COMMENT EXPLOITER TOUTES LES RESSOURCES ET AUGMENTER LES PERFORMANCES DE VOTRE

ATARI®



ÉDITIONS WEKA



5

Sommaire

5/0 Plus loin
5/0.1 L'affichage graphique de l'Atari STE

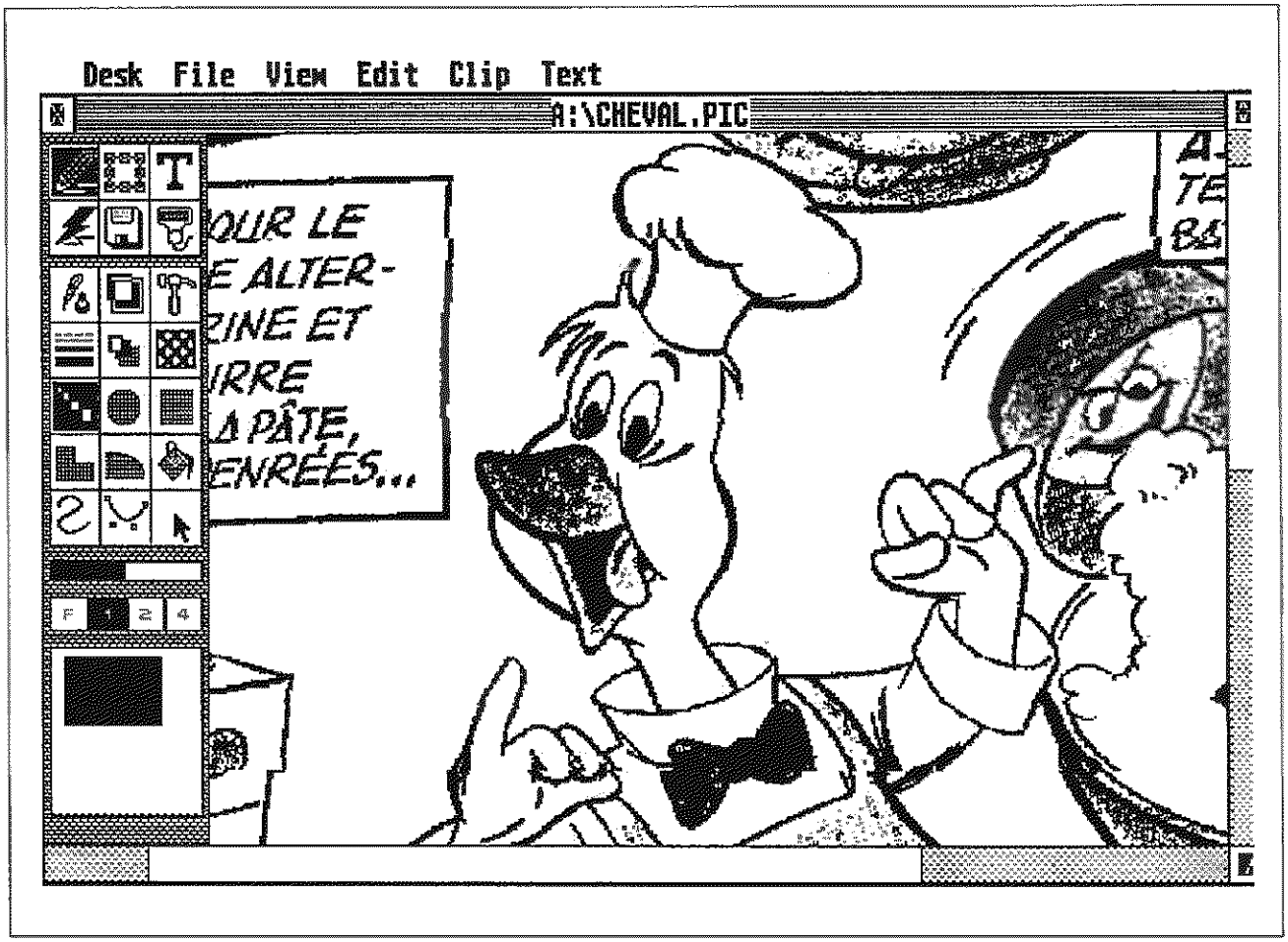
5/1 Le système d'exploitation graphique GDOS

5/2 Le VDI
5/2.1 L'initialisation des fonctions GEM
5/2.2 L'appel des fonctions VDI
I. Les fonctions de contrôle du périphérique
II. Les fonctions de sortie du périphérique
III. Les fonctions d'attribut
IV. Les opérations de copie
V. Les fonctions d'entrée
VI. Les fonctions d'état
5/2.3 Comment programmer le VDI

5/3 Les routines de la LINE_A
5/3.1 Quelques exemples d'appels de fonctions
5/3.2 La structure de la mémoire vidéo

5/4 Les programmes de dessin du marché
5/4.1 DEGAS ELITE
I. Des outils pour dessiner
II. Les paramètres de travail
III. Les facilitateurs
IV. Les fonctions spéciales
V. Archivage et impression des images. Menu déroulant FILE
5/4.2 CYBERSTUDIO: CAD-3D 2.0

5/5 Les programmes de graphisme
5/5.1 La loupe Weka



Dès 200 points, la restitution d'une page de bande dessinée est très bonne.

5/5.2 Routines compactages/décompactages de sauvegarde d'écran en format DEGAS.PC3 pour programmes en GFA-Basic
5/5.2.1 Décomptage des images DEGAS toutes résolutions
5/5.3 Outil de dessin
5/5.3.1 Programme de dessin et sa reproduction en Basic OMIKRON
5/5.4 ROTABLOCK Programme de rotation - 90° pour section d'écran saisie par GET X, Y, X1, Y1, ?$
En Basic GFA & en haute résolution
5/5.5 Julia, Mandelbrot et compagnie
5/5.6 Le mouvement brownien sur Atari
5/5.7 Agrégation fractale par diffusion limitée

5/6 Traitement d'image
5/6.1 Le digitaliseur d'image VIDI-ST
5/6.2 Le logiciel de retouche Touch-Up et le scanner Golden Image

5/0

Plus loin

L'affichage graphique est un des domaines où les machines modernes ont fait le plus de progrès. Les résolutions sont passées de quelques milliers de pixels à plusieurs millions pour les machines professionnelles. Les palettes de couleurs ont vu aussi leur taille augmenter considérablement et surtout, un paramètre sur lequel les constructeurs restent en général très discrets, le nombre de couleurs parmi cette palette qui sont affichables simultanément.

L'Atari STE étant une de ces machines où le progrès essentiel vient des capacités d'affichage accrues, il inaugure donc le chapitre *Plus loin* dans le domaine graphique, à savoir la partie 5 de votre ouvrage.

5/0.1

L'affichage graphique de l'Atari STE[1)]

RÉSOLUTION ET PALETTE

Si la résolution graphique maximum du STE reste la même :

— 640×400 en mode monochrome, haute résolution ;

1) Les éditions WEKA remercient ATARI MAGAZINE de les avoir autorisées à reprendre certaines informations publiées dans le Dossier STE du numéro 6 du mois de novembre 1989.

— 640 x 200 en 4 couleurs en moyenne résolution ;

— 320 x 200 en 16 couleurs en basse résolution ;

sa palette, par contre, se trouve étendue à 4096 couleurs contre 512 pour les anciennes versions. En effet, 16 niveaux de rouge, 16 niveaux de vert et 16 niveaux de bleu se combinent pour vous donner cette gamme étendue.

Ne nous méprenons pas : sur le STE, il est impossible, et ne sera jamais possible, d'afficher simultanément et indépendamment 4096 couleurs différentes. Seules 16 couleurs choisies parmi ces 4096 pourront être affichées par chacun d'entre vous sans problème, quelques-uns, possédant les notions suffisantes d'assembleur, sauront biaiser en affichant 16 couleurs simultanées sur une même ligne en changeant ce choix à chaque ligne (certains logiciels du commerce utilisent déjà ce truc). Les plus avancés trouveront le moyen d'afficher simultanément 48 couleurs sur une même ligne. Ce nouveau mode d'affichage ne permettra pas au STE de faire grand chose de plus, et l'affichage de graphismes aussi colorés ne pourra donc se faire qu'en statique. Inutile d'imaginer des jeux d'arcades ou d'aventures dans ces modes.

SYNCHRONISATION VIDÉO

Le système GENLOCK, présent sur le STE, permet maintenant de synchroniser toute image avec une source extérieure comme une caméra ou un téléviseur. Des effets multiples peuvent être alors créés. L'incrustation vidéo, existant sur les ordinateurs Thomson en 1986, est chose facile maintenant. Vous pourrez alors dessiner, en surimpression, toutes sortes de grimaces aux présentateurs TV que vous détestez. Vous pourrez incruster des titres, commentaires, déroulants en couleurs sur vos films vidéo. Vous pourrez agrémenter vos programmes d'images vidéo tournées sur votre caméscope. Nous sommes sûrs que les idées ne vous manqueront pas.

Un mystère a enfin trouvé son explication, celui de la broche GPO (voir Partie 2, chapitre 2.1 page 5). Cette broche au nom poétique mais ô combien vague de *General Purpose Output*... Littéralement : sortie à usage général (peut-on faire plus vague !) trouve maintenant une mission précise, la synchronisation vidéo avec une source externe.

Cette opération était et reste certes possible avec les versions précédentes mais elle ne peut se faire qu'au prix de modifications périlleuses de la carte mère. Modifications que nous n'aborderons pas.

Gageons que nous ne resterons pas longtemps sans voir apparaître des interfaces multiples permettant tous ces trucages sans aucun problème. La partie 10 de votre ouvrage se verra, quant à elle, peut-être bientôt agrémentée d'une de ces interfaces.

Une autre petite révolution dans l'affichage graphique du STE est son *scrolling cablé* , c'est-à-dire que les mouvements d'écran verticaux ou horizontaux sont cette fois gérés par des programmes résidant en mémoire de circuits spécialisés. Le résultat de cette modification est la rapidité d'affichage époustouflante et surtout la possibilité de disposer d'un écran virtuel de dimensions gigantesques, l'écran physique n'étant alors qu'une petite fenêtre que vous déplacerez sur ce grand écran, un peu à la manière d'un tableur.

5/1

Le système d'exploitation graphique GDOS

GDOS est une extension du GEM offrant des possibilités étendues de gestion de périphériques et de polices de caractères. Il se présente sous forme d'un programme GDOS.PRG, accompagné d'un fichier ASSIGN.SYS. Au chargement, GDOS lit le fichier ASSIGN contenant le nom des polices et des drivers à charger, et la résolution associée à chacun d'eux. Par la suite, le programme utilisant GDOS n'aura plus qu'à charger la police ou le driver choisi pour un périphérique par l'appel GEM « VST_LOAD_FONT ».

Attention :

GDOS ne fait pas partie du GEM standard. Digital Research (DR) en est le propriétaire mais ne permet pas sa diffusion gratuite. Ainsi, si vous écrivez une application sous GDOS, il vous faudra rétribuer DR pour avoir l'autorisation de la diffuser.

La dernière version de GDOS est la 1.1, bien que certaines versions aient un numéro de série allant jusqu'à 2.00.

LA STRUCTURE DU FICHIER ASSIGN.SYS

Ce type de fichier est éditable avec tout traitement de texte ou éditeur et contient le nom des périphériques à utiliser, les drivers et les polices associées.

Chaque périphérique doit avoir un numéro d'identification associé, appelé ID (*Identification Device*).

Type de périphérique	numéro d'ID
écran	1 à 10
table traçante	11 à 20
imprimante	21 à 30
fichier métafile	31 à 40
caméra	41 à 50
tablette graphique	51 à 60

Remarque :
Les drivers d'écran sont présents en ROM, sinon vous ne pourriez même pas utiliser votre ordinateur !
Détail des ID d'écran :
défaut : 1
basse résolution : 2
moyenne résolution : 3
haute résolution : 4
extension future Atari : 5
Chaque driver peut avoir son numéro d'ID suivi d'une lettre :
« **r** » : le driver est chargé à l'initialisation GDOS et reste résidant en mémoire jusqu'à l'extinction de l'ordinateur ou son RESET
« **p** » : le driver est placé en ROM (cas des drivers écran)
pas de lettre : le driver est chargé à l'appel GEM d'ouverture d'une station physique.

Exemple de fichier ASSIGN.SYS :
(les polices associées aux drivers ont toutes l'extension .FNT)

```
path = a:\drivers\   ; Dossier des polices et drivers
;
01p screen.sys       ; Ecran par défaut (en ROM)
;
;
02p screen,sys       ; Basse résolution (en ROM)
;
COMPHS08.FNT
COMPHD08.FNT
;
03p screen.sys       ; Moyenne résolution (en ROM)
;
COMPSS16.FNT
COMPSD16.FNT
;
04p screen.sys       ; Haute résolution (en ROM)
;
COMPDS32.FNT
COMPDD32.FNT
;
21 fx80.sys          ; Driver d'imprimante à charger
EPSHSS10.FNT
```



COMMENT SAVOIR SI GDOS EST EN MÉMOIRE ?

Un simple test permet à tout programmeur de savoir si GDOS a été chargé en mémoire, et donc de pouvoir réagir en conséquence par l'affichage d'une fenêtre d'alerte du type « ATTENTION, GDOS non installé. ».

```
; test GDOS en assembleur
        move.w      # -2,d0       ; Code de demande de l'état
        trap        #2            ; Appel en ROM
        cmp.w       # -2,d0       ; Résultat dans D0
        beq         gdos_non_installé
        ...
        ...
gdos_non_installé :
        ...                       ; affichage d'un message
        ...                       ; d'alerte et sortie.
```

Note : D'après la documentation d'ATARI CORP., avec certaines anciennes versions de GDOS, cet appel renvoie toujours la valeur −2 dans D0, même si GDOS est en mémoire.

UTILISATION DE GDOS

Une fois GDOS en mémoire, votre programme doit faire une ouverture de station non pas virtuelle mais physique en tant qu'initialiation GEM (v_opnwk et non pas v_opnvwk).

Exemple d'utilisation en C

```
/* déclaration des variables */
int work_in[11], work_out[57], handle, contrl[12];
int intin[128], intout[128], ptsin[256], ptsout[256] ;

  main ( )
  {
    int gdos_present, i;
    appl_init ( );
    asm
    {
        move.w      # -2,d0      /* test de présence        */
        trap        #2           /* de gdos                 */
        move.w      d0,gdos_present (a6)
    }
    if (gdos_present == -2)
        {
        appl_exit( );            /* gdos non présent        */
        }
    for(i=1; i<10 ; work_in[i++]=1) ;
        work_in[0]=21 ;  /* driver d'imprimante 21                */
        work_in[10]=0 ;  /* ouverture d'une station physique      */
```

```
v_opnwk (work_in, &handle, work_out) ;
if (handle==0)                  /* non présent                  */
     {
     appl_exit ( ) ;
     }
v_cls_wk(handle);               /* ici, votre programme         */
appl_exit ( ) ;                 /* car tout s'est bien passé    */
}                               /* fermeture de la station      */
```

FORMAT DES POLICES ÉCRAN GDOS (EXTENSION .FNT)

Une police écran GDOS est composée de quatre parties bien distinctes : Un header, une table d'offset des caractères, une table d'offset horizontal, ainsi que les données de chaque caractère.

Le header

octets 0-1 : Numéro d'identification du jeu
octets 2-3 : Taille de la police en points
octets 4-35 : Nom du jeu de caractères en ASCII
octets 36-37 : Valeur ASCII la plus basse du jeu
octets 38-39 : Valeur ASCII la plus haute du jeu
octets 40-41 : Taille du haut des caractères du jeu (valeur positive)
octets 42-43 : Taille de la partie supérieure des caractères du jeu (valeur positive)
octets 44-45 : Taille du milieu des caractères du jeu (valeur positive)
octets 46-47 : Taille de la partie inférieure des caractères du jeu (valeur positive)
octets 48-49 : Taille du bas des caractères du jeu (valeur positive)
octets 50-51 : Largeur du caractère le plus large
octets 52-53 : Largeur du plus gros masque de caractère
octets 54-55 : Décalage à gauche (italique)
octets 56-57 : Décalage à droite (italique)
octets 58-59 : Nombre de pixels d'élargissement (caractères gras)
octets 60-61 : Taille du souligné (caractères soulignés)
octets 62-63 : Masque pour les caractères de type fin (Normalement $5555)
octets 64-65 : Masque pour les caractères italiques
octets 66-67 : Drapeaux divers
  bit 0 : si le jeu utilisé est le jeu du système donc normalement 0 si c'est une police chargée par vos soins.
  bit 1 : 1 si la table d'offset vertical est présente (police proportionnelle)
  bit 2 : définit le format des données : 0 pour le format INTEL (LSB-MSB) et 1 pour le format MOTOROLA (MSB-LSB)
  bit 3 : 1 si tous les caractères ont la même largeur.
octets 68-71 : Pointeur sur la table d'offset horizontal ou 0 si celle-ci n'est pas présente



octets 72-75 : Pointeur sur la table d'offset des caractères
octets 76-79 : Pointeur sur les datas du jeu de caractères
octets 80-81 : Largeur totale des caractères. Utile pour se positionner en début de n'importe quelle ligne du jeu
octets 82-83 : Hauteur en nombre de lignes des caractères
octets 84-87 : Pointeur sur le jeu de caractères suivant. Utile pour retrouver facilement une police lorsque plusieurs ont été chargées en mémoire.

Attention : GDOS ne reconnaît bien que les polices au format INTEL (LSB-MSB), malgré la présence du bit MOTOROLA/INTEL.

La table d'offset des caractères

Cette table contient l'adresse relative de chacun des caractères du jeu dans la police, à partir du premier caractère. Utilisé par GDOS pour retrouver rapidement le début de tout caractère.

Attention, cette table compte le décalage en bits et non pas en octets, pour une économie de stockage des données du jeu.

La table d'offset horizontale

Rarement utilisée, cette table permet un agrandissement ou une réduction du masque des caractères. Le stockage est identique à celui de la table d'offset des caractères.

Datas du jeu de caractères

Datas en bitmap de chacun des caractères du jeu. Grâce au stockage en bits de la table d'offset des lettres, chaque caractère n'a pas à commencer au début d'un octet mais dès que le caractère précédent est fini.



5/2

Le VDI

Le VDI (*Virtual Device Interface*) est une partie du GEM, l'interface graphique écrite par Digital Research et adoptée sur la gamme ST. Son rôle est tout simplement de s'occuper de tout affichage graphique. Cela va d'un rectangle à une ellipse en passant par du texte ou le remplissage de formes.

Les fonctions VDI sont de plusieurs types :

- fonctions de contrôle du périphérique (indispensables pour l'initialisation de tout périphérique, y compris l'écran) ;
- fonctions d'édition de formes de base (cercle, rectangle, lignes, remplissages, etc.) ;
- fonctions d'attribut (pour les attributs d'édition sur chaque périphérique) ;
- fonctions d'interrogation (pour le dialogue entre le programme et l'utilisateur) ;
- fonctions de recherche (pour connaître les attributs choisis pour l'édition sur chaque périphérique).

Leur appel assembleur se fait par l'intermédiaire du **TRAP #2.**

Plusieurs tableaux de paramètres sont utilisés. Ils sont de deux types : ceux d'entrées et ceux de récupération de paramètres (avec suffixe IN ou OUT)

CONTRL (24 octets)
INTIN (256 octets)
INTOUT (256 octets)
PTSIN (256 octets)
PTSOUT (256 octets)

INTIN, INTOUT, PTSIN et PTSOUT contiennent les paramètres nécessaires à la fonction appelée. Le tableau CONTRL est quant à lui un peu plus particulier. Chacune de ses composantes a un rôle particulier :

CONTRL : Opcode de la fonction (son numéro d'identification)
CONTRL+2 : Nombre de valeurs du tableau PTSIN
CONTRL+4 : Nombre de valeurs du tableau PTSOUT
CONTRL+6 : Nombre de valeurs du tableau INTIN
CONTRL+8 : Nombre de valeurs du tableau INTOUT

5/2.1

Initialisation des fonctions GEM

Un programme faisant appel à des fonctions GEM doit faire certaines initialisations avant de les utiliser.

Ces initialisations sont :

APPL_INIT (*Application Init*) ouverture d'une application
GRAF_HANDLE (*Graphic Handle*) recherche du handle de l'application. Ce handle sera par la suite utilisé par tous les appels VDI.

V_OPNWK ou V_OPNVWK (*Open Workstation ou Open Virtual Workstation* suivant que vous utilisez GDOS ou non). Ouverture de la station de travail (non obligatoire si vous n'ouvrez pas de fenêtre).

Le programme doit aussi fermer la station de travail et l'application ouverte avant de se terminer par :

V_CLSWK ou V_CLSVWK (*Close Workstation* ou *Close Virtual Workstation)* (Voir OPEN WORKSTATION).

APPL_EXIT (*Application Exit*) Fermeture de l'application.

Attention :

Les fonctions APPL_INIT, APPL_EXIT et GRAF_HANDLE sont des fonctions AES (voir Partie 3, chapitre 5).

Exemple d'initialisation GEM en C :

```
/* déclaration des variables */
int work_in[11], work_out[57], handle, contrl[12] ;
int intin[128], intout[128], ptsin[128], ptsout[128] ;

main( )
  appl_init( )                          ; /* ouverture d'une application */
  for(i=0 ; i<10 ; work_in[i++]=1)      ; /* mise en place des paramètres */
  work_in[10]=0 ;
  v_opnvwk (work_in, &handle, work_out ; /* open virtual workstation */
/*
Votre programme ici
/*
  v_cls_vwk(handle)                     ; /* close virtual workstation */
  appl_exit( )                          ; /* fermeture de l'application ouverte */
```



5/2.2

L'appel des fonctions VDI

Cet appel peut se faire de plusieurs façons, suivant le langage de programmation utilisé. Par un appel à une routine installée dans les librairies standards (pour le C ou certains assembleurs) ou plus simplement par l'instruction assembleur **TRAP #2.** Bien sûr, avant tout appel, tous les paramètres utilisés par la fonction doivent être placés dans les tableaux d'entrée appropriés.

Sous GEM, le programmeur peut définir tout type de périphérique comme étant une station de travail. Toutes les commandes graphiques sont ainsi exécutées de façon identique que ce soit pour une imprimante ou pour l'écran.

De façon générale, le programmeur n'utilisant pas GDOS n'aura normalement besoin que des fonctions OPEN et CLOSE VIRTUAL WORKSTATION qui initialisent ou ferment l'écran GEM.

I - Les fonctions de contrôle du périphérique

OPEN WORKSTATION

Ouverture d'une station de travail physique. Normalement, cette fonction n'est utilisable que lorsque le GDOS est en mémoire (voir Partie 5, chapitre 2).

Détail des paramètres d'entrée :

CONTRL : 1
CONTRL+2 : 0
CONTRL+6 : 11

INT_IN : Numéro d'identification du périphérique (vois GDOS)
INT_IN+2 : Modèle de ligne
INT_IN+4 : Couleur de ligne
INT_IN+6 : Type de marque
INT_IN+8 : Couleur de marque
INT_IN+10 : Numéro de jeu de caractères
INT_IN+12 : Couleur d'écriture
INT_IN+14 : Type de « pattern » de remplissage
INT_IN+16 : Index du pattern
INT_IN+18 : Couleur du pattern
INT_IN+20 : Type de coordonnées

Détail des paramètres de sortie :

CONTRL+4 : Nombre de données de ptsout (/2)
CONTRL+8 : Nombre de données de intout
CONTRL+12 : Etat du système
INT_OUT : Longueur maximale d'une ligne du périphérique en points
INT_OUT+2 : Hauteur maximale d'une ligne du périphérique en points
INT_OUT+4 : Largeur d'un point en millièmes de millimètre
INT_OUT+6 : Hauteur d'un point en millièmes de millimètre
INT_OUT+8 : Nombre de corps d'écriture acceptés par le périphérique
INT_OUT+10 : Nombre de types de lignes
INT_OUT+12 : Nombre de largeurs de ligne
INT_OUT+14 : Nombre de marques
INT_OUT+16 : Nombre de tailles de marques
INT_OUT+18 : Nombre de types de polices
INT_OUT+20 : Nombre de motifs
INT_OUT+22 : Nombre de types de hachures
INT_OUT+24 : Nombre de couleurs du périphérique
INT_OUT+26 : Nombre de fonctions graphiques fondamentales
INT_OUT+28 : Liste des fonctions graphiques fondamentales protégées
INT_OUT+30 : Trait
INT_OUT+32 : Courbe
INT_OUT+34 : Arc de cercle
INT_OUT+36 : Cercle
INT_OUT+38 : Ellipse
INT_OUT+40 : Courbe elliptique
INT_OUT+42 : Arc d'ellipse
INT_OUT+44 : Rectangle aux coins arrondis
INT_OUT+46 : Rectangle rempli aux coins arrondis
INT_OUT+48 : Texte graphique
INT_OUT+50 à INT_OUT+68 : liste des fonctions d'attributs fondamentales
INT_OUT+70 : Drapeau pour la représentation des couleurs
INT_OUT+72 : Drapeau de défilement rouleau du texte
INT_OUT+74 : Drapeau de remplissage
INT_OUT+76 : Inutilisé
INT_OUT+78 : Nombre de couleurs disponibles
INT_OUT+80 : Type de contrôle du curseur graphique
INT_OUT+82 : Entrées de variables
INT_OUT+84 : Type de touches de sélection
INT_OUT+86 : Type d'entrées alphanumériques
INT_OUT+88 : Type de travail du périphérique
PTS_OUT : Largeur minimale des caractères
PTS_OUT+2 : Hauteur minimale des caractères
PTS_OUT+4 : Largeur maximale des caractères
PTS_OUT+6 : Hauteur maximale des caractères
PTS_OUT+8 : Largeur minimale des lignes
PTS_OUT+10 : Non utilisé
PTS_OUT+12 : Largeur maximale des lignes
PTS_OUT+14 : Non utilisé
PTS_OUT+16 : Largeur minimale des marques
PTS_OUT+18 : Hauteur maximale des marques
PTS_OUT+20 : Largeur maximale des marques
PTS_OUT+22 : Hauteur maximale des marques



OPEN VIRTUAL WORKSTATION

Ouverture d'une station de travail virtuelle. Cette fonction est celle normalement utilisée en temps qu'initialisation par les programmes utilisant GEM.

Détail des paramètres d'entrée :

CONTRL : 100

Les autres paramètres à ceux de OPEN WORKSTATION.

Détail des paramètres de sortie :

Aucun.

CLOSE WORKSTATION

Fermeture d'une station de travail physique. Fonction utilisée en temps que fin d'un programme sous GDOS (voir OPEN WORKSTATION).

Détail des paramètres d'entrée :

CONTRL : 2
CONTRL + 12 : Handle de la station ouverte.

Détail des paramètres de sortie :

Aucun.

CLOSE VIRTUAL WORKSTATION

Fermeture d'une station de travail virtuelle.

Détail des paramètres d'entrée :

CONTRL : 101
CONTRL + 12 : Handle de la station ouverte.

Détail des paramètres de sortie :

Aucun.

CLEAR WORKSTATION

Effacement de la station de travail.

Détail des paramètres d'entrée :

CONTRL : 3
CONTRL + 12 : Handle de la station ouverte.

Détail des paramètres de sortie :

Aucun.

UPDATE WORKSTATION

Le GEM utilisant un tampon temporaire de travail, les commandes envoyées ne sont pas toujours effectuées immédiatement. Cette commande permet d'exécuter prioritairement les ordres déjà stockés. Cela

n'est bien entendu pas utile lorsque vous travaillez sur l'écran, celui-ci étant un périphérique rapide.

Détail des paramètres d'entrée :

CONTRL : 4
CONTRL+12 : Handle de la station ouverte.

Détail des paramètres de sortie :

Aucun.

SET CLIPPING RECTANGLE

Fonction de base du *clipping*. Après cet appel, les tracés sur la station ouverte se font uniquement à l'intérieur du rectangle choisi.

Détail des paramètres d'entrée :

CONTRL : 129
CONTRL+12 : Handle de la station ouverte
INTIN : Drapeau de clipping : 0 découpage hors fonction
1 découpage en fonction
PTS_IN : Coordonnée x du coin supérieur gauche
PTS_IN+2 : Coordonnée y du coin supérieur gauche
PTS_IN+4 : Coordonnée x du coin inférieur droit
PTS_IN+6 : Coordonnée y du coin infrérieur droit

Détail des paramètres de sortie :

Aucun.

LOAD FONTS

Sous GDOS, permet de charger des polices pour n'importe quel périphérique.

Détail des paramètres d'entrée :

CONTRL : 119
CONTRL+12 : Handle de la station ouverte
INT_IN : Inutilisé (réserve)
INT_OUT : Nombre de caractères supplémentaires obtenus

Détail des paramètres de sortie :

Aucun.

UNLOAD FONTS

Libère la place mémoire prise par les polices chargées avec l'ordre LOAD FONTS.

Détail des paramètres d'entrée :

CONTRL : 120
CONTRL+12 : Handle de la station ouverte
INT_IN : Inutilisé
INT_OUT : Nombre de caractères enlevés



Détail des paramètres de sortie :

Aucun.

II - Les fonctions de sortie du périphérique

Celles-ci ne servent qu'à l'édition sur le périphérique. Heureusement, les fonctions d'attribut (voir Partie 5, chapitre 3.2.III) permettent de définir auparavant toute une série de paramètres qui seront alors pris en compte.

Exemple :

La fonction V_GTEXT n'affiche en standard que du texte « normal ». VST_EFFECT permet quant à elle des effets tels que le gras, l'italique, etc.

De même, pour toutes les routines d'affichage de formes remplies, le motif de remplissage doit avoir été au préalable précisé.

V_PLINE

Grâce à cette fonction, vous pouvez aisément tracer tout type de polygone en précisant les coordonnées de chacune des extrémités des segments à tracer. Bien sûr, le type et la couleur du trait sont entièrement paramétrables avec les fonctions d'attribut correspondantes.

Détail des paramètres d'entrée :

CONTRL : 6
CONTRL+2 : Nombre de points du polygone *2
CONTRL+12 : Handle de la station ouverte
PTS_IN : Coordonnée x du premier point
PTS_IN+2 : Coordonnée y du premier point
PTS_IN+4 : Coordonnée x du deuxième point
PTS_IN+6 : Coordonnée y du deuxième point
...
etc.

Détail des paramètres de sortie :

Aucun.

V_PMARKER

Cette routine permet d'afficher des formes prédéfinies à des coordonnées bien précises. Ces formes doivent être auparavant définies dans les fonctions d'attributs et peuvent aller d'un simple point à un cube ou tout autre type de graphisme.

Détail des paramètres d'entrée :

CONTRL : 7
CONTRL+2 : Nombre de marquages *2
CONTRL+12 : Handle de la station ouverte
PTS_IN : Coordonnée en x du premier marquage

PTS_IN+2 : Coordonnée en y du premier marquage
PTS_IN+4 : Coordonnée en x du deuxième marquage
PTS_IN+6 : Coordonnée en y du deuxième marquage
...
etc.

Détail des paramètres de sortie :

Aucun.

V_GTEXT

Voici une fonction simple d'édition de texte. Cependant, si cela a été défini auparavant, tout type d'effet est possible.

Détail des paramètres d'entrée :

CONTRL : 8
CONTRL+2 : 2
CONTRL+4 : Nombre de caractères de la chaîne à éditer
CONTRL+12 : Handle de la station ouverte
INT_IN : Code ASCII du premier caractère
INT_IN+2 : Code ASCII du deuxième caractère
INT_IN+4 : Code ASCII du troisième caractère
...
etc.

Remarque :

Les variables INTIN ayant une taille de 2 octets chacune et un code ASCII en prenant un seul, chacun des caractères est séparé par un octet à 0.

Le premier est nul, le second contient le code ASCII.

La chaîne doit se terminer par 2 octets nuls.

Détail des paramètres de sortie :

Aucun.

V_FILLAREA

Ceci est une routine de remplissage de polygones. Comme pour les fonctions précédentes, tout type de variante est possible avec l'emploi des fonctions d'attribut.

Détail des paramètres d'entrée :

CONTRL : 9
CONTRL+2 : Nombre de points
CONTRL+12 : Handle de la station ouverte
PTS_IN : Coordonnée x du premier point
PTS_IN+2 : Coordonnée y du premier point
PTS_IN+4 : Coordonnée x du deuxième point
PTS_IN+6 : Coordonnée y du deuxième point
...
etc.



Détail des paramètres de sortie :

Aucun.

V_CONTOURFILL

Cette fonction, assez similaire à V_FILLAREA remplit une surface quelconque. Il suffit pour cela de lui donner les coordonnées d'un point à l'intérieur de la surface. La routine remplira alors tous les pixels adjacents jusqu'à rencontrer des pixels d'autre couleur que celle du point de départ. V_CONTOURFILL est la routine employée par la plupart des logiciels de dessin pour les fonctions de remplissage.

Détail des paramètres d'entrée :

CONTRL : 103
CONTRL+2 : 1
CONTRL+6 : 1
CONTRL+12 : Handle de la station ouverte
INT_IN : Index de la couleur de remplissage
PTS_IN : Coordonnée x du point
PTS_IN+2 : Coordonnée y du point

Détail des paramètres de sortie :

Aucun.

VR_RECTFL

Très semblable à V_FILLAREA et à V_CONTOURFILL, cette routine offre la possibilité de remplir des rectangles.

Détail des paramètres d'entrée :

CONTRL : 114
CONTRL+2 : 2
CONTRL+12 : Handle de la station ouverte
PTS_IN : Coordonnée x du coin supérieur gauche
PTS_IN+2 : Coordonnée y du coin supérieur gauche
PTS_IN+4 : Coordonnée x du coin inférieur droit
PTS_IN+6 : Coordonnée y du coin inférieur droit

Détail des paramètres de sortie :

Aucun.

V_BAR

Cette fonction trace un rectangle rempli grâce aux attributs choisis précédemment (voir Partie 5, chapitre 3.2.III).

Détail des paramètres d'entrée :

CONTRL : 11
CONTRL+2 : 2
CONTRL+10 : 1
CONTRL+12 : Handle de la station ouverte

PTS_IN : Coordonnée x du coin supérieur gauche
PTS_IN+2 : Coordonnée y du coin supérieur gauche
PTS_IN+4 : Coordonnée x du coin inférieur droit
PTS_IN+6 : Coordonnée y du coin inférieur droit

Détail des paramètres de sortie :

Aucun.

V_ARC

Trace un arc de cercle de rayon et d'angle définissables.

Détail des paramètres d'entrée :

CONTRL : 11
CONTRL+2 : 4
CONTRL+6 : 2
CONTRL+10 : 3
CONTRL+10 : 2
CONTRL+12 : Handle de la station ouverte
INT_IN : Angle de départ en 1/10 de degrés
INT_IN+2 : Angle d'arrivée en 1/10 de degrés
PTS_IN : Coordonnée x du centre de l'arc de cercle
PTS_IN+2 : Coordonnée y du centre de l'arc de cercle
PTS_IN+12 : Rayon de l'arc de cercle

Détail des paramètres de sortie :

Aucun.

V_PIESLICE

Identique à la fonction précédente, à la seule différence près que l'arc de cercle dessiné est rempli.

Détail des paramètres d'entrée :

CONTRL : 11
CONTRL+2 : 4
CONTRL+6 : 2
CONTRL+12 : Handle de la station ouverte
INT_IN : Angle de départ en 1/10 de degrés
INT_IN+2 : Angle d'arrivée en 1/10 de degrés
PTS_IN : Coordonnée x du centre de l'arc de cercle
PTS_IN+2 : Coordonnée y du centre de l'arc de cercle
PTS_IN+12 : Rayon de l'arc de cercle

Détail des paramètres de sortie :

Aucun.



V_CIRCLE

Cette routine vous permet de tracer des cercles remplis.

Détail des paramètres d'entrée :

CONTRL : 11
CONTRL+2 : 3
CONTRL+6 : 0
CONTRL+10 : 4
CONTRL+12 : Handle de la station ouverte
PTS_IN : Coordonnée x du centre du cercle
PTS_IN+2 : Coordonnée y centre de cercle
PTS_IN+8 : Rayon du cercle

Détail des paramètres de sortie :

Aucun.

V_ELLARC

Cette routine trace un arc d'ellipse.

Détail des paramètres d'entrée :

CONTRL : 11
CONTRL+2 : 2
CONTRL+6 : 2
CONTRL+10 : 6
CONTRL+12 : Handle de la station ouverte
INT_IN : Angle de départ en 1/10 de degrés
INT_IN+2 : Angle d'arrivée en 1/10 de degrés
PTS_IN : Coordonnée x du centre de l'arc d'ellipse
PTS_IN+2 : Coordonnée y du centre de l'arc d'ellipse
PTS_IN+4 : Rayon dans l'axe des x
PTS_IN+6 : Rayon dans l'axe des y

Détail des paramètres de sortie :

Aucun.

V_ELLPIE

Voici l'extension de la fonction précédente, pour le calcul d'arcs d'ellipse pleins.

Détail des paramètres d'entrée :

CONTRL : 11
CONTRL+2 : 2
CONTRL+6 : 2
CONTRL+10 : 7
CONTRL+12 : Handle de la station ouverte
INT_IN : Angle de départ en 1/10 de degrés
INT_IN+2 : Angle d'arrivée en 1/10 de degrés
PTS_IN : Coordonnée x du centre de l'arc d'ellipse
PTS_IN+2 : Coordonnée y du centre de l'arc d'ellipse

PTS_IN+4 : Rayon dans l'axe des x
PTS_IN+6 : Rayon dans l'axe des y

Détail des paramètres de sortie :

Aucun.

V_ELLIPSE

Ceci est une extension des deux fonctions précédentes, utilisée pour le tracé d'ellipses entières pouvant être remplies.

Détail des paramètres d'entrée :

CONTRL : 11
CONTRL+2 : 2
CONTRL+6 : 0
CONTRL+10 : 5
CONTRL+12 : Handle de la station ouverte
PTS_IN : Coordonnée x du centre de l'ellipse
PTS_IN+4 : Rayon dans l'axe des x
PTS_IN+6 : Rayon dans l'axe des y

Détail des paramètres de sortie :

Aucun.

V_RBOX

Cette fonction vous offre la possibilité de tracer des rectangles aux coins arrondis.

Détail des paramètres d'entrée :

CONTRL : 11
CONTRL+2 : 2
CONTRL+6 : 0
CONTRL+10 : 8
CONTRL+12 : Handle de la station ouverte
PTS_IN : Coordonnée x du coin supérieur gauche
PTS_IN+2 : Coordonnée y du coin supérieur gauche
PTS_IN+4 : Coordonnée x du coin inférieur droit
PTS_IN+6 : Coordonnée y du coin inférieur droit

V_JUSTIFIED

Cette routine vous permet d'éditer justifié d'une longueur quelconque.

Détail des paramètres d'entrée :

CONTRL : 11
CONTRL+2 : 3
CONTRL+6 : n+2 (n= le nombre de caractères à éditer)
CONTRL+10 : 10
CONTRL+12 : Handle de la station ouverte
INT_IN : Drapeau d'extension des mots
INT_IN+2 : Drapeau d'extension des caractères



INT_IN+4 : Premier caractère de la chaîne
INT_IN+6 : Deuxième caractère de la chaîne
...
PTS_IN : Coordonnée x de positionnement du texte
PTS_IN+2 : Coordonnée y de positionnement du texte
PTS_IN+4 : Longueur du texte à l'écran

Détail des paramètres de sortie :

Aucun.

III - Les fonctions d'attribut

Elles sont utilisées avant les fonctions de sortie, pour définir les types de ligne, de remplissage ou de texte à afficher. Grâce à elles, toute une série de possibilités vous est offerte pour personnaliser et démultiplier la puissance des fonctions précédentes.

VSWR_MODE

Définit la façon dont une forme s'affiche à l'écran. En effet, l'affichage se fait normalement sans tenir compte de ce qui est déjà présent. ex. : Si votre écran est entièrement rempli de hachures et que vous désiriez afficher une boîte, celle-ci écrase normalement les hachures là où elle s'affiche.

Quatre modes d'affichage peuvent être ici définis :

— Mode remplacement : mode classique ou toute édition écrase ce qui était précédemment à l'écran.

— Mode transparent : les nouveaux points ne s'affichent à l'écran que là où il n'y en avait pas.

— Mode xor : identique au mode transparent, à la seule différence près que les points précédemment placés sont maintenant effacés.

— Mode inverse transparent : inverse du mode transparent.

Détail des paramètres d'entrée :

CONTRL : 32
CONTRL+2 : 0
CONTRL+6 : 1
CONTRL+12 : Handle de la station ouverte
INT_IN : Mode d'affichage choisi
1 mode remplacement
2 mode transparent
3 mode xor
4 mode transparent inverse

Détail des paramètres de sortie :

Aucun.

VS_COLOR

Permet de changer une des couleurs d'affichage. Celle-ci est définie selon le mode classique RVB (Rouge-Vert-Bleu). Pour chacune de ces trois couleurs fondamentales, il faut choisir une intensité allant de 1 à 1000.

Détail des paramètres d'entrée :

CONTRL : 14
CONTRL+2 : 0
CONTRL+6 : 4
CONTRL+12 : Handle de la station ouverte
INT_IN : Numéro de couleur choisie
INT_IN+2 : Valeur du rouge (0 à 1000)
INT_IN+4 : Valeur du vert (0 à 1000)
INT_IN+6 : Valeur du bleu (0 à 1000)

Détail des paramètres de sortie :

Aucun.

VSL_TYPE

Choix du type de ligne à tracer. Six types différents sont déjà définis en ROM, mais vous pouvez bien sûr créer le vôtre (voir fonction suivante).

Numéro du type	Type de ligne
1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 ligne continue
2	1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 ligne pointillée
3	1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 ligne pointillée
4	1 1 1 1 1 1 1 0 0 0 1 1 1 0 0 0 ligne pointillée
5	1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 ligne pointillée classique
6	1 1 1 1 0 0 0 1 1 0 0 1 1 0 0 0 ligne pointillée
7	Type à définir par l'utilisateur

Détail des paramètres d'entrée :

CONTRL : 15
CONTRL+2 : 0
CONTRL+6 : 1
CONTRL+12 : Handle de la station ouverte
INT_IN : Type de ligne choisi

Détail des paramètres de sortie :

Aucun.

VSL_UDSTY

C'est avec cette fonction que vous définissez votre propre type de ligne pour les fonctions polyline.

Détail des paramètres d'entrée :

CONTRL : 113
CONTRL+2 : 0
CONTRL+6 : 1
CONTRL+12 : Handle de la station ouverte
INT_IN : Représentation binaire de votre type de ligne



Détail des paramètres de sortie :

Aucun.

VSL_WIDTH

La largeur des lignes est elle aussi définissable. Attention, car toute largeur n'est pas possible. Seules sont prises correctement en compte les valeurs impaires et strictement supérieures à 3.

ex. : 5, 7, 9 etc.

Détail des paramètres d'entrée :

CONTRL : 16
CONTRL+2 : 1
CONTRL+6 : 0
CONTRL+12 : Handle de la station ouverte
PTS_INT : Largeur de la ligne

Détail des paramètres de sortie :

Aucun.

VSL_COLOR

La couleur de tracé des droites est ici définie.

Détail des paramètres d'entrée :

CONTRL : 17
CONTRL+2 : 0
CONTRL+6 : 1
CONTRL+12 : Handle de la station ouverte
INT_IN : Numéro de la couleur choisie

Détail des paramètres de sortie :

Aucun.

VSL_ENDS

Chacune des extrémités des segments de droite peut avoir plusieurs formes différentes. Coupée (type normal), en flèche ou arrondie.

Détail des paramètres d'entrée :

CONTRL : 108
CONTRL+2 : 0
CONTRL+6 : 2
CONTRL+12 : Handle de la station ouverte
INT_IN : Forme du début de la ligne
0 coupée du type normal
1 en flèche
2 arrondie
INT_IN+2 : Forme de la fin de la ligne
0 coupée du type normal
1 en flèche
2 arrondie

Détail des paramètres de sortie :

Aucun.

VSM_TYPE

La fonction V_PMARKER peut non seulement réaliser des marquages à base de points, mais aussi avec d'autres types plus complexes de formes.

Numéro du type	Type de marquage
1	point
2	croix (+)
3	étoile (*)
4	carré
5	croix (x)
6	diamant

Détail des paramètres d'entrée :

CONTRL : 18
CONTRL+2 : 0
CONTRL+6 : 1
CONTRL+12 : Handle de la station ouverte
INTIN : Type de marquage sélectionné

Détail des paramètres de sortie :

Aucun.

VSM_HEIGHT

La taille des marques est elle aussi entièrement redéfinissable.

Détail des paramètres d'entrée :

CONTRL : 19
CONTRL+2 : 1
CONTRL+6 : 0
CONTRL+12 : Handle de la station ouverte
PTS_IN : 0
PTS_IN+2 : Taille du marquage

Détail des paramètres de sortie :

Aucun.

VSM_COLOR

La couleur de tracé des marques est ici définie.

Détail des paramètres d'entrée :

CONTRL : 20
CONTRL+2 : 0
CONTRL+6 : 1
CONTRL+12 : Handle de la station ouverte
INT_IN : Numéro de la couleur à sélectionner



Détail des paramètres de sortie :
Aucun.

VST_HEIGHT

Cette fonction vous permet de faire varier la taille des caractères à éditer.

Détail des paramètres d'entrée :
CONTRL : 12
CONTRL+2 : 1
CONTRL+6 : 0
CONTRL+12 : Handle de la station ouverte
PTS_IN : 0
PTS_IN+2 : Nouvelle hauteur des caractères

Détail des paramètres de sortie :
PTS_OUT : Largeur des caractères
PTS_OUT+2: Hauteur des caractères
PTS_OUT+4: Largeur du rectangle d'encadrement des caractères
PTS_OUT+6: Hauteur du rectangle d'encadrement des caractères

VST_POINT

Identique à la fonction précédente, mais travaille non pas en pixels mais en points de 1/72 de pouces (1 pouce = 2,54 cm).

Détail des paramètres d'entrée :
CONTRL : 107
CONTRL+2 : 0
CONTRL+6 : 1
CONTRL+12 : Handle de la station ouverte
INT_IN : Hauteur du rectangle d'encadrement des caractères

Détail des paramètres de sortie :
PTS_OUT : Largeur des caractères
PTS_OUT+2: Hauteur des caractères
PTS_OUT+4: Largeur du rectangle d'encadrement des caractères
PTS_OUT+6: Hauteur du rectangle d'encadrement des caractères

VST_ROTATION

Permet d'éditer du texte en rotation par pas de 1/10 de degrés. Le sens de rotation est le sens trigonométrique classique.

Détail des paramètres d'entrée :
CONTRL : 13
CONTRL+2 : 0
CONTRL+6 : 1
CONTRL+12 : Handle de la station ouverte
INT_IN : Angle de rotation du texte 0 à 3600

Détail des paramètres de sortie :
INT_OUT : angle sélectionné

VST_FONT

Permet de changer de jeu de caractères. En standard, le ST n'en possède qu'un en mémoire. Si besoin est, vous pouvez en charger vous-même par l'appel LOAD_FONT quand GDOS est en mémoire.

Détail des paramètres d'entrée :

CONTRL : 21
CONTRL+2 : 0
CONTRL+6 : 1
CONTRL+12 : Handle de la station ouverte
INT_IN : Numéro du nouveau jeu de caractères

Détail des paramètres de sortie :

INT_OUT : Jeu de caractères sélectionné

VST_COLOR

Détermination de la couleur d'édition du texte.

Détail des paramètres d'entrée :

CONTRL : 22
CONTRL+2 : 0
CONTRL+6 : 1
CONTRL+12 : Handle de la station ouverte
INT_IN : Index de la couleur du jeu de caractères à choisir

Détail des paramètres de sortie :

INT_OUT : Index de la couleur du jeu de caractères sélectionné

VST_EFFECTS

Choix d'un ou de plusieurs des effets spéciaux du texte :

Numéro de bit	Type d'effet spécial
0	gras
1	fin
2	italique
3	souligné
4	effilé
5	ombrage

Détail des paramètres d'entrée :

CONTRL : 106
CONTRL+2 : 0
CONTRL+6 : 1
CONTRL+12 : Handle de la station ouverte
INT_IN : Valeur des effets de texte (voir tableau ci-dessus)

Détail des paramètres de sortie :

INT_OUT : effet(s) de texte sélectionné(s)



VST_ALIGNMENT

Routine de centrage horizontal et vertical du texte.

Possibilités offertes en alignement verticalement :

- ligne du bas
- ligne supérieure du bas
- ligne inférieure du milieu
- ligne supérieure du milieu
- ligne inférieure du haut
- ligne du haut

Possibilités offertes en alignement horizontal :

- aligné à gauche
- centré
- aligné à droite

Détail des paramètres d'entrée :

CONTRL : 239
CONTRL+2 : 0
CONTRL+6 : 2
CONTRL+12 : Handle de la station ouverte
INT_IN : Alignement horizontal
0 : à gauche
1 : au centre
2 : à droite
INT_IN+2 : Alignement vertical :
0 : ligne du bas
1 : ligne supérieure du bas
2 : ligne inférieure du milieu
3 : ligne supérieure du milieu
4 : ligne inférieure du haut
5 : ligne du haut

Détail des paramètres de sortie :

INT_OUT : Alignement horizontal pris
INT_OUT+2 : Alignement vertical pris

VSF_INTERIOR

Cette fonction fixe le type de remplissage à effectuer pour les fonctions de sortie du périphérique.

Numéro du type	Type de remplissage
0	aucun remplissage
1	remplissage avec une couleur
2	remplissage avec des points
3	remplissage avec des hachures
4	Code réservé à l'utilisateur pour ses propres remplissages

Détail des paramètres d'entrée :

CONTRL : 23
CONTRL+2 : 0
CONTRL+6 : 1
CONTRL+12 : Handle de la station ouverte
INT_IN : Type de remplissage choisi

Détail des paramètres de sortie :

INT_OUT : Type de remplissage pris

VSF_STYLE

Cette fonction permet d'étendre les possibilités de remplissage du ST à 24 types de remplissages en points et 12 en hachures.

Détail des paramètres d'entrée :

CONTRL : 24
CONTRL+2 : 0
CONTRL+6 : 1
CONTRL+12 : Handle de la station ouverte
INT_IN : Type de remplissage (1 à 12 ou 1 à 24 suivant le type de remplissage sélectionné avec la fonction précédente)

Détail des paramètres de sortie :

INT_OUT : Type de remplissage pris

VSF_COLOR

Détermine la couleur utilisée au cours des opérations de remplissage.

Détail des paramètres d'entrée :

CONTRL : 25
CONTRL+2 : 0
CONTRL+6 : 1
CONTRL+12 : Handle de la station ouverte
INT_IN : Index de la couleur de remplissage

Détail des paramètres de sortie :

INT_OUT : Index de la couleur prise

VSF_PERIMETER

Normalement, toute surface remplie est encadrée par un trait plein. VSF_PERIMETER vous permet d'enlever ou de remettre à volonté ce cadre.

Détail des paramètres d'entrée :

CONTRL : 104
CONTRL+2 : 0
CONTRL+6 : 1



CONTRL + 12 : Handle de la station ouverte
INT_IN : 0 : sans le cadre
1 : avec le cadre

Détail des paramètres de sortie :

INT_OUT : Option prise

VSF_UPDAT

Cette routine permet au programmeur de créer un motif de remplissage personnel.

Détail des paramètres d'entrée :

CONTRL : 112
CONTRL + 2 : 0
CONTRL + 6 : 16* (1 en haute résolution, 2 en moyenne et 4 en basse)
CONTRL + 12 : Handle de la station ouverte
INT_IN à INT_IN + 30 : premier plan des 32 octets de remplissage
INT_IN + 32 à INT_IN + 62 : deuxième plan du motif de remplissage en moyenne et basse résolution uniquement
INT_IN + 64 à INT_IN + 94 : troisième plan du motif de remplissage en basse résolution uniquement
INT_IN + 96 à INT_IN + 126 : quatrième plan du motif de remplissage en basse résolution uniquement

Détail des paramètres de sortie :

Aucun.

IV - Les opérations de copie

Ces quelques fonctions vous permettent de faire des découpages et des copies de portions d'écran. A cet effet, le VDI a besoin de deux tables appelées des MFDB (une pour la source, l'autre pour la destination).

Structure d'une table MFDB :

(un mot = deux octets)

mots 0 et 1 : adresse en mémoire du bloc (multiple de 2 octets)
mot 3 : longueur du bloc en pixels (multiple de 16 pixels)
mot 4 : hauteur du bloc en pixels
mot 5 : longueur du bloc en pixels/16
mot 6 : drapeau de format de quadrillage
0 : quadrillage de l'ordinateur
1 : quadrillage standard du VDI
mot 7 : nombre de plans du bloc (voir Partie 5, chapitre 3.2)
mot 8 : inutilisé (réservé)
mot 9 : inutilisé (réservé)
mot 10 : inutilisé (réservé)

16 modes de transfert sont possibles, basés sur les opérations logiques NOT, OR AND et XOR.

mode : 0 : 0 → destination
1 : source AND destination → destination
2 : source AND NOT → destination
3 : source → destination
4 : NOT source AND destination → destination
5 : destination → destination
6 : source XOR destination → destination
7 : source OR destination → destination
8 : NOT source AND NOT destination → destination
9 : NOT source XOR destination → destination
10 : NOT destination → destination
11 : source OR NOT destination → destination
12 : NOT source → destination
13 : NOT source OR destination → destination
14 : NOT source OR NOT destination → destination
15 : 1 → destination

VRO_CPYFM

Cette fonction effectue une opération logique entre les blocs source et destination, suivant un des modes ci-dessus. Si les blocs source et destination n'ont pas la même taille, la taille du source est prise pour la destination.

Détail des paramètres d'entrée :

CONTRL : 109
CONTRL+2 : 4
CONTRL+6 : 1
CONTRL+12 : Handle de la station ouverte
CONTRL+14 et CONTRL+16 : Adresse du mfdb source
CONTRL+18 et CONTRL+20 : Adresse du mfdb destination
INTIN : Mode de transfert (voir ci-dessus)
PTS_IN : Coordonnée x du coin supérieur gauche de la source
PTS_IN+2 : Coordonnée y du coin supérieur gauche de la source
PTS_IN+4 : Coordonnée y du coin inférieur droit de la source
PTS_IN+6 : Coordonnée y du coin inférieur droit de la source
PTS_IN+8 : Coordonnée x du coin supérieur gauche de la destination
PTS_IN+10 : Coordonnée y du coin supérieur gauche de la destination
PTS_IN+12 : Coordonnée x du coin inférieur droit de la destination
PTS_IN+14 : Coordonnée y du coin inférieur droit de la destination

Détail des paramètres de sortie :

Aucun.

