D=YC-R*Z/RM*SIN(T)
290 IF A<0 OR A>LG OR B<0 OR B>HT THEN 320
300 IF C<0 OR C>LG OR D<0 OR D>HT THEN 320
310 DROITE A,B  A C,D
320 A=C
330 B=D
340 NEXT T
350 END
```

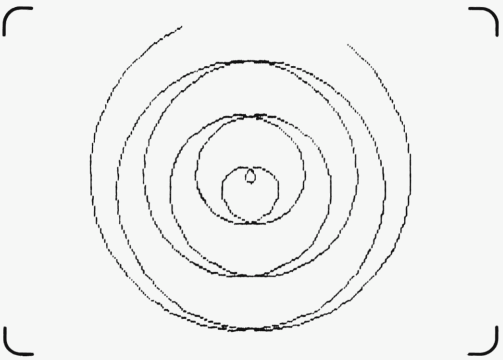
Le pôle est placé au milieu de l'écran (XC,YC). Z représente la longueur maximum disponible. Le programme demande le rayon maximum RM que prendra la courbe, pour assurer une belle mise en page. Le reste du programme est quasiment identique à celui donné pour les représentations paramétriques.

FNR(T)	RM	TI,TS	P
COS(T)	1	0, 3.14	.1
COS(T)*COS(2*T)	1	0, 3.14	.05
COS(T)+COS(2*T)	2	0, 6.28	.05
COS(9*T/1Ø)	1	0, 62.9	.05
COS(4*T)	1	0, 6.28	.05
COS(T/1Ø)+.2	1.2	0, 62.9	.05
SIN(T)/T	1	-20, 20	.1
Ø.1*T	2	-20, 20	.2
1-2*SIN(2*T)	3	0, 6.29	.1
SQR(1+SIN(2*T))+SQR(1-SIN(2*T))	2	0, 6.29	.1
COS(2*T/3)	1	0, 19	.1
1+COS(2Ø*T)*.5	1.5	0, 6.29	.01
COS(6*T/11)+2	3	0, 70	.1
2Ø*COS(T)-COS(2Ø*T)	21	0, 6.29	.02
SIN(3*T)	1	0, 6.29	.05



FNR(T)=SIN(T)/T
RM= 1
TI=-2Ø : TS=2Ø
P = .1

```
FNR(T)=Ø.1*T
RM=  2
TI=-2Ø : TS=2Ø
P = .2
```



Les pas que nous vous avons donnés dans le catalogue ne sont là qu'à titre indicatif, vous pouvez les modifier suivant la résolution de votre système et votre patience. Pour une résolution de 256 x 192 (qu'on retrouve sur beaucoup d'appareils), ils nous ont donné des résultats très acceptables. Les perfectionnistes qui trouvent que leur écran n'est pas assez orthonormé, pourront bien sûr multiplier les longueurs par le fameux coefficient KE aux lignes 23Ø et 27Ø.

Variantes

Les fonctions du genre R=SIN(L*T) sont toujours assez esthétiques : on obtient des fleurs aux nombreux pétales si L est un nombre entier, des volutes intéressantes pour L=.2, L=1/3, L=1.5 ... On peut avoir envie de faire un programme qui démarre par INPUT L pour explorer tranquillement cette famille de courbes. C'est ce que nous avons fait en rajoutant une option supplémentaire : au lieu de multiplier le rayon R par COS(T) et SIN(T), pourquoi ne pas le multiplier par COS(M*T) et SIN(N*T), où M et N sont deux nombres à choisir à chaque fois. Voilà les réflexions qui ont conduit au programme suivant :

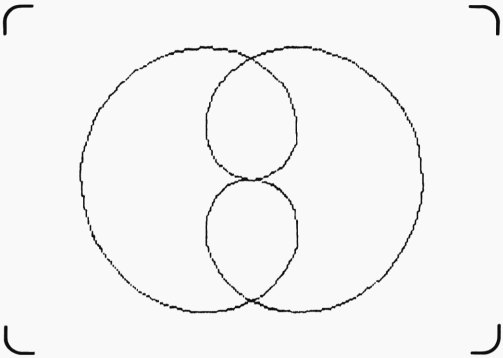
```
10 REM COURBES EN POLAIRES DU TYPE R=SIN(L*T)
30 LG=    :HT=
40 XC=    :YC=
50 Z=HT/2
100 REM BORNES D'ETUDE
110 REM MODE TEXTE,EFFACAGE D'ECRAN
120 INPUT"COEFFICIENTS L,M,N";L,M,N
140 INPUT"BORNES SUR LE PARAMETRE";TI,TS
150 IF TS<=TI THEN 140
160 INPUT"PAS D'ETUDE";P
170 IF P<=0 THEN 160
200 REM TRACE
210 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
230 A=XC
240 B=YC
250 FOR T=TI TO TS STEP P
260 R=SIN(L*T)
270 C=XC+R*Z*COS(M*T)
280 D=YC-R*Z*SIN(N*T)
290 IF A<0 OR A>LG OR B<0 OR B>HT THEN 320
300 IF C<0 OR C>LG OR D<0 OR D>HT THEN 320
310 DROITE A,B  A C,D                    →
```

```
320 A=C
330 B=D
340 NEXT T
350 A$=INKEY$:REM OU GET A$
360 IF A$="" THEN 350
370 RUN
```

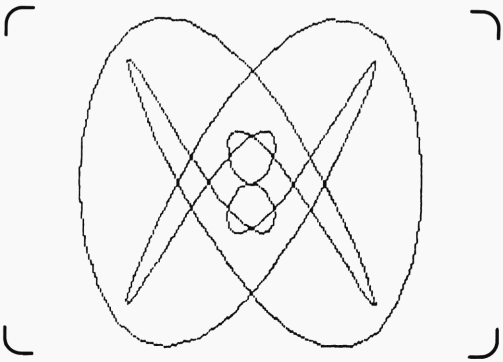
Maintenant, allez-y ! Les résultats sont souvent très esthétiques. Vous pourrez commencer avec des valeurs entières de L, M, N :

L	M	N
1	1	4
1	1	8
1	2	2
1	4	6
3	1	6

auquel cas TS=6,28 conviendra.



L =1 : M=2 : N=2
TS=6.3 : P=.Ø5



L =1 : M=4 : N=6
TS=6.3 : P=.Ø5

Vous pourrez poursuivre avec des fractions, puis modifier la ligne 26Ø pour explorer une autre famille de courbes. Ce genre de programme est très dangereux : on tape quelques lignes, on introduit quelques nombres et... on regarde, on regarde, on regarde sans faire grand-chose d'autre.

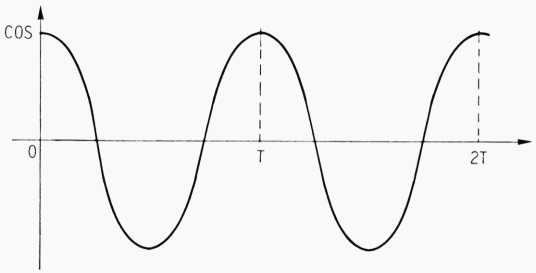
Chapitre II
Approximations et fonctions

Nous allons, sur deux exemples, faire des mathématiques illustrées. Que les non-matheux ne s'effraient pas, il est très facile de comprendre ce qui se passe sur l'écran, et les méthodes de calcul décrites peuvent être utilisées dans bien d'autres cas.

SERIES DE FOURIER

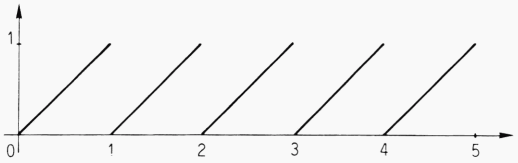
Développement en séries de Fourier

Les lecteurs ayant représenté sur leur écran les fonctions COS et SIN savent ce qu'est une fonction périodique :



La représentation graphique répète le même motif à intervalles réguliers. La longueur de cet intervalle est appelée période. Pour COS et SIN, la période est de 2¶ ; pour TAN, c'est ¶.

On peut donner bien d'autres exemples de fonctions périodiques. Essayez de représenter la fonction X-INT(X).



La période est ici de 1. Voilà une fonction à l'aspect bien différent des ondulations d'une sinusoïde. Pourtant, on a montré que cette fonction, comme la plupart des fonctions périodiques, pouvait s'exprimer uniquement à l'aide de COS et SIN. Votre ordinateur disposant de ces deux fonctions dans sa bibliothèque, aucune fonction périodique ne vous est inaccessible.

Tout d'abord, on peut tracer des sinusoïdes ayant une période différente de 2π . Par exemple, sin 2x a une période de π, sin 3x a une période de $\frac{2\pi}{3}$. Pour avoir une période donnée T, il suffit de considérer sin $\frac{2\pi}{T}x$. On écrit en général : sin ωx, ω étant appelé la pulsation de la fonction, avec $\omega=\frac{2\pi}{T}$ (ou alors $\omega=2\pi F$, où F est la fréquence).

La formule donnant une fonction périodique de période T avec des cosinus et des sinus est :

$$f(x) = a_0+a_1 \cos \omega x + b_1 \sin \omega x + a_2 \cos 2\omega x + b_2 \sin 2\omega x + \ldots + a_n \cos n\omega x + b_n \sin n\omega x + \ldots \quad (\text{avec } \omega=\frac{2\pi}{T})$$

On dit qu'on a écrit la fonction sous forme de série trigonométrique (ou série de Fourier). Les termes d'indice 1 : $a_1 \cos \omega x + b_1 \sin \omega x$ donnent une fonction de période T, de fréquence $F=\frac{1}{T}$; les termes d'indice 2 : $a_2 \cos 2\omega x + b_2 \sin 2\omega x$ donnent une fonction de période $\frac{T}{2}$, de fréquence 2F, etc. La fréquence étant multipliée, les termes d'indices successifs sont appelés des harmoniques (si votre micro préféré est musicien, vous pourrez vous rendre compte que multiplier la fréquence d'une note par deux la fait monter d'une octave, d'où le terme harmonique).

Avant de passer à la programmation, il nous reste à voir comment calculer les coefficients a_0, a_1, b_1, etc. On les obtient par les formules :

$$a_0 = \frac{1}{T}\int_A^{A+T} f(x)dx$$

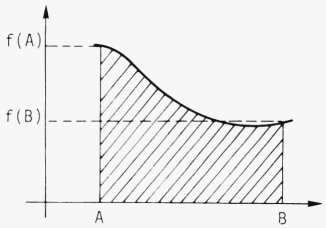
$$a_n = \frac{2}{T}\int_A^{A+T} f(x) \cos n\omega x \, dx$$

$$b_n = \frac{2}{T}\int_A^{A+T} f(x) \sin n\omega x \, dx$$

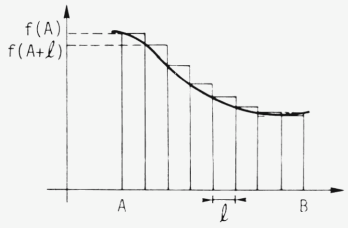
Rappels sur l'intégration

Rappelons d'abord ce que signifie le signe $\int$ (intégrale) :

$\int_A^B f(t)dt$ représente la surface hachurée :



Nous allons calculer une surface voisine de celle qui est hachurée :



On découpe l'intervalle [A,B] en petits intervalles de largeur ℓ et on trace des rectangles de hauteurs f(A), f(A+ℓ), f(A+2ℓ), etc. On calcule l'aire de ces rectangles : f(A)xℓ, f(A+ℓ)xℓ, etc.

On comprend que si la longueur ℓ est suffisamment petite, la somme des aires de ces rectangles sera voisine de l'intégrale à calculer. Il y a bien sûr une erreur et il existe des méthodes de calcul plus performantes, mais on pourra se rendre compte que la précision est suffisante pour cette application.

Programmation

Les méthodes étant exposées, nous pouvons commencer à rédiger notre programme : il comprendra bien sûr la représentation graphique de la fonction à étudier, un module de calcul et la représentation de l'approximation, à chaque étape. On pourra voir sur l'écran comment une somme de COS et de SIN approche de plus en plus une fonction où ces deux termes n'interviennent pas.

Nous avons choisi de représenter la fonction sur deux périodes (autant bien montrer que l'on a affaire à une fonction périodique). La longueur L2 (ligne 4Ø) représente la moitié de l'écran. L'utilisateur doit donner l'intervalle d'étude [A,B]. La fonction sera automatiquement répétée sur les intervalles suivants. Il n'est pas nécessaire d'étudier des fonctions périodiques au départ, le programme rend n'importe quoi périodique. On demande ensuite le nombre de termes Z que vous désirez voir calculer. Ce nombre est une fonction complexe dont la variable principale est votre patience : le programme va d'abord calculer le terme, afficher le résultat, et ainsi de suite jusqu'à Z. Dans la pratique, vous pouvez prendre Z assez grand (2Ø ou 3Ø par exemple) et arrêter le programme dès que le résultat vous semble satisfaisant.

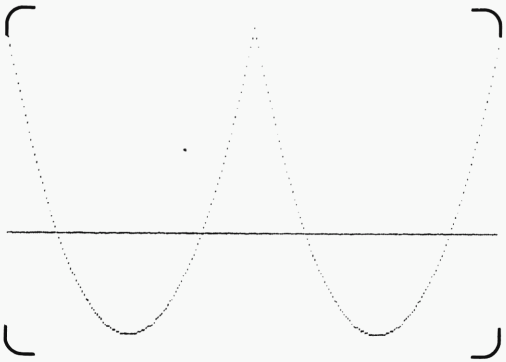
Le programme ci-après calcule, aux lignes 9Ø à 14Ø, les variables nécessaires. On trouve, à la ligne 12Ø, l'initialisation de trois tableaux : les tableaux A et B vont contenir les coefficients de la série trigonométrique, le tableau T contiendra les ordonnées des points de la courbe représentant la fonction : comme celle-ci sera répétée souvent, autant ne faire les calculs qu'une seule fois. Les variables DY et C1 servent à la mise en page : nous calculons des approximations qui vont sans doute dépasser le maximum ou le minimum de la fonction. Celle-ci sera donc représentée sur les 7/8 de la hauteur de l'écran, en laissant 1/16 de part et d'autre.

On trouve, à partir de la ligne 1ØØØ, le calcul de la première intégrale : on calcule d'abord la valeur de la fonction en un point (H), puis on multiplie par la largeur L du pas d'étude avant d'ajouter à la somme en cours. Pour ce premier tracé, sachant qu'il s'agit d'une droite horizontale, point n'est besoin de la tracer point par point. On rentre ensuite dans la boucle principale où les calculs se déroulent comme ci-dessus.

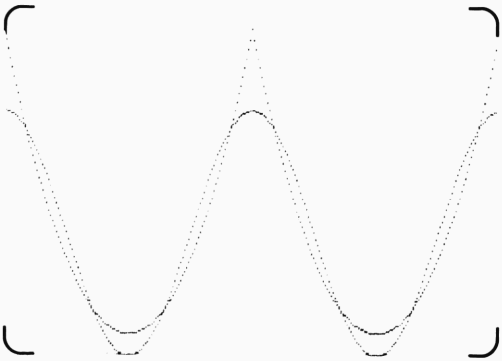
En faisant tourner ce programme, vous trouverez certainement des améliorations à lui apporter : de la couleur pour bien distinguer les courbes, des alternances de pages graphiques, etc. L'utilisateur matheux aura sans doute la curiosité de regarder les valeurs numériques des coefficients A(0), A(1) ..., et de les comparer avec leurs valeurs théoriques. La précision n'est pas le fait de ce programme (cela peut être amélioré avec des méthodes plus sophistiquées), mais les résultats graphiques sont généralement très satisfaisants : celui qui connaît la notion de convergence uniforme ne peut que s'indigner d'avoir été torturé avec des epsilon et autres quantités, alors que, sur l'écran, tout s'éclaire.

```
10 REM APPROXIMATION PAR SERIES DE FOURIER
20 DEFFNF(X)=X*X
30 PI=4*ATN(1)
40 LG=    HT=
50 L2=INT((LG+1)/2)-1:REM MOITIE DE LA
   HAUTEUR DE L'ECRAN
60 REM MODE TEXTE,EFFACAGE D'ECRAN
70 INPUT"BORNES DE L'INTERVALLE";A,B
80 IF B<=A THEN 70
90 INPUT"COMBIEN DE TERMES";Z
100 T=B-A REM PERIODE
110 PU=2*PI/T:REM PULSATION
120 L=T/(L2+1) REM PAS D'ETUDE
130 DIM T(L2),A(Z),B(Z)
140 DY=7*(HT+1)/8
150 C1=HT-(HT+1)/16:REM MISE EN PAGE
200 REM CALCUL MINIMAX
210 MAX=-1E30:MIN=1E30
220 FOR I=0 TO L2
230 X=A+L*I:H=FNF(X)
240 IF H>MAX THEN MAX=H
250 IF H<MIN THEN MIN=H
260 NEXT I
300 REM PREMIER TRACE
310 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
320 FOR I=0 TO L2
330 X=A+L*I:H=FNF(X)
340 C=DY*(H-MIN)/(MAX-MIN):C=C1-INT(C)
350 T(I)=C
360 POINT I,C
370 NEXT I
380 REM DEUXIEME MOITIE
390 FOR I=0 TO L2
400 POINT L2+1+I,T(I)
410 NEXT I
1000 REM CALCUL A(0)
1010 FOR I=0 TO L2
1020 X=A+L*I:H=FNF(X)
1030 A(0)=A(0)+H*L
1040 NEXT I
1050 A(0)=A(0)/T
1100 REM TRACE DROITE
1110 C=DY*(A(0)-MIN)/(MAX-MIN)
1120 C=C1-INT(C)
1130 DROITE 0,C A LG,C
1900 N=1
2000 REM BOUCLE
2500 REM CALCUL DES INTEGRALES
2510 FOR I=0 TO L2
2520 X=A+L*I:H=FNF(X)                    →
```

```
2530 A(N)=L*H*COS(N*PU*X)+A(N)
2540 B(N)=L*H*SIN(N*PU*X)+B(N)
2550 NEXT I
2560 A(N)=A(N)*2/T:B(N)=B(N)*2/T
3000 REM TRACE DE LA FONCTION
3010 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
3020 FOR I=0 TO L2
3030 POINT I,T(I):POINT L2+1+I,T(I)
3040 NEXT I
3500 REM TRACE APPROXIMATION
3510 FOR I=0 TO L2
3520 X=A+L*I
3530 S=A(0)
3540 REM CALCUL DES SOMMES DE COS ET SIN
3550 FOR J=1 TO N
3560 S=A(J)*COS(J*PU*X)+B(J)*SIN(J*PU*X)+S
3570 NEXT J
3580 REM TRACE
3590 C=(S-MIN)*DY/(MAX-MIN):C=C1-INT(C)
3600 IF C<0 OR C>HT THEN 3620
3610 POINT I,C:POINT L2+1+I,C
3620 NEXT I
3630 N=N+1:IF N<=Z THEN 2000
3640 END
```

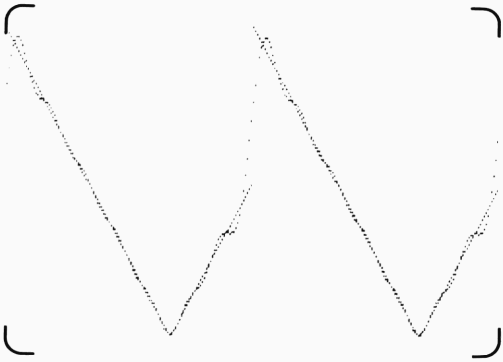


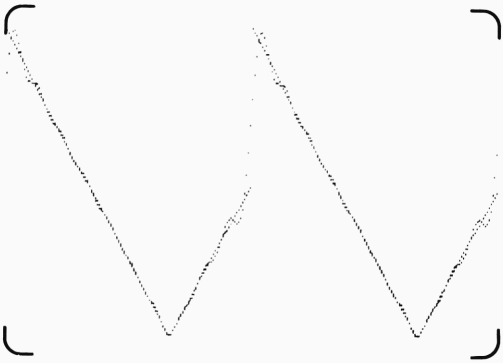
```
FNF(X)=X*X
INTERVALLE : [-2, 2]
ETAPE 0
```



FNF(X)=X*X
INTERVALLE : [-2, 2]
ETAPE 1

FNF(X)=ABS(X)
INTERVALLE : [-2, 1]
ETAPE 7





FNF(X)=ABS(X)
INTERVALLE : [-2, 1]
ETAPE 1Ø

APPROXIMATION PAR DES POLYNÔMES

Principe

Une fonction polynôme est une fonction du type :

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

On voit donc qu'elle s'exprime uniquement avec des multiplications et des additions, ce qui la rend facile à calculer. Il existe un théorème disant que toute fonction continue peut être approchée par des polynômes. Le mot continu ne doit pas effrayer. Une fonction continue, c'est une fonction dont on peut tracer la courbe sans lever le crayon du papier. Parmi les fonctions dont dispose votre petit système individuel, COS, SIN, ATN, EXP par exemple, sont continues, tandis que INT ne l'est pas.

La méthode que nous allons employer est simple à comprendre, mais nécessitera quelques petits calculs. Vous avez pu vous rendre compte, lors des tracés de fonctions, qu'un polynôme de degré 1 ($f(x)=a_1x+a_0$) donnait une droite. On sait bien qu'il suffit de deux points pour déterminer une droite. Un polynôme de degré 2 ($f(x)=a_2x^2+a_1x+a_0$) donne une courbe appelée parabole, et il faut trois points pour déterminer une parabole. De même, pour connaître un polynôme de degré 3, il nous faudra quatre points, etc.

Le procédé sera donc le suivant : nous allons d'abord tracer la courbe de la fonction choisie, choisir deux points de cette courbe, puis tracer la droite passant par ces deux points, recommencer avec trois points pour tracer la parabole passant par ces trois points, et ainsi de suite. On comprend que plus nous prendrons de points, plus nous approcherons de la fonction de départ.

Le programme que nous allons donner pourra intéresser les physiciens qui se posent souvent le problème suivant : après avoir fait une série de mesures et reporté tout cela sur un graphique, comment trouver la fonction passant par tous ces points ?

Méthode du pivot

Montrons maintenant comment nous allons calculer les polynômes. Pour trouver une droite passant par les deux points de coordonnées (x_1,y_1) et (x_2,y_2), nous allons poser :

$$\begin{cases} a_1 x_1 + a_0 = y_1 \\ a_1 x_2 + a_0 = y_2 \end{cases}$$

Nos inconnues sont ici a_1 et a_0.

Pour trouver un polynôme de degré 2, nous aurons besoin de trois points :

$$a_2 x_1^2 + a_1 x_1 + a_0 = y_1$$

$$a_2 x_2^2 + a_1 x_2 + a_0 = y_2$$

$$a_2 x_3^2 + a_1 x_3 + a_0 = y_3$$

Les inconnues sont a_2, a_1 et a_0.

On reconnaît un problème classique : résoudre un système de n équations à n inconnues. Avant de décrire la méthode de résolution, nous allons prendre les notations employées dans le programme. Nos inconnues sont les coefficients du polynôme cherché que nous noterons A(N), A(N-1) ...,A(Ø).

Nous écrirons le système sous la forme suivante :

```
A(Ø)*X(Ø,Ø)+A(1)*X(Ø,1)+...+A(N)*X(Ø,N)=X(Ø,N+1)
A(Ø)*X(1,Ø)+A(1)*X(1,1)+...+A(N)*X(1,N)=X(1,N+1)
  .
  .
  .
A(Ø)*X(N,Ø)+A(1)*X(1,1)+...+A(N)*X(N,N)=X(N,N+1)
```

Nous mettons tous les coefficients du système dans un même tableau X, dimensionné à (N,N+1). Par rapport aux notations précédentes, nous avons : $X(Ø,2)=x_1$, $X(1,N+1)=y_2$. Nous verrons tout à l'heure comment remplir le tableau des X(I,J) dans notre cas particulier.

Pour résoudre le système, il faut savoir qu'on a le droit de faire deux opérations : multiplier ou diviser tous les termes d'une ligne par un même nombre, et ajouter ou soustraire, terme à terme, les lignes entre elles. Nous allons employer cela pour essayer d'avoir le maximum de Ø dans notre tableau. Nos calculs nous feront parvenir à la situation suivante :

```
A(Ø)*X(Ø,Ø)+A(1)*X(Ø,1)+A(2)*X(Ø,2)+...+A(N-1)*X(Ø,N-1)+A(N)*X(Ø,N)=X(Ø,N+1)
    Ø      +A(1)*X(1,1)+A(2)*X(1,2)+...+A(N-1)*X(1,N-1)+A(N)*X(1,N)=X(1,N+1)
    Ø      +   Ø      +A(2)*X(2,2)+...+A(N-1)*X(2,N-1)+A(N)*X(2,N)=X(2,N+1)
    .
    .
    .
    Ø      +   Ø      +   Ø      +...+A(N-1)*X(N-1,N-1)+A(N)*X(N-1,N)
                                      =X(N-1,N+1)
    Ø      +   Ø      +   Ø      +...       Ø         +A(N)*X(N,N)=X(N,N+1)
```

La dernière ligne nous permettra alors de calculer A(N) par une simple division, puis nous pourrons calculer A(N-1) en reportant la valeur de A(N) dans la ligne précédente et, en remontant, nous obtiendrons tous les termes jusqu'à A(Ø). Voyons comment cela s'organise dans la pratique :

```
50000 REM RESOLUTION D'UN SYSTEME PAR LA
      METHODE DU PIVOT
50100 FOR K=Ø TO N
50110 P=X(K,K):IF P=Ø THEN GOSUB 55000
50120 FOR J=K TO N+1
50130 X(K,J)=X(K,J)/P
50140 NEXT J
50150 IF K=N THEN 50500
50200 FOR I=K+1 TO N
50210 A=X(I,K)
50220 FOR J=K TO N+1
50230 X(I,J)=X(I,J)-A*X(K,J)
50240 NEXT J
50250 NEXT I
50260 NEXT K
50500 REM SOLUTIONS
50510 A(N)=X(N,N+1)
50520 FOR K=N-1 TO Ø STEP -1
50530 A=X(K,N+1)
50540 FOR J=K+1 TO N
50550 A=A-A(J)*X(K,J)
50560 NEXT J
50570 A(K)=A
50580 NEXT K
50590 RETURN
55000 REM TRUC
55010 FOR I=K+1 TO N
55020 IF X(K,I)<>Ø THEN 55500
55030 NEXT I                                    →
```

```
55040 PRINT"ERREUR":END
55500 FOR J=K TO N+1
55510 X(K,J)=X(K,J)+X(I,J)
55520 NEXT J
55530 RETURN
```

Nous avons présenté la résolution de systèmes comme un sous-programme que vous pourrez sans peine insérer dans vos réalisations. Expliquons le détail du calcul :

- on prend successivement tous les termes de la diagonale du tableau (ligne 5Ø11Ø). Ce terme est appelé pivot (la méthode que nous décrivons est dite méthode du pivot). Nous verrons plus loin ce qu'il faut faire quand ce terme est nul ;
- on divise les termes de la ligne par le pivot : le pivot devient égal à 1 ;
- pour annuler les termes situés en dessous du pivot : on les met dans la variable A (ligne 5Ø21Ø), on multiplie la ligne du pivot par A, puis on fait la soustraction des lignes terme à terme (ligne 5Ø23Ø) ;
- si le pivot est nul, il faut remédier à cette situation par une addition de lignes : on cherche dans sa colonne un terme non nul. Si l'on en trouve un, on fait l'addition de lignes et on continue la résolution. Si malheureusement il n'y en a pas, cela signifie que le système n'admet pas de solutions, le programme doit donc s'arrêter. Rassurez-vous, c'est une situation qui ne se produira jamais dans le programme que nous présentons, le tableau X(I,J) est rempli de façon à ce que le système admette une solution (et une seule !).

Algorithme de Hörner

Maintenant, presque toutes les difficultés sont résolues, mais nous avons encore un point à examiner avant de donner le programme complet. Nous venons de voir comment calculer les coefficients d'un polynôme, il nous reste à savoir calculer les valeurs que prend ce polynôme.

Considérons par exemple :

```
A=A(2)*X↑2+A(1)*X+A(Ø)
```

On pourrait calculer ce terme avec le petit programme suivant :

```
10 A=Ø
20 FOR I=Ø TO N
30 A=A(I)*(X↑I)+A
40 NEXT I
```

On a donc, dans ce cas, à chaque passage, une élévation à la puissance I, une multiplication et une addition. On peut supprimer l'élévation à la puissance (opération coûteuse en temps et manquant parfois de précision dans nos machines) en prenant le schéma de calcul suivant, appelé algorithme de Hörner :

```
A=A(2)
A=A(2)*X+A(1)
A=[A(2)*X+A(1)]*X+A(Ø)(=A(2)*X↑2+A(1)*X+A(Ø))
```

Notre programme devient :

```
10 A=A(N)
20 FOR I=N-1 TO 0 STEP -1
30 A=A*X+A(I)
40 NEXT I
```

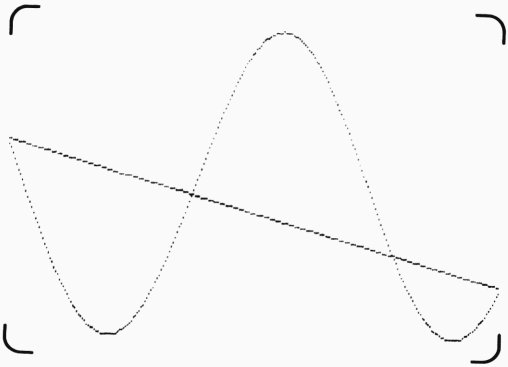
Nous n'avons plus qu'une multiplication et une addition à chaque passage. Ouf !

Programmation

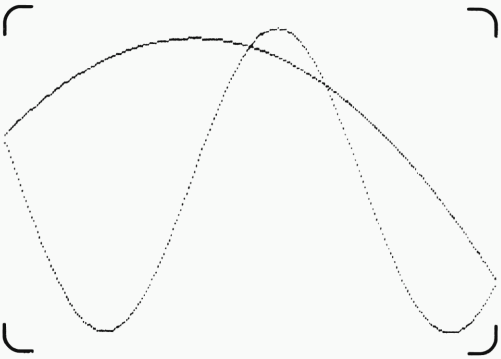
Si vous nous avez suivi jusqu'ici, vous serez ravi de taper ce programme (qui nécessite d'avoir tapé le précédent) :

```
10 REM APPROXIMATION PAR POLYNOMES
20 DEFFNF(X)=COS(X)
30 LG=    HT=
40 DY=7*(HT+1)/8
50 C1=HT-(HT+1)/16:REM MISE EN PAGE
100 REM MODE TEXTE,EFFACAGE D'ECRAN
110 INPUT"BORNES DE L'INTERVALLE D'ETUDE";XI,XS
120 IF XS<=XI THEN 110
130 DIM X(30,31),A(30)
140 N=1
200 REM CALCUL DE MINIMUM ET DE MAXIMUM
210 MAX=-1E30:MIN=1E30
220 FOR I=0 TO LG
230 X=XI+I*(XS-XI)/LG
240 F=FNF(X)
250 IF F>MAX THEN MAX=F
260 IF F<MIN THEN MIN=F
270 NEXT I
1000 REM TRACE FONCTION
1010 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
1020 FOR I=0 TO LG
1030 X=XI+I*(XS-XI)/LG
1040 F=FNF(X)
1050 C=DY*(F-MIN)/(MAX-MIN)
1060 C=C1-INT(C)
1070 POINT I,C
1080 NEXT I
2000 REM INITIALISATION DES TABLEAUX
2010 D=(XS-XI)/N
2020 FOR I=0 TO N
2030 X=XI+D*I
2040 FOR J=0 TO N
2050 X(I,J)=X^J
2060 NEXT J
2070 X(I,N+1)=FNF(X)
2080 NEXT I
2500 GOSUB 50000:REM RESOLUTION DU SYSTEME
3000 REM TRACE APPROXIMATION
3010 FOR I=0 TO LG
3020 X=XI+I*(XS-XI)/LG
3100 REM CALCUL DE LA VALEUR D'UN POLYNOME
     PAR LA METHODE DE HORNER
3110 F=A(N)
3120 FOR J=N-1 TO 0 STEP -1
3130 F=F*X+A(J)
3140 NEXT J                                    →
```

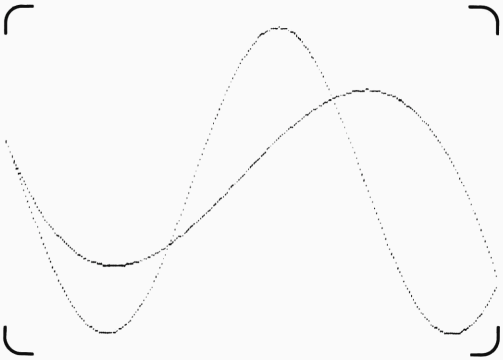
```
3150 C=DY*(F-MIN)/(MAX-MIN)
3160 C=C1-INT(C)
3170 IF C<0 OR C>HT THEN 3190
3180 POINT I,C
3190 NEXT I
4000 REM ATTENTE APRES TRACE
4010 A$=INKEY$:REM OU GET A%
4020 IF A$="" THEN 4010
4030 N=N+1:REM LE DEGRE AUGMENTE
4040 GOTO 1000
```



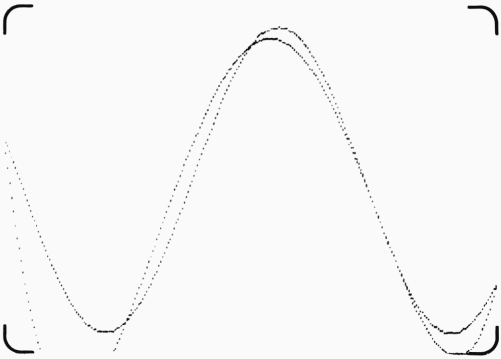
FNF(X)=COS(X)
XI=-5 : XS=4
ETAPE 1



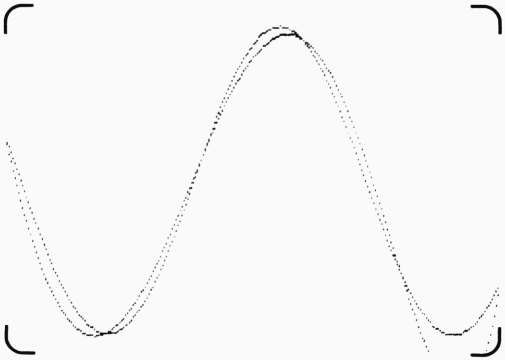
FNF(X)=COS(X)
XI=-5 : XS=4
ETAPE 2



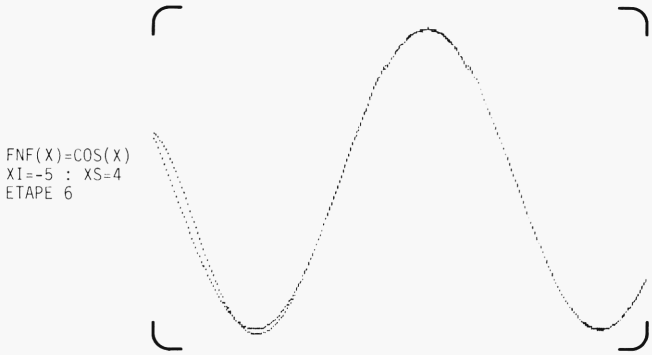
FNF(X)=COS(X)
XI=-5 : XS=4
ETAPE 3



FNF(X)=COS(X)
XI=-5 : XS=4
ETAPE 4



FNF(X)=COS(X)
XI=-5 : XS=4
ETAPE 5



Seuls deux endroits de ce programme appellent un commentaire :

- à la ligne 13Ø, nous avons initialisé arbitrairement les tableaux avec N=30. Généralement, le calcul d'une quinzaine de termes est très suffisant, et il en faut souvent beaucoup moins ;
- l'initialisation du tableau X se fait aux lignes 2ØØØ et suivantes ;
- la longueur D représente l'écart entre les abscisses des points par où devra passer la fonction polynôme ;
- on remplit le tableau X(I,J) avec les puissances des abscisses, sauf pour la dernière colonne où l'on prend les valeurs de la fonction étudiée.

Au niveau utilisation du programme, il suffit, pour passer d'une approximation à la suivante, de taper n'importe quelle touche (4ØØØ à 4Ø4Ø).

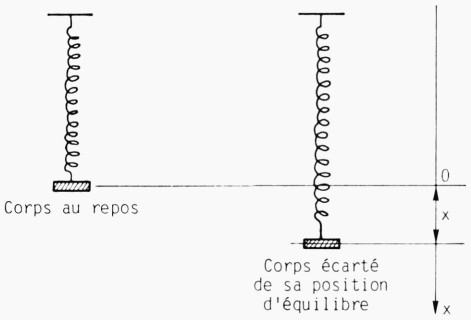
Il est bien sûr possible d'améliorer ce programme en mettant de la couleur, en assurant un tracé plus rapide de la fonction. Pour ce qui est de la partie calcul, il remplit convenablement son rôle : toute fonction polynôme est retrouvée avec exactitude, les fonctions continues sont approchées graphiquement de manière satisfaisante, et très rapidement si les variations ne sont pas trop irrégulières. Par contre, il ne faut pas lui demander plus qu'il ne peut donner : essayer d'approcher une fonction discontinue donne des résultats assez curieux (essayez INT(X)), et le calcul n'est pas assez précis pour donner un développement limité ou en série entière.

Chapitre III
Systèmes différentiels

Les courbes tracées jusqu'à présent étaient définies par différents types d'équation : cartésienne, paramétrique, polaire. Nous allons étudier une façon de définir non pas une courbe, mais toute une famille de courbes, comme solution d'un système différentiel.

Encore un "gros mot" après les épicycloïdes, hypocycloïdes ... Et il eut été plus exact d'écrire courbes intégrales d'un système différentiel ! Pour préciser le sens de ces "gros mots", prenons quelques exemples empruntés à la physique de terminale C ou D. Dans chacun des phénomènes, les fonctions cherchées dépendent de la variable temps t.

Corps suspendu à un ressort



Le mouvement d'un corps de masse m suspendu à un ressort, puis écarté de sa position d'équilibre, est régi par l'équation :

$$mx'' = -kx$$

(k : constante de raideur du ressort).

Mouvement d'un pendule

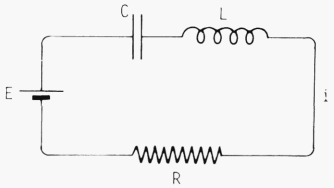
Le mouvement du pendule de masse m tournant autour d'un axe passant par O est régi par l'équation :

$$m\ell^2\theta'' = -mg\ell \sin\theta$$

soit $$\theta'' = -\frac{g}{\ell} \sin\theta$$

En général, en classe terminale, on suppose θ petit pour assimiler θ et sin θ, ce qui fournit $\theta'' = -\frac{g}{\ell}\theta$, équation qui est résoluble.

Intensité d'un courant



Dans un circuit comprenant une résistance R, un condensateur C, une inductance L et une f.e.m.E, l'intensité du courant et la quantité d'électricité q vérifient :

$$\begin{cases} E = \frac{q}{C} + Ri + L\frac{di}{dt} \quad \text{(loi d'Ohm)} \\ i = \frac{dq}{dt} \end{cases}$$

soit $$\begin{cases} q' = i \\ i' = \frac{1}{L}\left(E - \frac{q}{C} - Ri\right) \end{cases}$$

Dans les deux premiers cas, la fonction cherchée, x ou θ, vérifie une équation différentielle, c'est-à-dire une équation où figurent des dérivées de la fonction inconnue. La première équation ($x'' = -\frac{k}{m}x$) se résoud facilement, mais pas la seconde ($\theta'' = -\frac{g}{\ell}\sin\theta$). Dans le dernier cas, il s'agit d'un système différentiel, c'est-à-dire de plusieurs équations différentielles.

Nous nous placerons d'emblée dans le cas de systèmes différentiels. Cela peut paraître bizarre : a priori, un système différentiel est plus compliqué qu'une équation différentielle. Oui..., mais de nombreuses équations

différentielles se ramènent à des systèmes. Ainsi, pour nos deux premiers exemples :

$x'' = -\frac{k}{m}x$ s'écrit aussi $\begin{cases} z = x' \\ z' = x'' = -\frac{k}{m}x \end{cases}$ soit $\begin{cases} x' = z \\ z' = -\frac{k}{m}x \end{cases}$

$\theta'' = -\frac{g}{\ell}\sin\theta$ s'écrit aussi $\begin{cases} z = \theta' \\ z' = \theta'' = -\frac{g}{\ell}\sin\theta \end{cases}$ soit $\begin{cases} \theta' = z \\ z' = -\frac{g}{\ell}\sin\theta \end{cases}$

Physiquement z, dérivée de x ou de θ, traduit une vitesse (angulaire pour θ).

Toutes les situations physiques vues plus haut se ramènent donc à la même formalisation mathématique : deux fonctions inconnues x et y d'une même variable t, dont les dérivées x' et y' se calculent en fonction de x et y, soit :

$$(S) \begin{cases} x' = f(x,y) \\ y' = g(x,y) \end{cases}$$

Les fonctions f et g ne dépendent donc pas de la variable t.

Nous allons chercher à représenter les points M d'un plan tels que ses coordonnées x et y vérifient un tel système (S). Les mathématiciens ont montré que, pour de "bonnes fonctions" f et g, il n'y a qu'une seule courbe intégrale de (s) passant par un point donné M_0 (x_0, y_0). Préciser le sens du terme "bonnes fonctions" est hors du cadre de ce livre (niveau maîtrise de mathématiques pour les amateurs !). Prenant différents points M_0, nous chercherons simplement à représenter les courbes passant par ces points. Comme bien souvent nous limiterons notre étude à une certaine fenêtre définie par :

$$Xi \leqslant X \leqslant XS \quad \text{et} \quad Yi \leqslant Y \leqslant YS$$

Comment parvenir à de telles représentations ?

METHODE D'EULER

Il existe plusieurs procédés permettant d'obtenir, de proche en proche, des points des courbes-solutions. Un des plus simples est la méthode dite d'Euler.

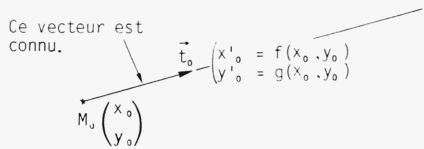
Revenons au système :

$$(S) \begin{cases} x' = f(x,y) \\ y' = g(x,y) \end{cases}$$

Si une courbe passe par le point M_0 (x_0, y_0), une demi-tangente à la courbe en M_0 est dirigée par le vecteur $\vec{t_0}$ de coordonnées :

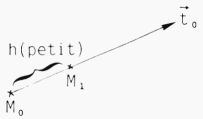
$$x'_0 = f(x_0,y_0) \quad \text{et} \quad y'_0 = g(x_0,y_0)$$

qu'on peut déterminer exactement ... enfin aux erreurs de calcul de la machine près :



Comme une courbe est assez voisine de sa tangente, nous considèrerons que la courbe cherchée passe par un point M_1 de la demi-tangente assez proche

de M_0. Précisons : M_1 est situé à une distance h, assez petite, fixée par l'utilisateur, de M_0. L'arc M_0M_1 de la courbe est remplacé par le segment M_0M_1... ce qui a été fait une fois peut être répété : M_1 remplace M_0 et on applique le même procédé :



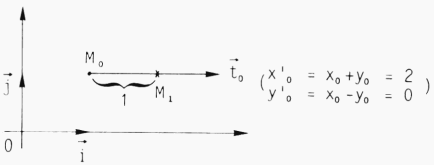
$\vec{t}_0$ dirige la demi-tangente.

On supposera que la courbe passe par M_1.

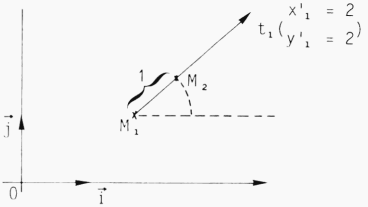
Prenons un exemple numérique :

$$(S)\quad \begin{cases} x' = x + y \\ y' = x - y \end{cases}$$

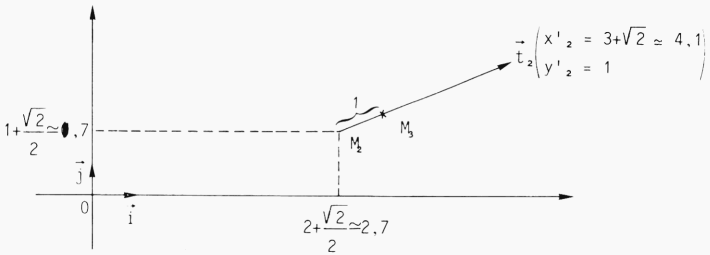
Supposons $x_0 = y_0 = 1$ et prenons h=1. Ce choix de h est mauvais. Mais il va nous permettre d'illustrer la méthode et de tracer une figure lisible.



On a facilement $M_1 \begin{pmatrix} 2 = x_1 \\ 0 = y_1 \end{pmatrix}$ d'où une demi-tangente en M_1 dirigée par $\vec{t}_1 \begin{pmatrix} x'_1 = x_1 + y_1 = 2 \\ y'_1 = x_1 - y_1 = 2 \end{pmatrix}$.

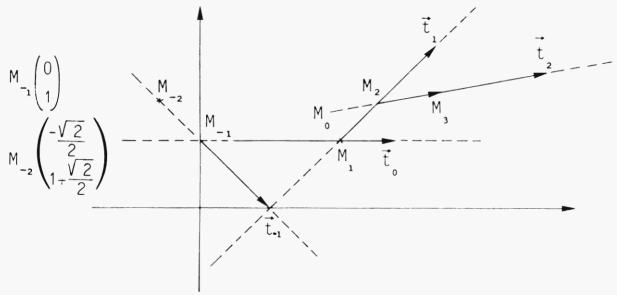


Là encore, on a facilement $M_2 \begin{pmatrix} x_2 = 2 + \sqrt{2}/2 \\ y_2 = 1 + \sqrt{2}/2 \end{pmatrix}$ d'où $\vec{t}_2 \begin{pmatrix} x'_2 = x_2 + y_2 = 3+\sqrt{2} \\ y'_2 = x_2 - y_2 = 1 \end{pmatrix}$



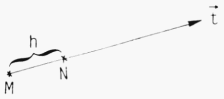
Nous nous arrêterons là, pensant que le principe est compris. La courbe cherchée est remplacée par la ligne brisée $M_0 M_1 M_2 M_3 \ldots$

Nous n'avons ainsi obtenu qu'une branche de la courbe : des points situés "après M_0". Peut-on obtenir l'autre branche (points "avant M_0") ? La réponse est affirmative : il suffit de reporter la distance h sur les demi-tangentes orientées en sens inverse et de répéter le procédé.



L'algorithme utilisé est donc le suivant : le point M est remplacé par le point N tel que :

$$\vec{MN} = h \frac{\vec{t}}{\|\vec{t}\|}$$



ou encore en posant :

$DX = F(X,Y)$ $\quad DY = G(X,Y)$ $\quad$ (dérivées de X et Y)

X est remplacé par $X + \dfrac{h\,DX}{\sqrt{DX^2 + DY^2}}$

et Y par $Y + \dfrac{h\,DY}{\sqrt{DX^2 + DY^2}}$

tant que X reste entre Xi et XS, Y entre Yi et YS, et que $DX^2 + DY^2$ est non nul.

La programmation, tout en restant simple, ne sera pas aussi élémentaire qu'il peut sembler à première vue.

PROGRAMMATION

Il nous faut :

- introduire les données indispensables :

  Xi, XS, Yi, YS pour la fenêtre-écran
  H pour le pas sur la tangente (positif) ;

- choisir un point initial M_0 (XØ,YØ) dans la fenêtre, puis utiliser la méthode d'Euler.

L'algorithme donné ne permet de visualiser qu'une branche de courbe (points "après M_0"). Pour passer simplement à l'autre branche, remarquons qu'il suffit de changer H en son opposé : cela inverse le sens de parcours sur la tangente. Lorsque H aura changé deux fois de signe, donc sera redevenu positif, c'est que les deux branches de la courbe auront été tracées.

Le programme suivant doit donc convenir :

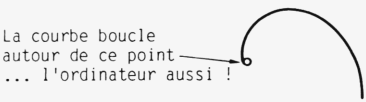
```
40 REM SYSTEMES DIFFERENTIELS
50 LG=    :HT=
100 REM INTRODUCTION DES DONNEES
110 REM MODE TEXTE
120 INPUT"ABSCISSE INFERIEURE";XI
130 INPUT"ABSCISSE SUPERIEURE";XS
140 IF XI>=XS THEN 120
150 INPUT"ORDONNEE INFERIEURE";YI
160 INPUT"ORDONNEE SUPERIEURE";YS
170 IF YI>=YS THEN 150
180 INPUT"PAS SUR LA TANGENTE";H
190 IF H<=0 THEN 180
200 REM TRACE
210 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
    RND<0) OU RND<1) SELON VOTRE BASIC
220 X0=XI+(XS-XI)*RND(0):Y0=YI+(YS-YI)*RND(0)
230 X=X0:Y=Y0
240 DX=    :DY=     :REM DERIVEES DE X ET DE Y
250 D=SQR(DX*DX+DY*DY)
260 IF D=0 THEN 400
270 X1=X+DX*H/D:Y1=Y+DY*H/D
280 IF X1<XI OR X1>XS OR Y1<YI OR Y1>YS
    THEN 400
290 DROITE LG*(X-XI)/(XS-XI),HT*(YS-Y)/(YS-YI)
    A LG*(X1-XI)/(XS-XI),HT*(YS-Y1)/(YS-YI)
300 X=X1:Y=Y1:GOTO 240
400 REM PLUS POSSIBLE DE CONTINUER
410 H=-H:IF H>0 THEN 220:REM AUTRE COURBE
420 GOTO 230:REM AUTRE BRANCHE DE LA MEME
    COURBE
```

En ligne 250, DX et DY doivent être égaux aux fonctions f(x,y) et g(x,y) définissant le système qu'on veut étudier.

En prenant par exemple :

$$DX = \frac{2X-3Y}{4} \qquad DY = \frac{2X+Y}{2}$$

avec Xi = Yi = -5, XS = YS = 5, H = Ø,1, vous constaterez une chose : à la place d'une famille de courbes, seule une courbe est tracée... avec ses deux branches tout de même ! Mais le résultat est décevant. Nos ennuis viennent du fait que la courbe obtenue est une spirale qui s'enroule autour d'un point :



Rectifions donc notre programme pour éviter ces ennuis. Rajoutons d'abord une variable N totalisant le nombre de segments tracés. Si N est plus grand qu'un nombre NMAX que nous nous serons fixé, nous arrêterons le tracé de la branche en cours. Nous éviterons ainsi de boucler trop longtemps autour de certains points.

Prenons une autre précaution que certains lecteurs attendent depuis quelques pages. Les points où DX = DY=Ø sont particuliers, même très particuliers : le vecteur $\vec{t}$ qui dirige en général la tangente est nul ! De tels points sont dits singuliers. Dans la pratique, compte tenu de l'accumulation d'erreurs, on a bien peu de chances d'obtenir exactement DX = DY = Ø. Si DX et DY sont tous deux assez petits, on ne doit pas être loin d'un point singulier. On remplacera donc le test :

```
IF D = Ø THEN...
```

par

```
IF D < S THEN...
```

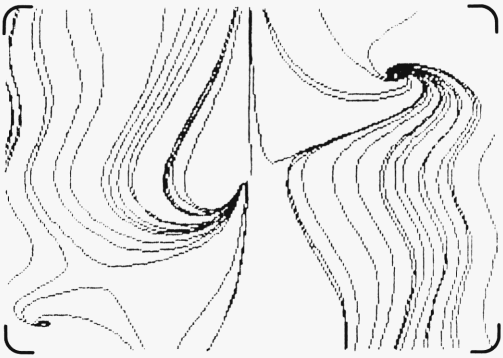
S étant un seuil fixé à l'avance par l'utilisateur.

Ces modifications conduisent à rajouter une ligne et à en changer trois. Il suffit de taper :

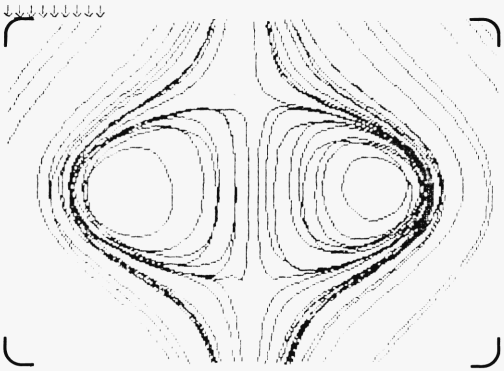
```
50 NMAX=    :S=
230 N=1:X=X0:Y=Y0
260 IF D<S THEN 400
270 N=N+1:X1=X+DX*H/D:Y1=Y+DY*H/D
280 IF N>NMAX OR X1<XI OR X1>XS OR Y1<YI
    OR Y1>YS THEN 400
```

tout le reste étant inchangé.

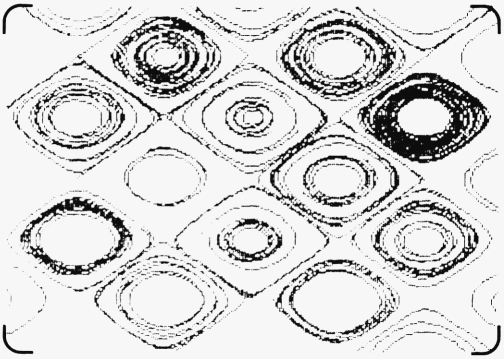
Ainsi rectifié, le programme doit bien tourner... Mais ne vous lancez pas n'importe comment dans cette jungle des systèmes différentiels où le dément et le sublime se côtoient !



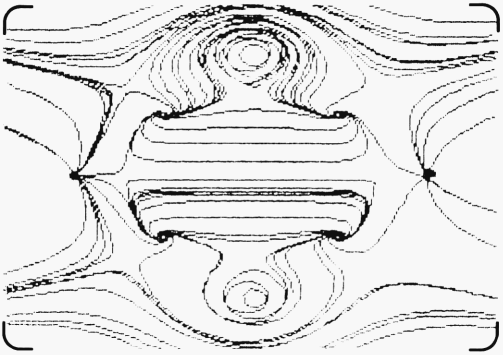
```
DX=SIN(X*Y)  : DY=X-Y
XI=-3 : XS=3 : YI=-3 : YS=3
H=.1
```



```
DX=X*Y : DY=1-X*X-Y*Y
XI=-2  : XS=2 : YI=-2 : YS=2
H=.1
```



```
DX=SIN(Y) : DY=SIN(X)
XI=-9 : XS=9 : YI=-9 : YS=9
H=.1
```



```
DX=(X*X+Y*Y-1)*(X*X+Y*Y-2) : DY=X*Y*(X*X-Y*Y)
XI=-2 : XS=2 : YI=-2 : YS=2
H=.Ø5
```

Fonction X'(DX)	Fonction Y'(DY)	XI	XS	YI	YS
SIN(XY)	X-Y	-3	3	-3	3
XY	$1-X^2-Y^2$	-2	2	-2	2
XY^2	1-XY	-2	2	-2	2
X^2Y^2	1-XY	-3	3	-3	3
$\frac{XY}{1+X^2Y^2}$	X-Y	-5	5	-5	5
X^2-Y	Y^2-X	-2	2	-2	2
1-X-Y	X^2+Y^2-1	-5 ou -2	5 2	-5 -2	5 2
X SIN(Y)	Y SIN(X)	-10	10	-10	10
X-XY	Y+XY	-5	5	-5	5
X^2-Y^2	|XY|	-5	5	-5	5
$(X^2+Y^2-1)(X^2+Y^2-2)$	X^2-Y^2	-3	3	-3	3
SIN(Y)	SIN(X)	-9	9	-9	9
$(X^2+Y^2-1)(X^2+Y^2-2)$	$XY(X^2-Y^2)$	-2	2	-2	2
$((X+Y)^2-1)((X-Y)^2-1)$	$XY(X^2+Y^2-1)$	-2	2	-2	2
$((X+Y)^2-1)((X-Y)^2-1)$	X^2-Y^2	-2	2	-2	2
$SIN(X^2+Y^2)$	$SIN(X^2-Y^2)$	-3	3	-3	3
X^4-Y^2	X^2-Y^4	-2	2	-2	2

EXEMPLES

Comment choisir convenablement les quantités NMAX, H et S ? Plus H est petit, plus le tracé est lent puisque les longueurs M_0M_1, M_1M_2, M_2M_3... sont toutes égales à H... et donc plus le nombre de segments requis pour tracer la courbe est grand. Comme il est souhaitable de représenter un assez grand nombre de courbes (vingt à trente, parfois beaucoup plus), il s'agit de trouver un compromis entre la vitesse d'exécution (les tracés sont assez longs) et la précision. De plus, la résolution de l'écran utilisé joue toujours dans le même sens : meilleure elle est, plus on a intérêt à tracer de segments.

NMAX=100 et H=0,1 nous ont donné des résultats corrects. Si vous diminuez H de moitié, doublez NMAX... et patientez. Quant à S, il dépend a priori de H. $S=10^{-2}$ s'est révélé satisfaisant. Ces valeurs sont données à titre indicatif : adaptez-les à votre système.

Avant de vous donner un catalogue d'exemples, nous vous invitons à tester des systèmes du type :

$$(S)\ \begin{cases} x' = ax + by \\ y' = cx + dy \end{cases} \qquad a, b, c, d : \text{nombres.}$$

(systèmes différentiels linéaires à coefficients constants !).

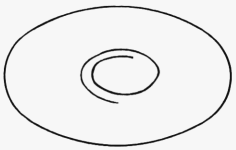
Essayez donc avec XI = YI = -3, XS = YS = 3, H = 0,1 :

$$\begin{cases} X' = 3X \\ Y' = 3Y \end{cases} \quad \begin{cases} X' = 3X - 2Y \\ Y' = 2X - Y \end{cases} \quad \begin{cases} X' = 3X - Y \\ Y' = 3X - Y \end{cases} \quad \begin{cases} X' = X \\ Y' = 2X + 3Y \end{cases}$$

$$\begin{cases} X' = 3X + Y \\ Y' = -4X - 2Y \end{cases} \quad \begin{cases} X' = \dfrac{2X - 3Y}{4} \\ Y' = \dfrac{2X + Y}{2} \end{cases} \quad \begin{cases} X' = Y \\ Y' = -X \end{cases}$$

sous la même forme, quelle variété de courbes ! Des demi-droites, des droites parallèles, des spirales... Dans tous ces exemples, l'origine du repère (X=Y=Ø) est un point singulier qui peut être un pôle attractif (cas des spirales, des demi-droites) ou répulsif (courbes à allure hyperbolique)... On obtient aussi des familles de courbes toutes tangentes à une même droite... A vous de retrouver le(s) système(s) correspondant à la(aux) situation(s) évoquée(s) !

Le dernier système, X'=Y, Y'=-X, mérite des commentaires. A l'écran, vous obtiendrez des ovales : les plus grands sont à peu près fermés, les plus petits s'enroulent :



Pourtant, si X'=Y et Y'=-X,

$$XX'+YY'=XY-XY=0$$

et comme $XX'+YY'=\frac{1}{2}(X^2+Y^2)'$, X^2+Y^2 est constant : les courbes étudiées sont des cercles. Comme l'écran n'est pas orthonormé, ces cercles sont représentés par des ellipses. Bizarres ces ellipses qui s'enroulent ! Et oui ! On peut incriminer le choix de H trop grand : essayez H=0,05 ou même

moins. Vous obtiendrez peut-être une légère amélioration, en précision, sûrement une nette augmentation en temps d'exécution. Ne cherchez pas trop longtemps ce qui est en cause : c'est la méthode elle-même, ni plus, ni moins. Pour obtenir des ellipses réussies, il faut utiliser des méthodes plus précises que la méthode d'Euler. Par exemple, la méthode de Runge-Kutta, assez classique : voyez-la dans les ouvrages cités dans la bibliographie.

Dans les exemples du catalogue, si vous observez des courbes qui s'enroulent, pensez qu'il s'agit peut-être de courbes fermées. Ayez toujours présent à l'esprit que la méthode d'Euler a ses limites... et qu'elles sont vite atteintes. Si c'est le hasard qui vous joue des tours en remplissant systématiquement certaines zones et en en ignorant d'autres, intervenez. Arrêtez le programme. Tapez :

XØ = valeur que vous voulez : YØ = valeur que vous voulez : H = ce que vous avez introduit : GOTO 24Ø

Retapez la valeur de H car, si vous arrêtez le programme lorsque H est négatif, vous n'obtiendrez pas toute la courbe souhaitée. Ainsi, vous imposerez, à une courbe, de passer par un point voulu et vous peuplerez des zones délaissées par le hasard.

Le programme ainsi rédigé boucle sur lui-même (dès que la courbe sort de la fenêtre-écran choisie, il passe soit à l'autre branche, soit à une autre courbe). A vous de l'arrêter, lorsque vous estimez les résultats satisfaisants. C'est affaire de goût... et de temps : il faut tracer au minimum une trentaine de courbes, souvent une cinquantaine, parfois jusqu'à la centaine !

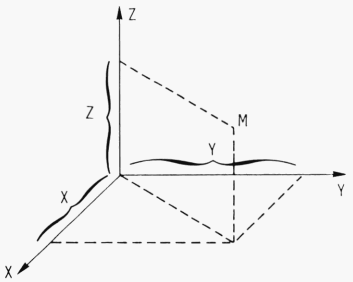
Maintenant, explorez la jungle. Observez bien les évolutions de certaines courbes : débiles, démentes, presque toujours belles !

Troisième partie
ESPACE

Chapitre I
Surfaces

Après le plan, étudions quelques méthodes graphiques concernant l'espace. Il est toujours assez difficile de bien voir ce qui se passe dans l'espace car toutes les figures que nous traçons sont planes.

Dans l'espace, un point M est repéré par trois quantités X, Y, Z (abscisse, ordonnée, cote).



Le repère OXYZ sera toujours représenté comme ci-dessus : OZ vers le haut, OY vers la droite, OX vers le bas à gauche. Dans l'espace, ces trois axes sont deux à deux perpendiculaires. Sur chacun, nous prendrons la même unité : il s'agit alors d'un repère orthonormé. Il faut donc imaginer OX perpendiculaire à cette page et dirigé vers vous.

Comment généraliser à l'espace ce que nous venons d'étudier, des courbes ? Nous avons vu que, dans un plan, la façon la plus générale de définir une courbe est la représentation paramétrique :

$$X = F(T)$$
$$Y = G(T)$$

Les autres cas envisagés, équation cartésienne ou équation polaire, s'y ramènent. La généralisation à l'espace est aisément compréhensible : il suffit que Z soit lui aussi fonction de T :

$$X = F(T)$$
$$Y = G(T)$$
$$Z = H(T)$$

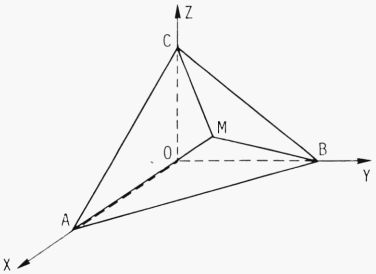
détermine paramétriquement une courbe dans l'espace. Une telle courbe, lorsqu'elle n'est pas plane, est qualifiée de "gauche".

Nous n'étudierons pas de telles courbes. Cela peut vous paraître bizarre, cher lecteur ! Ce n'est ni par paresse, ni parce que, malgré tous nos efforts, nous ne sommes pas parvenus à rédiger un programme adéquat. La raison est plus simple : cela n'a guère d'intérêt. Rien ne différenciera, sur un écran, une courbe plane d'une courbe gauche : pas moyen de donner un effet de profondeur à une telle courbe.

Par contre, représenter plusieurs courbes, avec perspective, est possible. Mais alors, nous sortons du domaine des courbes pour entrer dans celui des surfaces.

Comment définir une surface ? Prenons des exemples.

Premier exemple



Le plan passant par les trois points : A(1,0,0), B(0,1,0) et C(0,0,1) est une surface. Tout point M de ce plan est défini par :

$$\vec{CM} = u\ \vec{CA} + v\ \vec{CB}$$

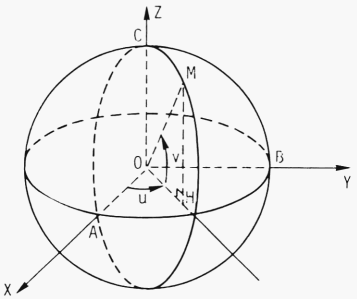
u et v étant deux nombres.

Comme $\vec{CA}(1,0,-1)$ et $\vec{CB}(0,1,-1)$, les coordonnées X, Y, Z de M vérifient :

$$X = u$$
$$Y = v$$
$$Z = -u-v+1$$

(d'où X+Y+Z=1, forme qui ne sera pas utile).

Deuxième exemple



La sphère de centre O de rayon 1 est elle aussi une surface. Si M, de la sphère, se projette en H sur le plan ABC, posons :

$(\vec{OA},\vec{OH}) = u \qquad (\vec{OH},\vec{OM}) = v$

d'où $\overline{OH} = \cos v \qquad \overline{HM} = \sin v$

Les coordonnées de M sont donc :

$X = \overline{OH} \cos u \qquad Y = \overline{OH} \sin u \qquad Z = \overline{HM}$
$X = \cos u \cos v \qquad Y = \sin u \cos v \qquad Z = \sin v$

(d'où $X^2+Y^2+Z^2=1$, forme qui ne nous intéresse pas).

Il ne semble pas y avoir beaucoup de points communs entre ces deux exemples ! Pourtant, il en existe un : dans les deux cas, les coordonnées X, Y, Z des points de la surface dépendent toutes les trois de deux quantités appelées u et v.

De façon générale, une surface est formée de points dont les coordonnées X, Y, Z dépendent de deux paramètres :

$X = F(u,v)$
$Y = G(u,v)$
$Z = H(u,v)$

Avec un peu de recul, de telles expressions sont assez normales pour des surfaces. Nous avons dit que si X, Y, Z dépendent d'une seule quantité T, on définit une courbe. Pour considérer une famille de courbes, donc une surface, il faut nécessairement un premier paramètre permettant de définir une courbe et un second permettant de faire varier la courbe dans la famille.

Voilà pourquoi les expressions signalées, $X+Y+Z=1$ pour le plan, $X^2+Y^2+Z^2=1$ pour la sphère, sont inutiles : elles masquent le fait plus important que X, Y, Z dépendent tous trois de deux paramètres.

La notion de surface étant précisée, du moins nous l'espérons, nous pouvons tenter de représenter une surface à l'écran. D'emblée, il convient de choisir. Ou bien nous visualisons une surface définie de la façon la plus générale ($X=F(u,v)$, $Y=G(u,v)$, $Z=H(u,v)$), sans tenir compte des parties vues ou cachées, ou bien nous restreignons nos ambitions quant au type de surface et nous tenons compte des parties vues ou cachées.

Nous avons choisi la seconde solution, plus délicate, mais plus intéressante. Il existe des algorithmes simples tenant compte des parties vues ou cachées pour les surfaces du type :

$Z = H(X,Y)$

Une telle expression ne semble pas exactement du type voulu : où sont u, v et les fonctions F et G ? Il suffit de poser :

$X = u$
$Y = v$
$Z = H(u,v)$

pour se convaincre que Z fonction de X et Y détermine bien une surface. Dorénavant, nous écrirons $Z=F(X,Y)$: appeler F et non plus H la fonction Z de X et Y ne change rien.

Maintenant, assez de théorie, bien que nous n'ayons, pour l'instant, traité ces questions qu'en surface !

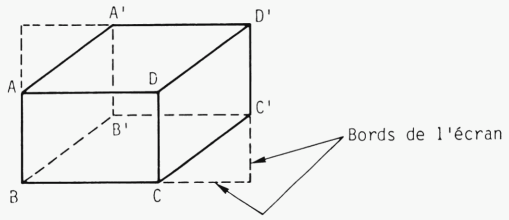
SURFACES Z=F(X,Y)

Que va-t-il nous falloir et comment allons-nous procéder ? Lors de l'étude de l'enveloppe d'une famille de droites, nous avons vu que nous ne représentions sur l'écran que ce qui correspondait à $Xi \leqslant X \leqslant XS$ et $Yi \leqslant Y \leqslant YS$; Xi, XS, Yi, YS définissaient la fenêtre-écran.

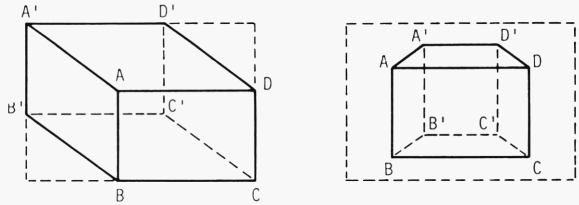
Ici, il faut préciser le volume que nous allons représenter. Imaginez que vous emballiez la surface dans un carton : le volume dont nous parlons est celui qui est intérieur au carton. Il s'agit d'un parallélépipède défini par six quantités :

Xi : valeur minimale de X
XS : valeur maximale de X
Yi : valeur minimale de Y
YS : valeur maximale de Y
Zi : valeur minimale de Z
ZS : valeur maximale de Z

Sur l'écran, nous le représenterons ainsi :



Ce choix est arbitraire. Nous aurions aussi bien pu choisir de le représenter ainsi :



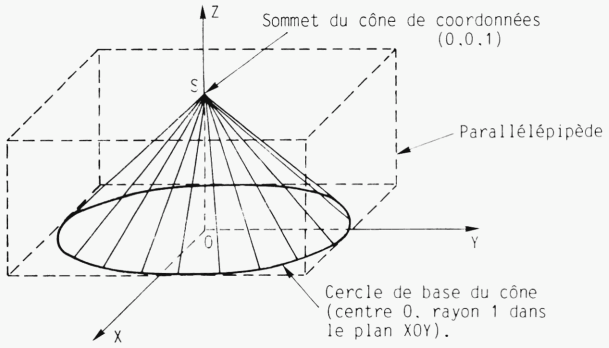
Nous avons fait un autre choix, mais une fois la méthode comprise, les adaptations à ces cas seront assez faciles à réaliser.

Ce parallélépipède ne sera pas tracé, mais il va nous servir de référence dans toute la suite.

La méthode utilisée

Pour bien comprendre la méthode utilisée, le mieux est d'analyser un exemple : un cône de révolution, le parallélépipède précédent étant défini par :

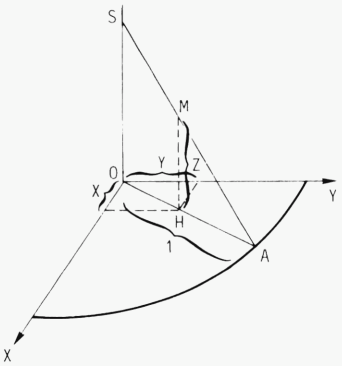
Xi = -1 XS = 1 Yi = -1 YS = 1 Zi = 0 ZS = 1

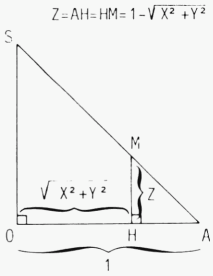


Rapporté au repère OXY, ce cône a pour équation :

$$Z = 1 - \sqrt{X^2+Y^2}$$

puisque, le triangle OSA étant rectangle isocèle :





Cette équation importe peu dans la méthode utilisée, mais autant expliquer d'où elle vient.

Pour visualiser le cône, nous allons le couper par des plans parallèles à YOZ. Ce choix, comme celui du parallélépipède, est arbitraire : nous aurions aussi bien pu prendre des plans parallèles à XOZ. Il faut bien faire un choix. Les plans parallèles à YOZ présentent l'avantage d'être de face par rapport à l'observateur. Si vous ne voyez pas bien ces coupes, imaginez un pain conique qu'on découpe en tranches (parallèles) : nos coupes correspondent à ces tranches. Il nous faudra donc introduire :

N : nombre de coupes à visualiser

Prenons, par exemple, sept coupes régulièrement espacées pour notre cône : la première correspondra au plan ABCD, la septième au plan A'B'C'D'.

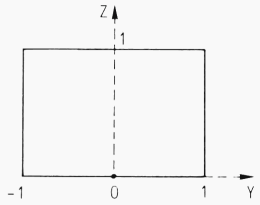
Elles seront espacées chacune de $\frac{2}{6}$ (dans le cas général de $\frac{XS-XI}{N-1}$), donc obtenues pour :

$X = 1 - 0 \times \frac{2}{6} = 1$ $\quad X = 1 - 1 \times \frac{2}{6} = \frac{2}{3}$ $\quad X = 1 - 2 \times \frac{2}{6} = \frac{1}{3}$ $\quad X = 1 - 3 \times \frac{2}{6} = 0$

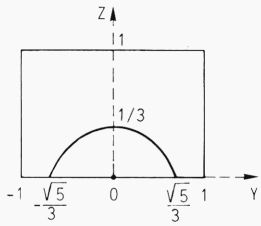
$X = 1 - 6 \times \frac{2}{6} = -1$ $\quad X = 1 - 5 \times \frac{2}{6} = -\frac{2}{3}$ $\quad X = 1 - 4 \times \frac{2}{6} = -\frac{1}{3}$

(cas général : $X = XS - (i - 1)\ \frac{XS - Xi}{N - 1}$ avec i variant de 1, pour la première coupe, à N pour la dernière).

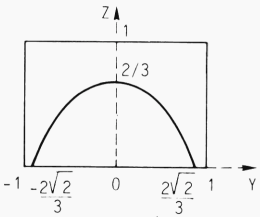
La coupe obtenue a pour équation $Z = 1 - \sqrt{X^2 + Y^2}$, X devant être remplacé par la valeur qu'il a pour la coupe considérée (± 1, $\pm \frac{2}{3}$, $\pm \frac{1}{3}$, 0 pour le cône, $XS - (i-1)\ \frac{XS - Xi}{N - 1}$ pour la ième dans le cas général) ; Z est alors fonction de Y seul. Représentons les courbes de ces différentes fonctions :



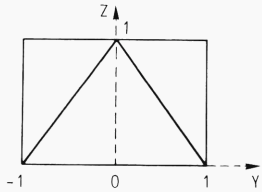
pour $X = \pm 1$
$Z = 1 - \sqrt{1 + Y^2}$
coupe réduite à un point.



pour $X = \pm \frac{2}{3}$
$Z = 1 - \sqrt{\frac{4}{9} + Y^2}$
(parabole)

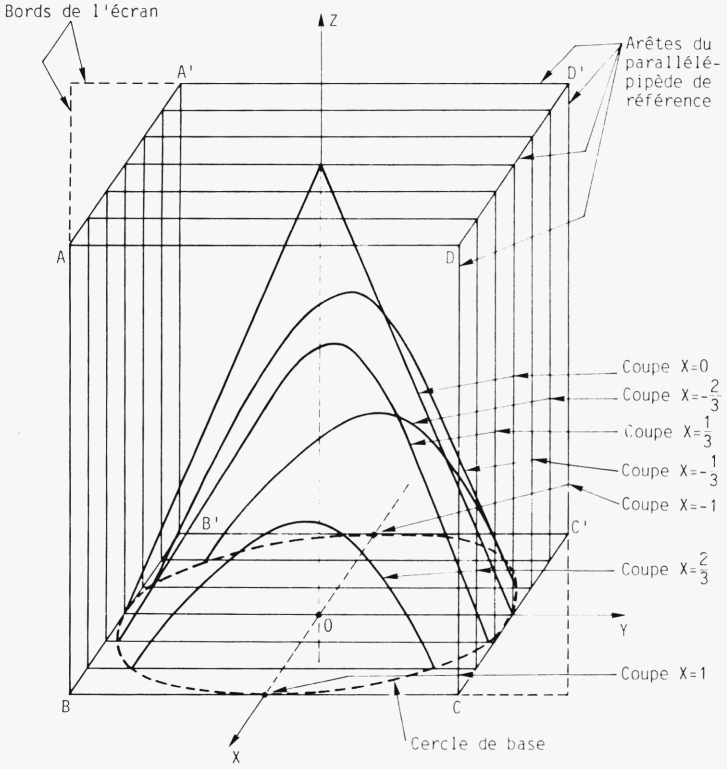


pour $X = \pm \frac{1}{3}$
$Z = 1 - \sqrt{\frac{1}{9} + Y^2}$
(parabole)



pour X = 0
$Z = 1 - |Y|$
(2 segments de droite)

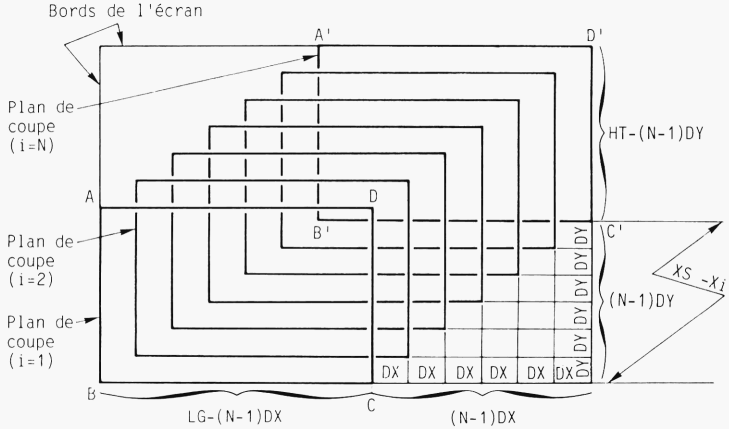
Visualiser le cône, c'est tracer ces sept coupes et bien les placer par rapport au parallélépipède de référence, ce qui se fait assez facilement en divisant chacun des segments AA', BB', CC', DD' en six segments égaux (N-1 dans le cas général) :



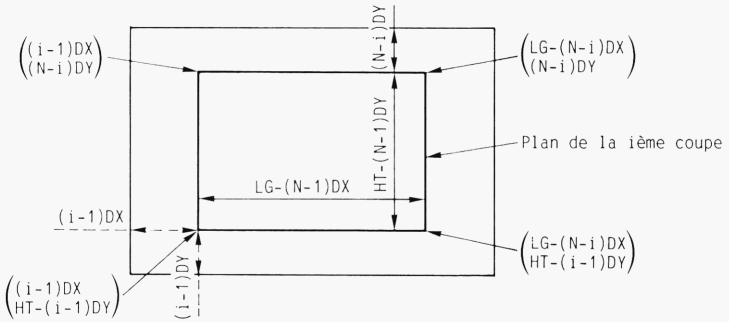
Nous avons vu que, pour N coupes régulièrement espacées, cet espacement est de $\frac{XS-XI}{N-1}$ dans le repère OXYZ. Deux quantités vont nous permettre de tenir compte de cet espacement, au décalage, entre les coupes à l'écran :

DX : décalage sur les X (écran)
DY : décalage sur les Y (écran)
} d'une coupe à une autre

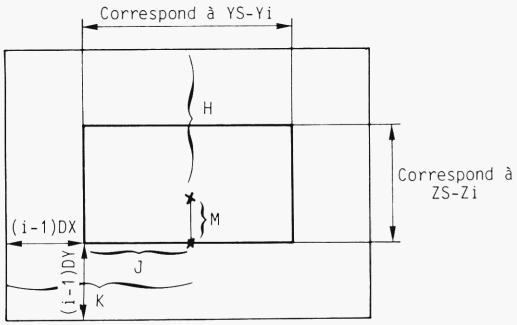
de sorte qu'à l'écran on ait (figure faite pour N=7) :



La ième coupe, qui correspond à $X=XS-(i-1)\frac{XS-XI}{N-1}$, est la courbe d'équation $Z=F(XS-(i-1)\frac{XS-XI}{N-1}, Y)$, lorsque Y varie de Yi à YS (Z étant compris entre Zi et ZS). Elle sera tracée, sur l'écran, dans le rectangle joignant les points :



Grâce à deux règles de trois, on en déduit que le point de coordonnées (X,Y,Z) dans l'espace est représenté sur l'écran par le point de coordonnées (K,H) (voir schéma ci-contre).



$$\frac{J-\emptyset}{LG-(N-1)DX} = \frac{Y-Yi}{YS-Yi}$$

d'où $J = (LG-(N-1)DX)\ \frac{Y-Yi}{YS-Yi} = L1(Y-Yi)$

soit $Y = Yi+J/L1$

$$\frac{M-\emptyset}{HT-(N-1)DY} = \frac{Z-Zi}{ZS-Zi}$$

d'où $M = (HT-(N-1)DY)\ \frac{Z-Zi}{ZS-Zi} = Zi(Z-Zi)$

$K = (i-1)DX+J$

$H = HT-((i-1)DY-M)$

Au niveau programmation, pour visualiser la ième coupe, nous aurons intérêt à effectuer une boucle sur J (de 0 à LG-(N-1)DX par pas de 1) plutôt que sur Y (de Yi à YS par pas de $\frac{YS-Yi}{LG-1-(N-1)DX}$) car J est entier alors que Y a toute chance de ne pas l'être.

Le programme suivant permet de visualiser la ième coupe, ainsi que son "cadre" (rectangulaire) pour la surface d'équation Z=F(X,Y). Par rapport à ce que nous venons de dire, nous avons ajouté dans ce programme :

- un rejet des points dont les cotes ne sont pas comprises entre Zi et ZS. Cela évitera un appel illégal de fonction dans l'affichage des points ;
- un test sur DX et DY. C'est arbitraire. Cherchant à représenter des surfaces, nous avons choisi que chaque cadre rectangulaire de coupe ait pour dimension les 2/3 de celle de l'écran. On ne prend donc que :

$0 < DX < \frac{LG}{3(N-1)}$ $\qquad$ $0 < DY < \frac{HT}{3(N-1)}$

```
20 REM COUPE D'UNE SURFACE
30 LG=    :HT=
110 GOSUB 1200:REM INTRODUCTION DES DONNEES
120 GOSUB 1500:REM TRACE D'UNE COUPE
130 END
1200 REM INTRODUCTION DES DONNEES
1210 REM MODE TEXTE,EFFACAGE D'ECRAN
1220 INPUT"NOMBRE DE COUPES";N
1230 IF N<=0 OR N<>INT(N) THEN 1220        →
```

```
1240 INPUT"ABSCISSE INFERIEURE";XI
1250 INPUT"ABSCISSE SUPERIEURE";XS
1260 IF XS<=XI THEN 1240
1270 INPUT"ORDONNEE INFERIEURE";YI
1280 INPUT"ORDONNEE SUPERIEURE";YS
1290 IF YS<=YI THEN 1270
1300 INPUT"COTE INFERIEURE";ZI
1310 INPUT"COTE SUPERIEURE";ZS
1320 IF ZS<=ZI THEN 1300
1330 INPUT"DECALAGE SUR X";DX
1340 IF DX<=0 OR DX<>INT(DX) OR DX>=LG/3/(N-1)
     THEN 1330
1350 INPUT"DECALAGE SUR Y";DY
1360 IF DY<=0 OR DY<>INT(DY) OR DY>=HT/3/(N-1)
     THEN 1350
1370 PRINT"TRACE DE LA COUPE NUMERO (ENTRE 1
     ET ";N;")";
1380 INPUT I
1390 IF I<1 OR I>N THEN 1370
1400 RETURN
1500 REM TRACE
1510 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
1520 L1=(LG-DX*(N-1))/(XS-XI)
1530 Z1=(HT-DY*(N-1))/(ZS-ZI)
1550 X=XS-((XS-XI)*(I-1)/(N-1))
1560 FOR J=0 TO LG-DX*(N-1)
1570 Y=YI+J/L1
1580 K=(I-1)*DX+J
1600 Z=    'REM EQUATION DE LA SURFACE
1610 IF Z<ZI OR Z>ZS THEN 1670
1620 M=Z1*(Z-ZI)+H=INT(HT-(I-1)*DY-M+.5)
1650 POINT K,M
1670 NEXT J
1700 REM TRACE DU CADRE DE LA COUPE
1710 DROITE (I-1)*DX,HT-(I-1)*DY
     A LG-(N-I)*DX,HT-(I-1)*DY
1720 DROITE LG-(N-I)*DX,HT-(I-1)*DY
     A LG-(N-I)*DX,(N-I)*DY
1730 DROITE LG-(N-I)*DX,(N-I)*DY
     A (I-1)*DX,(N-I)*DY
1740 DROITE (I-1)*DX,(N-I)*DY
     A (I-1)*DX,HT-(I-1)*DY
1750 RETURN
```

Si vous souhaitez visualiser toutes les coupes, il suffit d'enlever les lignes 137Ø à 139Ø où l'on introduit le rang de la coupe et de faire exécuter les différentes coupes par :

```
12Ø FOR i=1 TO N : GOSUB 152Ø : NEXT i
```

Vous pouvez aussi enlever le tracé du cadre rectangulaire qui n'apporte rien mais peut permettre, au départ, une meilleure compréhension de la méthode.

Si vous ne faites pas tracer ce cadre, ne soyez pas étonné, en prenant l'exemple du cône, de ne pas obtenir les coupes réduites à un point : il faut prendre exactement X=1 (ou -1) et Y=0... ce qui a toute chance de ne pas être fait... et le seul point de la coupe ne sera pas visualisé.

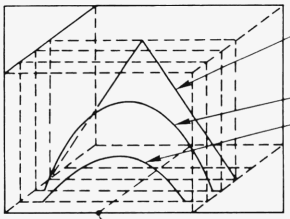
Lorsque l'on représente les N coupes, comme nous l'avons indiqué plus haut, il reste une question délicate : comment tenir compte des parties vues et cachées ?

Parties vues, parties cachées

Pour comprendre l'algorithme que nous allons utiliser, analysons encore l'exemple du cône. Traçons toutes les coupes en commençant par celle qui est la plus proche de l'observateur (X=1 dans l'exemple, X=XS dans le cas général). Autant certains des choix faits plus haut sont arbitraires (façon de représenter le parallélépipède, bornes sur DX et DY), autant commencer par la coupe la plus proche de l'observateur est essentiel.

Autre point important : que le lecteur suive bien, à partir des dessins de ce livre. Ce qui est vu ou caché dépend des décalages choisis entre les coupes, donc de DX et DY, du nombre de coupes N. Ce que nous allons dire sur le cône n'est pas forcément ce que vous aurez vu sur votre écran !

Sur le cône, donc, jusqu'à la troisième coupe, pas de problème :



La quatrième coupe, elle, est cachée au départ puis vue ensuite. Les termes caché et vu sont difficilement interprétables. Mieux vaut dire : au départ, la cinquième coupe est au-dessous de la quatrième, donc cachée ; dès qu'elle repasse au-dessus de la quatrième, elle devient visible. Et être "au-dessus" ou "au-dessous", c'est être plus grand ou plus petit que...

Précisons encore. Pour cela, revenons à ce qui se passe à l'écran. Appelons :

ligne de crête maximale : la courbe formée par les points les plus haut tracés ;

ligne de crête minimale : la courbe formée par les points les plus bas tracés.

Plus haut et plus bas doivent s'entendre dans le sens normal : au-dessus et au-dessous. Pourquoi cette précision ? Parce que, sur l'écran, l'axe des Y étant orienté vers le bas, un point sera plus haut qu'un autre si son Y (écran) est plus petit. Le programme tiendra compte des Y écran, l'énoncé de l'algorithme, non.

On comprend ainsi que :

- un point est caché s'il est entre la ligne de crête maximale et la ligne de crête minimale ;
- un point est vu s'il est soit au-dessus de la ligne de crête maximale, soit au-dessous de la ligne de crête minimale.

Ces lignes de crête seront prises en compte par deux tableaux Hi( ) et HS( ), dimensionnés à la longueur de l'écran :

Hi( ) : gère la ligne de crête minimale ;
HS( ) : gère la ligne de crête maximale.

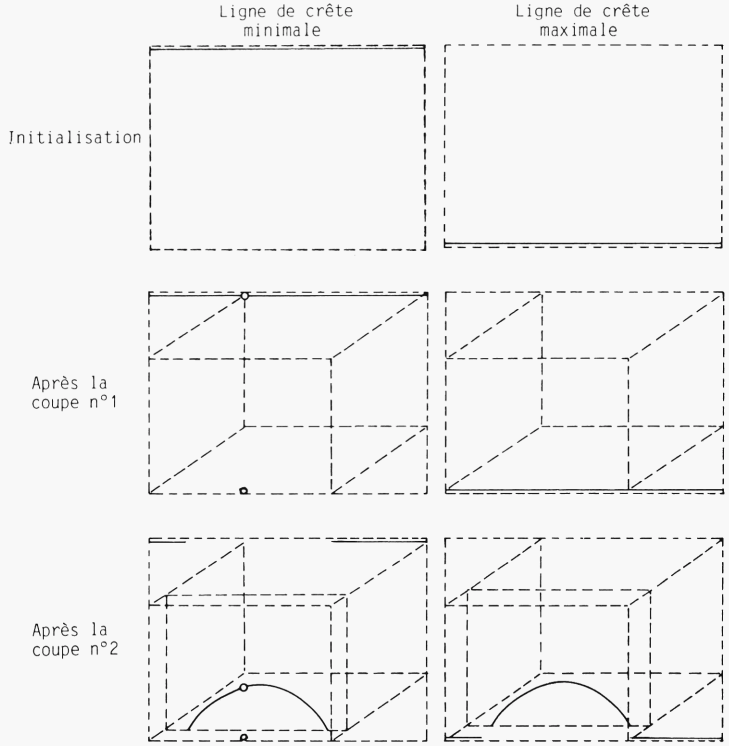
Par précaution, nous initialiserons ces tableaux à l'inverse de ce qu'ils signifient : au départ, Hi( ) représentera la première ligne de l'écran (tous les Hi( ) à Ø), HS représentant la dernière ligne (tous les HS à HT).

L'algorithme cité plus avant s'exprime alors pour un point de coordonnées K et H sur l'écran (K et H ont été définis, page 145) :

H > HS(K) : le point est vu, HS(K) devient H.
H < Hi(K) : le point est vu, Hi(K) devient H.
Hi(K) ⩽ H ⩽ HS(K) : le point est caché.

Ces comparaisons sont les comparaisons "normales". Compte tenu de l'orientation de l'axe des Y écran vers le bas, elles seront inversées dans le programme !

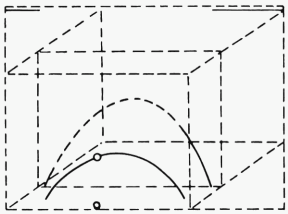
Suivons alors, pour le cône déjà considéré, l'évolution de ces lignes de crêtes :

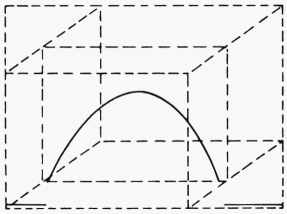


Ligne de crête minimale

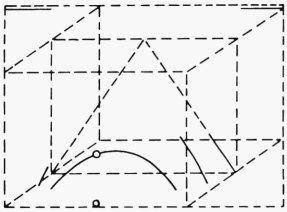
Ligne de crête maximale

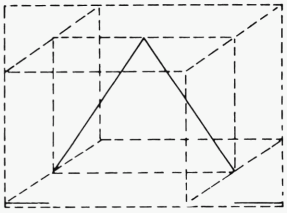
Après la coupe n°3



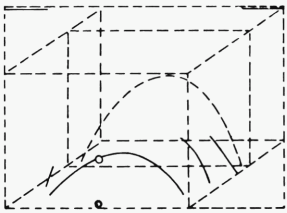


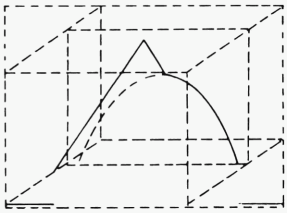
Après la coupe n°4



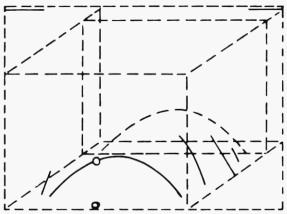


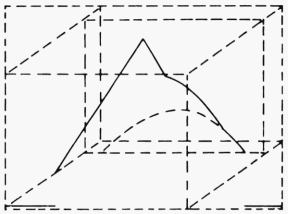
Après la coupe n°5



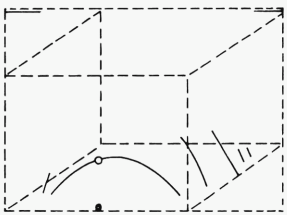


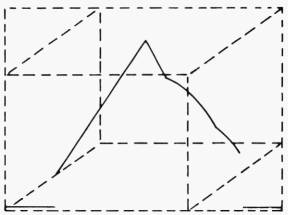
Après la coupe n°6





Après la coupe n°7 (pas de changement : le point est caché)





Programmation

Nous avons maintenant tous les outils indispensables pour rédiger le programme voulu. Nous l'avons scindé en trois sous-programmes :

- sous-programme **d'initialisation** (tableaux Hi et HS) ;
- sous-programme **d'introduction** des quantités N, Xi, XS, Yi, YS, Zi, ZS, DX, DY ;
- sous-programme **de tracé** proprement dit. Il comprend deux boucles :

  l'une sur le nombre de coupes (FOR i=1 TO N...)
  l'autre traçant chaque coupe (FOR J=Ø FO LG-(N-1)*DX).

  Ce sous-programme tient compte des parties vues et cachées et met à jour les tableaux Hi et HS.