VRT_CPYFM

Copie du bloc source monochrome dans le bloc destination. Ici, les modes vus au-dessus ne sont pas accessibles. Seuls sont disponibles REMPLA-



CEMENT, TRANSPARENT, XOR, TRANSPARENT INVERSE (voir VSWR_MODE).

Si les blocs source et destination n'ont pas la même taille, la taille du source est prise pour la destination.

Détail des paramètres d'entrée :

CONTRL : 121
CONTRL+2 : 4
CONTRL+6 : 1
CONTRL+12 : Handle de la station ouverte
CONTRL+14 et CONTRL+16 : Adresse du mfdb source
CONTRL+18 et CONTRL+20 : Adresse du mfdb destination
INTIN : Mode graphique
PTS_IN : Coordonnée x du coin supérieur gauche de la source
PTS_IN+2 : Coordonnée y du coin supérieur gauche de la source
PTS_IN+4 : Coordonnée x du coin inférieur droit de la source
PTS_IN+6 : Coordonnée y du coin inférieur droit de la source
PTS_IN+8 : Coordonnée x du coin supérieur gauche de la destination
PTS_IN+10 : Coordonnée y du coin supérieur gauche de la destination
PTS_IN+12 : Coordonnée x du coin inférieur droit de la destination
PTS_IN+14 : Coordonnée y du coin inférieur droit de la destination

Détail des paramètres de sortie :

Aucun.

VR_TRNFM

Par cette fonction, on peut passer du format de quadrillage standard à un format propre au périphérique, et vice versa.

Détail des paramètres d'entrée :

CONTRL : 110
CONTRL+2 : 0
CONTRL+6 : 0
CONTRL+12 : Handle de la station ouverte
CONTRL+14 et CONTRL+16 : Adresse du mfdb source
CONTRL+18 et CONTRL+20 : Adresse du mfdb destination

Détail des paramètres de sortie :

Aucun.

V_GET_PIXEL

Cette routine permet de connaître l'état d'un pixel dont on donne les coordonnées (sa couleur, ainsi que sa présence ou non dans le périphérique).

Exemple :

Un pixel de coordonnées 10000,10000 ne se trouve pas dans un écran Atari, 10000 étant supérieur à la résolution de ce périphérique.

Détail des paramètres d'entrée :

CONTRL : 110
CONTRL+2 : 2
CONTRL+6 : 0
CONTRL+12 : Handle de la station ouverte
PTS_IN : Coordonnée x du point
PTS_IN+2 : Coordonnée y du point

Détail des paramètres de sortie :

INT_OUT : Pixel allumé ou éteint (1 ou 0)
INT_OUT+2 : Index de couleur du pixel

V - Les fonctions d'entrée

Elles permettent le dialogue entre l'utilisateur et le programme. Ce sont elles qui gèrent en partie le clavier et la souris.

VEX_TIMV

Cette fonction autorise l'utilisation par le programmeur d'un timer de l'Atari pour y placer sa propre interruption. Au retour de l'appel, ce dernier reçoit l'adresse de l'ancienne interruption, ainsi que son timing en ms.

Détail des paramètres d'entrée :

CONTRL : 118
CONTRL+2 : 0
CONTRL+6 : 0
CONTRL+12 : Handle de la station ouverte
CONTRL+14 à CONTRL+16 : Adresse de la nouvelle routine de gestion de la souris

Détail des paramètres de sortie :

CONTRL+18 à CONTRL+20 : Adresse de l'ancienne routine de gestion de la souris
INT_OUT : Timing de la routine en ms

V_SHOW_C

Routine d'apparition à l'écran du curseur graphique

Détail des paramètres d'entrée :

CONTRL : 122
CONTRL+2 : 0
CONTRL+6 : 1
CONTRL+12 : Handle de la station ouverte
INT_IN : Drapeau
0 : nombre d'appels non pris en compte
1 : nombre d'appels pris en compte



Détail des paramètres de sortie :

Aucun.

V_HIDE_C

Routine de disparition à l'écran du curseur graphique. Deux choix sont possibles pour V_SHOW_C et V_HIDE_C : ou le nombre d'appel des fonctions est pris en compte (après cinq appels à V_HIDE_C, il faut appeler cinq fois V_SHOW_C pour faire apparaître le curseur), ou il ne l'est pas.

Détail des paramètres d'entrée :

CONTRL : 123
CONTRL+2 : 0
CONTRL+6 : 1
CONTRL+12 : Handle de la station ouverte
INT_IN : Drapeau
0 : nombre d'appels non pris en compte
1 : nombre d'appels pris en compte

Détail des paramètres de sortie :

Aucun.

VQ_MOUSE

Cette routine permet de contrôler la position et l'état des boutons de la souris.

Détail des paramètres d'entrée :

CONTRL : 124
CONTRL+2 : 0
CONTRL+6 : 0
CONTRL+12 : Handle de la station ouverte

Détail des paramètres de sortie :

CONTRL+4 : 2
CONTRL+8 : 1
INT_OUT : Etat de la souris :
0 : bouton non pressé
1 : bouton pressé
PTS_OUT : Coordonnée x de la souris
PTS_OUT+2: Coordonnée y de la souris

VEX_BUTV

Cette fonction autorise au programmeur d'utiliser sa propre gestion des pressions sur les boutons de la souris.

Détail des paramètres d'entrée :

CONTRL : 125
CONTRL+2 : 0

CONTRL+6 : 0
CONTRL+12 : Handle de la station ouverte
CONTRL+14 à CONTRL+16 : Adresse de la nouvelle routine de pression des boutons de la souris

Détail des paramètres de sortie :

CONTRL+18 à CONTRL+20 : Adresse de l'ancienne routine.

VEX_MOTV

Cette fonction autorise au programmeur d'utiliser sa propre gestion des déplacements de la souris.

Détail des paramètres d'entrée :

CONTRL : 126
CONTRL+2 : 0
CONTRL+6 : 0
CONTRL+12 : Handle de la station ouverte
CONTRL+14 à CONTRL+16 : Adresse de la nouvelle routine de gestion des déplacements de la souris

Détail des paramètres de sortie :

CONTRL+18 à CONTRL+20 : Adresse de l'ancienne routine

VQ_CURV

Assez proche de la fonction précédente, cette routine force l'ordinateur à exécuter une routine dès que la souris bouge à l'écran.

Détail des paramètres d'entrée :

CONTRL : 127
CONTRL+2 : 0
CONTRL+6 : 0
CONTRL+12 : Handle de la station ouverte
CONTRL+14 à CONTRL+16 : Adresse de la nouvelle routine de gestion des déplacements de la souris

Détail des paramètres de sortie :

CONTRL+18 à CONTRL+20 : Adresse de l'ancienne routine

VQ_KEY_S

L'état des touches spéciales du ST peut avec cette fonction être demandé.

Détail des paramètres d'entrée :

CONTRL : 128
CONTRL+2 : 0
CONTRL+6 : 0
CONTRL+12 : Handle de la station ouverte



Détail des paramètres de sortie :

INT_OUT : état du clavier

numéro de bit	touche
0	Shift de droite
1	Shift de gauche
2	CONTROL
3	ALTERNATE

Si un bit est à 0, c'est que la touche correspondante est pressée.

VI - Les fonctions d'état

Au nombre de 10, elles permettent au programmeur de connaître à tout instant tous les attributs des périphériques ou des fonctions GEM.

VQ_EXTND

Cette routine renvoie les paramètres donnés par V_OPN_WRK (opn workstation) ou, au choix, tous les paramètres de l'ordinateur.

Détail des paramètres d'entrée :

CONTRL : 102
CONTRL+2 : 0
CONTRL+6 : 1
CONTRL+12 : Handle de la station ouverte
INT_IN : Drapeau :
0 : paramètres d'V_OPNWRK
1 : tous les paramètres

Détail des paramètres de sortie :

Tous les paramètres demandés (voir V_OPNWRK) sont dans INT_OUT à INT_OUT+88, et PTS_OUT à PTS_OUT+22

INT_OUT : Mode d'écran (normalement 4)
INT_OUT+2 : Nombre de couleurs disponibles – 1
INT_OUT+4 : Nombre d'effets de textes différents (sont comptabilisés les mélanges d'effets ex : italique-gras)
INT_OUT+6 : Drapeau de possibilité de modification de résolution :
0 : non disponible
1 : disponible
INT_OUT+8 : Nombre de plans graphiques (voir le chapitre sur le graphisme)
INT_OUT+10 : Possibilité de consultation de la table des paramètres :
0 : oui
1 : non
INT_OUT+12 : Réservé

INT_OUT+14 : Drapeau d'implantation de la fonction V_CONTOURFILL :
0 : non
1 : oui
INT_OUT+16 : Drapeau de rotation du texte :
0 : pas de rotation
1 : rotation par pas de 90 degrés
2 : rotation par pas de 1/10 de degrés
INT_OUT+18 : Nombre de modes graphiques
INT_OUT+20 : Réservé
INT_OUT+22 : Drapeau d'alignement de texte :
0 : non
1 : oui
INT_OUT+24 : Drapeau de commande de changement de couleur sur les imprimantes
0 : oui
1 : non
INT_OUT+26 : Drapeau de changement de couleur sur les imprimantes par décalage du ruban vers le haut ou vers le bas (pour certaines imprimantes matricielles)
0 : oui
1 : non
INT_OUT+28 : Nombre maximal de coordonnées*2 dans les fonctions V_PLINE, V_PMARKER ou V_FILLAREA
−1 si taille illimitée
INT_OUT+30 : Taille du tableau INTIN
−1 si taille infinie
INT_OUT+32 : Nombre de boutons de la souris
INT_OUT+34 : Drapeau d'élargissement de lignes
0 : oui
1 : non
INT_OUT+36 à INT_OUT+88 : Réservé
PTS_OUT à PTS_OUT+22 : Réservé

VQ_COLOR

Cette routine permet de déterminer une des couleurs du périphérique.

Détail des paramètres d'entrée :
CONTRL : 26
CONTRL+2 : 0
CONTRL+6 : 2
CONTRL+12 : Handle de la station ouverte
INT_IN : Numéro d'index si INTIN+2=0
INT_IN+2 : Drapeau sur l'index de la couleur
0 : index à déterminer
1 : index actuellement sélectionné

Détail des paramètres de sortie :
INT_OUT : Intensité de la couleur en rouge (0 à 1000)
INT_OUT+2 : Intensité de la couleur en vert (0 à 1000)
INT_OUT+4 : Intensité de la couleur en bleu (0 à 1000)



VQL_ATTRIBUTES

Par cette fonction, on détermine tous les attributs de ligne.

Détail des paramètres d'entrée :

—CONTRL : 35
CONTRL+2 : 0
CONTRL+6 : 0
CONTRL+12 : Handle de la station ouverte

Détail des paramètres de sortie :

INT_OUT : Type de ligne sélectionné
INT_OUT+2 : Couleur sélectionnée
INT_OUT+4 : Mode graphique sélectionné
INT_OUT+6 : Forme du début de la ligne
INT_OUT+8 : Forme de fin de la ligne

VQM_ATTRIBUTES

Idem à la fonction précédente, sauf que c'est le type de marquage qui est ici fourni.

Détail des paramètres d'entrée :

CONTRL : 36
CONTRL+2 : 0
CONTRL+6 : 0
CONTRL+12 : Handle de la station ouverte

Détail des paramètres de sortie :

INT_OUT : Type de marque sélectionné
INT_OUT+2 : Couleur sélectionnée
INT_OUT+4 : Mode graphique sélectionné
INT_OUT+6 : Longueur de la marque sélectionnée
INT_OUT+8 : Hauteur de la marque sélectionnée

VQF_ATTRIBUTES

Même utilisation que précédemment, mais concerne le type de remplissage qui est renvoyé.

Détail des paramètres d'entrée :

CONTRL : 37
CONTRL+2 : 0
CONTRL+6 : 0
CONTRL+12 : Handle de la station ouverte

Détail des paramètres de sortie :

INT_OUT : Type de remplissage sélectionné
INT_OUT+2 : Couleur sélectionnée
INT_OUT+4 : Forme du remplissage
INT_OUT+6 : Mode graphique sélectionné
INT_OUT+8 : Avec ou sans cadre

VQT_ATTRIBUTES

Cette routine renvoie tous les attributs d'édition de texte en cours.

Détail des paramètres d'entrée :

—CONTRL : 38
CONTRL+2 : 0
CONTRL+6 : 0
CONTRL+12 : Handle de la station ouverte

Détail des paramètres de sortie :

INT_OUT : Numéro du jeu de caractères
INT_OUT+2 : Couleur d'édition du texte
INT_OUT+4 : Angle courant de rotation
INT_OUT+6 : Type de centrage horizontal
INT_OUT+8 : Type de centrage vertical
INT_OUT+10 : Mode graphique sélectionné
PTS_OUT : Longueur des caractères
PTS_OUT+2 : Hauteur des caractères
PTS_OUT+4 : Longueur du cadre des caractères
PTS_OUT+6 : Hauteur du cadre des caractères

VQT_EXTEND

Cette routine fort intéressante vous permet de connaître la taille de vos textes entrés avec V_GTEXT. Les paramètres renvoyés sont les coordonnées des quatre coins d'un rectangle englobant exactement le texte.

Détail des paramètres d'entrée :

—CONTRL : 116
CONTRL+2 : 0
CONTRL+6 : x (nombre de lettres de la chaîne de caractères)
CONTRL+12 : Handle de la station ouverte
INT_IN : Première lettre de la chaîne de caractères (voir la remarque dans V_GTEXT
INT_IN+2 : Deuxième lettre de la chaîne de caractères
...
...

Détail des paramètres de sortie :

PTS_OUT : Coordonnée x du premier point du rectangle
PTS_OUT+2 : Coordonnée y du premier point du rectangle
PTS_OUT+4 : Coordonnée x du deuxième point du rectangle
PTS_OUT+6 : Coordonnée y du deuxième point du rectangle
PTS_OUT+8 : Coordonnée x du troisième point du rectangle
PTS_OUT+10 : Coordonnée y du troisième point du rectangle
PTS_OUT+12 : Coordonnée x du quatrième point du rectangle
PTS_OUT+14 : Coordonnée y du quatrième point du rectangle

5/2.3

Comment programmer le VDI

GÉNÉRALITÉS

Le VDI *(Virtual Digital Interface)* regroupe les fonctions graphiques du GEM. Toutes les sorties graphiques et textes d'une application devraient utiliser les sous-programmes du VDI. C'est le cas de l'AES *(Application Environment Service)*, autre plan principal de la structure du GEM, qui gère les fenêtres, boîtes de dialogues, les menus et les événements.

De façon générale une fonction GEM s'appelle en donnant son nom suivi d'une liste de paramètres comme l'appel des fonctions TOS en C. La première chose à faire est de déterminer si la fonction appartient au VDI ou à l'AES. Reportez-vous à la documentation complète de l'ouvrage sur les bibliothèques AES et VDI. Toutes les autres informations (données et adresses) sont stockées dans des tableaux avant l'appel des fonctions. Ces tableaux peuvent aussi, suivant la fonction appelée, retourner des valeurs de paramètres dans des tableaux.

Avant de nous lancer dans la programmation du VDI, il est bon de repréciser rapidement l'existence et les fonctions des différents tableaux.

CONTRL (CONTROL pour l'AES)

Le tableau **CONTRL** reçoit le numéro de la fonction AES et le nombre de paramètres passés aux autres tableaux.

GLOBAL

Pour l'instant ce tableau est sans intérêt car il ne reçoit pas de paramètres mais des informations plus spécifiquement système destinées à la gestion interne. Nous proposerons ultérieurement un exemple qui utilise ce tableau.

INTIN

INT est l'abréviation de **Integer** (entier). Ce tableau reçoit en entrée (**IN**) des paramètres sur deux octets. Le nombre et la signification des deux octets changent suivant la fonction appelée.

INTOUT

C'est le tableau équivalent au précédent mais en sortie de fonction (par exemple coordonnées de lecture de souris).

PTSIN (ADDRIN pour l'AES)

Même principe que le tableau **INTIN**, mais les valeurs sont codées comme des adresses sur 4 octets.

PTSOUT
(ADDROUT pour l'AES)

Comme pour **INTOUT** mais pour des adresses.

INITIALISATION

Le numéro de fonction et le nombre de paramètres d'entrées de la fonction sont précisés dans le tableau **CONTROL**, les valeurs sont stockées dans **INTIN** ou **PTSIN** et retournées après appel de la fonction dans **INTOUT** ou **PTSOUT** suivant leur taille.

Les appels GEM sont normalisés. En C ce sont des fonctions standards. En Basic on ne devrait avoir aucune modification par rapport à l'appel en C. Suivant les langages, il existe des différences dans l'appel de ces fonctions :

Avant de faire un appel de fonction il est nécessaire d'ouvrir un périphérique de sortie avec la fonction **v_opnvwk** (pour **Vdi Open Virtual Workstation**, ouvrir la station de travail VDI, numéro de fonction **100 VDI**. Cette fonction renvoie l'identificateur graphique **(handle)**.

Pour fermer le périphérique de sortie, il suffit d'appeler la fonction **v_clsvwk(Vdi CLose Workstation** numéro de fonction **101 VDI)**.

PROGRAMMES D'INITIALISATION SUIVANT LES LANGAGES

Le langage C

La bibliothèque de tout bon compilateur en C comporte les bindings de l'AES et du VDI. Il suffit de déclarer les tableaux de paramètres après quoi les bindings se chargeront des accès. On peut considérer que ces bindings font partie intégrale du C :

```
int contrl[12],
    intin[128],
    ptsin[128],
    intout[128],
    ptsout[128];
```

Il faut ensuite effectuer le linkage avec les fonctions correspondantes de la bibliothèque VDI **(apstart, vdibind, aesbind, osbind, ibf,...)**.

En C vous n'avez pas besoin de vous préoccuper du bloc de paramètres car les bibliothèques le font pour vous, mais en voici tout de même les noms :

vdipb[0] à vdipb[4]



Programme d'initialisation en C :

```
/* les tableaux ci -dessous doivent toujours être déclarés
dans les programmes utilisant les fonctions VDI*/

int contrl[12],
    intin[128],
    ptsin[128],
    intout[128],
    ptsout[128];

int ap_id,
    handle,
    x_max,
    y-max;

void  gem_init()            /*pour ouvrir la station de travail
                            à appeler au début du programme*/
{
int  work_in[11],           /*ces tableaux sont nécessaires à*/
     workout[57],           /* v_opvwk*/
     i;
     ap_id = appl_init();
     for (i = 0; i<10; work_in[i] = 1);
          work_in[10] = 2;                  /*drapeau de coordonnées =2*/

     v_opnvwk( work_in, &handle, work_out)  /* le handle est stocké
                                               dans &handle*/

     x_max = work_out[0];        /*résolution stockée*/
     y_max = work_out[1];
}

void  gem_exit()                 /*cette fonction ferme la station de
                                 travail ne l'appeler qu'à la fin*/
{
     v_clsvwk (handle);
     appl_exit();
}
```

Le Basic GFA

Il ne possède pas de fonction VDI à proprement parler car toutes les routines VDI ont leur équivalent en instructions du langage Basic. Vous pouvez toutefois, si vous en avez envie, programmer le VDI comme cela se fait en assembleur. Les tableaux énoncés précédemment sont reconnus comme mots clés du langage en GFA et il n'est pas nécessaire de les déclarer.

Par analogie avec la démarche en assembleur, nous vous présentons comment programmer et initialiser un appel au VDI ; la structure de l'initialisation et de la fermeture de la station de travail, ainsi que l'emplacement du programme appelant, sont les mêmes qu'en C.

Programme en version GFA :

```
DIM work_in%(11),pxarray%(20)
'-------Initialisation-----
@grafhandle
FOR x%=0 to 9
  work_in%(x%)=1
NEXT x%
work_in%(10)=2
@v_opvwk(*work_in%(),*handle%)
@vsclip(handle%)          ! pas de clipping
@vclrwk(handle%,0,0,0,0,0,0)
'---------programme principal----
'
'-------fin du programme
@v_clswk(handle%)
END
'
'-------BIBLIOTHEQUE DE SOUS-PROGRAMMES
'
'------fonction AES
PROCEDURE graf_handle
Dpoke GCONTRL+2,0
Dpoke GCONTRL+4,5
Dpoke GCONTRL+6,1
Dpoke GCONTRL+8,0
GEMSYS 77
handle=DPEEK(GINTOUT)
RETURN
'
'----------Définition de fonctions VDI--------
'
'VDI OPEN VIRTUAL WORK STATION
Procedure v_opnvwk(awork_in%,a_handle%)
DPOKE CONTRL+2,0
!nbre de points dans ptsin
DPOKE CONTRL+6,11
!nbre de points dans intin
DPOKE CONTRL+12,*a_handle%
!handle
SWAP *awork_in%,x%()
!échange valeurs tableau/dimension
FOR i%=0 to 10
    DPOKE INTIN+2*i%,x%(i%)
```



(suite)

```
!transfert de tableau
NEXT i%
SWAP *awork_in%,x%()
VDISYS 100
! appel de la fonction 100
*a_handle%=DPEEK(CONTRL+12)
!nouveau handle
RETURN
'--------
'VDI CLOSE VIRTUAL WORK STATION
'
PROCEDURE v_clsvwk(handle%)
DPOKE CONTRL+2,0
DPOKE CONTRL+6,0
DPOKE CONTRL+12,handle%
VDISYS 101
RETURN
'
'-----
'CLEAR WORKSTATION
'
PROCEDURE v_clrwk(handle%)
DPOKE CONTRL+2,0
DPOKE CONTRL+6,0
DPOKE CONTRL+12,handle%
VDISYS 3
RETURN
'
'
'-----------
'SET CLIPPING RECTANGLE
'
PROCEDURE vs_clip(handle%,flag%,x0%,y0%,x1%,y1%)
DPOKE CONTRL+2,2
DPOKE CONTRL+6,1
DPOKE CONTRL+12,handle%
DPOKE INTIN,flag%
DPOKE PTSIN,x0%
DPOKE PTSIN+2,y0%
DPOKE PTSIN+4,x1%
DPOKE PTSIN+6,y1%
'
VDISYS 129
RETURN
```

Remarques :

— L'utilisation de "*" devant une variable signifie qu'au lieu de considérer une variable on a affaire à un pointeur sur cette variable. C'est l'équivalent du & en langage C.

— Les fonctions **appl_exit** et **appl_init** ne sont pas nécessaires en GFA comme en C (voir l'exemple plus haut) car l'interpréteur se charge lui-même de cette tâche.

Le Basic OMIKRON

L'OMIKRON ne dispose d'aucune fonction GEM. Pour charger le GEM, il suffit de taper en début de programme l'instruction :

LIBRARY Gem,"GEM.LIB"

pour pouvoir ensuite utiliser chaque fonction GEM comme vous le feriez avec un sous-programme normal. Reportez-vous au paragraphe précédent et à la bibliothèque AES VDI pour connaître les valeurs des paramètres d'appel.

L'assembleur

C'est le langage pour lequel il faut tout écrire ; ce n'est pas compliqué mais seulement ennuyeux. Il faut être extrêmement méthodique pour ne pas commettre d'erreurs d'aiguillage dans l'utilisation des tableaux précédemment cités. En récompense les programmes assembleur seront toujours les plus rapides et les plus concis. Le GEM, comme le BIOS et le XBIOS, s'appelle par un **trap**. La différence essentielle est que les paramètres et les numéros de fonction sont passés sur des tableaux et non sur la pile.

Voici la structure de base d'un programme de base pour programmer en assembleur les fonctions VDI et AES. Pour tout programme assembleur par la suite, il vous suffira de recopier l'initialisation qui est donnée ci-dessous et de mettre votre programmme à l'étiquette **prog** (c'est ce qui sera fait dans tous les exemples ultérieurs).



Programme init en assembleur :

```
;PROGRAMME D'INITIALISATION ROUTINES VDI
gemdos    equ 1
xbios     equ 14
bios      equ 13

          .TEXT

;initialisation du programme
;sauvegarde de l'ancien pointeur de pile

          movea.l   sp,a5
          lea.l     stackend,sp

;place mémoire pour setblock de gemdos(numéro de fonction $4a)

          movea.l   4(a5),a5          ;header
          move.l    $c(a5),d0         ;secteur proramme
          add.l     $14(a5),d0        ;données initialisées
          add.l     $1c(a5),d0        ;données non initialisées
          addi.l    #$100,d0          ;longueur du header
          move.l    d0,-(sp)

;dans d0 on a la longueur du secteur mémoire
; à réserver pour paramètre du TOS

          move.l    a5,-(sp)
          move.w    #0,-(sp)
          move.w    #$4a,-(sp)
          trap      #gemdos
          adda.l    #12,sp
          tst.l     d0
          bne       error

;appel du programme
          bsr       main

; attente d'une touche pour sortir
          move.w    #1,-(sp)
          trap      #gemdos
          addq.l    #2,sp

;retour au programme appelant
error:    clr.l     (sp)
          trap      #gemdos
```

(suite)

```
;Routine d'appel général de fonction VDI et AES
;-------------
;appel de fonction VDI

vdi:        movem.l   d0/d1,-(sp)
            move.l    #vdipb,d1          ;bloc paramètres vdi
            move.w    #$73,d0            ;numéro dans la
bibliothèque
            trap      #2
            movem.l   (sp)+,d0/d1
            rts

;Routine d'appel général de fonction AES
;-------
; appel de fonction AES

aes:        movem.l   d0/d1,-(sp)
            move.l    #aespb,d1          ;bloc paramètres AES
            move.w    #$c8,d0            ;numéro dans la
bibliothèque
            trap      #2
            movem.l   (sp)+,d0/d1
            rts
;-------


;c'est ici qu'il faudra inclure votre programme
;utilisant les fonctions VDI ou AES
;initialisation et ouverture des stations de travail
; on met tous les paramètres à zéro

main:       move.w    #10,opcode
            clr.w     aintin
            clr.w     aaddrin
            clr.l     ap1resv
            clr.l     ap2resv
            clr.l     ap3resv
            clr.l     ap4resv
            jsr       aes

            move.w    #77,opcode
            clr.w     aintin
            clr.w     aaddrin
            move.w    #5,aintout
            jsr       aes
            move.w    intout,handle
```



(suite)

```
                move.w    #100,opcode          ;opcode de v_opnvwk
                clr.l     sptsin
                move.w    #11,sintin           ; 11 valeurs dans le
;tableau intin
                move.w    #1,intin
                move.w    #1,intin+2
                move.w    #1,intin+4
                move.w    #1,intin+6
                move.w    #1,intin+8
                move.w    #1,intin+10
                move.w    #1,intin+12
                move.w    #1,intin+14
                move.w    #1,intin+16
                move.w    #1,intin+18
                move.w    #2,intin+20          ;coordonnées raster
                jsr       vdi
                move.w    intout,xrsol         ;resolution en x
                move.w    intout+2,yrsol       ;resolution en y
                move.w    #3,opcode            ;opcode de v_clrwk
                clr.w     sintin
                clr.w     sptsin
                jsr       vdi                  ;effacement de l'écran

; ici commence le corps de votre programme
prog:
;
;
;
;
;

; fin du programme
                move.w    #101,opcode          ;opcode de v_clsvwk
                clr.w     sintin
                clr.w     sptsin
                jsr       vdi

                move.w    #19,opcode           ;appl exit
                clr.l     aintin
                clr.l     aaddrin
                jsr       aes
                rts
```

(suite)
```
;-------------
;secteurs de données
                .DATA
;paramètres AES
aespb:          .DC.l contrl
                .DC.l global
                .DC.l intin
                .DC.l intout
                .DC.l addrin
                .DC.l addrout

;paramètres VDI
vdipb:          .DC.l contrl
                .DC.l intin
                .DC.l ptsin
                .DC.l intout
                .DC.l ptsout

;tableaux communs à l'AES et à la VDI
;revoir les définitions des tableaux au début de l'article
contrl:
opcode:         .DS.w 1     ;numéro de fonction

aintin:
sptsin:         .DS.w 1

aintout:
sptsout:        .DS.w 1

aadrin:
sintin:         .DS.w 1

subcode:  .DS.w  1      ; numéro de sous-fonction

handle:         .DS.w 1
                .DS.w 4
;tableau global pour l'AES uniquement

global:
apvers:         .DS.w 1
apcount:        .DS.w 1
apid:           .DS.w 1
appriv:         .DS.l 1
apptree:        .DS.l 1
ap1resv:        .DS.l 1
ap2resv:        .DS.l 1
ap3resv:        .DS.l 1
ap4resv:        .DS.l 1
```



(suite)

```
;tableau intin
intin:      .DS.w 128

;tableau ptsin
ptsin:      .DS.w 256

;tableau intout
intout:     .DS.w 128

:tableau    ptsout
ptsout:     .DS.w 12

;tableau addrin (AES uniquement)
addrin:     .DS.w 128

;tableau addrout (AES uniquement)
addrout:    .DS.w 128

;vos données initialisées
xrsol:      .DS.w  1
yrsol:      .DS.w  1
;Données non initialisées
            .BSS
            .EVEN
stkstart: .DS.l 400
stackend: .DS.l 1

;vos données non initialisées
;
;
            .END
```

Par la suite, tous les exemples de programmes assembleurs traitant du VDI ou de l'AES ne reprendront pas l'initialisation présentée ici. Il vous suffira d'inclure vos données et votre programme aux endroits réservés dans la présentation.

Exemple de programme de dessin en VDI

Nous nous proposons de tracer un cercle dans un rectangle aux coins arrondis. La sortie du programme se fait par pression d'une touche.

Version en langage C (n'oubliez pas d'inclure le fichier d'initialisation défini plus haut) :

```
#include <osbind.h>
int pxyarray[4];/*2 points soit 4 coordonnées*/

main()
{
     gem_init();
     Cconws ("\33E");       /*effacer l'écran avec une séquence
escape*/

     pxyarray[0] = 10;
     pxyarray[1] = 10;
     pxyarray[2] = x_max-40;
     pxyarray[3] = y_max-20;

     v_rbox (handle, pxyarray);
     v-ellipse(handle, x_max/2, y_max/2-20, x_max/2-20,y_max/2-30);

     Ccrawcin();            /*attendre touche*/

     gem_exit();
}
```

Version en Basic OMIKRON :

```
LIBRARY Gem, "GEM.LIB"
Appl_Init
V-opnvwk
CLS
' Déterminer la résolution par Getrez
X_max=640
Y-max=400
XBIOS (Rez,4)
IF Rez<2 THEN Y_max=200
IF Rez=0 THEN X_max=320

'tracer les deux figures
V_Rbox(10,10,X_max-40,Y_max-20)
V_Ellipse(X_max/2,Y_max/2-20,X_max/2-20,X_max/2-30)

'attente d'une touche
GEMDOS (,7)' Crawcin

V_Clsvwk
Appl_Exit
END
```



Version en assembleur :

```
main:   jsr     gem_init
        pea     clrscr                  ;effacer l'écran VT52
        move.w  #9,-(sp)
        trap    #1
        addq.l  #6,sp

;---------tracé du rectangle--------

        move.w  #11,contrl              ;V_Rbox
        move.w  #2,contrl+2
        move.w  #0,contrl+4
        move.w  #3,contrl+6
        clr.w   contrl+8
        move.w  #8,contrl+10            ;sous-fonction
        move.w  handle,contrl+12

;coordonnées du rectangle

        move.w  #10,ptsin
        move.w  #10,ptsin+2
        move.w  x_max,d0
        subi.w  #40,d0
        move.w  d0,ptsin+4
        move.w  y_max,d0
        subi.w  #20,d0
        move.w  d0,ptsin+6

;tracer le rectangle:
        jsr     vdi

;--------tracé de l'ellipse--------

        move.w  #11,contrl              ;V_Ellipse
        move.w  #2,contrl+2
        move.w  #0,contrl+4
        move.w  #3,contrl+6
        clr.w   contrl+8
        move.w  #5,contrl+10            ;sous fonction
        move.w  handle,contrl+12

;coordonnées du centre et rayons

        clr.l   d0
        move.w  x_max,d0
        divu.w  #2,d0
        move.w  d0,ptsin                ;abcisse
        subi.w  #20,d0
```

(suite)

```
                move.w    d0,ptsin+4          ;rayon en x
                clr.l     d0
                move.w    y_max,d0
                divu.w    #2,d0
                subi.w    #20,d0
                move.w    d0,ptsin+2          ;ordonnée
                subi.w    #10,d0
                move.w    d0,ptsin+6          ;rayon en y

;tracer l'ellipse
                jsr       vdi

                .DATA
clrscr:         .DC.b 27,'E',0
                .END
```



5/3

Les routines de la LINE_A et leur utilisation

Dans les ROM du ST, il existe une quinzaine de routines graphiques de base, grâce auxquelles le GEM peut tracer fenêtres, menus et formulaires. Elles sont appelées les routines de la LINE_A car leur mise en œuvre est assez particulière : elle se fait par l'intermédiaire d'une instruction « dc.w ».

(appel par dc.w $a000) :
— LINE A INIT

Initialisation des routines LINE_A. Renvoie l'adresse de base de tous les paramètres pouvant être utilisés.

Paramètres d'entrée :

Aucun.

Paramètres de sortie :

— Adresse de début des variables LINE_A dans les registres d0 et a0.

— Adresse d'un tableau de trois pointeurs sur les headers des polices système.

— Adresse d'un des tableaux de données de police (suivant la résolution en cours).

(appel par dc.w $a001) :
— PUT PIXEL

Allume un pixel. L'origine des graduations est située en haut à gauche de l'écran. C'est à vous de contrôler les coordonnées maximales pouvant être atteintes suivant la résolution, la routine n'effectuant aucune vérification de ce type.

Paramètres d'entrée :
— INTIN : Couleur du pixel
— PTSIN : Coordonnée x du pixel
— PTSIN+2 : Coordonnée y du pixel

Paramètres de sortie :
Aucun.

(appel par dc.w $a002)
— GET PIXEL

Détermine la couleur d'un pixel (voir fonction **$a001**).

Paramètres d'entrée :
— PTSIN : Coordonnée x du pixel
— PTSIN+2 : Coordonnée y du pixel

Paramètres de sortie :
— Couleur du pixel dans le registre d0

(appel par dc.w $a003)
— LINE

Trace une droite entre deux points de coordonnées xdeb, ydeb et xfin, yfin.

Paramètres d'entrée :
— _fg_bp_1 à _fg_bp_4 : Valeur de la couleur pour les plans
— _ln_mask : Echantillon de ligne
— _wrt_mode : Type de mode d'écriture (0 à 3)
— _1stlin : Réservé. Doit se trouver à −1
— _x1, _y1, _x2, _y2 : Coordonnées des points de début et de fin de ligne

Paramètres de sortie :
Aucun.



(appel par dc.w $a004)
— HORIZONTAL LINE

Identique à la fonction précédente uniquement pour les droites horizontales. L'avantage de cette fonction est sa rapidité sans comparaison avec LINE.

Paramètres d'entrée :

— _fg_bp_1 à _fg_bp_4 : Valeur de la couleur pour les plans
— _ln_mask : Echantillon de ligne
— _wrt_mode : Type de mode d'écriture (0 à 3)
— _x1, _y1, _x2, _y2 : Coordonnées des points de début et de fin de ligne
— _patptr : Adresse de l'échantillon de ligne
— _patmsk : Longueur de l'échantillon pointé par _patptr en (octets/2) – 1

Paramètres de sortie :

Aucun.

(appel par dc.w $a005)
— FILLED RECTANGLE

Traçage de rectangle pouvant être rempli ou non.

Paramètres d'entrée :

— _fg_bp_1 à _fg_bp_4 : Valeur de la couleur pour les plans
— _wrt_mode : Type de mode d'écriture (0 à 3)
— _x1, _y1, _x2, _y2 : Coordonnées des points de début et de fin de ligne
— _patptr : Adresse de l'échantillon de ligne
— _patmsk : Longueur de l'échantillon pointé par _patptr en (octets/2) – 1
— _clip : Drapeau de clipping
— _xmn_clip : Coordonnée x minimum du clipping
— _ymn_clip : Coordonnée y minimum du clipping
— _xmx_clip : Coordonnée x maximum du clipping
— _ymx_clip : Coordonnée y maximum du clipping

Paramètres de sortie :

Aucun.

(appel par dc.w $a006)
— FILLED POLYGONE

Attention :

Contrairement à ce que son nom pourrait laisser croire, cette fonction ne remplit qu'une ligne horizontale du polygone. Ce qui pourrait sembler comme un désavantage est en fait tout le contraire, car à peu près tout type de remplissage est ainsi possible.

Paramètres d'entrée :

— ptsin : Coordonnée x du premier point du polygone
— ptsin + 2 : Coordonnée y du premier point du polygone
— ptsin + 4 : Coordonnée x du deuxième point du polygone
— ptsin + 6 : Coordonnée y du deuxième point du polygone

etc.
— contrl + 2 : Nombre de points formant le polygone
— _fg_bp_1 à _fg_bp_4 : Valeur de la couleur pour les plans
— _wrt_mode : Type de mode d'écriture (0 à 3)
— _y1 : Coordonnées y de la ligne à remplir
— _patptr : Adresse de l'échantillon de ligne
— _patmsk : Longueur de l'échantillon pointé par _patptr en (octets/2) – 1
— _clip : Drapeau de clipping
— _xmn_clip : Coordonnée x minimale du clipping
— _ymn_clip : Coordonnée y minimale du clipping
— _xmx_clip : Coordonnée x maximale du clipping
— _ymx_clip : Coordonnée y maximale du clipping

Paramètres de sortie :

Aucun.

(appel par dc.w $a007)
— BITBLT

Copie d'une portion de mémoire.

Paramètres d'entrée :

Registre a6 : Pointeur sur une table de transfert



Table de transfert :

b_wd (.w)	: Longueur du bloc en pixels
b_ht (.w)	: Hauteur du bloc en pixels
plane_ct (.w)	: Nombre de plans de la destination
fg_col (.w)	: Couleur du premier plan de la destination
bg_col (.w)	: Couleur de l'arrière plan de la destination
op_tab (.l)	: Mode de transfert
s_xmin (.w)	: Valeur x minimale de la source
s_ymin (.w)	: Valeur y minimale de la source
s_form (.l)	: Adresse de base du bloc de la source
s_nxwd (.w)	: Décalage vers le mot suivant dans une ligne de la source
s_nxln (.w)	: Décalage vers la ligne suivante dans une ligne de la source
s_nxpl (.w)	: Décalage vers le plan suivant à partir du début d'une ligne de la source
d_xmin (.w)	: Valeur x minimale de la destination
d_ymin (.w)	: Valeur y minimale de la destination
d_form (.l)	: Adresse de base du bloc de la destination
d_nxwd (.w)	: Décalage vers le mot suivant dans une ligne de la destination
d_nxln (.w)	: Décalage vers la ligne suivante dans une ligne de la destination
d_nxpl (.w)	: Décalage vers le plan suivant à partir du début d'une ligne de la destination
p_addr (.l)	: Adresse du tampon de destination
p_nxln (.w)	: Décalage vers la ligne suivante dans la matrice de remplissage
p_nxpl (.w)	: Décalage vers le plan suivant dans la matrice de remplissage
p_mask (13.w)	: Index de masque de la matrice de remplissage
p_block_len	: Fin de la table de transfert

Paramètres de sortie :

Aucun.

(appel par dc.w $a008)
— TEXTBLT

Affichage d'un caractère à l'écran. Les polices utilisées sont au format GDOS (voir Partie 5, chapitre 2). Tous les attributs classiques peuvent être utilisés.

Paramètres d'entrée :

_wrt_mode	: Type d'écriture
_text_fg	: Couleur du premier plan du texte

_text_bg	: Couleur d'arrière-plan du texte
_fbase	: Pointeur sur le début des datas de la police de caractères
_fwidth	: Nombre de lettres dans le jeu de caractères
_sourcex	: Coordonnée en x du caractère dans les datas de la police
_sourcey	: Coordonnée en y du caractère dans les datas de la police
_destx	: Position en x du caractère à l'écran
_desty	: Position en y du caractère à l'écran
_delx	: Longueur du caractère en pixels
_dely	: Hauteur du caractère en pixels
_style	: Vecteur des effets spéciaux du caractère
_litemask	: Masque pour les caractères grisés
_skewmask	: Masque pour les caractères italiques
_weight	: Taille de l'épaisseur pour les caractères gras
_r_off	: Décalage au-dessus de la ligne de base pour les caractères en italique
_l_off	: Décalage au-dessous de la ligne de base pour les caractères en italique
_scale	: Drapeau de changement échelle (0 = pas de changement d'échelle)
_xacc_dda	: Accumulateur de pente horizontale
_dda_inc	: Facteur d'agrandissement ou de réduction
_t_sclsts	: Drapeau de direction d'échelle
_chup	: Vecteur de rotation des caractères
_mono_status	: Drapeau de police normale ou à espacement proportionnel
_scrtchp	: Pointeur sur le tampon de travail des effets spéciaux
_scrpt2	: Décalage du tampon pour les effets spéciaux

Paramètres de sortie :

Aucun.

(appel par dc.w $a009)
— SHOW MOUSE

Cette fonction affiche la souris à l'écran. Deux fonctionnements sont possibles : soit le nombre d'appel des fonctions $a008 et $a009 est pris en compte, soit il ne l'est pas. Dans le premier cas, si on appelle 5 fois HIDE MOUSE, il faudra également appeler 5 fois SHOW MOUSE pour faire réapparaître la souris.



Paramètres d'entrée :

CONTRL+2 : 0
CONTRL+6 : 1
INTIN : 0 si le nombre d'appel n'est pas pris en compte
1 sinon

Paramètres de sortie :

Aucun.

(appel par dc.w $a00a)
— HIDE MOUSE

Voici le pendant de la fonction précédente qui, lui, efface la souris.

Paramètres d'entrée :

CONTRL+2 : 0
CONTRL+6 : 1
INTIN : 0 si le nombre d'appel n'est pas pris en compte
1 sinon

Paramètres de sortie :

Aucun.

(appel par dc.w $a00b)
— TRANSFORM MOUSE

Toujours au sujet de la souris, cette routine modifie la forme graphique qui lui est associée (une flèche en standard).

Paramètres d'entrée :

INTIN : Coordonnée x du coin supérieur de la forme
INTIN+2 : Coordonnée y du coin supérieur de la forme
INTIN+4 : 1
INTIN+6 : 0
INTIN+8 : 1
INTIN+10 à INTIN+40 : Données du masque de la nouvelle souris
INTIN+42 à INTIN+72 : Données de la nouvelle souris.

Paramètres de sortie :

Aucun.

(appel par dc.w $a00c)
— UNDRAW SPRITE

Cette fonction, en liaison avec la routine $a00d, efface un sprite à l'écran.

(appel par dc.w $a00d)
— DRAW SPRITE

Routine d'affichage d'un sprite à l'écran.
Ces deux routines ne seront pas expliquées, car leur fonctionnement est buggé sur certaines vieilles versions du TOS.

(appel par dc.w $a00e)
— COPY RASTER FORM

Cette routine permet la copie des portions de mémoire. Elle est la primitive de la fonction VDI de copie de blocs. Elle en a donc les caractéristiques, y compris l'utilisation obligatoire du MFD (*Memory Form Definition*).

LISTE DES VARIABLES DE LA LINE_A

L'adresse de chacune des variables est donnée relativement à l'adresse de base renvoyée par l'appel dc.w $a000. Certains paramètres auront sans doute une signification étrange, mais ne craignez rien.

0	2 octets	V_PLANES	Nombre de plans de la mémoire vidéo (1 = mono, 2 = moyenne, 4 = basse)
2	2 octets	V_LIN_WR	Nombre d'octets par ligne de la RAM vidéo (80 en mono, 160 en couleur)
4	4 octets	CONTRL	Adresse du tableau GEM CONTROL
8	4 octets	INTIN	Adresse du tableau GEM INTIN
12	4 octets	PTSIN	Adresse du tableau GEM PTSIN
16	4 octets	INTOUT	Adresse du tableau GEM INTOUT
20	4 octets	PTSOUT	Adresse du tableau GEM PTSOUT
24	2 octets	_FG_BP_1	Etat du plan 1 (toute résolution)
26	2 octets	_FG_BP_2	Etat du plan 2 (moyenne et basse résolution seulement)
28	2 octets	_FG_BP_3	Etat du plan 3 (basse résolution)
30	2 octets	_FG_BP_4	Etat du plan 4 (basse résolution seulement)
32	2 octets	_LSTLIN réservé	Doit toujours être égal à −1
34	2 octets	_LN_MASK	Echantillon de ligne ex : $ffff pour une ligne normale, $cccc pour une ligne pointillée
36	2 octets	_WRT_MODE	Mode d'écriture (0 = remplacement, 1 = transparent, 2 = xor, 3 = transparent inverse)



38	2 octets	_X1	Coordonnée x initiale
40	2 octets	_Y1	Coordonnée y initiale
42	2 octets	_X2	Coordonnée x finale
44	2 octets	_Y2	Coordonnée y finale
46	4 octets	_PATPTR	Echantillon de remplissage
50	2 octets	_PATMSK	Longueur de _PATPTR en (octets/2) – 1
52	2 octets	_MULTIFILL	
54	2 octets	_CLIP	Drapeau d'activation du clipping 0 = non, 1 = oui
56	2 octets	_XMN_CLIP	Coordonnée x initiale du clipping
58	2 octets	_YMN_CLIP	Coordonnée y initiale du clipping
60	2 octets	_XMX_CLIP	Coordonnée x finale du clipping
62	2 octets	_YMX_CLIP	Coordonnée y finale du clipping
64	2 octets	_XACC_DDA	Doit être positionné à $8000
66	2 octets	_DDA_INC	Fraction de mise à l'échelle d'un caractère
68	2 octets	_T_SCLSTS	Drapeau de réduction/agrandissement 0 = réduction, 1 = agrandissement
70	2 octets	_MONO_STATUS	0 = police proportionnelle, 1 = police non proportionnelle
72	2 octets	_SOURCEX	Coordonnée x de la lettre à afficher dans la police
74	2 octets	_SOURCEY	Coordonnée y de la police à afficher dans la police
76	2 octets	_DESTX	Coordonnée x d'affichage de la lettre
78	2 octets	_DESTY	Coordonnée y d'affichage de la lettre
80	2 octets	_DELX	Longueur du caractère
82	2 octets	_DELY	Hauteur du caractère
84	4 octets	_FBASE	Pointeur sur le début de la table des données des caractères
88	2 octets	_FWIDTH	Taille de la police de caractères
90	2 octets	_STYLE	Drapeau des effets spéciaux des caractères. bit 0 = gras, bit 1 = grisé, bit 2 = italique, bit 3 = souligné, bit 4 = contours
92	2 octets	_LITEMASK	Masque pour le type grisé
94	2 octets	_SKEWMASK	Masque pour le type italique
96	2 octets	_WEIGHT	Elargissement des caractères (type gras)
98	2 octets	_R_OFF	Décalage au-dessus de la ligne de base (type italique)
100	2 octets	_L_OFF	Décalage au-dessous de la ligne de base (type italique)
102	2 octets	_SCALE	Drapeau pour le changement d'échelle (0 = non, 1 = oui)
104	2 octets	_CHUP	Vecteur de rotation de caractères (0 = pas de rotation, 900 = rotation de 90 deg., 1800 = de 180 deg., 2700 = 270 deg.) (sens de rotation inverse au sens trigonométrique classique)
106	2 octets	_TXT_FG	Couleur du premier plan pour l'affichage de texte
108	4 octets	_SCRTCHP	Adresse d'un tampon de travail pour les effets spéciaux du texte
112	2 octets	_SCRPT2	Décalage du tampon d'échelle par rapport à _SCRTCHP
114	2 octets	_TEXT_BG	Couleur d'arrière-plan du texte
116	2 octets	_COPYTRAN	Drapeau de type de copie de zone pour BITBLT

5/3.1

Quelques exemples d'appels de fonctions

Tracé d'une ligne diagonale

```
;
; cet exemple trace une ligne diagonale
; allant du coin supérieur gauche au
; coin inférieur droit de l'écran

v_planes equ 0
_fg_bp_1 equ 24
_fg_bp_2 equ 26
_fg_bp_3 equ 28
_fg_bp_4 equ 30
_lstlin equ 32
_ln_mask equ 34
_wrt_mode equ 36
_x1 equ 38
_y1 equ 40
_x2 equ 42
_y2 equ 44

; initialisation
 dc.w $a000
 move.l d0,line_a_basepage      ; adresse des variables line_a
 move.w #639,x_max
 move.w #399,y_max
 move.l line_a_basepage,a0
 cmp.w #1,v_planes(a0)          ; haute résolution? si oui->suite1
 beq.s suite1
 move.w #199,y_max
```



(suite)

```
    cmp.w #2,v_planes(a0)          ; moyenne résolution? si oui->suite2
    beq.s suite1
    move.w #319,x_max              ; c'est donc la basse résolution
suite1:
    move.w x_min,_x1(a0)           ; x initial à 0
    move.w y_min,_y1(a0)           ; y initial à 0
    move.w x_max,_x2(a0)           ; x final suivant la résolution (319
                                   ; ou 639)
    move.w y_max,_y2(a0)           ; y final suivant la résolution (199
                                   ; ou 399)
    move.w #-1,_fg_bp_1(a0)        ; couleur
    clr.w _fg_bp_2(a0)
    clr.w _fg_bp_3(a0)
    clr.w _fg_bp_4(a0)
    move.w #-1,_ln_mask(a0)        ; ligne continue
    clr.w _wrt_mode(a0)    ; en mode remplacement
    move.w #-1,_lstlin(a0)
    dc.w $a003             ; et appel de la fonction en ROM
    ...
    etc...

line_a_basepage:
    ds.l 1
x_min:
    ds.w 1
y_min:
    ds.w 1
x_max:
    ds.w 1
y_max:
    ds.w
```

Tracé d'un carré

```
;
; cet exemple trace un carré rempli en rayé
; de 100 pixels de coté
; coin inférieur droit de l'écran
; avec la fonction $a005 (filled rectangle)



v_planes equ 0
_fg_bp_1 equ 24
_fg_bp_2 equ 26
_fg_bp_3 equ 28
_fg_bp_4 equ 30
_wrt_mode equ 36
_x1 equ 38
_y1 equ 40
_x2 equ 42
_y2 equ 44
_patptr equ 46
_patmsk equ 50
_clip equ 54


; initialisation
 dc.w $a000
 move.l d0,line_a_basepage     ; adresse des variables line_a
 move.l line_a_basepage,a0
suite1:
 move.w #100,_x1(a0)    ; x initial à 0
 move.w #100,_y1(a0)    ; y initial à 0
 move.w #200,_x2(a0)    ; x final suivant la résolution (319
                        ; ou 639)
 move.w #200,_y2(a0)    ; y final suivant la résolution (199
                        ; ou 399)
 move.w #-1,_fg_bp_1(a0)       ; couleur
 clr.w _fg_bp_2(a0)
 clr.w _fg_bp_3(a0)
 clr.w _fg_bp_4(a0)
 move.l #fill,_patptr(a0)
 move.w #0,_patmsk(a0)
```



(suite)

```
 clr.w _wrt_mode(a0)    ; en mode remplacement
 clr.w _clip(a0)
 dc.w $a005             ; et appel de la fonction en ROM
 ...
 etc...

fill:
 dc.w $ff00
line_a_basepage:
 ds.l 1
x_min:
 ds.w 1
y_min:
 ds.w 1
x_max:
 ds.w 1
y_max:
 ds.w 1
```

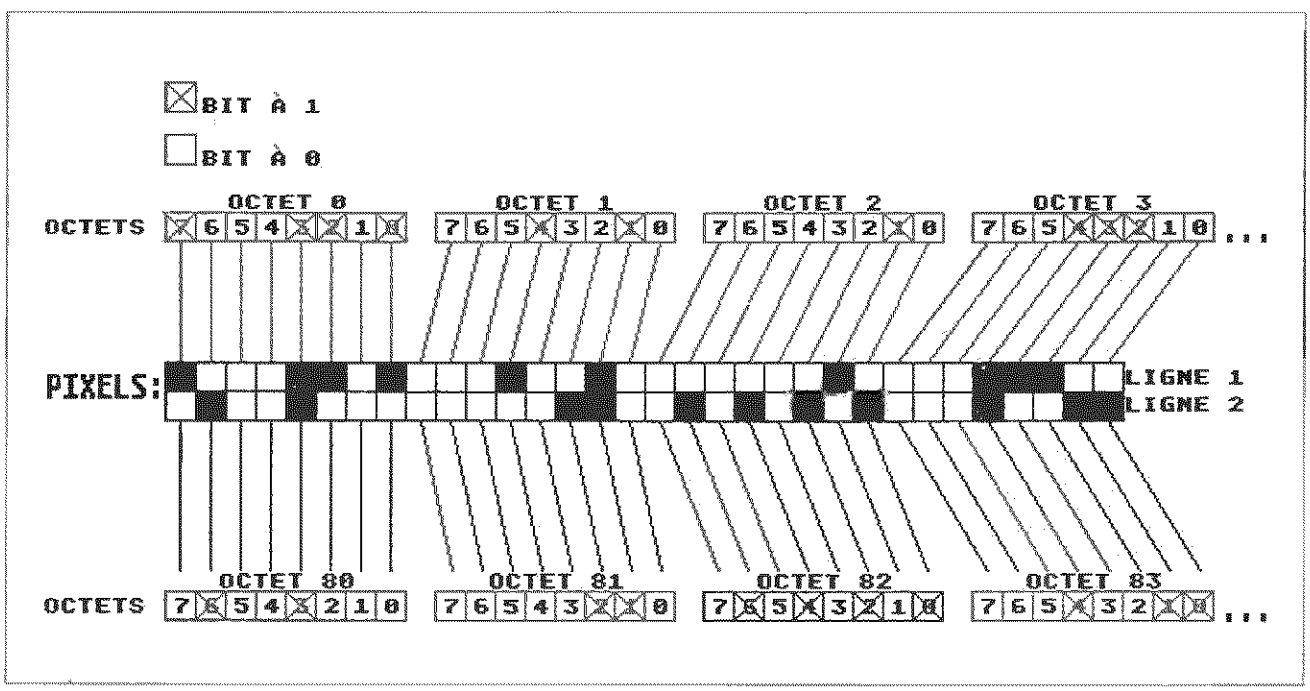
5/3.2

La structure de la mémoire vidéo

L'Atari ST possède 3 résolutions différentes :

La basse et la moyenne (de respectivement 320 × 200 en seize et 640 × 200 pixels en quatre couleurs) accessibles avec un moniteur couleur, ou la haute résolution en 640 × 400 pixels monochromes (2 couleurs disponibles : le noir et le blanc). Quel que soit le mode choisi, le mémoire utilisé par l'ordinateur pour sa gestion est de 32000 octets.