```
20 REM SURFACE Z=F(X,Y)
30 LG=    HT=
40 DIM HI(LG),HS(LG)
100 GOSUB 1000 REM INITIALISATION DES TABLEAUX
110 GOSUB 1200 REM INTRODUCTION DES DONNEES
120 GOSUB 1500 REM TRACE
130 END
1000 REM INITIALIATION DES TABLEAUX
1010 FOR I=0 TO LG
1020 HI(I)=0:HS(I)=HT
1030 NEXT
1040 RETURN
1200 REM INTRODUCTION DES DONNEES
1210 REM MODE TEXTE,EFFACAGE D'ECRAN
1220 INPUT"NOMBRE DE COUPES";N
1230 IF N<=0 OR N<>INT(N) THEN 1220
1240 INPUT"ABSCISSE INFERIEURE";XI
1250 INPUT"ABSCISSE SUPERIEURE";XS
1260 IF XS<=XI THEN 1240
1270 INPUT"ORDONNEE INFERIEURE";YI
1280 INPUT"ORDONNEE SUPERIEURE";YS
1290 IF YS<=YI THEN 1270
1300 INPUT"COTE INFERIEURE";ZI
1310 INPUT"COTE SUPERIEURE";ZS
1320 IF ZS<=ZI THEN 1300
1330 INPUT"DECALAGE SUR X";DX
1340 IF DX<=0 OR DX<>INT(DX) OR DX>=LG/3/(N-1)
     THEN 1330
1350 INPUT"DECALAGE SUR Y";DY
1360 IF DY<=0 OR DY<>INT(DY) OR DY>=HT/3/(N-1)
     THEN 1350
1370 RETURN
1500 REM TRACE
1510 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
1520 L1=(LG-DX*(N-1))/(YS-YI)
1530 Z1=(HT-DY*(N-1))/(ZS-ZI)
1540 FOR I=1 TO N
1550 X=XS-((XS-XI)*(I-1)/(N-1))
1560 FOR J=0 TO LG-DX*(N-1)
1570 Y=YI+J/L1
1580 K=(I-1)*DX+J
1600 Z=
1610 IF Z<ZI OR Z>ZS THEN 1670
1620 M=Z1*(Z-ZI):H=INT(HT-(I-1)*DY-M+.5)
1630 IF H<HI(K) AND H>HS(K) THEN 1670
1640 IF H>HI(K) THEN HI(K)=H                    →
```

```
1650 IF H<HS(K) THEN HS(K)=H
1660 POINT K,H
1670 NEXT J
1680 NEXT I
1690 RETURN
```

Z=F(X,Y)	Yi	YS	Yi	YS	Zi	ZS
$Z=X^3Y^3$	-1	1	-1	1	-1	1
$Z=1-X^2Y^2$	-1	1	-1	1	0	1
$Z=X^2Y^2(1-Y^2)$	-1	1	-1	1	0	0,5
$Z=e^{-(X^2+Y^2)}$	-2	2	-2	2	0	1
$Z=1-\sqrt{X^2+Y^2}$	-1	1	-1	1	0	1
$Z=\frac{1}{1+X^2+Y^2}$	-2	2	-2	2	0	2
$Z=1-\sqrt{\lvert XY\rvert}$	-1	1	-1	1	0	1
$Z=-X\sqrt{\lvert Y\rvert}$	-1	1	-1	1	-1	1
$Z=\lvert XY\rvert$	-1	1	-1	1	0	1
$Z=X^2-Y^2$	-1	1	-1	1	-1	1
$Z=X^Y$	0,1	3	-1	1	0	3
$Z=e^{\sqrt{\lvert X^2Y\rvert}}$	-1	1	-1	1	0	3
$Z=X\ SIN(Y)$	-1	1	-6,28	6,28	-1	1
$Z=e^{SIN\ X\ SIN\ Y}$	-6,28	6,28	-6,28	6,28	0	4
$Z=(X+SIN(6Y))(Y+SIN(4X))$	-1	1	-1	1	-4	4
$Z=\frac{SIN(\sqrt{X^2+Y^2})}{\sqrt{X^2+Y^2}}$ (Z=1 si X=Y=0)	-15	15	-15	15	-0,3	1
$Z=e^{-Y/5}\ SIN(XY)$	-3,14	3,14	-5	5	-3	3
$Z=-\lvert 1-X^2-Y^2\rvert$	-2	2	-2	2	-3	0
$Z=1-\lvert SIN(\sqrt{X^2+Y^2})\rvert$	-6,28	6,28	-6,28	6,28	-2	2
$Z=1-e^{-XY^2}$	0	1	-1,5	1,5	0	1
$Z=-ATN(XY)$	-3	3	-3	3	-2	2
$Z=\frac{X^2-Y^2}{X^2+Y^2}$	-2	2	-2	2	-1	1
$Z=1-e^{(-Y^2/X)}$	-2	2	-2	2	0	1
$Z=X\times\frac{Y^2-2}{Y^4}$	-1	1	-3	3	-3	3
$Z=1-\lvert X-INT(X)-0,5\rvert-\lvert Y-INT(Y)-0,5\rvert$	0	2	-2	2	0	2
$Z=Log(X^2+Y^2)/(X^2+Y^2)$	-3	3	-3	3	-2	0,5
$Z=\lvert\sqrt{X^2+Y^2-1}\rvert\ \sqrt{\frac{2}{\sqrt{X^2+Y^2}}-1}$	-2	2	-2	2	0	2
$Z=SIN^2(\sqrt{X^2+Y^2})\times COS(2*\sqrt{X^2+Y^2})$	-3,14	3,14	-3,14	3,14	-1	1

→

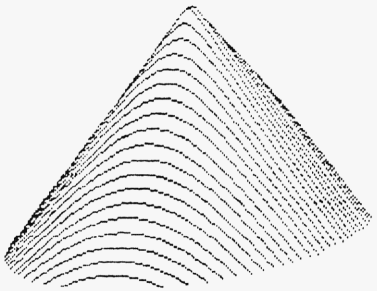
Z=F(X,Y)	Yi	YS	Yi	YS	Zi	ZS
X=X-INT(X)-0,5 Y=Y-INT(Y)-0,5 si $X^2+Y^2 > 0,25$ Z=1 sinon $Z=1-\sqrt{0,25-X^2-Y^2}$	0	2	-2	2	0	1
$Z=2-\sqrt{X^2+Y^2}$ si $X^2+Y^2 \leqslant 4$ $Z=1-\frac{2}{\sqrt{X^2+Y^2}}$ si $X^2+Y^2 \geqslant 4$	-4	4	-4	4	0	2

On pourrait penser que les deux tests IF H>Hi(K)... et H<HS(K)... seraient avantageusement regroupés en :

```
IF H>Hi(K) THEN Hi(K)=H ELSE HS(K)=H
```

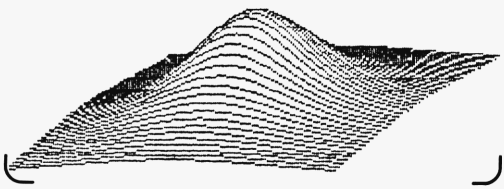
Si votre Basic le permet, testez cela avec une surface assez ondulée ($Z=SIN(\sqrt{X^2+Y^2})/(\sqrt{X^2+Y^2})$, par exemple) et constatez... Pour comprendre d'où viennent les problèmes, faites afficher les tableaux Hi et HS : ce ELSE ne conduit pas à une mise à jour correcte des tableaux Hi et HS !

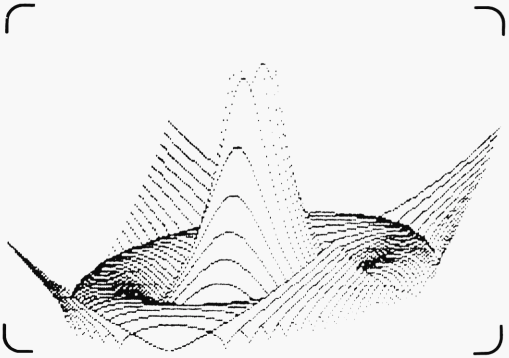
Si la fonction Z n'est pas toujours définie et si votre matériel le permet, rajoutez à ce programme un traitement d'erreur.



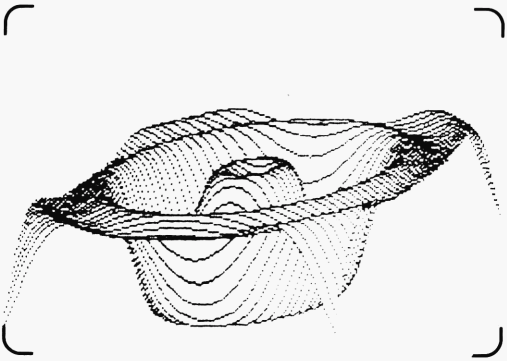
```
Z=1-SQR(X*X+Y*Y)
XI=-1 : XS=1
YI=-1 : YS=1
ZI= Ø : ZS=1
4Ø COUPES
```

```
Z=1/(1+X*X+Y*Y)
XI=-2 : XS=2
YI=-2 : YS=2
ZI= Ø : ZS=2
4Ø COUPES
```

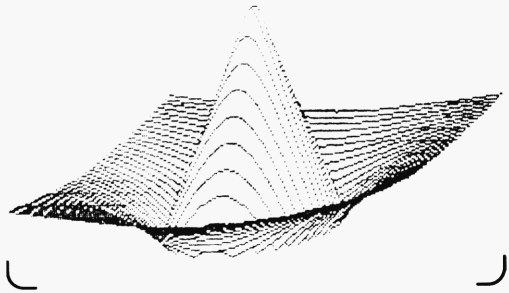




```
Z=ABS(SQR(X*X+Y*Y)-1)*SQR(ABS(2/SQR(X*X+Y*Y)-1))
XI=-2 : XS=2
YI=-2 : YS=2
ZI= Ø : ZS=2
4Ø COUPES
```

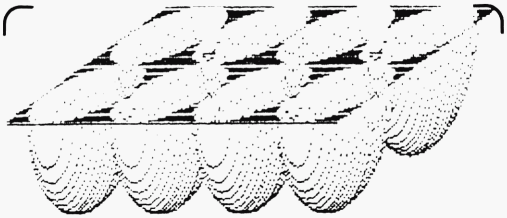


```
Z=(SIN(SQR(X*X+Y*Y)))^2*COS(2*SQR(X*X+Y*Y))
XI=-3.14 : XS=3.14
YI=-3.14 : YS=3.14
ZI=-1    : ZS=1
5Ø COUPES
```



```
Z=2-SQR(X*X+Y*Y) SI X*X+Y*Y<4
Z=1-2/SQR(X*X+Y*Y) SI X*X+Y*Y>=4
XI=-4 : XS=4
YI=-4 : YS=4
ZI= Ø : ZS=2
4Ø COUPES
```

```
X=X-INT(X)-.5 : Y=Y-INT(Y)-.5
Z=1-SQR(.25-X*X-Y*Y) SI X*X+Y*Y<=.25
Z=1 SI X*X+Y*Y>.25
XI= Ø : XS=2
YI=-2 : YS=2
ZI= Ø : ZS=1
5Ø COUPES
```

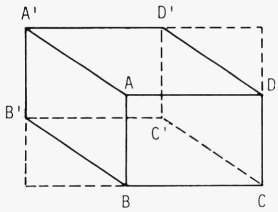


Petite bibliothèque de surfaces

Comme d'habitude, pour vous éviter de perdre trop de temps dans la recherche de bonnes surfaces Z=F(X,Y), de bonnes bornes Xi,XS,Yi,YS,Zi,ZS, nous vous livrons un bon nombre d'exemples. Prenez pour N un nombre de 25 à 40, cela suffit en général (choisir N trop grand allonge le temps d'exécution). Adaptez DX et DY à votre matériel et à vos goûts.

Quelques variantes

Arrivé à ce point, tout lecteur ayant suivi ce qui vient d'être expliqué devrait être capable d'adapter le programme au cas où le parallélépipède de base est représenté ainsi :



Il suffit de remplacer K=J+(i-1)*DX par K=J+(N-i)*DX. Des figures analogues à celles des pages 144 et 145 le montreront aisément. La modification étant minime, vous pouvez même faire choisir le type de représentation par l'utilisateur !

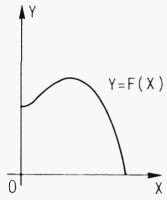
Une variante plus intéressante est de visualiser des coupes parallèles à XOY (ou ABA'B'). Nous allons rédiger cela pour le choix de représentation du parallélépipède de départ. Là, le choix a de l'importance : notre algorithme "partie vue, partie cachée" doit commencer par la coupe la plus proche de l'observateur (CDC'D' ou Y=YS). Dans l'autre cas de représentation, on commencerait par la coupe ABB'A' ou Y=Yi.

```
2000 REM COUPES PARALLELES A XOZ
2010 FOR I=1 TO N
2020 Y=YS-(I-1)*(YS-YI)/(N-1)
2030 FOR J=0 TO (N-1)*DX
2040 X=XS-J*(XS-XI)/((N-1)*DX)
2050 K=J-(N-I)*DX+INT((N-I)*LG/(N-1)+.5)
2100 Z=    :REM MEME EQUATION QU'EN LIGNE 1600
2110 IF Z<ZI OR Z>ZS THEN 2170
2120 M=ZI+(Z-ZI)*H=INT(HI-J*DY/DX-M+.5)
2130 IF H<HI(K) AND H>HS(K) THEN 2170
2140 IF H>HI(K) THEN HI(K)=H
2150 IF H<HS(K) THEN HS(K)=H
2160 POINT K,H
2170 NEXT J
2180 NEXT I
2190 RETURN
```

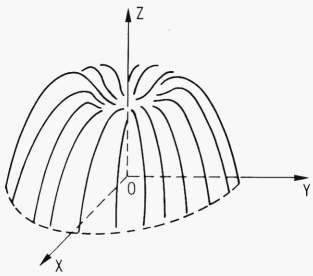
Si vous ajoutez ces coupes aux précédentes, n'oubliez surtout pas, auparavant, de réinitialiser les tableaux Hi et HS (par un GOSUB INITIALISATION DES TABLEAUX).

Une autre variante intéressante consiste à tracer des surfaces de révolution obtenues en faisant tourner une courbe autour d'un axe. Supposons que

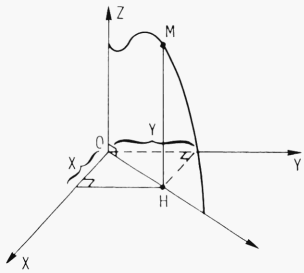
la courbe ait pour équation Y=F(X) et qu'on la fasse tourner autour de OY :



donne



(a priori, les X et Y du plan et ceux de l'espace ne sont pas les mêmes). En remarquant que tout plan passant par OZ coupe la surface selon la coupe de départ, on comprend que :



$OH = \sqrt{X^2+Y^2}$ (Pythagore)

HM = F(OH) (F : fonction de départ)

et la surface a pour équation :

$Z = F(\sqrt{X^2+Y^2})$

Si on s'intéresse seulement à la partie de courbe Y=F(X) correspondant à $0 \leqslant X \leqslant T$, on aura intérêt à prendre :

```
Xi = -T    XS = T
Yi = -T    YS = T
```

et à modifier le programme donné.

Les lignes 1240 à 1290 (introduction des bornes Xi,XS,Yi,YS) sont à remplacer par les deux suivantes :

```
1240 INPUT "BORNE SUR X" ; T
1250 IF T< =0 THEN 1240
```

Il faut aussi rajouter :

```
1590 R=SQR(X*X+Y*Y):IF R> T THEN 1670
```

justement, la ligne 1590 n'était pas utilisée !

En ligne 1600, il convient d'écrire Z = la fonction étudiée. Cette fonction ne doit pas dépendre des variables X et Y, mais de la variable R. Faire dépendre Z de R est fondamental puisque, en ligne 1590, c'est cette variable R qui représente $OH = \sqrt{X^2+Y^2}$.

Ainsi, vous comprendrez mieux comment nous avons obtenu certaines surfaces du tableau joint. Pour toutes celles qui comportent X^2+Y^2, nous n'avons

fait que tourner autour de OY une courbe Y=F(X), puis nous avons pris $Z=F(\sqrt{X^2+Y^2})$. Vous pourrez ainsi faire de nombreux autres essais... Toutes les courbes d'équation Y=F(X) du chapitre I (deuxième partie) vous prendront déjà un moment. Dans ce cas, Zi est le minimum de F, ZS son maximum.

Dans le tableau, vous avez aussi certainement remarqué des dénominateurs qui risquent de s'annuler. Si vous souhaitez éviter tout ennui et si votre matériel le permet, ajoutez un traitement d'erreur ON ERROR GOTO... Le programme doit reprendre à la ligne NEXT J après l'interception de l'erreur.

Quand vous disposerez d'un bon catalogue de surfaces, toutes plus jolies les unes que les autres, vous pourrez rédiger le programme qui gèrera le fichier de ces surfaces ou celui qui vous fournira une recopie d'écran sur imprimante, ou celui qui tracera les surfaces en langage machine... car le Basic s'avère un peu lent. Pour ce dernier programme, contactez-nous, il nous intéresse vivement.

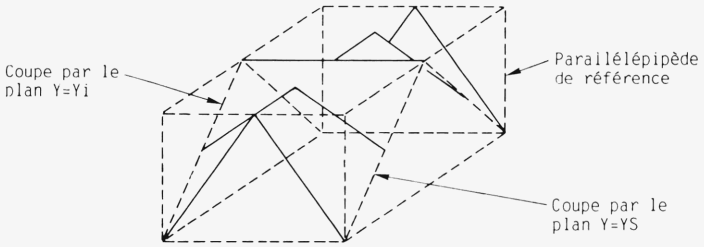
Avant d'étudier des surfaces un peu plus compliquées, autant vous signaler quelques insuffisances de l'algorithme utilisé. Vous les avez peut-être déjà constatées en utilisant ce programme.

Première insuffisance : des points sont affichés alors que, normalement, ils devraient être cachés. Ainsi pour :

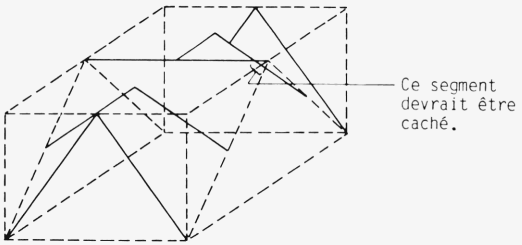
Z=1-|XY| Xi=Yi=-1 XS=YS=ZS=1 Zi=Ø

toutes les coupes sont formées par deux segments de droites.

Pour cinq coupes, on devrait obtenir :



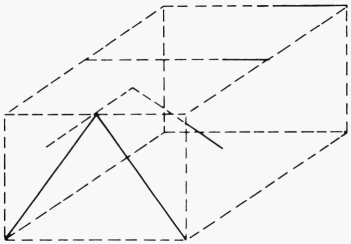
alors qu'on obtient à l'écran :



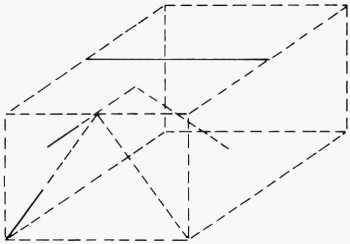
Voyez-vous l'anomalie ? Elle est masquée par les tracés en pointillé du parallélépipède de référence et des coupes par les plans Y=Yi et Y=YS. Comme elle est d'autant moins perceptible que N est grand, vous ne l'avez peut-être même pas remarquée. Elle est manifeste pour N petit (5 à 9), DY petit lorsqu'on visualise la coupe par le plan Y=YS. Pour que vous ne cherchiez pas trop longtemps, nous l'avons signalée.

D'où provient cette anomalie ? Reprenons la gestion de nos lignes de crêtes. A la fin de la troisième coupe, pour N=5, nous obtenons :

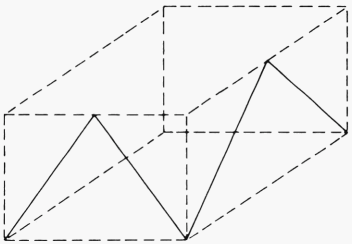
Ligne de crête minimale



Ligne de crête maximale

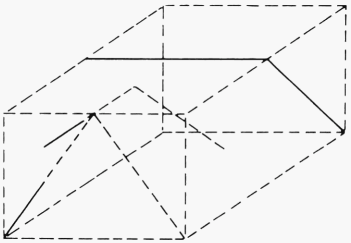


Nos ennuis viennent des initialisations des lignes de crêtes. Au moment où nous avons introduit les tableaux Hi et HS (page 148), volontairement, nous n'avons soulevé aucune difficulté. De même que, lors d'études de maxima et de minima de fonctions, au départ, on prend pour maximum -1E 30 et pour minimum 1E 30, nous avons initialisé les tableaux Hi et HS à l'inverse de leur sens : première et dernière ligne de l'écran. Il eut été plus judicieux d'initialiser Hi et HS ainsi, pour notre exemple :

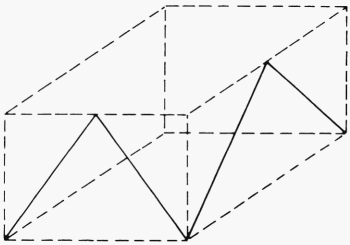


Avec un tel choix, à la fin de la troisième coupe, pour N=5, nous aurions obtenu :

Ligne de crête maximale



Ligne de crête minimale



et la partie de courbe, source d'anomalie, aurait été comprise entre les lignes de crêtes maximales et minimales et n'aurait donc pas été tracée.

De façon générale, pour le parallélépipède tel que nous l'avons représenté :

pour i variant de Ø à LG-DX*(N-1), Hi et HS correspondent à la coupe X=Xi ;

pour i variant de LG-DX*(N-1) à LG, Hi et HS correspondent à la coupe Y=YS

le sous-programme correspondant n'est pas très compliqué à rédiger.

Pour notre part, nous considérons que le programme donné est suffisant, à condition de savoir d'où proviennent d'éventuelles anomalies.

Une autre anomalie peut être rencontrée pour des fonctions Z bizarres, disons non continues. Par exemple :

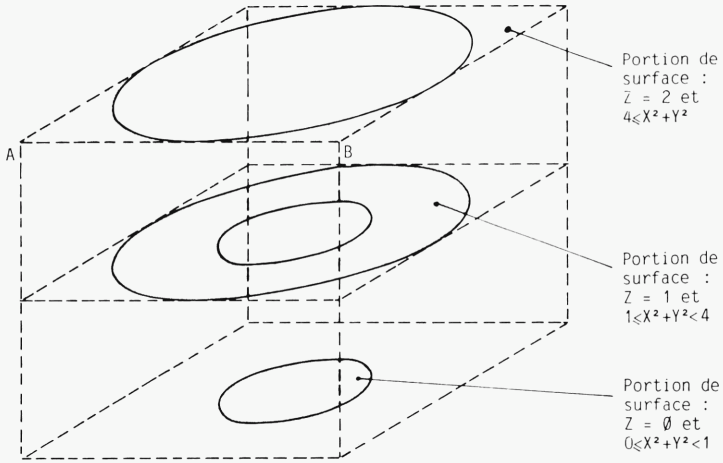
$Z=INT(\sqrt{X^2+Y^2})$ avec Xi=Yi=-2 XS=YS=2 Zi=Ø ZS=2

D'accord, pour choisir de telles fonctions, il faut avoir un "p'tit grain" ! Et encore, si vous saviez ce que nos ordinateurs ont eu à subir ! Ce ne sont d'ailleurs pas eux mais nos épouses qui ont le plus souffert... mais c'est une autre histoire.

Pour notre surface donc :

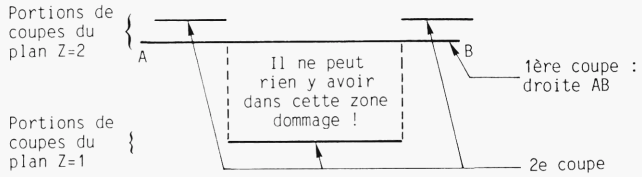
$$Z = \begin{cases} 0 & \text{si} \quad 0 \leqslant X^2+Y^2 < 1 \\ 1 & \text{si} \quad 1 \leqslant X^2+Y^2 < 4 \\ 2 & \text{si} \quad 4 \leqslant X^2+Y^2 \end{cases}$$

Notre surface devrait donc avoir l'allure suivante :



puisqu'une condition du type $X^2+Y^2 \leqslant R^2$ traduit, dans un plan, l'intérieur d'un cercle de rayon R.

A l'écran, la première coupe nous fournit normalement la droite AB, puis la seconde nous donne :



D'après notre algorithme, il ne peut y avoir aucun point dans la zone signalée. Laissez le programme tourner : la surface obtenue à l'écran n'a pas grand rapport avec ce qu'on attendait : seule la portion correspondant à Z=2 est correcte ; la couronne (Z=1) et le disque (Z=Ø) sont incomplets !

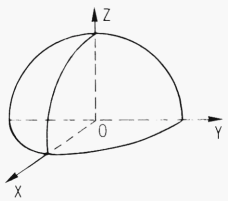
La seule solution, dans ce cas, est de ne pas tenir compte des parties vues et cachées : notre algorithme ne tient pas compte des cas où Z passe brusquement d'une valeur à une autre (fonctions non continues). Pour la même raison, des dénominateurs ou des logarithmes qui s'annulent peuvent conduire à des tracés inattendus.

Un programmeur averti en vaut deux : vous pouvez maintenant représenter toutes les surfaces Z=F(X,Y) imaginables et même interpréter d'éventuelles bizarreries.

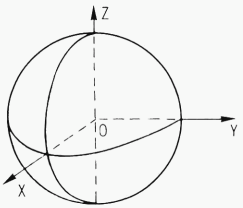
SURFACES EN Z^2

Nous venons de tracer des surfaces en tenant compte des parties vues et cachées même si, parfois, l'algorithme donné présente des insuffisances. Nous savons donc représenter une demi-sphère d'équation :

$$Z = \sqrt{R^2-X^2-Y^2}$$ (R : rayon)

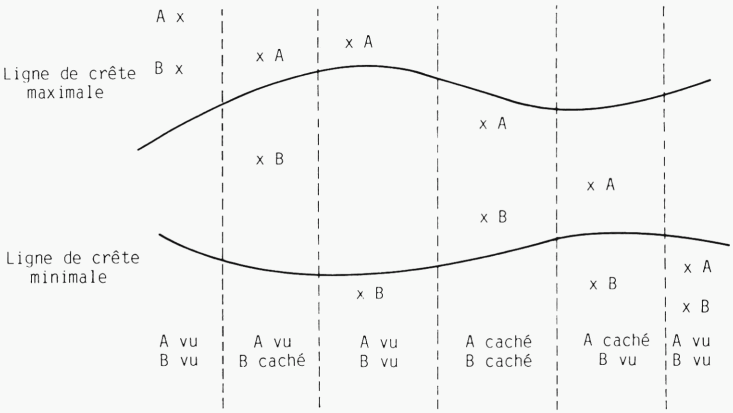


mais pas encore une sphère d'équation $Z^2+X^2+Y^2=R^2$.



Cela peut paraître bizarre, mais c'est ainsi ! La difficulté, dans le cas de la sphère, vient du fait que, pour X et Y donnés, Z prend deux valeurs.

On pourrait reprendre l'algorithme des lignes de crête maximale et minimale et le modifier. Soit A et B les points dont les cotes sont les deux valeurs de Z. Comme on peut toujours s'arranger pour que A soit le point de cote la plus grande, on aurait :



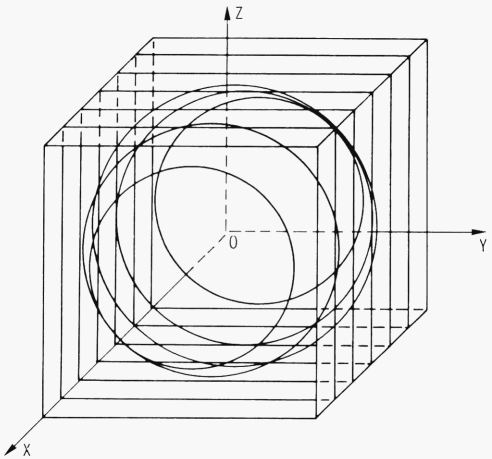
Une fois les points vus affichés, les lignes de crêtes seraient mises à jour.

Une telle méthode est envisageable mais elle risque d'être bien lourde : nombreux tests... et nous n'avons pas encore envisagé les cas où l'un des points A ou B est en dehors du parallélépipède de référence !

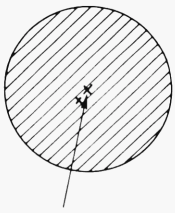
De toute façon, il y a plus simple.

La méthode utilisée

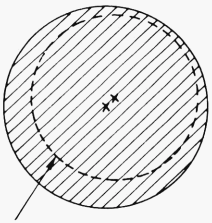
Considérons notre sphère. Fixons son rayon : 1. Ces coupes, par des plans parallèles à YOZ, sont des cercles centrés sur OX :



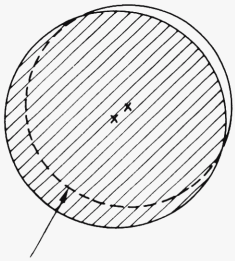
Si, comme nous l'avons fait pour le cône, nous traçons ces coupes en commençant par celle qui est la plus proche de l'observateur (X=1 ici, X=XS dans le cas général), nous n'aboutirons à rien. Au contraire, commençons par la coupe la plus éloignée de l'observateur (X=-1 ici, X=Xi dans le cas général).



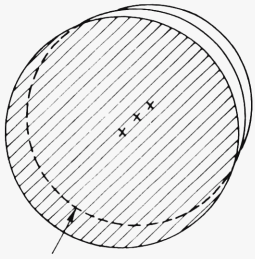
La première coupe (un point) est cachée par la seconde.



La seconde coupe est elle-même cachée par la troisième



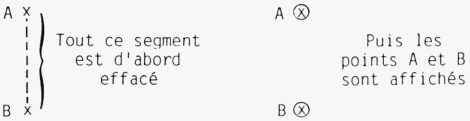
Partie de la troisième coupe cachée par la quatrième



Partie de la quatrième coupe cachée par la cinquième

Il n'est peut-être pas indispensable de poursuivre jusqu'à la dernière coupe : tout cercle cache les points situés derrière lui.

De façon générale si, pour des valeurs de X et Y, Z prend deux valeurs correspondant à deux points A et B, on procèdera ainsi : en partant de la coupe la plus éloignée de l'observateur (X=Xi), on effacera tout le segment AB, puis on visualisera les points A et B.



Il faut bien réaliser qu'à l'écran les points A et B ayant même abscisse et même ordonnée (ils ne diffèrent que par leur cote), seront sur une même verticale.

Cette méthode s'avère beaucoup plus simple que celle que nous avons évoquée plus haut.

Programmation

Au niveau programmation, que nous faut-il ?

- Toujours notre parallélépipède de référence : c'est du déjà vu.
- Il nous faut préciser la façon dont nous obtiendrons les deux points correspondant aux deux valeurs de Z.

Le plus simple, pour que Z puisse prendre deux valeurs, est que Z soit solution d'une équation du second degré, du type :

$$A\,Z^2 + BZ + C = 0$$

où A, B, C sont, bien entendu, des fonctions de X et Y.

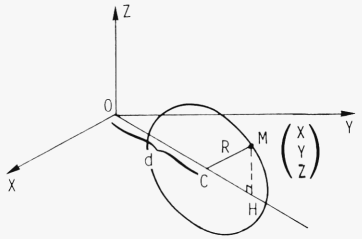
Pour notre sphère de rayon 1 par exemple :

$$Z^2 + X^2 + Y^2 - 1 = 0$$

donc $A=1$, $B=0$, $C=X^2+Y^2-1$.

Prenons pour autre exemple celui d'un tore, surface engendrée par un cercle tournant autour d'un axe. Ce cercle variable aura pour rayon R ;

son centre C variera sur un cercle de centre O, de rayon d, dans le plan XOY. Enfin, un tore, c'est une chambre à air sans valve.



On a :

$M\begin{pmatrix}X\\Y\\Z\end{pmatrix}$ $H\begin{pmatrix}X\\Y\\0\end{pmatrix}$

$HM = Z$

$CH = \sqrt{X^2+Y^2} - d$

$CM^2 = CH^2 + HM^2$ (Pythagore)

Donc :

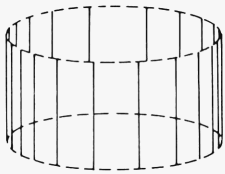
$Z^2 + (\sqrt{X^2+Y^2} - d)^2 = R^2$

d'où A=1, B=0, $C=(\sqrt{X^2+Y^2}-d)^2-R^2$.

La discussion de l'équation du second degré $AZ^2+BZ+C=0$ est très classique : nous n'y reviendrons pas. Précisons que le cas A=B=0 a été écarté au sens où le programme passe alors aux points suivants. La raison d'un tel choix est simple à comprendre. Si $C<>\emptyset$, l'équation est impossible : on continue. Par contre, si C=0, Z peut prendre toutes les valeurs comprises entre Zi et ZS : on doit donc écrire ZA=ZS et ZB=Zi. Mais A, B, C sont calculés avec des erreurs et le cas théorique A=B=C=0 risque pratiquement de ne pas se produire. Un exemple simple, celui d'un cylindre d'axe de révolution OZ, d'équation :

$$X^2+Y^2-1=0$$

(donc A=B=0, $C=X^2+Y^2-1$) illustrera mieux cette situation. Les différentes coupes sont des droites parallèles à OZ, représentées par des segments. A l'écran, on devrait donc obtenir :

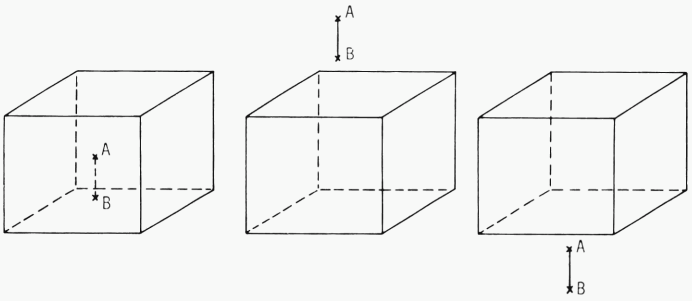


Mais, pour obtenir exactement cela, il faut que, pour toutes ces droites, X^2+Y^2-1 soit exactement 0. La moindre différence et rien n'est tracé à l'écran. Autant dire que ce qui sera tracé à l'écran a peu de chance de correspondre à ce que l'on souhaite...

Bien sûr, ce choix d'écarter le cas A=B=0 est arbitraire : vous pourrez aisément le rectifier s'il ne vous convient pas !

- Il nous faut revenir sur le cas où Z prend deux valeurs ZA et ZB (avec $ZA \geqslant ZB$) : il mérite plus d'attention.

En effet, par rapport aux bornes Zi et ZS définissant le parallélépipède, plusieurs cas de figure sont possibles :

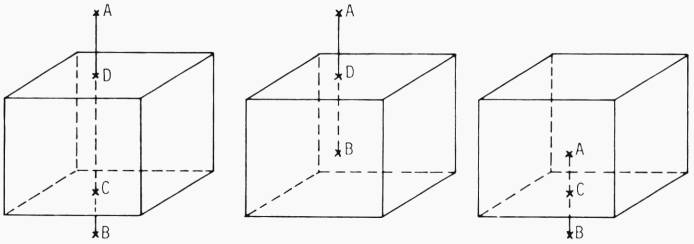


A et B intérieurs au parallélépipède : on utilise la méthode vue.

A et B au-dessus du parallélépipède

A et B au-dessous du parallélépipède

on continue



A au-dessus, B au-dessous du parallélépidède

A au-dessus, B à l'intérieur du parallélépidède

A à l'intérieur, B au-dessous du parallélépipède

Détaillons ces trois derniers cas : les deux points A et B ont pour coordonnées :

$$A\begin{pmatrix} X \\ Y \\ ZA \end{pmatrix} \quad B\begin{pmatrix} X \\ Y \\ ZB \end{pmatrix} \quad \text{dans l'espace}$$

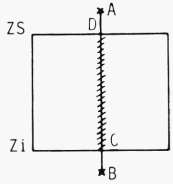
et sont représentés par :

$$A\begin{pmatrix} K \\ HA \end{pmatrix} \quad B\begin{pmatrix} K \\ HB \end{pmatrix} \quad \text{à l'écran}$$

Le sens de K, la façon dont on calcule H, ont été étudiés au début de ce chapitre. Nous avons aussi vu (page 144) comment se place la énième coupe sur l'écran.

Donc si :

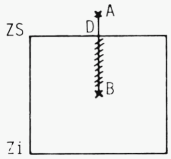
ZA > ZS et ZB < Zi



on efface sur l'écran la droite joignant les points :

$$C\ (^{K}_{HT-(i-1)\ DY}) \quad \text{et} \quad D\ (^{K}_{(N-i)\ DY})$$

ZA > ZS et Zi ⩽ ZB ⩽ ZS

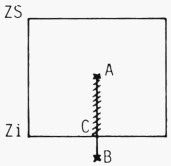


on efface sur l'écran la droite joignant les points :

$$B\ (^{K}_{HB}) \quad \text{et} \quad D\ (^{K}_{(N-i)\ DY})$$

on affiche le point B.

Zi ⩽ ZA ⩽ ZS et ZB < Zi



on efface sur l'écran la droite joignant les points :

$$A\ (^{K}_{HA}) \quad \text{et} \quad C(^{K}_{HT-(i-1)\ DY})$$

on affiche le point A.

Pour faciliter la compréhension du programme, le sous-programme de tracé proprement dit fait lui-même appel à deux sous-programmes : l'un traite le cas le plus simple (A et B tous deux intérieurs au parallélépipède), l'autre traite les trois derniers cas signalés.