La haute résolution



Structure de la mémoire vidéo en haute résolution.

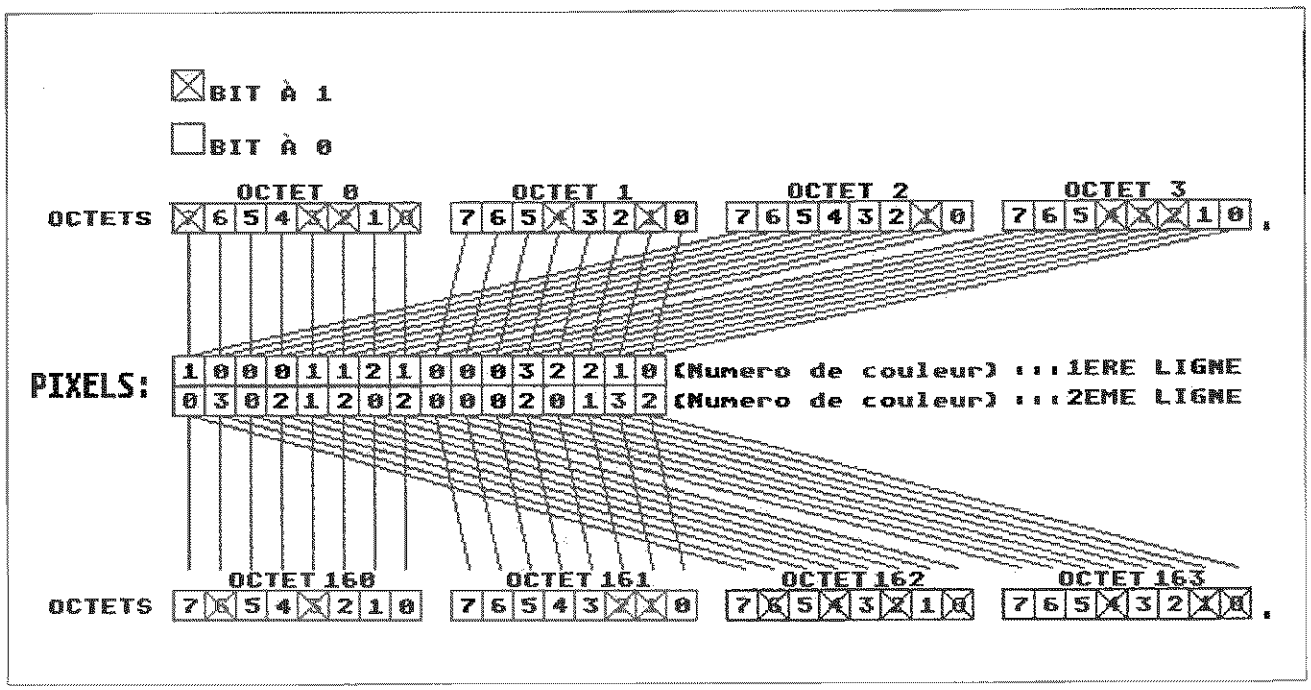
Ce mode de résolution permet la plus grande finesse d'affichage. De plus, la haute qualité des moniteurs Atari SM 124 et SM 125 la prédestine aux applications professionnelles, car la fatigue visuelle est alors quasiment nulle.

L'ordinateur gère 640 × 400 = 256000 pixels. 256000/32000 octets donnent 8 pixels par octet. Or, un octet est composé de 8 bits. A chaque bit de la mémoire vidéo est associé un pixel. Celui-ci sera allumé ou éteint selon la valeur du bit correspondant (1 ou 0).

Chaque ligne a une longueur de 80 octets (80 × 8 = 640 pixels). Si la mémoire vidéo est placée à l'adresse X, le début de la seconde ligne est en X + 80, celui de la troisième en X + 160, etc.

La moyenne résolution

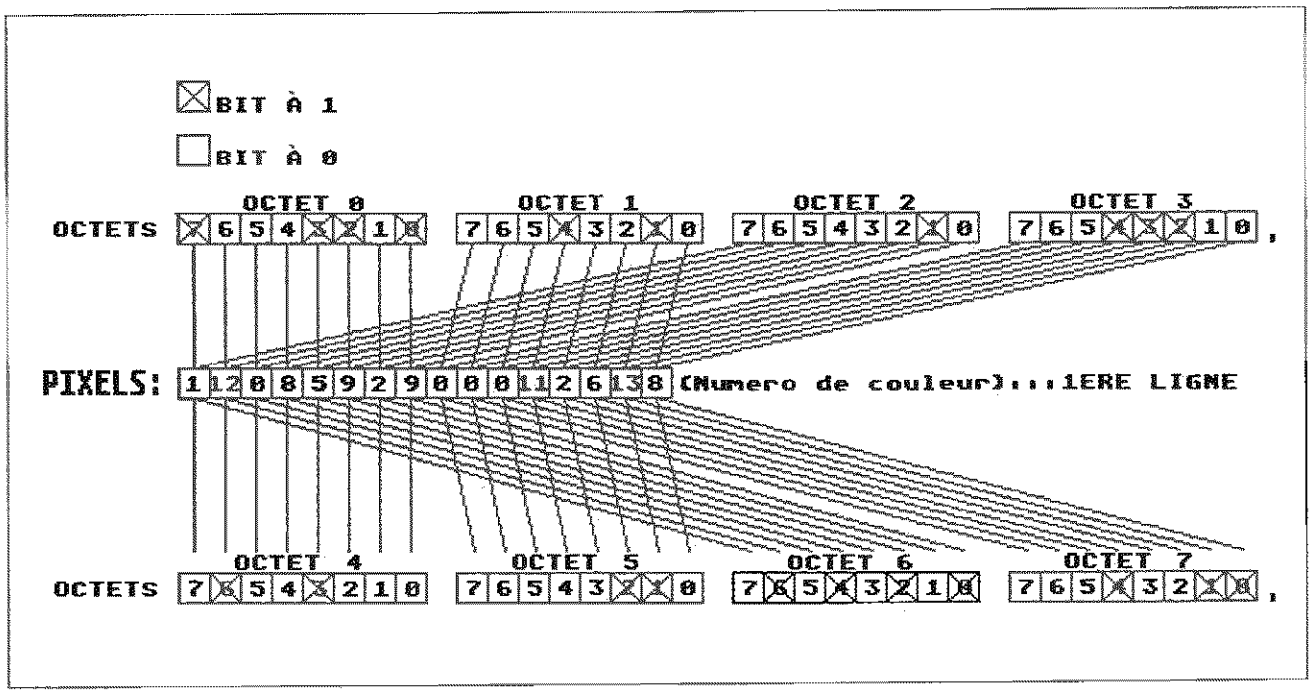
Cette résolution assez batarde est surtout utile pour faire fonctionner les programmes professionnels en couleur. Quatre couleurs sont disponibles (zéro à trois). Deux bits sont nécessaires pour coder le numéro de couleur de chaque pixel. Une ligne a une couleur de cent soixante octets divisés par groupes de 4. Chaque groupe de 2 octets fournit la couleur de 16 pixels différents.



Structure de la mémoire vidéo en moyenne résolution.

La basse résolution

C'est la plus souvent utilisée pour les jeux. L'écran est alors divisé en $320 \times 200 = 64000$ pixels. Vous avez alors accès à seize couleurs (en théorie, seulement ! Rassurez-vous...). Chaque pixel est codé sur quatre bits. Sur ces bits, on stocke le numéro de la couleur choisie, numéro allant de zéro à quinze.



Structure de la mémoire vidéo en basse résolution.

En fait, plus de 16 couleurs sont affichables. Il suffit pour cela de programmer une interruption s'exécutant à chaque retour de ligne du moniteur, et de changer la palette des couleurs à chaque fois. En pratique, c'est un peu plus compliqué que cela.

Remarque :

Cette technique marche aussi pour la moyenne résolution, et même pour changer de résolution au milieu de l'image. L'Atari, même si ces possibilités peuvent sembler, à la base, faibles face à son principal concurrent l'AMI..., est en fait une machine très ouverte qui permet de nombreux bricolages logiciels.

5/4

Les programmes de dessin du marché

— **SPECTRUM 512**

(développé par ANTIC, édité par Upgrade Editions)
Basse résolution.

SPECTRUM est un programme de dessin aux possibilités uniques et d'une puissance rare sur un micro-ordinateur.

Voici ses principales fonctions :

— 512 couleurs disponibles simultanément à l'écran avec un maximum de 48 couleurs par ligne ;

— dessin à main levée ;

— fonctions polygone ouvert ou fermé, ellipse ou cercle ;

— spray (aérographe) de taille et de vitesse variable avec possibilité de cyclage des couleurs ;

— pinceaux de taille variable avec possibilité de cyclage des couleurs ;

— remplissage de surfaces avec patterns modifiables par l'utilisateur ;

— modification de la palette utilisée avec des options uniques : augmentation ou diminution de la luminosité ;

— couper/coller de blocs en modes déplacement, copie, opaque ou transparent avec déformations possibles ;

— buffer de travail pour entreposer des blocs ;

— possibilité d'utilisation de masque pour le remplissage ;

— antialiasing puissant et paramétrable ;

— lissage de lignes, polygones cercles ou ellipses ;

— gomme.

La puissance sans équivalent de SPECTRUM 512 en fait un produit unique, mais d'utilisation assez complexe. Un temps d'apprentissage est donc nécessaire avant d'utiliser ses fonctions les plus puissantes sans problème.

Heureusement, le manuel est fort bien conçu et « prend en main » l'utilisateur novice afin que celui-ci ne se sente pas perdu au milieu de toutes les fonctions.

— **DEGAS ELITE**

(développé par Batteries Included, édité par Electronic Arts)
Toutes résolutions.

Degas Elite est un des premiers programmes de dessin apparu sur l'Atari ST. A ce titre, il a bien sûr quelque peu vieilli, mais sa facilité d'utilisation est telle qu'il reste toujours utilisé par de nombreuses personnes. Contrairement à SPECTRUM 512, il n'utilise que 16 couleurs en basse résolution, mais possède l'avantage de fonctionner dans tous les modes graphiques de l'ordinateur (320×200 16 couleurs, 640×200 4 couleurs et 640×400 monochrome).

Mis à part ce détail, ses possibilités sont tout à fait classiques :

— dessin à main levée ;

— tracé des droites, boîtes, cercles, ellipses, polygones ;

— remplissage de surfaces ;

— fonction texte ;

— rotation ou déformations ou diverses transformations de blocs ;

— fonction texte avec les attributs standards (*Underlined, Skewed, Outlined, Thickened,* etc.) ;

— spray réglable ;

— deux écrans sur 520ST et 8 sur 1040ST ;

— gomme ;

— loupe de taille variable ;

etc.

En bref, un bon outil d'initiation, aux fonctions classiques mais très intuitif pour l'utilisateur : la lecture du manuel n'est même pas nécessaire avant son utilisation.

Les seuls défauts de ce programme sont peut-être son trop grand âge qui fait qu'il soit maintenant dépassé du point de vue performances, ainsi que sa relative lenteur pour toutes les opérations de manipulation de blocs.

— **CYBER STUDIO**

(développé par ANTIC, édité par Upgrade Editions)
Moyenne et haute résolutions

CYBER STUDIO est un logiciel de création et d'animation de formes en 3 dimensions. Etant à notre connaissance le seul sur ST entièrement dédié à la 3D, c'est donc actuellement le meilleur.

Ses principales caractéristiques sont :

— fonctions Spin et Extrude pour la génération automatique d'objet 3D à partir d'une forme 2D ;



— visualisation des formes créées en mode fil de fer, faces cachées ou faces cachées avec gestion des ombres ;

— réunions diverses d'objets pour la création de formes complexes ;

— animations en temps réel de formes en mode fil de fer ;

— trois sources lumineuses réglables indépendamment.

D'une manipulation aisée, il constitue un bon outil pour s'initier à la 3D et même pour réaliser de petits films. Attention, l'utilisation d'un 1040 ST minimum est indispensable, ce logiciel étant très gourmand en mémoire. Il peut aussi être complété par le logiciel CYBER CONTROL, véritable langage dédié à la 3D. D'une utilisation assez ardue (c'est une sorte de langage Forth), CYBER CONTROL se révèle pourtant un outil puissant lorsqu'il est convenablement utilisé.

Le seul gros reproche qui puisse être fait à CYBER STUDIO est sa lenteur de rafraîchissement des fenêtres et de calcul de l'intersection de plusieurs objets.

Attention :

La société UPGRADE a récemment annoncé la sortie prochaine de CYBER SCULPT, modeleur 3D complétant la gamme CYBER. Ayant eu la chance d'assister à une démonstration de ce logiciel, je peux vous affirmer qu'il est d'une puissance rarement atteinte, et très rapide... Les formes les plus complexes peuvent aisément être réalisées, ce qui tenait du tour de force avec CYBER STUDIO.

— ZZ LAZY PAINT

(développé et édité par Human Technologies)
Haute résolution.

ZZ LAZY PAINT (LAZY pour les intimes), est un outil graphique sans équivalent sur la gamme ST. Non content de s'arrêter à la pauvre résolution monochrome de 640×400 pixels monochromes, LAZY vous permet d'exploiter au maximum votre imprimante laser (Postcript ou Atari) en travaillant en 300 dpi (*Dop Per Inch* ou points par pouce). Il intègre toutes les fonctions classiques, avec quelques plus :

— vous pouvez utiliser l'ensemble des outils en mode loupe ;

— un driver pour scanner Canon IX12 ou IX12F est intégré ;

— vous pouvez filtrer l'image ou déparasiter une zone ;

des sauvegardes au format IMG ou POSTCRIPT sont prévues ;

— tous les outils de dessin sont réglables ;

— l'impression sur laser Atari est d'une impressionnante rapidité ;

— des utilitaires disquette sont implantés.

En bref, un superbe logiciel de dessin monochrome, pour peu que vous possédiez une imprimante laser.

5/4.1

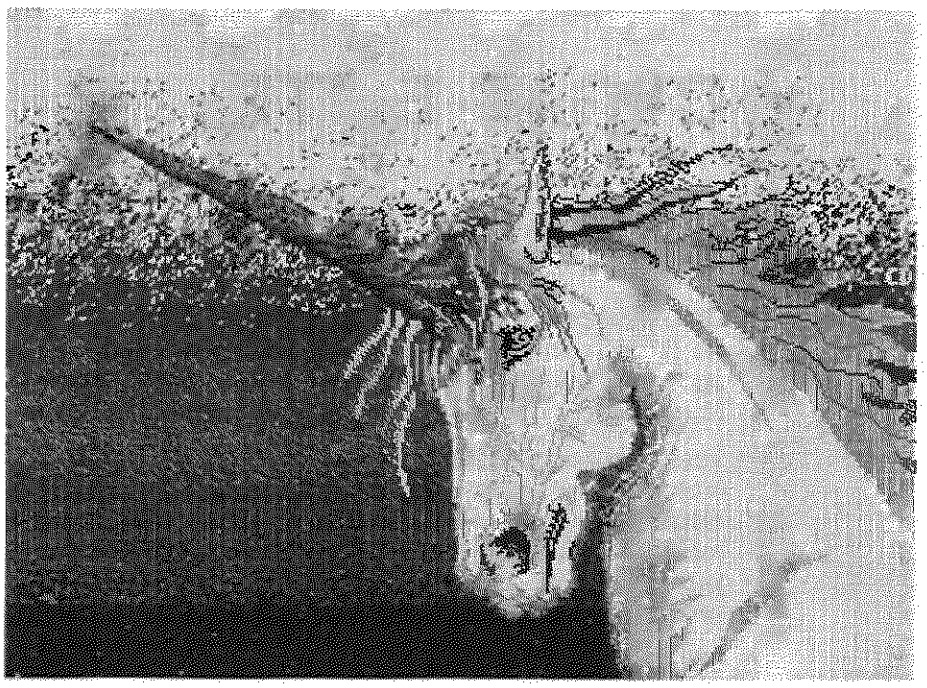
DEGAS ELITE

Créé par Tom Hudson et édité par Batteries Included en 1987, ce logiciel de dessin succède à la première version de Degas. Dès sa sortie il s'est acquis les faveurs des graphistes et bon nombre de jeux ont vu et voient toujours leurs décors et leurs personnages naître de ses pinceaux.

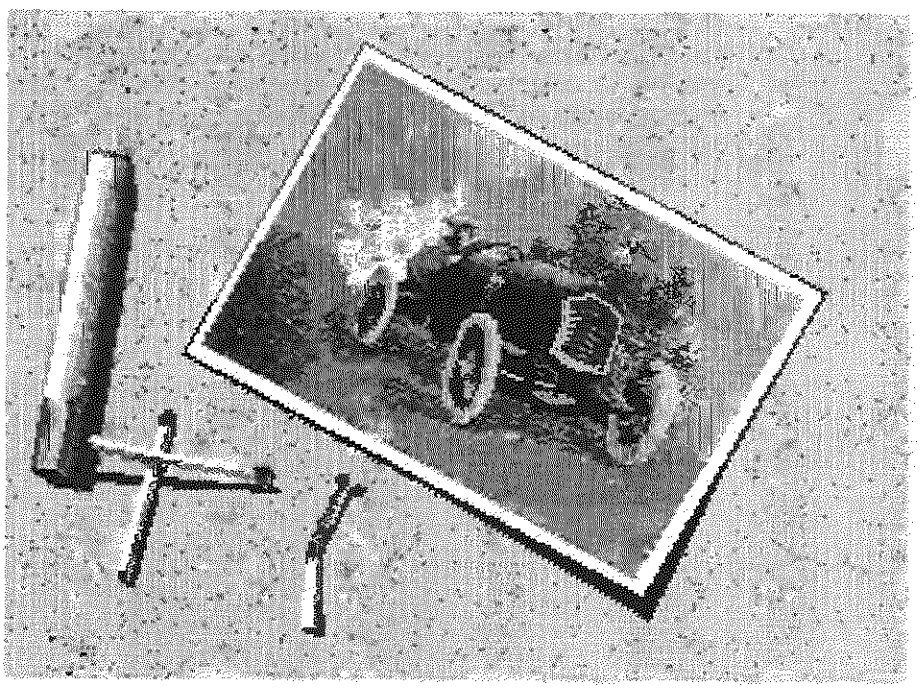
Tout ce qui est réalisable en bit-map peut l'être depuis Degas, et ses seules limites résident dans le talent de ses utilisateurs ainsi qu'en témoignent les dessins qui illustrent ce chapitre. Ils ont été réalisés sur 520 ST spécialement pour Weka par Didier Cackel, graphiste d'une équipe qui a plusieurs jeux à son actif. L'impression monochrome atténue malheureusement l'impact produit par les 16 couleurs qu'il avait choisies au départ.

Degas se révèle simple d'emploi car il n'égare pas l'utilisateur dans un dédale de menus, de sous-menus, de sous-sous-menus... Tom Hudson a joué habilement avec la touche droite de la souris de l'Atari pour faire alterner tantôt une page de dessin plein écran, tantôt un unique menu présentant simultanément toutes les fonctions disponibles.

Les paramètres éventuellement liés à chaque fonction peuvent à chaque fois être modifiés par appel d'une boîte de dialogue. Celle-ci s'obtient soit en double-cliquant sur la case de la fonction, soit en ouvrant les menus déroulants SET ou MAKE.



« Licorne »



« Photo »

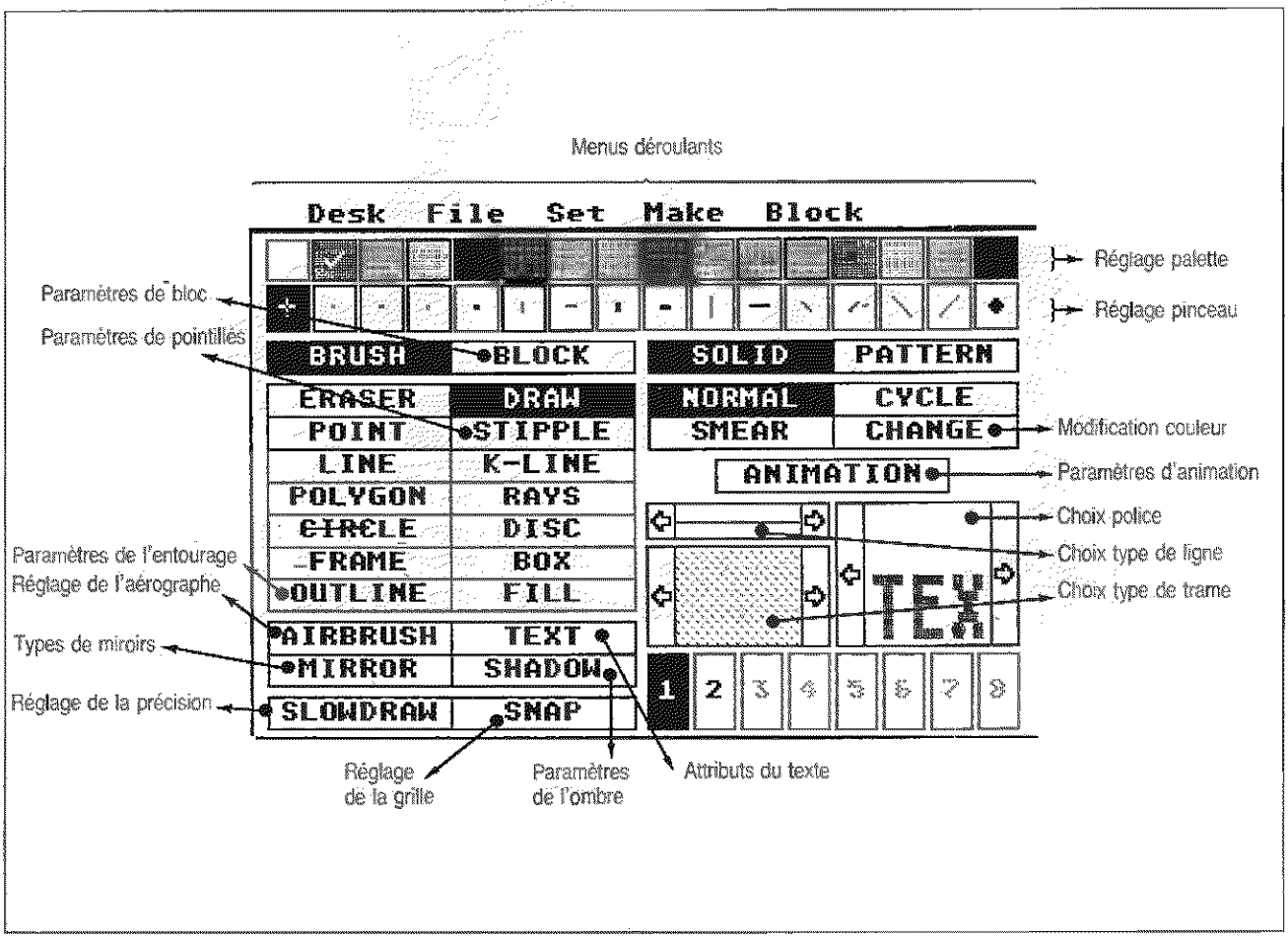


« Pépère »



« Fantasme »

Le schéma suivant récapitule les cases donnant accès à un sous-menu permettant un réglage, une modification des paramètres, ou un choix d'éléments :



Cases donnant accès à une boîte de dialogue.

Degas fait partie des rares logiciels permettant de travailler une même image dans des résolutions d'écran différentes. C'est cependant la basse résolution qui produit les œuvres les plus élaborées car on dispose alors de seize couleurs simultanément. Les quatre couleurs de la moyenne résolution sont décidément trop étriquées pour permettre une réelle liberté d'expression et si le monochrome peut faire retrouver le style accrocheur de certaines bandes dessinées des années cinquante, il ne semble pas avoir la faveur des amateurs de jeux. Dans ce mode, les possibilités de plusieurs fonctions sont réduites, voire deviennent indisponibles. Tout transfert de dessin d'une résolution à l'autre, pour automatique qu'il soit, nécessite cependant un travail de retouche manuel au niveau de la palette des couleurs ou du choix des trames. Le choix basse ou moyenne résolution doit être réalisé depuis le bureau GEM (Option/Définir les préférences) avant de charger DEGAS (DEGELITE.PRG).



QUELQUES CONSEILS AVANT DE COMMENCER

Il arrive souvent qu'un tracé ne soit pas correct au premier essai, aussi une des touches les plus importantes de Degas est la touche **UNDO** qui annule le dernier tracé réalisé, à condition qu'aucune touche du clavier ou de la souris n'ait été frappée entre temps. En cours de dessin, on peut également interrompre le tracé d'une figure en appuyant sur le bouton droit de la souris.

Une bonne habitude de travail consiste à gérer intelligemment les deux pages écran que DEGAS propose même dans la version mémoire la plus limitée des ATARI (520 ST). La première page sert à réaliser le dessin. Chaque fois qu'une étape nouvelle de réalisation est franchie, le dessin est transféré dans la page numéro 2 (cliqué, glissé depuis le menu principal). En cas de fausse manœuvre par la suite (remplissage de tout l'écran, effacement accidentel, trait qui gâche tout...) on peut toujours récupérer l'image sauvegardée en page 2. Cette page fonctionne alors comme sauvegarde mais sans être affectée de la lenteur d'un banal lecteur de disquette. Le passage d'une page de dessin à l'autre s'effectue en cliquant la case correspondante (1 ou 2 sur 520 ST, 1 à 8 sur 1040 ST).

I - Des outils pour dessiner

LES OUTILS DE BASE

DRAW Dessiner

Draw permet le dessin à main levée. L'effet produit dépend de la couleur sélectionnée et de la forme du pinceau (BRUSH) ou du bloc (BLOCK) choisi.

ERASER Gommer

ERASER gomme les inévitables maladresses en les recouvrant de la couleur du fond. Son efficacité ou sa précision dépendent de sa taille qui est liée à la forme du pinceau sélectionnée. La gomme étant de la couleur du fond et ses limites n'étant pas visualisées, il n'est malheureusement pas très facile de la localiser lorsqu'elle ne se trouve pas sur un élement du dessin.

— Rappelons que la touche **UNDO** supprime le dernier tracé réalisé.

— Pour gommer très rapidement de vastes surfaces on peut sélectionner un bloc vide (voir BLOCK) puis le déplacer en mode **ERASER**.

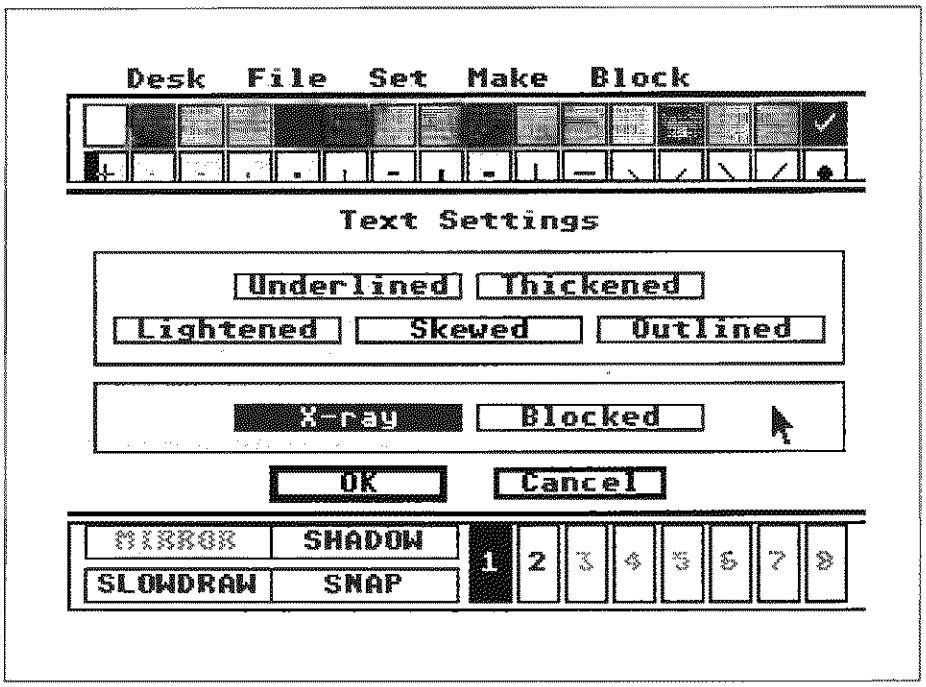
— Pour tout effacer, il faut utiliser dans le menu déroulant **FILE** l'option **ERASE PIC**.

— Si un motif couleur est sélectionné et si la case PATTERN est activée, la gomme recouvre l'objet à effacer à l'aide du motif choisi et non de la couleur de fond.

TEXT Ecrire

Les attributs de caractères dépendent de deux cases différentes sur l'écran du menu.

La petite case **TEXT** qui permet d'activer la fonction dans le dessin et qui par un double clic affiche les attributs suivants :



Les attributs de texte.

UNDERLINE : souligné

THICKENED : graissage

LIGHTENED : affaiblissement

SKEWED : italique

OUTLINE : détourage (uniquement en mode X-RAY)

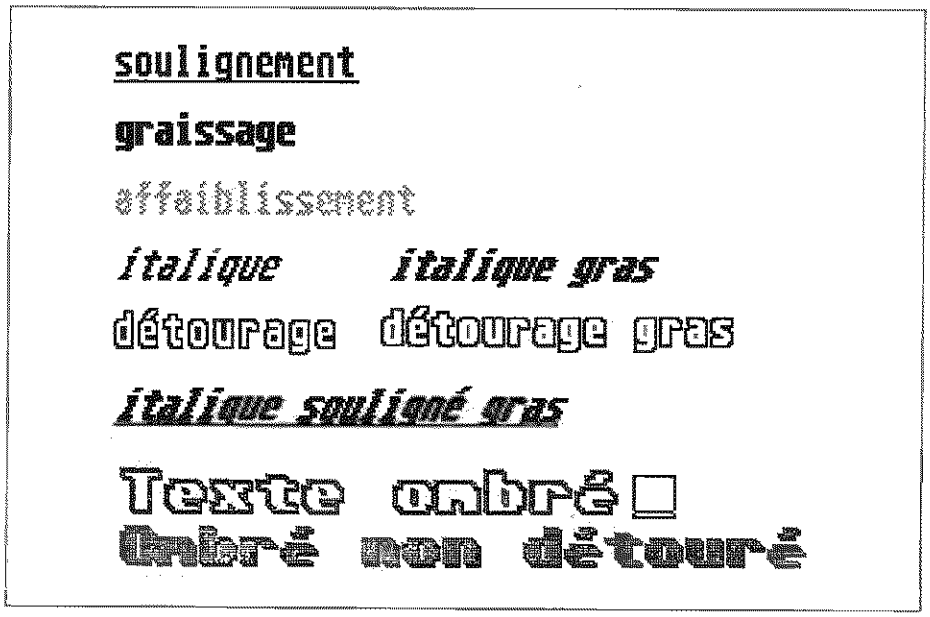
— Ces options peuvent évidemment se combiner entre elles et se voir ajouter une ombre (SHADOW) qui en accentue le relief.

X-RAY : le texte laisse voir les éléments qu'il recouvre.

BLOCKED : le texte se trouve dans une étiquette qui masque les éléments qu'il recouvre.

— La grande case située au-dessus des icônes des écrans 5 à 8 propose des tailles différentes que les flèches droite et gauche visualisent successivement (6 tailles en standard). Un double clic dans cette zone fait accéder à la liste des polices disponibles.

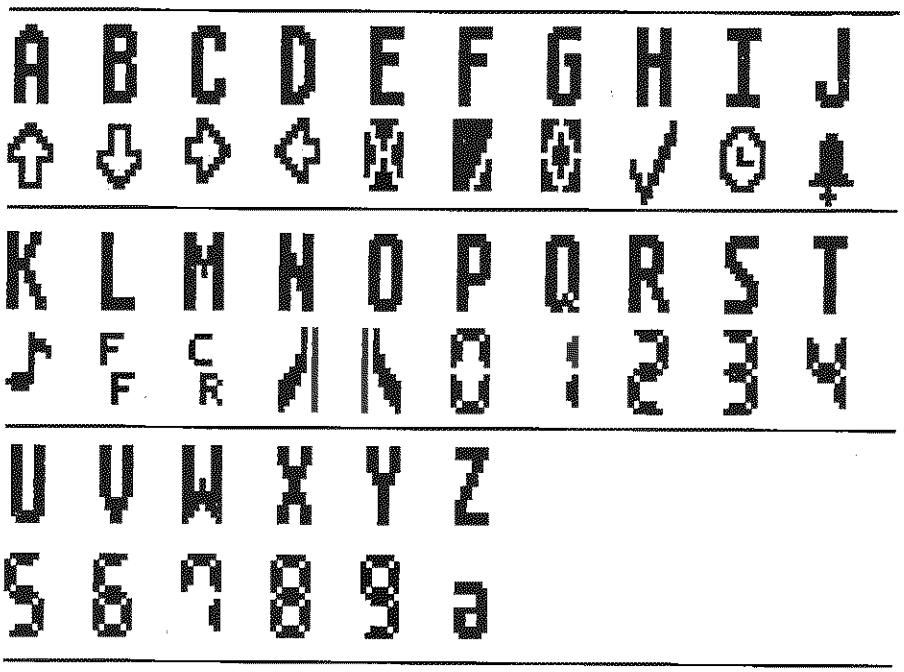
Les attributs de caractères. ▶



— Lors de la saisie de texte, la touche **BACKSPACE** permet les corrections, le déplacement de la souris autorise un positionnement exact de l'ensemble de la ligne. La touche **ENTREE** fige le texte et produit un passage à la ligne.

— En mode texte, les touches du clavier combinées à la touche **CONTROL** produisent un jeu de caractère étendu comme en témoigne notre essai avec la fonte système. La première ligne représente le caractère frappé seul, la ligne suivante le montre combiné avec **CONTROL** :

Le jeu de caractères étendu. ▶



LES OUTILS SPÉCIALISÉS

POINT Placer un point, une tache

POINT place à la position du curseur un tracé de la forme du pinceau chaque fois qu'on clique. Des petits motifs de huit points sur huit (taille maximale du pinceau) peuvent ainsi être rapidement dupliqués.

AIRBRUSH L'aérographe

AIRBRUSH projette des points de couleur à la manière d'une bombe à peinture. Il permet d'obtenir assez rapidement des fondus et certains effets de matière qu'interdisent les habituels aplats ou les trames trop régulières. Ainsi l'aérographe se prête-t-il particulièrement bien à la création de végétations, de feuillages qui peuvent faire songer à certains tableaux impressionnistes ou pointillistes.

— Un double clic sur la case **AIRBRUSH** permet de régler les paramètres de largeur de jet et de débit :

AIRBRUSH SIZE : trois tailles de buses, toutes circulaires, sont disponibles.

AIRBRUSH FLOW : le débit peut être réglé selon trois intensités.

Remarques :

L'activation de l'ombre lors de l'utilisation de l'aérographe facilite les effets de relief ou de granité. Si un motif de remplissage est activé (**PATTERN**), celui-ci apparaît progressivement au fur et à mesure que les taches se déposent.

STIPPLE La gravure au pointillé

STIPPLE propose un aérographe particulier et beaucoup plus grossier qui projette des taches de la forme du pinceau sélectionné. Les paramètres de débit et de rayon d'action peuvent être ajustés par un double clic :

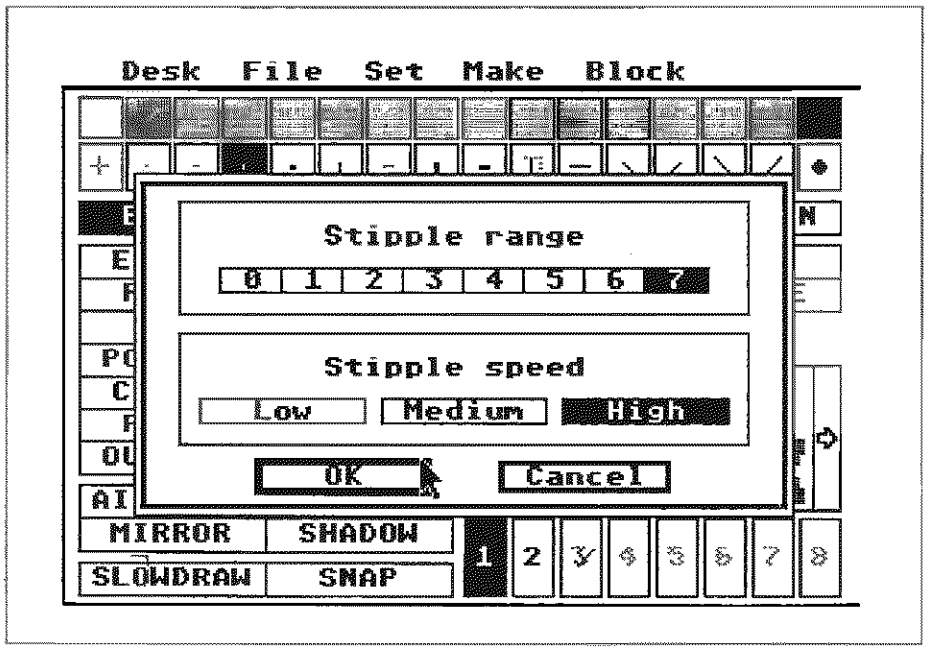
(Voir écran page suivante.)

STIPPLE RANGE définit un rayon de projection allant de 0 à 7 unités.

STIPPLE SPEED ajuste le débit.

PATTERN/SOLID Dessiner avec un motif

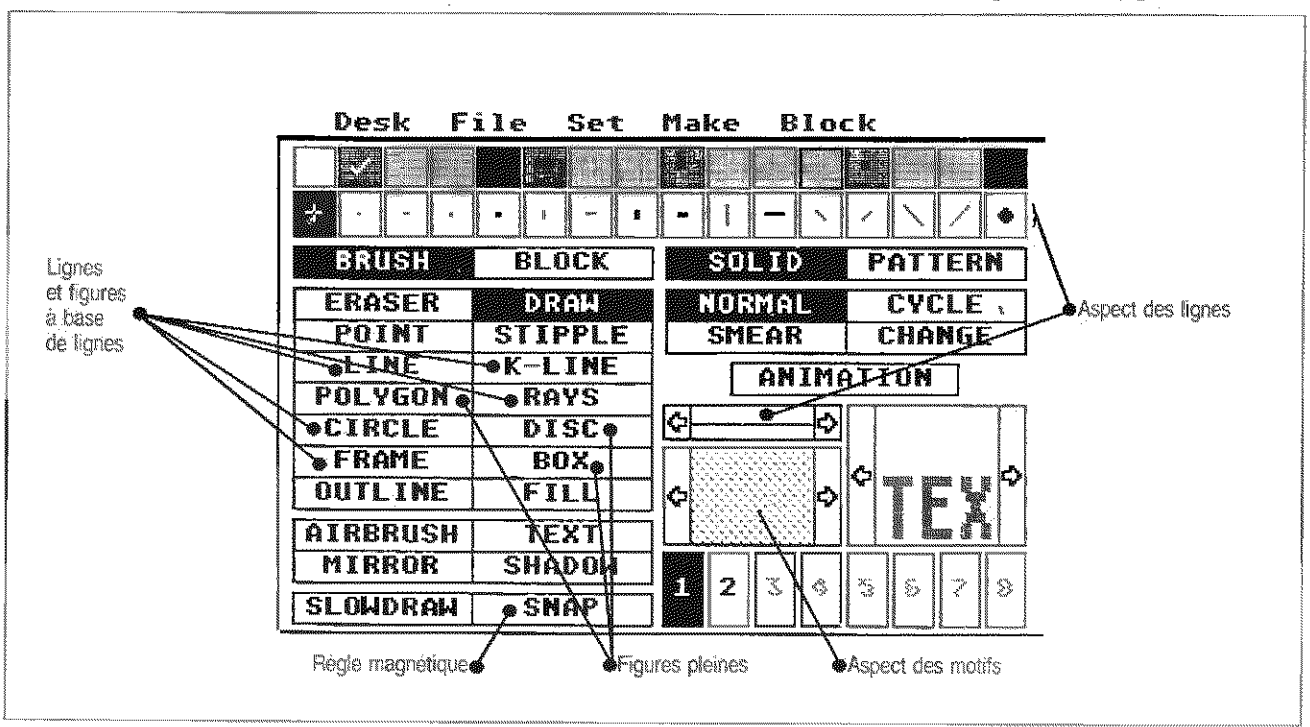
Ces deux cases fonctionnent comme une bascule pour la plupart des menus qui utilisent le pinceau et permettent de tracer des figures non remplies. **SOLID** trace la figure en traits pleins ou à l'aide du type de ligne sélectionné. **PATTERN** les trace avec la forme du motif courant, ce qui n'est pas toujours évident si le pinceau est trop fin.



Réglages de la gravure au pointillé.

LES OUTILS GÉOMÉTRIQUES

L'utilisation de la règle magnétique (voir SNAP) peut faciliter les tracés géométriques en résolvant les problèmes liés aux figures orthogonales.



Les figures géométriques.

Les figures composées de lignes

L'aspect des lignes, ou des figures composées de lignes dépend du type de ligne sélectionné dans le menu principal, de la couleur et du pinceau actifs. On a intérêt à travailler en mode **SOLID**.

LINE, K-LINE, RAYS — Les lignes

LINE trace un segment délimité par ses deux extrémités à l'aide de la souris.

K-LINE trace des segments consécutifs formant une ligne brisée. Le tracé doit être interrompu par appui sur la touche droite de la souris.

RAYS trace des rayons partant du premier point sélectionné, les autres points étant les extrémités des rayons. L'interruption du tracé se fait comme dans **K-LINE**. **RAYS** peut être utilisé pour tracer rapidement les traits de construction d'un dessin en perspective.

CIRCLE, FRAME — Les figures vides

CIRCLE trace une ellipse dont on définit le centre et un sommet du rectangle dans lequel elle s'inscrit.

Remarque :

Pour obtenir un cercle parfait il faut maintenir la touche ALTERNATE enfoncée lors du tracé, l'ellipse s'inscrit alors dans un carré.

FRAME (cadre) trace un rectangle ou un carré dont on définit deux sommets opposés.

Remarque :

Pour obtenir un cadre à bords arrondis, il faut maintenir la touche ALTERNATE enfoncée lors du tracé.

POLYGON, DISC, BOX — Les figures pleines

Ces trois figures se remplissent du motif ou de l'aplat sélectionné. **SOLID** et **PATTERN** permettent alors de masquer ou non les parties de dessins qui se trouvent recouvertes. La transparence est assurée par la couleur du fond du motif, à condition que ce dernier soit monochrome.

POLYGON permet de tracer des figures à 25 côtés maximum. Le dernier point doit se trouver à proximité immédiate du point de départ.

Remarques :

- Pour être certain de fermer le polygone exactement sur le point de départ, il suffit de taper **RETURN** après avoir placé l'avant dernier segment.

— **DISC** fonctionne comme **CIRCLE**.

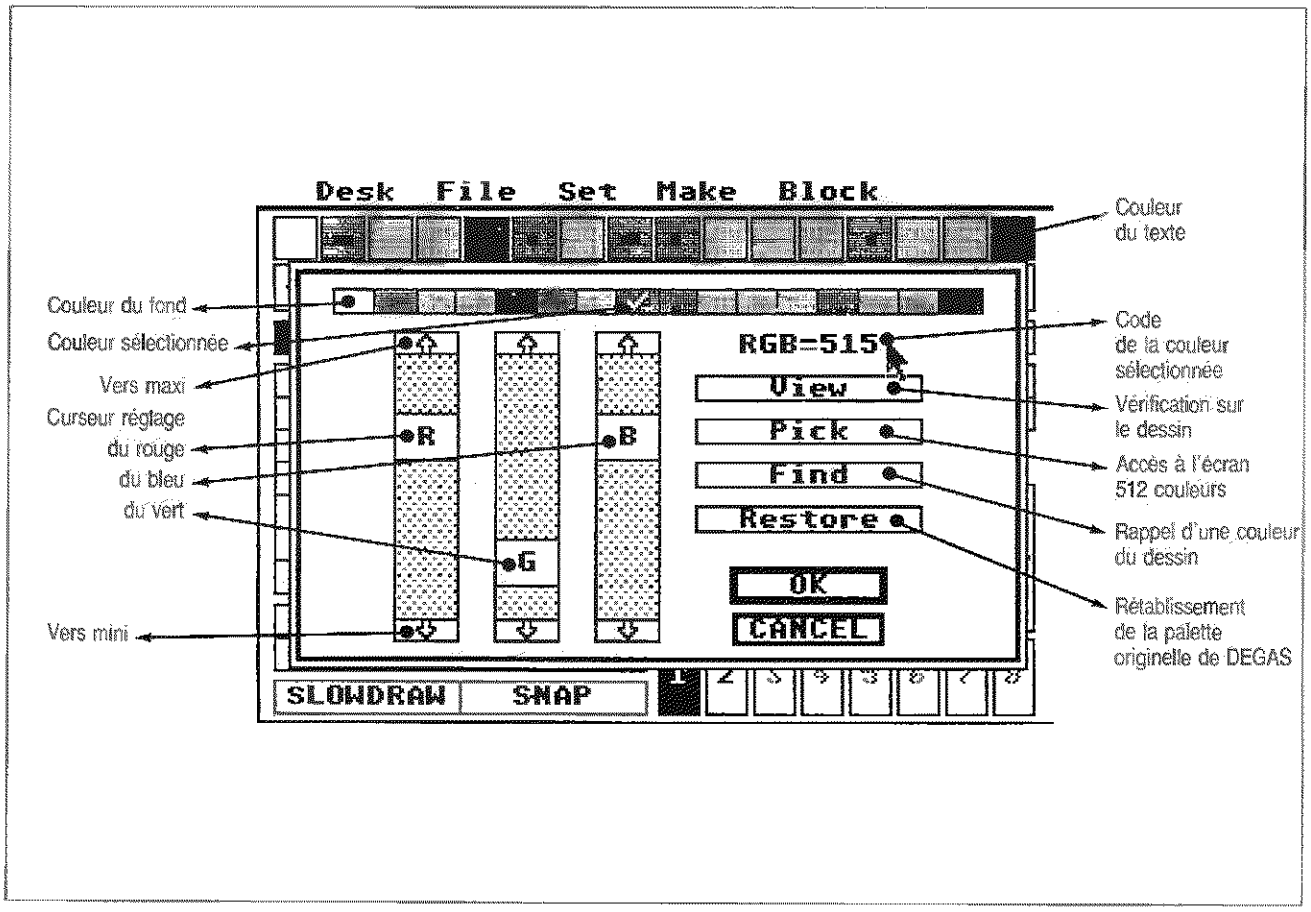
— **BOX** comme FRAME.

• Lorsqu'on trace l'une de ces trois figures en la remplissant d'un motif, on constate que sa bordure n'est pas visible, elle n'est pas délimitée. Pour obtenir une limite, il faut appuyer sur **SHIFT** au moment du dernier clic qui valide le tracé. Le maintien de la touche CONTROL lors du tracé permet d'obtenir un aplat à l'intérieur de la figure au lieu du motif courant s'il est sélectionné.

II - Les paramètres de travail

CHOIX D'UNE COULEUR

En basse résolution, seize couleurs sont disponibles sur la première ligne de menu et quatre seulement en moyenne résolution. Toutes peuvent être modifiées par appel d'une boîte de dialogue. On double-clique sur une case de couleur.



La boîte de réglage des couleurs.

Les couleurs peuvent être modifiées à l'aide des trois curseurs qui réalisent le mélange des primaires Red, Green, Blue (rouge, vert, bleu). La correspondance numérique (ici RGB = 515) permet de retrouver précisément un ton souhaité. La palette s'étend de 777 (blanc) à 000 (noir).

Toute modification de la palette provoque une modification sur les couleurs déjà utilisées dans tous les écrans contenant des dessins. On ne dispose donc dans tous les cas que de 16 couleurs pouvant être affichées simultanément.

VIEW Visualiser

Affiche immédiatement la page de dessin courante et aussi longtemps qu'on maintient le clic. On peut ainsi juger du résultat produit lors d'une modification.

PICK Choisir

Affiche un écran présentant simultanément toutes les nuances résultant des 512 combinaisons possibles effectuées à partir des trois primaires. Chaque primaire comporte huit densités possibles, on obtient donc bien $8 \times 8 \times 8 = 512$ combinaisons. On clique directement sur la nuance souhaitée. La couleur choisie apparaît dans une zone verticale à droite de l'écran où on peut alors la valider avec la souris.

FIND Chercher

Fait réapparaître la feuille de dessin. Un réticule permet de sélectionner directement une couleur faisant partie du dessin pour la ramener dans la boîte de modification.

RESTORE Annuler

Rétablit la palette d'origine de Degas.

En agissant sur la ligne de couleurs qui apparaît dans la partie supérieure de la boîte, on peut soit choisir une autre couleur à modifier, soit dupliquer une couleur en la prenant à l'aide de la souris puis en la glissant et en la déposant ailleurs.

Le logiciel peut automatiquement calculer un dégradé de couleurs. On clique d'abord sur la couleur de départ dans le bandeau supérieur de la boîte de dialogue puis on appuie sur **ALTERNATE** tout en cliquant sur la couleur d'arrivée.

Remarque :

La réussite d'un dessin est essentiellement liée au choix judicieux des couleurs qui le composent. On a intérêt à disposer de très peu de couleurs différentes (trois ou quatre) mais qui existent en de nombreux tons, permettant une fine gradation du plus clair au plus foncé. Les effets de matière, de volume ou de profondeur sont alors faciles à créer.

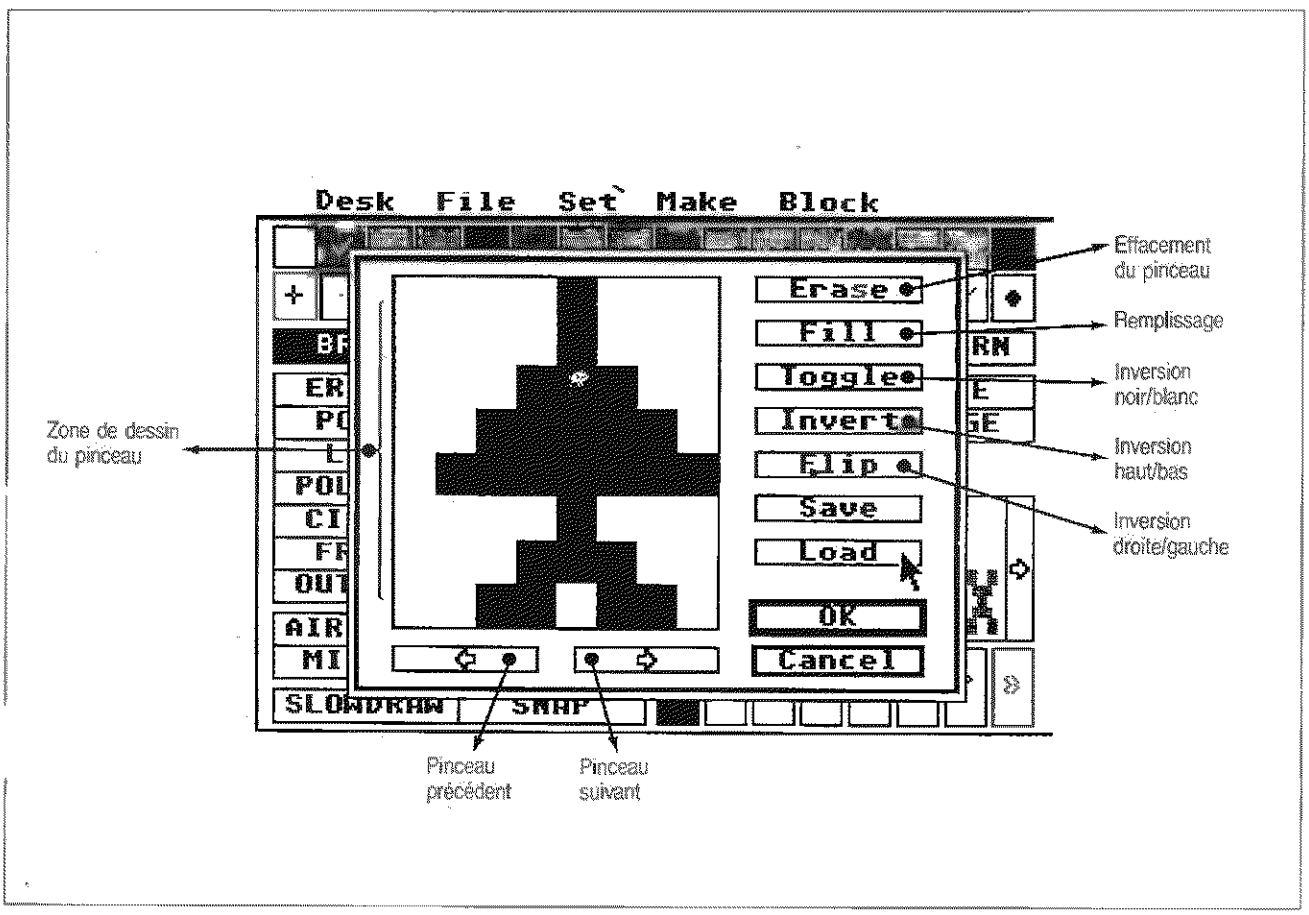
On évitera de trop modifier les deux cases extrêmes (la blanche, couleur de fond et la noire, couleur de texte) qui influent également sur les couleurs de l'écran des menus. On peut éventuellement les inverser pour travailler sur un fond noir qui magnifie naturellement tous les tons. De plus, un tour d'écran noir délimite plus efficacement un dessin qu'un tour blanc.

Raccourcis : Lorsqu'on dessine, les flèches haute et basse permettent de faire défiler la palette de couleur sans qu'il soit nécessaire de revenir au menu.

CHOIX D'UN PINCEAU BRUSH

L'aspect de tout tracé, qu'il soit fait à « main levée », ou à l'aide des outils géométriques (rectangle vide, cercle...) dépend de la forme de pinceau choisie.

La deuxième ligne de l'écran de menus présente seize formes de pinceaux différentes. La première, le réticule, possède une précision d'un point et ne peut pas être modifiée ; un double clic sur l'une des quinze autres ou l'appel du menu déroulant **MAKE** puis **MAKE BRUSH** ouvre la case de dialogue suivante :



La boîte de création des pinceaux.

ERASE efface la forme affichée.

FILL remplit entièrement la case.

TOGGLE provoque l'inversion noir/blanc.

INVERT inverse la forme suivant l'axe horizontal (haut/bas).

FLIP l'inverse suivant l'axe vertical (droite/gauche).

SAVE sauvegarde la forme de pinceau sur le disque. On choisira de préférence le répertoire **BRUSHES**.

LOAD ramène une forme précédemment sauvée. Quatre formes sont fournies sur la disquette : JET.BRU (avion), MAN.BRU (personnage), THREEDOTS.BRU (trois points), TWODOTS.BRU (deux points).

Les deux flèches situées sous la forme permettent de passer à la forme précédente ou à la forme suivante de la ligne sans sortir de la boîte.

Remarque :

Les formes sophistiquées, telles que celles du personnage ou de l'avion sont surtout utiles dans le menu **POINT**. Pour dessiner, on choisira des formes de pinceau plus classiques.

Raccourcis : Lorsqu'on dessine, les flèches droite et gauche font défiler les formes de pinceaux sans qu'il soit nécessaire de revenir au menu. La touche « **b** » fonctionne comme une bascule qui fait passer du mode **BRUSH** au mode **BLOCK**.

UN PINCEAU SUR MESURE BLOCK

Le mode **BLOCK** (voir ci-dessous), s'il est sélectionné, permet de dessiner à l'aide du bloc courant.

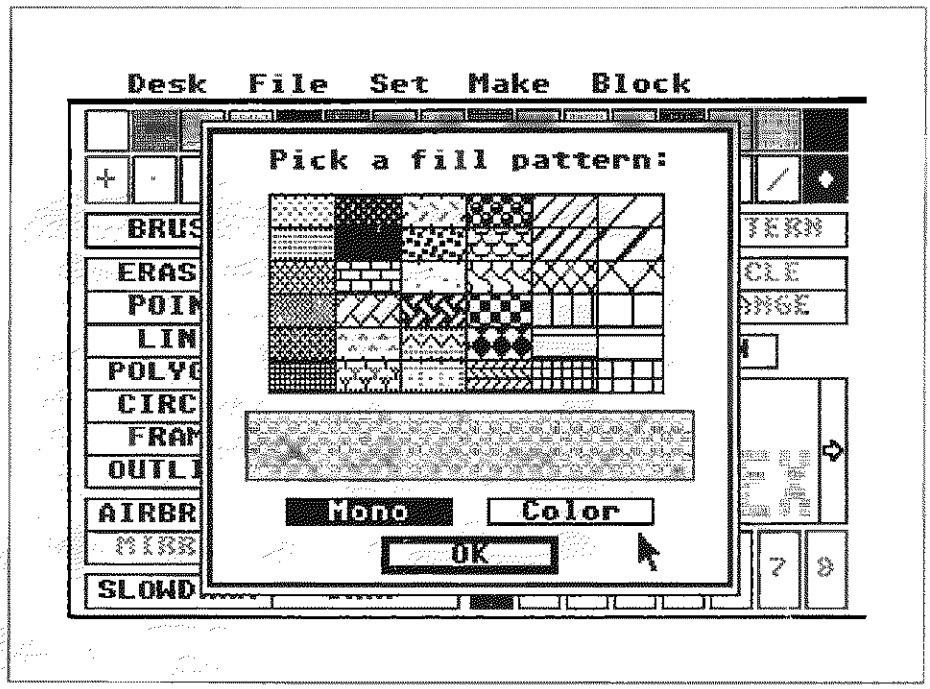
Le bloc suit les mouvements de la souris et il s'utilise comme un pinceau en mode dessin, en mode point et même lors du tracé de figures géométriques.

CHOIX D'UN MOTIF DE REMPLISSAGE

Le cadre, situé au-dessus des icônes représentant les écrans 1 à 4 du menu principal, contient le motif de remplissage courant qui apparaît dans la couleur choisie. Ce motif peut être sélectionné soit en cliquant sur les flèches droite et gauche qui bordent le cadre, on fait alors défiler 36 trames différentes, soit en double-cliquant à l'intérieur du cadre :

(Voir écran page suivante.)

La sélection s'effectue directement avec la souris et une large zone rectangulaire donne une idée de l'effet produit.



COLOR présente 36 autres motifs colorés.

Le menu **MAKE** suivi de **MAKE FILL** permet de modifier soit les trames monochromes, soit les trames couleurs et propose la boîte de dialogue suivante (voir écran page suivante) :

On retrouve les mêmes possibilités que pour la construction d'un pinceau. Deux options supplémentaires sont disponibles en mode monochrome et quatre en mode couleur : les quatre flèches font défiler le motif à l'intérieur du cadre de modification.

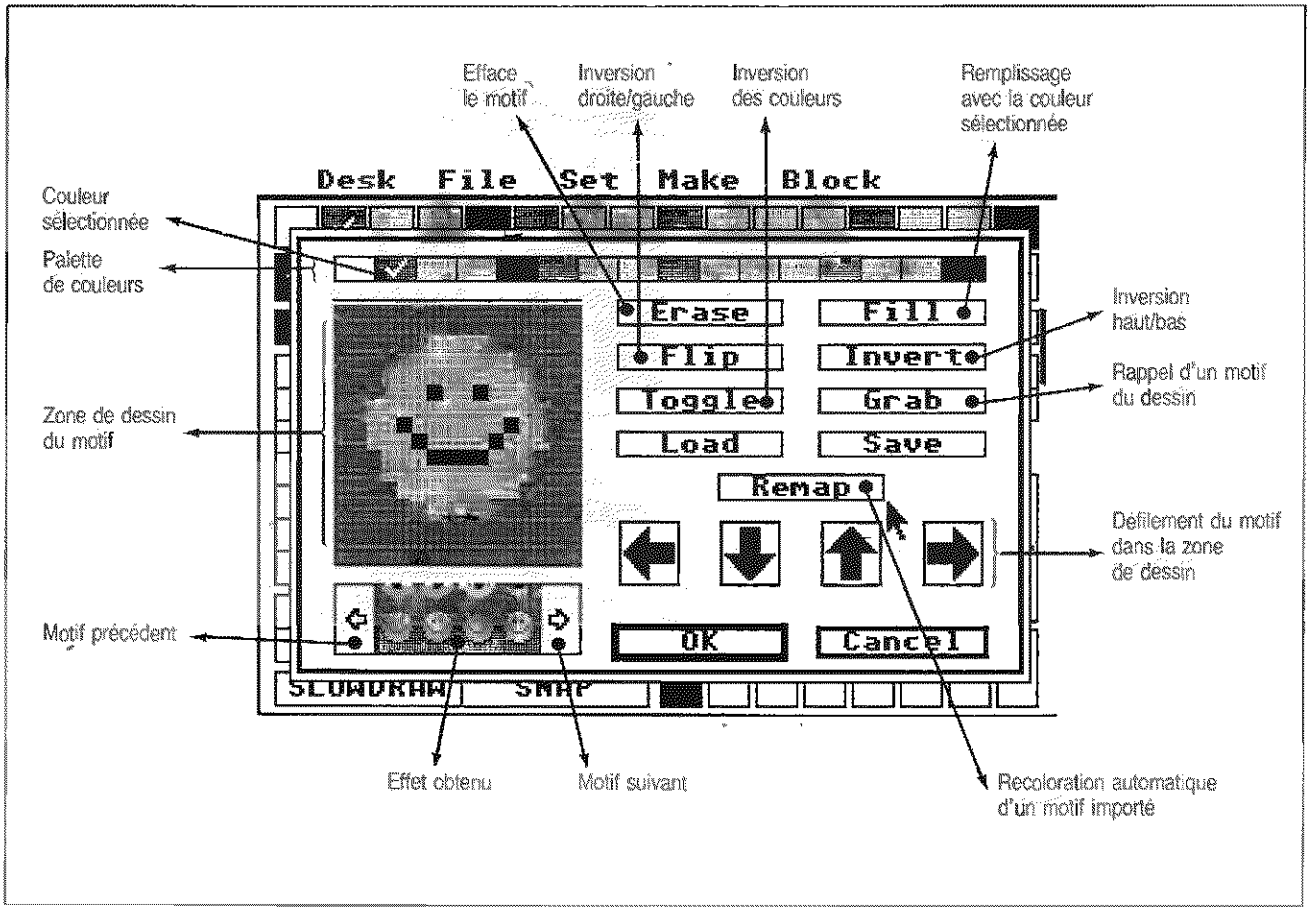
L'option **GRAB** permet d'aller chercher une partie du dessin courant pour le transformer en trame. La palette affichée offre le choix des 16 couleurs disponibles et l'option **REMAP** ajuste les couleurs d'une trame importée, au plus près de la trame existante.

TOGGLE inverse les couleurs en prenant leur symétrique par rapport à l'axe central de la palette.

Les motifs sont stockés dans le répertoire **FILLS** et quatre exemples sont proposés **CRACKER.FI1**, **SMILEY.FI1** (couleur basse résolution), **SMILES.FI2** (couleur moyenne résolution) et **DIAGONAL.FIL** (monochrome).

CHOIX D'UN STYLE DE LIGNE

Ce cadre situé au-dessus de celui des trames fonctionne sur le même principe que ce dernier. Sept styles de lignes sont accessibles (continu, pointillé, tiret, mixtes...).



La boîte de création des motifs.

Les lignes sont modifiées à partir du menu **MAKE** suivi de **MAKE LINE**. Dans la boîte de dialogue qui apparaît alors, on peut faire défiler les sept types de lignes disponibles à l'aide des flèches droite et gauche et les modifier en ajoutant ou en enlevant des points.

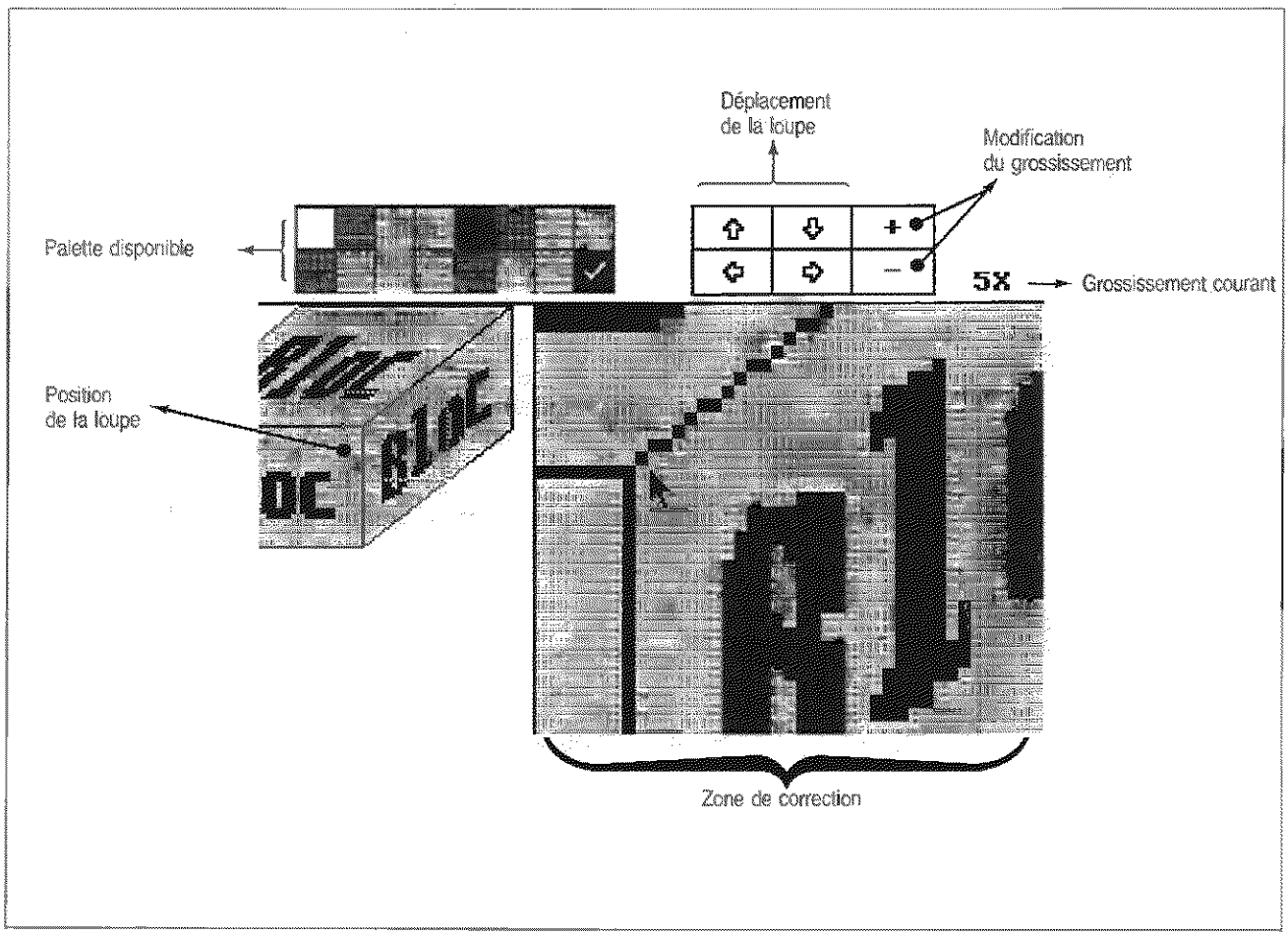
III - Les facilitateurs

LA LOUPE

La loupe est l'outil indispensable pour produire des dessins de qualité. Elle permet notamment de reprendre au point près tous les traits qui ont pu subir des déformations lors de modification d'échelle, rotations, déplacements, copies.

Elle est accessible depuis le mode dessin par appui sur une touche de fonction (**F1** à **F10**). Le retour au dessin normal se fait en cliquant la touche droite de la souris.

Une loupe de plus ou moins grande taille se déplace sur le dessin. Un clic affiche un écran ayant l'aspect suivant :



L'écran loupe.

La zone droite permet des corrections en agissant sur chaque point individuellement. Les couleurs sont sélectionnées dans le bandeau supérieur. Les flèches permettent les déplacements de la loupe. Les cases **plus** et **moins** font varier le grossissement dans un rapport de 3 à 12.

LE RALENTISSEUR. SLOW DRAW

Il démultiplie les mouvements de la souris dans un rapport de 2 à 8 après sélection par un double clic.

LA RÈGLE MAGNÉTIQUE. SNAP

Une grille invisible est activée par cette case. Le curseur de dessin ne peut désormais plus se positionner en dehors des intersections de cette grille et il semble se déplacer par sauts à l'écran. La grille se révèle très utile pour les tracés orthogonaux, les objets géométriques en volume ou pour tracer des cercles concentriques. L'espacement peut être réglé dans un rapport de 2 à 16 points (par pas de 2 points) entre chaque maille, ceci en fonction de la précision et de la densité du travail entrepris.

COPIES, DÉPLACEMENTS ET DÉFORMATIONS. BLOCK

Dans le menu principal, l'ouverture de la case **BLOCK** propose :

— X-RAY donne une couleur de fond du bloc transparente qui laisse voir les motifs recouverts. C'est le mode qu'il faut choisir pour effectuer des copies ou des déplacements.

— **BLOCKED** masque les motifs recouverts. Ce peut être utile pour définir un bloc vide de grande taille qui permettra de gommer rapidement un dessin.

— **FREE MOVE** laisse sortir le bloc de l'écran.

— **EDGE STOP** l'enferme dans les limites de l'écran.

Remarque :

Un bloc n'est supprimé que si on en sélectionne un autre. Il est désactivé si on choisit le mode BRUSH depuis le menu ou si on appuie sur la touche « **b** » depuis le dessin. Tout élément, quelle que soit sa taille, peut être dupliqué ou déplacé.

Sélection du bloc

— Le bloc peut être sélectionné de deux manières. Soit on inclut les éléments du dessin dans un rectangle et on est contraint de saisir alors les éléments voisins, soit on crée un polygone, épousant au plus près la forme de l'élément à déplacer (le polygone est néanmoins limité à 25 côtés).

• *Sélection dans un rectangle*

Un appui sur **ESC** fait apparaître un réticule. Celui-ci doit être positionné avec la souris dans l'angle supérieur gauche du rectangle dans lequel s'inscrira l'élément à copier ou à déplacer. Un clic maintenu suivi d'un glissement fait se matérialiser le rectangle de sélection. Le relâchement du clic valide la sélection.

• *Sélection dans un polygone*

On appuie sur la touche **SHIFT** + **ESC** lors de l'appel du mode création de bloc. Le réticule n'apparaît alors pas et on utilise la souris pour entourer de segments l'élément à saisir dans le bloc. L'appui sur **RETURN** ferme le polygone et valide la sélection.

Copie de bloc

La validation du bloc fait apparaître le bloc superposé au dessin et centré par rapport au curseur. On peut alors le déplacer et le recopier à un autre endroit ou sur un autre écran en cliquant une fois. On peut également dessiner avec le bloc en déplaçant la souris tout en maintenant cliqué. Un appui sur la touche « **b** » fait revenir au mode dessin.

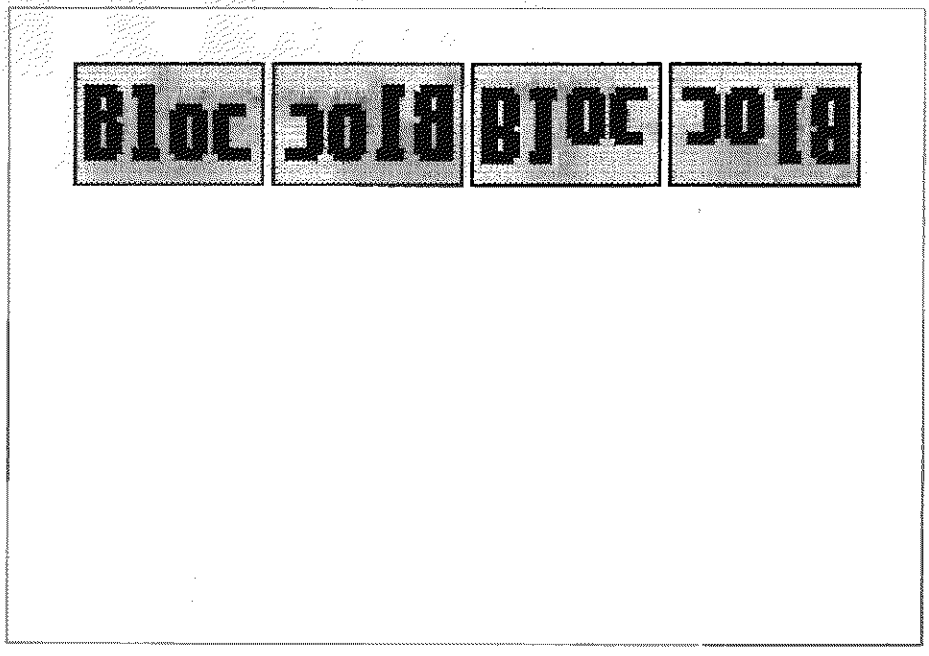
Déplacements de bloc

Si on souhaite déplacer le bloc au lieu de le copier, on doit appuyer sur **ALTERNATE** au moment de valider (relâchement du clic ou appui sur ENTREE).

Opérations sur les blocs

- *Inversions*

Lorsqu'un bloc est sélectionné, les flèches du clavier permettent les inversions dans les quatre directions.



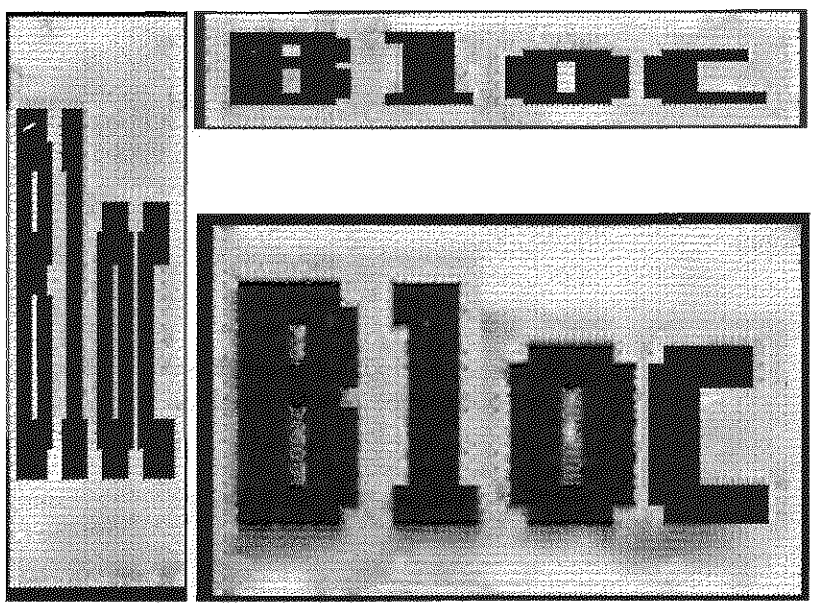
Les autres opérations sont disponibles depuis le menu déroulant **BLOCK**. A chaque fois, le bloc sélectionné se retrouve sur un écran vide où il peut être modifié.

L'appui sur la touche droite de la souris le restitue avec ses modifications sur l'écran de dessin.

- *Etirement (ou réduction)* : STRETCH

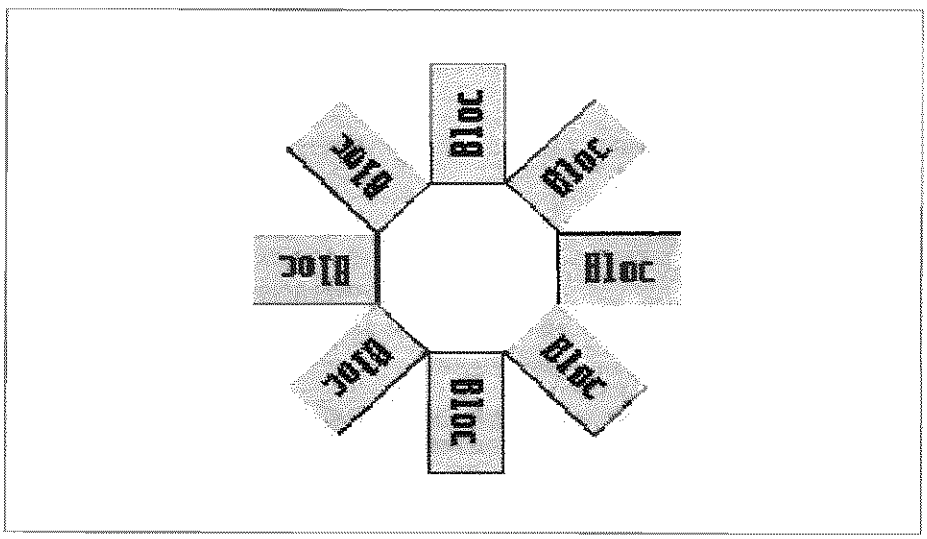
Le bloc apparaît calé dans le coin supérieur gauche de l'écran. On peut étirer son angle opposé à l'aide de la souris.

La touche **SHIFT** activée en même temps que la souris ne permet qu'un étirement horizontal, la touche **ALTERNATE** un étirement vertical et la touche **CONTROL** une homothétie.



- *Rotation* : ROTATE

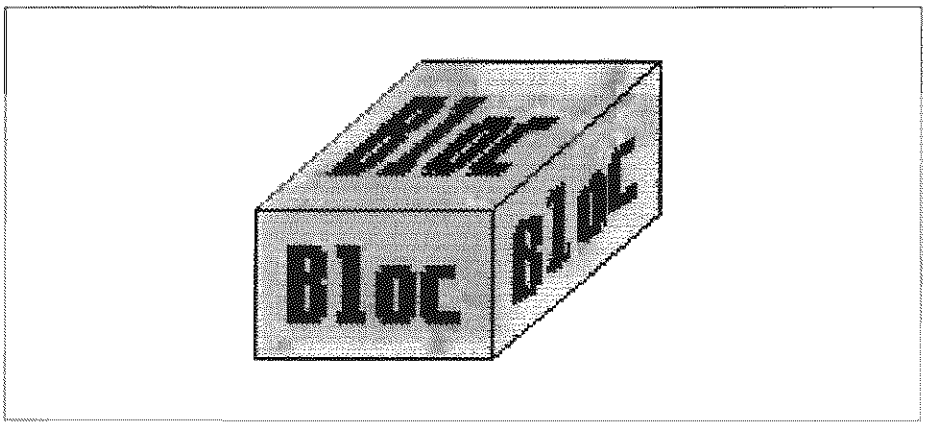
Le bloc se trouve au centre de l'écran, tout en maintenant cliqué un des angles du rectangle de sélection on déplace la souris et la valeur de l'angle de rotation apparaît en haut d'écran.





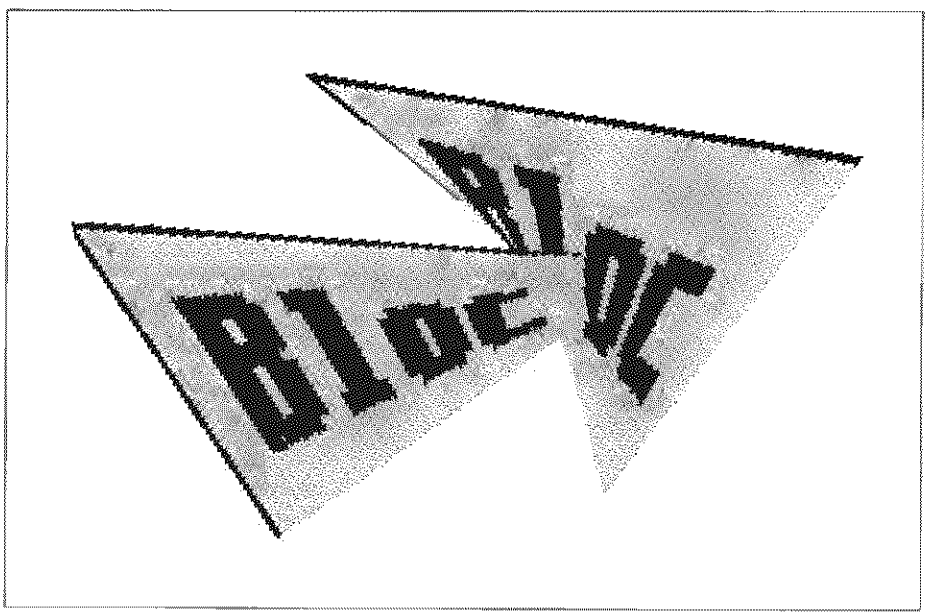
- *Inclinaisons horizontales et verticales :* H-SKREW et V-SKREW

Le rectangle de sélection se trouve déformé en un losange plus ou moins aplati. Cette fonction permet d'obtenir des effets de perspective.



- *Distorsions :* DISTORT

Cette puissante fonction permet de donner n'importe quelle forme au rectangle de sélection. Ses quatre sommets peuvent être déplacés en tout point de l'écran et même des croisements d'image peuvent être réalisés. L'image ne s'inscrit dans la nouvelle forme que lorsque la touche **ALTERNATE** est pressée en même temps que le bouton gauche de la souris. Le calcul de l'image n'est pas immédiat et dépend de la taille du bloc ainsi que de la gravité de la torture infligée.



• *Archivage :* **FORMAT**, **LOAD**, **SAVE**, **REMAP**

Cinq exemples de blocs sont fournis avec DEGAS dans le répertoire **BLOCKS**. Il s'agit de **ASTERISK.BL3** (haute résolution), **REDDOT.BL2** (moyenne résolution), **SPACESHIP.BL1** (basse résolution), **TREE.BL1**, **DISK.ICN** (toutes résolutions).

Le format icône (ICN) ou bloc (BL1 à BL3), dépend du menu **FORMAT** qu'il faut donc appeler en premier : un bloc ICN perd ses couleurs lors de la sauvegarde mais peut être rechargé dans n'importe quelle résolution. Un bloc BL1 ou BL2 conserve ses couleurs mais ne peut être importé que dans un écran de même résolution.

Lors du chargement par **LOAD** d'un fichier BL1, une boîte de dialogue propose que le bloc prenne les couleurs les plus proches de la palette active (Re-color YES) ou qu'il prenne les couleurs situées à la même place dans la palette (Re-color NO). Si on a répondu **NON** lors du chargement, **REMAP** permet de modifier par la suite les couleurs du dernier bloc importé.

Lors de l'importation, un fichier .ICN prend la couleur la plus à droite dans la palette courante.

IV - Les fonctions spéciales

SYMÉTRIES

Le noircissement de la case **MIRROR** active le mode miroir dont les effets peuvent être modifiés par un double clic qui propose le menu suivant :

— horizontal : contrairement à son nom, partage l'écran verticalement. Ce qui est dessiné dans la partie droite est tracé inversé dans la partie gauche ;

— vertical : ici, l'écran est partagé horizontalement (reflets dans l'eau) ;

— diagonal : le point de symétrie correspond au centre de l'écran (inversion des quadrants) ;

— all three : tous les miroirs sont activés et quatre crayons sont simultanément manipulés.

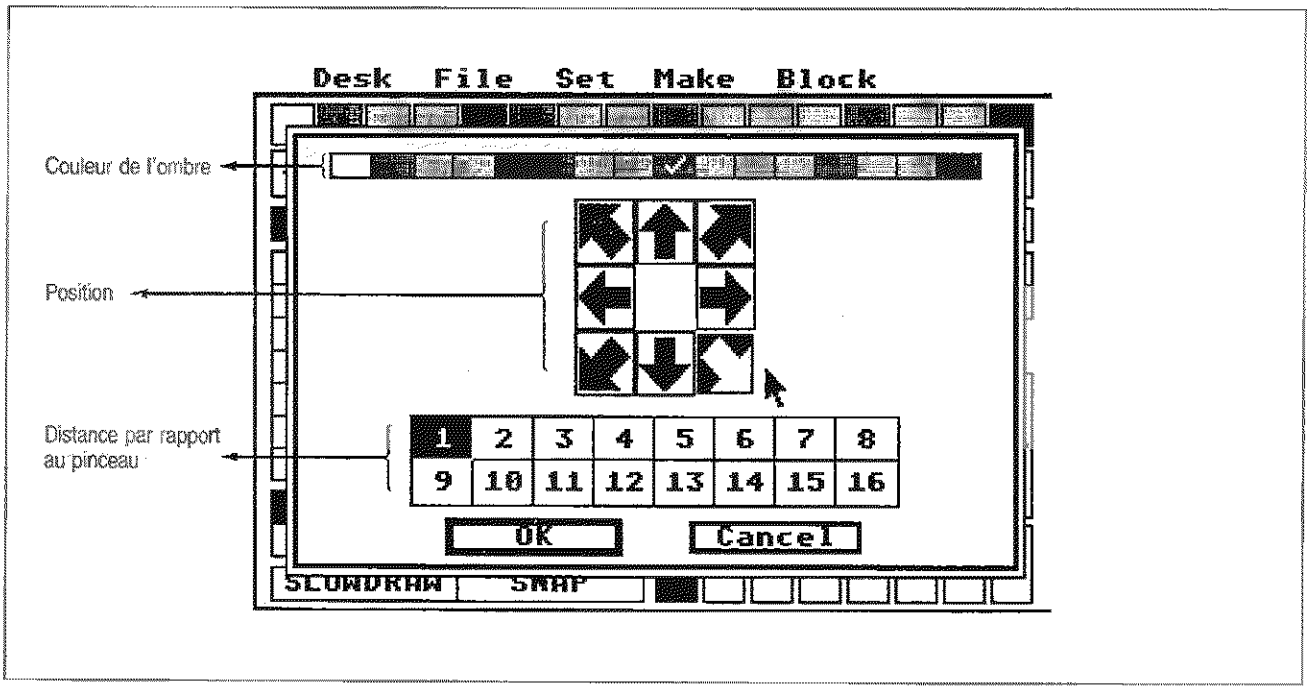
Remarque :

L'option MIROIR ne fonctionne pas avec tous les menus, c'est par exemple le cas avec le texte, le remplissage. Le mode miroir est un excellent moyen pour retrouver le centre exact de l'écran.

OMBRES (SHADOWS)

Tout tracé peut être doublé d'une ombre située à une distance et à une position variables.

L'ouverture de la case **SHADOW** par un double clic produit la boîte de dialogue suivante :



La boîte de dialogue SHADOW.

Comme dans la réalité, dans DEGAS l'ombre n'est pas forcément noire. Un choix de couleurs en haut de boîte permet toutes les fantaisies.

Les huit flèches déterminent la position de l'ombre par rapport au tracé. Une seule est active à la fois.

La distance de l'ombre par rapport au tracé varie entre 1 et 16 points.

LA BORDURE. OUTLINE

Cette fonction intéressante met en valeur certains éléments du dessin. Elle trace une bordure d'un point dans la couleur courante autour de toutes les figures que l'on sélectionne par la suite, y compris chacune des lettres d'un texte. On peut répéter l'opération plusieurs fois de suite pour accroître la taille de la bordure, il faut auparavant avoir sélectionné **ALL** dans le menu **OUTLINE**. Certains détails s'estompent alors progressivement, surtout si on a sélectionné **ROUND** dans le menu **OUTLINE**, et on obtient un dessin de style mou.

Un double clic sur **OUTLINE** propose :

— **INITIAL/ALL** : **INITIAL** ne trace une bordure qu'autour de la couleur que l'on sélectionnera par la suite. **ALL** considère que toutes les figures de couleurs différentes qui se touchent ne forment qu'un seul élément, **OUTLINE** ne trace alors que la limite séparant la figure de la couleur du fond, les autres couleurs ne sont pas séparées par la bordure.

— **TOUCHING/ADJACENT** détermine l'ordre de recherche des figures à encadrer. **TOUCHING** les recherche horizontalement et verticalement. **ADJACENT** ajoute une recherche diagonale.

— **ROUND/SQUARE** définit la forme des coins des figures.

Le remplissage. FILL

La zone à remplir doit être parfaitement délimitée. Le moindre trou laisserait la couleur s'échapper et remplir une figure voisine ou même la totalité de l'écran. Le remplissage s'effectue avec le motif sélectionné. Un appui sur **CONTROL** remplace le motif par un aplat de la couleur courante.

En cas de remplissage intempestif, la touche **UNDO** rétablit l'écran originel.

Le fondu. SMEAR

Cette option permet de mélanger les couleurs de deux figures le long de leur zone de jonction. On a intérêt à choisir un pinceau relativement gros et sa couleur n'a aucune importance pour cette opération. Lorsqu'on déplace le pinceau sur la limite de deux zones colorées, les points de couleurs se répartissent aléatoirement en passant d'une zone à l'autre. La transition n'est plus franche et on obtient donc l'équivalent d'un fondu.

Le changement de couleur. CHANGE

La modification des couleurs peut s'effectuer depuis la palette, mais alors, toutes les zones de même couleur se voient concernées simultanément par ce changement.

L'option **CHANGE** permet de ne modifier que certaines zones en les repassant à l'aide du pinceau. Seule la couleur choisie se trouve modifiée et les couleurs adjacentes ne varient pas, même si le pinceau les recouvre accidentellement.

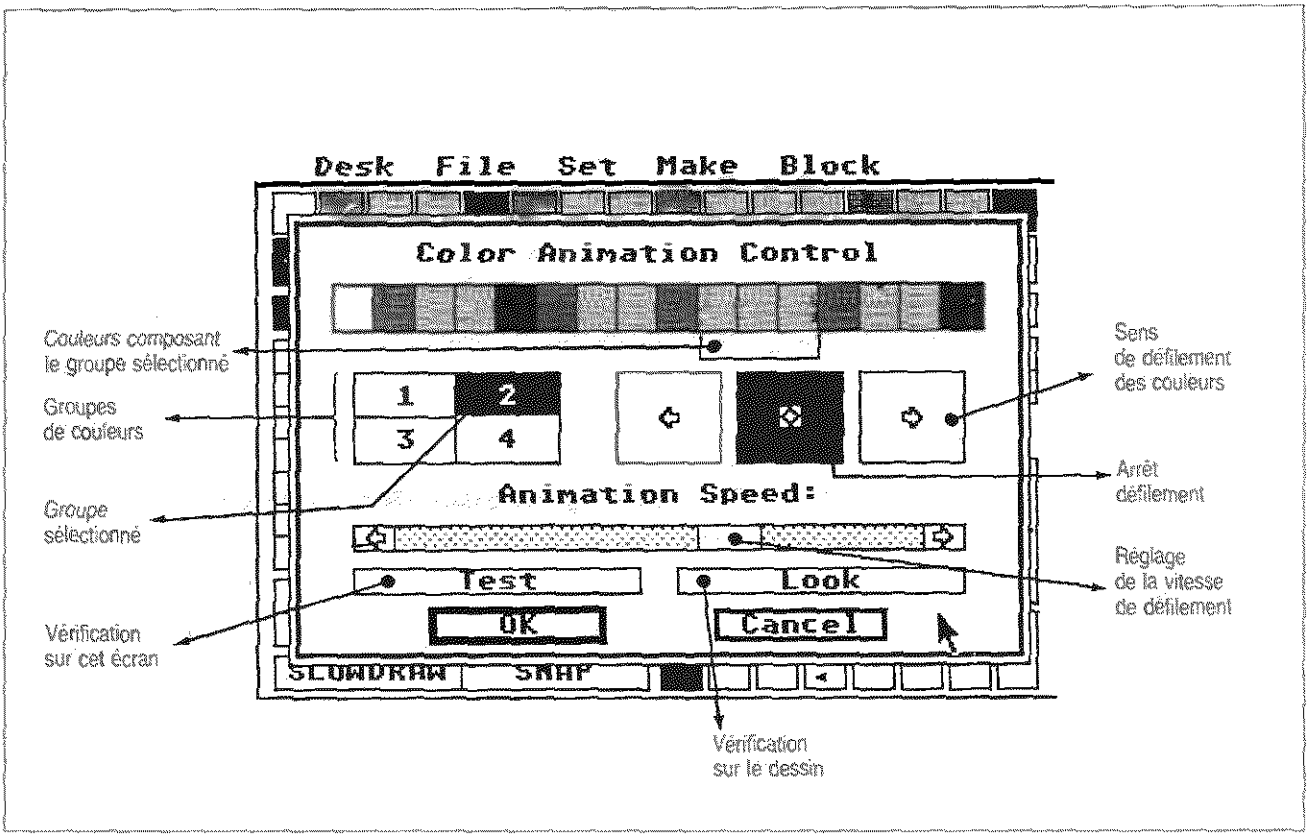
Pour cela il faut d'abord double-cliquer sur la case **CHANGE**, choisir la couleur à remplacer dans la palette qui s'affiche dans la boîte de dialogue, confirmer ce choix, choisir dans la palette du menu la couleur de remplacement, revenir au dessin et repasser sur les zones à transformer à l'aide d'un pinceau de taille appropriée.

ANIMATION

L'animation dans DEGAS se limite à faire défiler plus ou moins rapidement certaines couleurs choisies par l'utilisateur mais obligatoirement jointives dans la palette. Si le dessin est réalisé pour tirer parti de cette possibilité, des effets intéressants peuvent être obtenus, comme en témoigne le fichier d'exemple MAGICMTN.PC1 du répertoire PICTURES. Ce dessin, réalisé par Tom Hudson lui-même, présente, entre autres, un océan qui se met en mouvement si on active la case ANIMATION.

On peut, dans le même esprit, créer des effets de scintillement sur la mer, des mouvements de feuillages, animer des cascades, donner l'illusion qu'un objet tourne sur lui-même.

Un double-clic sur cette case propose la boîte suivante :



La boîte d'animation.

Quatre groupes peuvent être activés tour à tour pour permettre de définir les couleurs les composant, l'ordre d'apparition de ces dernières ainsi que leur vitesse.

Sur notre recopie d'écran, le deuxième groupe est sélectionné (case 1, 2, 3, 4). La sélection des couleurs s'effectue en cliquant-glissant de la première à la dernière du groupe (ici trois couleurs sont sélectionnées). On constate donc qu'il est impératif que les couleurs à sélectionner soient voisines.

Les deux flèches droite et gauche définissent le sens dans lequel les couleurs apparaîtront, pour le moment les couleurs ne défilent pas dans ce groupe car c'est la case « arrêt défilement » (située entre les flèches) qui est activée.

La vitesse d'animation dépend de la position du curseur de réglage. Cette dernière peut être contrôlée par la touche **TEST**. La touche **LOOK**, tant qu'elle est maintenue, permet de juger de l'effet produit sur le dessin.

Remarque :

Si quatre groupes sont définis avec un sens de défilement, ils se retrouvent actifs simultanément lors du passage au dessin. L'effet sera plus saisissant si la vitesse de défilement est différente pour chacun d'eux.

On évitera d'incorporer la couleur de fond dans un cycle d'animation, à moins de la faire varier très lentement ou de souhaiter produire des effets psychédéliques.

PINCEAU ANIMÉ. CYCLE

Cette fonction permet de dessiner avec un pinceau aux couleurs animées à condition qu'**ANIMATION** soit activée. Si la couleur choisie fait partie d'un groupe d'animation possédant un sens de défilement, le pinceau passe simultanément par toutes les couleurs de ce groupe pendant le tracé. La vitesse de tracé à une influence primordiale sur l'aspect de l'animation ; si on le déplace lentement avec un pinceau épais, on obtient une sorte de chenille qui semble avancer à l'écran. Le tracé reste ensuite animé à l'écran. Si la couleur fait partie de plusieurs groupes, c'est celui de plus faible numéro qui est choisi.

V - Archivage et impression des images. Menu déroulant FILE

DEGAS offre la possibilité de charger des images créées dans des résolutions différentes, la procédure de calcul permettant la modification de l'image est tout à fait transparente à l'utilisateur. Il autorise également les importations à partir d'autres logiciels.

SET DRIVE permet de choisir l'unité de disque courante.

PIC TYPE propose le menu suivant (voir écran ci-contre) :

Les transferts et décompressions sont automatiques.

COMPRESSION réduit la taille des fichiers sur le disque mais augmente leur temps de chargement. Attention, un fichier compressé risque de ne pas être reconnu par une autre application que DEGAS. La compression n'est active qu'au moment de la sauvegarde ; au chargement DEGAS effectue automatiquement la décompression d'une image si c'est nécessaire.

LOAD PIC charge une image et sa palette associée uniquement dans l'écran numéro 1 (vérifiez **PIC TYPE** si le catalogue est vide).

LOAD COLORS charge la palette de couleurs associée à une image du disque mais ne modifie pas le dessin courant si ce n'est en lui affectant la nou-



velle palette. Les palettes ne peuvent pas être sauvegardées seules, contrairement aux formes de pinceaux par exemple, mais toujours avec une image.

LOAD IMAGE charge une image sans sa palette de couleurs, une boîte de dialogue propose que l'image prenne les couleurs les plus proches de la palette active (Re-color YES) ou qu'elle prenne les couleurs situées à la même place dans la palette (Re-color NO). Si on a répondu NON lors du chargement, **REMAP** permet de modifier par la suite les couleurs de l'image importée. **LOAD IMAGE** est la seule option de chargement possible si on charge une image dans une autre page écran que la page 1 (**LOAD PIC** est alors inactif dans le menu).

SAVE PIC sauve l'image avec sa palette de couleurs.

DELETE PIC efface un fichier image du disque.

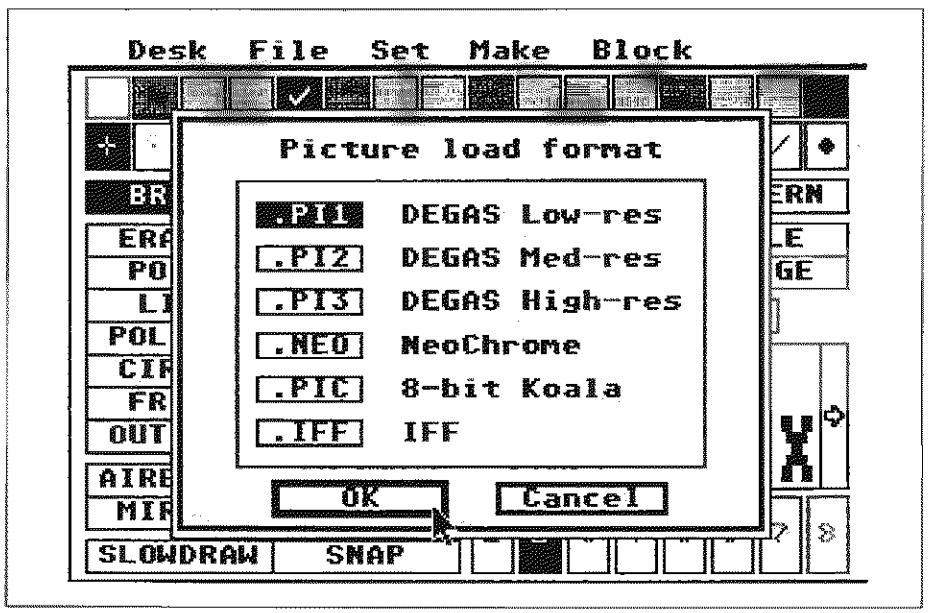
ERASE PIC efface l'écran courant.

REMAP PIC Cette fonction très sophistiquée change les couleurs d'une image chargée par **LOAD IMAGE** ou modifiée par **LOAD COLORS** en cherchant dans la palette courante les couleurs s'approchant au plus près de celles de la palette originelle.

PRINT TYPE offre un choix de gestionnaires d'imprimantes.

PRINT PIC imprime l'image de l'écran courant. Les couleurs sont remplacées par des nuances de gris. Il est nécessaire de modifier la palette pour obtenir des tirages au contraste satisfaisant (augmenter au maximum les écarts dans la palette) ;

QUIT ramène à GEM.



Les formats de données reconnus.

ANNEXES

LES EXTENSIONS DE FICHIERS

Fichiers images

Les exemples fournis se trouvent dans le répertoire PICTURES. Ils sont gérés par le menu **FILE**.

PI1 image avec palette 16 couleurs basse résolution.

PI2 image avec palette 4 couleurs moyenne résolution.

PI3 image haute résolution.

PC1 image avec palette 16 couleurs basse résolution compressée.

PC2 image avec palette 4 couleurs moyenne résolution compressée.

PC3 image haute résolution compressée.

Fichiers blocs

Les exemples fournis se trouvent dans le répertoire BLOCKS. Ils sont gérés par le menu **BLOCK**.

BL1 bloc avec palette 16 couleurs basse résolution.

BL2 bloc avec palette 4 couleurs moyenne résolution.

BL3 bloc haute résolution.

ICN bloc monochrome icône multi-résolutions.

Fichiers pinceaux

Ils sont gérés par le menu **MAKE/BRUSH** ou en double-cliquant sur une forme de pinceau du menu principal.

Les exemples fournis se trouvent dans le répertoire BRUSHES :

BRU fichier contenant une forme de pinceau.

Fichiers de remplissage

Les exemples fournis se trouvent dans le répertoire FILLS.

Ils sont gérés par le menu **MAKE/FILL**.



FI1 remplissage avec palette 16 couleurs basse résolution.

FI2 remplissage avec palette 4 couleurs moyenne résolution.

FIL remplissage monochrome multi-résolutions.

Utilisation des touches en mode dessin.

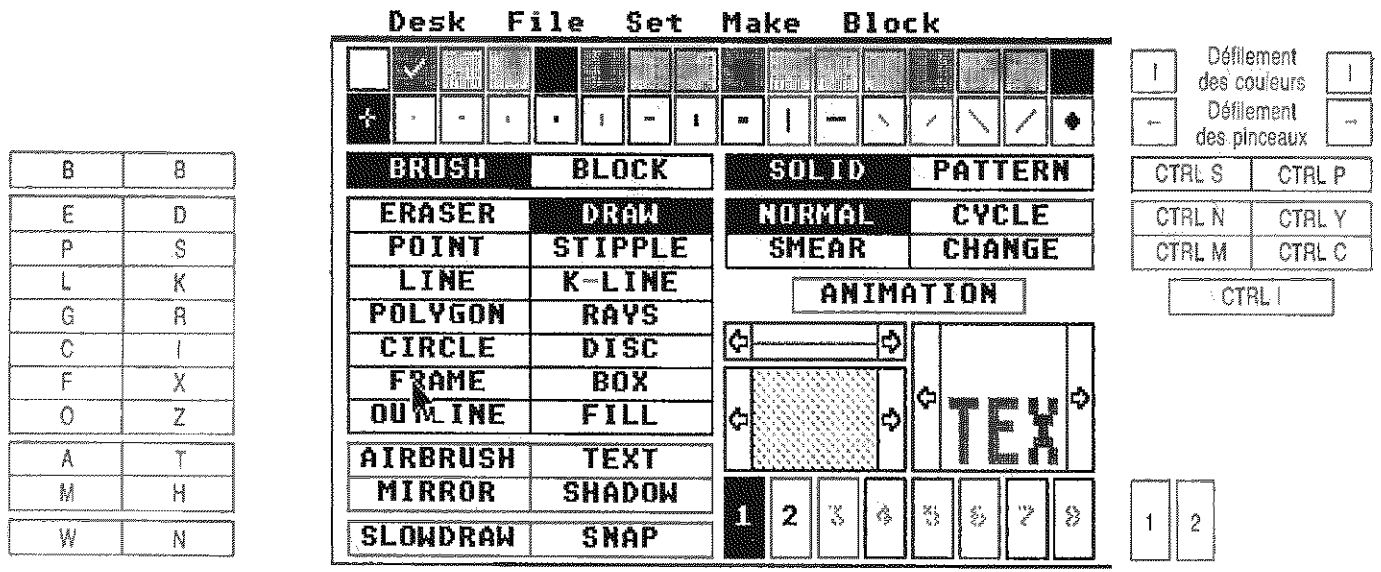
LES RACCOURCIS CLAVIER

A l'intérieur de certains menus, les touches **CONTROL**, **SHIFT** ou **ALT** ont des significations particulières qui ont été décrites au fur et à mesure de la description de chaque fonction.

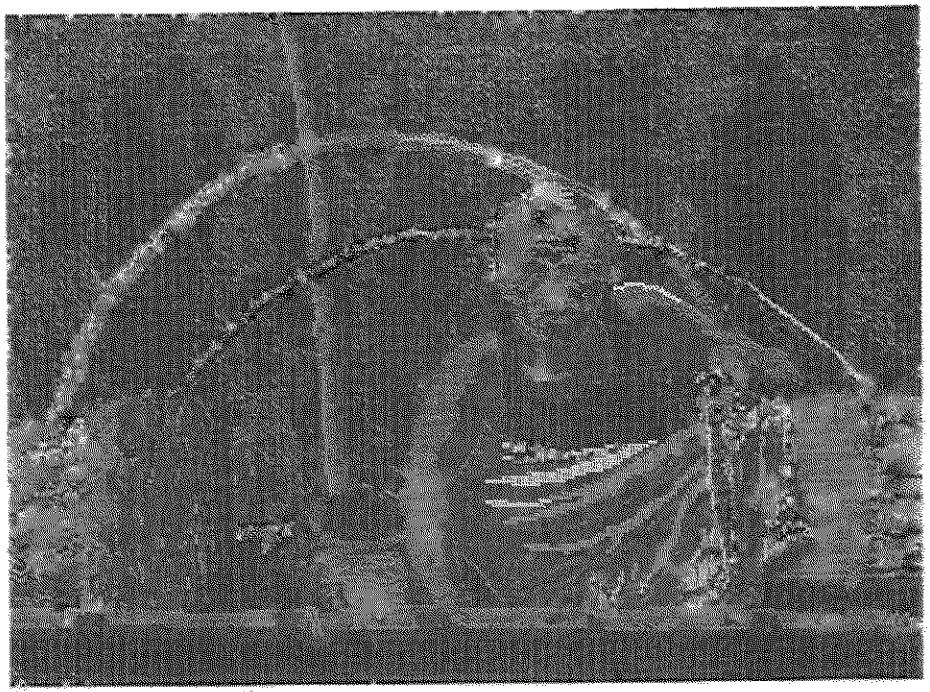
De plus, la plupart des menus peuvent être appelés directement depuis la feuille de dessin, sans qu'il soit nécessaire de revenir au menu principal. Les lettres à frapper correspondent soit à la première lettre du nom de la commande, soit à une lettre faisant partie de ce nom :

Insérer

F1 à **F8**	appel de la loupe avec grossissement croissant.
HELP	affiche la liste des écrans auxquels on peut associer un nom (utile si on dispose de plus de deux écrans).
ALT + flèches	déplacement du pinceau à l'écran.
ALT + **SHIFT** + flèches	idem mais d'un point à la fois.
ESC	active la création de bloc en affichant le grand réticule.
SHIFT ESC	permet de définir les blocs polygonaux.



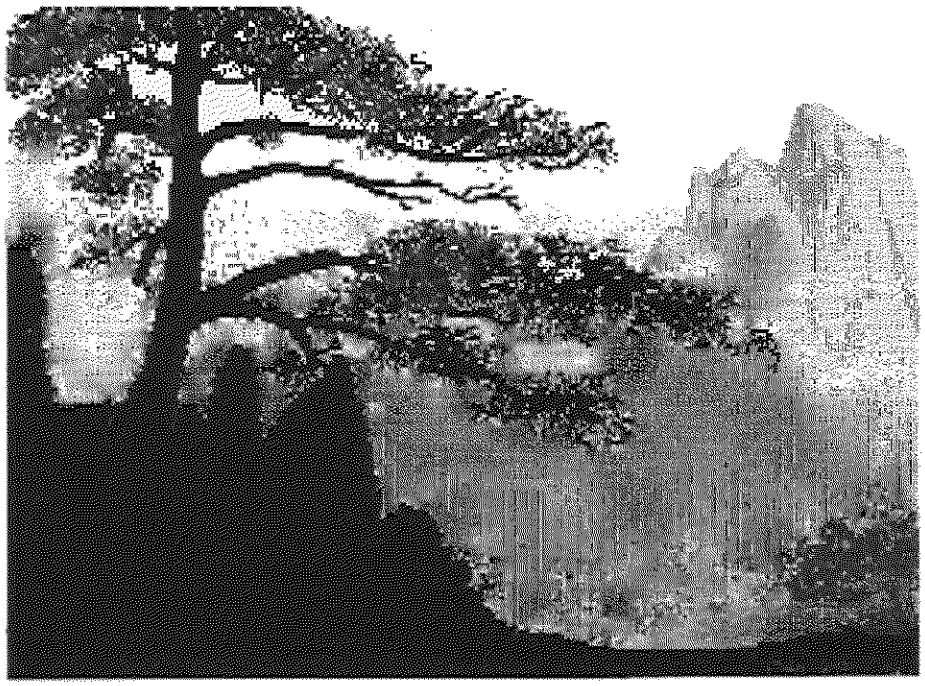
Raccourcis clavier depuis l'écran de dessin normal.