```
30 REM SURFACES EN Z2
40 LG=   :HT=
110 GOSUB 1200:REM INTRODUCTION DES DONNEES
120 GOSUB 1500:REM TRACE
150 END
1200 REM INTRODUCTION DES DONNEES
1210 REM MODE TEXTE,EFFACAGE D'ECRAN
1220 INPUT"NOMBRE DE COUPES";N
1230 IF N<=0 OR N<>INT(N) THEN 1220
1240 INPUT"ABSCISSE INFERIEURE";XI
1250 INPUT"ABSCISSE SUPERIEURE";XS
1260 IF XS<=XI THEN 1240
1270 INPUT"ORDONNEE INFERIEURE";YI
1280 INPUT"ORDONNEE SUPERIEURE";YS
1290 IF YS<=YI THEN 1270
1300 INPUT"COTE INFERIEURE";ZI
1310 INPUT"COTE SUPERIEURE";ZS
1320 IF ZS<=ZI THEN 1300                    →
```

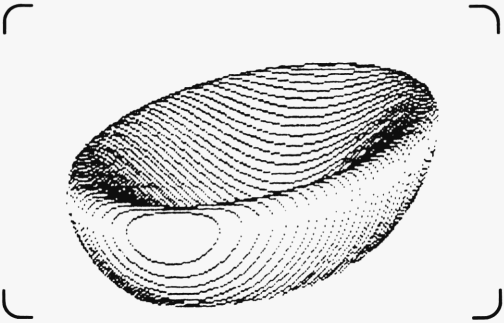
```
1330 INPUT"DECALAGE SUR X";DX
1340 IF DX<=0 OR DX<>INT(DX) OR DX>=LG/(N-1)
     THEN 1330
1350 INPUT"DECALAGE SUR Y";DY
1360 IF DY<=0 OR DY<>INT(DY) OR DY>=HT/(N-1)
     THEN 1350
1370 RETURN
1500 REM TRACE
1510 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
1520 L1=(LG-DX*(N-1))/(XS-XI)
1530 Z1=(HT-DY*(N-1))/(ZS-ZI)
1540 FOR I=N TO 1 STEP -1
1550 X=XS-((XS-XI)/(I-1)*(N-1))
1560 FOR J=0 TO LG-DX*(N-1)
1570 Y=YI+J/L1
1580 K=(I-1)*DX+J
1600 A=   :B=   :C=   :
     REM COEFFICIENTS DE L'EQUATION DONNANT Z
1610 D=B*B-4*A*C
1620 IF A=0 OR D<0 THEN 1700
1630 D=SQR(D):ZA=(-B-D)/2/A:ZB=(-B+D)/2/A
1640 IF ZA<ZB THEN Z=ZA:ZA=ZB:ZB=Z
1650 IF ZA<ZI OR ZB>ZS THEN 1700
     REM A ET B EXTERIEURS AU PARALLELEPIPEDE
1660 HA=INT(HT-(I-1)*DY-ZI)*(ZA-ZI)+.5)
1670 HB=INT(HT-(I-1)*DY-ZI)*(ZB-ZI)+.5)
1680 IF ZA<=ZS AND ZB>=ZI THEN GOSUB 2000 :
     GOTO 1700 :REM REM A ET B INTERIEURS AU
     PARALLELEPIPEDE
1690 GOSUB 2500:REM UN SEUL POINT INTERIEUR
     AU PARALLELEPIPEDE
1700 NEXT J
1710 NEXT I
1720 RETURN
2000 REM A ET B TOUS DEUX VUS
2010 DROITE K,HA A K,HB EFFACE
2020 POINT K,HA:POINT K,HB
2030 RETURN
2500 REM UN SEUL POINT EST VU
2510 IF ZA>ZS AND ZB<ZI THEN DROITE K,(N-I)*DY
     A K,HT-(I-1)*DY EFFACE
2520 IF ZA>ZS AND ZB>ZI THEN DROITE K,(N-I)*DY
     A K,HB EFFACE:POINT  K,HB
2530 IF ZA<ZS AND ZB<ZI THEN DROITE K,HA
     A K,HT-(I-1)*DY EFFACE:POINT K,HA
2540 RETURN
```

A	B	C	Xi	XS	Yi	YS	Zi	ZS
1	0	$-X^2-Y^2$	-1	1	-1	1	-1	1
1	[-2 COS X]	Y^2	-3,14	3,14	-2	2	-2	2
1	0	$Y^2-X^2(1-X^2)$	-1	1	-1	1	-1	1
1	0	$(1-X^2)(1-Y^2)(X^2-Y^2)$	-1	1	-1	1	-1	1
1	0	$(1-X^2)(1-Y^2)(Y^2-X^2)$	-1	1	-1	1	-1	1
1	0	$(X^2-1)\sqrt{1-\sqrt{\lvert Y\rvert}}$	-1	1	-1	1	-1	1
1	2XY	X^2+Y^2-1	-1 -2	1 2	-1 -2	-1 +2	-1 -2	1 +2
1	[2(1-X²-Y²)]	$1-3(X^2+Y^2)+2(X^2+Y^2)^2$	-1	1	-1	1	-1	1
1	0	$Y^2(1-Y^2)(1-X^2)$	-1,4	1,4	-1,4	1,4	-2	2
X^2+Y^2	0	X^4+Y^4-1	-1	1	-1	1	-3	3
1	SIN(XY)+SIN(X-Y)	SIN(XY) x SIN(X-Y)	-6,3	6,3	-6,3	6,3	-5	5
$(X^2+Y^2)^2$	$(X^2+Y^2)(X^2-Y^2+2XY)$	$2XY(X^2-Y^2)$	-3	3	-3	3	-2	2
1	0	$\frac{(\sqrt{X^2+Y^2-1})^2(3-\sqrt{X^2+Y^2})}{1+\sqrt{X^2+Y^2}}$	-3	3	-3	3	-3	3
X^2+Y^2	[-2XY]	[X^2Y^2]	-1	1	-1	1	-1	1
$4X^2+Y^2$	0	$-4(X^2+Y^2)(1-X^2-Y^2)$	-1	1	-1	1	-2	2
X^2+4Y^2	0	$-4(X^2+Y^2)(1-X^2-Y^2)$	-1	1	-1	1	-2	2

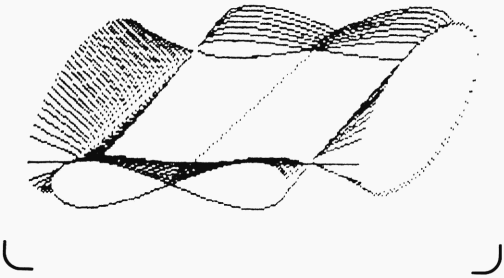
L'instruction d'effaçage d'une droite est à adapter à votre matériel : c'est sans doute un affichage de la droite dans la couleur du fond de l'écran. Si notre choix de passer au point suivant lorsque A=0 ne vous convient pas, modifiez la ligne 162Ø... Si les fonctions A, B, C ne sont pas toujours définies, rajoutez un traitement d'erreurs.

Exemples

Comme d'habitude, nous vous joignons un catalogue d'exemples.



```
A=1 : B=2*(1-X*X-Y*Y) : C=1-3*(X*X+Y*Y)+2*(X*X+Y*Y)^2
XI=-1 : XS=1
YI=-1 : YS=1
ZI=-1 : ZS=1
5Ø COUPES
```

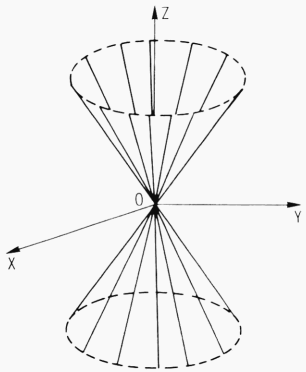


```
A=1 : B=Ø : C=Y*Y*(1-Y*Y)*(1-X*X)
XI=-1.4 : XS=1.4
YI=-1.4 : YS=1.4
ZI=-2   : ZS=2
5Ø COUPES
```

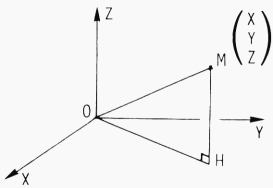
Les mises en garde vues pour les surfaces du type Z=F(X,Y) sont encore valables. Nous devons en ajouter une autre. Prenons :

$A = 1 \quad B = 0 \quad C = -X^2 - Y^2$

d'où $Z^2 = X^2 + Y^2$. Il s'agit de l'équation d'un cône de révolution d'axe OZ, de sommet O, d'angle 45° :



En effet, on a (H : projection de M sur OXY) :



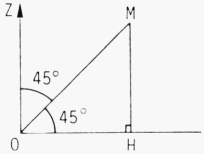
$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$ avec $Z^2 = X^2 + Y^2$

$OH^2 = X^2 + Y^2$

$MH^2 = Z^2$

donc MH = OH

Le triangle rectangle OHM est isocèle :

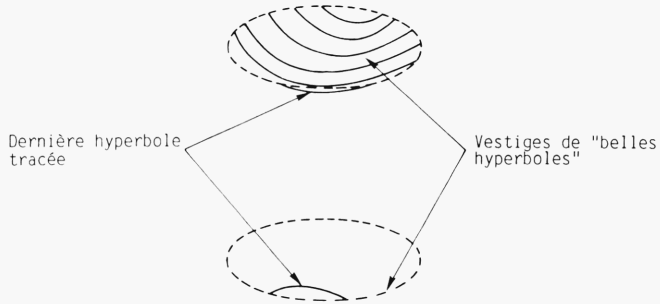


$\widehat{HOM} = 45°$

La droite OM fait donc un angle constant, de 45° avec OZ.

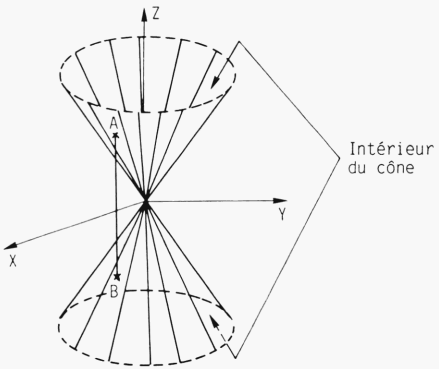
Avec Xi=-1, XS=Ø, Yi=-1, YS=1, Zi=-1, ZS=1, tout se passe bien : les sections du cône considéré sont de belles hyperboles ; pour X=Ø, on obtient même deux droites.

Par contre, en prenant XS=1, toutes les autres quantités ayant les mêmes valeurs, nos belles hyperboles s'effacent progressivement pour donner :



Le mieux est de le constater sur votre écran.

Que se passe-t-il ? Pour le comprendre, réfléchissons à notre méthode : nous effaçons le segment AB puis nous affichons les points A et B. Cela suppose que le segment AB soit intérieur à la surface. L'exemple d'introduction choisi, une sphère le confirme bien. Or, pour notre cône, la situation est totalement différente :

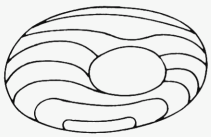


Tous les segments AB que nous effaçons sont extérieurs au cône ! A partir d'un certain moment, l'effaçage affecte des parties de la surface qui devraient être vues !

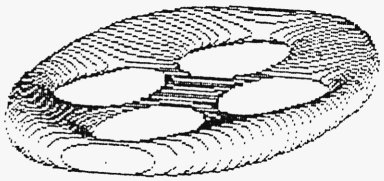
Que faire alors ? Première solution : modifier complètement l'algorithme (ce qui est entre A et B est vu, ce qui est au-dessus de A ou au-dessous de B est caché). Ce n'est pas très compliqué : tout ce qui est utile figure plus haut. Dans ce cas, les sphères, les tores... vont avoir une drôle d'allure ! Seconde solution : n'utiliser ce programme que si AB est intérieur à la surface. C'est ce que nous préférons. Mais ne vous fiez pas trop à votre intuition : les notions d'intérieur et d'extérieur ne sont pas aussi simples qu'il paraît. Souvenez-vous des fractals et des lignes qui finissent par remplir des surfaces... S'il arrivait la même mésaventure à l'intérieur et à l'extérieur ?

Pour tous les exemples joints, l'algorithme utilisé donne de bons résultats. Oui, oui, nous avons même choisi des exemples qui marchent ! Nous avons déjà évoqué le tore (page 164). Nous souhaitons faire une place à part, dans cette modeste galerie, aux tores percés ! Qu'est-ce qu'ils ne vont pas chercher, ces matheux ! Des enveloppes, des épi, des hypo, des systèmes différentiels. Et maintenant, ils percent les tores !!

Imaginons donc une galette, de forme circulaire, un peu aplatie, et perçons-la :

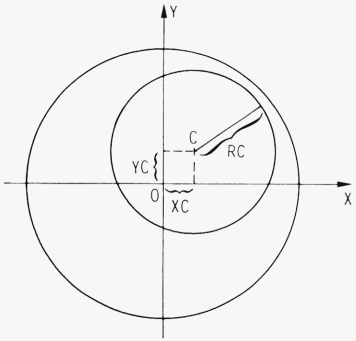


Tore à 1 trou



Tore à 4 trous

La base de notre tore, dans le plan XOY, est un cercle de centre O, de rayon 1, d'équation $X^2+Y^2-1=0$:



Perçons un trou. Dans ce plan XOY, ce trou est un cercle de centre C, de coordonnées XC,YC et de rayon R_C. L'équation du tore tracé correspondant est :

$$Z^2+(X^2+Y^2-1)((X-XC)^2+(Y-YC)^2-RC^2) = 0$$

donc $A=1$, $B=0$, $C=(X^2+Y^2-1)[(X-XC)^2+(Y-YC)^2-RC^2]$.

Vous voulez ajouter un autre trou correspondant à un autre cercle de centre D(XD,YD), de rayon RD ? Rien de plus simple, son équation est :

$$Z^2+(X^2+Y^2-1)[(X-XC)^2+(Y-YC)^2-RC^2][(X-XD)^2+(Y-YD)^2-RD^2] = 0$$

Vous avez compris le principe : au départ, $C=X^2+Y^2-1$ (sans trou, l'équation se réduit à $Z^2+X^2+Y^2-1 = 0$: une vulgaire sphère). A chaque fois que l'on ajoute un trou, on multiplie C par :

(X-abscisse centre)²+(Y-ordonnée centre)²-rayon²

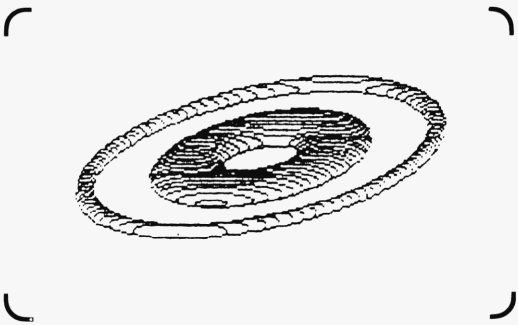
et le trou est creusé.

Bien entendu, Xi=Yi=Zi=-1, XS=YS=ZS=1. Si le choix d'un rayon égal à 1 pour le cercle de base vous gêne (XC,YC sont compris entre 0 et 1 et élever RC au carré peut être délicat à la main !), vous pouvez prendre n'importe quel nombre R, mais alors Xi=Yi=Zi=-R, XS=YS=ZS=R, XC, YC et RC devant vérifier :

$$XC^2+YC^2 < R^2$$
$$\text{et } XC^2+YC^2 \leqslant (R-RC)^2$$

pour que le trou soit bien intérieur au tore !

A elle seule, la nombreuse famille des tores percés valait bien le détour ! Et malgré une mise en garde, osez, si vous ne l'avez déjà fait, percer un tore par un trou assez grand, disons empiétant sur l'extérieur ($C=(X^2+Y^2-1)(X^2+(Y-0,5)^2-1)$ par exemple). Et voilà qu'en creusant un trou à l'extérieur d'un tore, on se retrouve à l'intérieur ! On vous avait prévenu : s'il arrivait une mésaventure à l'intérieur et à l'extérieur. Le mieux, dans l'exemple cité, est d'ailleurs de prendre YS=1,5, les choses s'éclairent un peu, à vous de les découvrir.



```
A=1 : B=Ø : C=(X*X+Y*Y-1)*(X*X+Y*Y-.81)*(X*X+Y*Y-.36)*(X*X+Y*Y-.Ø4)
XI=-1 : XS=1
YI=-1 : YS=1
ZI=-1 : ZS=1
5Ø COUPES
```

Chapitre II
Manipulations d'objets

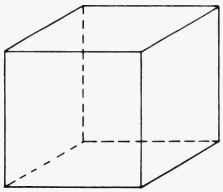
Laissons maintenant les surfaces et intéressons-nous à des manipulations d'objets en dimension 3. Que recouvrent ces deux termes "**manipulations**" et "**objets**" ?

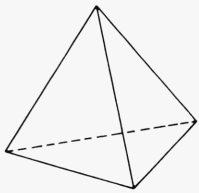
Par **manipulations**, nous entendons les possibilités suivantes :

- faire tourner un objet comme nous le voulons ;
- le regarder de n'importe quel point de l'espace, sous n'importe quel angle.

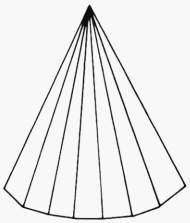
Dans la pratique, ces possibilités correspondent à deux attitudes. La première est illustrée par le Rubik-cubiste. Il reste à peu près fixe et effectue des rotations sur les faces du cube. Une personne prenant une photo illustre la seconde : elle cherche une bonne place, compte tenu de critères dont nous ne discuterons pas et choisit un type d'objectif approprié (normal, téléobjectif pour rapprocher l'objet ou photographier un détail, grand angle pour couvrir la plus grande étendue possible). Nous détaillerons un peu plus loin.

Par **objets**, nous entendons n'importe quoi ou presque... Certains lecteurs ironisent déjà. Ah, la belle définition ! Nous entendons n'importe quel solide défini par des arêtes. Pour un cube, une pyramide, c'est simple. Par construction même, ils sont constitués de sommets reliés par des arêtes :

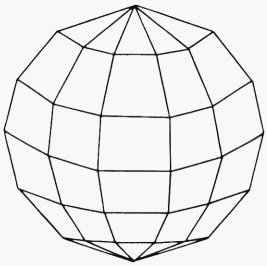




Un cône ou une sphère ne peuvent pas, a priori, entrer dans cette catégorie. Pourtant, un ballon de football est considéré comme sphérique... bien qu'il soit réalisé à partir d'hexagones et de pentagones réguliers ! La plupart des surfaces peuvent être considérées comme formées de facettes assez petites. Comme chaque facette est elle-même délimitée par des segments, la définition d'un solide par des arêtes est assez générale.



Cône à facettes



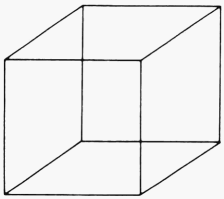
Sphère à facettes

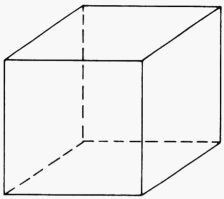
Cette présentation pose déjà une question. Que faut-il privilégier : les facettes, les arêtes ou même les sommets (extrémités d'arêtes) ? Nous verrons qu'il n'y a pas de réponse valable dans tous les cas. C'est le type d'application envisagée qui conduit à choisir les arêtes de préférence aux faces, ou le contraire.

Voilà pour les objets et les types de manipulations que nous envisagerons. Mais comment représenterons-nous l'objet sur l'écran de notre ordinateur ? Il existe de nombreuses façons de visualiser, dans un plan, un objet qui, lui, est dans l'espace. Citons, par exemple, les perspectives cavalières, les projections avec vue de face, de dessus, de droite... chères au dessinateur industriel, les perspectives à point de fuite, point pouvant être au pluriel ! Plus tous les cas que nous oublions.

Ainsi, un simple cube peut être représenté :

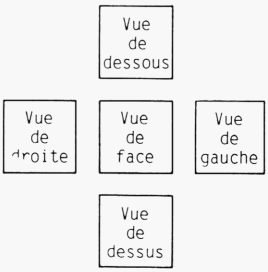
- *en perspective cavalière*



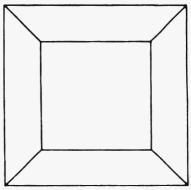


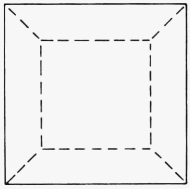
(arêtes cachées en pointillé)

- *en dessin industriel*



- en perspective à un point de fuite (et vu de face)





Récapitulons : pour un objet donné, nous avons deux types de manipulations, au moins trois types de représentations... Donc, six cas... à multiplier par deux si nous voulons tenir compte des parties cachées. Ne voulant pas rédiger une encyclopédie, nous allons nous limiter à trois cas :

- rotation d'un objet et visualisation par projection (sans tenir compte des arêtes cachées) ;
- déplacement d'un observateur autour de l'objet avec la possibilité de regarder cet objet sous n'importe quel angle (toujours sans tenir compte des arêtes cachées) ;
- comment tenir compte des arêtes cachées ?

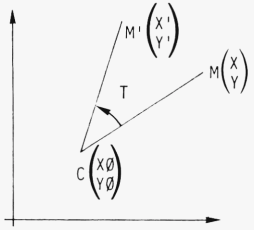
Même si, pour l'instant, certaines expressions (visualisation par projection, regarder sous n'importe quel angle...) ne sont pas claires, leur sens sera précisé dans chaque étude.

ROTATION D'OBJETS

Rotation

Le plus simple, avant de faire tourner un objet est de faire tourner un simple point dans l'espace.

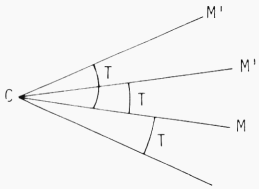
Rappelons (voir première partie, chapitre II) que, dans un plan, pour un repère orthonormé (mêmes unités sur les axes perpendiculaires, orientation classique), les formules définissant une rotation d'angle T autour d'un point C sont :




```
X -XØ=(X-XØ)*COS(T)-(Y-YØ)*SIN(T)
Y -YØ=(X-XØ)*SIN(T)+(Y-YØ)*COS(T)
```

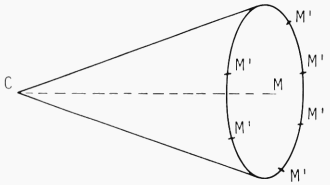
Il s'agit de coordonnées vraies dans le repère du matheux, pas du tout de coordonnées-écran qui feraient intervenir le coefficient KE de la page 50.

Et dans l'espace ? On ne peut pas tourner autour d'un point :

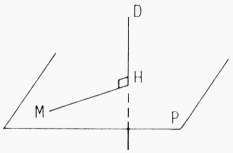


Tous les points M' sont tels que $\widehat{MCM'} = T$

De façon précise, les points M' tels que $\widehat{MCM'}=T$ sont sur le cône de révolution d'axe CM, d'angle T. Comme CM=CM', les points M' appartiennent aussi à une sphère de centre C, de rayon CM. Bilan : les points M' appartiennent à un cercle :



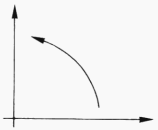
Entre tous ces points M', lequel choisir ? Dans le doute, abstenons-nous : dans l'espace, on ne peut pas tourner autour d'un seul point. On ne peut éventuellement tourner qu'autour d'une droite.



Appelons :

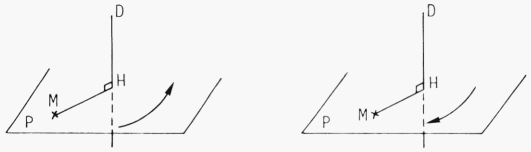
D : la droite
P : le plan perpendiculaire à D passant par M
H : la projection orthogonale de M sur D.

Tourner autour de D d'un angle T c'est tourner autour du point H, dans le plan P. Nous savons tourner dans un plan... enfin, dans le cas où le plan est orienté :

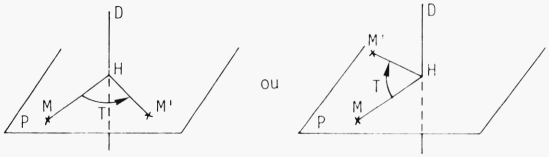


Sens usuel d'orientation d'un plan

Mais pour notre plan P qui est dans l'espace, quel est le sens de rotation dit normal ?

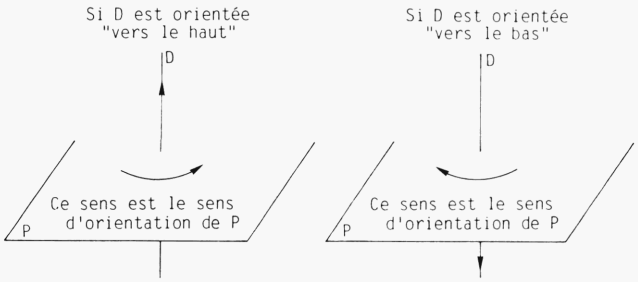


La question est importante puisque, selon le choix de sens de rotation, le point déduit de M par cette rotation d'angle T est :

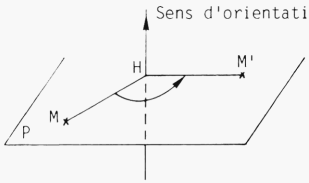


Par rapport à notre première tentative (rotation autour d'un point), la situation est tout de même un peu plus simple : nous n'avons à choisir qu'entre deux points.

Ce qui permet de trancher nécessite une précision. Tourner autour d'une droite n'est pas possible. On ne peut, dans l'espace, que tourner autour d'un axe. Quelle différence y a-t-il entre une droite et un axe ? Un axe est une droite orientée, c'est-à-dire une droite sur laquelle on a choisi un des deux sens de parcours possibles. Ce sens est arbitraire, mais il faut en choisir un !

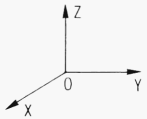


Cela est une convention, tout comme le sens inverse des aiguilles d'une montre pour l'orientation du plan. Il faut bien comprendre qu'une fois l'orientation de D choisie, la rotation d'angle T est parfaitement définie. Ainsi, pour T=90° :

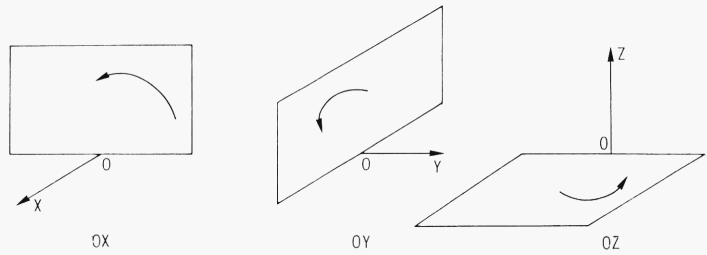


Dans l'espace, l'expression "tourner autour d'une droite d'un angle de..." manque de précision. Seule la phrase "Tourner autour d'un axe d'un angle de..." a un sens (admirez l'utilisation du mot sens !).

Notre but est de manipuler des objets. Si, à chaque fois qu'on veut effectuer une rotation, il faut demander à l'utilisateur dans quel sens l'axe est orienté, les manipulations vont vite devenir fastidieuses. Aussi, nous nous limiterons à trois rotations pour lesquelles il n'y a pas de problème : les rotations autour des axes OX, OY, OZ :

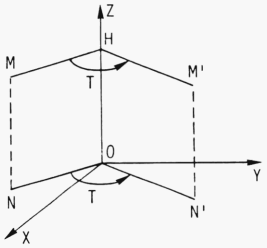


D'après notre convention, un plan P perpendiculaire à OX, OY ou OZ est orienté ainsi, selon qu'on tourne autour de :



Cela va nous permettre d'obtenir les formules définissant ces trois types de rotations. Prenons, par exemple, les rotations autour de OZ, d'angle T.

Par construction même Z'=Z, N et N' étant les projections de M et M' sur le plan XOY, on passe, dans ce plan, de N à N' par la rotation de centre O et d'angle T. Il suffit donc de reprendre les formules définissant une rotation plane et rappelées plus haut avec XØ=YØ=0.



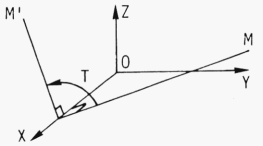
X'=X COS(T)-Y SIN(T)
Y'=X SIN(T)+Y COS(T)
Z'=Z

Rotation autour de OZ angle T

Les deux autres cas s'en déduisent facilement. Pour les rotations autour de OX :

OX joue le rôle que jouait OZ ;
OY joue le rôle que jouait OX ;
OZ joue le rôle que jouait OY.

Il suffit donc de remplacer Z par X, X par Y et Y par Z dans les formules précédentes :



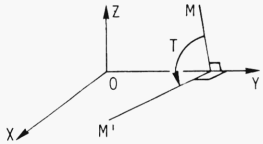
X'=X
Y'=Y COS(T)-Z SIN(T)
Z'=Y SIN(T)+Z COS(T)

Rotation autour de OX angle T

Enfin, pour les rotations autour de OY :

OY joue le rôle que jouait OZ ;
OZ joue le rôle que jouait OX ;
OX joue le rôle que jouait OY.

Remplaçons donc Z par Y, X par Z, Y par X dans les premières formules :



X'=Z SIN(T)+X COS(T)
Y'=Y
Z'=Z COS(T)-X SIN(T)

Rotation autour de OY angle T

Voilà une chose réglée, nous savons faire tourner un point autour de chacun des trois axes OX, OY, OZ. Vous reconnaîtrez sans peine ces formules dans le programme (page 192, lignes 45ØØ). La seule modification concerne X, Y? Z remplacés par ce que va entraîner notre façon de définir un objet.

Stockage d'un objet

Nous l'avons déjà dit, la façon de définir un objet dépend du type d'application envisagée. Pour les manipulations étudiées, les faces de l'objet n'ont pas d'importance, nous voulons faire tourner les sommets et tracer les arêtes joignant ces sommets.

Comme chaque arête est un segment, la première idée est de définir toute arête par ses deux extrémités. Pour faire tourner l'objet, on fera tourner successivement chacun des segments. Cette idée simple conduit à de nombreux calculs inutiles. Dans le cas d'un cube, chaque sommet est extrémité de trois des douze arêtes et les calculs seront effectués trois fois.

Une variante de cette idée consiste toujours à définir l'objet par ses sommets, en précisant comment ils sont joints. Cette méthode est assez simple à mettre en oeuvre, elle nécessite des tableaux X( ), Y( ), Z( ) pour les coordonnées et un tableau de liens entre les sommets.

Ce n'est pas la méthode que nous adopterons... Nous avons déjà rencontré un problème de ce genre. Oui, oui, vous ne vous souvenez pas ? Pensez au tout début de ce livre, aux motifs à déformer... Cela vous a mis la puce à l'oreille ? Nous avions défini alors un motif comme une suite de segments consécutifs ayant une "couleur" (1 : segment vu, Ø : segment caché). Il suffit de reprendre le même schéma en ajoutant la coordonnée Z. Ne considérant qu'un seul objet et non plus un motif à déformer en un autre, des tableaux à un seul indice suffisent.

	Coordonnées	*Couleurs*
Segment numéro 1	X(Ø), Y(Ø), Z(Ø)	COU(1)
Segment numéro 2	X(1), Y(1), Z(1)	COU(2)
	X(2), Y(2), Z(2)	COU(3)
...	...	...

et ainsi de suite.

Par rapport à l'introduction de motifs, une autre différence est à signaler. Autant il était utile pour les motifs que le dialogue soit interactif, autant il est illusoire de penser que nous parviendrons à définir, de façon interactive, un objet de l'espace. Conséquence : les objets manipulés seront définis en DATA. Ces DATAS auront la structure suivante :

- nombre de segments définissant l'objet ;
- DATAS par groupes de quatre : trois de coordonnées (X, Y, Z), puis la "couleur" du segment.

Le programme de lecture des DATAS est très simple (lignes 1ØØØ-1Ø7Ø page 191).

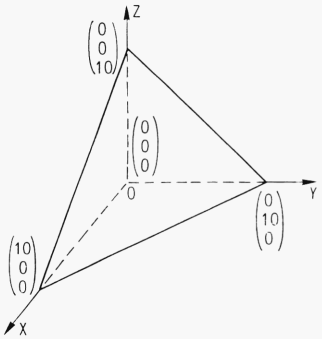
Deux précisions pour terminer. Les couleurs du segment sont indexées à partir de 1 (contrairement à ce que nous avons fait pour les motifs où cou(Ø) ne servait qu'à forcer la couleur du premier segment.
Cou(Ø) ne sert à rien ici. Nous lui donnerons un rôle dans une autre application. Contrairement encore à ce que nous avons fait pour les motifs, le programme de lecture des DATAS n'effectue aucune vérification sur les couleurs : pas de test par rapport à zéro ou un, pas de test pour qu'après un segment caché (couleur Ø) figure obligatoirement un segment vu (couleur 1). A vous de bien définir vos objets, cher lecteur.

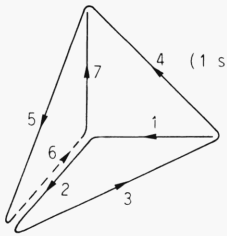
Pourquoi ce manque de rigueur ? Parce que la meilleure façon de procéder est de créer des fichiers d'objets. Une fois un objet bien défini, écrivez un fichier définissant cet objet. A la place de la lecture de DATAS, donnez à l'utilisateur la possibilité de lire des fichiers définissant des objets. C'est volontairement que nous n'avons pas rédigé ces parties d'écriture ou de lecture de fichiers : la rédaction diffère selon le matériel utilisé... Nous avons essayé de rédiger des programmes graphiques adaptables à tout matériel. Nous ne tenterons pas de faire la même chose pour la gestion de fichiers : nous vous laissons ce travail.

Cependant, pour vous aider, nous vous donnons quelques exemples d'objets définis par une suite de segments. Les X, Y, Z sont les coordonnées dans l'espace du mathématicien, enfin l'espace usuel.

Nous avons défini la façon de stocker les objets manipulés. Nous savons les faire tourner autour des axes OX, OY, OZ. Reste une question, sûrement la plus importante de toutes : qu'allons-nous représenter à l'écran ?

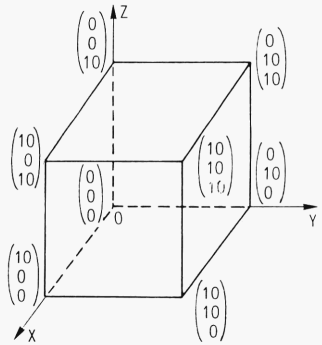
Une pyramide

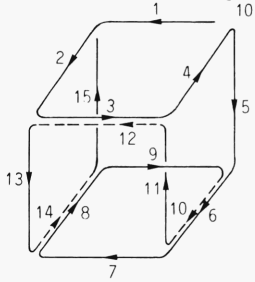




7 segments
(1 seul caché : 6)

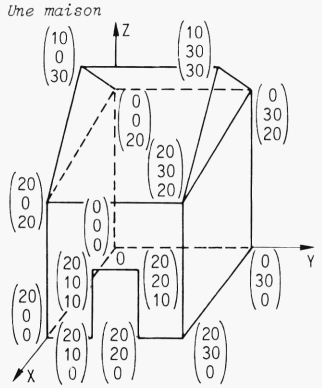
Un cube

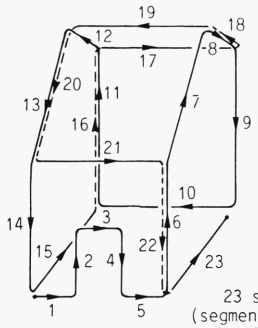




15 segments
(segments cachés :
10, 12,14)

Une maison





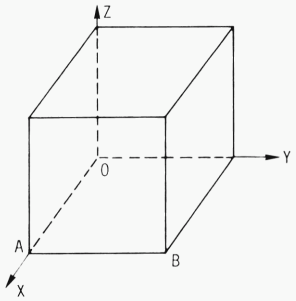
23 segments
(segments cachés :
16, 18, 20, 22)

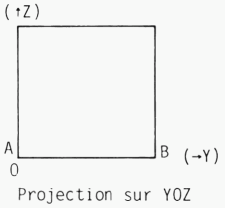
Représentation sur l'écran

Là, les choses se compliquent nettement. Tant qu'on ne se soucie que de ce qui se passe au niveau des coordonnées des points dans l'espace, on utilise, presque bêtement, des formules mathématiques.

Représenter l'objet manipulé sur l'écran est plus délicat. Pour éviter trop de difficultés, nous ne tiendrons pas compte des lignes vues ou cachées : tout sera vu sur l'écran. Par contre, sur les schémas, nous tracerons en pointillé les lignes cachées, cela facilitera la vision dans l'espace. Pour l'instant, nous représenterons des projections de nos objets sur le plan YOZ qui sera celui de notre écran.

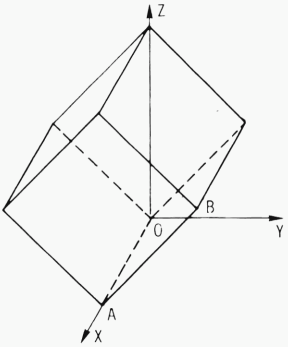
Ainsi, pour un cube :

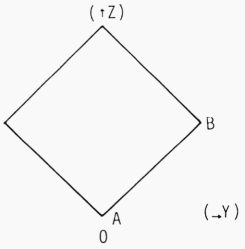




Projection sur YOZ

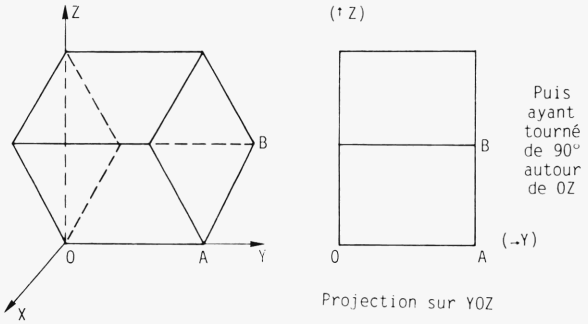
Cube au départ





Projection sur YOZ

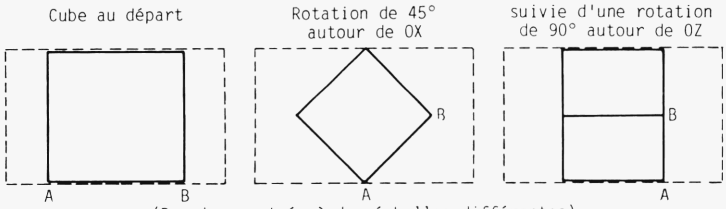
Cube ayant tourné de 45° autour de OX



Cela précise le terme projection, mais ne répond pas au problème de représentation sur l'écran.

Deux idées assez simples peuvent venir à l'esprit :

- utiliser la plus grande partie possible de l'écran pour représenter l'objet. On verra ainsi le maximum de détails ; dans cette optique, nous aurions à l'écran :



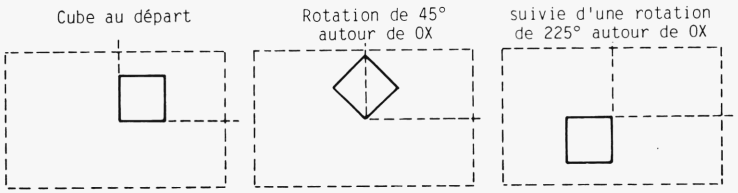
(Dessins centrés à des échelles différentes)

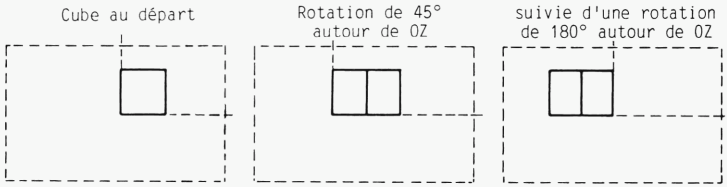
Là encore, nouveau choix possible sur les schémas ci-dessus, le même segment AB est représenté par des longueurs différentes. Il serait judicieux que tous nos dessins soient à la même échelle... et bien centrés, ce que nous avons fait sur les schémas.

Nous verrons comment tenir compte de tout cela au niveau programmation, mais auparavant, détaillons l'autre idée :

- ne pas représenter l'objet sur la plus grande partie de l'écran, mais essayer de tenir compte de sa position dans l'espace pour "voir" l'effet de la rotation dans l'espace.

Ainsi, pour notre cube revenu à sa position de départ :





(Dessins à la même échelle)

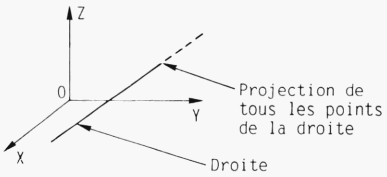
Nos dessins seront aussi bien centrés, là ce n'est pas difficile : O doit se projeter au centre de l'écran. Ils seront aussi tous à la même échelle.

Entrons dans le détail de la programmation.

PREMIERE REPRESENTATION

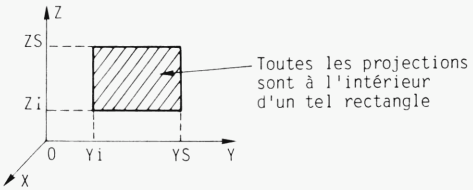
Deux questions sont à examiner : l'échelle et le centrage.

Commençons par la plus simple : le centrage. Tous les points d'une même droite parallèle à OX ont la même projection sur le plan YOZ.



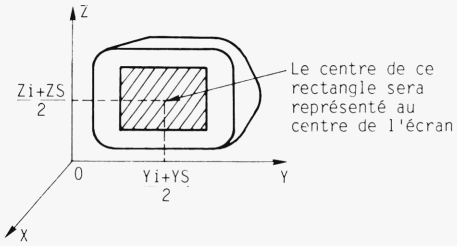
Nous ne considérons donc plus l'abscisse X des points de l'espace : elle n'a aucun rôle dans les projections. Seules, les coordonnées Y et Z sont utiles.

Toutes les projections, sur le plan YOZ, de tous les sommets de notre objet sont à l'intérieur d'un certain rectangle qui varie selon les manipulations effectuées :



Obtenir les valeurs de Yi, YS (ordonnées minimale et maximale) Zi, ZS (cotes minimale et maximale) ne pose pas de difficulté : on initialise toutes ces variables grâce au premier point, puis on effectue une boucle sur les autres points (sous-programme lignes 3020-3100, page 196).

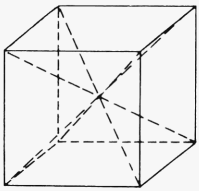
Afin de centrer notre image sur l'écran, nous conviendrons que le centre de ce rectangle sera dessiné au centre de l'écran :



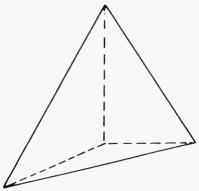
Le point de coordonnées $(\frac{Yi+YS}{2}, \frac{Zi+ZS}{2})$ est donc représenté par le point de coordonnées (XC,YC) sur l'écran.

Comment tenir compte du facteur échelle ? Cette question en recouvre deux. Premièrement, comment faire tenir tout le rectangle des points projetés sur l'écran ? Deuxièmement, comment faire que des longueurs projetées égales soient représentées par des segments de même longueur sur l'écran ?

Au cours des manipulations, les distances entre les différents points de l'objet restent les mêmes. Nous ne déformons pas l'objet. Par contre, les longueurs des segments projetés sur le plan YOZ varient. La plus grande distance projetée possible est la plus grande distance entre deux sommets de l'objet.



Plus grande distance mutuelle possible entre 2 sommets de notre cube : $10\sqrt{3}$

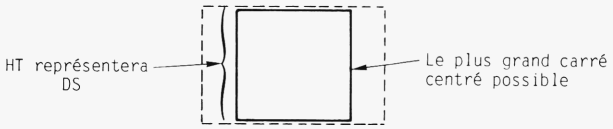


Plus grande distance mutuelle possible entre 2 sommets de notre pyramide : $10\sqrt{2}$

Nous appellerons DS cette plus grande longueur. Sa détermination est aisée : on initialise DS à Ø puis on compare les distances mutuelles entre les sommets d'indice I et J qui sont :

$$\sqrt{(X(I)-X(J))^2+(Y(I)-Y(J))^2+(Z(I)-Z(J))^2}$$

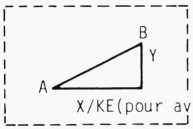
Sur l'écran, cette distance sera représentée par le côté du plus grand carré possible, soit HT :



Si HT correspond à DS, à quoi l'autre côté correspond-il, en fonction de LG ? Là, un petit rappel est indispensable. Les repères des écrans

d'ordinateurs ne sont pas orthonormés... Alors, ou bien on touche au réglage horizontal ou vertical du moniteur (solution "hard"), ou bien on utilise le bon vieux coefficient KE du chapitre II de la première partie (solution "soft").

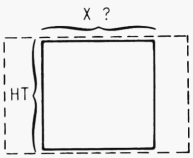
Rappelons que, quand on prend pour unité de longueur l'unité selon Y de l'écran, la distance entre deux points A et B est :



$$AB = \sqrt{\frac{X^2}{KE^2} + Y^2}$$

X/KE(pour avoir les mêmes unités sur les 2 axes)

Nous, nous voulons trouver X tel que la longueur horizontale AB soit égale à HT. Donc :



$$HT = \sqrt{\frac{X^2}{KE^2} + 0^2}$$

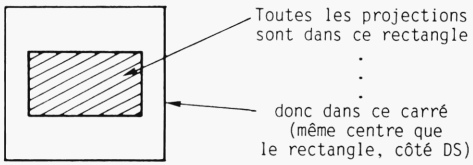
d'où $X^2 = HT^2 \times KE^2$

$X = HT \times KE$

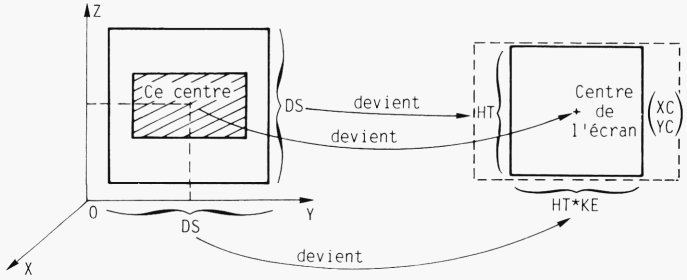
Bilan : le plus grand carré traçable à l'écran a pour côtés HT et HT*KE. Vous pouvez facilement le dessiner en tapant :

```
10 LG=    HT=
20 KE=
30 DROITE 0,0 A HT*KE,0
40 DROITE HT*KE,0 A HT*KE,HT
50 DROITE HT*KE,HT A 0,HT
60 DROITE 0,HT A 0,0
```

Comme YS-Yi et ZS-Zi sont au maximum les plus grandes distances projetées, donc égales à DS, il est normal de dire que :



En appelant YM=(Yi+YS)/2 et ZM=(ZS+Zi)/2 l'ordonnée et le côté du centre du rectangle contenant tous les projetés sur YOZ, la correspondance entre points projetés et points dessinés sur l'écran est représentée page ci-contre.



Deux règles de trois donnent les valeurs cherchées pour XØ et YØ, coordonnées écran du point projeté, de coordonnées (X,Y,Z) dans l'espace :

$$\frac{Y-YM}{DS} = \frac{XØ-XC}{HT*KE} \qquad XØ = XC + \frac{Y-YM}{DS} * HT * KE$$

$$-\frac{Z-ZM}{DS} = \frac{YØ-YC}{HT} \qquad YØ = YC - \frac{Z-ZM}{DS} * HT$$

(le signe - tient compte de l'orientation de l'axe des Y sur l'écran).

Maintenant, tout devrait s'éclairer. Récapitulons :

- on détermine la plus grande distance mutuelle DS entre deux sommets de l'objet. Elle sera représentée par le côté du plus grand carré affichable à l'écran ;
- on détermine Yi, YS, Zi, ZS définissant le rectangle où tous les points se projettent. Son centre (YM=(Yi+YS)/2, ZM=(Zi+ZS)/2) sera représenté au centre de l'écran ;
- le point de coordonnées (X,Y,Z) de l'espace est représenté par le point de coordonnées :

XC+(Y-YM)*HT*KE/DS et YC-(Z-ZM)*HT/DS sur l'écran.

On effectue alors une boucle sur les sommets pour représenter toutes les arêtes projetées. Si la couleur est 1, l'arête est affichée, sinon le programme passe à l'arête suivante.

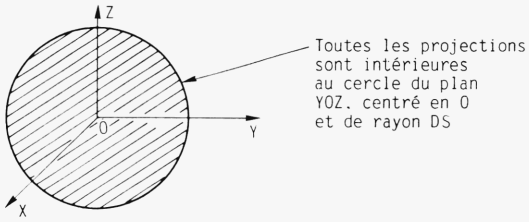
L'étape 1 correspond au sous-programme commençant à la ligne 2ØØØ. Les étapes 2 et 3 sont regroupées et commencent à la ligne 3ØØØ.

SECONDE REPRESENTATION

Là, la méthode utilisée est un peu plus simple.

Pour le centrage, l'origine du repère est représentée par le centre de l'écran. Cela ne pose aucun problème.

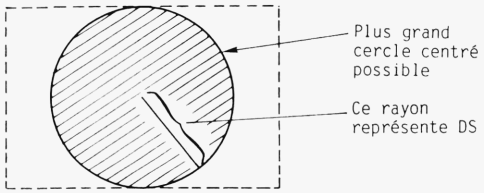
Pour l'échelle, il suffit de bien comprendre que, dans ce cas, tous les points projetés sont à l'intérieur d'un cercle de centre O dont le rayon est la plus grande des distances entre O et les sommets de l'objet :



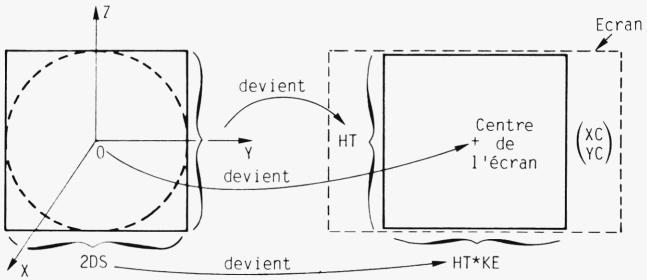
Appelons encore DS cette plus grande distance. Cette fois, c'est le plus grand de tous les nombres :

$$\sqrt{X(I)^2 + (Y(I)^2 + Z(I)^2}$$

Notre cercle sera représenté sur l'écran par le plus grand cercle possible :



HT devra donc représenter 2 DS. Là encore, l'unité la plus naturelle est celle des Y ! La correspondance entre points projetés sur le plan YOZ et points sur l'écran est :



Les règles de trois permettant d'obtenir les coordonnées (XØ,YØ) sur l'écran du point de coordonnées (X,Y,Z) de l'espace sont :

$$\frac{Y-Ø}{2*DS} = \frac{XØ-XC}{HT*KE} \qquad XØ = XC + \frac{Y}{2*DS} * HT * KE$$

$$-\frac{Z-Ø}{2*DS} = \frac{YØ-YC}{HT} \qquad YØ = YC - \frac{Z}{2*DS} * HT$$

(le signe - tient compte de l'orientation de l'axe des Y sur l'écran).

Récapitulons :

- on détermine la plus grande distance DS entre l'origine du repère et les sommets de l'objet. Elle sera représentée par le rayon du plus grand cercle affichable à l'écran ;
- le point de coordonnées (X,Y,Z) de l'espace est représenté par le point de coordonnées :

  XC+Y*HT*KE/2/DS et YC-Z*HT/2/DS

  sur l'écran.

Une boucle tenant compte de la couleur de chaque segment permet de visualiser tout l'objet.

Programmation

Vous vous en êtes sans doute aperçu, il y a de grandes similitudes entre les deux types de représentations. Par la rédaction même, nous l'avons fait sentir... peut-être même un peu lourdement.

Cela met en évidence la structure commune aux deux cas envisagés. Le programme en tient compte. Il appelle quatre sous-programmes principaux :

- sous-programme de lecture des DATAS définissant l'objet (à partir de la ligne 1ØØØ) ;
- calcul de la plus grande distance DS (à partir de la ligne 2ØØØ) ;
- représentation sur l'écran (à partir de la ligne 3ØØØ) ;
- type de manipulation. Ce sous-programme appelle lui-même trois sous-programmes qui effectuent, sur les coordonnées, les rotations voulues (à partir de la ligne 4ØØØ).

Nous donnerons deux versions des sous-programmes traitant les étapes 2 et 3 : il s'agit des seules différences entre les deux types de visualisation.

Vous remarquerez que le calcul de DS s'effectue en comparant non pas les distances, mais leurs carrés (lignes 2Ø3Ø ou 2Ø5Ø) ; ce n'est qu'à la fin que nous prenons la racine carrée... à laquelle nous ajoutons une petite quantité (Ø,1) : elle sert à éviter d'éventuelles erreurs d'arrondi pouvant conduire à des appels illégaux de fonctions pour des segments tout en haut ou tout en bas de l'écran. Vous pouvez évidemment modifier cette quantité, à la seule condition de ne pas la prendre négative !

Vous remarquerez aussi que nous avons simplifié un peu les expressions des coordonnées écran des points projetés : elles font toutes intervenir le terme HT/DS (ou HT/2/DS). Nous l'avons noté M (lignes 2Ø7Ø ou 21ØØ), rapport de mise en page.

Nous n'avons pas encore parlé du sous-programme demandant le type de manipulation. Il est très simple : on teste la frappe d'une des touches X, Y ou Z par GET ou INKEY, puis on introduit l'angle dont on tourne en degrés (lignes 4Ø7Ø à 4Ø9Ø).

Nous avons ajouté un affichage de "position" (lignes 4Ø3Ø à 4Ø6Ø) : avant d'effectuer la rotation voulue par l'utilisateur, l'ordinateur affiche les angles dont on a tourné l'objet selon X, selon Y, selon Z.

A quoi cela sert-il ? A illustrer une propriété bizarre baptisée par les mathéux du nom barbare de "non commutativité de la composition de rotations dans l'espace" (c'est fini !). Ayant bien manipulé votre objet, vous pouvez souhaiter revenir au point de départ. A première vue, il suffit d'effectuer une rotation autour de chaque axe de moins l'angle dont on a tourné. Essayez après plusieurs manipulations et constatez. Nous avons bien dit : à première vue. Et oui, notre intuition nous trompe souvent !

Nous raisonnons comme si, après de nombreuses rotations, l'objet avait globalement tourné autour de chaque axe d'un angle égal à la somme de tous ceux dont on a tourné autour de cet axe.

ROT(OX,9Ø)ROT(OY,45)ROT(OX,2Ø)ROT(OZ,42)ROT(OY,15)ROT(OX,15)

donnerait simplement :

ROT(OX,125)ROT(OY,6Ø)ROT(OZ,42)
(125=9Ø+2Ø+15 et 6Ø=45+15)

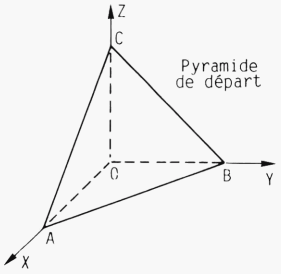
Pour regrouper toutes les rotations selon OX, toutes les rotations selon OY, toutes les rotations selon OZ, il faudrait que l'on puisse les échanger (les matheux disent commuter) et que :

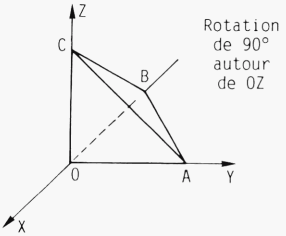
- effectuer d'abord ROT(OZ,9Ø) puis ROT(OY,9Ø)

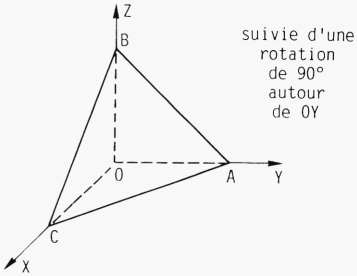
revienne au même que :

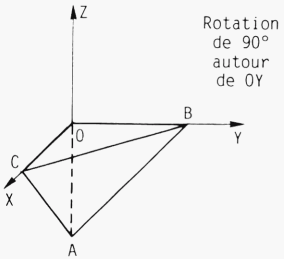
- effectuer d'abord ROT(OY,9Ø) puis ROT(OZ,9Ø)

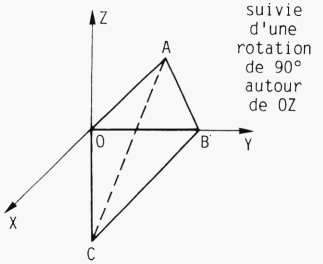
Mais cela est faux, comme le prouve le cas de la pyramide :











Méfiance donc ! La seule façon de ramener un objet à sa position initiale est de refaire des rotations d'angles opposés, mais en ordre inverse. Si vous avez effectué :

ROT(OZ,9Ø) puis ROT(OY,9Ø)

pour revenir à la position de départ, faites :

ROT(OY,-9Ø) puis ROT(OZ,-9Ø)

Dans la plupart des cas, on ne revient pas exactement à la position de départ. Les fonctions trigonométriques sont calculées de façon approchée et les erreurs de calcul se cumulent. Ne vous étonnez donc pas de "bavures" : elles sont normales et dépendent à la fois de la précision de votre machine et de la finesse de son graphisme. Des valeurs inexactes peuvent passer inaperçues à faible résolution.

Si vous ne souhaitez pas tenir compte de ces renseignements, détruisez les lignes 5Ø, 4Ø3Ø à 4Ø6Ø, ainsi que les instructions A(i)=A(i)+T des lignes 411Ø à 413Ø.

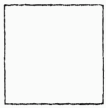
Après l'affichage de l'objet, pour effectuer une rotation, il suffit de taper sur la touche "retour-chariot" (enter, return, new-line, entrée...). Alors, l'écran s'efface et le programme demande "autour de X, Y, Z ?" puis l'angle dont on souhaite tourner.

Assez de discours, passons au programme pour la seconde représentation envisagée.

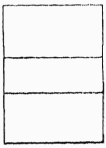
```
10 REM INITIALISATIONS
20 LG=    :HT=    :KE=
30 XC=    :YC=
40 PI=4*ATN(1)
50 DIM A(3)
100 GOSUB 1000:REM LECTURE D'UNE FIGURE
110 GOSUB 2000:REM CALCUL DE DS
120 GOSUB 3000:REM VISUALISATION
130 GOSUB 4000:REM AXE DE ROTATION
140 GOTO 120
1000 REM LECTURE D'UN OBJET EN DATA
1010 RESTORE
1020 READ S:REM NOMBRE DE SEGMENTS
1030 DIM X(S),Y(S),Z(S),COU(S)
1040 FOR I=0 TO S
1050 READ X(I),Y(I),Z(I),COU(I)
1060 NEXT I
1070 RETURN
2000 REM PLUS GRANDE DISTANCE
2010 DS=0
2020 FOR I=0 TO S
2030 D=X(I)*X(I)+Y(I)*Y(I)+Z(I)*Z(I)
2040 IF D>DS THEN DS=D
2050 NEXT I
2060 DS=SQR(DS)+0.1
2070 M=HT/2/DS
2080 RETURN
3000 REM VISUALISATION
3010 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
3020 X0=XC+Y(0)*M*KE
3030 Y0=YC-Z(0)*M
3040 FOR I=1 TO S
3050 X1=XC+Y(I)*M*KE
3060 Y1=YC-Z(I)*M
3070 IF COU(I)=1 THEN DROITE X0,Y0 A X1,Y1
3080 X0=X1                                    →
```

```
3090 Y0=Y1
3100 NEXT I
3110 A$=INKEY$:REM OU GET A$
3120 IF A$<>CHR$(13) THEN 3110
3130 RETURN
4000 REM AXE DE ROTATION
4010 REM MODE TEXTE
4020 REM EFFACAGE D'ECRAN
4030 PRINT"POSITION :"
4040 PRINT"X=";A(1)
4050 PRINT"Y=";A(2)
4060 PRINT"Z=";A(3)
4070 PRINT"AUTOUR DE X,Y,Z ?"
4080 A$=INKEY$ REM OU GET A$
4090 IF A$<"X" OR A$>"Z" THEN 4080
4100 INPUT"ANGLE (EN DEGRES)";T
4110 IF A$="X" THEN A(1)=A(1)+T:CH=1
4120 IF A$="Y" THEN A(2)=A(2)+T:CH=2
4130 IF A$="Z" THEN A(3)=A(3)+T:CH=3
4140 T=T*PI/180
4500 REM NOUVELLES COORDONNEES
4510 CO=COS(T):SI=SIN(T)
4520 ON CH GOSUB 4600,4700,4800
4530 RETURN
4600 REM ROTATION AUTOUR DE OX
4610 FOR I=0 TO S
4620 Y=Y(I):Z=Z(I)
4630 Y(I)=Y*CO-Z*SI
4640 Z(I)=Y*SI+Z*CO
4650 NEXT I
4660 RETURN
4700 REM ROTATION AUTOUR DE OY
4710 FOR I=0 TO S
4720 X=X(I):Z=Z(I)
4730 Z(I)=Z*CO-X*SI
4740 X(I)=Z*SI+X*CO
4750 NEXT I
4760 RETURN
4800 REM ROTATION AUTOUR DE OZ
4810 FOR I=0 TO S
4820 X=X(I):Y=Y(I)
4830 X(I)=X*CO-Y*SI
4840 Y(I)=X*SI+Y*CO
4850 NEXT I
4860 RETURN
```

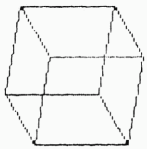
Manipulations d'un cube (seconde représentation)



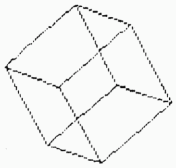
X = Ø
Y = Ø
Z = Ø



X = Ø
Y = 3Ø
Z = Ø

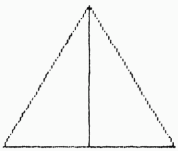


X = Ø
Y = 3Ø
Z = 2Ø

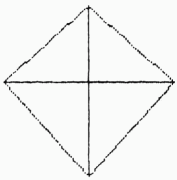


X = 40
Y = 30
Z = 20

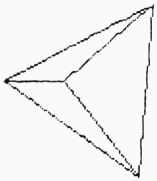
Manipulations d'un tétraèdre (seconde représentation)



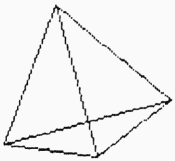
X = Ø
Y = Ø
Z = Ø



X = Ø
Y = 40
Z = Ø



X = Ø
Y = 4Ø
Z =11Ø



X = 3Ø
Y = 29Ø
Z = 11Ø

Pour la représentation plein écran, la première envisagée, le sous-programme de calcul de la plus grande distance DS, à partir de la ligne 2ØØØ, doit être remplacé par le suivant :

```
2000 REM PLUS GRANDE DISTANCE
2010 DS=0
2020 FOR I=0 TO S-1
2030 FOR J=I+1 TO S
2040 DX=X(J)-X(I):DY=Y(J)-Y(I):DZ=Z(J)-Z(I)
2050 D=DX*DX+DY*DY+DZ*DZ
2060 IF D>DS THEN DS=D
2070 NEXT J
2080 NEXT I
2090 DS=SQR(DS)+0.1
2100 M=HT/DS
2110 RETURN
```

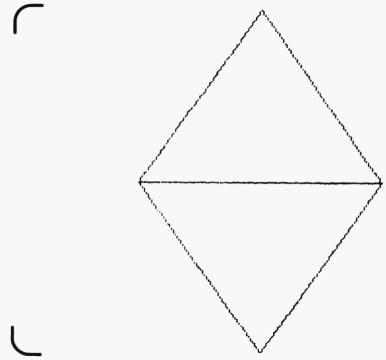
Pour cette représentation, le sous-programme de visualisation, à partir de la ligne 3000 du programme des pages 191-192, doit être remplacé par :

```
3000 REM VISUALISATION
3010 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
3020 REM CADRAGE
3030 YI=Y(0):YS=Y(0)
3040 ZI=Z(0):ZS=Z(0)
3050 FOR I=1 TO S
3060 IF Y(I)>YS THEN YS=Y(I)
3070 IF Y(I)<YI THEN YI=Y(I)
3080 IF Z(I)>ZS THEN ZS=Z(I)
3090 IF Z(I)<ZI THEN ZI=Z(I)
3100 NEXT I
3110 YM=(YI+YS)/2
3120 ZM=(ZI+ZS)/2
3200 REM TRACE
3210 X0=XC+(Y(0)-YM)*M*KE
3220 Y0=YC-(Z(0)-ZM)*M
3230 FOR I=1 TO S
3240 X1=XC+(Y(I)-YM)*M*KE
3250 Y1=YC-(Z(I)-ZM)*M
3260 IF COUL(I)=1 THEN DROITE X0,Y0 A X1,Y1
3270 X0=X1
3280 Y0=Y1
3290 NEXT I
3300 A$=INKEY$:REM OU GET A$
3310 IF A$<>CHR$(13) THEN 3300
3320 RETURN
```

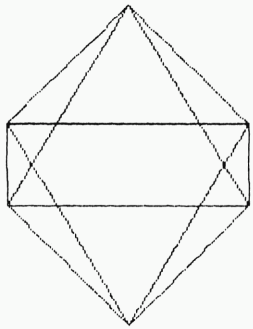
Lorsque l'on utilise ces programmes, après toute représentation, il faut taper sur la touche retour-chariot (return, enter, entrée, valide, new-line, enfin CHR$(13)) pour effectuer une manipulation (lignes 3110 et 3120).

Comme exemples d'objets, voici les cinq solides de Platon.

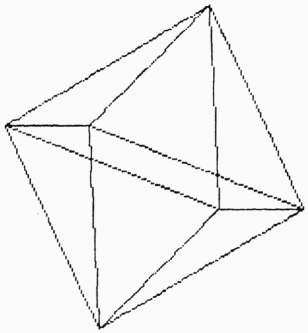
Manipulations d'un octaèdre (première représentation)



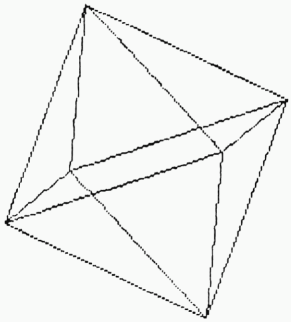
X = 0
Y = 0
Z = 0



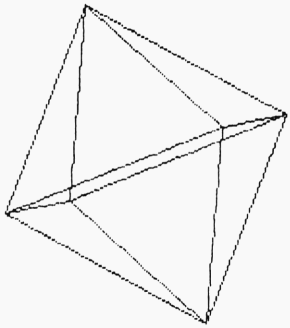
X = Ø
Y = 2Ø
Z = Ø



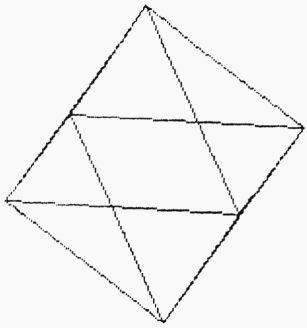
X = Ø
Y = 2Ø
Z = 7Ø



X = 4Ø
Y = 2Ø
Z = 7Ø



X = 40
Y = 10
Z = 70



X = 40
Y = 10
Z =160

```
10000  REM DATA TETRAEDRE
10010  DATA   7
10020  DATA    0.00000 ,    0.00000 ,    0.00000 ,   0
10030  DATA    0.00000 ,   10.00000 ,    0.00000 ,   1
10040  DATA    8.66026 ,    5.00000 ,    0.00000 ,   1
10050  DATA    0.00000 ,    0.00000 ,    0.00000 ,   1
10060  DATA    2.88675 ,    5.00000 ,    8.16497 ,   1
10070  DATA    8.66026 ,    5.00000 ,    0.00000 ,   1
10080  DATA    0.00000 ,   10.00000 ,    0.00000 ,   0
10090  DATA    2.88675 ,    5.00000 ,    8.16497 ,   1
                                                      →
```

```
10000  REM DATA CUBE
10010  DATA   15
10020  DATA    0.00000 ,   0.00000 ,   0.00000 ,  0
10030  DATA   10.00000 ,   0.00000 ,   0.00000 ,  1
10040  DATA   10.00000 ,  10.00000 ,   0.00000 ,  1
10050  DATA    0.00000 ,  10.00000 ,   0.00000 ,  1
10060  DATA    0.00000 ,   0.00000 ,   0.00000 ,  1
10070  DATA    0.00000 ,   0.00000 ,  10.00000 ,  1
10080  DATA   10.00000 ,   0.00000 ,  10.00000 ,  1
10090  DATA   10.00000 ,  10.00000 ,  10.00000 ,  1
10100  DATA    0.00000 ,  10.00000 ,  10.00000 ,  1
10110  DATA    0.00000 ,   0.00000 ,  10.00000 ,  1
10120  DATA    0.00000 ,  10.00000 ,  10.00000 ,  0
10130  DATA    0.00000 ,  10.00000 ,   0.00000 ,  1
10140  DATA   10.00000 ,  10.00000 ,   0.00000 ,  0
10150  DATA   10.00000 ,  10.00000 ,  10.00000 ,  1
10160  DATA   10.00000 ,   0.00000 ,  10.00000 ,  0
10170  DATA   10.00000 ,   0.00000 ,   0.00000 ,  1


10000  REM DATA OCTAEDRE
10010  DATA   12
10020  DATA    0.00000 ,   0.00000 ,   0.00000 ,  0
10030  DATA   10.00000 ,   0.00000 ,   0.00000 ,  1
10040  DATA    5.00000 ,   5.00000 ,   7.07107 ,  1
10050  DATA    0.00000 ,   0.00000 ,   0.00000 ,  1
10060  DATA    0.00000 ,  10.00000 ,   0.00000 ,  1
10070  DATA    5.00000 ,   5.00000 ,   7.07107 ,  1
10080  DATA   10.00000 ,  10.00000 ,   0.00000 ,  1
10090  DATA    5.00000 ,   5.00000 ,  -7.07107 ,  1
10100  DATA   10.00000 ,   0.00000 ,   0.00000 ,  1
10110  DATA   10.00000 ,  10.00000 ,   0.00000 ,  1
10120  DATA    0.00000 ,  10.00000 ,   0.00000 ,  1
10130  DATA    5.00000 ,   5.00000 ,  -7.07107 ,  1
10140  DATA    0.00000 ,   0.00000 ,   0.00000 ,  1


10000  REM DATA DODECAEDRE
10010  DATA   39
10020  DATA    4.58794 ,   6.31446 ,  -1.49071 ,  1
10030  DATA    7.42344 ,   1.49041 ,  -2.41202 ,  1
10040  DATA    7.42344 ,  -1.49101 ,   2.41202 ,  1
10050  DATA    4.58794 ,   1.49041 ,   6.31476 ,  1
10060  DATA    0.00000 ,  -1.49071 ,   7.80547 ,  1
10070  DATA    0.00000 ,  -6.31476 ,   4.82405 ,  1
10080  DATA    4.58794 ,  -6.31476 ,   1.49071 ,  1
10090  DATA    2.83550 ,  -6.31446 ,  -3.90273 ,  1
10100  DATA    4.58794 ,  -1.49071 ,  -6.31476 ,  1
10110  DATA    0.00000 ,   1.49071 ,  -7.80547 ,  1
10120  DATA   -4.58794 ,  -1.49041 ,  -6.31476 ,  1
10130  DATA   -7.42344 ,   1.49101 ,  -2.41202 ,  1
10140  DATA   -4.58794 ,   6.31476 ,  -1.49071 ,  1
10150  DATA    0.00000 ,   6.31476 ,  -4.82405 ,  1
10160  DATA    4.58794 ,   6.31446 ,  -1.49071 ,  1
10170  DATA    2.83550 ,   6.31446 ,   3.90273 ,  1
10180  DATA   -2.83550 ,   6.31446 ,   3.90273 ,  1
10190  DATA   -4.58794 ,   1.49071 ,   6.31476 ,  1
10200  DATA   -7.42344 ,  -1.49041 ,   2.41202 ,  1
10210  DATA   -4.58794 ,  -6.31446 ,   1.49071 ,  1
10220  DATA   -2.83550 ,  -6.31446 ,  -3.90273 ,  1
10230  DATA    2.83550 ,  -6.31446 ,  -3.90273 ,  1
10240  DATA    4.58794 ,  -1.49071 ,  -6.31476 ,  0
                                                  →
```

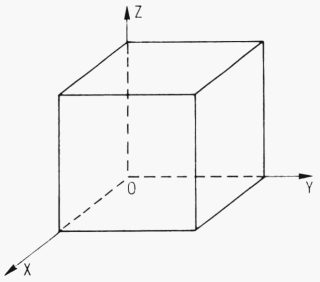
```
10250  DATA    7.42344 ,   1.49041 ,  -2.41202 ,  1
10260  DATA    0.00000 ,   6.31476 ,  -4.82405 ,  0
10270  DATA    0.00000 ,   1.49071 ,  -7.80547 ,  1
10280  DATA   -4.58794 ,   6.31476 ,  -1.49071 ,  0
10290  DATA   -2.83550 ,   6.31446 ,   3.90273 ,  1
10300  DATA   -4.58794 ,   1.49071 ,   6.31476 ,  0
10310  DATA    0.00000 ,  -1.49071 ,   7.80547 ,  1
10320  DATA    0.00000 ,  -6.31476 ,   4.82405 ,  0
10330  DATA   -4.58794 ,  -6.31446 ,   1.49071 ,  1
10340  DATA    4.58794 ,  -6.31476 ,   1.49071 ,  0
10350  DATA    7.42344 ,  -1.49101 ,   2.41202 ,  1
10360  DATA    4.58794 ,   1.49041 ,   6.31476 ,  0
10370  DATA    2.83550 ,   6.31446 ,   3.90273 ,  1
10380  DATA   -7.42344 ,  -1.49041 ,   2.41202 ,  0
10390  DATA   -7.42344 ,   1.49101 ,  -2.41202 ,  1
10400  DATA   -4.58794 ,  -1.49041 ,  -6.31476 ,  0
10410  DATA   -2.83550 ,  -6.31446 ,  -3.90273 ,  1


10000  REM DATA ICOSAEDRE
10010  DATA   35
10020  DATA    0.00000 ,   4.47214 ,   8.94427 ,  1
10030  DATA   -5.25731 ,  -4.47214 ,   7.23607 ,  1
10040  DATA    5.25731 ,  -4.47214 ,   7.23607 ,  1
10050  DATA    0.00000 ,   4.47214 ,   8.94427 ,  1
10060  DATA    8.50651 ,   4.47124 ,   2.76393 ,  1
10070  DATA    5.25731 ,  -4.47214 ,   7.23607 ,  1
10080  DATA    8.50651 ,  -4.47214 ,  -2.76393 ,  1
10090  DATA    8.50651 ,   4.47124 ,   2.76393 ,  1
10100  DATA    5.25731 ,   4.47214 ,  -7.23607 ,  1
10110  DATA    8.50651 ,  -4.47214 ,  -2.76393 ,  1
10120  DATA    0.00000 ,  -4.47214 ,  -8.94427 ,  1
10130  DATA    5.25731 ,   4.47214 ,  -7.23607 ,  1
10140  DATA   -5.25731 ,   4.47214 ,  -7.23607 ,  1
10150  DATA    0.00000 ,  -4.47214 ,  -8.94427 ,  1
10160  DATA   -8.50651 ,  -4.47124 ,  -2.76393 ,  1
10170  DATA   -5.25731 ,   4.47214 ,  -7.23607 ,  1
10180  DATA   -8.50651 ,   4.47214 ,   2.76393 ,  1
10190  DATA   -8.50651 ,  -4.47124 ,  -2.76393 ,  1
10200  DATA   -5.25731 ,  -4.47214 ,   7.23607 ,  1
10210  DATA   -8.50651 ,   4.47214 ,   2.76393 ,  1
10220  DATA    0.00000 ,   4.47214 ,   8.94427 ,  1
10230  DATA    0.00000 ,  10.00000 ,   0.00000 ,  1
10240  DATA   -8.50651 ,   4.47214 ,   2.76393 ,  1
10250  DATA   -5.25731 ,   4.47214 ,  -7.23607 ,  0
10260  DATA    0.00000 ,  10.00000 ,   0.00000 ,  1
10270  DATA    5.25731 ,   4.47214 ,  -7.23607 ,  1
10280  DATA    8.50651 ,   4.47124 ,   2.76393 ,  0
10290  DATA    0.00000 ,  10.00000 ,   0.00000 ,  1
10300  DATA    0.00000 , -10.00000 ,   0.00000 ,  0
10310  DATA   -8.50651 ,  -4.47124 ,  -2.76393 ,  1
10320  DATA    0.00000 ,  -4.47214 ,  -8.94427 ,  0
10330  DATA    0.00000 , -10.00000 ,   0.00000 ,  1
10340  DATA    8.50651 ,  -4.47214 ,  -2.76393 ,  1
10350  DATA    5.25731 ,  -4.47214 ,   7.23607 ,  0
10360  DATA    0.00000 , -10.00000 ,   0.00000 ,  1
10370  DATA   -5.25731 ,  -4.47214 ,   7.23607 ,  1
```

→