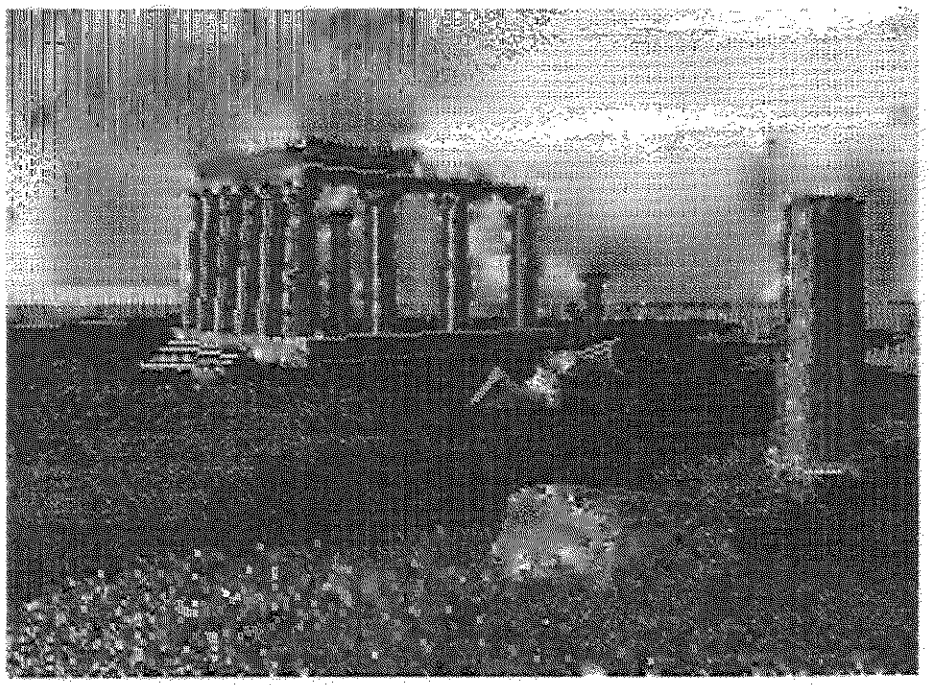
« Pêcheur »



« Tête enfant »



« Forêt »

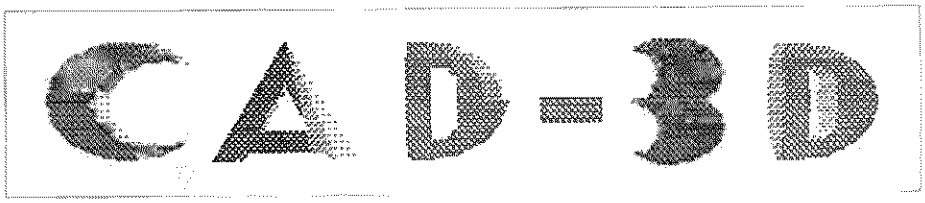


« Ruines »

5/4.2

CYBERSTUDIO : CAD-3D 2.0

CAD-3D, le logiciel de modelage en trois dimensions de Tom Hudson (le créateur de DEGAS) constitue dans sa version 2, un des maillons de la gamme infographique CYBER éditée par Upgrade et Antic Publishing. Il y côtoie d'autres prestigieuses réalisations telles que CYBERCONTROL, CYBERPAINT, CYBERTEXTURE, CYBERSCULPT.



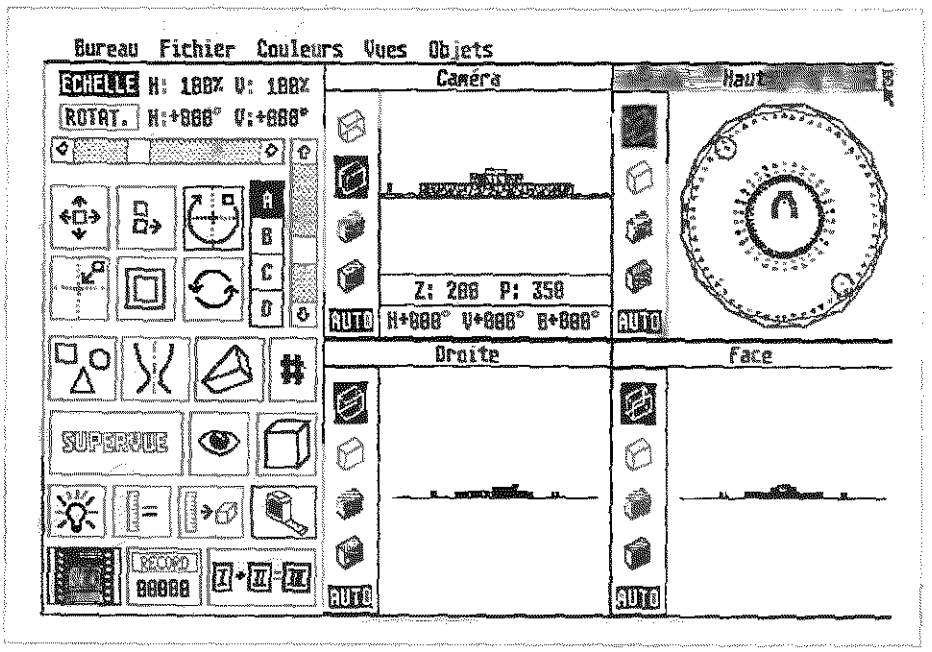
Sous la nouvelle appellation de CYBERSTUDIO, le package inclut en plus de CAD-3D 2.0, le langage de programmation CYBERMATE qui anime les scènes constituées avec le logiciel de modelage. A partir d'une silhouette en deux dimensions, CAD-3D génère très simplement des objets tridimensionnels grâce à un tour ou une extrudeuse. Après les avoir assemblés et positionnés pour constituer des scènes complexes, il les visualise en gérant les faces cachées.

L'utilisateur peut alors s'initier aux subtilités de la prise de vue cinématographique : peaufinage des éclairages, recherche du meilleur cadrage, vol de caméra, effets de zoom... et même visualisation en relief par procédé stéréoscopique (nécessitant des lunettes spéciales).

CAD-3D peut produire des images compatibles DEGAS et Néochrome. Il nécessite au moins un méga-octet de mémoire que l'on lance en cliquant sur CAD3D2.PRG.

L'ECRAN DE TRAVAIL

Le tiers gauche de l'écran de travail est occupé par une boîte à outils qui n'existait pas dans la version 1 de CAD-3D. Cette boîte réduit agréablement le nombre d'appels aux menus déroulants mais ampute une bonne partie de la surface réservée aux représentations des objets à l'intérieur des quatre fenêtres de vues.



La fenêtre **Camera** montre l'univers de travail en 3D tel qu'il apparaîtra en plein écran ou à l'impression. Lorsque cette fenêtre est sélectionnée, on peut déplacer la caméra dans toutes les directions autour des objets et zoomer pour ne faire apparaître que certains détails ; la caméra regarde toujours le centre de l'univers, comme si son objectif était plaqué en permanence contre une sphère transparente.

Les trois autres fenêtres **Haut**, **Droite** et **Face** présentent en vision fixe les différentes vues de l'univers en 2D. Depuis ces menus, on peut sélectionner un objet ou un groupe d'objets pour le dupliquer, le déplacer ou en modifier les dimensions. Le menu déroulant **Vue** propose par ailleurs d'autres options de visualisation : **Bas** au lieu de **Haut**, **Gauche** au lieu de **Droite** et **Dos** au lieu de **Face**. Ces fenêtres montrent les limites de l'univers et les objets ne peuvent en sortir : si le cas risque de se produire lors d'une manipulation, le logiciel le signale et il faut réduire leur taille pour que l'opération puisse se dérouler.



Les quatre fenêtres affichent sous forme de petits cubes le mode de représentation de l'objet qui peut d'ailleurs être différent d'une fenêtre à l'autre :



Sous forme fil de fer.



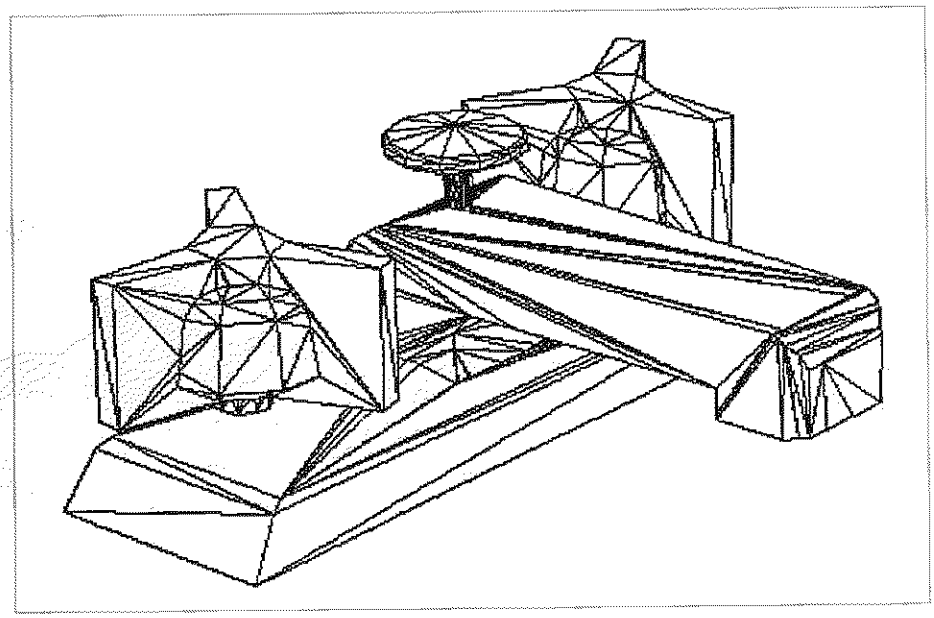
Avec les faces cachées.



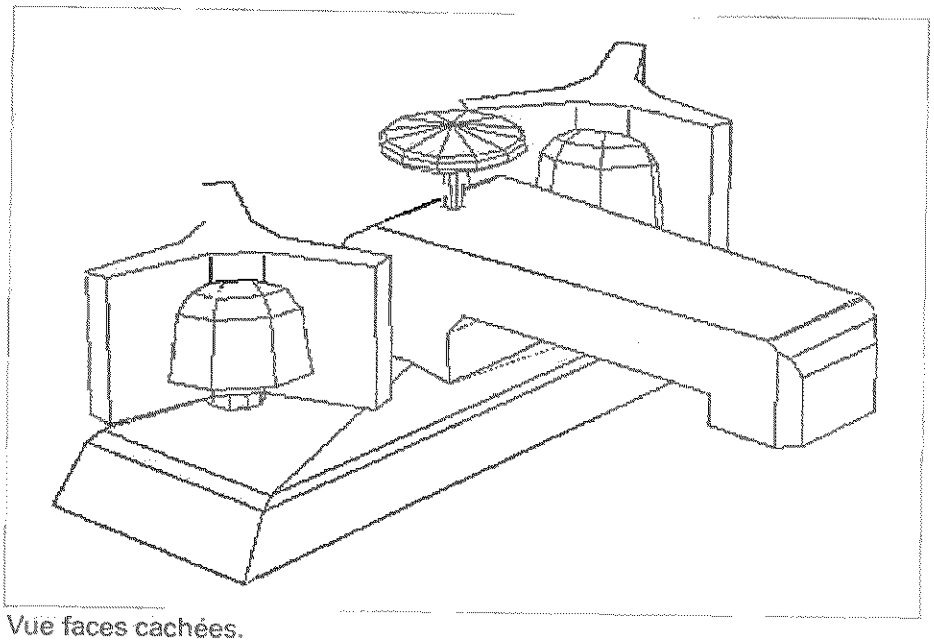
Avec ombrage ou coloriage des faces (mode **Solide**).



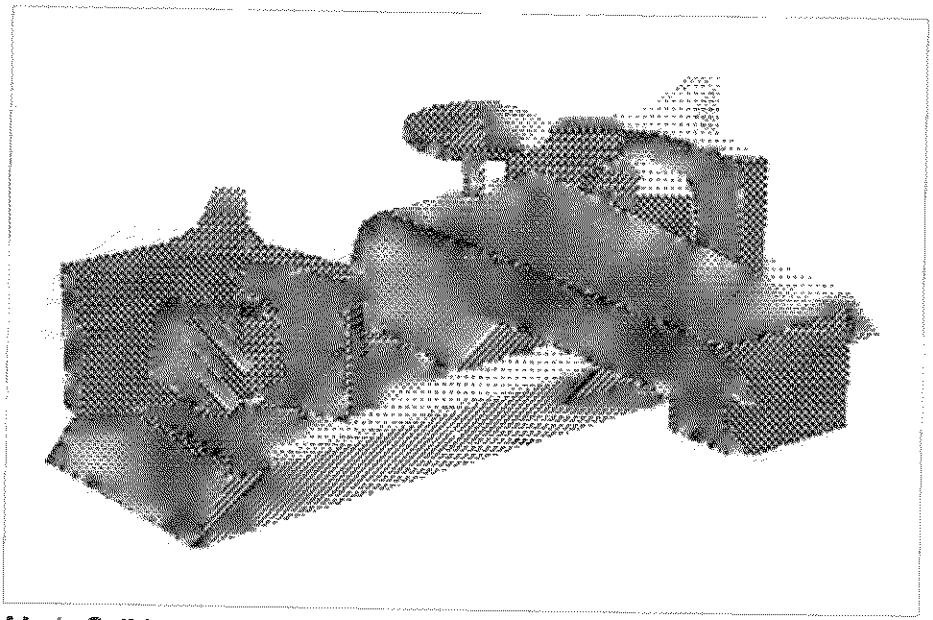
Avec ombrage ou coloriage des faces et visualisation des arêtes (mode **Solide contour**).



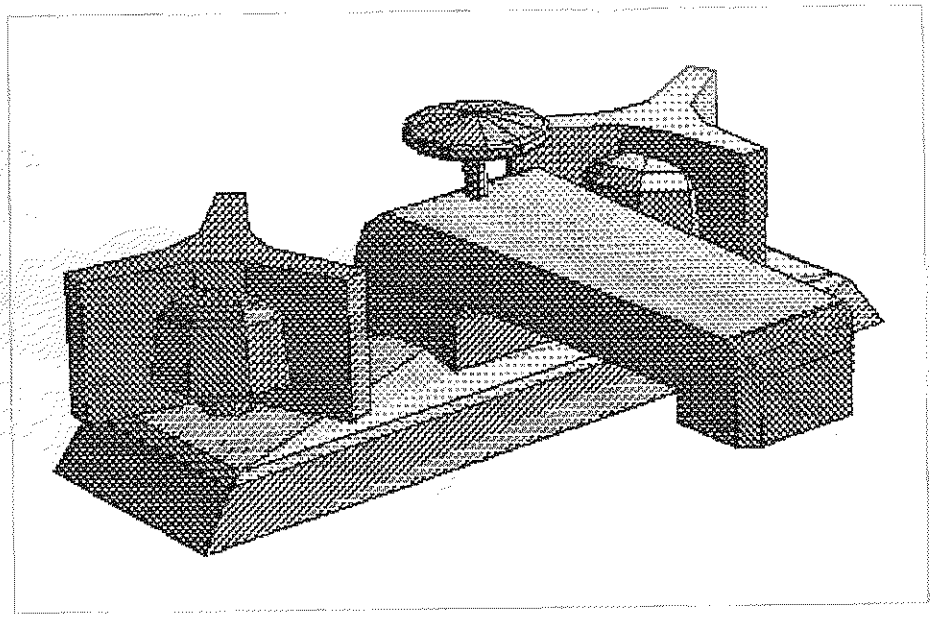
Vue « fil de fer ».



Vue faces cachées.



Mode Solide.

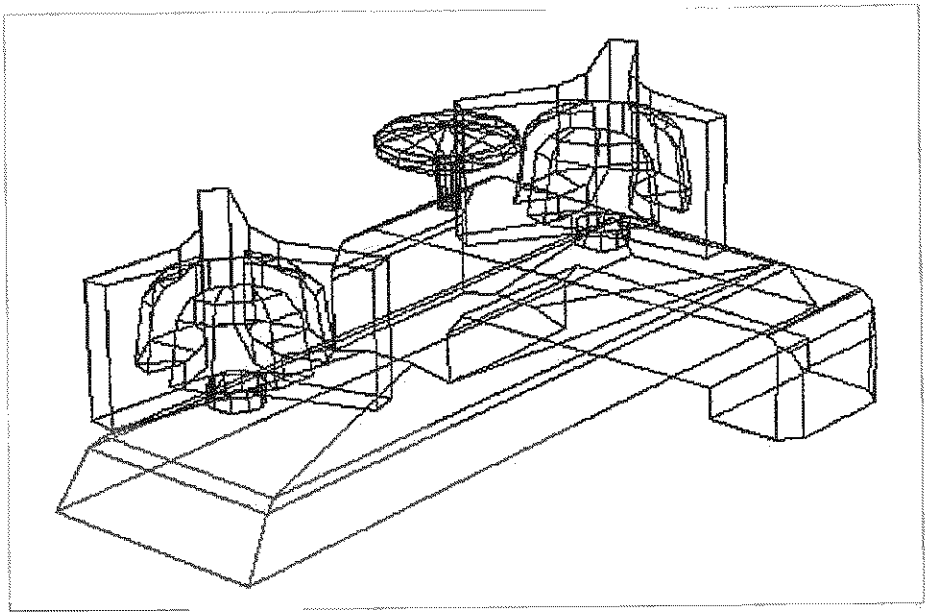


Mode Solide contour.

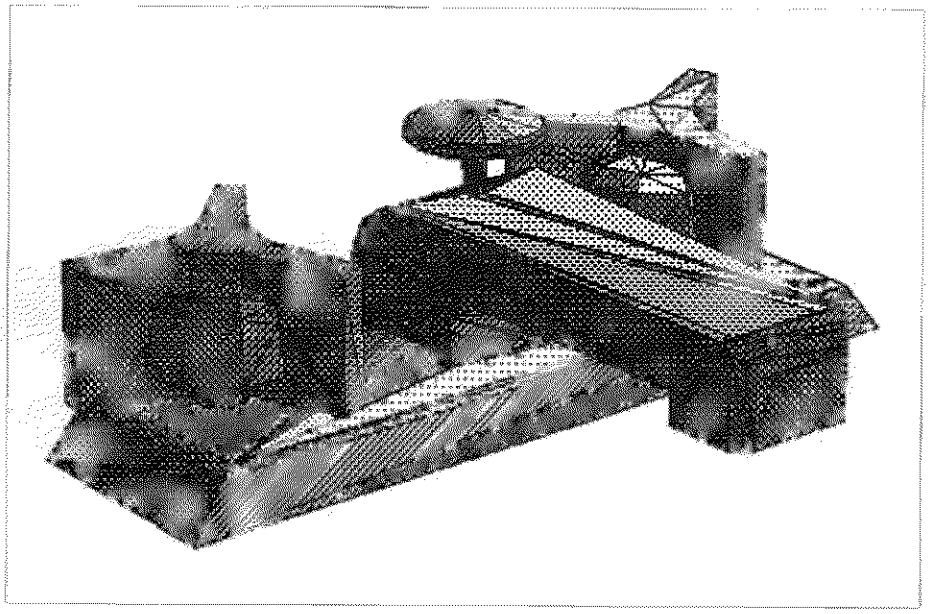




Si l'icône **Toutes lignes** est activée, les triangles constitutifs des faces des objets apparaissent produisant des effets graphiques qui dans certains cas peuvent s'avérer heureux.



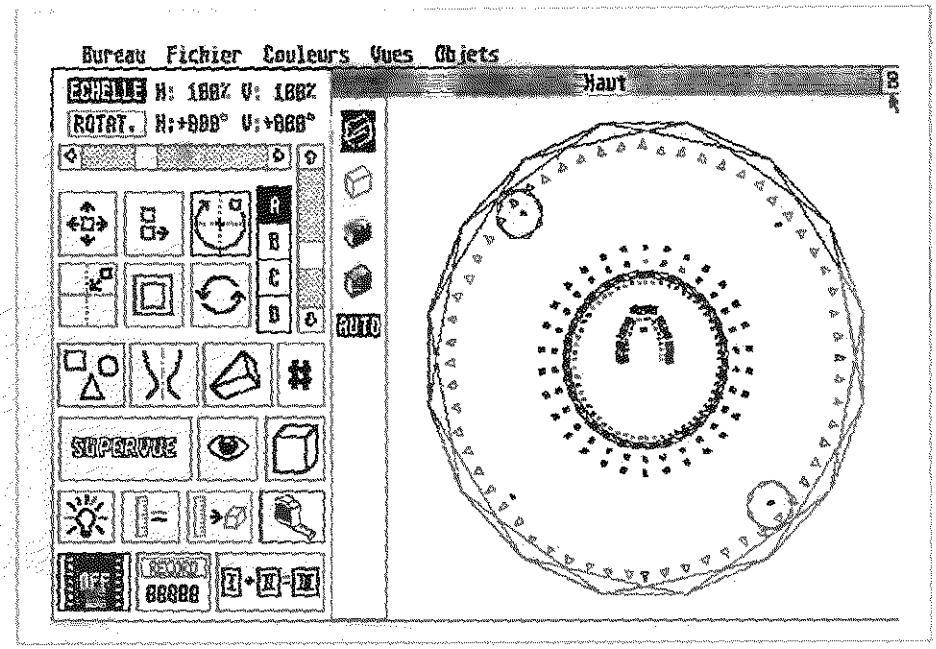
Affichage en mode Toutes lignes.



Les lignes délimitent les zones d'ombre.

Le bouton **Auto** lorsqu'il est activé à l'intérieur d'une fenêtre permet le rafraîchissement de cette dernière en cas de modification depuis n'importe quelle vue.

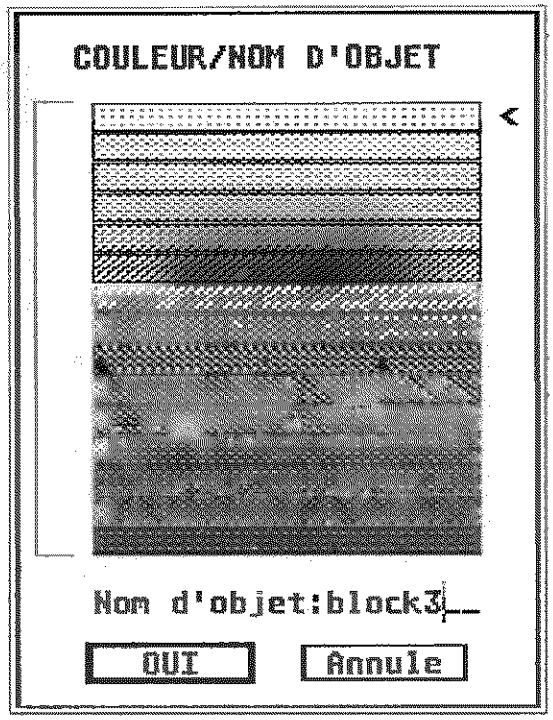
Pour un travail précis à l'intérieur d'une fenêtre, il est nécessaire de la faire passer en mode plein écran, d'abord en la sélectionnant par un clic puis en activant la case **Agrandissement/Reduction** située dans l'angle supérieur droit.



Agrandissement d'une fenêtre pour un travail plus précis.

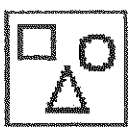
Construction d'un objet

Tout nouvel objet peut être coloré (ou tramé) et doit surtout être nommé par l'utilisateur qui dispose pour cela de huit caractères. Ce nom permettra par la suite certaines opérations telles que la sélection d'objets, la réunion de deux objets en un, la création de trous à l'intérieur d'un objet, etc. La boîte de saisie du nom apparaît chaque fois que l'on vient de créer un objet suivant une des différentes méthodes décrites ci-dessous.



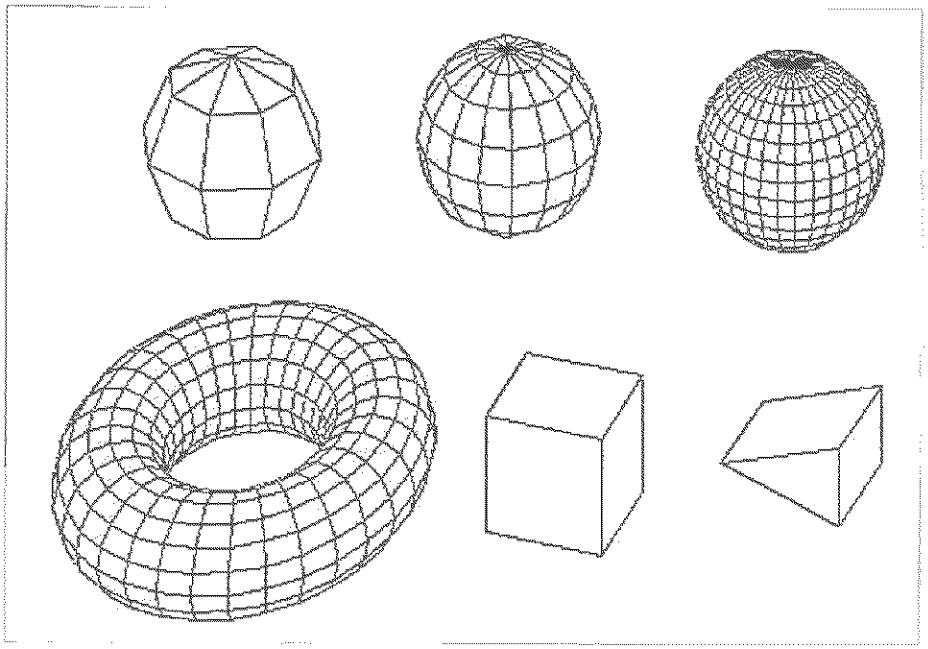
Un objet résulte :

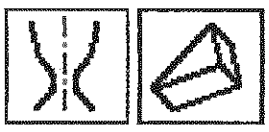
- soit du menu des formes primitives ;
- soit d'une extrusion ;
- soit d'un tournage ;
- soit enfin de la réunion de deux objets.



Menu des formes primitives

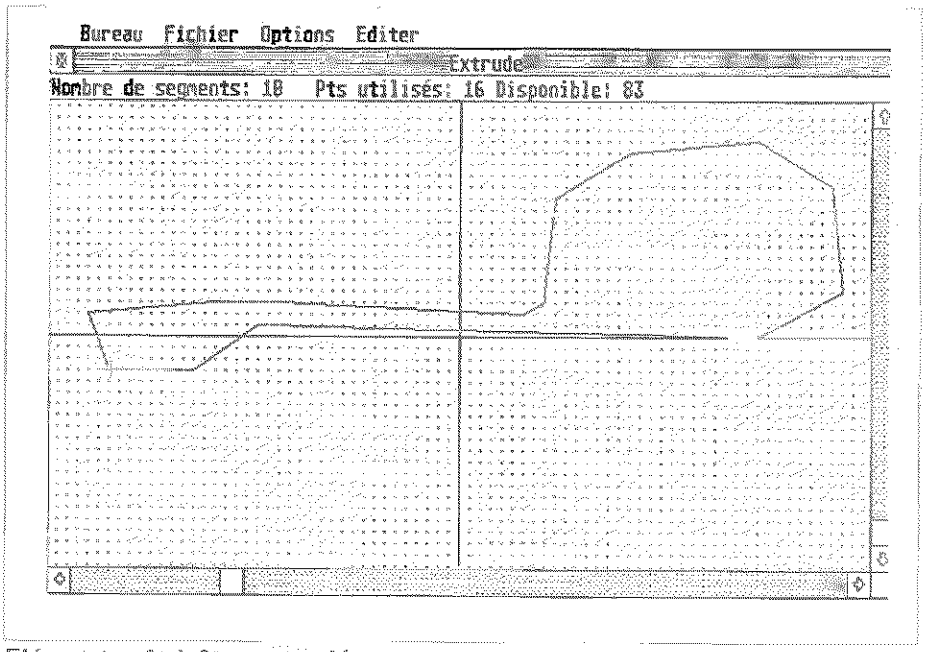
Six formes de base sont disponibles dans ce menu à titre de démonstration : deux formes extrudées, un cube et un prisme, et quatre formes tournées, un tore et trois sphères dans des résolutions de tracé différentes (ce qui permet de constater que le temps d'affichage ou de rafraîchissement est bien proportionnel au nombre de faces de l'objet).





Items communs aux menus Extrusion et Tournage

Les pages de construction d'objets accessibles depuis les menus Extrusion et Tournage possèdent beaucoup d'éléments en commun :



Elément prêt à être extrudé.

Le tracé des objets s'effectue en deux dimensions. On en réalise la silhouette plane (le gabarit) en plaçant dans le sens des aiguilles d'une montre la succession des sommets du polygone constitutif. On peut placer de la sorte un maximum de 99 points successifs ; la ligne d'état située en haut de la fenêtre rend compte au fur et à mesure du tracé du nombre de points encore disponibles.

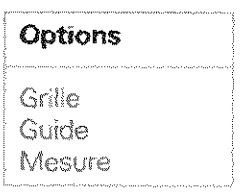
Attention : il est impératif que les segments constitutifs d'un objet ne se coupent pas.

Menu **Fichier**



Dans certains cas on peut souhaiter pouvoir sauvegarder puis recharger ultérieurement un gabarit car ce dernier ne peut pas être régénéré par l'objet 3D qu'il a fabriqué. Un fichier de gabarit occupe de plus une taille minime sur le disque puisqu'il ne mémorise que quelques points en 2D.

Menu **Options**



Comme dans la plupart des logiciels de dessin vectorisé, la **Grille** et le **Guide** facilitent le positionnement des points constitutifs du gabarit. Il est dommage que le pas de la grille ne puisse pas être modifié. Les médianes du rectangle de travail aident également au repérage des sommets, surtout dans le cas d'objets symétriques.

Mesure n'est actif que si un objet servant de référence a auparavant été défini. La mesure de chaque segment est alors affichée au fur et à mesure de la construction de l'objet.



Menu **Editer**

Editer
Effacer Enlever
Elastique

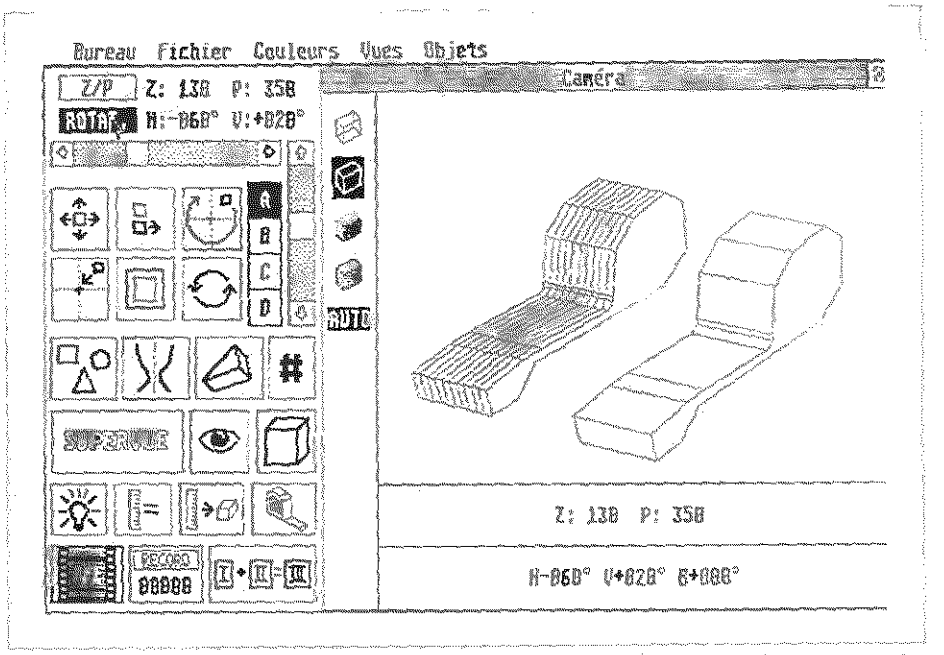
Effacer produit une grille de tracé vierge (le gabarit du dernier objet créé reste mémorisé sur la page) alors que **Enlever** (ou touche **Undo**) supprime le dernier point placé puis l'avant dernier etc., selon le principe d'une pile.

Elastique affiche ou non le segment élastique de retour au point initial.

L'ascenseur vertical déplace le réticule de pointage d'un point à l'autre afin d'effectuer des corrections par effacement puis remplacement du point courant.

L'extrusion

A l'aide d'un emporte-pièce de forme quelconque, on découpe l'objet à l'intérieur d'une plaque d'épaisseur constante. Le tapis roulant (ascenseur « horizontal ») définit le nombre de couches constitutives de l'objet : ce nombre de couches, indiqué dans la ligne d'état, n'influe absolument pas sur l'épaisseur de l'objet et permet simplement de créer des effets de texture (il est généralement préférable de le laisser à 1). Le menu **Option/Extruder** génère l'objet au centre de chacune des fenêtres actives de l'écran principal.



Deux essais d'extrusion en multi et mono-couche. L'épaisseur est la même.

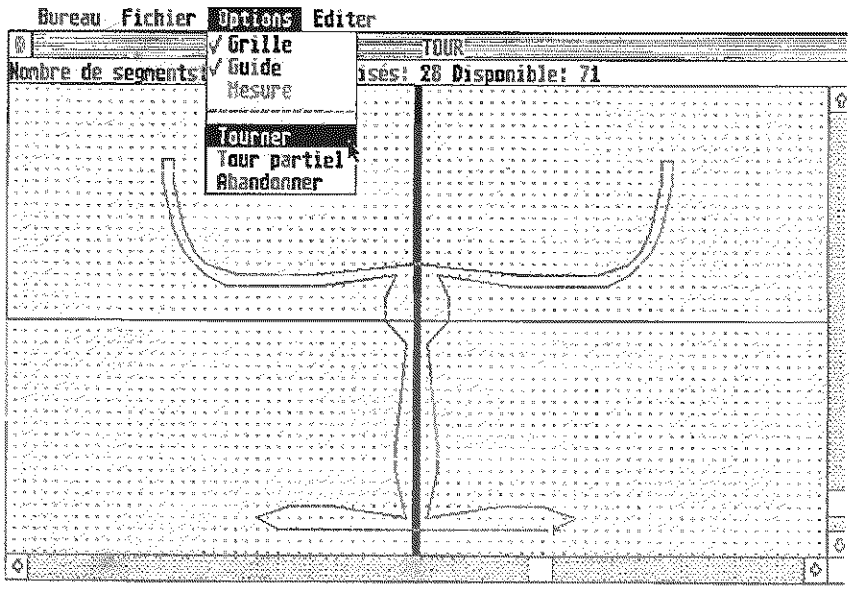
Remarque

Si l'objet est complexe, il est intéressant de le réaliser à partir de plusieurs formes simples plus faciles à travailler et à modifier. Il suffit ensuite de réunir ces différentes formes en un seul objet. Il est toutefois regrettable que le logiciel ne permette pas d'afficher l'image fantôme d'un gabarit existant lors de la création d'une nouvelle forme : cette option faciliterait énormément le placement de nouveaux éléments (trous par exemple).

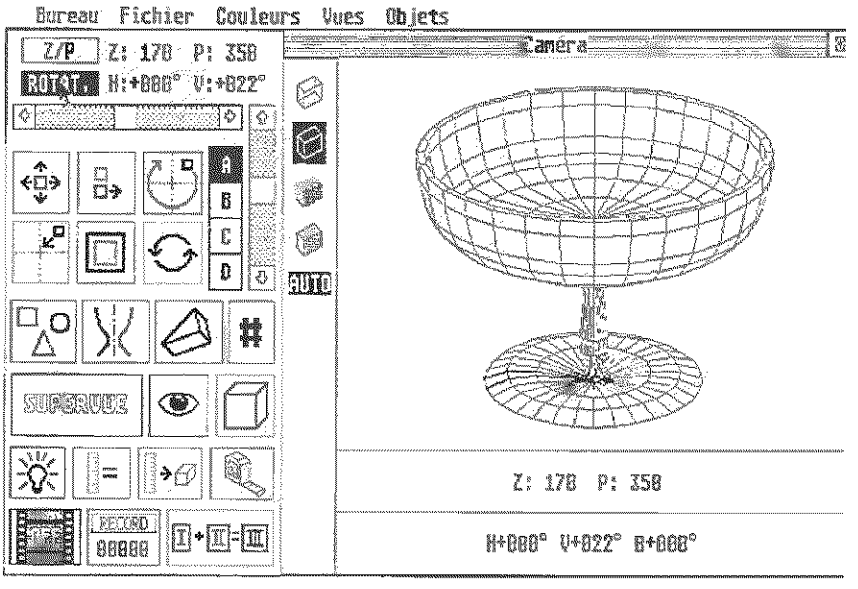
Le tournage

La ligne d'état en haut d'écran indique de combien de segments ou de faces l'objet résultant sera constitué : un nombre important accroît la qualité du lissage, mais au détriment de la vitesse de tracé.

Le tracé s'effectue dans la partie droite de l'écran. Le menu **Editer/Connecter** relie le dernier point tracé au premier comme dans le menu **Extrusion**. On doit désactiver **Connecter** si on souhaite réaliser un objet sans épaisseur ou lorsque certains points touchent l'axe vertical.

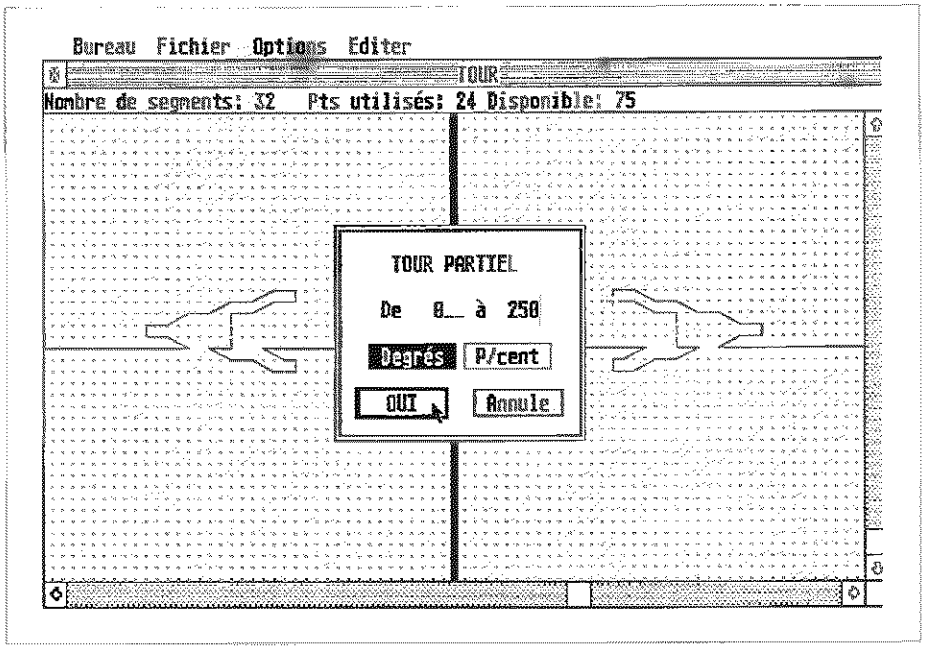


Préparation du gabarit...

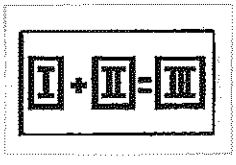


... et résultat du tournage.

Option/Tour partiel ne tourne que la partie de l'objet située entre les limites définies par l'utilisateur alors que Option/Tourner génère un objet complet.

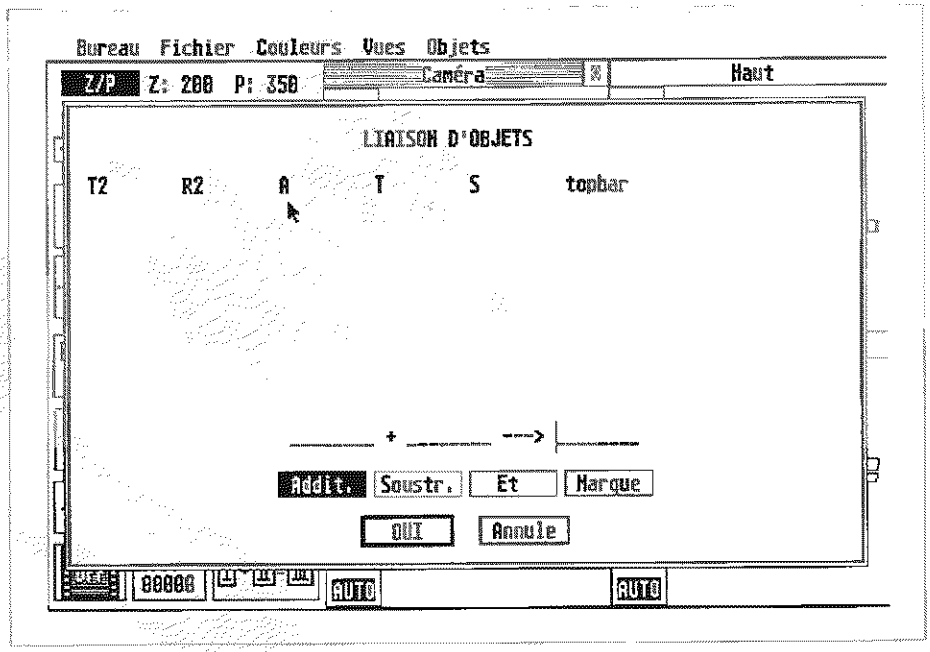


Réalisation d'un tour incomplet.



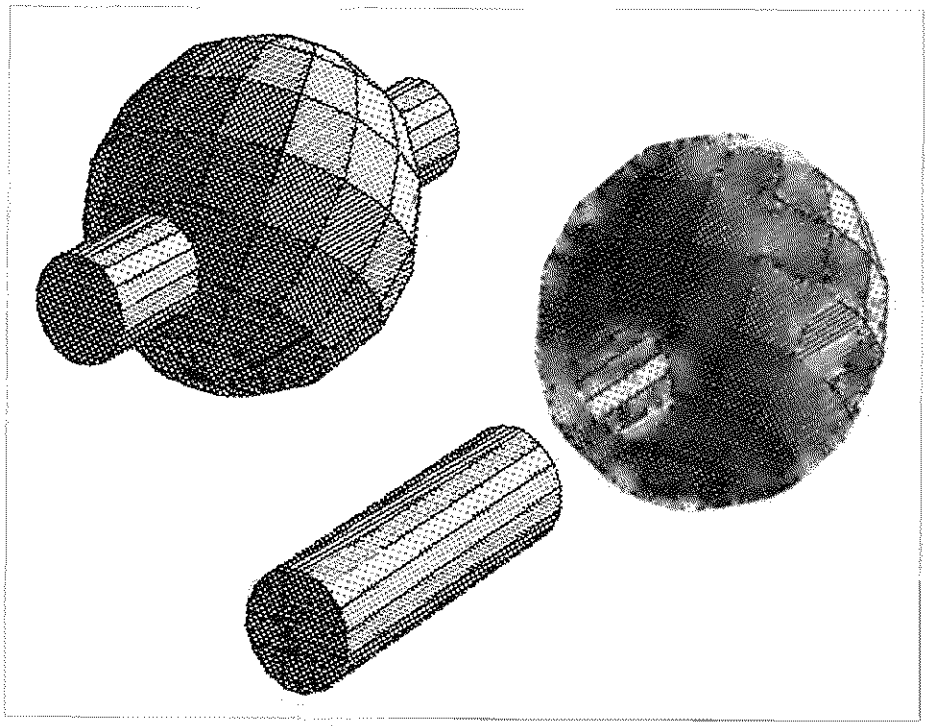
La réunion d'objets

Cette option, une des plus puissantes de CAD-3D, n'est disponible que si plus de deux objets sont présents en mémoire. On dispose alors par l'intermédiaire d'une boîte de saisie de quatre types d'opérations :

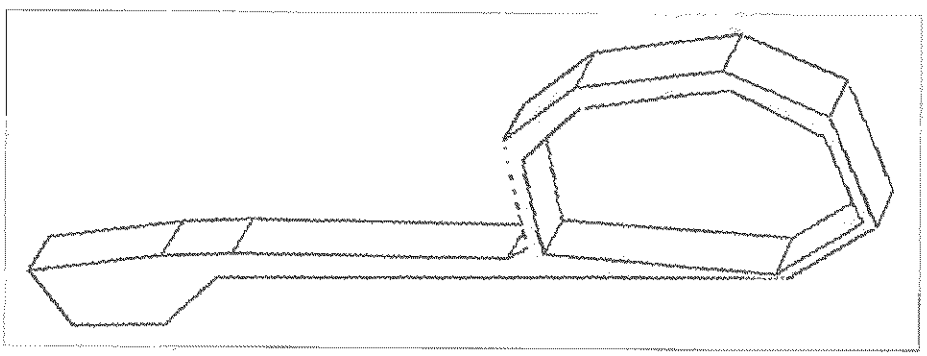


– l'addition de deux objets (sans qu'il soit nécessaire que ceux-ci se touchent ; on retrouve dans ce cas la fonction **Groupement** présente dans de nombreux logiciels de dessins vectorisés : la dissociation ultérieure est cependant impossible) ;

– la soustraction de deux objets qui permet la réalisation de trous ;

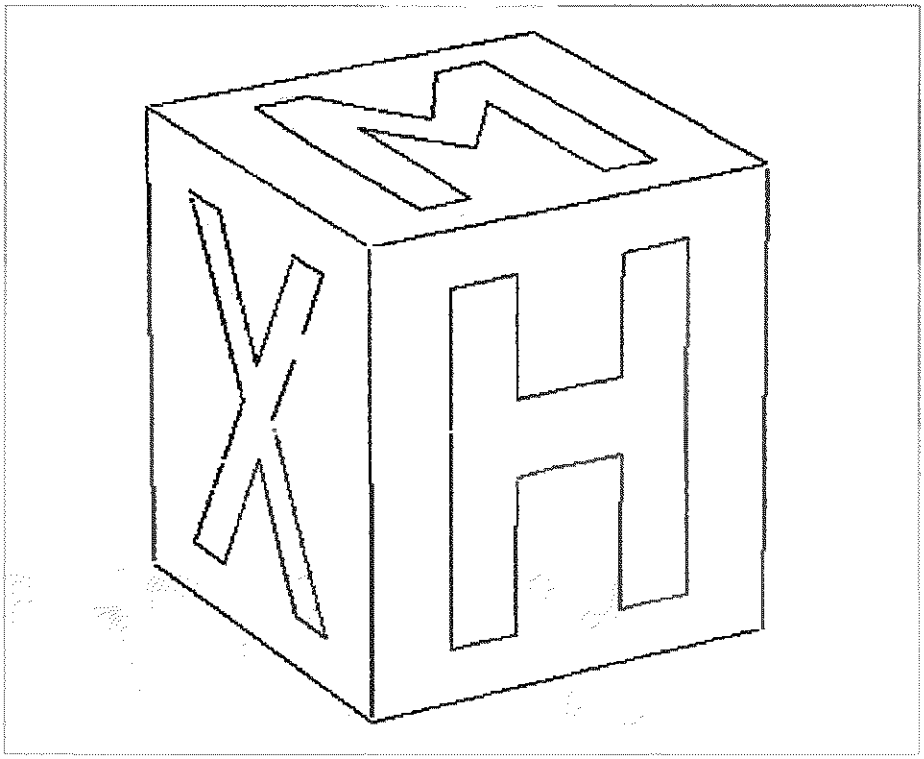


Addition, soustraction et ET logique entre une sphère et un cylindre.

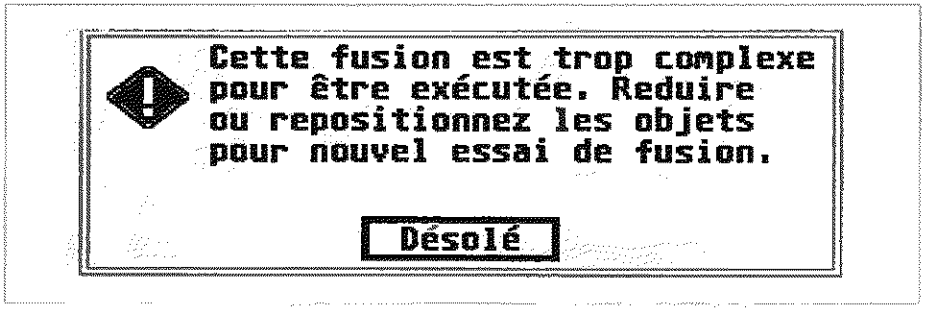


Pour que le trou apparaisse l'objet à soustraire doit être plus large que l'objet initial.

– la création d'un ET logique entre deux objets (seul le volume d'intersection est conservé) ;
– le marquage d'un objet sur la face d'un autre.



A l'issue de l'opération, particulièrement gourmande en temps de calcul (au delà de 20 minutes, il est préférable d'interrompre la jonction qui risque de se terminer par un échec car les formes sont trop complexes), on obtient un troisième objet qu'il est également nécessaire de nommer.



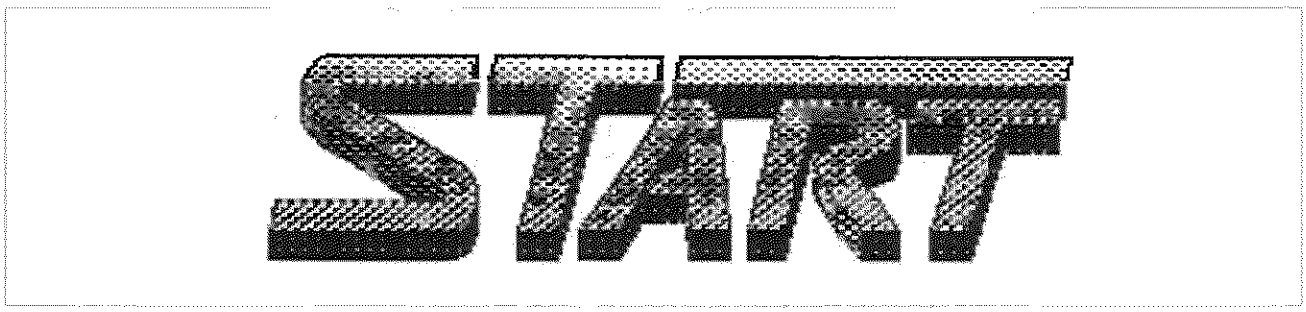
Attention : pour ne pas encombrer la mémoire qui peut contenir un maximum de 80 objets différents, les deux objets initiaux sont effacés et donc définitivement perdus (il est préférable d'en faire une sauvegarde préalable au cas où l'opération se passerait mal).

MOUVEMENTS DE CAMERA

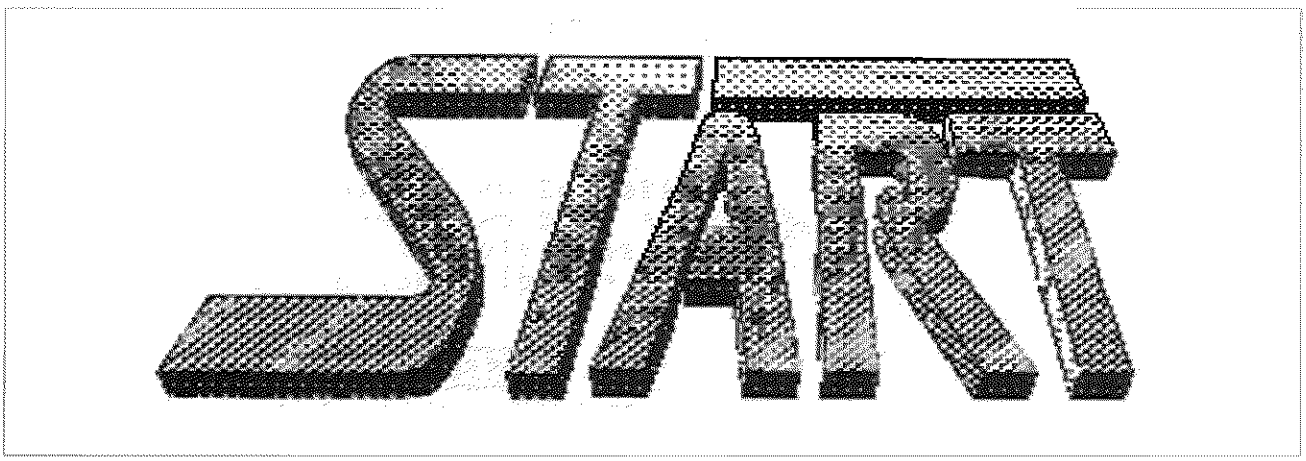
Contrairement à ce qui se produit dans les autres fenêtres de visualisation 2D, toute action effectuée dans la fenêtre **Camera** n'apporte aucune modification aux objets définis : c'est seulement le point de vue de l'utilisateur qui change. Le viseur de la fenêtre caméra affiche plusieurs paramètres sous l'image affichée :

– le facteur de zoom : par défaut 200 (à 0 l'objet apparaît comme un point lointain ; le grossissement maximal est de 999) ;

– le facteur de perspective : par défaut 350 (999 correspond à la déformation obtenue avec un grand angle alors que 0 annule toute déformation de perspective comme c'est le cas avec un télé-objectif) ;



Du télé-objectif (0)...



au grand-angle (999).