```
10000  REM DATA DE QUOI S'AGIT-IL
10010  DATA    43
10020  DATA     4.000000 ,    0.000000 ,    0.000000 ,   1
10030  DATA     2.000000 ,    0.000000 ,    4.000000 ,   1
10040  DATA     1.000000 ,    0.000000 ,    8.000000 ,   1
10050  DATA     0.000000 ,    0.000000 ,   20.000000 ,   1
10060  DATA    -1.000000 ,    0.000000 ,    8.000000 ,   1
10070  DATA    -2.000000 ,    0.000000 ,    4.000000 ,   1
10080  DATA    -4.000000 ,    0.000000 ,    0.000000 ,   1
10090  DATA     0.000000 ,    4.000000 ,    0.000000 ,   0
10100  DATA     0.000000 ,    2.000000 ,    4.000000 ,   1
10110  DATA     0.000000 ,    1.000000 ,    8.000000 ,   1
10120  DATA     0.000000 ,    0.000000 ,   20.000000 ,   1
10130  DATA     0.000000 ,   -1.000000 ,    8.000000 ,   1
10140  DATA     0.000000 ,   -2.000000 ,    4.000000 ,   1
10150  DATA     0.000000 ,   -4.000000 ,    0.000000 ,   1
10160  DATA     1.000000 ,   -3.000000 ,    0.000000 ,   1
10170  DATA     1.000000 ,   -2.000000 ,    2.000000 ,   1
10180  DATA     2.000000 ,   -1.000000 ,    2.000000 ,   1
10190  DATA     3.000000 ,   -1.000000 ,    0.000000 ,   1
10200  DATA     4.000000 ,    0.000000 ,    0.000000 ,   1
10210  DATA     3.000000 ,    1.000000 ,    0.000000 ,   1
10220  DATA     2.000000 ,    1.000000 ,    2.000000 ,   1
10230  DATA     1.000000 ,    2.000000 ,    2.000000 ,   1
10240  DATA     1.000000 ,    3.000000 ,    0.000000 ,   1
10250  DATA     0.000000 ,    4.000000 ,    0.000000 ,   1
10260  DATA    -1.000000 ,    3.000000 ,    0.000000 ,   1
10270  DATA    -1.000000 ,    2.000000 ,    2.000000 ,   1
10280  DATA    -2.000000 ,    1.000000 ,    2.000000 ,   1
10290  DATA    -3.000000 ,    1.000000 ,    0.000000 ,   1
10300  DATA    -4.000000 ,    0.000000 ,    0.000000 ,   1
10310  DATA    -3.000000 ,   -1.000000 ,    0.000000 ,   1
10320  DATA    -2.000000 ,   -1.000000 ,    2.000000 ,   1
10330  DATA    -1.000000 ,   -2.000000 ,    2.000000 ,   1
10340  DATA    -1.000000 ,   -3.000000 ,    0.000000 ,   1
10350  DATA     0.000000 ,   -4.000000 ,    0.000000 ,   1
10360  DATA     0.000000 ,   -2.000000 ,    4.000000 ,   0
10370  DATA     2.000000 ,    0.000000 ,    4.000000 ,   1
10380  DATA     0.000000 ,    2.000000 ,    4.000000 ,   1
10390  DATA    -2.000000 ,    0.000000 ,    4.000000 ,   1
10400  DATA     0.000000 ,   -2.000000 ,    4.000000 ,   1
10410  DATA     0.000000 ,   -1.000000 ,    8.000000 ,   0
10420  DATA     1.000000 ,    0.000000 ,    8.000000 ,   1
10430  DATA     0.000000 ,    1.000000 ,    8.000000 ,   1
10440  DATA    -1.000000 ,    0.000000 ,    8.000000 ,   1
10450  DATA     0.000000 ,   -1.000000 ,    8.000000 ,   1
```

Après avoir bien manipulé ces polyèdres, vous aurez sans doute constaté que, pour le cube, l'octaèdre, le tétraèdre, les deux représentations diffèrent beaucoup tandis que pour le dodécaèdre et l'icosaèdre, la différence est difficilement perceptible.

D'où cela provient-il ? Du centrage des polyèdres. Prenons l'exemple du cube, tel qu'il est donné :



- pour la première représentation (plein écran) :

  $DS = 10\sqrt{3}+0,1$

  et la hauteur de l'écran représente DS, soit $10\sqrt{3}+0,1$;

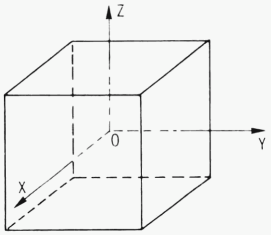
- pour la seconde représentation :

  $DS = 10\sqrt{3}+0,1$

  et la hauteur de l'écran représente 2DS, soit $20\sqrt{3}+0,2$.

Entre les deux représentations, l'échelle passe du simple au double et le centrage sur l'écran diffère !

Par contre, si nous plaçons le centre du cube à l'origine du repère :



- pour la première représentation

  $DS = 10\sqrt{3}+0,1$

  et la hauteur de l'écran représente toujours DS, soit $10\sqrt{3}+0,1$ (dans ce cas, DS ne dépend pas du centrage du cube) ;

- pour la seconde représentation :

  $DS = 10\sqrt{3}/2+0,1$

  et la hauteur de l'écran représente 2DS, soit $10\sqrt{3}+0,2$

L'échelle est à peu près la même, le centrage sur l'écran est le même... à vous de conclure !

Pour centrer ou décentrer à volonté ces polyèdres, voici les coordonnées de leur centre :

Type du polyèdre	*Coordonnées du centre du polyèdre*
Tétraèdre	(2,8875315;5;2,04124146)
Cube	(5;5;5)
Octaèdre	(5;5;Ø)
Dodécaèdre	(Ø;Ø;Ø)
Icosaèdre	(Ø;Ø;Ø)

Plus le centre d'un polyèdre est éloigné de l'origine du repère, plus les représentations diffèrent.

DEPLACEMENT D'UN OBSERVATEUR

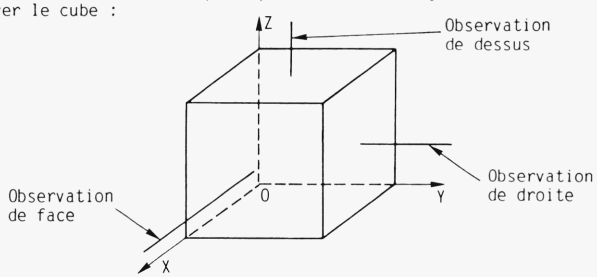
Nous venons de voir comment faire subir, à un objet, des rotations autour des axes OX, OY, OZ. Les représentations sur l'écran se faisaient par simple projection. Maintenant, l'objet va rester fixe et c'est l'observateur qui va bouger.

L'objet observé sera défini exactement comme ci-avant, inutile de redétailler.

Par contre, il est indispensable de préciser comment la position de l'observateur est prise en compte et ce que signifie "regarder un objet sous un certain angle".

Position de l'observateur

La position de l'observateur sera repérée simplement par les coordonnées (XO, YO, ZO) du point où il se trouve dans l'espace. Ces coordonnées sont considérées dans le même repère que celles de l'objet observé. Ainsi, pour observer le cube :

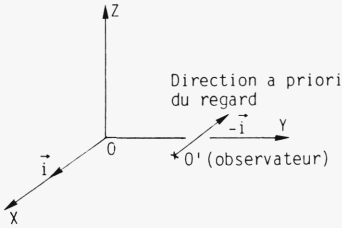


- de face : se placer en (X;5;5) avec X > 1Ø ;
- de dessus : se placer en (5;5;Z) avec Z > 1Ø ;
- de droite : se placer en (5;Y;5) avec Y > 1Ø.

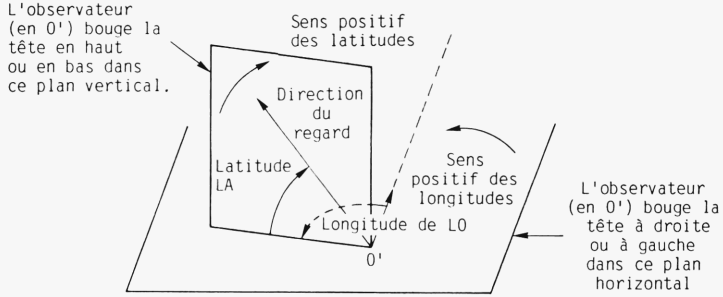
L'observateur peut même se placer, s'il le désire, à l'intérieur du cube (en (X,Y,Z) avec Ø < X < 1Ø et Ø < Y < 1Ø et Ø < Z < 1Ø) !

Du point où il se trouve, l'observateur peut regarder n'importe où, comment préciser la direction de son regard ? Nous allons choisir un moyen facile à comprendre. Nous pouvons bouger la tête à droite ou à gauche, en restant dans un plan horizontal. Nous pouvons aussi bouger la tête en haut ou en bas, en restant dans un plan vertical. La combinaison de ces deux mouvements autorise à regarder n'importe où.

Précisons encore. A priori, l'observateur regarde droit devant lui, selon la direction inverse de OX, celle du vecteur $-\vec{i}$.



La direction du regard est déterminée par deux angles. Par analogie au repérage d'un point sur une sphère, nous les appellerons longitude et latitude. Ces deux angles sont orientés. Un schéma sera plus explicite que de longues phases.



Ces angles seront introduits par l'utilisateur en degrés, le programme les convertira en radians. Ainsi, pour regarder :

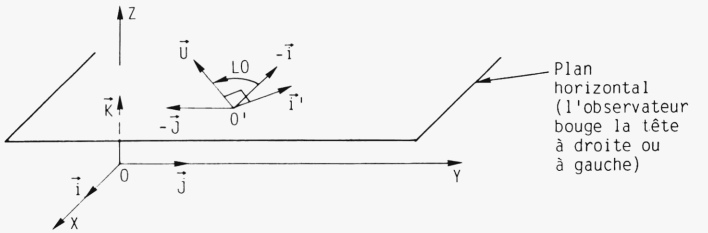
- droit devant : longitude = Ø, latitude = Ø
- droit derrière : longitude = 18Ø, latitude = Ø
- au-dessus : longitude = Ø, latitude = 9Ø
- au-dessous : longitude = Ø, latitude = -9Ø
- à gauche : longitude = 9Ø, latitude = Ø
- à droite : longitude = -9Ø, latitude = Ø

Dans la programmation, il sera indispensable de savoir passer du repère de départ (axes OX, OY, OZ, les seuls dont il a été question jusqu'à présent) à un repère dont l'origine est la position de l'observateur. Quel repère choisir ? Plusieurs choix sont possibles. Aucun ne s'impose plus qu'un autre.

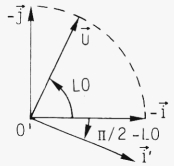
Dans toute la suite, tous les vecteurs considérés seront unitaires (longueur 1) et nous noterons :

- $\vec{U}$: le vecteur de la direction du regard lorsque l'on ne tient compte que de la longitude (l'observateur n'a bougé la tête qu'à droite ou à gauche) ;
- $\vec{I}'$: le vecteur du même plan horizontal que U, perpendiculaire à U, dirigé vers la droite de l'observateur.

Une figure sera peut-être plus claire.



Des calculs trigonométriques simples donnent :



$\vec{U} = -\vec{I}\ COS(LO) - \vec{J}\ SIN(LO)$

$\vec{I}' = -\vec{I}\ COS(LO-\pi/2) - \vec{J}\ SIN\ (LO-\pi/2)$
$= -\vec{I}\ SIN(LO) + \vec{J}\ COS(LO)$

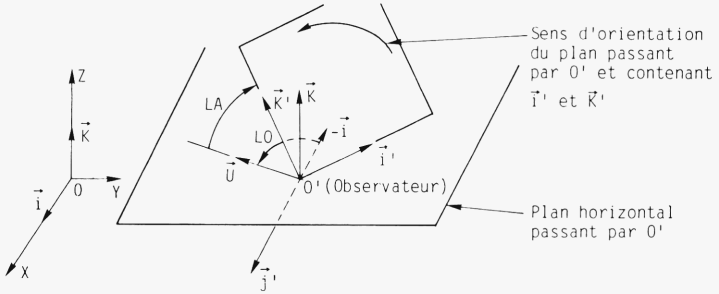
Notons enfin :

$\vec{K}'$: le vecteur définissant la direction du regard, une fois tenu compte de la longitude et de la latitude (c'est-à-dire des deux mouvements possibles) ;

$\vec{J}'$: pour le définir, orientons le plan déterminé par $\vec{K}'$ et $\vec{I}'$, de $\vec{K}'$ vers $\vec{I}'$. Nous avons vu (page 177) qu'alors toute perpendiculaire au plan $\vec{K}'$, $\vec{I}'$ a une certaine orientation ... définie par $\vec{J}'$.

Relisez attentivement cette définition. Les matheux auront reconnu en $\vec{J}'$ un vulgaire produit vectoriel ($\vec{J}' = \vec{K}' \wedge \vec{I}'$) et en ($\vec{I}'$, $\vec{J}'$, $\vec{K}'$) une vulgaire base orthonormée directe. Si, même après relecture, vous n'avez pas très bien compris, ce n'est pas trop grave. Le problème est : nous avons deux vecteurs $\vec{K}'$ et $\vec{I}'$ perpendiculaires et nous voulons en construire un troisième qui leur soit perpendiculaire. Il n'y a que deux possibilités (correspondant aux deux orientations possibles d'un axe)... Nous en avons choisi une, il fallait bien le faire.

Là encore, un dessin vaut mieux que de trop longues explications :



Des calculs trigonométriques fournissent :

$\vec{K}' = \vec{U}\ COS(LA) + \vec{K}\ SIN(LA)$
$= -\vec{I}\ COS(LA)\ COS(LO)$
$-\vec{J}\ COS(LA)\ SIN(LO)$
$-\vec{K}\ SIN(LA)$

Puis $\vec{J}' = \vec{K}' \wedge \vec{I}' = -\vec{I}\ SIN(LA)\ COS(LO) - \vec{J}\ SIN(LA)\ SIN(LO) - \vec{K}\ COS(LA)$ (ouf !). L'obtention de ce vecteur $\vec{J}'$ sera précisée page 227, lors de la définition du produit vectoriel. Nous préférons, pour l'instant, admettre ce résultat et insister sur ce qui est indispensable : formules de changement de repère et observation sous un certain angle.

Cette question peut se formuler ainsi : si un point M a pour coordonnées (X,Y,Z) dans le repère d'origine O, de vecteurs $\vec{I}, \vec{J}, \vec{K}$, quelles sont ses coordonnées (X',Y',Z') dans le repère ayant pour origine la position O' de l'observateur et pour vecteurs $\vec{I}', \vec{J}', \vec{K}'$?

Rappelons que :

$\vec{I}' = -\vec{I}\ SIN(LO) + \vec{J}\ COS(LO)$

$\vec{J}' = -\vec{I}\ SIN(LA)\ COS(LO) - \vec{J}\ SIN(LA)\ SIN(LO) - \vec{K}\ COS(LA)$

$\vec{K}' = -\vec{I}\ COS(LA)\ COS(LO) - \vec{J}\ COS(LA)\ SIN(LO) + \vec{K}\ SIN(LA)$

On peut en déduire qu'inversement :

$\vec{I} = -\vec{I}'\ SIN(LO) - \vec{J}'\ SIN(LA)\ COS(LO) - \vec{K}'\ COS(LA)\ COS(LO)$

$\vec{J} = \vec{I}'\ COS(LO) - \vec{J}'\ SIN(LA)\ SIN(LO) - \vec{K}'\ COS(LA)\ SIN(LO)$

$\vec{K} = -\vec{J}'\ COS(LA) + \vec{K}'\ SIN(LA)$

Nous ne détaillerons pas ces calculs. Ils sont lourds, mais simples (il est préférable d'utiliser U pour les alléger un peu).

A partir de cela, on peut écrire :

$\vec{OM} = \vec{OO'} + \vec{O'M}$ (relation de Chasles)

sous la forme :

$\vec{X}\ I + Y\ \vec{J} + Z\ \vec{K} = XO\ \vec{I} + YO\ \vec{J} + ZO\ \vec{K} + X'\ \vec{I}' + Y'\ \vec{J}' + Z'\ \vec{K}'$

ou bien :

$X'\ \vec{I}' + Y'\ \vec{J}' + Z'\ \vec{K}' = (X-XO)\ \vec{I} + (Y-YO)\ \vec{J} + (Z-ZO)\ \vec{K}$

Enfin, en remplaçant $\vec{I}, \vec{J}, \vec{K}$ par les expressions trouvées quelques lignes plus haut, on parvient à obtenir X', Y' et Z' en fonction de X, Y, Z, XO, YO, ZO, LG, LT !

Lecteurs non ou peu matheux, vous avez le droit à toutes nos excuses. Nous sommes conscients que ces pages ne sont pas faciles à lire. Nous ne donnons pas tous les détails et n'insistons volontairement que sur ce qui sera utile dans la rédaction du programme. C'est le fait que X', Y', Z' se calculent par les formules :

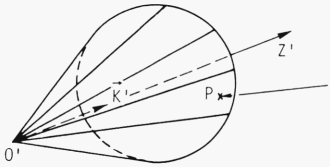
```
X' = -(X-XO) SIN(LO) + (Y-YO) COS(LO)
Y' = -(X-XO) SIN(LA) COS(LO) - (Y-YO) SIN(LA) SIN(LO) - (Z-ZO) COS(LA)
Z' = -(X-XO) COS(LA) COS(LO) - (Y-YO) COS(LA) SIN(LO) + (Z-ZO) SIN(LA)
```

Que vous ayez compris ou non le détail des calculs, si vous tapez le programme d'observation sous un certain angle, vous utiliserez ces formules. Autant savoir comment elles sont obtenues.

Nous savons ainsi passer du repère initial au repère de l'observateur. Mais que signifie l'expression "regarder sous un certain angle ?".

Observation sous un certain angle

Nous, observateur, sommes donc en un point O' et regardons dans une direction définie par le vecteur $\vec{K}'$. Regarder sous un angle OU c'est ne s'intéresser qu'aux points P' tels que l'angle entre $\vec{K}'$ et $\vec{O'P}$ soit, au plus, OU,



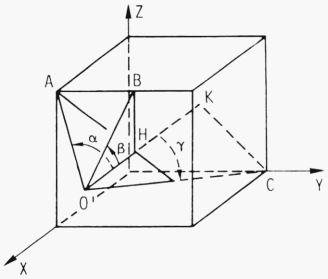
ce qui revient à limiter notre champ de vision aux points intérieurs au cône de révolution d'axe O'Z', de demi-angle au sommet OU que nous appellerons "ouverture".

Au niveau photographique :

OU petit (<1Ø) revient à utiliser un téléobjectif.
OU voisin de 5Ø revient à la vision normale.
OU voisin de 8Ø revient à utiliser un grand angle.

Nous n'accepterons pas OU>=9Ø (effet fish-eye) ce qui permet de regarder derrière soi ! Dans ce cas, il faut tenir compte d'une distorsion des droites - elles doivent être visualisées par des courbes - et nous aurons assez de travail sans cela.

Prenons l'exemple de notre cube. Plaçons-nous en O' (15,5,5) et regardons-le de face (LG=LT=Ø) :

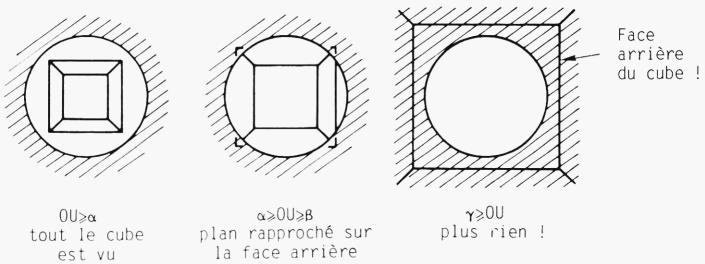


HA =K=5√2
O'H=HB=5
O'K=15

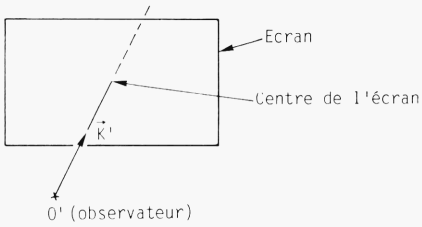
On a facilement :

TAN(α)=5 √2/5 d'où α ≃ 54° 44
TAN(β)=5/5 d'où β = 45°
TAN(γ)=5√2/15 d'où γ ≃ 25° 15

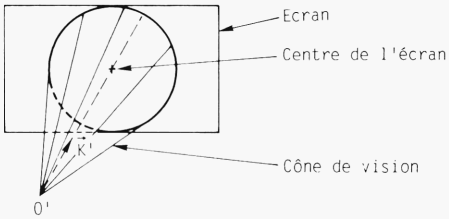
Observer le cube sous un angle supérieur à α permet de le voir complètement. Pour un angle compris entre α et β, seule la face de derrière est totalement vue ; toutes les autres (face de devant, faces latérales, faces haut et bas) ne sont vues que partiellement. Pour un angle inférieur à γ, aucun sommet du cube n'est vu.



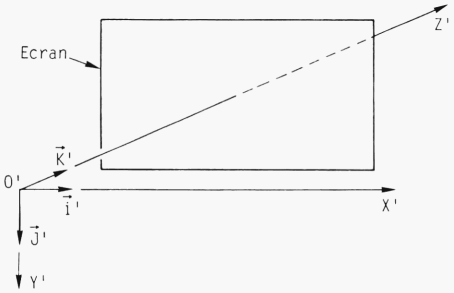
Comment tenir compte de tout cela sur l'écran de notre machine ? Pas évident a priori ! Nous regardons à la fois l'objet - et en tant que tel nous regardons dans la direction de $\vec{K'}$ - et l'écran de notre ordinateur sur lequel nous représentons cet objet - et en tant que tel nous regardons (en général) le centre de l'écran - il est donc normal de nous situer ainsi.



Regardant l'objet sous l'angle OU, il est normal de dire que notre cône de vision a la position suivante par rapport à l'écran :

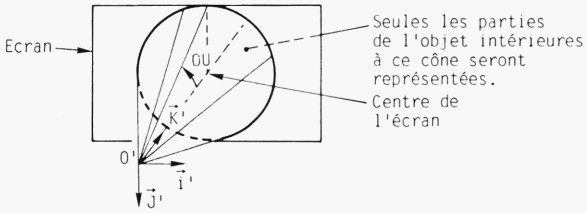


Enfin, notre vecteur $\vec{I'}$ est horizontal ($\vec{I'}$ est défini lorsque nous bougeons la tête à droite ou à gauche, pas en haut ou en bas) et sur la gauche de l'observateur. $\vec{J'}$, lui, est alors nécessairement vertical. Est-il dirigé vers le haut ou vers le bas ? Le choix que nous avons fait ($\vec{J'}=\vec{K'} \wedge \vec{I'}$) impose à $\vec{J'}$ d'être orienté vers le bas.

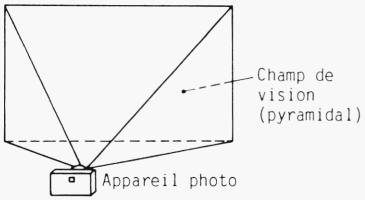


Bien sûr, ce repère n'a pas une allure très classique. Comme nous avions le choix, nous avons fait en sorte que nos axes O'X' et O'Y' aient le même sens d'orientation que les axes en X et en Y des écrans de la plupart des machines. C'est ce qui nous a guidé pour le choix de $\vec{I}'$ et de $\vec{J}'$... Mais, nous pouvions difficilement l'expliquer sans lier ces choix à la visualisation.

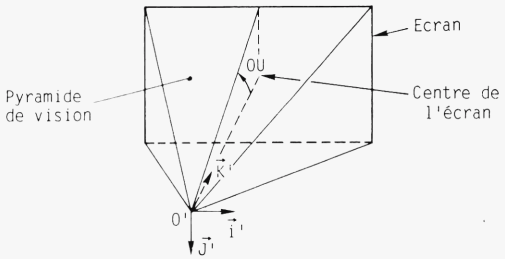
Avant d'aborder la représentation proprement dite, il nous faut revenir sur le cône de vision. D'après ce qui précède, nous ne représenterions sur l'écran que la partie de l'objet intérieure au cône d'axe O'Z', de demi-angle au sommet OU.



Si nous reprenons l'analogie avec la photographie, vous admettrez qu'il est rare de n'obtenir d'image qu'à l'intérieur d'un cercle. L'image est rectangulaire, parfois carrée. En photographie, le champ de vision n'est pas conique, mais pyramidal :



De même, ici, nous remplacerons notre cône de vision par une pyramide de vision :



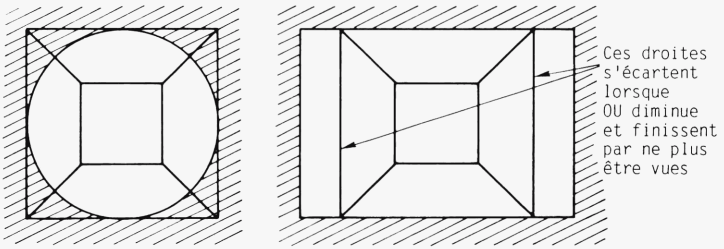
Notre angle d'ouverture OU représente donc la moitié de l'angle sous lequel on regarde les Y écran. Les X seront vus sous un angle un peu plus grand.

Remarquons que, vis-à-vis de l'écran, en prenant pour unité de longueur celle des Y écran - encore une fois, c'est la plus naturelle - l'observateur se trouve à une distance D :

D = HT/2/TAN(OU)

car TAN(OU) = HT/2/D.

Adopter une pyramide de vision à la place d'un cône présente l'avantage d'utiliser tout l'écran de l'ordinateur, donc de donner le plus de détails possible. Pour notre cube observé sous un angle OU compris entre α et β, les représentations sont :



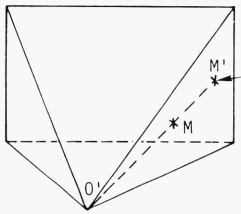
Avec un cône de vision

Avec une pyramide de vision

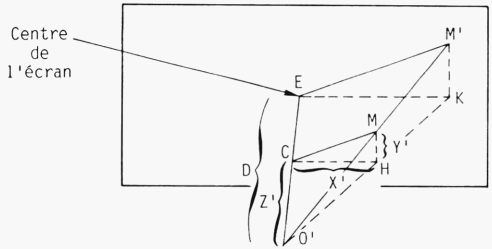
Représentation sur l'écran

Des formules assez compliquées nous permettent de connaître les coordonnées (X',Y',Z') d'un point M dans le repère $(O',\vec{I'},\vec{J'},\vec{K'})$ lorsque l'on connaît ses coordonnées (X,Y,Z) dans le repère $(O,\vec{I},\vec{J},\vec{K})$ et les angles LA et LO.

Reste à répondre à deux questions très liées : où ce point ne sera-t-il représenté sur l'écran et comment tenir compte de notre pyramide de vision ?



Répondons à une première question : comment déterminer M' lorsque l'on connaît M ? Une petite difficulté provient, comme toujours, du fait que les axes sur l'écran n'ont pas les mêmes unités. Sans en tenir compte, nous aurions :



$$\frac{\overline{EK}}{\overline{CH}} = \frac{D}{Z'} \text{ et } \frac{\overline{KM'}}{\overline{HM}} = \frac{D}{Z'}$$

On en déduit :

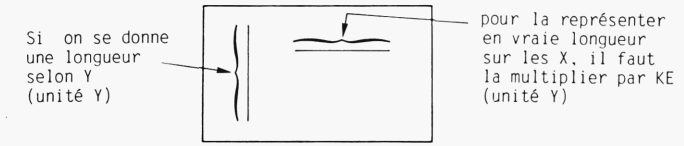
$\overline{EK}$ = X'*D/Z' et $\overline{KM'}$ = Y'*D/Z'

E, centre de l'écran, a pour coordonnées écran XC et YC. On serait donc tenté de dire que celles de M' sont :

XC+X'*D/Z' et YC+Y'*D/Z'

Pour les Y, pas de problème puisque D est calculé en prenant pour unité celles des Y... mais pour XC+X'*D/Z' ?

Nous avons vu (page 186 pour le tracé du plus grand carré possible à l'écran) que :



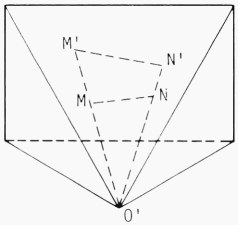
Il se passe le même phénomène pour EK : sur l'écran, EK est représenté par X'*D*KE/Z' (et non pas par X'*D/Z').

Bilan : le point M(X',Y',Z'), dans le repère $(O',\vec{I},\vec{J},\vec{K})$ est représenté sur l'écran par le point :

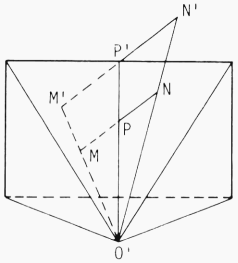
M'(XC+X'*D*KE/Z',YC+Y'*D/Z')

ce qui suppose évidemment Z' non nul.

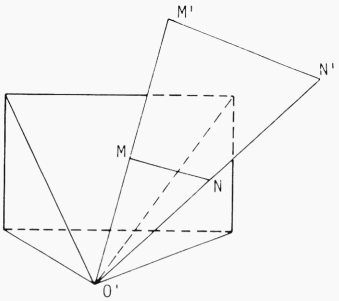
Obtenir le point M' est une bonne chose. N'oublions pas que notre objet est défini par des segments et que, par rapport à notre pyramide de vision, un segment peut se comporter de façons différentes :



Si le segment MN est intérieur à la pyramide, pas de problème : M'N' est vu

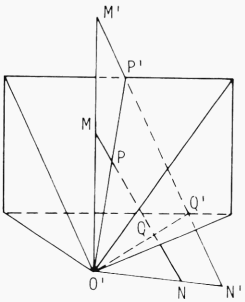


Si M est intérieur et P est extérieur à la pyramide, MN la coupant en P, seul M'P' est vu sur l'écran



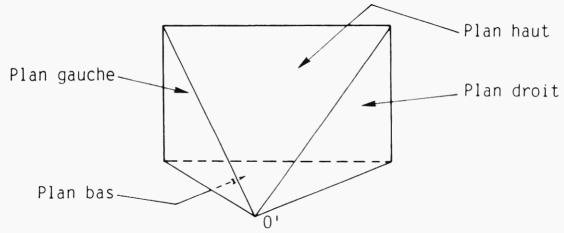
Si le segment MN est extérieur à la pyramide, il ne peut pas la couper alors M'N' est invisible (hors écran)

ou bien



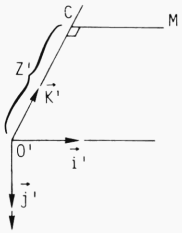
MN peut la couper en deux points P et Q et alors, seule la partie P'Q' de M'N' est vue !

Pour plus de clarté, nous parlerons de plans haut, bas, droite, gauche :



Pour un point M(X',Y',Z'), être à l'intérieur de la pyramide de vision, c'est :

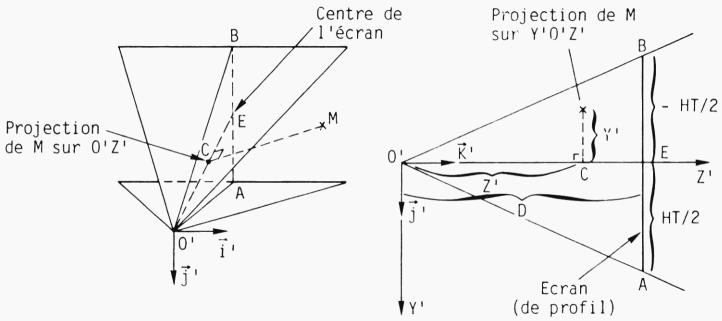
- d'abord être "devant O'" :



Etre devant O' c'est avoir Z' positif :

$Z' > 0$

- ensuite, entre les plans haut et bas :



Etre entre ces plans, pour M', c'est se projeter sur Y'OZ' en un point situé à l'intérieur de l'angle AÔ'B. Les droites O'A et O'B ayant pour équation respective :

```
      Y'/Z'=HT/2/D    et    Y'/Z' = -HT/2/D
soit  2*D*Y' = HT*Z'  et    2*D*Y'= -HT*Z'
```

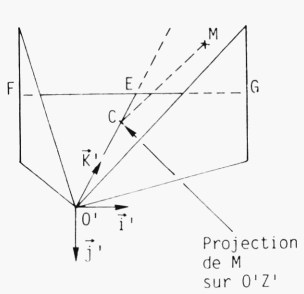
La condition cherchée est :

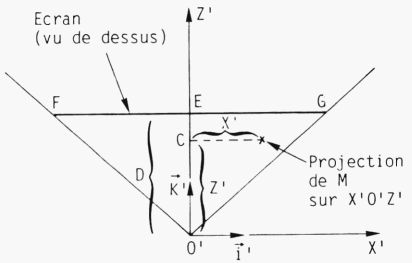
```
      2*D*Y'<=HT*Z'  et  2*D*Y'>=-HT*Z'
```

ou encore :

```
      |2*D*Y'|<= |HT*Z'|
```

- enfin, être entre les plans droit et gauche :





Pour M', être entre ces plans, c'est se projeter sur X'O'Z' en un point situé à l'intérieur de l'angle FÔ'G. Nous serions tentés de dire que :

```
      EF = EG = LG/2
```

Ce serait oublier que l'unité qui s'est imposée est celle des Y ! Toujours le même problème. Là, la démarche est l'inverse de la précédente. Une longueur nous est donnée, selon les X (LG/2). Comment la convertir selon les Y ? Dans le cas précédent, il fallait multiplier par KE ; ici, il faut donc diviser par KE et :

```
      EF = EG = LG/KE/2.
```

(en unité Y écran !). Les droites O'F et O'G ont donc pour équation respective :

```
      X'/Z' = LG/KE/2/D   et   X'/Z' = -LG/KE/2/D
soit  2*KE*D*X' = LG*Z'   et   2*KE*D*X' = -LG*Z'
```

La condition cherchée est :

```
      2*KE*D*X'<= LG*Z'   et   2*KE*D*X'>=-LG*Z'
```

ou encore :

```
      |2*KE*D*X'|<= |LG*Z'|
```

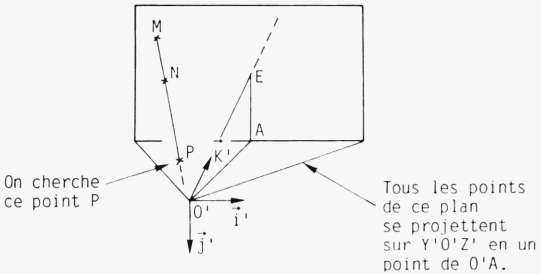
Récapitulons : le point M (X',Y',Z') est intérieur à la pyramide de vision lorsque :

```
      Z' >= Ø
et    |2*D*Y'| ≤ |HT*Z'|
et    |2*KE*D*X'| ≤ |LG*Z'|
```

La première des deux questions se trouve résolue.

Pour répondre à la seconde, nous n'envisagerons que l'intersection d'un segment avec le plan bas. Les autres cas sont analogues.

Au lieu de chercher l'intersection du segment MN avec ce plan, nous allons déterminer l'intersection de la droite MN.



Appartenir au plan bas, c'est se projeter sur Y'O'Z' en un point de la droite O'A. Nous avons vu que c'est vérifier :

```
2*D*Y' = HT*Z'
```

Cette relation entre les coordonnées X', Y', Z' d'un point s'appelle l'équation du plan (bas).

Dans toute la suite, nous noterons (XM,YM,ZM) et (XN,YN,ZN) les coordonnées de M et N. La droite MN coupe notre plan en P. On doit avoir :

$$\vec{MP} = \alpha \vec{MN}$$

et les coordonnées XP, YP, ZP de P sont de la forme :

```
XP=XM+α*(XN-XM), YP=YM+α*(YN-YM), ZP=ZM+α*(ZN-ZM)
```

P devant appartenir au plan bas, YP et ZP vérifient la condition du type 2*D*Y'=HT*Z' :

```
      2*D*[YM+α*(YN-YM)] = HT*[ZM+α*(ZN-ZM)]
soit  α*[2*D*(YN-YM)-HT*(ZN-ZM)] = HT*ZM-2*D*YM
```

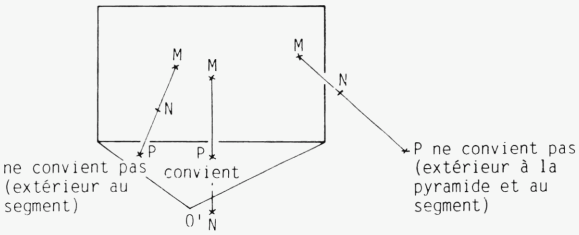
ce qui nous permet de calculer α. En reportant cette valeur de α dans les égalités XP=..,YP=..,ZP=.. vues plus haut, nous obtenons les coordonnées de P.

Si 2*D*(YN-YM)-HT*(ZN-ZM) est nul, on ne peut plus calculer α. On montre, faites-nous confiance sur ce point, qu'alors la droite MN est parallèle au plan bas.

Comment utiliser ce point P ? Il n'a d'intérêt que si M et N ne sont pas tous deux intérieurs à la pyramide de vision. Sinon, nous avons vu comment déterminer les points M' et N' : il suffit alors de tracer le segment M'N' sur l'écran.

Si donc M' et N' ne sont pas tous deux intérieurs à la pyramide de vision, seule, éventuellement, une partie du segment M'N' est vue. P sera une extrémité de cette partie à deux conditions :

- P appartient au segment MN ;
- P est intérieur à la pyramide de vision.



La réponse à ces deux questions est simple. Comme $\vec{MP} = \alpha \vec{MN}$, P sera entre M et N si $0 < \alpha < 1$. Quant à être intérieur à la pyramide de vision, il s'agit d'une question déjà résolue. Ouf !

Nous avons maintenant tout ce qui est indispensable à la représentation. Remettons en ordre tous les éléments de ce puzzle.

Lorsque seule une partie d'un segment MN est visible, l'important est de déterminer les extrémités de ce segment. F désignant le nombre d'extrémités trouvées pour ce segment, il suffit de procéder ainsi :

- initialiser F à 0.
- M est-il intérieur à la pyramide ?
  Si oui : F augmente de 1 et M' convient, sinon on continue.
- N est-il intérieur à la pyramide ?
  Si oui : F augmente de 1 et N' convient, sinon on continue.
- F est-il égal à 2 ?
  Si oui, on trace M'N' et on arrête, sinon on continue.
- La droite MN coupe-t-elle le plan haut ?
  Si oui : le point P correspondant convient-il ?
  (a-t-on $0<\alpha<1$ et P intérieur à la pyramide ?)
  Si oui : F augmente de 1 et P' convient.
  F est-il égal à 2 ?
  Si oui : on trace le segment (M'P' ou N'P') et on arrête,
  sinon on continue.

Le "on continue" signifie : on effectue le même processus avec les autres plans (bas, droite, gauche). Le "on arrête" signifie lui : on arrête ce processus. Ce processus doit être répété pour chaque segment de couleur 1 de l'objet. Bien sûr, la meilleure rédaction serait un tant que... mais peu de Basics ont cette possibilité. Nous ferons donc des tests nombreux et lourds. La façon de réaliser chacun des tests a été détaillée ci-dessus. Reste la traduction en Basic.

Programmation

Par un heureux hasard, le programme reprend ce que nous venons de voir. Sa structure est la suivante :

- sous-programme de lecture des DATAS définissant l'objet (à partir de la ligne 1000) ; c'est le même que le précédent ;
- sous-programme demandant les conditions d'observations (position de l'observateur, direction du regard) (à partir de la ligne 4000) ;
- représentation sur l'écran puis retour à l'étape 2 (à partir de la ligne 3000).

Certaines précisions sont indispensables :

- les tableaux X(), Y(), Z() contiennent les coordonnées des sommets de l'objet dans le repère de départ $(O,\vec{I},\vec{J},\vec{K})$. Voir page 180 si votre mémoire est défaillante. Ils ne changent pas au cours du programme.

  Par contre, les tableaux X1(), Y1(), Z1() qui eux contiennent les coordonnées des mêmes sommets, dans le repère $(O',\vec{I}',\vec{J}',\vec{K}')$ (voir page 205) changent à chaque déplacement de l'observateur. Leur mise à jour est assurée par le sous-programme commençant à la ligne 2ØØØ qui reprend les formules épouvantables de la page 206 ;

- les extrémités de la partie éventuellement vue du segment MN sont stockées dans un tableau U() :

  U(1) = abscisse écran
  U(2) = ordonnée écran
  } de la première extrémité visible

  U(3) = abscisse écran
  U(4) = ordonnée écran
  } de la seconde extrémité visible

- certaines écritures ont été allégées. L'introduction de :

  DH = HT/2 et DL = LG/2/KE

  permet d'écrire plus rapidement les textes de visibilité puisque :

  |2*D*Y'| <= |HT*Z'| s'écrit |D*Y'| <= |DH*Z'|

  et |2*D*KE*X'|<= |LG*Z'| s'écrit |D*X'| <= |DL*Z'|

  Autre avantage : les conditions s'écrivent alors sous des formes analogues.

  Ce changement se répercute évidemment sur les expressions de L, variable qui, dans le programme, remplace celle que nous avons appelée α.

  Nous avons même pris, pour DH et DL, des parties entières, ce qui restreint un peu la pyramide de vision.

- le passage d'un plan haut à un plan bas, ou d'un plan droit à un plan gauche, s'effectue par les simples instructions :

  DH = -DH et DL = -DL

  (lignes 313Ø, 317Ø, 381Ø, 391Ø). Il suffit en effet de changer HT ou LG en son opposé dans les formules, ce qui revient à changer le signe de DH en DL. ;

- la position de l'observateur est initialisée à :

  XØ = 2Ø, YØ = Ø, ZØ = Ø (ligne 4Ø)

  Elle convient pour les objets donnés.

  La visée est "normale" puisque :

  LO = Ø, LA = Ø (observation de face)
  OU = 45

(ligne 5Ø)

Enfin, quelques remarques concernant l'utilisation du programme. Tous les objets déjà définis sont utilisables : le stockage est identique. Le passage d'une représentation à une manipulation s'effectue en tapant sur la touche retour-chariot (CHR$(13)). L'affichage de la nouvelle position et de la visée subsiste pendant les calculs de changement de repère. Puis, l'écran est effacé pour passer à la représentation.

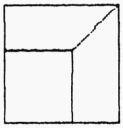
```
10 REM OBJETS EN DIMENSION 3 AVEC DEPLACEMENT
   DE L'OBSERVATEUR
20 LG=    :HT=
30 KE=    :PI=4*ATN(1)
40 X0=20:Y0=0:Z0=0:REM POSITION DE DEPART DE
   L'OBSERVATEUR
50 LO=0 LA=0:OU=45:REM OBSERVATION DE FACE
60 XC=    :YC=
70 DL=INT(LG/2):DH=INT(HT/2):DL=DL/KE
80 DIM U(4):REM COORDONNEES DU SEGMENT TRACE
100 GOSUB 1000:REM LECTURE DES DATAS
110 GOSUB 2000:REM CHANGEMENT DE REPERE
120 GOSUB 3000:REM VISUALISATION
130 GOSUB 4000:REM AFFICHAGE DE LA POSITION
140 GOTO 110
1000 REM LECTURE D'UN OBJET EN DATA
1010 READ S:REM NOMBRE DE SEGMENTS
1020 DIM X(S),Y(S),Z(S),COU(S):REM TABLEAU
     PRINCIPAL
1030 DIM X1(S),Y1(S),Z1(S)     :REM TABLEAU
     AUXILIAIRE
1040 FOR I=0 TO S
1050 READ X(I),Y(I),Z(I),COU(I)
1060 NEXT I
1070 RETURN
2000 REM CHANGEMENT DE REPERE
2010 CO=COS(LO*PI/180):SO=SIN(LO*PI/180)
2020 CA=COS(LA*PI/180):SA=SIN(LA*PI/180)
2030 FOR I=0 TO S
2040 X=X(I)-X0
2050 Y=Y(I)-Y0
2060 Z=Z(I)-Z0
2070 X1(I)=-X*SO+Y*CO
2080 Y1(I)=-X*SA*CO-Y*SA*SO-Z*CA
2090 Z1(I)=-X*CA*CO-Y*CA*SO+Z*SA
2100 NEXT I
2110 RETURN
3000 REM VISUALISATION
3010 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
3020 D=DH/TAN(OU*PI/180)
3030 FOR I=0 TO S-1
3040 IF COU(I+1)=0 THEN 3200
3050 F=0
3060 DX=X1(I+1)-X1(I)
3070 DY=Y1(I+1)-Y1(I)
3080 DZ=Z1(I+1)-Z1(I)
3090 X=X1(I):Y=Y1(I):Z=Z1(I):GOSUB 3500
3100 X=X1(I+1):Y=Y1(I+1):Z=Z1(I+1):GOSUB 3500
3110 IF F=2 THEN GOSUB 3700:GOTO 3200
3120 GOSUB 3800
3130 IF F=2 THEN DL=-DL:GOSUB 3700:GOTO 3200
3140 GOSUB 3800
3150 IF F=2 THEN GOSUB 3700:GOTO 3200
3160 GOSUB 3900
3170 IF F=2 THEN DH=-DH:GOSUB 3700:GOTO 3200
3180 GOSUB 3900
3190 IF F=2 THEN GOSUB 3700
3200 NEXT I
3210 A$=INKEY$:REM OU GET A$
3220 IF A$<>CHR$(13) THEN 3210
3230 RETURN
3500 REM INTERSECTION OUI-NON ?                →
```

```
3510 IF Z<0 OR ABS(D*Y)>ABS(DH*Z) OR
     ABS(D*X)>ABS(DL*Z) THEN RETURN
3520 GOSUB 3600
3530 RETURN
3600 REM MISE EN TABLEAU
3610 F=F+1
3620 IF Z=0 THEN U(2*F-1)=XC:U(2*F)=YC:RETURN
3630 U(2*F-1)=KE*D*X/Z+XC
3640 U(2*F)=D*Y/Z+YC
3650 RETURN
3700 REM TRACE
3710 DROITE U(1),U(2) A U(3),U(4)
3720 RETURN
3800 REM PLANS DROITE ET GAUCHE
3810 DL=-DL
3820 IF D*DX+DL*DZ=0 THEN RETURN
3830 L=-(D*X1(I)+DL*Z1(I))/(D*DX+DL*DZ)
3840 IF L<0 OR L>1 THEN RETURN
3850 X=X1(I)+L*DX:Y=Y1(I)+L*DY:Z=Z1(I)+L*DZ
3860 IF Z<0 OR ABS(D*Y)>ABS(DH*Z) THEN RETURN
3870 GOSUB 3600
3880 RETURN
3900 REM PLANS HAUT ET BAS
3910 DH=-DH
3920 IF D*DY+DH*DZ=0 THEN RETURN
3930 L=-(D*Y1(I)+DH*Z1(I))/(D*DY+DH*DZ)
3940 IF L<0 OR L>1 THEN RETURN
3950 X=X1(I)+L*DX:Y=Y1(I)+L*DY:Z=Z1(I)+L*DZ
3940 IF L<0 OR L>1 THEN RETURN
3950 X=X1(I)+L*DX:Y=Y1(I)+L*DY:Z=Z1(I)+L*DZ
3960 IF Z<0 OR ABS(X*D)>ABS(DL*Z) THEN RETURN
3970 GOSUB 3600
3980 RETURN
4000 REM OBSERVATEUR-VISEE
4010 REM MODE TEXTE,EFFACAGE D'ECRAN
4020 GOSUB 4700:REM AFFICHAGE DE LA POSITION
4030 PRINT"MODIFICATION DE LA POSITION (O/N)"
4040 A$=INKEY$:REM OU GET A$
4050 IF A$="O" THEN 4100
4060 IF A$="N" THEN 4210
4070 GOTO 4040
4100 PRINT"NOUVELLE POSITION:"
4110 INPUT"X=";X0
4120 INPUT"Y=";Y0
4130 INPUT"Z=";Z0
4200 REM MODIFICATION VISEE
4210 PRINT"MODIFICATION VISEE (O/N)"
4220 A$=INKEY$:REM OU GET A$
4230 IF A$="O" THEN 4300
4240 IF A$="N" THEN 4400
4250 GOTO 4220
4300 PRINT"NOUVELLE VISEE"
4310 INPUT"LONGITUDE =";LO
4320 INPUT"LATITUDE  =";LG
4330 INPUT"OUVERTURE =";OU
4340 IF OU<0 OR OU>90 THEN 4330
4400 GOSUB 4700
4410 RETURN
4700 REM AFFICHAGE POSITION ET VISEE
4710 PRINT"POSITION :"
4720 PRINT"X=";X0
4730 PRINT"Y=";Y0                              →
```

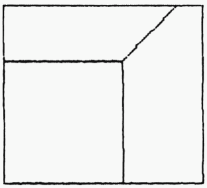
```
4740 PRINT"Z=";Z0
4750 PRINT
4760 PRINT"VISEE :"
4770 PRINT"LONGITUDE=";LO
4780 PRINT"LATITUDE =";LA
4790 PRINT"OUVERTURE=";OU
4800 RETURN
10000 REM DATA CUBE
10010 DATA 15:REM NOMBRE DE SEGMENTS
10020 DATA 0,0,0,0
10030 DATA 10,0,0,1
10040 DATA 10,10,0,1
10050 DATA 0,10,0,1
```

Ne soyez pas étonné de certains résultats. Parfois, l'écran reste vide. Comme seules les arêtes de l'objet sont tracées, si on l'observe de près, sous un angle faible, on risque de ne voir aucune arête. Parfois, l'écran est illisible. Un objet un peu compliqué, icosaèdre ou dodécaèdre, observé de loin, fournit un dessin petit, embrouillé. Une seule méthode, dans ce cas : ne représenter que les parties vues.

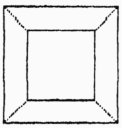
Exemple d'un cube



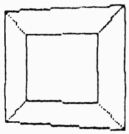
```
X=25 : Y=Ø : Z=Ø
LONGITUDE  :   Ø
LATITUDE   :   Ø
OUVERTURE  :  45
```



```
X=25 : Y=Ø : Z=Ø
LONGITUDE  :   Ø
LATITUDE   :   Ø
OUVERTURE  :  3Ø
```

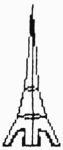


```
X=25 : Y=5 : Z=5
LONGITUDE   :   Ø
LATITUDE    :   Ø
OUVERTURE   :  45
```

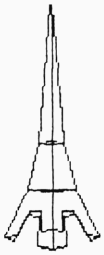