– la position de la caméra dans le plan horizontal H : par défaut 0° en face de l'objet. La caméra tourne autour de l'objet, l'objectif vise le centre de l'univers, ainsi à 180° l'objet est-il vu de dos... ;

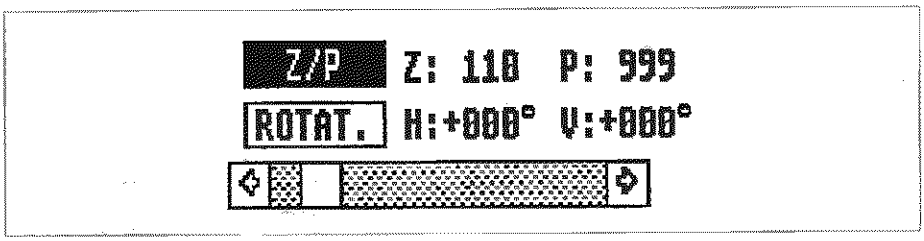
– la position de la caméra dans le plan vertical V : par défaut 0° au ras du sol. La caméra tourne autour de l'objet mais dans le plan vertical,



l'objectif vise le centre de l'univers ; ainsi à 90° l'objet est-il vu depuis le zénith, à 160° le monde est à l'envers... ;

– l'inclinaison de la caméra autour de l'axe de l'objectif B : par défaut 0° caméra horizontale.

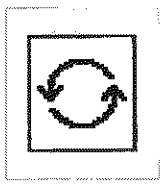
Ces paramètres correspondent à des positionnements absolus de la caméra dans l'espace et ils peuvent être modifiés depuis les deux premières lignes de la boîte à outils.



Réglage de la caméra.

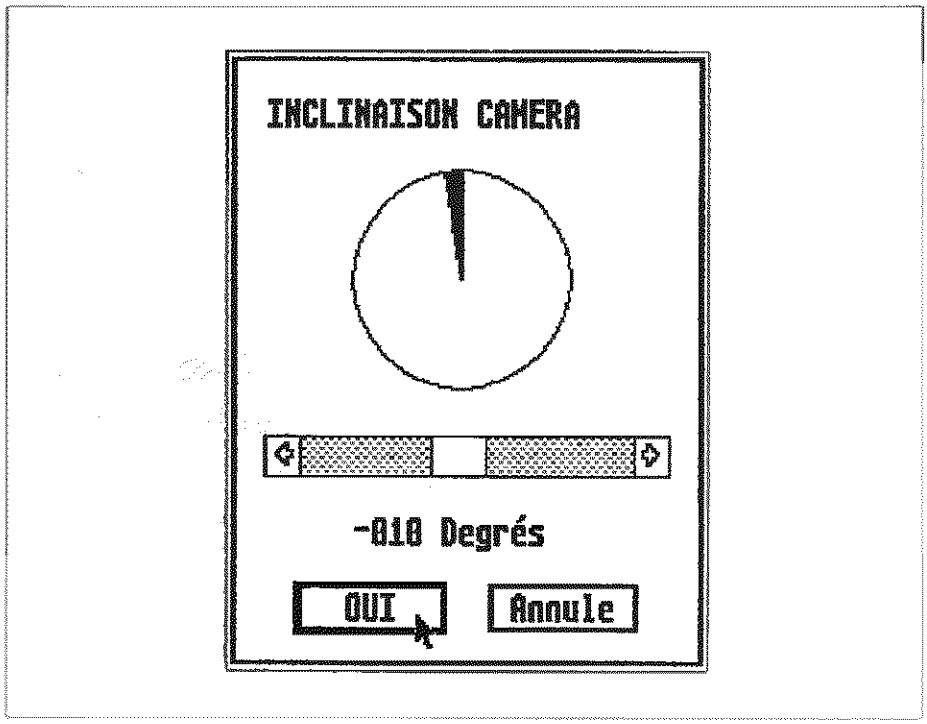
Les modifications n'ont pas lieu en temps réel et ne sont validées que lorsqu'on clique dans la fenêtre caméra ou qu'on appuie sur la barre d'espace :

– lorsque la case **Z/P** est activée, le tapis roulant ou les flèches gauche et droite du clavier éventuellement amplifiées par **SHIFT** permettent de modifier le facteur de zoom alors que l'ascenseur ou les flèches haute et basse également amplifiées modifient le rapport de perspective ;

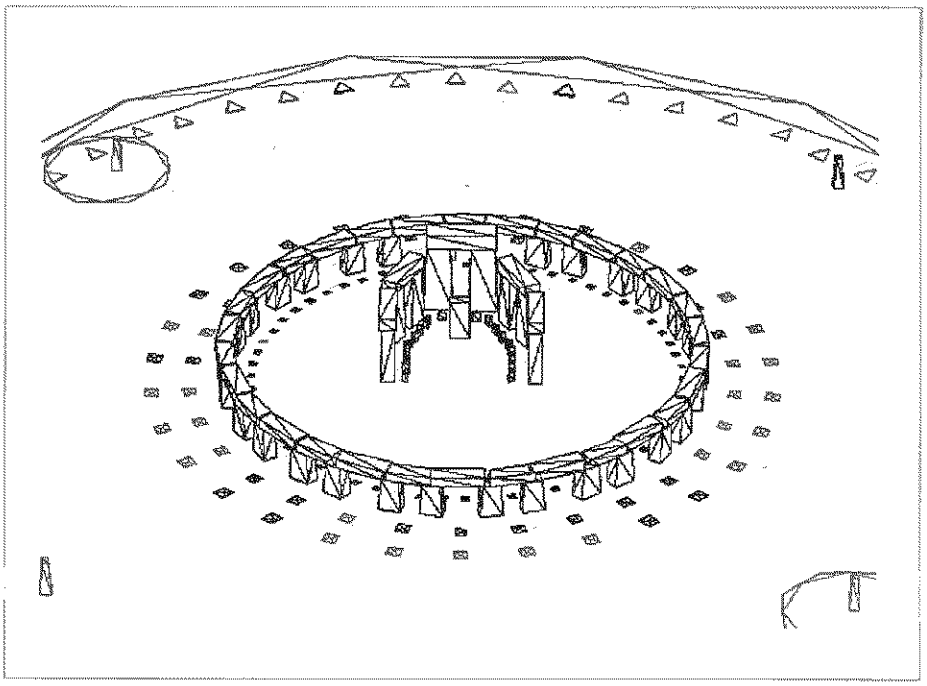


– si la case **ROTAT** est activée, les mêmes ascenseurs et les mêmes touches agissent sur la position horizontale ou verticale de la caméra ;

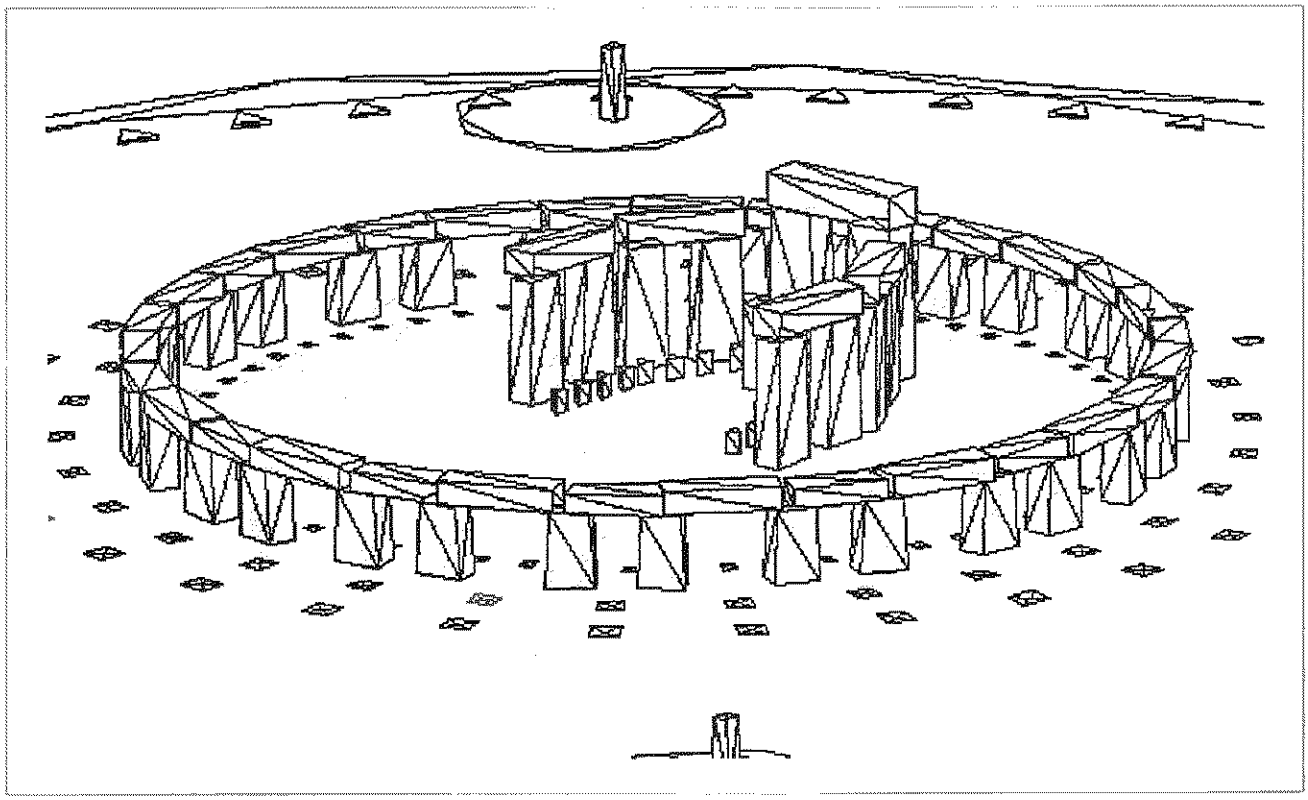
– l'icône inclinaison de la caméra donne accès à une boîte de réglage où la touche **SHIFT** permet d'amplifier la variation de l'angle d'inclinaison lorsqu'on clique sur les boutons du tapis roulant.



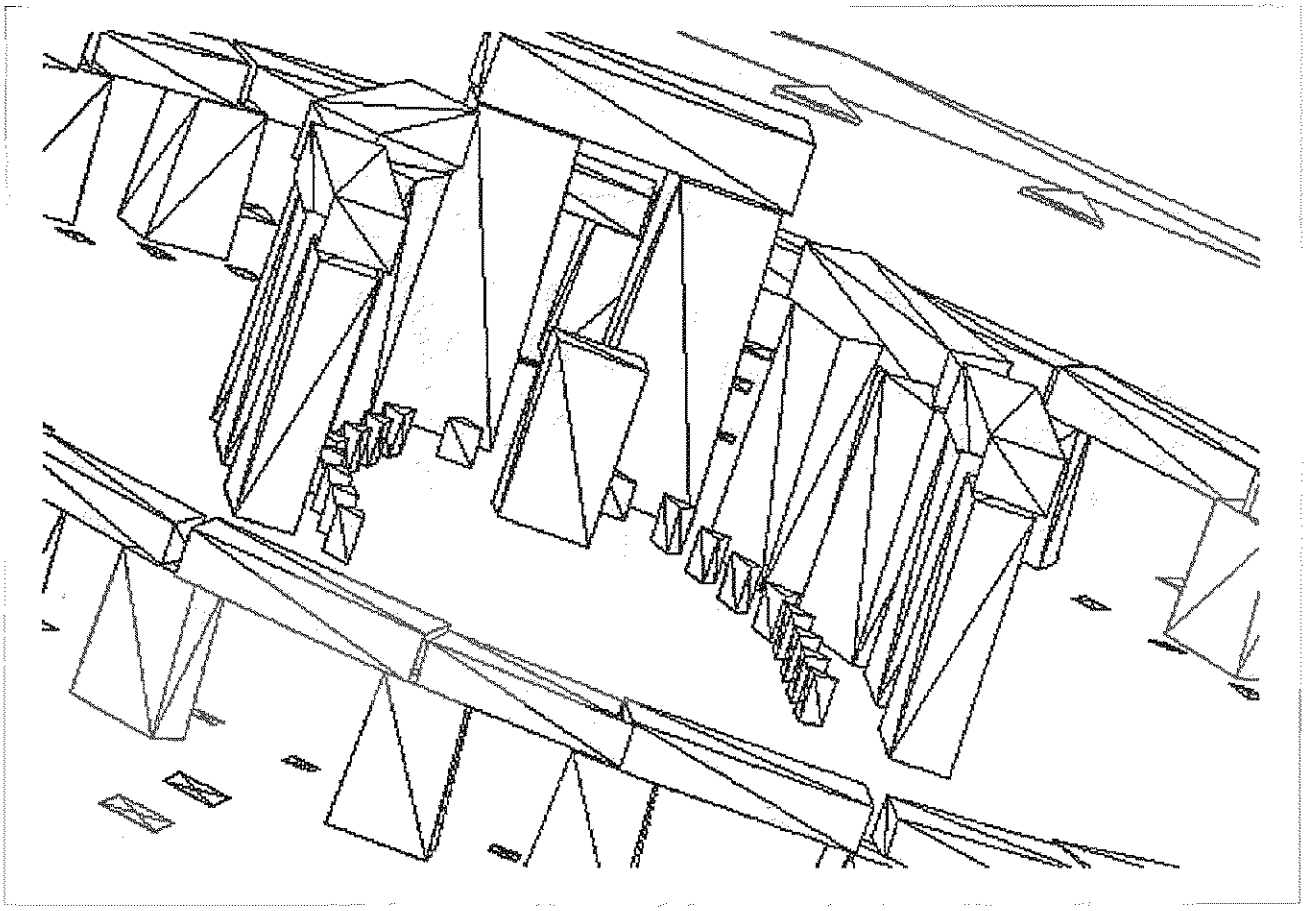
Ces quelques vues du site reconstitué de Stonehenge mettent en évidence les possibilités des mouvements de caméra :



Une première vue avec un mouvement de caméra de 25° vers le haut.



La caméra pivote de 30° vers la droite et redescend à 15° par rapport à l'horizontale ; le zoom passe à 300.



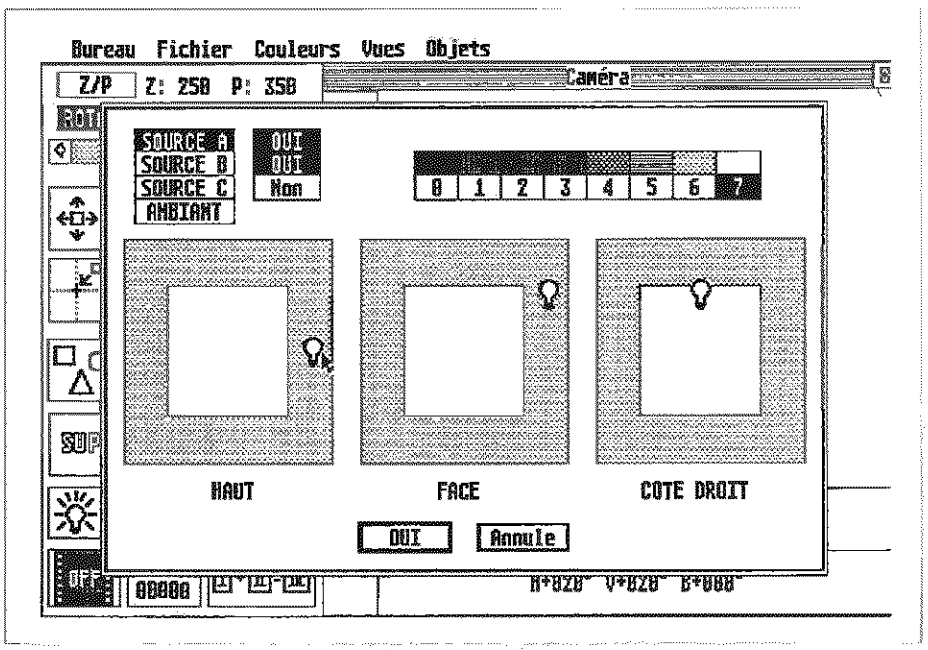
Vue dynamique avec un zoom de 650 et une caméra située à 10° vers la gauche, inclinée de 20° par rapport à l'axe de l'objectif : on se croirait aux commandes d'un simulateur de vol...



L'ECLAIRAGE



L'aspect final d'un objet dépend du soin apporté à l'éclairage de la scène. Le placement correct des spots requiert une certaine expérience et de nombreux essais sont généralement nécessaires avant de parvenir au but souhaité.

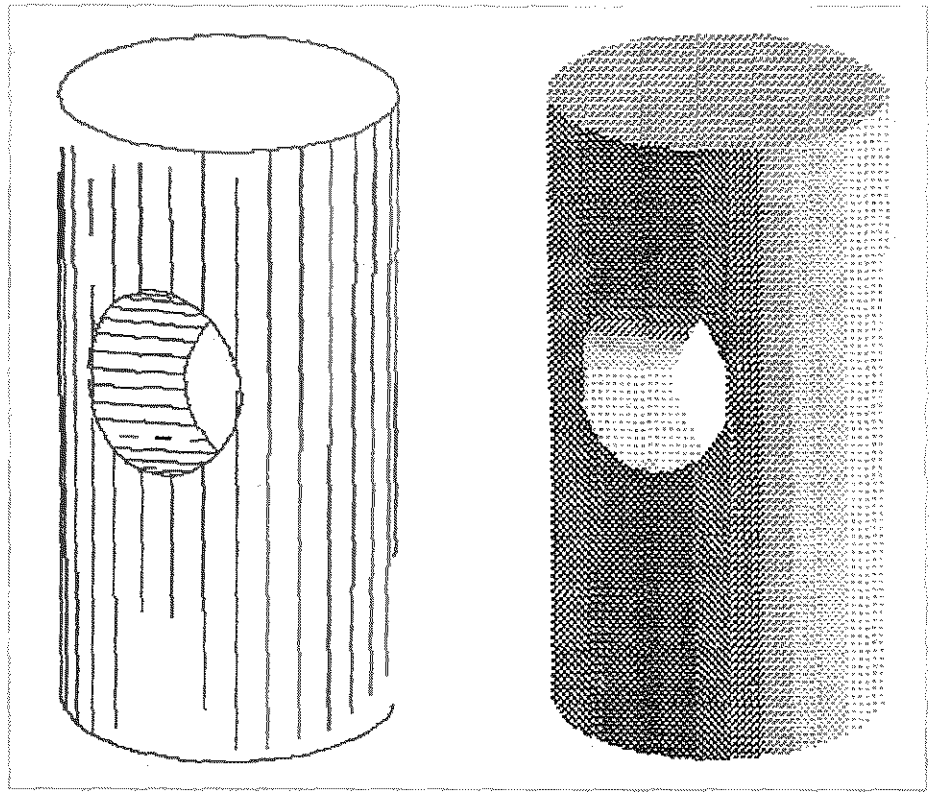


Cyberstudio propose trois spots (Sources A, B et C) qui peuvent être placés n'importe où, y compris à l'intérieur de la scène ainsi qu'une lumière d'ambiance destinée à adoucir les ombres. L'éclairage se passe en trois temps :

1 – Choix des spots à allumer : on en choisit généralement au moins deux afin de déboucher une ombre trop forte sur le côté opposé à la source principale.

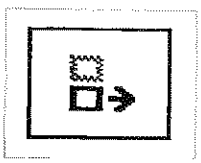
2 – Choix de l'intensité lumineuse de chaque source de lumière : on sélectionne la source en cliquant sur son nom, puis on lui attribue une intensité de 0 (éteint) à 7 (maxi) sur la réglette. On peut également donner une valeur à l'éclairage d'ambiance qui réduit l'échelle des gris ou l'éteindre complètement (0) afin d'obtenir l'équivalent photographique d'un tirage dur.

3 – Choix de la position des spots : on règle les spots les uns après les autres. Les trois fenêtres qui apparaissent (**Haut**, **Face** et **Côté droit**) représentent en réalité l'emplacement du même spot vu sous trois angles différents. Si l'ampoule est placée dans la zone grisée de la fenêtre, on considère que la lumière vient de l'infini et tous les objets auront la même face éclairée ; si elle se trouve dans la partie blanche, la lumière se situe à l'intérieur de l'espace où se trouvent les objets qui seront éclairés différemment en fonction de leur position par rapport à l'ampoule.



OPERATIONS SUR LES OBJETS

Un objet ou un groupe d'objets peut être sélectionné et retravaillé mais uniquement à l'intérieur des fenêtres qui en proposent un affichage 2D, c'est-à-dire les vues de **Haut**, de **Droite** ou de **Face**. La vue **Caméra** ne permet, elle, que la modification du point de vue sur la scène. Préalablement à toute sélection, il faut donc activer la fenêtre de vue désirée en cliquant à l'intérieur de façon à ce que son bandeau apparaisse en grisé. Si on souhaite un rafraîchissement à l'intérieur des autres fenêtres en cours de travail, la case **Auto** doit de plus être activée.



L'icône **Solo/Omni** joue ici un rôle essentiel puisqu'en fonction du mode choisi le travail s'effectuera soit sur un objet isolé (mode **Solo** : un seul carré noir) soit sur la totalité des objets affichés (mode **Omni** : les deux carrés de l'icône sont noirs).

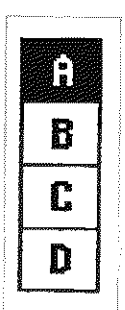
Sélection d'un objet

Un double clic sur un objet fait disparaître tous les autres objets de la fenêtre. L'élément restant peut alors être déformé indépendamment des autres comme nous le verrons par la suite.

Un double clic sur une partie vide de la fenêtre fait réapparaître tous les objets qui avaient été masqués.

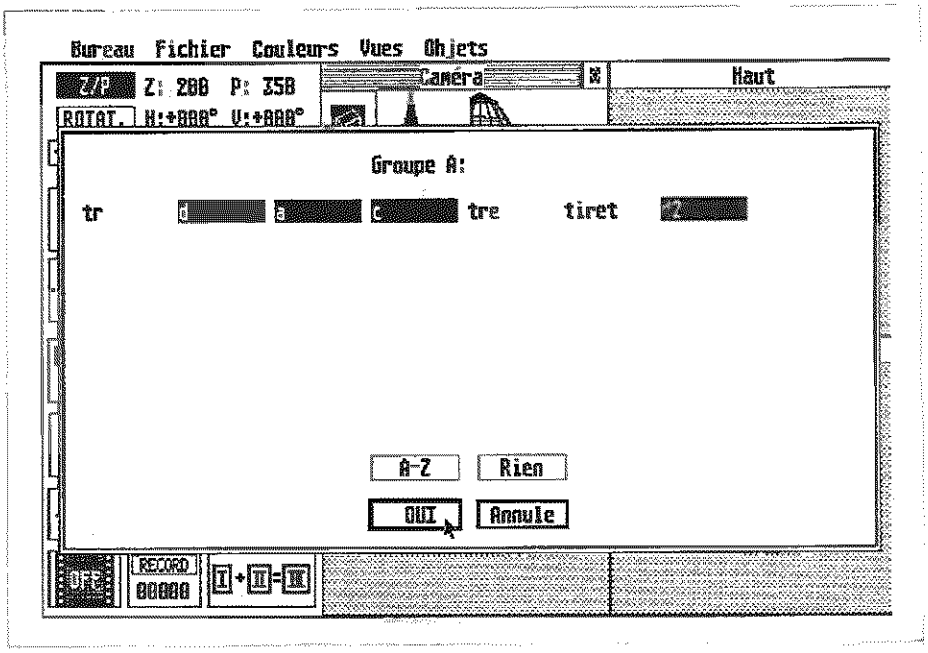
Sélection d'un groupe d'objets

Dans une scène comportant beaucoup d'éléments, il est intéressant de grouper les différentes parties constitutives d'un objet afin de pouvoir les afficher simultanément pour les retravailler tout en masquant les autres éléments de la scène.



Quatre groupements différents sont possibles et chacun correspond à une étiquette A, B, C ou D telles qu'elles apparaissent dans la boîte à outils.

Un double-clic sur une des étiquettes ouvre une boîte de saisie des éléments devant faire partie du groupe. Dans cette boîte, tous les noms des objets définis apparaissent, et on sélectionne ceux que l'on désire incorporer au groupe en les mettant en inversion vidéo. On accède également à cette boîte de sélection par l'intermédiaire du menu **Objet/Sélectionner** après avoir cliqué une fois sur une des étiquettes pour la mettre en inversion.



Par la suite un simple clic sur une des étiquettes A, B, C ou D aura pour effet de n'afficher que les éléments appartenant au groupe.

Sauvegarde d'un goupe

Le menu Fichier/Sauver groupe ne sauve sur le disque que les éléments appartenant au groupe courant.

Suppression d'objet

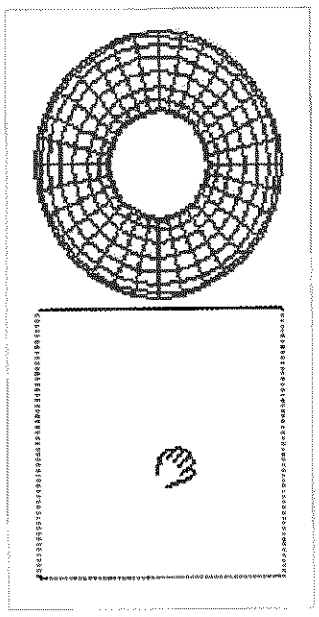
Une première solution consiste à ne pas afficher l'objet devenu inutile en recourant à la technique des groupements ou des sélections vue précédemment.

Si l'objet doit définitivement être effacé de la mémoire, le menu Objet/Effacer propose une boîte de saisie semblable à celle des créations de groupes et dans laquelle on peut sélectionner un ou plusieurs objets à effacer.

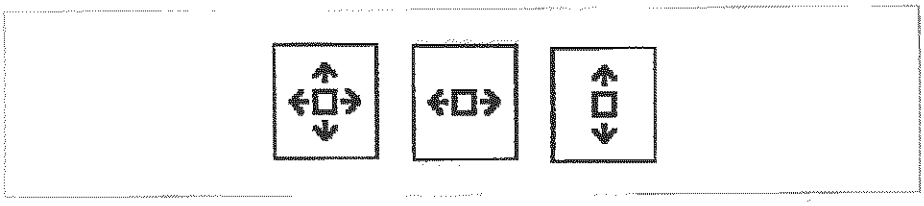


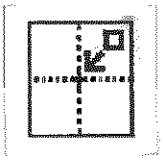
Déplacements d'objets

Si le bouton **Solo/Omni** est en mode **Solo**, un cliqué tiré sur un objet en modifie l'emplacement à l'intérieur de la fenêtre 2D sélectionnée ; sinon c'est l'ensemble des objets qui est déplacé.



L'icône **Direction du déplacement** donne l'accès à trois déplacements possibles accessibles par une succession de clics :





L'icône **Centrage** agit simultanément à l'intérieur de toutes les fenêtres sur l'ensemble des objets de l'univers quel que soit le mode **Solo/Omni** sélectionné. Le centrage s'effectue en se référant aux limites définies par le parallélépipède qui englobe la totalité des objets.

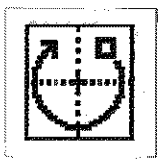
Copies d'objets

Le bouton **Solo/Omni** doit se trouver en mode Solo. Un Shift-cliqué-tiré sur un objet en crée une copie à un autre endroit de l'écran. La copie devient un objet à part entière qui doit être nommé comme l'y invite une boîte de saisie de nom.

Rotations

Lorsqu'une fenêtre 2D est active toute rotation demandée depuis le bouton **Rotat** de la boîte à outils influe réellement sur la position de l'objet dans l'univers. La sélection de la valeur de rotation s'effectue comme lorsqu'on travaille à l'aide de la caméra (ascenseurs, flèches de direction en combinaison avec **Shift** puis barre d'espace ou clic dans la fenêtre pour valider la modification). Cependant la valeur de la rotation est une valeur relative à la position courante de l'objet : cela signifie qu'à l'issue d'une rotation les valeurs **H** et **V** repassent automatiquement à zéro, prêtes pour une nouvelle rotation par rapport à la position actuelle (alors qu'avec la caméra il s'agissait de positionnements absolus : H=0 et V=0 correspondaient toujours à une vue de face au niveau du sol). Pour redonner à un objet son orientation précédente, il est donc nécessaire de lui appliquer une rotation inverse de valeur équivalente.

Les résultats obtenus diffèrent en fonction du point de pivot choisi par une succession de clics sur l'icône correspondant de la boîte à outils.



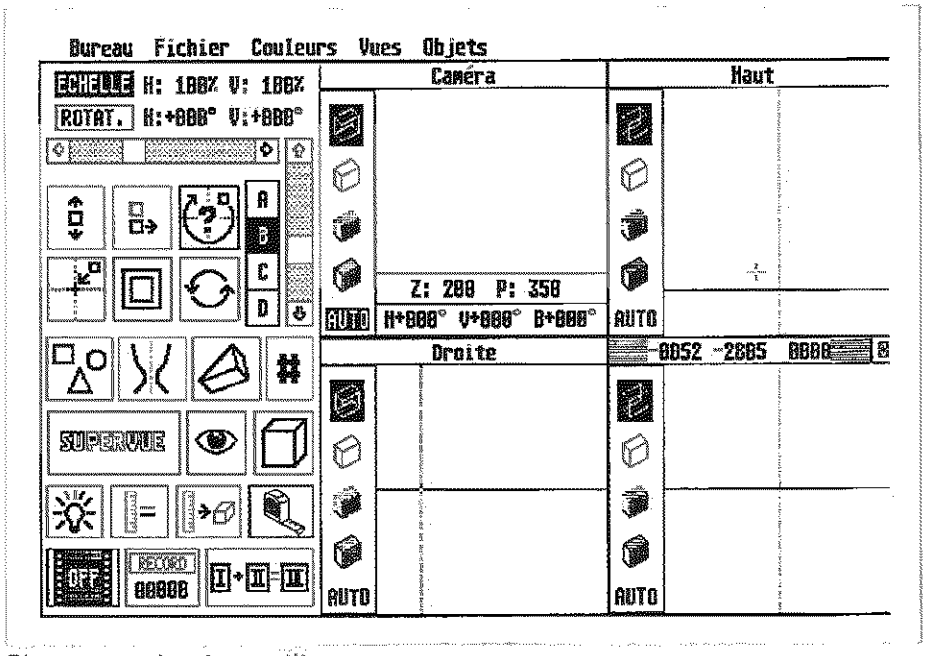
S'il s'agit du point central de l'univers, tous les objets visibles à l'écran pivotent par rapport au centre de l'univers.



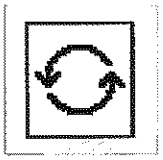
Dans le cas du pivot de groupe, la rotation s'effectue par rapport au centre du groupe des objets affichés dans la fenêtre (au centre de l'objet s'il n'y en a qu'un).



Le point de pivot arbitraire doit d'abord être placé par un double clic sur l'icône de sélection de point de pivot : un réticule apparaît dans les fenêtres et il suffit de le placer par un cliqué-tiré pour définir le point de pivot (dans l'espace, donc dans au moins deux fenêtre 2D) ; un clic-droit permet de sortir de mode positionnement du point de pivot.



Placement du pivot utilisateur.

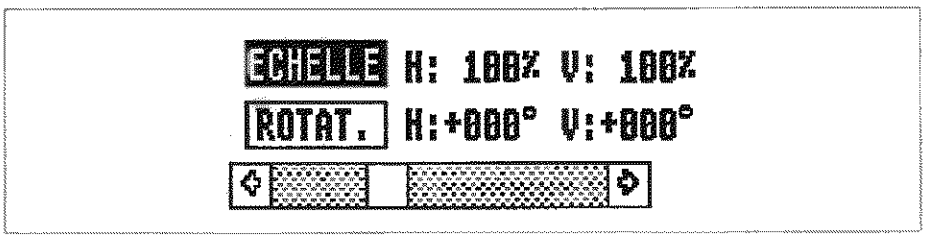


Un type de rotation différent d'horizontal ou de vertical est possible depuis l'icône **Rotation** qui ouvre une boîte de saisie semblable à celle qui permet d'incliner la caméra autour de son objectif.

Le principe est ici le même et la rotation s'effectue sur un plan parallèle à la fenêtre 2D active.

Déformations

La déformation d'un objet s'effectue depuis le bouton **Echelle** de la boîte à outils. Ce bouton n'apparaît que lorsqu'une fenêtre 2D est active.



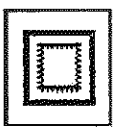
Modification des dimensions de l'objet.

On choisit la vue dans laquelle doit s'effectuer la déformation puis on agit soit sur l'échelle horizontale, soit sur l'échelle verticale, soit sur

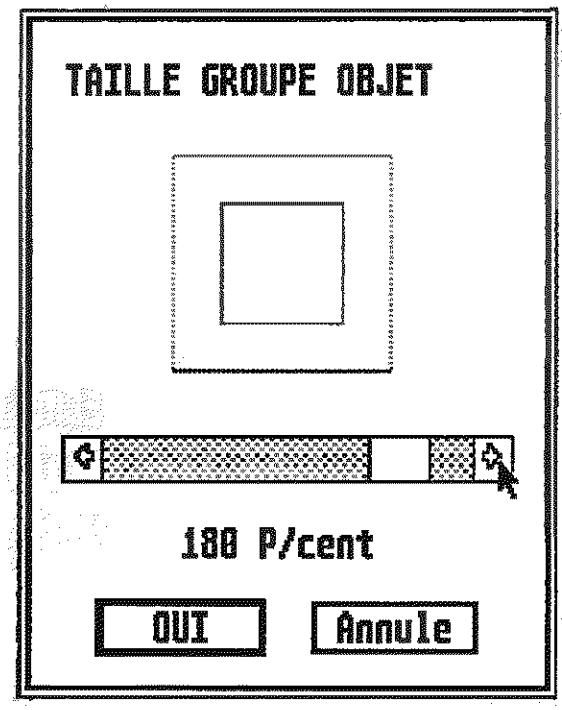
les deux simultanément pour modifier les proportions de l'objet et ce par l'intermédiaire des ascenseurs ou des touches fléchées du clavier :

- si c'est la vue face qui est active, on va modifier soit la longueur, soit la hauteur de l'objet ;
- si c'est la vue de haut, on intervient sur sa longueur ou son épaisseur ;
- dans le cas de la vue de droite on agit sur sa hauteur ou son épaisseur.

Si on souhaite réduire uniformément la taille d'un objet, il faut penser à travailler sur ses trois dimensions et donc utiliser au moins deux fenêtres (la hauteur et la largeur dans une et l'épaisseur dans une autre par exemple). On dispose également de l'icône **Taille** qui réduit tous les objets affichés de façon isométrique.

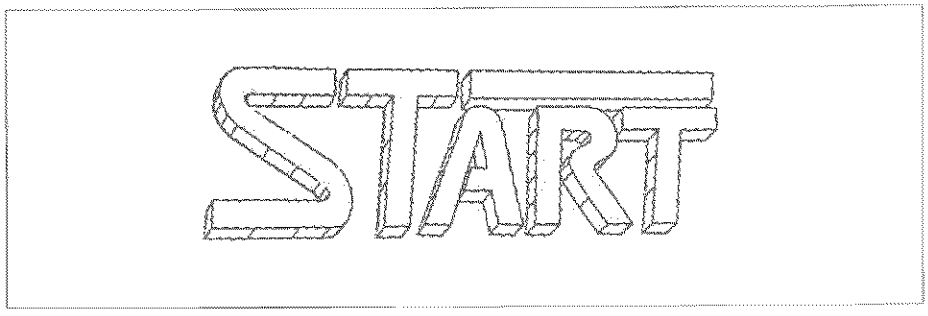


Cette icône ouvre une boîte de sélection qui visualise la modification en faisant varier la taille d'un carré pointillé alors qu'un carré en trait plein de taille fixe figure une échelle de 100 %. On peut appuyer sur **Shift** en même temps qu'on appuie sur les touches du tapis roulant afin de le déplacer par pas de 10.

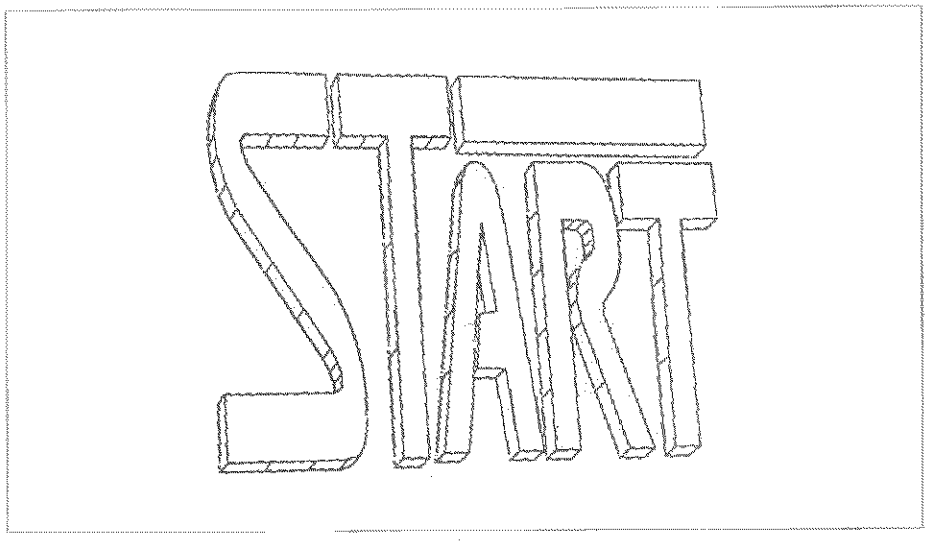


La modification d'échelle est toujours relative à la taille courante de l'objet. Après une modification, l'échelle redevient 100 %. Pour retrouver la taille initiale suite à une modification, il faut utiliser la valeur opposée (si on a agrandi à 200 %, il faut ensuite réduire à 50 %). Une réduction ou une augmentation de 100 % ne correspond bien sûr à aucune modification de l'objet qui conserve sa taille.

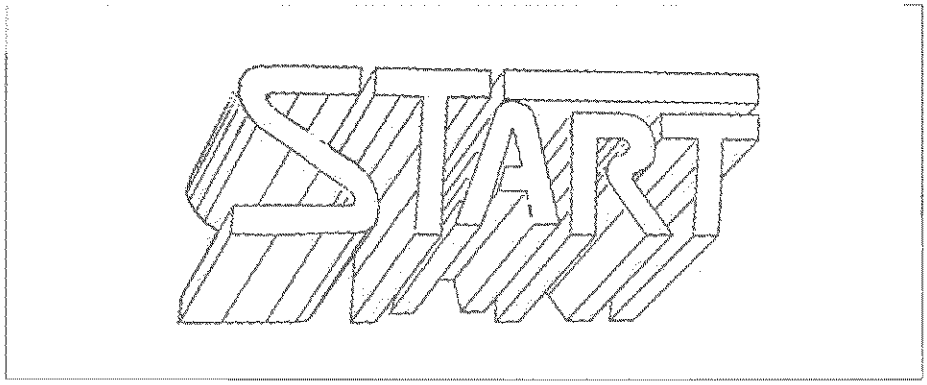
Les ascenseurs permettent de parcourir les valeurs de 50 % à 200 %. On peut aller au delà en faisant plusieurs agrandissements ou réductions successifs, à condition de ne pas sortir des limites de la fenêtre 2D.



Point de départ...



... étirement...

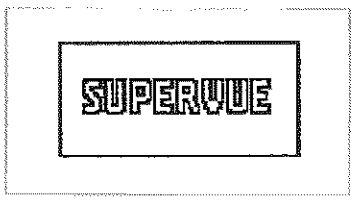


... ou épaississement

D'autres possiblités de modification de la taille des objets existent depuis les icônes **Mesures** qui sont détaillées par la suite.

IMPRESSION, SAUVEGARDE

La **Supervue**



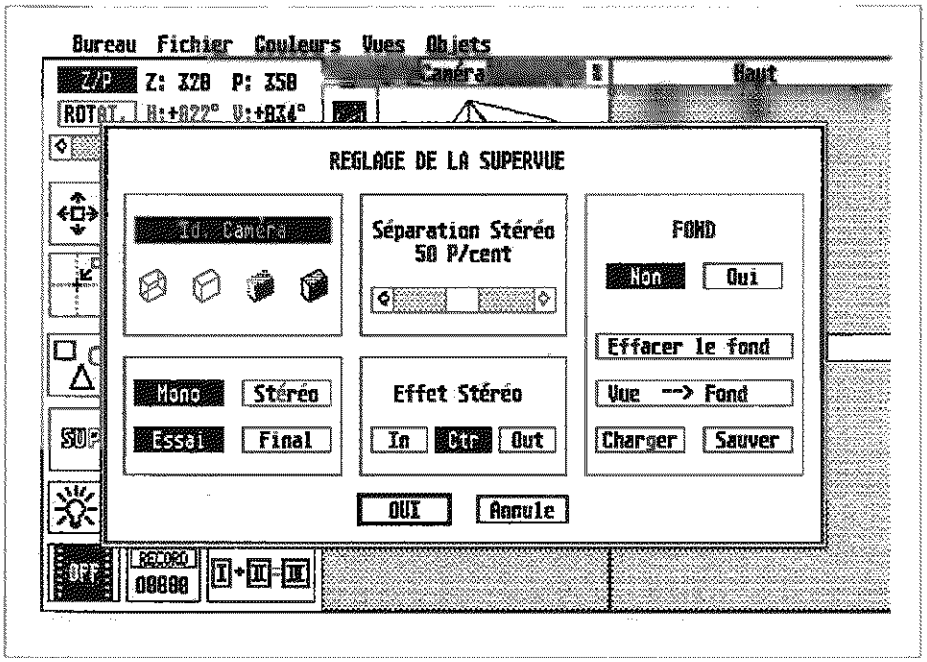
Cette représentation en plein écran de la fenêtre **Camera** est celle qui est utilisée par les programmes périphériques de CYBERSTUDIO ou qui est imprimée. Avec un écran couleur, bien que le logiciel travaille en moyenne résolution, la **Supervue** affiche la scène en basse résolution afin de profiter des seize couleurs de la palette.

Si aucune modification d'objet n'est possible en mode **Supervue**, certains mouvements de caméra sont autorisés grâce aux touches du clavier, à condition que la représentation soit faite dans le mode fil de fer.

La modification en temps réel des réglages de la caméra s'effectue par un cliqué tiré ou à l'aide des flèches du clavier éventuellement en combinaison avec **Shift**. Il faut auparavant taper sur la lettre correspondant aux options existantes :

- **R** : rotation horizontale ou verticale
- **B** : rotation autour de l'axe de la caméra
- **Z** : zoom
- **P** : perspective
- **Undo** : annule toute modification

Un double-clic sur l'icône **Supervue** ouvre la boîte de réglage suivante :



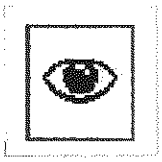
• Choix du type de représentation dans la **Supervue**.
Idem **Caméra** est actif par défaut.

• Mode **Final** : l'affichage requiert un algorithme plus sophistiqué qui évite les éventuelles erreurs de calcul des faces et des ombres en mode **Essai**. Le mode **Final** est particulièrement lent (l'affichage s'effectue ligne à ligne à l'écran) et il est conseillé de ne l'utiliser qu'au moment de l'impression ou de la sauvegarde.

• **Fond** : un fond au format bit-map peut être chargé depuis DEGAS (format PI1, couleur basse résolution ou PI3, monochrome). On obtient alors son affichage dans la **Supervue** en même temps que les objets définis (pratique pour effectuer des positionnements). On peut également figer la vue actuelle des objets 3D en un fond bit-map puis éventuellement sauvegarder ce dernier ; on fait de la sorte cohabiter une vue fil de fer figée en bit-map d'un objet avec la représentation en mode **Solide contour** d'un autre objet.

• Effets **Stereo** : sur Atari, à condition d'utiliser les lunettes STEREOTEK à cristaux liquides, il est possible de visualiser l'image d'une scène en relief avec la qualité qu'offrent les terminaux stéréoscopiques beaucoup plus chers (sans lunettes deux images décalées apparaissent très rapidement à tour de rôle sur l'écran). La **Separation** ajuste l'ampleur de l'**Effet** de profondeur alors que l'**Effet** positionne l'objet par rapport au moniteur :

- IN : dans le moniteur
- CTR : au niveau de l'écran
- Out : devant l'écran



A chaque entrée dans la **Supervue**, la présentation du dessin est entièrement recalculée. Si on n'a pas effectué de modification ou si on souhaite revoir l'avant dernière version, on retrouve le dernier contenu de la **Supervue** instantanément par l'icône **Voir**.



Nouveau

Vide la mémoire des objets mais réinitialise également la palette de couleurs et les réglages de la caméra. Si on souhaite conserver ces paramètres lors de la réalisation d'un nouveau dessin, on peut passer par le menu **Objet/Effacer**.

Charger

Cette option accepte également les fichiers créés avec la version 1.0 de CAD-3D.

Sauver tout

Sauvegarde même les objets qui ne sont pas sélectionnés ainsi que l'éclairage et les couleurs. Les gabarits ne sont par contre pas sauvegardés, ni la position de la caméra observant l'univers.

Sauver groupe

Sauve dans les mêmes conditions uniquement les objets sélectionnés et donc affichés à l'écran.



Ajouter

Charge de nouveaux éléments sans vider la mémoire de ce quelle contenait. On peut ainsi créer des bibliothèques d'éléments de base afin de les rappeler en cours de travail au fur et à mesure des besoins.

Liste des objets

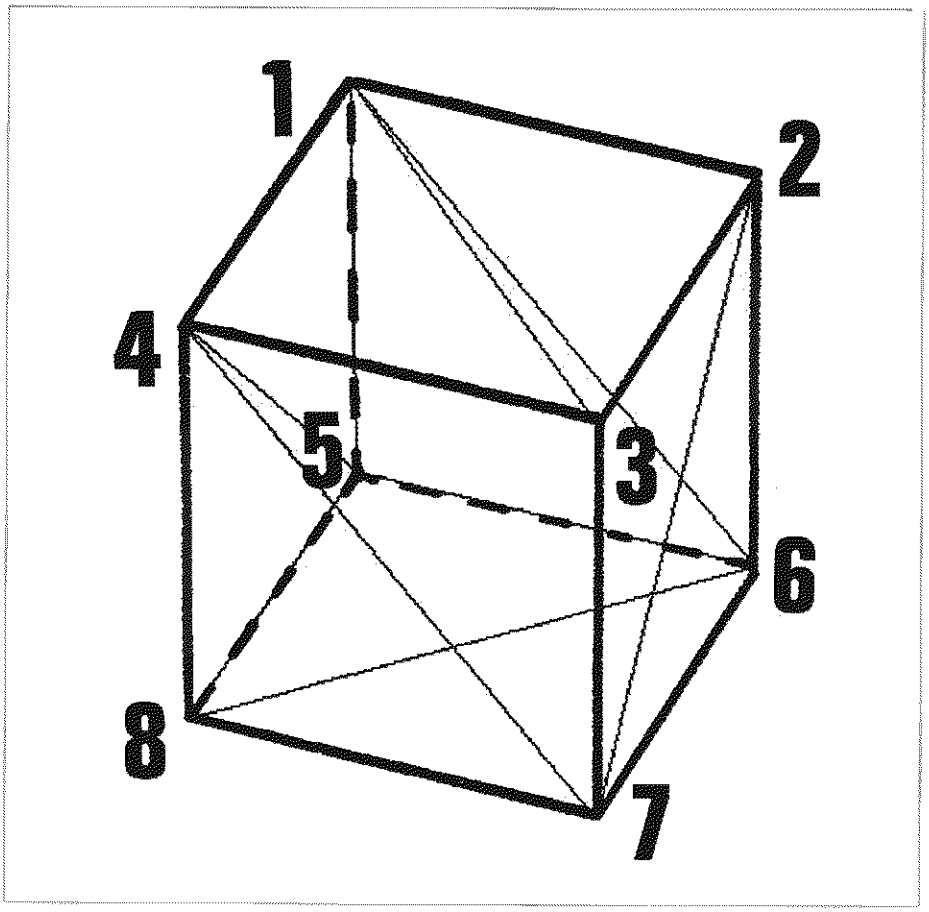
Sort à l'imprimante ou sur disque les coordonnées en X, Y et Z de chacun des sommets constitutifs des différents objets ainsi que la liste des différentes faces. Très utile à récupérer dans un programme utilisateur (pour, par exemple piloter une machine outil qui réalisera l'objet).

```
Nom d'objet: CUBE

Point#  ---X---   ---Y---  ---Z---
00001   -6.000    6.000     6.000
00002   6.000     6.000     6.000
00003   6.000     6.000     -6.000
00004   -6.000    6.000     -6.000
00005   -6.000    -6.000    6.000
00006   6.000     -6.000    6.000
00007   6.000     -6.000    -6.000
00008   -6.000    -6.000    -6.000

Face#    PointA   PointB   PointC
00001     00001    00005    00006
00002     00001    00006    00002
00003     00002    00006    00007
00004     00002    00007    00003
00005     00003    00007    00008
00006     00003    00008    00004
00007     00004    00008    00005
00008     00004    00005    00001
00009     00001    00002    00003
00010     00005    00007    00006
00011     00001    00003    00004
00012     00005    00008    00007
```

Dans cet exemple décrivant un cube, on obtient 12 faces (au lieu de 6) du fait de la décomposition des faces en triangles. Trois points suffisent en effet pour repérer une face : ainsi la face 1 est-elle repérée par les sommets 1, 5 et 6 ou 1,6 et 2.



Attention, la liste ou le fichier ainsi générés peuvent avoir une taille conséquente ; de plus aucune option ne permet d'annuler le déroulement de ce menu une fois qu'il a été choisi.

Type imprimante

Permet de choisir un des pilotes fournis sur la disquette de DEGAS Elite pour imprimer les dessins de la **Supervue** sans passer par une recopie écran. Il est dommage que les pilotes ne soient pas fournis avec CYBERSTUDIO.

Imprimer image

N'est actif que si un pilote DEGAS a préalablement été chargé. Deux options d'impressions sont alors proposées :
– **Trav** qui devrait permettre d'imprimer la page de travail mais qui ne nous a donné que des pages entièrement noires.
– **Super** qui imprime l'univers en mode Supervue.

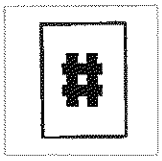
Sauver image

Sauvegarde l'écran de la **Supervue** dans le format sélectionné : DEGAS (haute résolution monochrome ou basse résolution couleur), COLR ou Néochrome, ces deux dernières options n'étant actives qu'avec un écran couleur.

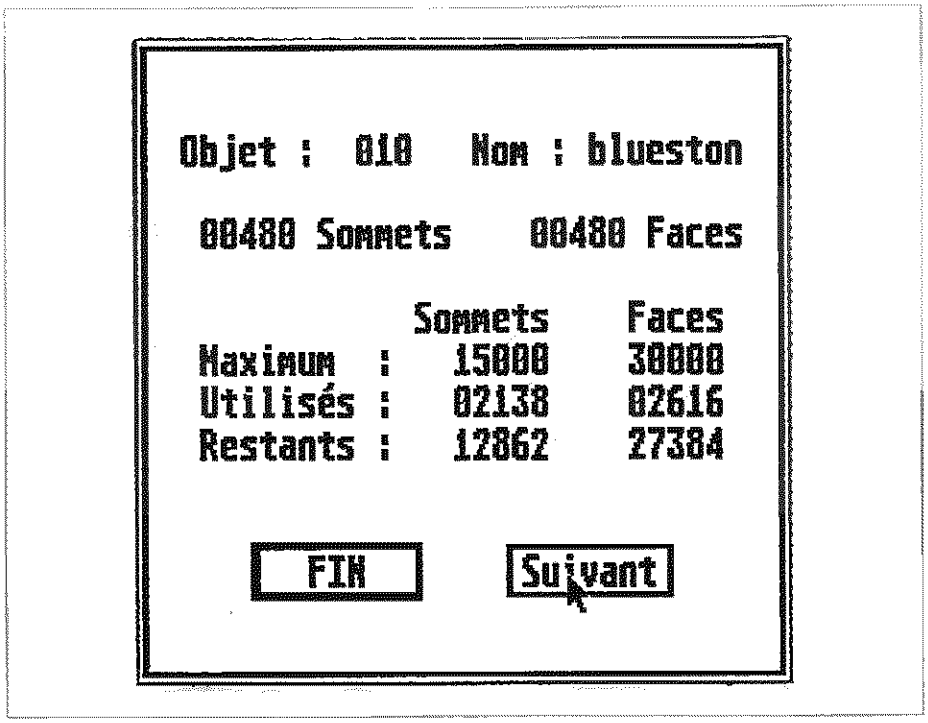
OPTIONS DIVERSES

Statistiques

CAD-3D 2.0 permet de créer 80 objets différents et chacun de ces objets peut lui-même être la réunion de plusieurs éléments de sorte qu'il n'existe pratiquement pas de limite à la création si ce n'est la capacité mémoire de l'ordinateur utilisé.

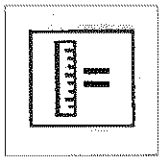


La boîte de données qui s'affiche lorsqu'on sélectionne cette option indique le nombre de sommets et de faces utilisés par chacun des objets ainsi que la place disponible en fonction de la configuration mémoire.



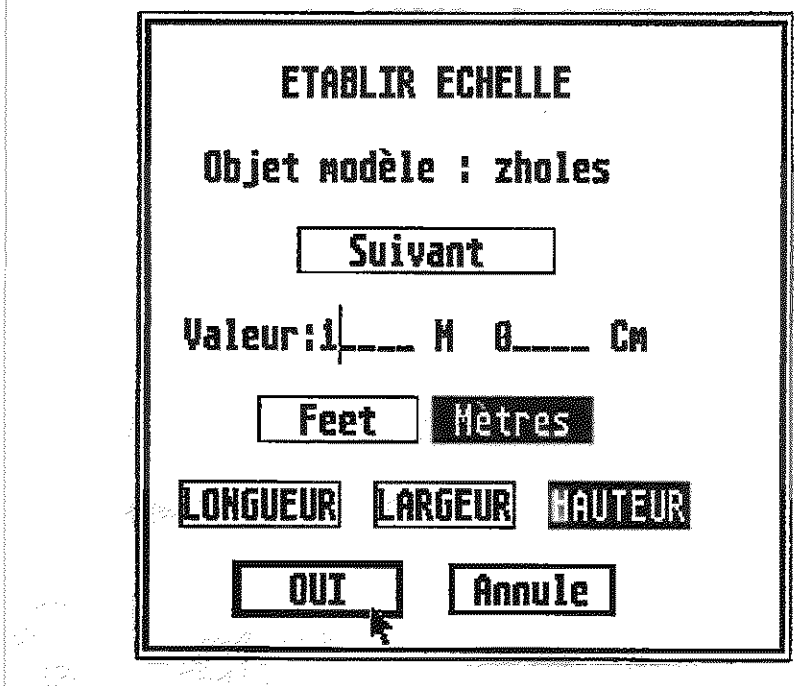
Mesures

Etablir l'échelle



Avant de pouvoir effectuer des mesures sous CAD-3D, on attribue une mesure précise (si on la connaît) ou quelconque à un objet qui

servira de référence pour la suite du travail et ce grâce à l'icône **Etablir l'échelle**. La boîte de saisie qui s'ouvre alors demande d'entrer la mesure d'une des dimensions de l'objet soit en pouces, soit en centimètres.

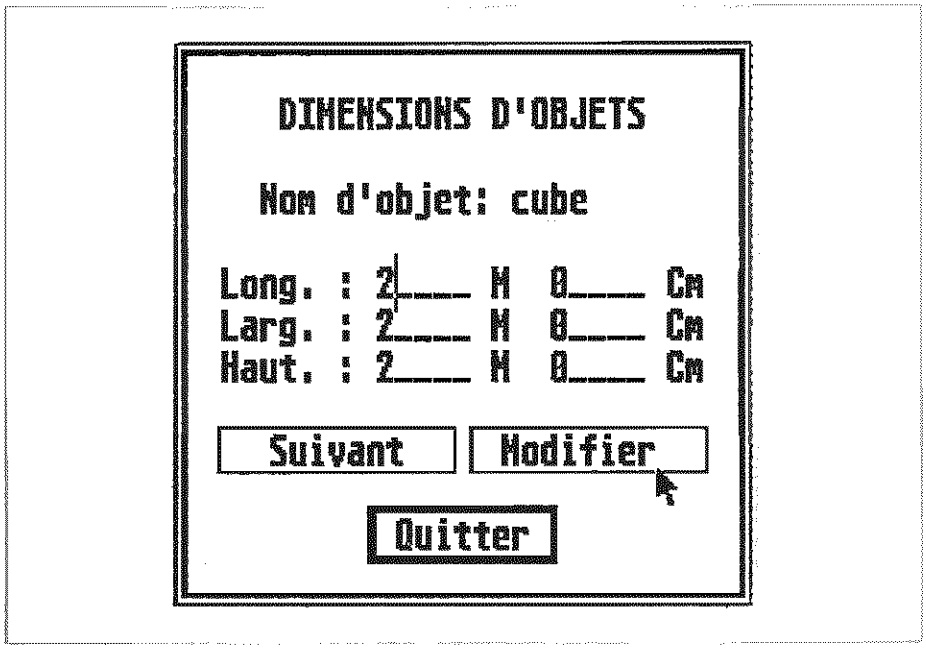


Toutes les mesures qui s'effectueront par la suite se feront relativement à cet objet de référence (si l'objet est défini avec une hauteur de 1 m., un objet deux fois et demi plus haut aura donc une mesure de 2,5 m.).

Dimensionner un objet

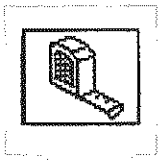


La boîte de saisie qui apparaît lorsqu'on active cette icône demande d'entrer les dimensions du parallélépipède rectangle à l'intérieur duquel s'inscrit l'objet.

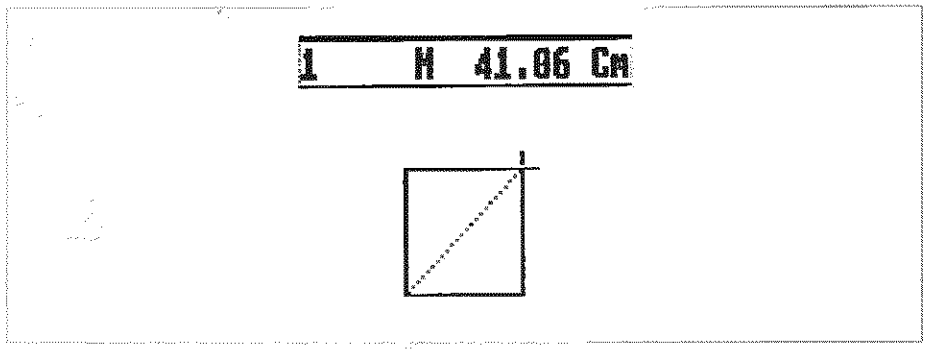


Modifier la taille d'un objet.

Mesurer



Lorsqu'on choisit cette option, le curseur se transforme en une petite croix. On peut alors cliquer-tirer sur n'importe quel objet de l'écran pour en prendre la mesure comme on le ferait à l'aide d'un mètre à ruban : on clique sur un sommet de l'objet puis on tire jusqu'à l'autre sommet ; la mesure s'affiche dans le bandeau de la fenêtre active. Seules les mesures effectuées sur les fenêtres 2D ont une signification.



Bonne précision de mesure 1,4106 au lieu de 1,414 pour la diagonale du carré de 1 m de côté.

Couleurs

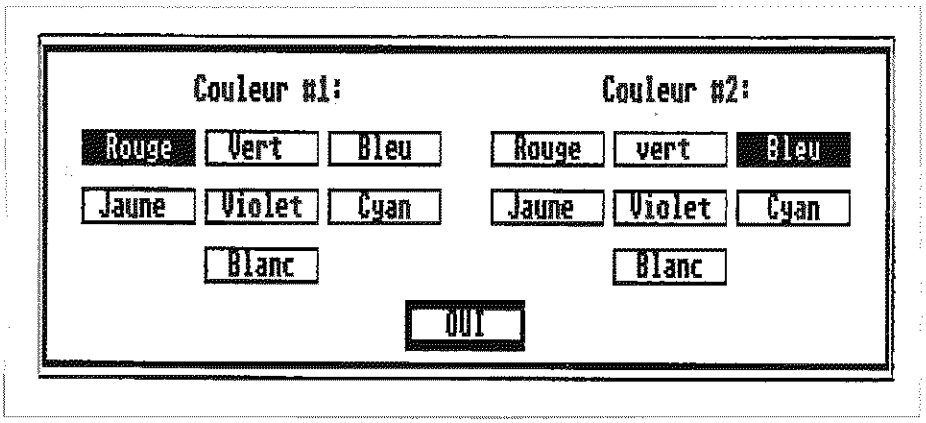
Couleurs
Installer
7 nuances 14 nuances Préférences
Fond blanc Fond noir
Recolorer

Sur un écran monochrome, le menu **Recolorer** définit le nombre de trames qui peuvent figurer les ombres sur les faces des objets. Il suffit de cliquer sur une trame pour que le pointeur en forme de < se positionne devant et la marque comme la nuance la plus claire pouvant être affichée ; les trames situées au dessus du pointeur ne seront pas utilisées.

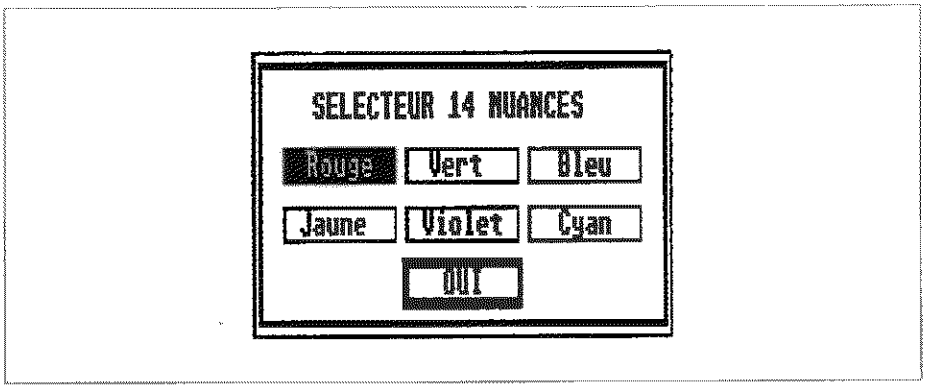
Sur un écran couleur les objets s'affichent en mode **Supervue** uniquement :

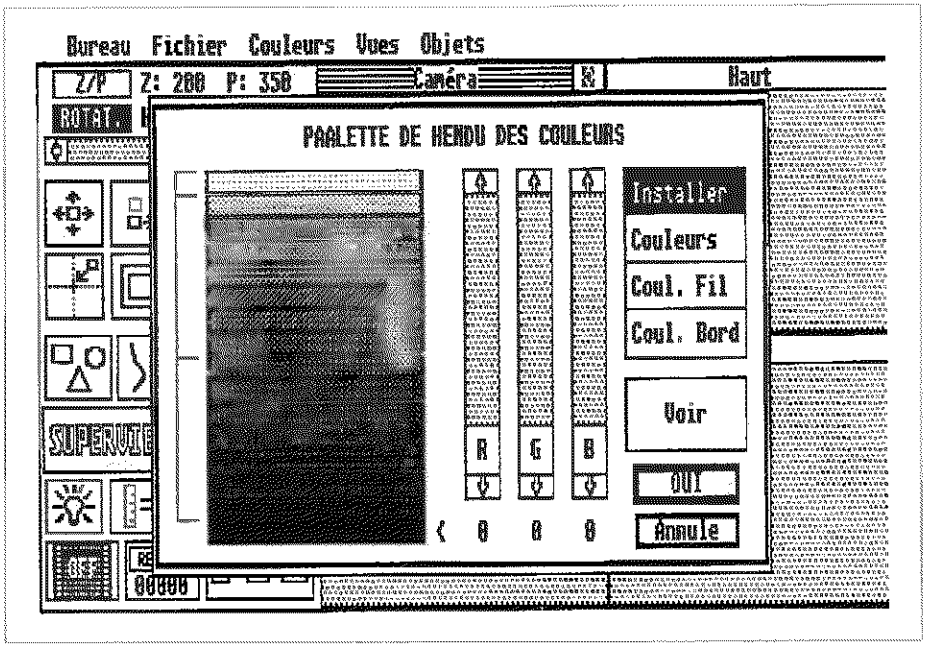
– soit en deux couleurs réparties chacune en 7 nuances ;
– soit en une seule couleur étagée en 14 nuances ;
– soit en un libre choix de couleurs définies depuis le menu **Préférences**.

En fonction de ce choix, le menu **Installer** propose des options différentes :



Sélecteur deux couleurs et sept nuances.





Répartition des couleurs depuis le menu Installer en mode Préférences.

La boîte de réglage **Palette de rendu des couleurs** qui s'ouvre lorsque le mode **Préférences** est actif permet de choisir 16 couleurs parmi les 512 teintes possibles sur l'Atari. Lorsque **Installer** est actif, chaque couleur peut être ajustée individuellement à l'aide des curseurs de dosage. Des dégradés ou des fondus se réalisent automatiquement si, après avoir défini une couleur d'arrivée, on appuie sur la touche **Alternate** tout en cliquant sur la couleur de départ. **Shift** cliqué-tiré copie une couleur à un autre endroit de la palette.

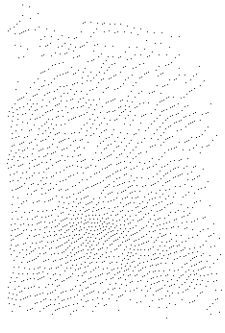
Couleur positionne les crochets à gauche de la palette par un cliqué tiré de la couleur de départ à la couleur d'arrivée : ces crochets définissent les différentes teintes que peut prendre un objet que l'on crée ou que l'on recolore. Ainsi, depuis le menu **Recolorer** on peut choisir la

palette de nuances à associer à un objet en cliquant sur la nuance située le plus près ; si on clique sur une couleur située ailleurs dans le crochet, l'objet pourra prendre uniquement les teintes comprises entre le pointeur < que l'on vient de placer et le bas du crochet et il apparaîtra moins contrasté.

Coul Fil Définit la couleur des arêtes en mode **Fil de fer**.

Coul Bord Définit la couleur des arêtes en mode **Solide contour**.

Voir Affiche l'écran **Supervue** le temps d'un cliqué-tenu.





5/5

Les programmes de graphisme

5/5.1

La loupe Weka

Voici un exemple du type d'outil graphique que l'on rencontre dans les programmes de dessin.

```
;                                Zoom WEKA
;
;
;              Copyright 1989 Editions WEKA et R. LEMOINE
;

 output .prg

 dc.w $a000                                  ; adresse des variables line_a
 sub.l #602,d0                               ; adresse des coordonnées
 move.l d0,mouse_addr                ; souris dans mouse_addr
 bsr constr_table                            ; construction de la table
                                                    ; du zoom x2
 clr.l -(sp)                                 ; passage en super
 move.w #$20,-(sp)
 trap #1
 addq.w #6,sp
 move.l d0,d6
 move.w sr,d7                                ; sauvegarde du sr dans d7
 move.w #$2700,sr                    ; blocage des interruptions
 lea fin_vbl(pc),a0                  ; et mise en place du zoom
 move.l $70\w,2(a0)                  ; sur la vbl
 move.l #debut_vbl,$70\w
 move.w d7,sr                                ; restauration des interruptions
 move.w #7,-(sp)                             ; attente d'appui sur une touche
 trap #1
 addq.w #2,sp
 move.w #$2700,sr                            ; blocage des interruptions
 lea fin_vbl(pc),a0                  ; remise en place de l'ancienne
 move.l 2(a0),$70\w                  ; routine vbl
 move.w d7,sr                                ; déblocage des interruptions
 move.l d6,-(sp)                             ; passage en user
 move.w #$20,-(sp)
 trap #1
 addq.w #6,sp
 clr.w -(sp)                                 ; et sortie au bureau GEM
 trap #1

debut_vbl:                                   ; interruption vbl
 movem.l d0-d5/a0-a3,-(sp)           ; sauvegarde des registres
 btst #0,$465\w                              ; compteur vbl pair?
 bne .fin_zoom                               ; si non, pas de zoom
 move.w #40,reticule_x               ; init. des variables
 move.w #40,reticule_y               ; du réticule
 move.l $44e\w,a0                            ; adresse de l'image à zoomer
```



(suite)

```
                                                    ; vous pouvez mettre ici
                                                    ; l'adresse de votre image
  lea tampon(pc),a1                ; tampon de construction du
                                                    ; zoom
  move.l mouse_addr,a2
  move.w 2(a2),d0                                   ; coord. y de la souris
  sub.w #40,d0                                      ; inférieure à 40?
  bpl.s .suite1                                     ; si oui, le réticule monte
  add.w d0,reticule_y
  clr.w d0                                          ; et zoom à partir du haut
                                                    ; de l'écran
.suite1:
  ext.l d0
  cmp.w #320,d0                                     ; sup à 360? (marqué 320 car
                                                    ; on a fait -40 avant)
  blt.s .suite3
  sub.w #320,d0                                     ; si oui, le réticule descend
  add.w d0,reticule_y
  move.w #320,d0
.suite3:
  mulu #$50,d0                                      ; adresse de la ligne écran
  add.w d0,a0                                       ; + adresse de base de l'image
  move.w (a2),d0                                    ; coord. x de la souris
  sub.w #40,d0                                      ; meme principe que pour y
  bpl.s .suite2
  add.w d0,reticule_x
  clr.w d0
.suite2:
  cmp.w #560,d0
  blt.s .suite4
  sub.w #560,d0
  add.w d0,reticule_x
  move.w #560,d0
.suite4:
  move.w d0,d1
  lsr.w #3,d0                                       ; recherche du premier bit à
                                                    ; afficher
  and.w #$7,d1
  add.w d0,a0
  move.w #79,d5                                     ; copie de la portion à zoomer
.boucle1:                                           ; dans tampon en la décalant
  move.l a0,a2                                      ; au début d'octet
  rept 10
  move.b 1(a2),d3
```

(suite)

```
    lsl.w #8,d3
    move.b (a2)+,d3
    rol.w d1,d3
    move.b d3,(a1)+
    endr                                        ; 1 ligne
    add.w #$50,a0                               ; ligne suivante
    dbra d5,.boucle1                            ; 80 lignes
    lea tampon(pc),a1                  ; dessin des deux traits
    moveq #0,d0                                 ; du réticule dans le ampon
    move.w reticule_y,d0
    mulu #10,d0
    move.l #-1,(a1,d0.l)               ; trait horizontal
    move.l #-1,4(a1,d0.l)
    move.w #-1,8(a1,d0.l)
    move.w reticule_x,d0
    move.w d0,d1
    lsr.w #3,d0
    add.w d0,a1
    not.b d1
    and.b #$7,d1
    rept 80
    bset d1,(a1)                                ; trait vertical
    add.w #10,a1
    endr
    lea tampon(pc),a0                  ; zoom de tampon dans tampon_2
    lea tampon_2(pc),a1                ; en utilisant la table
    lea table(pc),a2                            ; calculée à l'initialisation
    move.w #799,d1
.boucle2:
    clr.w d0                                    ; zoom de la portion stockée dans
                                                ; tampon
    move.b (a0)+,d0                             ; l'image zoomée est stockée
    lsl.w #1,d0                                 ; dans tampon_2
    move.w (a2,d0.w),(a1)+
    dbra d1,.boucle2
    lea tampon_2(pc),a0
    move.l $44e,a1                              ; et affichage de tampon_2
    rept 80                                     ; à l'écran
    movem.l (a0)+,d0-d4
    movem.l d0-d4,(a1)
    add.w #80,a1
    movem.l d0-d4,(a1)
    add.w #80,a1
    endr
```



(suite)

```
.fin_zoom:
 movem.l (sp)+,d0-d5/a0-a3          ; récupération des registres
fin_vbl:
 jmp $0                                        ; et saut à la vbl en ROM

constr_table:                                  ; table du zoom x2
 lea table(pc),a1                              ; elle contient les 256
 clr.w d4                                      ; valeurs possibles d'un octet
 move.w #$ff,d0                                ; avec un zoom:
.boucle3:                                      ; ex: 01011001  est codé
 move.w d4,d2                                  ; par 0011001111000011
 move.w #$f,d1                                 ; Cette table fait donc
 move.w #3,d3                                  ; 256*2=512 octets
.boucle1:                                      ; elle est crée une bonne
 lsl.b #1,d2                                   ; fois pour toutes à
 bcc.s .suite1                                 ; l'initialisation du zoom
 bset d1,(a1)
 subq.w #1,d1
 bset d1,(a1)
 addq.w #1,d1
.suite1:
 subq.w #2,d1
 dbra d3,.boucle1
 addq.w #1,a1
 move.w #7,d1
 move.w #3,d3
.boucle2:
 lsl.b #1,d2
 bcc.s .suite2
 bset d1,(a1)
 subq.w #1,d1
 bset d1,(a1)
 addq.w #1,d1
.suite2:
 subq.w #2,d1
 dbra d3,.boucle2
 addq.w #1,a1
 addq.w #1,d4
 dbra d0,.boucle3
 rts


mouse_addr:
 ds.l 1
tampon:
```

(suite)

```
 ds.l 200
tampon_2:
 ds.l 800
reticule_x:
 ds.w 1
reticule_y:
 ds.w 1
table:
 ds.b 512
```

5/5.2

Routines compactages/ décompactages de sauvegarde d'écran en format DEGAS.PC3 pour programmes en GFA-Basic

Si les routines de sauvegarde d'écran au format PC3 sont largement répandues, elles présentent toutes un inconvénient majeur : la lenteur. Les routines que nous proposons ici vous étonneront par leur rapidité d'exécution ; pour exemple, il faut moins d'une seconde pour effectuer un décompactage.

Il n'y a pas de secret, seul le langage machine est capable d'une telle prouesse. Nous avons donc *incrusté* sous forme de Datas les deux routines dont l'appel se fera par les instructions prévues par le Basic GFA soit : **CALL X% & Void C:x%( )**.

Voici donc les listings en Basic et assembleur qui vous permettront éventuellement d'adapter ces utilitaires à tous les interpréteurs disponibles à ce jour.

LISTING GFA DU COMPACTAGE DEGAS.PC3 (HAUTE RÉSOLUTION)

```
DATA &H48E7,&HFFFE,&H3F3C,&H0000,&H487A,&H00E8,&H3F3C,&H003C,&H4E41
DATA &H508F,&H4A40,&H6B00,&H00D4,&H3C00,&H3F3C,&H0002,&H4E4E,&H548F
DATA &H2040,&H43FA,&H00DA,&H2649,&H22FC,&H8002,&H0777,&HD3FC,&H0000
DATA &H001E,&H4284,&H4283,&H0C03,&H004F,&H6D08,&H4219,&H12D8,&H4243
DATA &H5244,&H6100,&H006C,&H675E,&H1010,&HB028,&H0001,&H672C,&H72FF
DATA &H2449,&H5289,&H5201,&H12C0,&H0C03,&H004E,&H670E,&H5288,&H5203
DATA &H1010,&HB028,&H0001,&H66EA,&H600A,&H5288,&H12D8,&H5201,&H4203
DATA &H5244,&H1481,&H60B8,&H7200,&H1340,&H0001,&H5201,&H5288,&H5203
DATA &H0C03,&H004F,&H6708,&HB028,&H0001,&H67EE,&H6004,&H76FF,&H5244
DATA &H5288,&H5203,&H4401,&H1281,&H5489,&H608E,&HD3FC,&H0000,&H0020
DATA &H6102,&H602A,&H2009,&H908B,&H6106,&H0C44,&H0190,&H4E75,&H48E7
DATA &HFFFE,&H2F0B,&H2F00,&H3F06,&H3F3C,&H0040,&H4E41,&HDFFC,&H0000
DATA &H000C,&H4CDF,&H7FFF,&H224B,&H4E75,&H3F06,&H3F3C,&H003E,&H4E41
DATA &H588F,&H4CDF,&H7FFF,&H4E75
'
'
DATA FIN



' SAISIE DES DATAS


While Dat$<>"FIN"
Read Dat$
If Dat$<>"FIN"
  Let Compdeg$=Compdeg$+Mki$(Val(Dat$))
Endif
Wend
Clr Dat$



' SELECTION NOM DU FICHIER RECEVEUR


Fileselect"A:\*.PC3","",A$
A$=Left$(A$+String$(16,0),16)
Comp$=Compdeg$+A$+String$(84,0)

' APPEL DE LA FONCTION :
C%=Varptr(Comp$)
Call C%

Clr Comp$,A$,C%
```



LISTING GFA DU DÉCOMPACTAGE DEGAS.PC3 (HAUTE RÉSOLUTION)

```
Data &H3F3C,&H0003,&H4E4E,&H548F,&H2040,&H2440,&H4280,&HD5FC,&H0000
Data &H7CFF,&H226F,&H0004,&HD3FC,&H0000,&H0022,&H1019,&H0C00,&H0080
Data &H6208,&H10D9,&H51C8,&HFFFC,&H600A,&H4400,&H1219,&H10C1,&H51C8
Data &HFFFC,&H4280,&HB1CA,&H6BE0,&H4E75
'
'
Data FIN



' SAISIE DES DATAS


While Dat$<>"FIN"
  Read Dat$
  If Dat$<>"FIN"
    Let Decompac$=Decompac$+Mki$(Val(Dat$))
  Endif
Wend

Clr Dat$

' SELECTION NOM DU FICHIER COMPACTé


Fileselect "A:\*.PC3","",A$

If Exist(A$)                    '( on ne sait jamais...)
  Open "I",#1,A$
  B$=Input$(Lof(#1),#1)
  Close #1

  ' APPEL DE LA FONCTION :

  D%=Varptr(Decompac$)
  Void C:D%(L:Varptr(B$))
  Clr B$,A$
Endif


'  Aprés avoir chargé la variable "B$" par le fichier .PC3, on
détermine l'adresse de la routine de décompactage par un habituel
Varptr(?$). On en effectue ensuite l'appel par Void C: non sans y
avoir chargé comme paramètre l'adresse de B$.
```

```
**********************************************************************
*                   ROUTINE ASSEMBLEUR  68000                        *
*                     ----------------------                         *
*                     Auteur : GUISIANO JM                           *
*                       COMPACTAGE *.PC3                             *
**********************************************************************

DEBUT ROUTINE :00000000

00000000 48E7FFFE          MOVEM.L D0-7/A0-6,-(A7)
00000004 3F3C0000          MOVE.W  #$0000,-(A7)      Ouvre Fichier PC3
00000008 487A00E8          PEA     $000000F2(PC)
0000000C 3F3C003C          MOVE.W  #$003C,-(A7)
00000010 4E41              TRAP    #$01
00000012 508F              ADDQ.L  #8,A7
00000014 4A40              TST.W   D0                Exit si non ouvert
00000016 6B0000D4          BMI     $000000EC
0000001A 3C00              MOVE.W  D0,D6

0000001C 3F3C0002          MOVE.W  #$0002,-(A7)      Adresse de la mém.
00000020 4E4E              TRAP    #$0E              d'écran.
00000022 548F              ADDQ.L  #2,A7
00000024 2040              MOVEA.L D0,A0

00000026 43FA00DA          LEA     $00000102(PC),A1  Adresse du Buffer
0000002A 2649              MOVEA.L A1,A3             de travail.
0000002C 22FC80020777      MOVE.L  #$80020777,(A1)+  Entête typique de
00000032 D3FC0000001E      ADDA.L  #$0000001E,A1     Degas.
00000038 4284              CLR.L   D4
0000003A 4283              CLR.L   D3

0000003C 0C03004F          CMPI.B  #$4F,D3           BOUCLE.
00000040 6D08              BLT.S   $0000004A
00000042 4219              CLR.B   (A1)+
00000044 12D8              MOVE.B  (A0)+,(A1)+
00000046 4243              CLR.W   D3
00000048 5244              ADDQ.W  #1,D4
0000004A 6100006C          BSR     $000000B8         Gosub calcul nbre
0000004E 675E              BEQ.S   $000000AE         Exit si écran
00000050 1010              MOVE.B  (A0),D0           compacté.
00000052 B0280001          CMP.B   $0001(A0),D0
00000056 672C              BEQ.S   $00000084
00000058 72FF              MOVEQ   #-$01,D1
0000005A 2449              MOVEA.L A1,A2
0000005C 5289              ADDQ.L  #1,A1
```



(suite)

```
0000005E 5201            ADDQ.B  #1,D1           compactage des
00000060 12C0            MOVE.B  D0,(A1)+         différents.
00000062 0C03004E        CMPI.B  #$4E,D3
00000066 670E            BEQ.S   $00000076
00000068 5288            ADDQ.L  #1,A0
0000006A 5203            ADDQ.B  #1,D3
0000006C 1010            MOVE.B  (A0),D0
0000006E B0280001        CMP.B   $0001(A0),D0
00000072 66EA            BNE.S   $0000005E
00000074 600A            BRA.S   $00000080

00000076 5288            ADDQ.L  #1,A0           stockage des dif.
00000078 12D8            MOVE.B  (A0)+,(A1)+
0000007A 5201            ADDQ.B  #1,D1
0000007C 4203            CLR.B   D3
0000007E 5244            ADDQ.W  #1,D4
00000080 1481            MOVE.B  D1,(A2)
00000082 60B8            BRA.S   $0000003C       GOTO BOUCLE

00000084 7200            MOVEQ   #$00,D1         Compactage de
00000086 13400001        MOVE.B  D0,$0001(A1)    égalités.
0000008A 5201            ADDQ.B  #1,D1
0000008C 5288            ADDQ.L  #1,A0
0000008E 5203            ADDQ.B  #1,D3
00000090 0C03004F        CMPI.B  #$4F,D3
00000094 6708            BEQ.S   $0000009E
00000096 B0280001        CMP.B   $0001(A0),D0
0000009A 67EE            BEQ.S   $0000008A
0000009C 6004            BRA.S   $000000A2

0000009E 76FF            MOVEQ   #-$01,D3        stockage des ég.
000000A0 5244            ADDQ.W  #1,D4
000000A2 5288            ADDQ.L  #1,A0
000000A4 5203            ADDQ.B  #1,D3
000000A6 4401            NEG.B   D1
000000A8 1281            MOVE.B  D1,(A1)
000000AA 5489            ADDQ.L  #2,A1

000000AC 608E            BRA.S   $0000003C       GOTO BOUCLE

000000AE D3FC00000020    ADDA.L  #$00000020,A1   +32 Octets
                                                 typique Degas

000000B4 6102            BSR.S   $000000B8       Gosub calcul nbre
000000B6 602A            BRA.S   $000000E2       Goto close
```

(suite)

```
000000B8 2009              MOVE.L  A1,D0              Procédure calcul
000000BA 908B              SUB.L   A3,D0              nbre.
000000BC 6106              BSR.S   $000000C4          Gosub écrit
000000BE 0C440190          CMPI.W  #$0190,D4          Fin Compactage?
000000C2 4E75              RTS

000000C4 48E7FFFE          MOVEM.L D0-7/A0-6,-(A7)    Procédure écrit
000000C8 2F0B              MOVE.L  A3,-(A7)
000000CA 2F00              MOVE.L  D0,-(A7)
000000CC 3F06              MOVE.W  D6,-(A7)
000000CE 3F3C0040          MOVE.W  #$0040,-(A7)
000000D2 4E41              TRAP    #$01
000000D4 DFFC0000000C      ADDA.L  #$0000000C,A7
000000DA 4CDF7FFF          MOVEM.L (A7)+,D0-7/A0-6
000000DE 224B              MOVEA.L A3,A1
000000E0 4E75              RTS

000000E2 3F06              MOVE.W  D6,-(A7)           Procédure close
000000E4 3F3C003E          MOVE.W  #$003E,-(A7)
000000E8 4E41              TRAP    #$01
000000EA 588F              ADDQ.L  #4,A7
000000EC 4CDF7FFF          MOVEM.L (A7)+,D0-7/A0-6
000000F0 4E75              RTS
000000F2 dc.b 16X          NOM DU FICHIER
00000102 dc.b 84X          BUFFER
```



```
**************************************************************
*                  ROUTINE ASSEMBLEUR  68000                 *
*                  ------------------------                  *
*                    Auteur : GUISIANO JM                    *
*                    DECOMPACTAGE *.PC3                      *
**************************************************************


DEBUT ROUTINE :00000000


00000000 3F3C0003          MOVE.W   #$0003,-(A7)     On détermine
00000004 4E4E              TRAP     #$0E             l'adresse de la
00000006 548F              ADDQ.L   #2,A7            mémoire d'écran
00000008 2040              MOVEA.L  D0,A0            A0 = pointeur
0000000A 2440              MOVEA.L  D0,A2
0000000C 4280              CLR.L    D0               Calcul de l'add
0000000E D5FC00007CFF      ADDA.L   #$00007CFF,A2     de la fin de
00000014 226F0004          MOVEA.L  $0004(A7),A1      la mém. d'écran.
00000018 D3FC00000022      ADDA.L   #$00000022,A1     A1 = Adresse du
                                                      fichier à décom-
                                                      pacter.
0000001E 1019              MOVE.B   (A1)+,D0
00000020 0C000080          CMPI.B   #$80,D0
00000024 6208              BHI.S    $0000002E

00000026 10D9              MOVE.B   (A1)+,(A0)+      Affichage des
00000028 51C8FFFC          DBF      D0,$00000026     différents
0000002C 600A              BRA.S    $00000038

0000002E 4400              NEG.B    D0
00000030 1219              MOVE.B   (A1)+,D1
00000032 10C1              MOVE.B   D1,(A0)+         Affichage des
00000034 51C8FFFC          DBF      D0,$00000032     égalités.
00000038 4280              CLR.L    D0

0000003A B1CA              CMPA.L   A2,A0              Compare l'add du
0000003C 6BE0              BMI.S    $0000001E        pointeur avec l'add
0000003E 4E75              RTS                         de fin d'écran.
```



5/5.2.1

Décompactage des images DEGAS toutes résolutions

DEGAS et DEGAS-ELITE étant les logiciels de création graphique les plus utilisés, il est intéressant d'utiliser leurs services pour créer des images dans des programmes et les importer par la suite.

Malheureusement les quelques 32034 ou 32066 octets que représentent une image écran codée par les logiciels précités occupent une place prépondérante en mémoire et pour peu que vous en utilisiez plusieurs, vous risquez de vous retrouver rapidement avec des problèmes de mémoire. La solution semble être le compactage que vous offrent les deux logiciels. Pour pouvoir ensuite utiliser ces images compactées dans un traitement personnel à l'intérieur d'un programme, il vous faudra les décompacter avant utilisation pour n'utiliser les 32000 octets de l'écran qu'au bon moment. Voyons d'abord la structure des fichiers DEGAS et DEGAS-ELITE.

Format non compressé	DEGAS	DEGAS ELITE	
2 octets	000x	000x	x = Résolution image 0 basse, 1 moyenne 2 haute
32 octets	0000 0777	0000 0777	palette des couleurs
32000 octets	xxxx	xxxx	image complète
32 octets		xxxx	cyclage des couleurs

Un fichier DEGAS non compressé a 32034 octets et un fichier DEGAS-ELITE non compressé a 32066 octets. La structure d'un fichier compressé est la même, le seul nombre qui change étant le nombre d'octets image compressée, ce résultat dépendant de l'image de départ. Voyons comment s'effectue la compression par ces logiciels.

La méthode est de réaliser cette compression ligne par ligne d'affichage et par plan de couleur dans cette ligne. Si l'image est en moyenne résolution (200 lignes de 80 octets sur deux plans de couleur), le programme compresse le premier plan de la première ligne d'affichage puis le deuxième plan de la première ligne d'affichage. Ces deux plans sont écrits l'un à la suite de l'autre, ce qui code de façon compressée la première ligne. Ce même processus se répète ainsi jusqu'à la dernière ligne de l'image. Pour compacter les plans de couleur, DEGAS utilise les octets

de commande dont nous allons vous donner la signification. Appelons COM la valeur de cet octet de commande (bit 7 = signe). Si COM est compris entre 0 et 127, il faut recopier les COM + 1 octets suivants tels quels dans le plan de couleur en cours.

Si COM est compris entre −127 et −1, il faut copier l'octet suivant 1 − COM fois dans le plan de couleur.

Si COM = 128 cette valeur est ignorée et on passe à l'octet suivant.

Les vérifications suivantes sont simples à envisager pour tester les différentes étapes d'un décompactage : la somme des octets des plans couleur est égale au nombre d'octets d'une ligne d'affichage (dépend de la résolution).

La somme des octets des lignes d'affichages est toujours 32000, quelle que soit la résolution.

Le premier octet significatif d'un fichier compressé est l'octet 34. Supposons que les octets 34 et 35 d'un fichier compressé en moyenne résolution aient les valeurs hexadécimales $45 et $12. L'octet 34 représente l'octet de commande. Sa valeur (69) est comprise entre 0 et 127. Il faut donc recopier l'octet qui suit (la valeur $12) 70 fois dans le premier plan couleur. En moyenne résolution, un plan couleur fait 80 octets, il reste donc 10 octets à remplir pour terminer cette ligne. Le même principe s'applique pour le second plan de la première ligne (octets 36 et 37) ; le premier plan de la seconde ligne (octets 38 et 39) complètera le premier plan qui n'était pas complet et ainsi de suite jusqu'à la 200e ligne.

La seule différence entre les résolutions est le nombre de plans et de lignes. Le compactage est lui issu de la même logique, coder une répétition d'octets par le nombre −1 suivi de la valeur répétée. Il est vrai qu'en graphisme cette méthode de compactage est assez efficace, mais d'autres méthodes de compactages existent et sont souvent encore plus drastiques. La recherche s'effectue en plus sur des répétitions de trames un peu sur le même principe de compactage ou en utilisant l'algorithme d'Huffman qui permet de compacter des fichiers beaucoup plus importants avec parfois des seuils de compactage qui frôlent les 80 % !!

Par souci de vélocité les programmes sont écrits en assembleur et peuvent être facilement intégrés dans un programme Basic GFA par INLINE ou en chargeant les codes assembleurs dans une variable chaîne de caractères.

En haute résolution, il n'y a qu'un plan de couleur donc nous n'avons pas besoin de tampon de stockage.

En moyenne résolution, il y a deux plans de couleur donc deux tampons de stockage de 16 000 octets chacun.

En basse résolution, il y a quatre plans de couleur donc quatre tampons de stockage de 8 000 octets chacun.

Les programmes basse et haute résolution attendent respectivement sur la pile l'adresse du tampon de travail et l'adresse de l'image à décompacter.



En haute résolution, il suffit de donner l'adresse de l'image à décompacter.

EXEMPLES D'APPEL EN BASIC GFA

Haute résolution:

```
INLINE decompact3%,&H76
OPEN "i",#1,"A:\DEGAS\COYOTE.PC3"
b$=INPUT$(LOF(#1),#1)
CLOSE #1
HIDEM
~C:decompact3%(L:V:b$)
CLR b$
~INP(2)
SHOWM
EDIT
```

Moyenne résolution:

```
a$=STRING$(32000,0)
INLINE decompact2%,&HC8
OPEN "i",#1,"A:\DEGAS\COYOTE.PC2"
b$=INPUT$(LOF(#1),#1)
CLOSE #1
~C:decompact2%(L:V:a$,L:V:b$)
CLR a$,b$
~INP(2)
EDIT
```

Basse résolution:

```
a$=STRING$(32000,0)
INLINE decompact1%,&HFA
OPEN "i",#1,"A:\DEGAS\COYOTE.PC1"
b$=INPUT$(LOF(#1),#1)
CLOSE #1
HIDEM
~C:decompact1%(L:V:a$,L:V:b$)
CLR a$,b$
~INP(2)
SHOWM
EDIT
```

LISTINGS EN ASSEMBLEUR

ROUTINES DE DECOMPACTAGE DES IMAGES DEGAS TOUTES RESOLUTIONS

ROUTINE DE DÉCOMPACTAGE IMAGES DEGAS BASSE RÉSOLUTION

```
.TEXT
        movem.l d0-d7/a0-a6,-(sp)
        move.l  a4,-(sp)
        move.w  #$3,-(sp)
        trap    #$e
        addq.l  #2,sp
        movea.l d0,a6                   ;ecran arrivee physique ici

        movea.l 72(sp),a4               ;adresse image compressée
        addq.l  #$2,a4
        move.l  a4,-(sp)
        move.w  #$6,-(sp)
        trap    #$e                     ;setpalette
        addq.l  #6,sp
        movea.l 68(sp),a1
        adda.l  #$20,a4
        movea.l a1,a0                   ;plan coul n°1 8000 octets
        adda.l  #8000,a1                ;plan couleur n° 2 8000 octets
        movea.l a1,a2
        adda.l  #8000,a2                ;plan couleur n°3 8000 octets
        movea.l a2,a3
        adda.l  #8000,a3                ;plan couleur n°4 8000octets
        move.l  a0,d4
        move.l  a1,d5
        move.l  a0,d6
        move.l  a1,d7
        move.w  #199,d0                 ; 200 lignes à decompacter
tes:    movea.l a0,a5
        bsr     dec
        movea.l a5,a0
        movea.l a1,a5
        bsr     dec
        movea.l a5,a1
        movea.l a2,a5
        bsr     dec
        movea.l a5,a2
        movea.l a3,a5
        bsr     dec
        movea.l a5,a3
        dbf     d0,tes
        bsr     aff
        movem.l (sp)+,a4
        bra     ter
dec:    move.b  #0,d1
```



```
dec10:  cmpi.b   #40,d1
        beq      fi
        move.b   (a4)+,d2
        cmpi.b   #0,d2
        blt      cas
        addq.b   #1,d2
dec20:  move.b   (a4)+,(a5)+
        addq.b   #1,d1
        subq.b   #1,d2
        cmpi.b   #0,d2
        bgt      dec20
        bra      dec10
cas:    neg.b    d2
        addq.b   #1,d2
        move.b   (a4)+,d3
cas1:   move.b   d3,(a5)+
        addq.b   #1,d1
        subq.b   #1,d2
        cmpi.b   #0,d2
        bgt      cas1
        bra      dec10


aff:    movea.l  a6,a4
        movea.l  d4,a0
        movea.l  d5,a1
        movea.l  d6,a2
        movea.l  d7,a3

        move.w   #7999,d0          ;8000 octets par plan
af1:    move.w   (a0)+,(a4)+
        move.w   (a1)+,(a4)+
        move.w   (a2)+,(a4)+
        move.w   (a3)+,(a4)+
        dbf      d0,af1
fi:     rts
ter:    movem.l  (sp)+,d0-d7/a0-a6
        rts
        .END
```

ROUTINE DE DÉCOMPACTAGE IMAGES DEGAS MOYENNE RÉSOLUTION

```
.TEXT
        movem.l  d0-d7/a0-a6,-(sp)
        move.w   #$3,-(sp)
        trap     #$e
        addq.l   #2,sp
        movea.l  d0,a5             ;écran d'affichage physique
        movea.l  68(sp),a2         ;adresse image compressée
```

```
            addq.l    #2,a2
            move.l    a2,-(sp)
            move.w    #6,-(sp)
            trap      #$e                ;setpalette
            addq.l    #6,sp
            movea.l   64(sp),a1          ;plan coul n°2 16000 octets
            adda.l    #$20,a2
            movea.l   a1,a0              ;plan coul n°1 16000 octets
            adda.l    #16000,a1
            move.l    a0,d5
            move.l    a1,d6
            move.w    #199,d0
test:       movea.l   a0,a3
            bsr       decomp
            movea.l   a3,a0
            movea.l   a1,a3
            bsr       decomp
            movea.l   a3,a1
            dbf       d0,test
            bsr       affic
            bra       term
decomp:     move.b    #0,d1
dec1:       cmpi.b    #80,d1
            beq       fin
            move.b    (a2)+,d2
            cmpi.b    #0,d2
            blt       cash
            addq.b    #1,d2
dec2:       move.b    (a2)+,(a3)+
            addq.b    #1,d1
            subq.b    #1,d2
            cmpi.b    #0,d2
            bgt       dec2
            bra       dec1
cash:       neg.w     d2
            addq.b    #1,d2
            move.b    (a2)+,d3
cash1:      move.b    d3,(a3)+
            addq.b    #1,d1
            subq.b    #1,d2
            cmpi.b    #0,d2
            bgt       cash1
            bra       dec1
affic:      movea.l   a5,a2
            movea.l   d5,a0
            movea.l   d6,a1
            move.w    #15999,d0
aff1:       move.w    (a0)+,(a2)+
            move.w    (a1)+,(a2)+
            dbf       d0,aff1
```



```
fin:    rts
term:   movem.l   (sp)+,d0-d7/a0-a6
        rts
        .END
```

ROUTINE DE DÉCOMPACTAGE DEGAS HAUTE RÉSOLUTION

```
.TEXT
        movem.l   d0-d7/a0-a6,-(sp)
        move.w    #$3,-(sp)
        trap      #$e
        addq.l    #2,sp
        movea.l   d0,a1
        movea.l   64(sp),a0
        adda.l    #$22,a0
        clr.w     d1
be:     cmpi.w    #32000,d1
        bge       fin
        move.b    (a0)+,d0
        cmpi.b    #0,d0
        blt       cn
        addq.b    #1,d0
cp:     move.b    (a0)+,(a1)+
        addq.w    #1,d1
        subq.b    #1,d0
        cmpi.b    #0,d0
        bgt       cp
        bra       be
cn:     neg.b     d0
        addq.b    #1,d0
        move.b    (a0)+,d2
stn:    move.b    d2,(a1)+
        addq.w    #1,d1
        subq.b    #1,d0
        cmpi.b    #0,d0
        bgt       stn
        bra       be
fin:    movem.l   (sp)+,d0-d7/a0-a6
        rts
        .END
```

LISTINGS POUR CHARGEUR

Programme basse résolution :

```
$43FAA:[ 48e7 fffe                ]          movem.l   d0-d7/a0-a6,-(a7)
$43FAE:[ 2f0c                     ]          move.l    a4,-(a7)
$43FB0:[ 3f3c 0003                ]          move.w    #$3,-(a7)
$43FB4:[ 4e4e                     ]          trap      #$e
$43FB6:[ 548f                     ]          addq.l    #$2,a7
$43FB8:[ 2c40                     ]          movea.l   d0,a6
$43FBA:[ 286f 0048                ]          movea.l   $48(a7),a4
$43FBE:[ 548c                     ]          addq.l    #$2,a4
$43FC0:[ 2f0c                     ]          move.l    a4,-(a7)
$43FC2:[ 3f3c 0006                ]          move.w    #$6,-(a7)
$43FC6:[ 4e4e                     ]          trap      #$e
$43FC8:[ 5c8f                     ]          addq.l    #$6,a7
$43FCA:[ 226f 0044                ]          movea.l   $44(a7),a1
$43FCE:[ d9fc 00000020            ]          adda.l    #$20,a4
$43FD4:[ 2049                     ]          movea.l   a1,a0
$43FD6:[ d3fc 00001f40            ]          adda.l    #$1f40,a1
$43FDC:[ 2449                     ]          movea.l   a1,a2
$43FDE:[ d5fc 00001f40            ]          adda.l    #$1f40,a2
$43FE4:[ 264a                     ]          movea.l   a2,a3
$43FE6:[ d7fc 00001f40            ]          adda.l    #$1f40,a3
$43FEC:[ 2808                       ]          move.l    a0,d4

$43FEE:[ 2a09                     ]          move.l    a1,d5
$43FF0:[ 2c08                     ]          move.l    a0,d6
$43FF2:[ 2e09                     ]          move.l    a1,d7
$43FF4:[ 303c 00c7                ]          move.w    #$c7,d0
$43FF8:[ 2a48                     ]          movea.l   a0,a5
$43FFA:[ 6100 002c                ]          bsr       $44028
$43FFE:[ 204d                     ]          movea.l   a5,a0
$44000:[ 2a49                     ]          movea.l   a1,a5
$44002:[ 6100 0024                ]          bsr       $44028
$44006:[ 224d                     ]          movea.l   a5,a1
$44008:[ 2a4a                     ]          movea.l   a2,a5
$4400A:[ 6100 001c                ]          bsr       $44028
$4400E:[ 244d                     ]          movea.l   a5,a2
$44010:[ 2a4b                     ]          movea.l   a3,a5
$44012:[ 6100 0014                ]          bsr       $44028
$44016:[ 264d                     ]          movea.l   a5,a3
$44018:[ 51c8 ffde                ]          dbra      d0,$43ff8
$4401C:[ 6100 0044                ]          bsr       $44062
$44020:[ 4cdf 1000                ]          movem.l   (a7)+,a4
$44024:[ 6000 0058                ]          bra       $4407e
$44028:[ 123c 0000                ]          move.b    #$0,d1
$4402C:[ 0c01 0028                ]          cmpi.b    #$28,d1
$44030:[ 6700 004a                ]          beq       $4407c
$44034:[ 141c                     ]          move.b    (a4)+,d2
$44036:[ 0c02 0000                ]          cmpi.b    #$0,d2
$4403A:[ 6d00 0012                ]          blt       $4404e
$4403E:[ 5202                     ]          addq.b    #$1,d2
$44040:[ 1adc                     ]          move.b    (a4)+,(a5)+
```



```
$44042:[ 5201                ]        addq.b    #$1,d1
$44044:[ 5302                ]        subq.b    #$1,d2
$44046:[ 0c02 0000           ]        cmpi.b    #$0,d2
$4404A:[ 6ef4                ]        bgt.s     $44040
$4404C:[ 60de                ]        bra.s     $4402c
$4404E:[ 4402                ]        neg.b     d2
$44050:[ 5202                ]        addq.b    #$1,d2
$44052:[ 161c                ]        move.b    (a4)+,d3
$44054:[ 1ac3                ]        move.b    d3,(a5)+
$44056:[ 5201                ]        addq.b    #$1,d1
$44058:[ 5302                ]        subq.b    #$1,d2
$4405A:[ 0c02 0000           ]        cmpi.b    #$0,d2
$4405E:[ 6ef4                ]        bgt.s     $44054
$44060:[ 60ca                ]        bra.s     $4402c
$44062:[ 284e                ]        movea.l   a6,a4
$44064:[ 2044                ]        movea.l   d4,a0
$44066:[ 2245                ]        movea.l   d5,a1
$44068:[ 2446                ]        movea.l   d6,a2
$4406A:[ 2647                ]        movea.l   d7,a3
$4406C:[ 303c 1f3f           ]        move.w    #$1f3f,d0
$44070:[ 38d8                ]        move.w    (a0)+,(a4)+
$44072:[ 38d9                ]        move.w    (a1)+,(a4)+
$44074:[ 38da                ]        move.w    (a2)+,(a4)+
$44076:[ 38db                ]        move.w    (a3)+,(a4)+
$44078:[ 51c8 fff6           ]        dbra      d0,$44070
$4407C:[ 4e75                ]        rts       ;
$4407E:[ 4cdf 7fff           ]        movem.l   (a7)+,d0-d7/a0-a6
$44082:[ 4e75                ]        rts       ;--
```

Programme moyenne résolution :

```
$45FAE:[ 48e7 fffe          ]    movem.l   d0-d7/a0-a6,-(a7)
$45FB2:[ 3f3c 0003          ]    move.w    #$3,-(a7)
$45FB6:[ 4e4e               ]    trap      #$e
$45FB8:[ 548f               ]    addq.l    #$2,a7
$45FBA:[ 2a40               ]    movea.l   d0,a5
$45FBC:[ 246f 0044          ]    movea.l   $44(a7),a2
$45FC0:[ 548a               ]    addq.l    #$2,a2
$45FC2:[ 2f0a               ]    move.l    a2,-(a7)
$45FC4:[ 3f3c 0006          ]    move.w    #$6,-(a7)
$45FC8:[ 4e4e               ]    trap      #$e
$45FCA:[ 5c8f               ]    addq.l    #$6,a7
$45FCC:[ 226f 0040          ]    movea.l   $40(a7),a1
$45FD0:[ d5fc 00000020      ]    adda.l    #$20,a2
$45FD6:[ 2049               ]    movea.l   a1,a0
$45FD8:[ d3fc 00003e80      ]    adda.l    #$3e80,a1
$45FDE:[ 2a08               ]    move.l    a0,d5
$45FE0:[ 2c09               ]    move.l    a1,d6
$45FE2:[ 303c 00c7          ]    move.w    #$c7,d0
$45FE6:[ 2648               ]    movea.l   a0,a3
$45FE8:[ 6100 0018          ]    bsr       $46002
$45FEC:[ 204b               ]    movea.l   a3,a0
$45FEE:[ 2649               ]    movea.l   a1,a3
$45FF0:[ 6100 0010          ]    bsr       $46002
$45FF4:[ 224b               ]    movea.l   a3,a1
$45FF6:[ 51c8 ffee          ]    dbra      d0,$45fe6
$45FFA:[ 6100 0040          ]    bsr       $4603c
$45FFE:[ 6000 0050          ]    bra       $46050
$46002:[ 123c 0000          ]    move.b    #$0,d1
$46006:[ 0c01 0050          ]    cmpi.b    #$50,d1
$4600A:[ 6700 0042          ]    beq       $4604e
$4600E:[ 141a               ]    move.b    (a2)+,d2
$46010:[ 0c02 0000          ]    cmpi.b    #$0,d2
$46014:[ 6d00 0012          ]    blt       $46028
$46018:[ 5202               ]    addq.b    #$1,d2
$4601A:[ 16da               ]    move.b    (a2)+,(a3)+
$4601C:[ 5201               ]    addq.b    #$1,d1
$4601E:[ 5302               ]    subq.b    #$1,d2
$46020:[ 0c02 0000          ]    cmpi.b    #$0,d2
$46024:[ 6ef4               ]    bgt.s     $4601a
$46026:[ 60de               ]    bra.s     $46006
$46028:[ 4442               ]    neg.w     d2
$4602A:[ 5202               ]    addq.b    #$1,d2
$4602C:[ 161a               ]    move.b    (a2)+,d3
$4602E:[ 16c3               ]    move.b    d3,(a3)+
$46030:[ 5201               ]    addq.b    #$1,d1
$46032:[ 5302               ]    subq.b    #$1,d2
$46034:[ 0c02 0000          ]    cmpi.b    #$0,d2
$46038:[ 6ef4               ]    bgt.s     $4602e
$4603A:[ 60ca               ]    bra.s     $46006
$4603C:[ 244d               ]    movea.l   a5,a2
$4603E:[ 2045               ]    movea.l   d5,a0
$46040:[ 2246               ]    movea.l   d6,a1
$46042:[ 303c 3e7f          ]    move.w    #$3e7f,d0
$46046:[ 34d8               ]    move.w    (a0)+,(a2)+
$46048:[ 34d9               ]    move.w    (a1)+,(a2)+
$4604A:[ 51c8 fffa          ]    dbra      d0,$46046
$4604E:[ 4e75               ]    rts       ;
$46050:[ 4cdf 7fff          ]    movem.l   (a7)+,d0-d7/a0-a6
$46054:[ 4e75               ]    rts       ;..
```



Programme haute résolution :

```
$45FB2:[ 48e7 fffe                ]        movem.l   d0-d7/a0-a6,-(a7)
$45FB6:[ 3f3c 0003                ]        move.w    #$3,-(a7)
$45FBA:[ 4e4e                     ]        trap      #$e
$45FBC:[ 548f                     ]        addq.l    #$2,a7
$45FBE:[ 2240                     ]        movea.l   d0,a1
$45FC0:[ 206f 0040                ]        movea.l   $40(a7),a0
$45FC4:[ d1fc 00000022            ]        adda.l    #$22,a0
$45FCA:[ 4241                     ]        clr.w     d1
$45FCC:[ 0c41 7d00                ]        cmpi.w    #$7d00,d1
$45FD0:[ 6c00 0030                ]        bge       $46002
$45FD4:[ 1018                     ]        move.b    (a0)+,d0
$45FD6:[ 0c00 0000                ]        cmpi.b    #$0,d0
$45FDA:[ 6d00 0012                ]        blt       $45fee
$45FDE:[ 5200                     ]        addq.b    #$1,d0
$45FE0:[ 12d8                     ]        move.b    (a0)+,(a1)+
$45FE2:[ 5241                     ]        addq.w    #$1,d1
$45FE4:[ 5300                     ]        subq.b    #$1,d0
$45FE6:[ 0c00 0000                ]        cmpi.b    #$0,d0
$45FEA:[ 6ef4                     ]        bgt.s     $45fe0
$45FEC:[ 60de                     ]        bra.s     $45fcc
$45FEE:[ 4400                     ]        neg.b     d0
$45FF0:[ 5200                     ]        addq.b    #$1,d0
$45FF2:[ 1418                     ]        move.b    (a0)+,d2
$45FF4:[ 12c2                     ]        move.b    d2,(a1)+
$45FF6:[ 5241                     ]        addq.w    #$1,d1
$45FF8:[ 5300                     ]        subq.b    #$1,d0
$45FFA:[ 0c00 0000                ]        cmpi.b    #$0,d0
$45FFE:[ 6ef4                     ]        bgt.s     $45ff4
$46000:[ 60ca                     ]        bra.s     $45fcc
$46002:[ 4cdf 7fff                ]        movem.l   (a7)+,d0-d7/a0-a6
$46006:[ 4e75                     ]        rts       ;..
```



5/5.3

Outil de dessin

Le langage STOS disponible depuis peu en version française facilite grandement la vie du programmeur. Si vous n'en êtes pas encore adepte, voici un petit programme qui devrait vous convaincre de sa facilité d'utilisation et de sa puissance. L'outil de dessin étudié ici est totalement écrit en STOS. Il permet de réaliser des dessins en basse résolution 320×200 points, 16 couleurs, format PI1.