```
X=25 : Y=5 : Z=5
LONGITUDE   :  1Ø
LATITUDE    :   Ø
OUVERTURE   :  45
```

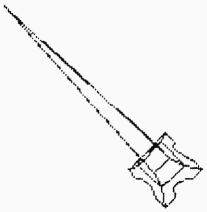
Sans commentaire !



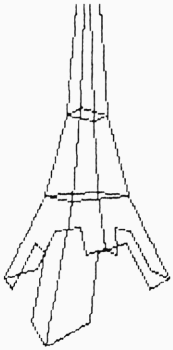
```
X=25 : Y=Ø : Z=Ø
LONGITUDE   :   Ø
LATITUDE    :   Ø
OUVERTURE   :  45
```



X=25 : Y=Ø : Z=5
LONGITUDE : Ø
LATITUDE : Ø
OUVERTURE : 3Ø



X=2 : Y=2 : Z=22
LONGITUDE : Ø
LATITUDE : -8Ø
OUVERTURE : 4Ø



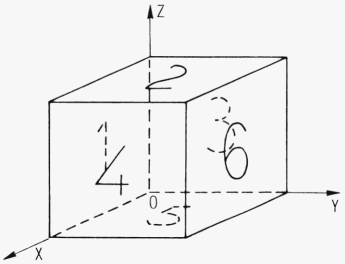
X=1Ø : Y=3 : Z=5
LONGITUDE : Ø
LATITUDE : Ø
OUVERTURE : 4Ø

PARTIES VUES, PARTIES CACHEES

Venons-en au troisième volet annoncé : comment, dans la représentation de l'objet sur l'écran, ne tenir compte que des parties vues ?

La résolution de ce problème, bien qu'assez simple, n'est pas évidente. Nous étudierons un exemple, celui d'un cube, afin de dégager la méthode. Cette méthode nécessite l'utilisation de deux outils mathématiques, produit scalaire et produit vectoriel : nous les définirons. Que les lecteurs non-matheux ne s'inquiètent pas trop, nous nous limiterons à ce qui s'avère indispensable à la résolution de notre problème. Pas question d'un cours magistral sur ces notions. Nous verrons enfin que notre méthode implique un stockage particulier des objets considérés. Ce n'est qu'après que nous pourrons envisager la programmation de ce type de représentation.

Exemple d'un cube

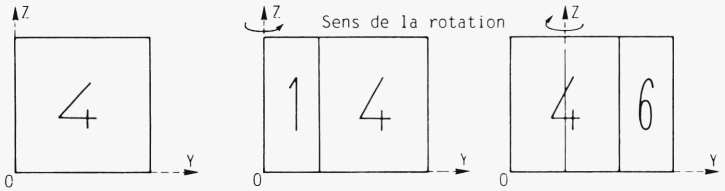


Reprenons donc le cube de la page 182. Pour plus de clarté, numérotons ses faces :

1 face de gauche
2 face de dessus
3 face de derrière
4 face de devant
5 face de dessous
6 face de droite

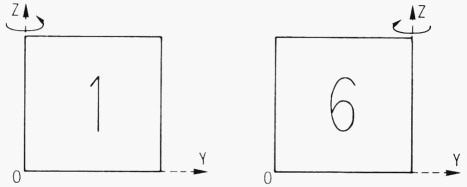
Contrairement à l'application précédente où l'observateur pouvait se placer n'importe où, y compris à l'intérieur de l'objet, ici, l'observateur sera obligatoirement à l'extérieur du cube.

Au départ, seule la face 4 est visible. Dès que l'on tourne un tout petit peu le cube autour de OZ, la face 1 ou la face 6 devient visible, selon le sens de la rotation effectuée.



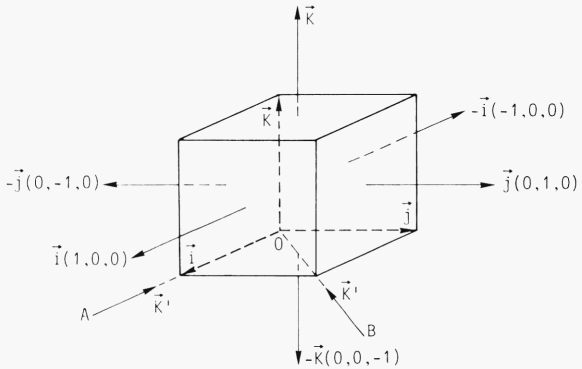
(dessins en projections sur YOZ).

Si nous continuons à effectuer de telles rotations, la face 1 (ou la face 6) va occuper la position de face, et la face 4 va devenir cachée :



Le problème des parties vues et cachées est sans doute un problème d'angle : les faces 2 et 5, qui restent horizontales, sont toujours cachées. Mais des faces parallèles doivent jouer un rôle différent, l'une pouvant être cachée, l'autre vue.

Compliquons notre figure et, à chaque face du cube, associons un vecteur qui lui soit perpendiculaire et orienté vers l'extérieur. Cette dernière condition est fondamentale. Prenons par exemple :



Analysons ce qui se passe lorsque l'observateur se place successivement aux points A, B et regarde dans la direction définie par le vecteur $\vec{K}'$:

- observateur en A, direction du regard $\vec{K}'=-\vec{I}$

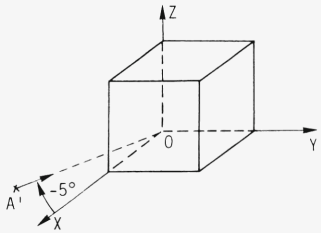
face 1 cachée	angle entre $\vec{K}'$ et $-\vec{J}$	90
face 2 cachée	angle entre $\vec{K}'$ et $\vec{K}$	90
face 3 cachée	angle entre $\vec{K}'$ et $-\vec{I}$	0
face 4 vue	angle entre $\vec{K}'$ et $\vec{I}$	180
face 5 cachée	angle entre $\vec{K}'$ et $-\vec{K}$	90
face 6 cachée	angle entre $\vec{K}'$ et $\vec{J}$	90

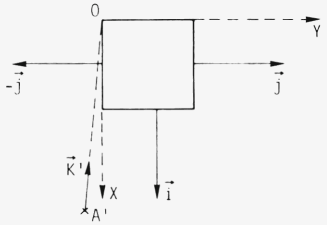
- observateur en B, direction du regard $\vec{K}'=-\vec{J}-\vec{I}$

face 1 cachée	angle entre $\vec{K}'$ et $-\vec{J}$	45
face 2 cachée	angle entre $\vec{K}'$ et $\vec{K}$	90
face 3 cachée	angle entre $\vec{K}'$ et $-\vec{I}$	45
face 4 vue	angle entre $\vec{K}'$ et $\vec{I}$	135
face 5 cachée	angle entre $\vec{K}'$ et $-\vec{K}$	90
face 6 vue	angle entre $\vec{K}'$ et $\vec{J}$	135

Précisons que ces angles, dans l'espace, ne sont pas orientés (nous n'avons orienté ni plan, ni perpendiculaire à ce plan, conformément à ce que nous avons dit page 177).

Comprenez-vous quel sera le critère vu-caché ? Peut-être pas tout à fait. Alors, plaçons-nous au point A' déduit de A par une rotation de -5 degrés, dans le plan XOY, et regardons vers le point O.





Que se passe-t-il ?

Face 1 vue	angle entre $\vec{K}'$ et $-\vec{J}$	95 (=9Ø+5)
Face 2 cachée	angle entre $\vec{K}'$ et $\vec{K}$	9Ø (inchangé)
Face 3 cachée	angle entre $\vec{K}'$ et $-\vec{I}$	5 (=Ø+5)
Face 4 vue	angle entre $\vec{K}'$ et $\vec{I}$	175 (=18Ø-5)
Face 5 cachée	angle entre $\vec{K}'$ et $-\vec{K}$	9Ø (inchangé)
Face 6 cachée	angle entre $\vec{K}'$ et $\vec{J}$	85 (=9Ø-5)

Ce que nous venons de traiter avec cinq degrés aurait pu l'être avec 2,1 degrés, avec un angle de quelques minutes ou de quelques secondes. Seules, certaines valeurs d'angles entre $\vec{K}'$ et certains vecteurs ($-\vec{J}$,$\vec{J}$,$-\vec{I}$,$\vec{I}$) auraient été changées ; la visibilité ou non d'une face n'aurait pas été modifiée.

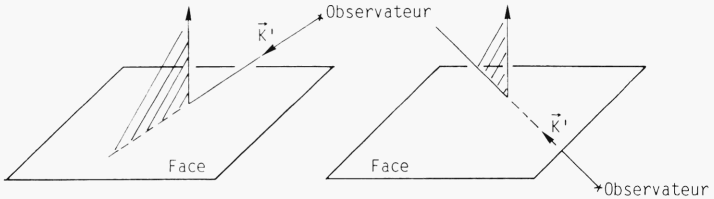
Cela montre que le test de visibilité d'une face peut s'énoncer ainsi : T étant l'angle déterminé par un vecteur perpendiculaire à une face et le vecteur définissant la direction dans laquelle regarde l'observateur :

si $0 < T < 9Ø$: la face est cachée ;
si $9Ø < T < 18Ø$: la face est vue.

Dans le cas où T=9Ø, la face est de profil, donc vue comme un segment. Il est inutile de la tracer puisque ce segment appartient aussi à une autre face qui, elle, est vue. Nous considérerons que, dans ce cas, la face est cachée.

Ce test revient à savoir si un angle est aigu (face cachée) ou obtus (face vue). Remarquons qu'on aurait pu avoir un résultat opposé si, à la place de considérer le vecteur définissant la direction du regard comme partant de l'oeil de l'observateur, nous l'avions considéré dans le sens opposé, donc aboutissant à l'oeil. L'important n'est pas de retenir "angle aigu, face cachée, angle obtus, face vue", mais de comprendre que c'est être aigu ou obtus qui importe... caché ou vu dépend alors des vecteurs considérés. Nous avons fait un choix d'orientation sur la direction du regard et un autre sur le sens des vecteurs perpendiculaires aux faces. Tout changement, parfaitement légitime, par rapport à ces choix, peut modifier l'angle qui permet de tester la visibilité d'une face. Un programmeur averti en vaut deux !

Pour nous, donc :



Angle obtus : face vue Angle aigu : face cachée

($\vec{K}'$: vecteur définissant la direction du regard).

Outils mathématiques : produit scalaire et produit vectoriel

Deux questions se posent naturellement. Comment réaliser pratiquement le test de visibilité ? Et comment obtenir des vecteurs perpendiculaires aux faces et dirigés vers l'extérieur de l'objet ?

Produit scalaire

Commençons par la question la plus simple, celle du test de visibilité. Elle peut se formuler autrement : comment savoir si un angle est aigu ou obtus ?

Sous cette forme, une réponse est facile à donner : il suffit de regarder le signe du cosinus de l'angle puisque :

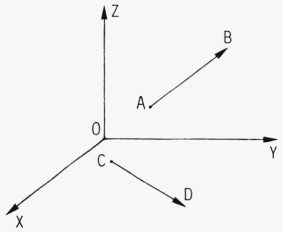
Cosinus négatif : angle obtus Cosinus positif : angle aigu

Nous avons vu (première partie, chapitre II) comment, dans un plan, calculer le cosinus de l'angle de deux vecteurs :

$$\text{COS } T = \frac{XX' + YY'}{\sqrt{X^2+Y^2}\ \sqrt{X'^2+Y'^2}}$$

Le nombre XX'+YY' s'appelle, dans le plan, produit scalaire des vecteurs de coordonnées (X,X') et (Y,Y'). Ces coordonnées ne sont pas des coordonnées écran (il faut alors introduire le coefficient KE, voir page 58), mais des coordonnées en repère orthonormé.

De même, dans l'espace, on définit le produit scalaire de deux vecteurs :



$\vec{AB}(X,Y,Z)$
$\vec{CD}(X',Y',Z')$
(en repère orthonormé) par :
$\vec{AB} . \vec{CD} = XX'+Y\ Y'+ZZ'$

On montre que ce nombre est le produit des longueurs AB et CD et du cosinus de leur angle :
$\vec{AB}.\vec{CD}=AB*CD*COS(\vec{AB},\vec{CD})$

Ce qui servira pour notre application est : le nombre XX'+YY'+ZZ' a le même signe que le cosinus de l'angle des vecteurs. Nous ne testerons donc que le signe de ce nombre, inutile de calculer l'angle. Bien entendu, X,X', Y,Y', Z,Z' seront remplacés, dans le programme final, par ce que seront les coordonnées du vecteur direction du regard et les coordonnées des vecteurs perpendiculaires aux faces.

Nous pouvons déjà rédiger un programme qui, moyennant l'introduction du vecteur de direction de vision, indiquera si la face du cube est vue ou cachée. Pas question, pour l'instant, de représentation sur l'écran ! Un peu de patience, cela viendra !

```
20 REM FACES VISIBLES D'UN CUBE
   SELON LE VECTEUR K(A,B,C)
30 S=6:REM LES 6 FACES DU CUBE
40 DIM X(S),Y(S),Z(S):REM COORDONNEES
   DES VECTEURS NORMAUX AUX FACES
100 GOSUB 1000
110 INPUT"DIRECTION LA VISEE (A,B,C)";A,B,C
120 GOSUB 4000
130 GOTO 110
1000 REM REMPLISSAGE DES TABLEAUX DES
     COORDONNEES DES VECTEURS NORMAUX
1010 FOR I=1 TO S
1020 READ X(I),Y(I),Z(I)
1030 NEXT
1040 RETURN
1050 DATA 0,-1,0,0,0,1,-1,0,0,1,0,0,0,0,-1,0,1,0
4000 REM TEST VISIBILITE
4010 FOR I=1 TO S
4020 PRINT"FACE NUMERO ";I;" ==>";
4030 IF A*X(I)+B*Y(I)+C*Z(I)>=0 THEN PRINT
     "CACHEE" :GOTO 4050
4040 PRINT"VUE"
4050 NEXT I
4060 RETURN
```

Produit vectoriel

La seconde question - obtenir des vecteurs orthogonaux aux faces et dirigés vers l'extérieur - est plus délicate.

En base orthonormée, le produit vectoriel des deux vecteurs :

$\vec{U}$ (X,Y,Z) et $\vec{U'}$(X',Y',Z')

est le vecteur, noté $\vec{U}\wedge\vec{U'}$ de coordonnées :

$\vec{U}\wedge\vec{U'}$ (YZ'-Y'Z, X'Z-XZ', XY'-X'Y)

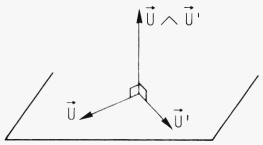
Voilà qui a l'évidence de certaines notions mathématiques ! Et, qui l'eût cru, c'est même une notion utile.

On peut vérifier que :

$$\vec{U}.(\vec{U} \wedge \vec{U'}) = X(YZ'-Y'Z) + Y(X'Z-XZ') + Z(XY'-X'Y) = \emptyset$$
$$\vec{U}\vec{U'}.(\vec{U} \wedge \vec{U'}) = X'(YZ'-Y'-Z)+Y'(X'Z-XZ')+Z'(XY'-X'Y) = \emptyset$$

(il s'agit de produits scalaires). D'après ce que nous venons de voir, si un produit scalaire est nul, c'est que le cosinus de l'angle des vecteurs est nul, donc qu'ils sont orthogonaux .

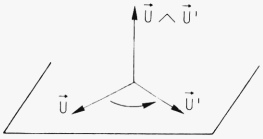
Ayant deux vecteurs, nous pouvons en construire un troisième qui leur soit perpendiculaire : leur produit vectoriel. Cela permet au moins de tracer une figure :

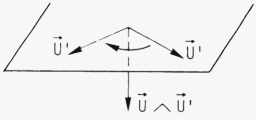


et répond déjà à la moitié de notre question.

Reste encore ce qui concerne l'orientation.

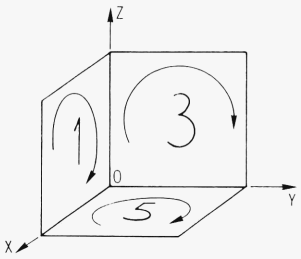
Les matheux montrent, et là faisons-leur confiance, que si l'on oriente le plan que définissent $\vec{U}$ et $\vec{U'}$ (on peut toujours supposer que deux vecteurs ont même origine) de $\vec{U}$ vers $\vec{U'}$, alors $\vec{U} \wedge \vec{U'}$ oriente, dans le bon sens, toute perpendiculaire a ce plan. Le terme "le bon sens" fait référence à ce que nous avons déjà dit page 177 sur l'orientation dans l'espace. Nous avons préféré ce raccourci parlant à de longues et lourdes phrases.

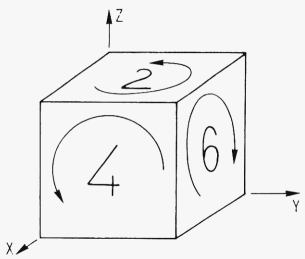




En définitive, ce produit vectoriel est exactement l'outil qu'il nous fallait : il fournit un vecteur perpendiculaire à deux vecteurs donnés et ayant un bon sens d'orientation (si les vecteurs de départ sont bien orientés). Comment imaginer un instant que l'on puisse se passer d'un tel instrument ?

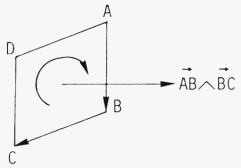
La principale difficulté est de ne pas se tromper dans l'orientation. Voyons comment orienter chaque face de notre cube afin que le produit vectoriel de deux vecteurs, pris dans le bon sens, soit dirigé vers l'extérieur du cube :





A quoi vont servir ces orientations des différentes faces ? A simplifier (oui, oui !) et à unifier la méthode.

A, B, C, D désignant les sommets de n'importe quelle face, à condition que le sens de parcours A vers B,B vers C,C vers D... soit le même que le sens d'orientation de la face.



$\vec{AB} \wedge \vec{BC}$ est un vecteur perpendiculaire à la face et orienté vers l'extérieur (c'est exactement ce que nous voulions).

Alors $\vec{K}'$ définissant la direction du regard, le signe de :

$$\vec{K}' \cdot (\vec{AB} \wedge \vec{BC})$$

permet de tester la visibilité de la face ABCD.

Ce qui vient d'être dit pour un cube est valable dans le cas général.

Stockage d'un objet

Dans l'application envisagée - tenir compte des faces vues ou cachées - il est évident que le stockage des objets doit privilégier les faces : tout ce qui précède a montré que leur rôle est fondamental.

Par rapport à ce qui précède, nous continuerons à définir un objet par une suite de segments consécutifs vus (couleur 1) ou cachés (couleur Ø). Attention, ces vus ou cachés n'ont rien à voir avec le fait qu'une face soit vue ou cachée.

Nous ajouterons les contraintes suivantes :

- une couleur égale à -1 indique le passage à une autre face. Nous prendrons donc COU(Ø)=-1 pour indiquer, dès le départ, que le premier segment est le premier de la première face ;
- pour chaque face, les trois premiers points A, B, C doivent orienter la face dans le sens tel que le produit vectoriel :

  $\vec{AB} \wedge \vec{BC}$

  soit orienté vers l'extérieur de l'objet. Pour les autres points, peu importe, le test de visibilité n'utilisera que $\vec{AB} \wedge \vec{BC}$.

On pourrait envisager de faire vérifier la bonne orientation des faces par l'ordinateur lui-même. Nous ne l'avons pas fait par souci de simplicité... et aussi pour vous contraindre à réfléchir un peu sur l'orientation !

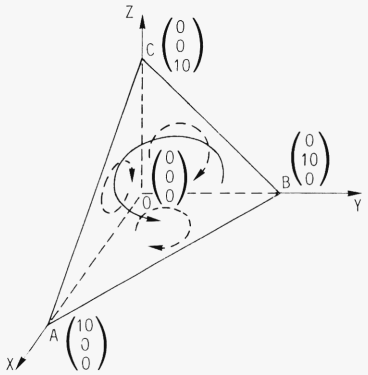
Les DATAS seront organisés de la façon suivante :

- nombre de segments définissant l'objet puis nombre de faces de l'objet ;
- DATAS par groupes de quatre : trois coordonnées (X,Y,Z) d'un point puis "couleur" du segment. Rappelons encore :

  Ø : segment caché
  1 : segment vu
  -1 : début d'une nouvelle face

Prenons, pour changer un peu, l'exemple d'une pyramide. Vous vérifierez que chaque face est bien orientée.

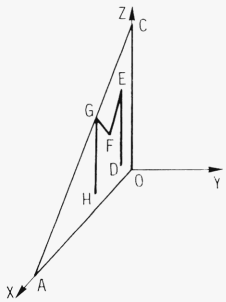
```
DATA 15,4 (15 segments, 4 faces)
REM DATAS face OAC
DATA Ø,Ø,Ø,-1 → première face
DATA 1Ø,Ø,Ø,1 → segment OA vu
DATA Ø,Ø,1Ø,1 → segment AC vu
DATA Ø,Ø,Ø,1  → segment CO vu

REM DATA face OBC
DATA Ø,Ø,Ø,-1 → seconde face
DATA Ø,Ø,1Ø,1 → segment OC vu
DATA Ø,1Ø,Ø,1 → segment CB vu
DATA Ø,Ø,Ø,1  → segment BO vu

REM DATA face OBA
DATA Ø,Ø,Ø,-1 → troisième face
DATA Ø,1Ø,Ø,1 → segment OB vu
DATA 1Ø,Ø,Ø,1 → segment BA vu
DATA Ø,Ø,Ø,1  → segment AO vu

REM DATA face BCA
DATA Ø,1Ø,Ø,-1→ quatrième face
DATA Ø,Ø,1Ø,1 → segment BC vu
DATA 1Ø,Ø,Ø,1 → segment CA vu
DATA Ø,1Ø,Ø,1 → segment AB vu
```



L'avantage de ce codage est sa simplicité. Il permet aussi de dessiner ce que l'on veut sur chacune des faces. Nous n'avons, pour la pyramide, que défini les contours des faces. Rien ne nous empêche de dessiner un M sur la face OAC. Il suffit de rajouter, avant les DATAS de la face OBC :

```
DATA  1,Ø,   1,Ø → segment OD caché
DATA  1,Ø,   5,1 → segment DE vu
DATA  3,Ø,   3,1 → segment EF vu
DATA  5,Ø,   5,1 → segment FG vu
DATA  5,Ø,   1,1 → segment GH vu
```

Le nombre de segments, premier DATA, doit être remplacé par 2Ø.

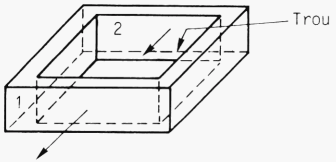


D'autres exemples de DATAS seront donnés, sans explications, dans la partie programmation : à vous de découvrir les figures qu'ils définissent.

Cette façon de stocker un objet est bien adaptée à notre application. Elle présente cependant un inconvénient : chaque sommet est stocké dans chacune des faces à laquelle il appartient. Ce procédé est donc assez gourmand en place mémoire.

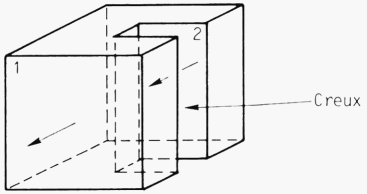
Une dernière remarque, très importante, avant de passer à la programmation. Nous n'avons pris que deux exemples, cube et pyramide, qui présentent une particularité... celle d'être convexes ! Encore un nom barbare. Après produit scalaire, produit vectoriel, cela fait beaucoup. Peut-être, mais c'est fondamental : l'algorithme vu-caché que nous avons donné n'est valable que pour des objets convexes.

Avant de donner une définition, il est évident que notre algorithme est en défaut pour des objets présentant des trous :



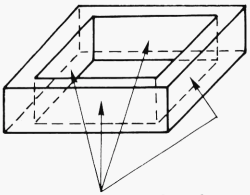
les vecteurs perpendiculaires aux faces 1 et 2 de cet objet et orientés vers l'extérieur de l'objet ont même sens... mais, lorsque la face 1 est visible, toute la face 2 ne l'est pas forcément ! D'où des difficultés si l'on applique notre algorithme sans précaution.

Nous aurions le même problème avec les mêmes faces pour un objet ne comportant pas de trous, mais un simple creux :

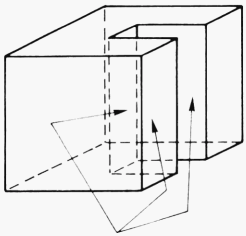


La méthode vue ne s'applique qu'à des objets dits convexes. Comment définir ce mot "convexe" ? Tous nos objets ont des faces. Chacune de ces faces est plane et sépare l'espace en deux régions. Un objet convexe est toujours contenu dans une seule de ces deux régions, ou encore, à aucun moment, il ne traverse le plan d'une de ses faces.

Les deux objets vus plus haut ne sont pas convexes puisque :

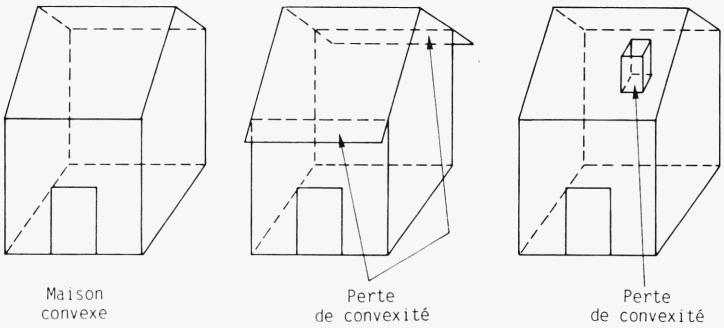


L'objet traverse le plan de chacune de ces faces



L'objet traverse le plan de chacune de ces faces

Une maison, formée de murs et d'un toit est, en première approximation, convexe... mais si le toit dépasse un tout petit peu des murs, si l'on place une cheminée sur le toit, un garage sur le côté (avec son propre toit), elle n'est plus un objet convexe.



Là encore, comme pour l'orientation, on pourrait faire vérifier la convexité de l'objet par l'ordinateur. Nous ne le ferons pas pour ne pas trop alourdir la programmation.

Programmation

Le programme reprend tout ce qui vient d'être précisé, parfois avec des notations différentes.

Il lit tout d'abord les DATAS définissant l'objet. Lors de l'étude du stockage, nous n'avons pas soulevé une question que vous vous êtes peut-être posée : pourquoi mettre en DATA le nombre NF de faces de l'objet ? Il peut facilement être calculé : c'est le nombre de segments de "couleurs" égales à -1. Les lignes suivantes :

```
NF=Ø
FOR I=1 TO S
IF COU(I)=-1 THEN NF=NF+1
NEXT I
```

permettent d'obtenir NF.

Nous faisons lire NF pour éviter des relectures de tableaux. Ce nombre de faces nous sert à dimensionner un tableau F(NF) fournissant l'indice de l'origine du premier segment de chaque face :

```
F(Ø) = Ø   toujours ainsi pour la première face
F(1) =     indice du premier point de la seconde face
...
F(NF-1) =  indice du premier point de la dernière face
```

Définir NF dès le départ présente l'avantage de remplir le tableau F( ) au cours de la lecture des DATAS. F(NF) est égal à S+1 afin que les indices des points de la Jième face (1<=J<=NF) varient toujours entre F(J) et F(J+1)-1.

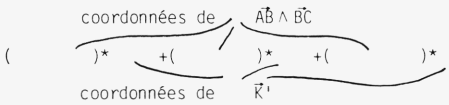
Les trois premiers points de la Jième face, que nous noterons A, B, C, comme à la page 229, ont donc pour coordonnées :

```
A ( X(F(J  )),Y(F(J  )),Z(F(J  )) )
B ( X(F(J)+1),Y(F(J)+1),Z(F(J)+1) )
C ( X(F(J)+2),Y(F(J)+2),Z(F(J)+2) )
```

D'où :

$\vec{AB}$(X(F(J)+1)-X(F(J)),Y(F(J)+1)-Y(F(J)),Z(F(J)+1)-Z(F(J)))
$\vec{BC}$(X(F(J)+2)-X(F(J)+1),Y(F(J)+2)-Y(F(J)+1),Z(F(J)+2)-Z(F(J)+1)

Le test de visibilité de la J-ième face consiste à étudier le signe de l'expression :



C'est pour ne pas vous effrayer que nous avons laissé quelques blancs... Les parenthèses sont à remplir par les coordonnées de $\vec{AB} \wedge \vec{BC}$ (ayant les coordonnées de $\vec{AB}$ et $\vec{BC}$, il suffit de se reporter aux formules de la page 227 et de veiller à ne pas oublier de parenthèses !).

Et à côté des parenthèses, quel va être le vecteur $\vec{K'}$? Là, nous retombons dans la jungle des différentes représentations possibles. Nous avons déjà évoqué : simple projection sur YOZ (pages 184 à 189 pour les deux visualisations), observation sous un certain angle (page 206). Tous les cas de figure sont permis ! Nous ne pouvons malheureusement pas les envisager tous. Nous nous limiterons à l'une des plus simples manipulations de l'objet (rotations autour de OX,OY,OZ), visualisation par simple projection sur YOZ, ce qui entraîne :

$\vec{K'} = -\vec{I}$

et donne une forme plus agréable au test de visibilité (seul, le premier terme subsiste). Nous avons choisi le second type de représentation. Il nous semble plus pédagogique et nécessite peu de calculs.

Récapitulons. Le programme opère comme suit :

- lecture des DATAS définissant l'objet. Remplissage des tableaux X(S), Y(S),Z(S),COU(S) et F(NF) (à partir de la ligne 1ØØØ) ;
- calcul de la plus grande distance entre O et les différents sommets de l'objet (à partir de la ligne 2ØØØ) ;
- test de visibilité sur chaque face. Si une face est vue, le sous-programme de tracé sur l'écran est appelé. Sinon, passage à la face suivante (à partir des lignes 35ØØ et 3ØØØ) ;
- demande du type de manipulation : rotation autour de OX,OY,OZ, calcul des nouvelles coordonnées des sommets de l'objet, puis retour à l'étape 3 (à partir de la ligne 4ØØØ).

Nous avons choisi une structure facilement adaptable à d'autres types de représentations. Nous avons gardé les mêmes numéros de lignes que pour les programmes déjà vus. Si vous préférez la première visualisation (pages 184 à 187) qui donne plus de détails, il suffit de :

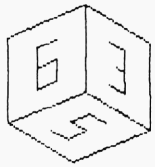
- remplacer l'étape 2 par le sous-programme de plus grande distance adapté à cette représentation (voir page 195) ;
- modifier le sous-programme de tracé sur l'écran (utiliser celui de la page 196).

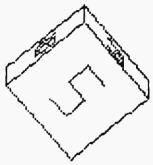
Aucune modification n'est à apporter au test de visibilité. Projeter sur YOZ, c'est toujours regarder dans la direction de $-\vec{I}$.

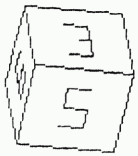
```
10 REM INITIALISATIONS
20 LG=319 :HT=199 :E=0.05
30 XC=XM:YC=YM/2
40 PI=4*ATN(1)
100 GOSUB 1000 REM LECTURE D'UNE FIGURE
110 GOSUB 2000:REM CALCUL DE DS
120 GOSUB 3500:REM TEST DE VISIBILITE ET
    VISUALISATION
130 GOSUB 4000:REM AXE DE ROTATION
140 GOTO 120
1000 REM LECTURE D'UN OBJET EN DATA
1010 RESTORE
1020 READ S,NF REM NOMBRE DE SEGMENTS ET DE
     FACES
1030 DIM X(S),Y(S),Z(S),COU(S),F(NF)
1040 NF=-1
1050 FOR I=0 TO S
1060 READ X(I),Y(I),Z(I),COU(I)
1070 IF COU(I)=-1 THEN NF=NF+1:F(NF)=I
1080 NEXT
1090 NF=NF+1:F(NF)=S+1
1100 RETURN
2000 REM PLUS GRANDE DISTANCE
2010 DS=0
2020 FOR I=0 TO S
2030 D=X(I)*X(I)+Y(I)*Y(I)+Z(I)*Z(I)
2040 IF D>DS THEN DS=D
2050 NEXT I
2060 DS=SQR(DS)+0.1
2070 M=HT/2/DS
2080 RETURN
3000 REM REPRESENTATION DE LA J-EME FACE ,
     SI ELLE EST VUE
3010 X0=XC+Y(F(J))*M*KE
3020 Y0=YC-Z(F(J))*M
3030 FOR I=F(J) TO F(J+1)-1
3040 X1=XC+Y(I)*M*KE
3050 Y1=YC-Z(I)*M
3060 IF COU(I)<>0 THEN DROITE X0,Y0 A X1,Y1
3070 X0=X1:Y0=Y1
3080 NEXT I
3090 RETURN
3500 REM TESTS DE VISIBILITE
3510 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
3520 FOR J=0 TO NF-1
3530 I1=F(J)
3540 TV=(Y(I1+1)-Y(I1))*(Z(I1+2)-Z(I1+1))
     -(Y(I1+2)-Y(I1+1))*(Z(I1+1)-Z(I1)) :
     REM QUANTITE TESTANT LA VISIBILITE D'UNE
     FACE
3550 IF TV>0 THEN GOSUB 3000
3560 NEXT J
3570 RETURN
4000 REM AXE DE ROTATION
4010 REM MODE TEXTE
4020 REM EFFACAGE D'ECRAN
4030 PRINT"AUTOUR DE X,Y,Z ?"
4040 A$=INKEY$:REM OU GET A$
4050 IF A$<"X" OR A$>"Z" THEN 4040
4060 INPUT"ANGLE (EN DEGRES)";T
4070 T=T*PI/180
4500 REM NOUVELLES COORDONNEES
4510 CO=COS(T):SI=SIN(T)
```

```
4520 ON CH GOSUB 4600,4700,4800
4530 RETURN
4600 REM ROTATION AUTOUR DE OX
4610 FOR I=0 TO S
4620 Y=Y(I):Z=Z(I)
4630 Y(I)=Y*CO-Z*SI
4640 Z(I)=Y*SI+Z*CO
4650 NEXT I
4660 RETURN
4700 REM ROTATION AUTOUR DE OY
4710 FOR I=0 TO S
4720 X=X(I):Z=Z(I)
4730 Z(I)=Z*CO-X*SI
4740 X(I)=Z*SI+X*CO
4750 NEXT I
4760 RETURN
4800 REM ROTATION AUTOUR DE OZ
4810 FOR I=0 TO S
4820 X=X(I):Y=Y(I)
4830 X(I)=X*CO-Y*SI
4840 Y(I)=X*SI+Y*CO
4850 NEXT I
4860 RETURN
```

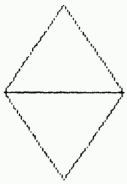
Exemple d'un dé

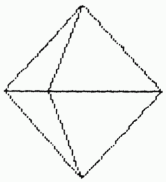


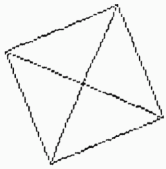




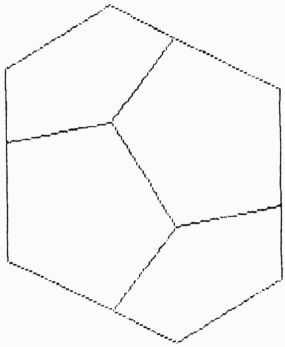
Exemple de l'octaèdre

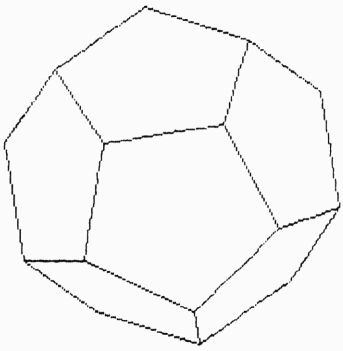


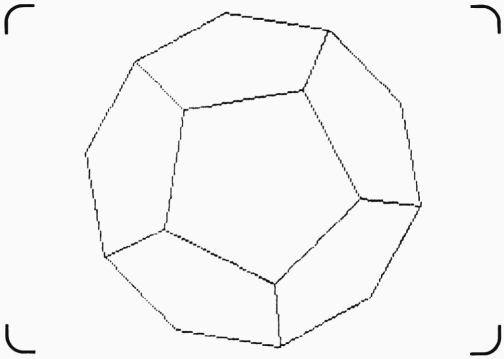




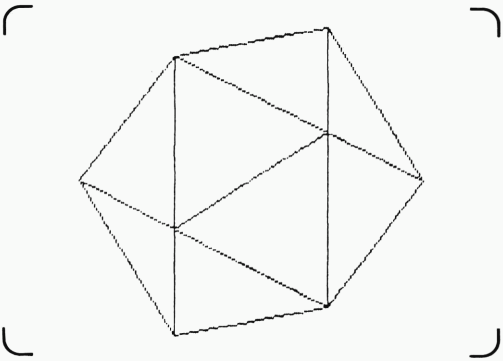
Exemple du dodécaèdre

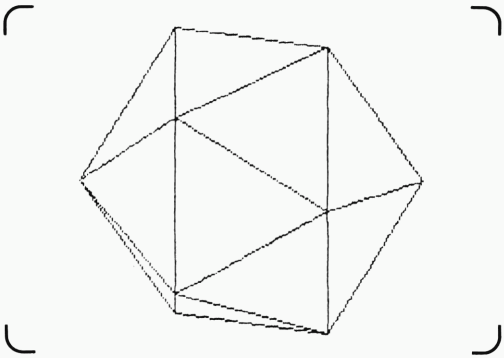


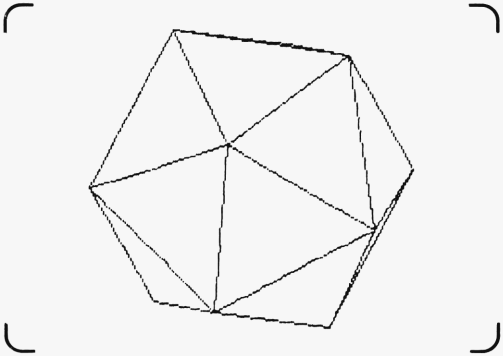




Exemple de l'icosaèdre







On pourrait aussi utiliser la possibilité d'effet de perspective vue précédemment. Nous ne le ferons pas ici : le programme se complique nettement et mieux vaut cerner une à une les difficultés. Vous disposez cependant de tous les éléments nécessaires à une telle adaptation. Et, autant vous l'avouer, nous avons trouvé que le dé, l'octaèdre, l'icosaèdre, le dodécaèdre sont assez jolis en simple projection pour nous limiter à ce cas.

Bien sûr, ce programme ne peut tourner qu'en ajoutant les DATAS définissant l'objet et il faut taper sur la touche retour-chariot pour passer d'une représentation à une manipulation.

```
10000 REM DATAS DE
10010 REM FACE 1
10020 DATA 0,0,0,-1,8,0,0,1,8,0,8,1,0,0,8,1,0,
      0,0,1,5,0,2,0,5,0,6,1
10030 REM FACE2
10040 DATA 0,0,8,-1,8,0,8,1,8,8,8,1,0,8,8,1,0,
      0,8,1,2,3,8,0,2,5,8,1,4,5,8,1,4,3,8,1,6,
      3,8,1,6,5,8,1
10050 REM FACE3
10060 DATA 0,0,0,-1,0,0,8,1,0,8,8,1,0,8,0,1,0,
      0,0,1,0,5,6,0,0,3,5,1,0,3,2,1,0,5,2,1,0,
      4,4,0,0,3,4,1
10070 REM FACE 4
10080 DATA 8,0,0,-1,8,8,0,1,8,8,8,1,8,0,8,1,8,
      0,0,1,8,3,6,0,8,3,4,1,8,5,4,1,8,5,6,0,8,
      5,2,1
10090 REM FACE 5
10100 DATA 0,0,0,-1,0,8,0,1,8,8,0,1,8,0,0,1,0,
      0,0,1,6,5,0,0,6,3,0,1,4,3,0,1,4,5,0,1,2,
      5,0,1,2,3,0,1
10110 REM FACE 6
10120 DATA 8,8,0,-1,0,8,0,1,0,8,8,1,8,8,8,1,8,
      8,0,1,3,8,6,0,5,8,6,1,5,8,2,1,3,8,2,1,3,
      8,4,1,5,8,4,1                          →
```

FICHIERS VU-CACHE

```
10000  REM DATA CUBE
10010  DATA   29 , 6   :REM SEGMENTS,FACES
10020  REM FACE  1
10030  DATA     0.00000 ,    0.00000 ,    0.00000 , -1
10040  DATA    10.00000 ,    0.00000 ,    0.00000 ,  1
10050  DATA    10.00000 ,    0.00000 ,   10.00000 ,  1
10060  DATA     0.00000 ,    0.00000 ,   10.00000 ,  1
10070  DATA     0.00000 ,    0.00000 ,    0.00000 ,  1
10080  REM FACE  2
10090  DATA     0.00000 ,    0.00000 ,   10.00000 , -1
10100  DATA    10.00000 ,    0.00000 ,   10.00000 ,  1
10110  DATA    10.00000 ,   10.00000 ,   10.00000 ,  1
10120  DATA     0.00000 ,   10.00000 ,   10.00000 ,  1
10130  DATA     0.00000 ,    0.00000 ,   10.00000 ,  1
10140  REM FACE  3
10150  DATA     0.00000 ,    0.00000 ,    0.00000 , -1
10160  DATA     0.00000 ,    0.00000 ,   10.00000 ,  1
10170  DATA     0.00000 ,   10.00000 ,   10.00000 ,  1
10180  DATA     0.00000 ,   10.00000 ,    0.00000 ,  1
10190  DATA     0.00000 ,    0.00000 ,    0.00000 ,  1
10200  REM FACE  4
10210  DATA    10.00000 ,    0.00000 ,    0.00000 , -1
10220  DATA    10.00000 ,   10.00000 ,    0.00000 ,  1
10230  DATA    10.00000 ,   10.00000 ,   10.00000 ,  1
10240  DATA    10.00000 ,    0.00000 ,   10.00000 ,  1
10250  DATA    10.00000 ,    0.00000 ,    0.00000 ,  1
10260  REM FACE  5
10270  DATA     0.00000 ,    0.00000 ,    0.00000 , -1
10280  DATA     0.00000 ,   10.00000 ,    0.00000 ,  1
10290  DATA    10.00000 ,   10.00000 ,    0.00000 ,  1
10300  DATA    10.00000 ,    0.00000 ,    0.00000 ,  1
10310  DATA     0.00000 ,    0.00000 ,    0.00000 ,  1
10320  REM FACE  6
10330  DATA    10.00000 ,   10.00000 ,    0.00000 , -1
10340  DATA     0.00000 ,   10.00000 ,    0.00000 ,  1
10350  DATA     0.00000 ,   10.00000 ,   10.00000 ,  1
10360  DATA    10.00000 ,   10.00000 ,   10.00000 ,  1
10370  DATA    10.00000 ,   10.00000 ,    0.00000 ,  1

10000  REM DATA TETRAEDRE
10010  DATA    15 , 4   :REM SEGMENTS,FACES
10020  REM FACE  1
10030  DATA     0.00000 ,    0.00000 ,    0.00000 , -1
10040  DATA     8.66025 ,    5.00000 ,    0.00000 ,  1
10050  DATA     2.88675 ,    5.00000 ,    8.16497 ,  1
10060  DATA     0.00000 ,    0.00000 ,    0.00000 ,  1
10070  REM FACE  2
10080  DATA     0.00000 ,   10.00000 ,    0.00000 , -1
10090  DATA     2.88675 ,    5.00000 ,    8.16497 ,  1
10100  DATA     8.66025 ,    5.00000 ,    0.00000 ,  1
10110  DATA     0.00000 ,   10.00000 ,    0.00000 ,  1
10120  REM FACE  3
10130  DATA     0.00000 ,    0.00000 ,    0.00000 , -1
10140  DATA     2.88675 ,    5.00000 ,    8.16497 ,  1
10150  DATA     0.00000 ,   10.00000 ,    0.00000 ,  1
10160  DATA     0.00000 ,    0.00000 ,    0.00000 ,  1
10170  REM FACE  4
10180  DATA     0.00000 ,    0.00000 ,    0.00000 , -1
10190  DATA     0.00000 ,   10.00000 ,    0.00000 ,  1
10200  DATA     8.66025 ,    5.00000 ,    0.00000 ,  1
10210  DATA     0.00000 ,    0.00000 ,    0.00000 ,  1
```

→

```
10000  REM DATA OCTAEDRE
10010  DATA   31 , 8   :REM SEGMENTS,FACES
10020  REM FACE  1
10030  DATA     0.00000 ,    0.00000 ,    0.00000 , -1
10040  DATA    10.00000 ,    0.00000 ,    0.00000 ,  1
10050  DATA     5.00000 ,    5.00000 ,    7.07107 ,  1
10060  DATA     0.00000 ,    0.00000 ,    0.00000 ,  1
10070  REM FACE  2
10080  DATA    10.00000 ,    0.00000 ,    0.00000 , -1
10090  DATA    10.00000 ,   10.00000 ,    0.00000 ,  1
10100  DATA     5.00000 ,    5.00000 ,    7.07107 ,  1
10110  DATA    10.00000 ,    0.00000 ,    0.00000 ,  1
10120  REM FACE  3
10130  DATA     0.00000 ,   10.00000 ,    0.00000 , -1
10140  DATA     5.00000 ,    5.00000 ,    7.07107 ,  1
10150  DATA    10.00000 ,   10.00000 ,    0.00000 ,  1
10160  DATA     0.00000 ,   10.00000 ,    0.00000 ,  1
10170  REM FACE  4
10180  DATA     0.00000 ,    0.00000 ,    0.00000 , -1
10190  DATA     5.00000 ,    5.00000 ,    7.07107 ,  1
10200  DATA     0.00000 ,   10.00000 ,    0.00000 ,  1
10210  DATA     0.00000 ,    0.00000 ,    0.00000 ,  1
10220  REM FACE  5
10230  DATA     0.00000 ,    0.00000 ,    0.00000 , -1
10240  DATA     5.00000 ,    5.00000 ,   -7.07107 ,  1
10250  DATA    10.00000 ,    0.00000 ,    0.00000 ,  1
10260  DATA     0.00000 ,    0.00000 ,    0.00000 ,  1
10270  REM FACE  6
10280  DATA    10.00000 ,    0.00000 ,    0.00000 , -1
10290  DATA     5.00000 ,    5.00000 ,   -7.07107 ,  1
10300  DATA    10.00000 ,   10.00000 ,    0.00000 ,  1
10310  DATA    10.00000 ,    0.00000 ,    0.00000 ,  1
10320  REM FACE  7
10330  DATA     0.00000 ,   10.00000 ,    0.00000 , -1
10340  DATA    10.00000 ,   10.00000 ,    0.00000 ,  1
10350  DATA     5.00000 ,    5.00000 ,   -7.07107 ,  1
10360  DATA     0.00000 ,   10.00000 ,    0.00000 ,  1
10370  REM FACE  8
10380  DATA     0.00000 ,    0.00000 ,    0.00000 , -1
10390  DATA     0.00000 ,   10.00000 ,    0.00000 ,  1
10400  DATA     5.00000 ,    5.00000 ,   -7.07107 ,  1
10410  DATA     0.00000 ,    0.00000 ,    0.00000 ,  1


10000  REM DATA DODECAEDRE
10010  DATA   71 , 12   :REM SEGMENTS,FACES
10020  REM FACE  1
10030  DATA     2.83550 ,    6.31446 ,    3.90273 , -1
10040  DATA     4.58794 ,    6.31446 ,   -1.49071 ,  1
10050  DATA     0.00000 ,    6.31476 ,   -4.82405 ,  1
10060  DATA    -4.58794 ,    6.31476 ,   -1.49071 ,  1
10070  DATA    -2.83550 ,    6.31446 ,    3.90273 ,  1
10080  DATA     2.83550 ,    6.31446 ,    3.90273 ,  1
10090  REM FACE  2
10100  DATA     2.83550 ,    6.31446 ,    3.90273 , -1
10110  DATA    -2.83550 ,    6.31446 ,    3.90273 ,  1
10120  DATA    -4.58794 ,    1.49071 ,    6.31476 ,  1
10130  DATA     0.00000 ,   -1.49071 ,    7.80547 ,  1
10140  DATA     4.58794 ,    1.49041 ,    6.31476 ,  1
10150  DATA     2.83550 ,    6.31446 ,    3.90237 ,  1
10160  REM FACE  3
10170  DATA     2.83550 ,    6.31446 ,    3.90273 , -1
10180  DATA     4.58794 ,    1.49041 ,    6.31476 ,  1
                                                    ──►
```

```
10190  DATA     7.42344 ,  -1.49101 ,   2.41202 ,  1
10200  DATA     7.42344 ,   1.49041 ,  -2.41202 ,  1
10210  DATA     4.58794 ,   6.31446 ,  -1.49071 ,  1
10220  DATA     2.83550 ,   6.31446 ,   3.90273 ,  1
10230  REM FACE  4
10240  DATA     4.58794 ,   6.31446 ,  -1.49071 , -1
10250  DATA     7.42344 ,   1.49041 ,  -2.41202 ,  1
10260  DATA     4.58794 ,  -1.49071 ,  -6.31476 ,  1
10270  DATA     0.00000 ,   1.49071 ,  -7.80547 ,  1
10280  DATA     0.00000 ,   6.31476 ,  -4.82405 ,  1
10290  DATA     4.58794 ,   6.31446 ,  -1.49071 ,  1
10300  REM FACE  5
10310  DATA     0.00000 ,   6.31446 ,  -4.82405 , -1
10320  DATA     0.00000 ,   1.49071 ,  -7.80547 ,  1
10330  DATA    -4.58794 ,  -1.49041 ,  -6.31476 ,  1
10340  DATA    -7.42344 ,   1.49101 ,  -2.41202 ,  1
10350  DATA    -4.58794 ,   6.31476 ,  -1.49071 ,  1
10360  DATA     0.00000 ,   6.31476 ,  -4.82405 ,  1
10370  REM FACE  6
10380  DATA    -4.58794 ,   6.31476 ,  -1.49071 , -1
10390  DATA    -7.42344 ,   1.49101 ,  -2.41202 ,  1
10400  DATA    -7.42344 ,  -1.49041 ,   2.41202 ,  1
10410  DATA    -4.58794 ,   1.49701 ,   6.31476 ,  1
10420  DATA    -2.83550 ,   6.31446 ,   3.90273 ,  1
10430  DATA    -4.58794 ,   6.31476 ,  -1.49071 ,  1
10440  REM FACE  7
10450  DATA    -4.58794 ,  -1.49041 ,  -6.31476 , -1
10460  DATA    -2.83550 ,  -6.31446 ,  -3.90273 ,  1
10470  DATA    -4.58794 ,  -6.31446 ,   1.49071 ,  1
10480  DATA    -7.42344 ,  -1.49041 ,   2.41202 ,  1
10490  DATA    -7.42344 ,   1.49101 ,  -2.41202 ,  1
10500  DATA    -4.58794 ,  -1.49041 ,  -6.31476 ,  1
10510  REM FACE  8
10520  DATA    -7.42344 ,  -1.49041 ,   2.41202 , -1
10530  DATA    -4.58794 ,  -6.31446 ,   1.49071 ,  1
10540  DATA     0.00000 ,  -6.31476 ,   4.82405 ,  1
10550  DATA     0.00000 ,  -1.49071 ,   7.80547 ,  1
10560  DATA    -4.58794 ,   1.49071 ,   6.31476 ,  1
10570  DATA    -7.42344 ,  -1.49041 ,   2.41202 ,  1
10580  REM FACE  9
10590  DATA     0.00000 ,  -1.49071 ,   7.80547 , -1
10600  DATA     0.00000 ,  -6.31476 ,   4.82405 ,  1
10610  DATA     4.58794 ,  -6.31476 ,   1.49071 ,  1
10620  DATA     7.42344 ,  -1.49101 ,   2.41202 ,  1
10630  DATA     4.58794 ,   1.49041 ,   6.31476 ,  1
10640  DATA     0.00000 ,  -1.49071 ,   7.80547 ,  1
10650  REM FACE  10
10660  DATA     7.42344 ,  -1.49101 ,   2.41202 , -1
10670  DATA     4.58794 ,  -6.31476 ,   1.49071 ,  1
10680  DATA     2.83550 ,  -6.31446 ,  -3.90273 ,  1
10690  DATA     4.58794 ,  -1.49071 ,  -6.31476 ,  1
10700  DATA     7.42344 ,   1.49041 ,  -2.41202 ,  1
10710  DATA     7.42000 ,  -1.49101 ,   2.41202 ,  1
10720  REM FACE  11
10730  DATA     0.00000 ,   1.49071 ,  -7.80547 , -1
10740  DATA     4.58794 ,  -1.49071 ,  -6.31476 ,  1
10750  DATA     2.83550 ,  -6.31446 ,  -3.90273 ,  1
10760  DATA    -2.83550 ,  -6.31446 ,  -3.90273 ,  1
10770  DATA    -4.58794 ,  -1.49041 ,  -6.31476 ,  1
10780  DATA     0.00000 ,   1.49071 ,  -7.80547 ,  1
10790  REM FACE  12
10800  DATA     0.00000 ,  -6.31476 ,   4.82405 , -1
10810  DATA    -4.58794 ,  -6.31446 ,   1.49071 ,  1
```

```
10820  DATA    -2.83550 ,   -6.31446 ,   -3.90273 ,   1
10830  DATA     2.83550 ,   -6.31446 ,   -3.90273 ,   1
10840  DATA     4.58794 ,   -6.31476 ,    1.49071 ,   1
10850  DATA     0.00000 ,   -6.31476 ,    4.82405 ,   1



10000  REM DATA ICOSAEDRE
10010  DATA    79 , 20   :REM SEGMENTS,FACES
10020  REM FACE  1
10030  DATA     0.00000 ,   10.00000 ,    0.00000 , -1
10040  DATA     0.00000 ,    4.47214 ,    8.94427 ,  1
10050  DATA     8.50651 ,    4.47214 ,    2.76393 ,  1
10060  DATA     0.00000 ,   10.00000 ,    0.00000 ,  1
10070  REM FACE  2
10080  DATA     0.00000 ,   10.00000 ,    0.00000 , -1
10090  DATA     8.50651 ,    4.47124 ,    2.76393 ,  1
10100  DATA     5.25731 ,    4.47214 ,   -7.23607 ,  1
10110  DATA     0.00000 ,   10.00000 ,    0.00000 ,  1
10120  REM FACE  3
10130  DATA     0.00000 ,   10.00000 ,    0.00000 , -1
10140  DATA     5.25731 ,    4.47214 ,   -7.23607 ,  1
10150  DATA    -5.25731 ,    4.47214 ,   -7.23607 ,  1
10160  DATA     0.00000 ,   10.00000 ,    0.00000 ,  1
10170  REM FACE  4
10180  DATA     0.00000 ,   10.00000 ,    0.00000 , -1
10190  DATA    -5.25731 ,    4.47214 ,   -7.23607 ,  1
10200  DATA    -8.50651 ,    4.47214 ,    2.76393 ,  1
10210  DATA     0.00000 ,   10.00000 ,    0.00000 ,  1
10220  REM FACE  5
10230  DATA     0.00000 ,   10.00000 ,    0.00000 , -1
10240  DATA    -8.50651 ,    4.47214 ,    2.76393 ,  1
10250  DATA     0.00000 ,    4.47214 ,    8.94427 ,  1
10260  DATA     0.00000 ,   10.00000 ,    0.00000 ,  1
10270  REM FACE  6
10280  DATA     0.00000 ,    4.47214 ,    8.94427 , -1
10290  DATA    -5.25731 ,   -4.47214 ,    7.23607 ,  1
10300  DATA     5.25731 ,   -4.47214 ,    7.23607 ,  1
10310  DATA     0.00000 ,    4.47214 ,    8.94427 ,  1
10320  REM FACE  7
10330  DATA     0.00000 ,    4.47214 ,    8.94427 , -1
10340  DATA     5.25731 ,   -4.47214 ,    7.23607 ,  1
10350  DATA     8.50651 ,    4.47124 ,    2.76393 ,  1
10360  DATA     0.00000 ,    4.47214 ,    8.94427 ,  1
10370  REM FACE  8
10380  DATA     8.50651 ,    4.47124 ,    2.76393 , -1
10390  DATA     5.25731 ,   -4.47214 ,    7.23607 ,  1
10400  DATA     8.50651 ,   -4.47214 ,   -2.76393 ,  1
10410  DATA     8.50651 ,    4.47124 ,    2.76393 ,  1
10420  REM FACE  9
10430  DATA     8.50651 ,    4.47214 ,    2.76393 , -1
10440  DATA     8.50651 ,   -4.47214 ,   -2.76393 ,  1
10450  DATA     5.25731 ,    4.47214 ,   -7.23607 ,  1
10460  DATA     8.50651 ,    4.47214 ,    2.76393 ,  1
10470  REM FACE  10
10480  DATA     0.00000 ,   -4.47214 ,   -8.94427 , -1
10490  DATA     5.25731 ,    4.47214 ,   -7.23607 ,  1
10500  DATA     8.50651 ,   -4.47214 ,   -2.76393 ,  1
10510  DATA     0.00000 ,   -4.47214 ,   -8.94427 ,  1
10520  REM FACE  11
10530  DATA     0.00000 ,   -4.47214 ,   -8.94427 , -1
10540  DATA    -5.25731 ,    4.47214 ,   -7.23607 ,  1
```

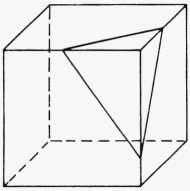
→

```
10550  DATA     5.25731 ,    4.47214 ,  -7.23607 ,  1
10560  DATA     0.00000 ,   -4.47214 ,  -8.94427 ,  1
10570  REM FACE  12
10580  DATA     0.00000 ,   -4.47214 ,  -8.94427 , -1
10590  DATA    -8.50651 ,   -4.47214 ,  -2.76393 ,  1
10600  DATA    -5.25731 ,    4.47214 ,  -7.23607 ,  1
10610  DATA     0.00000 ,   -4.47214 ,  -8.94427 ,  1
10620  REM FACE  13
10630  DATA    -5.25731 ,    4.47214 ,  -7.23607 , -1
10640  DATA    -8.50651 ,   -4.47214 ,  -2.76393 ,  1
10650  DATA    -8.50651 ,    4.47214 ,   2.76393 ,  1
10660  DATA    -5.25731 ,    4.47214 ,  -7.23607 ,  1
10670  REM FACE  14
10680  DATA    -8.50651 ,   -4.47214 ,  -2.76393 , -1
10690  DATA    -5.25731 ,   -4.47214 ,   7.23607 ,  1
10700  DATA    -8.50651 ,    4.47214 ,   2.76393 ,  1
10710  DATA    -8.50651 ,   -4.47214 ,  -2.76393 ,  1
10720  REM FACE  15
10730  DATA     0.00000 ,    4.47214 ,   8.94427 , -1
10740  DATA    -8.50651 ,    4.47214 ,   2.76393 ,  1
10750  DATA    -5.25731 ,   -4.47214 ,   7.23607 ,  1
10760  DATA     0.00000 ,    4.47214 ,   8.94427 ,  1
10770  REM FACE  16
10780  DATA     0.00000 , -10.00000 ,    0.00000 , -1
10790  DATA    -5.25731 ,   -4.47214 ,   7.23607 ,  1
10800  DATA    -8.50651 ,   -4.47214 ,  -2.76393 ,  1
10810  DATA     0.00000 , -10.00000 ,    0.00000 ,  1
10820  REM FACE  17
10830  DATA     0.00000 , -10.00000 ,    0.00000 , -1
10840  DATA     5.25731 ,   -4.47214 ,   7.23607 ,  1
10850  DATA    -5.25731 ,   -4.47214 ,   7.23607 ,  1
10860  DATA     0.00000 , -10.00000 ,    0.00000 ,  1
10870  REM FACE  18
10880  DATA     0.00000 , -10.00000 ,    0.00000 , -1
10890  DATA     8.50651 ,   -4.47214 ,  -2.76393 ,  1
10900  DATA     5.25731 ,   -4.47214 ,   7.23607 ,  1
10910  DATA     0.00000 , -10.00000 ,    0.00000 ,  1
10920  REM FACE  19
10930  DATA     0.00000 , -10.00000 ,    0.00000 , -1
10940  DATA     0.00000 ,   -4.47214 ,  -8.94427 ,  1
10950  DATA     8.50651 ,   -4.47214 ,  -2.76393 ,  1
10960  DATA     0.00000 , -10.00000 ,    0.00000 ,  1
10970  REM FACE  20
10980  DATA     0.00000 , -10.00000 ,    0.00000 , -1
10990  DATA    -8.50651 ,   -4.47124 ,  -2.76393 ,  1
11000  DATA     0.00000 ,   -4.47214 ,  -8.94427 ,  1
11010  DATA     0.00000 , -10.00000 ,    0.00000 ,  1
```

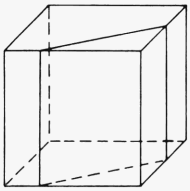
INTERSECTION D'UN PLAN AVEC UN POLYEDRE

Nous voici arrivés au dernier volet de nos études en dimension 3 : intersection d'un polyèdre (cube, tétraèdre, dodécaèdre...) et d'un plan.

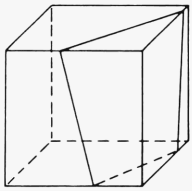
Pour un cube, on imagine facilement certaines de ces intersections :



triangulaires

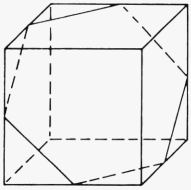


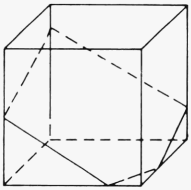
rectangulaires



trapézoïdales

mais, en général, on n'imagine pas des intersections :





hexagonales ou pentagonales !

Encore un domaine qui peut réserver des surprises !

La méthode utilisée

Les figures précédentes indiquent comment procéder : l'intersection d'un plan avec une face donne un segment (enfin, pas toujours). Il nous faudra donc :

- stocker les polyèdres par faces ;
- rechercher, pour chaque face s'il y a intersection en cherchant à obtenir, arête par arête, les extrémités du segment.

Mais comment définir le plan que nous considérons ? Quelques rappels mathématiques sont nécessaires pour bien comprendre la méthode. Nous avons déjà vu, en parlant d'enveloppes que, dans un plan, l'équation d'une droite est du type :

$ax + by + c = 0$ (a et b non tous deux nuls).

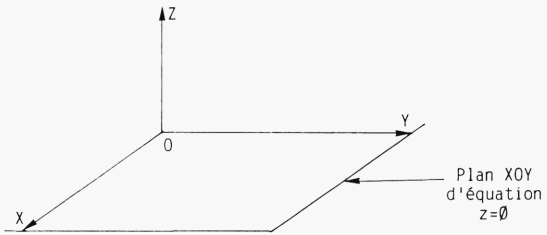
Dans l'espace, nous devons rajouter la troisième coordonnée z. L'équation générale d'un plan est :

$ax + by + cz + d = 0$ (a, b, c non tous trois nuls).

Tout point dont les coordonnées (x, y, z) vérifient l'équation appartient au plan et réciproquement.

Et, pour un point M (x, y, z) en dehors du plan, que se passe-t-il ? Bien entendu, si on calcule $ax+by+cz+d$, on ne trouvera pas $\emptyset$... mais on n'obtiendra pas n'importe quoi : d'un côté du plan, on obtiendra des valeurs positives et, de l'autre, des valeurs négatives.

L'exemple du plan XOY l'illustre bien. Il a pour équation z=Ø (donc a = b = d = 0 et c = 1) :



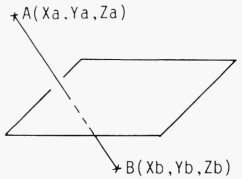
Pour tout point M (x,y,z), ax + by + cz + d = 0 ;

et $z > 0$ définit les points au-dessus du plan

tandis que :

$z < 0$ définit les points au-dessous du plan.

Sachant cela, nous disposons d'un test très simple pour déterminer si un plan coupe un segment : on reporte les coordonnées des deux extrémités du segment dans l'équation du plan et on regarde si les deux valeurs obtenues sont de même signe :



Si les points A et B sont de part et d'autre du plan d'équation AX + BY + CZ + D = Ø, les quantités :

$$PA = A\,X_a + B\,Y_a + C\,Z_a + D$$

et $$PB = A\,X_b + B\,Y_b + C\,Z_b + D$$

sont de signes opposés, et le produit PA*PB est négatif.

La façon dont on obtient les coordonnées du point d'intersection M du plan avec AB a déjà été vue (page 215). Rappelons-la rapidement.

Posons :

$$X1 = X_b - X_a \qquad Y1 = Y_b - Y_a \qquad Z1 = Z_b - Z_a$$

Si le point M appartient au segment A,B, ses coordonnées sont de la forme :

$$X = X_a + \alpha * X1 \qquad Y = Y_a + \alpha * Y1 \qquad Z = Z_a + \alpha * Z1$$

α étant un nombre compris entre Ø et 1.

Comme le point M appartient au plan, ses coordonnées vérifient AX + BY + CZ + D = Ø, soit :

$$A(X_a + \alpha * X1) + B(Y_a + \alpha * Y1) + C(Z_a + \alpha * Z1) + D = Ø$$

On en tire :

$$\alpha = -(A\,X_a + B\,Y_a + C\,Z_a)/(AX1 + BY1 + CZ1).$$

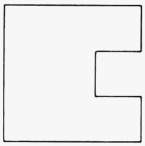
Ces expressions se retrouvent dans le corps du programme, L remplaçant α.

Stockage d'un objet

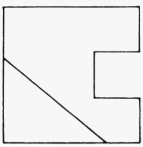
Nous n'avons précisé, jusqu'à présent, qu'une seule chose sur le stockage des objets : il doit être effectué face par face.

Cela n'est pas suffisant. Tout comme dans l'étude de parties vues et cachées, il faut limiter notre ambition, nous ne considérerons que des polyèdres convexes.

Imaginons qu'une face de l'objet ait la forme suivante :

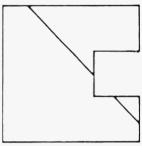


Elle peut être coupée par un plan selon :



1 segment

ou



2 segments

Dans le cas de deux segments, on pourrait facilement obtenir les coordonnées des extrémités des segments comme nous l'avons dit. Mais, quels points joindre ? Pour éviter de trop grandes difficultés, limitons-nous donc à des polyèdres convexes. La vérification de convexité devra être faite par l'utilisateur.

Un autre point important est que, pour cette application, seul le contour des faces importe. Des dessins sur les faces ne sont d'aucune utilité. Si nous cherchions les intersections d'un plan avec le dé numéroté (page 223), les chiffres inscrits sur chaque face ne contribueraient qu'à alourdir le dessin. Autant considérer le cube.

Cela revient à dire que tous les segments définissant l'objet seront vus, sauf ceux qui traduisent le passage d'une face à une autre. Ou encore, les seules "couleurs" considérées seront :

1 pour un segment vu ;
-1 pour un changement de face.

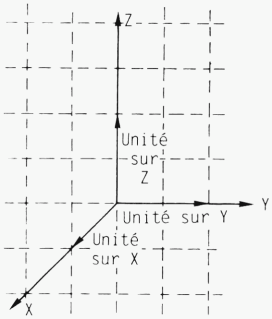
Pas de "couleur" Ø dans les DATAS ! Le programme n'effectue pas cette vérification : à vous de bien définir l'objet ou de rajouter les tests correspondants.

Cette façon de procéder n'est sans doute pas la plus économique. Pourquoi garder des "couleurs" alors qu'en définitive, seul le changement de face importe ? Pour des raisons de simplicité et d'homogénéité. Nous pouvons ainsi garder la même façon, ou presque, de définir un objet que dans l'application précédente (à la couleur Ø près) et les DATAS du tétraèdre, du cube, de l'octaèdre, du dodécaèdre, de l'icosaèdre (pages 240 à 244) peuvent être réutilisés tels quels ! Le sous-programme de lecture des DATAS aussi. Autant de lignes à ne pas retaper.

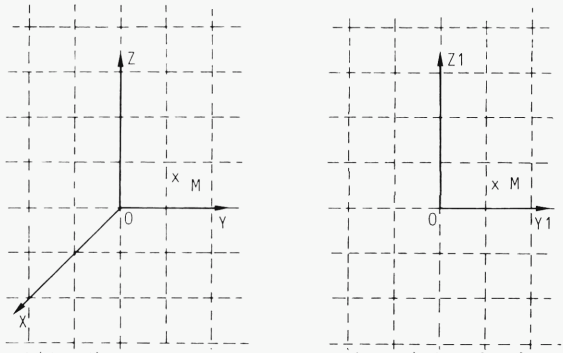
Représentation sur l'écran

Par contre, nous avons changé le type de représentation. Il est, ici, préférable de visualiser tout le polyèdre, y compris les parties cachées, avec une certaine perspective.

La représentation choisie est très naturelle dès que l'on utilise du papier quadrillé :



On trace d'abord des axes rectangulaires OY et OZ, puis un axe OX incliné à 45° et dirigé vers l'observateur. Pour choisir les unités, on peut prendre deux carreaux sur OY et OZ et un carreau sur la diagonale OX. On a alors un système très simple pour passer des coordonnées-espace (X,Y,Z) aux coordonnées-papier ou écran (Y1,Z1). On pourra vérifier que les formules suivantes assurent la conversion :



Y1=Y-0.5*X et Z1=Z-0.5*X

Ces expressions se retrouvent dans le corps du programme. Si la perspective obtenue ne vous satisfait pas, vous pourrez faire des essais en remplaçant 0,5 par d'autres valeurs. En gardant des valeurs identiques dans le calcul de Y1 et Z1, l'axe OX garde la même orientation, seule l'unité varie. Vous pourrez faire varier cette orientation en prenant des valeurs différentes.

Programmation

Nous pouvons maintenant passer à la programmation de cette application. Le programme principal comprend :

- La lecture des DATAS définissant le polyèdre face par face (à partir de la ligne 1000).

- Un sous-programme de mise en page (à partir de la ligne 2000).

  On calcule le minimum et le maximum sur les axes OY et OZ pour centrer la figure (variables Y2 et Z2) et déterminer le coefficient M de mise en page (qui est en général HT/DZ, mais qui peut être égal à LG/DY si le polyèdre est nettement plus large que haut).

- Un sous-programme de représentation du polyèdre à l'écran (à partir de la ligne 3000).

- Un sous-programme d'introduction de l'équation du plan (à partir de la ligne 4000).

  Les coefficients A, B, C sont d'abord introduits. Ils ne peuvent pas être tous les trois nuls. Puis c'est au tour de D, mais après un temps de calcul... occupé à déterminer entre quelles bornes doit se trouver D pour que le plan coupe le polyèdre.

  C'est volontairement que nous n'avons pas détaillé l'obtention de ces bornes. Cette partie du programme utilise le sens de AX+BY+CZ+D. Faites-nous confiance, ça marche bien !

- Un sous-programme de détermination de l'intersection d'une face avec le plan considéré (à partir de la ligne 5000).

  Pour chaque face, on examine successivement l'intersection du plan avec chaque arête (chaque segment), en excluant la seconde extrémité (ligne 5080). Si celle-ci est un point d'intersection, elle sera prise en compte au segment suivant.

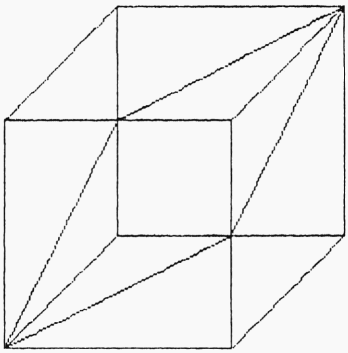
La variable F est un drapeau servant à compter le nombre de points d'intersection trouvés. Les cas les plus classiques sont :

F=0 : le plan ne coupe aucune des arêtes de la face ;
F=2 : le plan coupe la face selon un segment.

Le cas **F=1** sera assez rare : le plan coupe alors la face en un seul point. Les arrondis de calculs risquent plutôt de conduire à F=0 ou F=2 dans un tel cas. A priori, $F \geqslant 3$ est possible. Pour n'importe quel polyèdre, l'intersection d'une face avec le plan qu'elle définit est un polygone d'au moins trois côtés (trois exactement pour le tétraèdre, l'octaèdre, l'icosaèdre, quatre pour le cube, cinq pour le dodécagone !). Comme chacun des côtés du polygone est aussi une intersection du plan avec une autre face, il est inutile de considérer le cas $F \geqslant 3$. C'est le sens de la ligne 5220 : dès que F prend la valeur 2 on ne poursuit plus l'étude d'intersection.

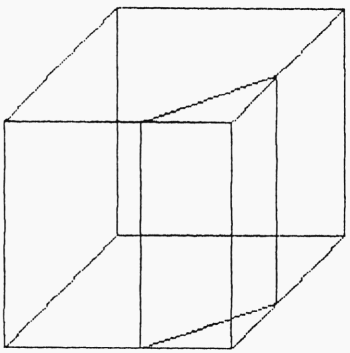
```
10 REM INTERSECTION D'UN POLYEDRE CONVEXE
   ET D'UN PLAN
20 LG=    :HT=
30 XC=    :YC=
40 KE=
50 DIM U(4):REM COORDONNEES DU SEGMENT TRACE
100 GOSUB 1000:REM LECTURE DES DATAS
110 GOSUB 2000:REM MISE EN PAGE
120 GOSUB 3000:REM TRACE DU POLYEDRE
130 GOSUB 3200:REM ATTENTE
140 GOSUB 4000:REM EQUATION DU PLAN
150 GOSUB 3000:REM TRACE DU POLYEDRE
160 GOSUB 5000:REM INTERSECTION
170 GOTO 130
1000 REM REMPLISSAGE DES TABLEAUX
1010 READ S,NF:REM NOMBRE DE SEGMENTS
     ET DE FACES
1020 DIM X(S),Y(S),Z(S),COU(S),F(NF)
1030 NF=-1
1040 FOR I=0 TO S
1050 READ X(I),Y(I),Z(I),COU(I)
1060 IF COU(I)=-1 THEN NF=NF+1:F(NF)=I
1070 NEXT I
1080 NF=NF+1
1090 F(NF)=S+1
1100 RETURN
2000 REM MISE EN PAGE
2010 MY=1E30:NY=-1E30
2020 MZ=1E30:NZ=-1E30
2030 FOR I=0 TO S
2040 Y=Y(I)-.5*X(I)
2050 Z=Z(I)-.5*X(I)
2060 IF Y<MY THEN MY=Y
2070 IF Y>NY THEN NY=Y
2080 IF Z<MZ THEN MZ=Z
2090 IF Z>NZ THEN NZ=Z
2100 NEXT I
2110 DY=NY-MY
2120 DZ=NZ-MZ
2130 M=HT/DZ:IF LG/DY<M THEN M=LG/DY
2140 Y2=(MY+NY)/2
2150 Z2=(MZ+NZ)/2
2160 RETURN
3000 REM TRACE DU POLYEDRE
3010 REM MODE GRAPHIQUE,EFFACAGE D'ECRAN
3020 FOR J=0 TO NF-1
3030 Y=Y(F(J))-.5*X(F(J))
3040 Z=Z(F(J))-.5*X(F(J))
3050 X0=XC+KE*M*(Y-Y2)
3060 Y0=YC-M*(Z-Z2)
3070 FOR I=F(J)+1 TO F(J+1)-1
3080 Y=Y(I)-.5*X(I)
3090 Z=Z(I)-.5*X(I)
3100 X1=XC+KE*M*(Y-Y2)
3110 Y1=YC-M*(Z-Z2)
3120 DROITE X0,Y0  A  X1,Y1
3130 X0=X1
3140 Y0=Y1
3150 NEXT I
3160 NEXT J
3170 RETURN
3200 REM ATTENTE D'UNE COMMANDE
```

```
3210 A$=INKEY$:REM OU GET A$
3220 IF A$<>CHR$(13) THEN 3210
3230 RETURN
4000 REM EQUATION DU PLAN
4010 REM MODE TEXTE,EFFACAGE D'ECRAN
4020 PRINT"EQUATION DU PLAN
4030 INPUT"A=";A
4040 INPUT"B=";B
4050 INPUT"C=";C
4060 IF A=0 AND B=0 AND C=0 THEN 4030
4500 PRINT"CALCUL"
4510 REM CALCUL DES BORNES ENTRE LESQUELLES
     D DOIT ETRE COMPRIS POUR QUE LE PLAN
     COUPE LE POLYEDRE
4520 DI=1E30:DS=-1E30
4530 FOR I=0 TO S
4540 D=A*X(I)+B*Y(I)+C*Z(I)
4550 IF D<DI THEN DI=D
4560 IF D>DS THEN DS=D
4570 NEXT I
4580 PRINT"D DOIT ETRE COMPRIS ENTRE ";-DS;
     " ET ";-DI
4590 PRINT
4600 INPUT"D=";D
4610 IF D<-DS OR D>-DI THEN 4600
4620 RETURN
5000 REM INTERSECTION PLAN-POLYEDRE:CALCUL
     ET TRACE
5010 REM MODE GRAPHIQUE,PAS D'EFFACAGE D'ECRAN
5020 FOR J=0 TO NF-1
5030 F=0
5040 FOR I=F(J) TO F(J+1)-2
5050 P1=A*X(I)+B*Y(I)+C*Z(I)+D
5060 P2=A*X(I+1)+B*Y(I+1)+C*Z(I+1)+D
5070 IF P1*P2>0 THEN 5230
5080 IF P2=0 AND P1<>0 THEN 5230
5090 F=F+1
5100 A1=X(I+1)-X(I)
5110 B1=Y(I+1)-Y(I)
5120 C1=Z(I+1)-Z(I)
5130 IF P1=0 THEN L=0:GOTO 5150
5140 L=-P1/(A*A1+B*B1+C*C1)
5150 X=X(I)+L*A1
5160 Y=Y(I)+L*B1
5170 Z=Z(I)+L*C1
5180 YY=Y-.5*X
5190 ZZ=Z-.5*X
5200 U(2*F-1)=XO+KE*M*(YY-Y2)
5210 U(2*F)=YO-M*(ZZ-Z2)
5220 IF F=2 THEN I=F(J+1)-2
5230 NEXT I
5240 IF F=0 THEN 5270
5250 IF F=1 THEN U(3)=U(1):U(4)=U(2)
5260 DROITE U(1),U(2) A U(3),U(4)
5270 NEXT J
5280 RETURN
```

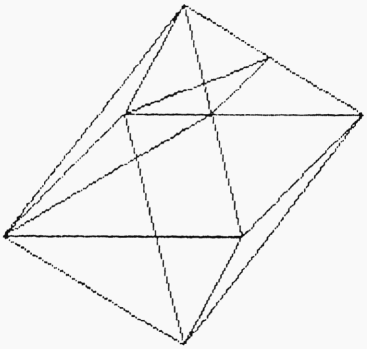


Coupes d'un cube par un plan

A = 1
B = -1
C = 2
D = -1Ø

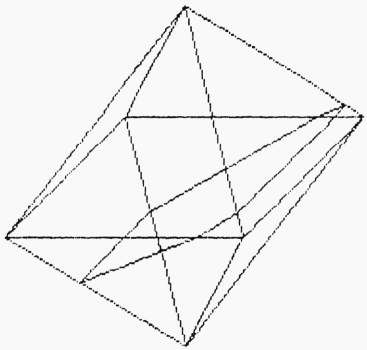


A = 1
B = 1
C = Ø
D = -16

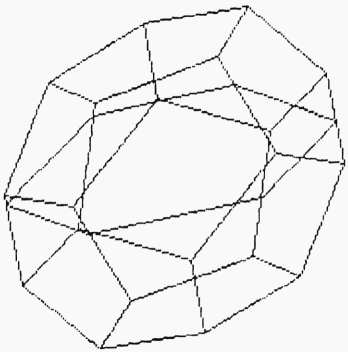


Coupes d'un octaèdre par un plan

A = Ø
B = -1
C = 2
D = Ø

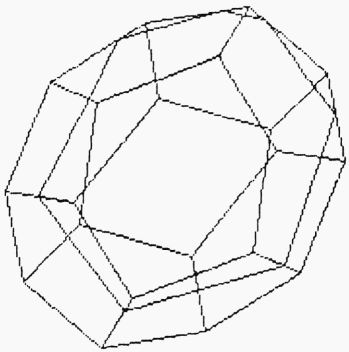


A = Ø
B = -1
C = 2
D = 8



Coupes d'un dodécaèdre par un plan

A = Ø
B = Ø
C = 1
D = -1



A = 1
B = 1
C = 1
D = -4

Ce programme doit être complété par des DATAS. Répétons-le : tous les polyèdres définis dans le paragraphe "parties vues, parties cachées" sont utilisables tels quels.

Commencez vos investigations dans cet univers des coupes par le cube avec :

```
A=1   B=1   C=1      D entre -2Ø et -1Ø
A=1   B=2   C=3      D entre -3Ø et -2Ø
A=1   B=-1  C=2      D entre -2Ø et -1Ø
```

pour bien vous convaincre de l'existence de coupes pentagonales ou hexagonales.

Regardez ce que donne l'octaèdre avec :

```
A=Ø   B=-1  C=2      D entre  Ø et 1Ø
A=2   B=-1  C=4      D entre -2Ø et 1Ø
```

Enfin, pour le dodécaèdre ou l'icosaèdre, n'importe quelles valeurs de A, B, C donnent des résultats surprenants !

Quel que soit le polyèdre choisi, vous constaterez que des plans parallèles sont coupés selon des droites parallèles : c'est l'illustration d'une propriété mathématique classique.

Nous vous suggérons de rédiger les variantes suivantes :

- si vous disposez de couleurs, tracez le polyèdre d'une certaine couleur, et la coupe d'une autre ;
- au lieu d'introduire la valeur de D, introduisez un certain nombre N de coupes et faites afficher N coupes par des plans parallèles.

Pour cela, remplacez les lignes 46ØØ et 461Ø par :

```
46ØØ INPUT "NOMBRES DE COUPES";N
461Ø IF N<=2 THEN 46ØØ
```

et rajoutez dans le programme principal :

```
155 FOR D=-DS TO -DI STEP(DS-DI)/(N-1)
165 NEXT D
```

La première et la dernière coupes ne sont pas toujours très lisibles : elles sont souvent réduites à un point ; parfois, des imprécisions de calcul peuvent conduire à des représentations erronées (cas où P1, P2 sont voisins de zéro et on teste des signes !)..

L'avantage de visualiser plusieurs coupes est de mieux comprendre l'évolution des intersections : passages de trapèzes à des pentagones, de triangles à des hexagones pour le cube... Et nous voici presque revenus à notre point de départ : des études de déformations mais, cette fois-ci, dans l'espace.

Notre étude s'achève.

Nous espérons que vous l'avez suivie jusqu'au bout sans trop de difficultés. Vous aurez constaté que, si les méthodes envisagées sont rarement très compliquées, la longueur des listings en témoigne, elles ne sont pas non plus toujours très simples.

Nous sommes conscients de n'avoir ni exposé les meilleures façons de procéder, lorsqu'elles existent, ni épuisé tout le domaine du graphisme sur micro-ordinateur. Si vous souhaitez des compléments, vous pouvez consulter la bibliographie.

Bibliographie

OUVRAGES GENERAUX

- **Modèles d'expression graphique**
  Jean-Pierre Blanger - Editions du P.S.I.

  Ouvrage simple où sont abordés les tracés de courbes, les transformations géométriques (translation, rotation), les questions d'agrandissement, de hachurage d'une surface plane. Rédigé pour Apple II.

- **La réalisation des logiciels graphiques interactifs**
  Sous la direction de *Michel Lucas* - Editions Eyrolles.

  Ouvrage très intéressant et très riche. On y trouve : tracé d'une droite (quatre algorithmes !), simulation de grisés, hachurage, codage de Freemann d'un chemin, découpage par une fenêtre, fonctions Z=F(X,Y) (deux algorithmes !), intersections de segments (quatre algorithmes !) et des tas d'autres choses sur les logiciels interactifs. Il s'agit d'informatique professionnelle. Programmes rédigés pour machines spécialisées dans le graphisme. Parfois confus pour le non initié.

- **Graphisme scientifique sur micro-ordinateur**
  Robert Dony - Editions Masson.

  Ouvrage de référence. Recoupe en grande partie le livre que vous avez entre les mains : courbes, transformations (rotations, symétries, translations, composition) en dimension 2 et 3, manipulations d'objets en dimension 3, surfaces, problèmes des lignes vues et cachées. Unifie les méthodes grâce au codage de Cohen et Sutherland. Programmes rédigés pour Apple II. Nécessite un assez bon niveau mathématique.

OUVRAGES SE RAPPORTANT A LA PREMIERE PARTIE DE CE LIVRE

Sur les **déformations**, on pourra consulter :

- **L'ordinateur individuel** n° 17 - page 94.

  Un paragraphe concerne les deux méthodes de déformation. Par exemple de l' L'OI en une oie.

Sur les **enveloppes**, on pourra consulter tout cours de mathématiques supérieures pour la théorie.

Sur les **fractals** :

- **Les objets fractals, forme, hasard et dimension**
  De *Benoit Maudelbrot* - Editions Flammarion

  Un bref résumé de cet ouvrage se trouve dans :

- **Penser les mathématiques** - Editions du Seuil.
  (article : "Des monstres de Cantor et Peano à la géométrie fractale de la nature")

Dans ces deux textes, on trouve un aperçu historique des exemples et la notion de dimension d'objets fractals.

- Deux articles de la revue **l'Ordinateur individuel** n° 51 (page 197) et n° 59 (page 162) traitent de la **programmation d'objets dérivés du flocon de Van Koch.** La méthode utilisée, beaucoup plus rapide que la nôtre, simule la récursivité en Basic. A vous de rédiger la synthèse des deux méthodes !

OUVRAGES SE RAPPORTANT A LA DEUXIEME PARTIE DE CE LIVRE

Pour tout ce qui concerne les **différents types de courbes**, consulter des cours de première, terminale ou d'enseignement supérieur.

Une très belle galerie de courbes se trouve dans :

- **Courbes mathématiques**
  Numéro spécial de la revue du Palais de la Découverte.

L'aspect numérique, sans graphisme, des questions approximations par des polygones, méthode du pivot, résolution d'équations différentielles se trouve dans :

- **Mathématiques et statistiques**
  Hervé Haut - Editions du P.S.I.
- **Méthodes de calcul numérique**
  Claude Nowakowski - Editions du P.S.I.

Par contre :

- **Systèmes différentiels, étude graphique**
  Michèle Artigue et Véronique Gautheran - Cedic/Nathan

  se place au **niveau graphique.** On y trouve l'exposé de la **méthode de Runge-Kutta...** et surtout d'admirables planches. Ouvrage à ne pas mettre entre toutes les mains, malheureusement, à cause de son niveau mathématique.

OUVRAGES SE RAPPORTANT A LA TROISIEME PARTIE DE CE LIVRE

Les **articles mathématiques** utilisés dans cette partie (produit scalaire, produit vectoriel, équation de plan...) sont définis dans tout cours de première ou de terminale.

En plus du livre de Monsieur Dony, déjà cité, on pourra consulter :

- **Graphisme 3.D sur votre micro-ordinateur**
  Jean-Louis Vuldy - Editions Eyrolles
- **Pangraphe**
  Jean-Pierre Petit - Editions du P.S.I.

Ces deux livres concernent le paragraphe que nous avons intitulé "déplacement d'un observateur". Les programmes y sont rédigés pour Apple II. "**Graphisme 3.D** reste simple. "**Pangraphe**" est beaucoup plus ambitieux : gestion des objets par fichiers, possibilité de créer des objets en effectuant des rotations, des symétries, en utilisant des cercles ; traitement du "fish-eye".

En cas d'erreur

Tous les listings de cet ouvrage sont des originaux sortis directement de l'imprimante. Une erreur serait donc exceptionnelle.

Si, malgré tout, l'un des programmes ne fonctionnait pas, assurez-vous d'avoir correctement recopié le listing en Basic.

Voici quelques conseils qui vous aideront à déceler l'erreur :

- ne confondez pas 0 (zéro) et O (lettre) ;
- assurez-vous d'avoir placé correctement les . (point) et ; (point-virgule) nécessaires ;
- comptez les lignes du programme à recopier et faites attention à ne pas en oublier ;
- donnez toujours le même nom à vos variables d'un bout à l'autre du programme.

Et surtout... Armez-vous de courage ! Vous obtiendrez de très belles figures à l'écran.

Achevé d'imprimer par l'imprimerie Tardy Quercy S.A. - Bourges
Dépôt légal : Avril 1985 - N° d'imprimeur : 12277 - N° d'édition : 2-86595-209-1

MATHEMATIQUES ET GRAPHISMES

Cet ouvrage permet aux informaticiens de tous niveaux de réaliser rapidement, à partir de fonctions mathématiques simples, de très beaux graphismes qui, sans l'ordinateur, auraient demandé de nombreuses heures de travail.

Outre la joie de créer de magnifiques pages-écran, dont l'ouvrage est abondamment illustré, "Mathématiques et graphismes" vous aidera à apprendre ou à vous remettre en mémoire les déformations et les enveloppes, l'étrange univers des fractals, l'algorithme de Horner, les surfaces en Z2, etc.

Vous maîtriserez vite les programmes BASIC et, en devenant de véritables artistes, vous conviendrez sûrement que la mathématique n'exclut en rien la création.

9 782865 952090

ÉDITIONS DU P.S.I.
BP 86 — 77402 LAGNY S/MARNE CEDEX — FRANCE

ISBN : 2 86595-209 6/130 F.F.

ILLUSTRATION : OLIVIER BARAUD

MATHEMATIQUES ET GRAPHISMES