A travers un menu déroulant, il vous est possible de :

— choisir la couleur de dessin parmi les 16 possibles ;

— déplacer une zone rectangulaire ;

— dupliquer une zone rectangulaire ;

— effacer une zone rectangulaire ;

— zoomer une zone rectangulaire dans une autre de format quelconque ;

— réduire l'écran à une zone rectangulaire quelconque.

COMMENT UTILISER LE PROGRAMME

Le programme est écrit en Basic STOS. Vous devez donc posséder ce langage pour pouvoir l'exécuter. Pas question par exemple de taper le listing sous GFA Basic. Cela vous mènerait inévitablement vers des erreurs insurmontables.

Les dessins étant effectués en basse résolution, vous devez également posséder un écran couleur pour pouvoir exécuter le programme.

Si ces deux conditions sont réunies, lancez le STOS en introduisant la disquette LANGAGE dans le lecteur A: et en bootant l'ordinateur. Quelques secondes après cette manipulation, vous vous trouvez dans l'éditeur STOS. Entrez les lignes du listing de la page suivante.

Exécutez le programme en appuyant sur la touche de fonction <F6>. Le curseur de la souris est représenté par une main dont l'index est pointé. Déplacez ce curseur sur l'écran. Lorsqu'il passe sur un des menus affichés en haut de l'écran, un menu se déroule.

— le menu "Couleurs" permet de choisir la couleur de dessin ;

— le menu "Bloc" donne accès à plusieurs fonctions de manipulation intéressantes ;

```
1000 rem ------------------
1010 rem Outil de dessin
1020 rem ------------------
1030 reserve as screen 5
1040 mode 0
1050 rem - - - - - - - - - - - - - - - - - -
1060 rem Définition des options du menu
1070 rem - - - - - - - - - - - - - - - - - -
1080 menu$ (1)="Couleurs"
1090 menu$ (2)="Bloc"
1100 menu$ (3)="Fin"
1110 menu$ (1,1)="      ",0,0
1120 menu$ (1,2)="      ",1,0
1130 menu$ (1,3)="      ",2,0
1140 menu$ (1,4)="      ",3,0
1150 menu$ (1,5)="      ",4,0
1160 menu$ (1,6)="      ",5,0
1170 menu$ (1,7)="      ",6,0
1180 menu$ (1,8)="      ",7,0
1190 menu$ (1,9)="      ",8,0
1200 menu$ (1,10)="      ",9,0
1210 menu$ (1,11)="      ",10,0
1220 menu$ (1,12)="      ",11,0
1230 menu$ (1,13)="      ",12,0
1240 menu$ (1,14)="      ",13,0
1250 menu$ (1,15)="      ",14,0
1260 menu$ (1,16)="      ",15,0
1270 menu$ (2,1)="Déplacer"
1280 menu$ (2,2)="Copier"
1290 menu$ (2,3)="Effacer"
1300 menu$ (2,4)="Zoom"
1310 menu$ (2,5)="Réduction"
1320 menu$ (3,1)="Fin du programme"
1330 menu on
1340 change mouse 2 : rem Curseur souris = main avec index pointé
1350 on menu goto 1450,1500,2270
1360 on menu on
1370 rem Test de la souris
1380 repeat
1390 M=mouse key
1400 until M<>0
1410 if M=1 then plot x mouse,y mouse,COULEUR-1
1420 if M=2 then plot x mouse,y mouse,0
1430 goto 1350
1440 end
1450 rem - - - - - - - - - - - - - - - - - -
1460 rem Choix d'une couleur de tracé
1470 rem - - - - - - - - - - - - - - - - - -
1480 COULEUR=mnselect
1490 goto 1350
1500 rem - - - - - - - - - - - - - - - -
1510 rem Opérations sur les blocs
1520 rem - - - - - - - - - - - - - - - -
1530 on mnselect goto 1560,1750,1910,2010,2140
```



(suite)

```
1540 goto 1350
1550 rem - - - - - - - - - - - - -
1560 rem Déplacement d'un bloc
1570 rem - - - - - - - - - - - - -
1580 gosub 2330 : rem Choix du bloc à déplacer
1590 repeat
1600 until mouse key
1610 hide
1620 ink 0
1630 gr writing 1
1640 box X1,Y1 to X2,Y2
1650 screen copy physic,X1+1,Y1+1,X2-1,Y2-1 to physic,x mouse,y mouse
1660 screen copy physic,X1+1,Y1+1,X2-1,Y2-1 to back,x mouse,y mouse
1670 bar X1,Y1 to X2,Y2
1680 ink COULEUR-1
1690 gr writing 3
1700 show
1710 repeat
1720 until mouse key=0
1730 goto 1350
1740 rem - - - - - - - - - -
1750 rem Copie d'un bloc
1760 rem - - - - - - - - - -
1770 gosub 2330 : rem Choix du bloc à déplacer
1780 repeat
1790 until mouse key
1800 hide
1810 ink 0
1820 box X1,Y1 to X2,Y2
1830 ink COULEUR-1
1840 screen copy physic,X1+1,Y1+1,X2-1,Y2-1 to physic,x mouse,y mouse
1850 screen copy physic,X1+1,Y1+1,X2-1,Y2-1 to back,x mouse,y mouse
1860 show
1870 repeat
1880 until mouse key=0
1890 goto 1350
1900 rem - - - - - - - - - - -
1910 rem Effacement d'un bloc
1920 rem - - - - - - - - - - -
1930 gosub 2330 : rem Choix du bloc à déplacer
1940 gr writing 1
1950 ink 0
1960 bar X1,Y1 to X2,Y2
1970 ink COULEUR-1
1980 gr writing 3
1990 goto 1350
2000 rem - - - - - - - - - - - - - - - -
2010 rem Zoom d'une zone rectangulaire
2020 rem - - - - - - - - - - - - - - - -
2030 gosub 2330 : rem Choix de la zone à réduire
2040 gosub 2500 : rem Choix de la zone de recopie
2050 zoom physic,X1+1,Y1+1,X2-1,Y2-1 to physic,X3,Y3,X4,Y4
2060 zoom physic,X1+1,Y1+1,X2-1,Y2-1 to back,X3,Y3,X4,Y4
2070 gr writing 1
```

(suite)

```
2080 ink 0
2090 box X1,Y1 to X2,Y2
2100 gr writing 3
2110 ink COULEUR-1
2120 goto 1350
2130 rem - - - - - - - - - - -
2140 rem Réduction de l'écran
2150 rem - - - - - - - - - - -
2160 screen copy physic to 5
2170 gosub 2330 : rem choix de la zone résultante
2180 hide
2190 cls
2200 reduce 5 to X1,Y1,X2,Y2
2210 gr writing 3
2220 ink COULEUR-1
2230 show
2240 change mouse 2
2250 goto 1350
2260 rem - - - - - - - - -
2270 rem Fin du programme
2280 rem - - - - - - - - -
2290 menu off
2300 mode 1
2310 end
2320 flash off : rem Inhibition du flashing
2330 rem - - - - - - - - - - - - - - - - - - - -
2340 rem Délimitation d'une zone rectangulaire
2350 rem - - - - - - - - - - - - - - - - - - - -
2360 gr writing 3 : rem Mode d'affichage XOR
2370 repeat
2380 until mouse key
2390 wait 10
2400 X1=x mouse : Y1=y mouse : X2=X1 : Y2=Y1
2410 repeat
2420 box X1,Y1 to X2,Y2
2430 X2=x mouse : Y2=y mouse
2440 box X1,Y1 to X2,Y2
2450 M=mouse key
2460 until M<>0
2470 repeat
2480 until mouse key=0
2490 return
2500 rem - - - - - - - - - - - - - - - - - - - - - - - -
2510 rem Délimitation d'une seconde zone rectangulaire
2520 rem - - - - - - - - - - - - - - - - - - - - - - - -
2530 repeat
2540 until mouse key
2550 wait 10
2560 X3=x mouse : Y3=y mouse : X4=X3 : Y4=Y3
2570 repeat
2580 box X3,Y3 to X4,Y4
2590 X4=x mouse : Y4=y mouse
2600 box X3,Y3 to X4,Y4
2610 M=mouse key
2620 until M<>0
2630 repeat
2640 until mouse key=0
2650 return
```



— le menu **Fin** permet de quitter le programme.

Sélectionnez la couleur dans le menu "**Couleurs**". Pour cela, déroulez ce menu et cliquez sur une couleur de votre choix. Dès à présent, il vous est possible de dessiner sur l'écran avec la couleur sélectionnée en appuyant sur le bouton gauche de la souris. Le bouton droit de la souris efface les pixels sur lesquels passe la souris. Pour changer la couleur courante, il vous suffit d'utiliser à nouveau le menu "**Couleurs**".

Les options du menu "**Bloc**" parlent d'elles-mêmes :

— "**Déplacer**" permet de déplacer une zone rectangulaire de l'écran ;

— "**Copier**" permet de copier une zone rectangulaire de l'écran ;

— "**Effacer**" permet d'effacer une zone rectangulaire de l'écran ;

— "**Zoom**" permet d'agrandir une zone rectangulaire de l'écran ;

— "**Réduction**" permet de réduire l'écran à une zone rectangulaire de votre choix.

Dessinez (ou gribouillez) un objet ou une forme quelconque sur l'écran. Pour déplacer cet objet, procédez comme suit :

1) sélectionnez l'option "**Déplacer**" dans le menu "**Bloc**" ;

2) amenez l'index du curseur en haut et à gauche de l'objet ;

3) cliquez ;

4) déplacez la souris pour encadrer l'objet ;

5) cliquez ;

6) amenez l'index du curseur en haut et à gauche de la zone où doit être déplacé l'objet ;

7) cliquez. L'objet est déplacé.

Les options "**Copier**" et "**Effacer**" s'utilisent de manière similaire à l'option "**Déplacer**".

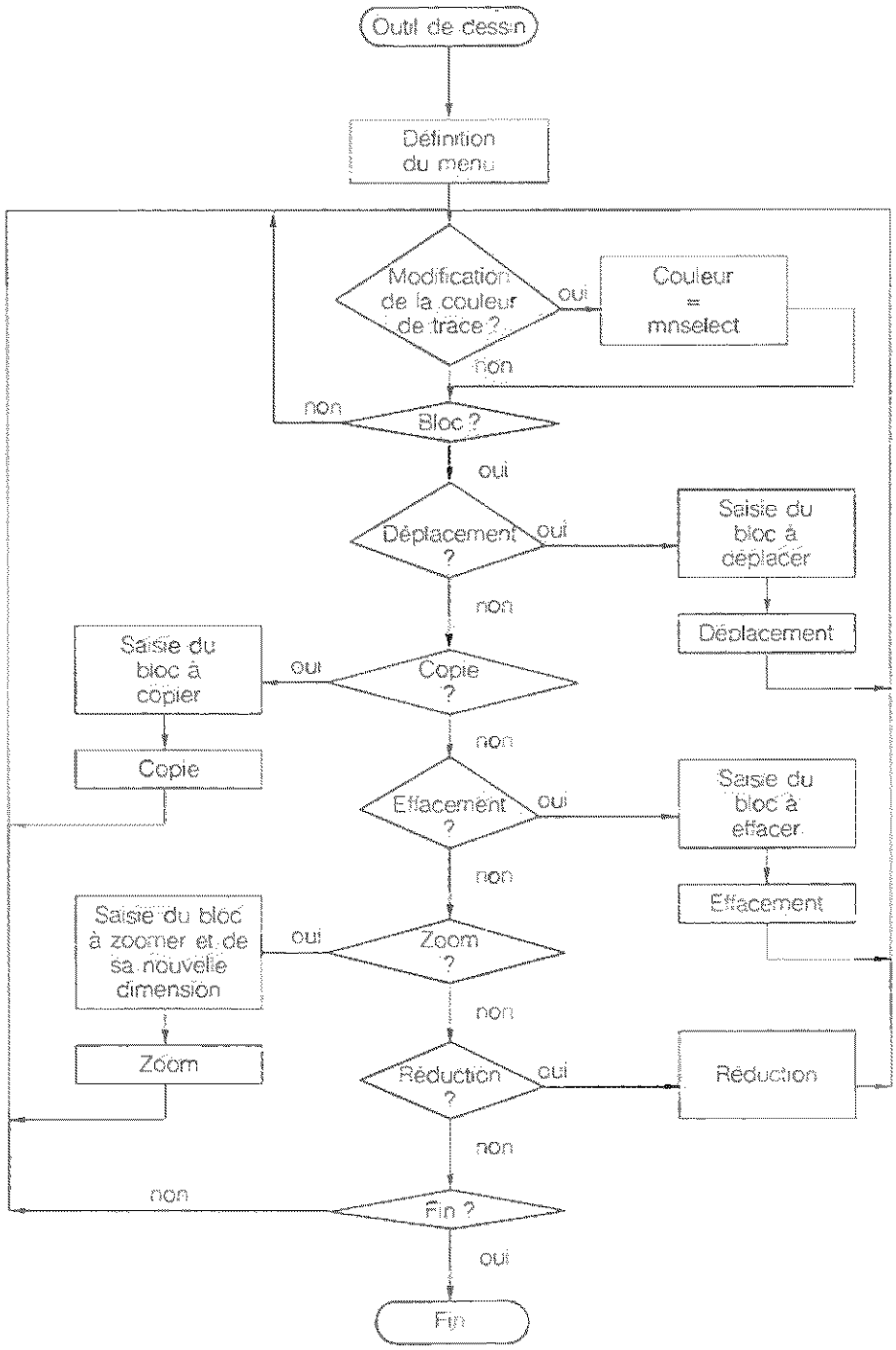
Dans l'option "**Zoom**", vous devez dans un premier temps encadrer l'objet à agrandir, et dans un second temps définir la zone d'agrandissement.

Dans l'option "**Réduction**", vous devez définir la zone dans laquelle l'écran sera réduit.

Le menu "**Fin**" met fin au programme et redonne le contrôle au STOS.

LE PROGRAMME EN DÉTAIL

La logique du programme apparaît dans l'ordinogramme suivant :





Le programme débute par la réservation d'une banque mémoire écran :

```
1030 reserve as screen 5
```

Le mode d'affichage est ensuite initialisé en basse résolution :

```
1040 mode 0
```

Les options du menu sont définies dans le tableau **menu$, menu$(1),** menu$(2) et menu$(3) contiennent les trois options principales de menu :

```
1080 menu$ (1) = "Couleurs"
1090 menu$ (2) = "Bloc"
1100 menu$ (3) = "Fin"
```

Le **menu 1** (Couleurs) contient des barres colorées qui représentent les 16 couleurs du stylo :

```
1110 menu$ (1,1) = " ",0,0
...
1260 menu$ (1,16) = " ",15,0
```

Le menu 2 (Bloc) contient cinq options :

```
1270 menu$ (2,1) = "Déplacer"
1280 menu$ (2,2) = "Copier"
1290 menu$ (2,3) = "Effacer"
1300 menu$ (2,4) = "Zoom"
1310 menu$ (2,5) = "Réduction"
```

Enfin, le menu 3 (Fin) contient une seule option :

```
1320 menu$ (3,1) = "Fin du programme"
```

Le menu est validé par l'instruction menu on :

```
1330 menu on
```

Le curseur de la souris est modifié en une main dont l'index est pointé :

```
1340 change mouse 2 : rem Curseur souris = main avec index pointé
```

Les trois options principales de menu sont traitées aux lignes 1450, 1500 et 2300, et les interruptions liées au menu sont validées :

```
1350 on menu goto 1450,1500,2300
1360 on menu on
```

Le programme se poursuit par une boucle dans laquelle les boutons de la souris sont testés :

```
1370 remp Test de la souris
1380 repeat
1390 M = mouse key
1400 until M <>0
```

Si le bouton gauche de la souris est pressé, le pixel qui se trouve aux coordonnées de la souris est allumé avec la couleur COULEUR – 1 :

```
1410 if M = 1 then plot x mouse,y mouse,COULEUR – 1
```

Si le bouton droit de la souris est pressé, le pixel qui se trouve aux coordonnées de la souris est éteint :

```
1420 if M=2 then plot x mouse,y mouse,0
```

Lorsqu'une des options du premier menu est sélectionnée, le gestionnaire d'interruptions donne le contrôle au programme qui se trouve en ligne 1450. Ce programme initialise la variable couleur en fonction de l'option sélectionnée et redonne le contrôle à la boucle principale :

```
1480 COULEUR=mnselect
1490 goto 1350
```

Lorsqu'une des options du menu "Bloc" est sélectionnée, le gestionnaire d'interruptions donne le contrôle au programme qui se trouve en ligne 1500. Ce programme active le traitement qui correspond à l'option choisie et redonne le contrôle à la boucle principale :

```
1530 on mnselect goto 1560,1750,1910,2010,2140
1540 goto 1350
```

Lorsque l'option "Déplacer" est sélectionnée, le programme situé en 1550 est exécuté.

La première action effectuée dans ce programme consiste à saisir les coordonnées extrémales du bloc à déplacer à l'aide du sous-programme situé en 2360 :

```
1580 gosub 2360 : rem Choix du bloc à déplacer
```

Lorsque le bouton de la souris est relâché :

```
1590 repeat
1600 until mouse key
```

le cadre matérialisant le bloc à déplacer est effacé :

```
1610 hide
1620 ink 0
1630 gr writing 1
1640 blox X1,Y1 to X2,Y2
```

Le rectangle situé entre les coordonnées X1,Y1 et X2,Y2 est recopié à la position courante de la souris :

```
1650 screen copy physic,X1+1,Y1+1,X2-1,Y2-1 to physic,x mouse,y mouse
1660 screen copy physic,X1+1,Y1+1,X2-1,Y2-1 to back,x mouse, y mouse
```

Il est ensuite effacé :

```
1670 bar X1,Y1 to X2,Y2
```

La couleur et le mode d'affichage sont ensuite restitués :

```
1680 ink COULEUR-1
1690 gr writing 3
```



et le programme rend le contrôle à la boucle principale lorsque les boutons de la souris sont relâchés :

```
1710 repeat
1720 until mouse key=0
1730 goto 1350
```

Lorsque l'option "**Copie**" est sélectionnée, le programme situé en 1740 est exécuté. Ce programme est la copie conforme du programme de déplacement dont nous venons de parler, excepté que le bloc copié n'est pas effacé.

Lorsque l'option "**Effacer**" est sélectionnée, le programme situé en 1900 est exécuté. Ce programme saisit les coordonnées extrémales du bloc à effacer à l'aide du sous-programme situé en ligne 2360 :

```
1930 gosub 2360 : rem Choix du bloc à déplacer
```

Le bloc est ensuite effacé à l'aide de l'instruction **bar** :

```
1940 gr writing 1
1950 ink 0
1960 bar X1,Y1 to X2,Y2
```

L'ancienne couleur et l'ancien mode d'affichage sont ensuite restitués, et le programme redonne le contrôle à la boucle principale :

```
1970 ink COULEUR-1
1980 gr writing 3
1990 goto 1350
```

Lorsque l'option "**Zoom**" est sélectionnée, le programme situé en 2000 est exécuté. Ce programme acquiert les coordonnées extrémales de la zone à réduire et de la zone de recopie à l'aide des sous-programmes situés en 2360 et en 2530 :

```
2030 gosub 2360 : rem Choix de la zone à réduire
2040 gosub 2530 : rem Choix de la zone de recopie
```

La zone rectangulaire comprise entre **X1,Y1 et X2,Y2** est ensuite zoomée dans le rectangle de coordonnées **X3,Y3 et X4,Y4** :

```
2050 zoom physic,X1+1,Y1+1,X2-1,Y2-1 to physic,X3,Y3,X4,Y4
2060 zoom physic,X1+1,Y1+1,X2-1,Y2-1 to back,X3,Y3,X4,Y4
```

Le rectangle situé autour de la zone à zoomer est effacé :

```
2070 gr writing 1
2080 ink 0
2090 box X1,Y1 to X2,Y2
```

L'ancienne couleur et l'ancien mode d'affichage sont ensuite restitués et le programme redonne le contrôle à la boucle principale :

```
2100 gr writing 3
2110 ink COULEUR-1
2120 goto 1350
```

Lorsque l'option "**Réduction**" est sélectionnée, le programme situé en 2130 est exécuté.

Ce programme sauvegarde l'écran physique courant dans la banque mémoire 5 réservée en début de programme :

```
2160 screen copy physic to 5
```

La zone rectangulaire dans laquelle l'écran sera réduit est ensuite saisie à l'aide du sous-programme situé en 2330 :

```
2170 gosub 2330 : rem choix de la zone résultante
```

L'écran est ensuite réduit à l'aide de l'instruction **reduce** :

```
2200 reduce 5 to X1,Y1,X2,Y2
```

Le mode d'affichage, la couleur et le curseur de la souris sont restitués et le programme redonne le programme à la boucle principale :

```
2210 gr writing 3
2220 ink COULEUR-1
2230 show
2240 change mouse 2
2250 goto 1350
```

Lorsque la seule option (Fin du programme) du menu 3 est sélectionnée, le gestionnaire d'interruptions donne le contrôle au programme situé en 2260.

Ce programme dévalide le menu, et efface l'écran avec une instruction **mode** :

```
2290 menu off
2300 mode 1
```

Les sous-programmes situés en 2330 et 2500 sont similaires. Ils permettent de délimiter une zone rectangulaire sur l'écran.

Examinons le premier de ces sous-programmes. Dans un premier temps, une action sur un des boutons de la souris est attendue :

```
2370 repeat
2380 until mouse key
```

Les coordonnées du curseur de la souris sont mémorisées dans les variables **X1,Y1** et **X2,Y2**.

```
2400 X1=x mouse : Y1=y mouse : X2=X1 : Y2=Y1
```

Les déplacements de la souris sont mémorisées dans les variables **X2** et **Y2**, et le rectangle situé entre les coordonnées **X1,Y1** et **X2,Y2** est tracé en temps réel. Le tracé du rectangle prend fin lorsqu'un bouton de la souris est pressé :

```
2410 repeat
2420 box X1,Y1 to X2,Y2
2430 X2=x mouse : Y2=y mouse
2440 box X1,Y1 to X2,Y2
2450 M=mouse key
2460 until M<>0
```

Le sous-programme rend le contrôle au programme appelant lorsque le bouton est relâché :

```
2470 repeat
2480 until mouse key=0
2490 return
```



5/5.3.1

Programme de dessin et sa reproduction en Basic OMIKRON

Ce petit programme se propose de vous permettre de dessiner des boîtes en suivant le clic gauche de la souris et de reproduire le même dessin après le clic droit de la souris. Le programme s'arrête sur un nouveau clic droit de souris.

Le principe est assez simple : définir des tableaux pour mémoriser les déplacements de la souris, puis rejouer les tableaux. Les dimensions prévues de tableaux sont ici de 5 000, ce qui vous permet de remplir pratiquement l'écran. Sur beaucoup plus de points, il faudra limiter la place du Basic en mémoire. Le mode de remplissage est le XOR ce qui permet au blanc de réapparaître sous un trait de noir. Par défaut, la zone de clipping est l'écran tout entier mais vous pouvez positionner le rectangle de votre choix à condition de prendre des valeurs cohérentes avec les valeurs maximales de pixels horizontaux et verticaux.

On attend un clic de souris pour démarrer mais on ne dessine et mémorise dans les tableaux que si on clique à gauche (MOUSEBUT AND 1). Le tableau B%L() mémorise les valeurs du clic de souris entre les clics gauches de souris, ce qui permet de déplacer la souris sans écrire dans ces moments là. C'est exactement ce qui se passera à la relecture des tableaux ultérieurement, puisqu'on alternera entre les modes 1 (Replace) et 3 (Xor). Notez que le programme ne stocke pas le mouvement et le rythme du dessin. Pour se faire, il faudrait créer un nouveau tableau pour stocker et relire les valeurs correspondantes du TIMER pour chacun des points.

Listing en Basic OMIKRON :

```
PRINT "CLIC GAUCHE:DESSIN"
PRINT "CLIC DROIT:REPRODUIRE"
T%L= TIMER
REPEAT UNTIL TIMER -T%L>200
PRINT ""
Mx%L=5000
DIM X%(Mx%L),Y%(Mx%L),B%F(Mx%L)
FILL COLOR =1:
FILL STYLE =0,1: MODE =3
CLIP 0,0,640,400
REPEAT X%L= MOUSEX -20
Y%L= MOUSEY -20 UNTIL MOUSEBUT
PBOX X%L,Y%L,20,20
PBOX X%L,Y%L,20,20
REPEAT
    IPL 7:X%L= MOUSEX -20
    Y%L= MOUSEY -20
    B%L= MOUSEBUT AND 1
    IPL 3
    X%(I%L)=X%L:Y%(I%L)=Y%L
    B%F(I%L)=B%L:I%L=I%L+1
    MODE =3-B%L*2: PBOX X%L,Y%L,20,20
    PBOX X%L,Y%L,20,20
UNTIL MOUSEBUT AND 2
REPEAT UNTIL MOUSEBUT =0
Fin%L=I%L
REPEAT
    PRINT "";
    FOR I%L=0 TO Fin%L-1
        MODE =3-B%F(I%L)*2
        PBOX X%(I%L),Y%(I%L),20,20
        PBOX X%(I%L),Y%(I%L),20,20
    NEXT I%L
UNTIL MOUSEBUT AND 2
```

5/5.4

ROTABLOCK Programme de rotation −90° pour section d'écran saisie par GET X, Y, X1, Y1, ?$ En Basic GFA & en haute résolution

Tous les utilisateurs de DEGAS ELITE connaissent la fonction **BLOCK ROTATE**. Une fois sélectionnée, et après pression sur la touche : ⇨, cette fonction effectue une rotation de −90° du block pré-défini.

Il n'existe pas de fonction identique en Basic GFA, et ce, malgré les instructions très puissantes **GET/PUT** de saisie et transfert de section d'écran.

Nous vous proposons donc un petit programme de haute performance (rapidité, efficacité), qui comblera cette lacune.

PERFORMANCES

DEGAS est un logiciel formidable et universellement employé ; mais, en ce qui concerne la rotation de block par pas de 90°, nous avons constaté que certains défauts apparaissaient après rotation selon les valeurs en largeur et en hauteur dudit block.

Cette situation est en fait la conséquence regrettable de la méthode de calcul employée par DEGAS faisant appel à la trigonométrie ; méthode en rien discutable pour les angles de rotation n'étant pas multiples de 90° mais malheureuse pour la perfection du résultat ainsi que pour les temps d'exécution.

Pour en faire la démonstration, nous avons choisi un block exemple en figure 1.

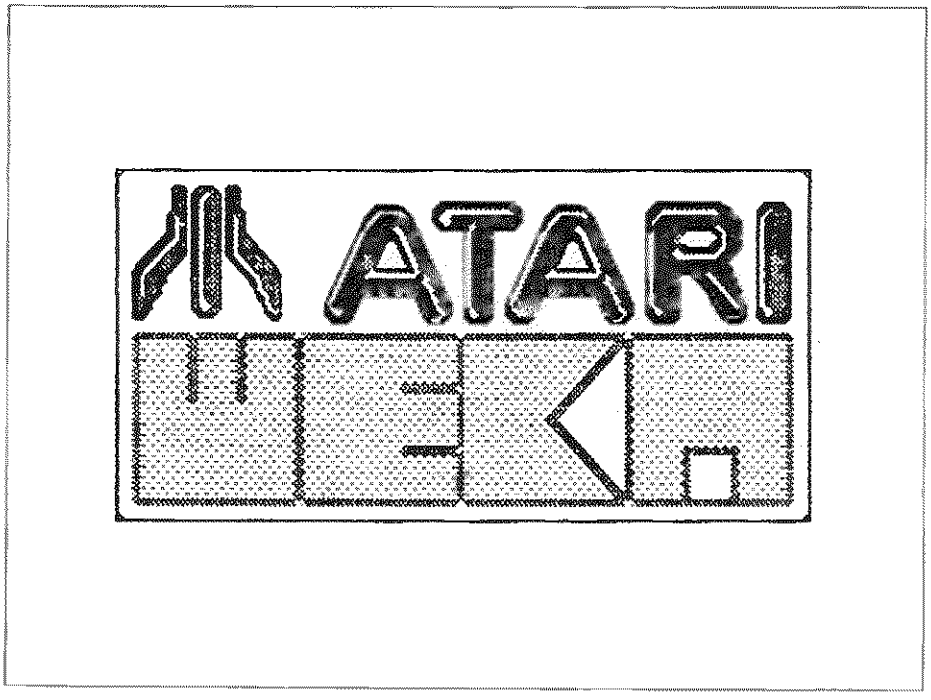


Fig. 1 : Le Block original qui va subir la rotation.

Après saisie par **Esc** pour DEGAS et par **Get** pour le GFA, il a fallu quarante-quatre secondes par Block-Rotate pour fabriquer le block de la figure 2a et moins d'une demi-seconde par Rotablock en figure 2b.

Résultats révélateurs en vérité, le block DEGAS présentant une gênante déformation verticale rendant le block inexploitable en cet état. Quant au block Rotablock, aucun reproche à formuler.

Ces résultats ont dû vous convaincre d'entreprendre la frappe du programme proposé.

Avant cela, nous allons effectuer un flash-back sur les systèmes numériques puis étudier ce qui se passe dans la mémoire lorsque l'on dessine à l'écran et la structure d'une variable **Get**.

LE CODAGE NUMÉRIQUE AU FORMAT BINAIRE

Pour certains d'entre vous, compter ne peut se faire qu'avec le système décimal. L'idée de compter avec des lettres voire avec seulement des « O » et des « 1 » leur semblerait impossible.

Et pourtant, outre le système décimal, il existe le système HEXADECIMAL, le système BINAIRE et le système OCTAL.

Le système hexadécimal a pour base le nombre « 16 » (« Hexa » = 6 + « déci » = 10) ; les nombres hexadécimaux de 0 à 9 y sont représentés de la même manière qu'au format décimal. Les nombres de 10 à 15, par contre, sont représentés par des lettres :

— 10 => A

— 11 => B

— 12 => C

— 13 => D

— 14 => E

— 15 => F

Pour exemple le nombre Hexa F3B0 se décompose de la manière suivante :

« F » = 15 * (16 puissance 3) = 61440

« 3 » = 3 * (16 puissance 2) = 2816

« B » = 11 * (16 puissance 1) = 176

« 0 » = 0 * (16 puissance 0) = 0

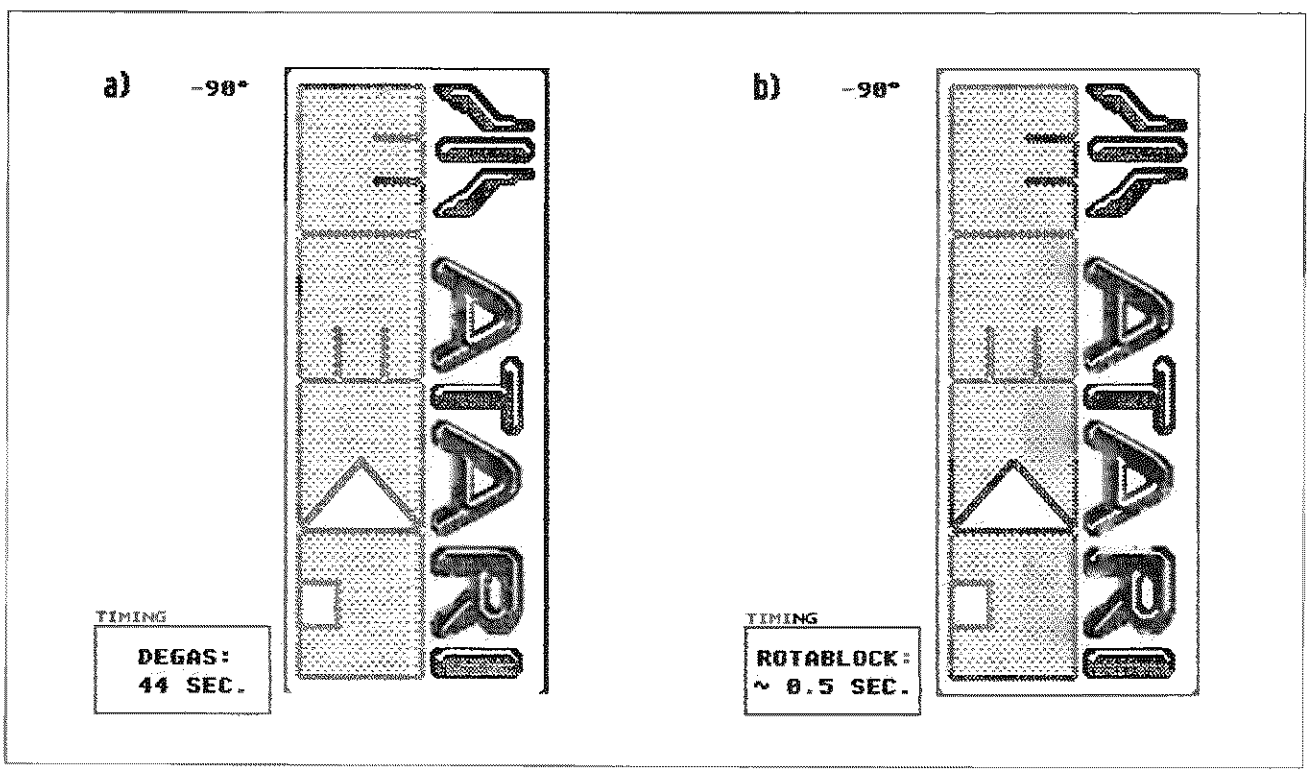


Fig. 2 : Block après rotation. En « a », par Degas, en « b », par Rotablock, (avec les temps d'exécutions).

En décimal, le résultat est égal à la somme des décompositions, soit :

F3B0 = 64432

4 car. 5 car.

Cette particularité de rétrécissement du nombre en fait le système le plus employé en informatique pour les listings de programmes (c'est le cas de Rotablock...).

Pour vérifier, chargez votre Basic GFA et faites :

Print Hex$(64432) puis "Run".

Le système binaire a, bien entendu, pour base 2. Tous les ordinateurs ne comprennent que ce système.

La mémoire de l'Atari ST est organisée sous forme d'octets, soit huit bits ; mais étant donné que notre ordinateur préféré est conçu autour d'un micro-processeur 68000 16/32 bits, les octets sont couplés pour former alors ce que l'on nomme un MOT.

Un mot étant donc égal à 16 bits, vous trouverez en figure 3 un tableau d'interprétation binaire ainsi qu'un exemple explicatif.

Dans la pratique, un « 1 » veut dire qu'une tension circule sur la broche impliquée, alors qu'un « 0 » signale l'absence de tension.

Les mémoires RAMs ou ROMs possèdent ce que l'on nomme un bus d'adresses. Comme dans une véritable agglomération, à chaque adresse correspond une boîte aux lettres.

Pour mieux comprendre les échanges processeur/mémoires, imaginons que dans chaque boîte se trouve un message codé sous forme binaire sur seize bits (un mot). Si l'adresse concernée fait partie de la mémoire d'exploitation (une ZI !), le message est en fait un ordre que le micro-processeur comprendra.

Par exemple, supposons qu'à l'adresse 20, le processeur lit le message codé « 0110000000000100 », 6004 en hexa, ce qui veut dire « va à l'adresse suivante + 4 ! ». Bête et discipliné, le processeur va poursuivre sa tournée à l'adresse 26 (adresse suivante − 22 + 4 = 26).

Si l'adresse concernée ne fait pas partie du système d'exploitation, le processeur se sert des boîtes pour y déposer ou venir y chercher des mots. Parfois le processeur se partage ces boîtes avec d'autres congénères, c'est le cas du processeur vidéo qui vient y chercher ce qu'il doit afficher sur l'écran. Essayez donc ceci :

Dpoke Xbios(2),&x0101010101010101
(Xbios(2) = l'adresse du début de la mémoire écran)

Envoyez **Run**, le micro-processeur dépose dans la boîte d'adresse du début de la mémoire d'écran le mot binaire 0101010101010101, pas le temps de faire ouf ! que le processeur vidéo est allé l'afficher sur l'écran. Observez donc votre moniteur en haut et à gauche...

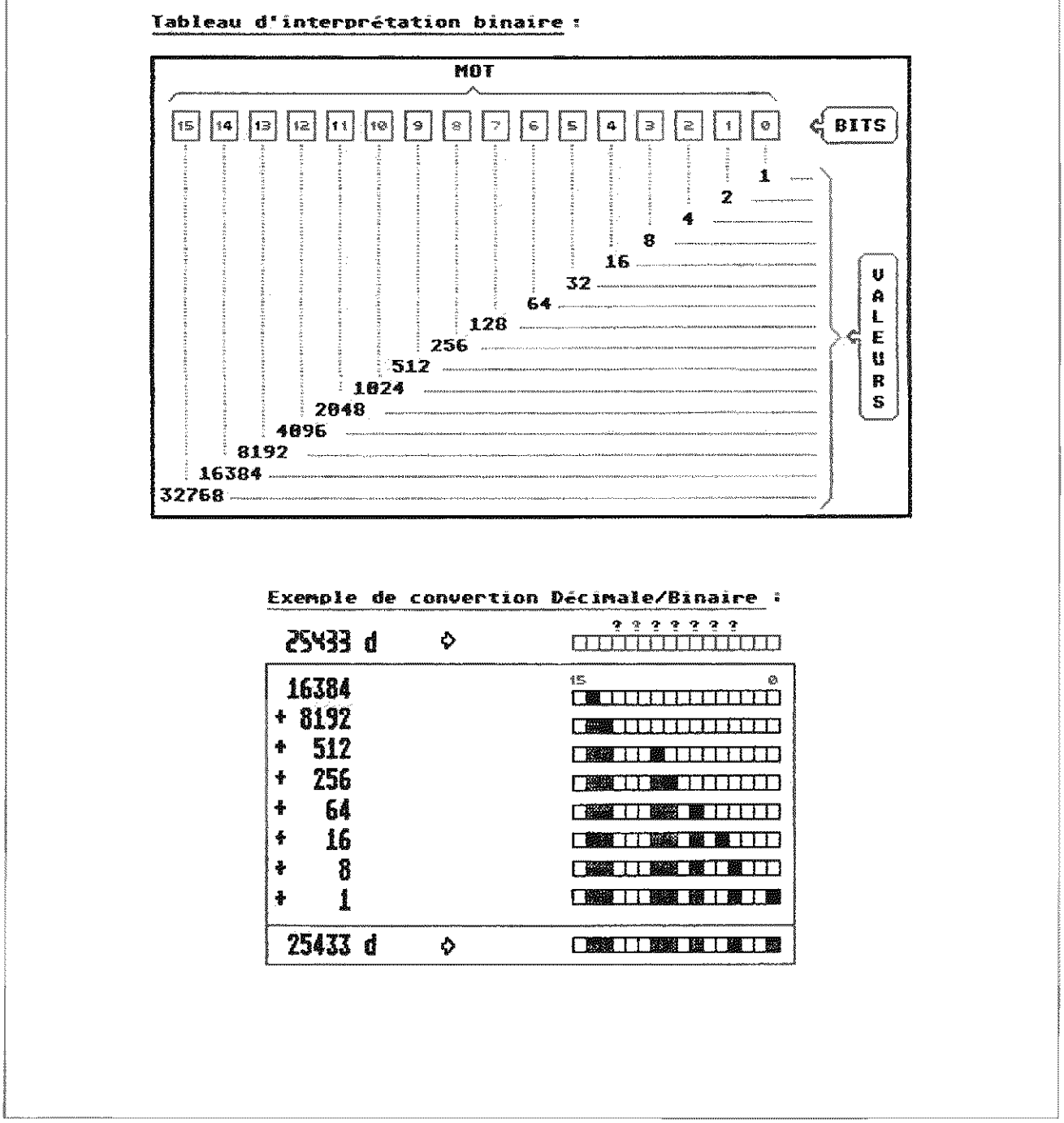


Fig. 3 Rappels sur le codage numérique au format binaire.

ORGANISATION DE LA MÉMOIRE D'ÉCRAN

Taper :

```
Cls
Box 11,2,32,10
```

La mémoire d'écran se trouve alors constituée selon la figure 4. On y découvre également la forme de la matrice soit :

40 mots horizontaux sur 400 lignes.

Pour affirmer la figure 4 taper à la suite :

```
Index% = 160
Ad% = Xbios(2) + Index%
Print At(1,3);Right$("0000000000000000" + Bin$(Dpeek(Ad%)),16)
```

Changer les valeurs de l'index (**toujours pair !!**) et recommencer.

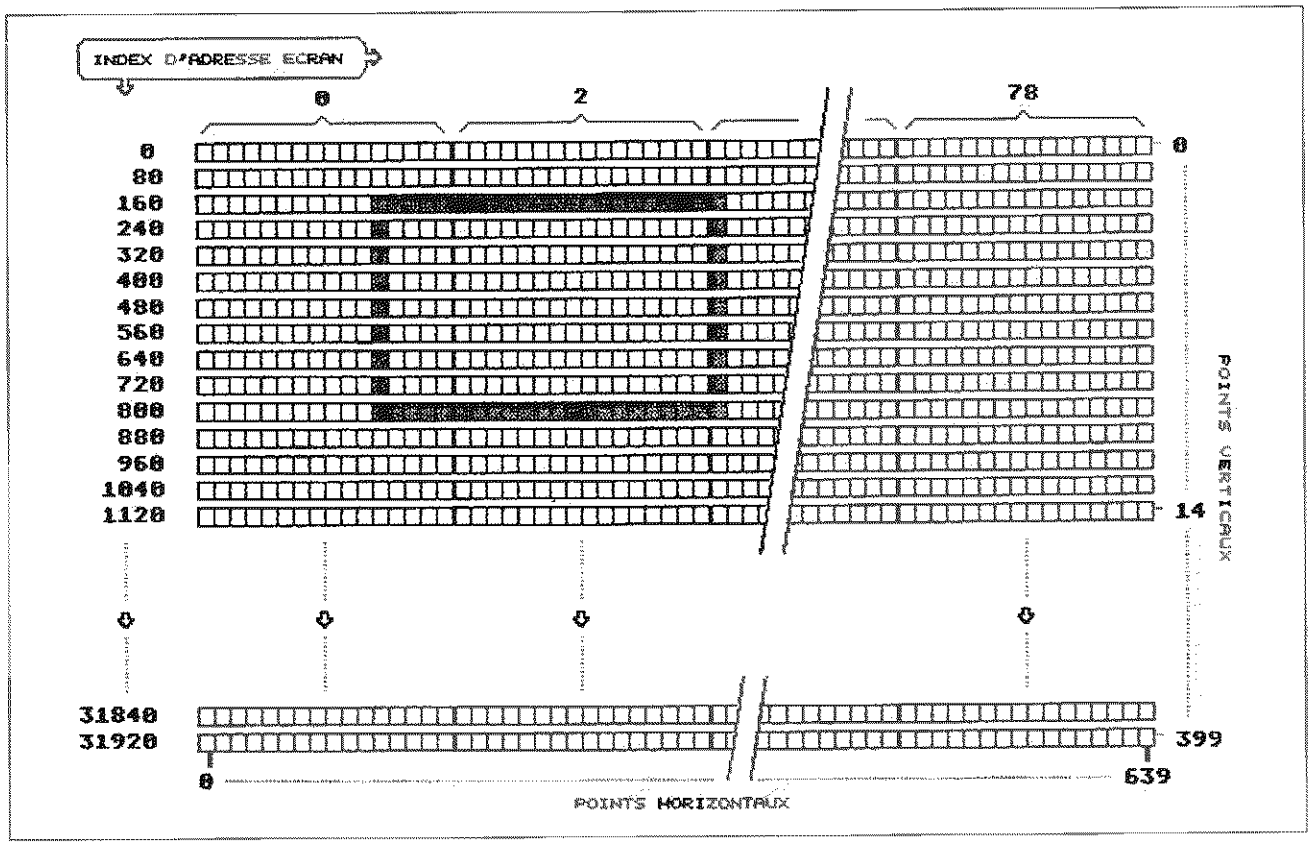


Fig. 4 : Organisation de la mémoire écran après BOX 11,2,32,10.

Etude de la structure d'une variable après saisie par GET

Avant d'aborder le fonctionnement de Rotablock, il est indispensable de bien connaître l'action de l'instruction **GET**. Pour ce faire, garder les quelque deux premières lignes de programme précédemment tapées, soit :

```
Cls
Box 11,2,32,10
```

Ajoutez-y :

```
Get 11,2,32,10,V$
```

A cet instant, **V$** présente l'aspect visualisé en figure 5. Les deux premiers mots de la variable correspondent respectivement à la largeur et la hauteur en points de la zone saisie.

Vérifions :
(Taper **Esc** pour accéder au mode direct du GFA, puis)

```
Print Dpeek(Varptr(V$)),Dpeek(varptr(V$)+2)
```

Nous n'avons pu découvrir la signification du troisième mot qui, quoi qu'il en soit, se trouve toujours fixé à « 1 ».

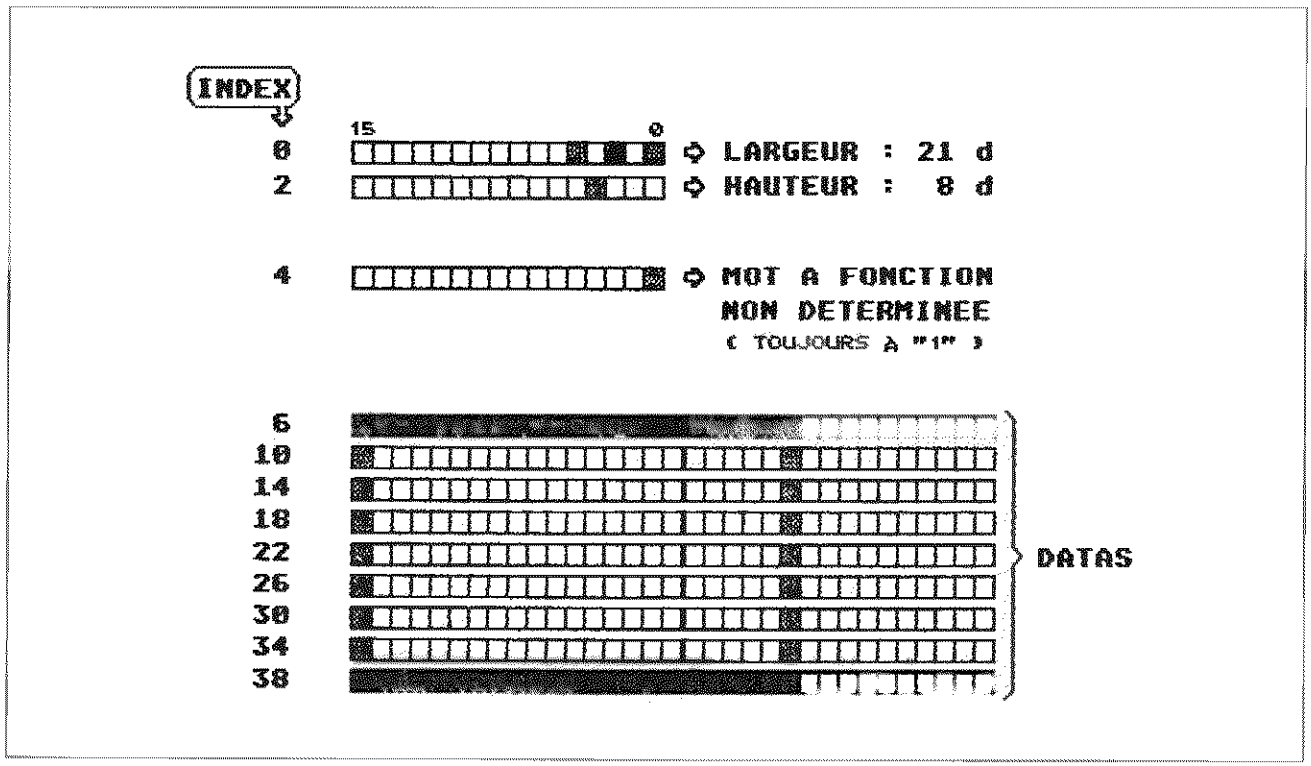


Fig. 5 : Aspect de la variable V$ après saisie effectuée par GET 11,2,32,18,V$.

Les mots suivants contiennent les données de l'image saisie ; à la vue de la figure 5, on peut tirer la conclusion suivante :

> quelles que soient les valeurs en points de la largeur et de la hauteur, la longueur minimale des datas sera d'un mot.

En effet, la matrice des datas est toujours exprimée par un nombre de mots, calculé selon la largeur en points, multiplié par la hauteur en points.

Le calcul de ce nombre de mots s'effectue de la manière suivante :
Nbre_Mots = Quotient de (Largeur/16)

si le reste de (Largeur/16) est <> de 0 :

Nbre_Mots = Nbre_Mots + 1

Dans notre exemple :

Nbre_Mots = 1 + 1 = 2

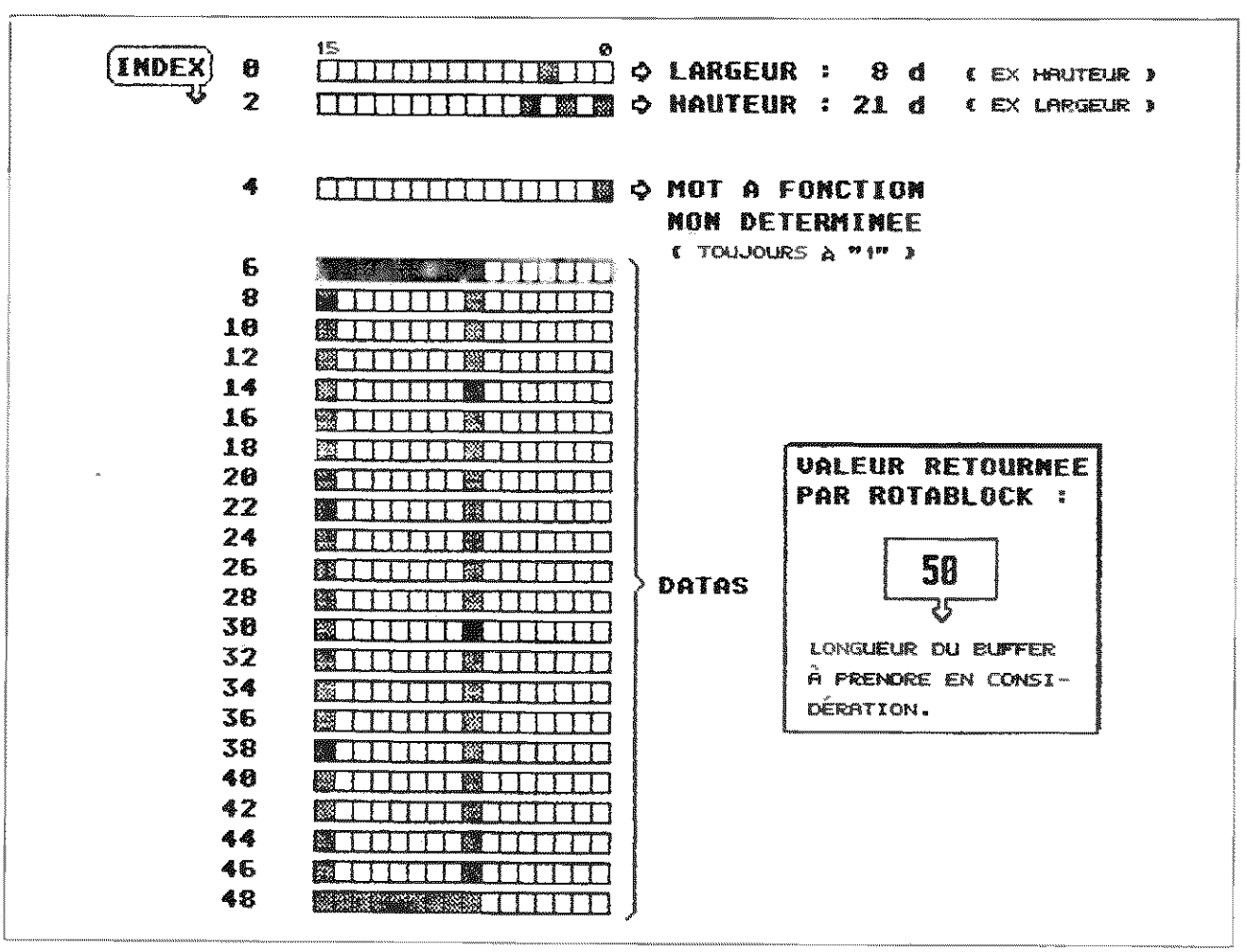


Fig. 6 : Aspect du buffer après rotation de VS.

La longueur des datas en mots est donc égale à :

Nbre_Mots * Hauteur = 2 * 8 = 16 (32 en octets)

Après toutes ces indications, vous ne devriez pas être trop troublé par la figure 6, laquelle nous montre la structure résultant de la rotation tant désirée.

MÉTHODE DE TRAVAIL DE ROTABLOCK

Le principe en a l'air simple bien qu'assez délicat à mettre en œuvre.

Il faut premièrement calculer l'index de la dernière ligne de la matrice. Ceci se fait très facilement puisque nous avons vu que nous pouvions calculer la longueur des datas :

Multiplier par deux cette longueur (l'index représentant toujours un nombre d'octets), et y ajouter enfin 6 (la longueur des trois premiers mots de la variable).

Exemple :

Index =(16 * 2) + 6 = 38

Une fois que l'index est calculé, on commence par tester le quinzième bit du mot s'y trouvant, puis l'on dépose celui-ci en lieu et place du quinzième bit du premier mot du buffer de réception (voir figure 7).

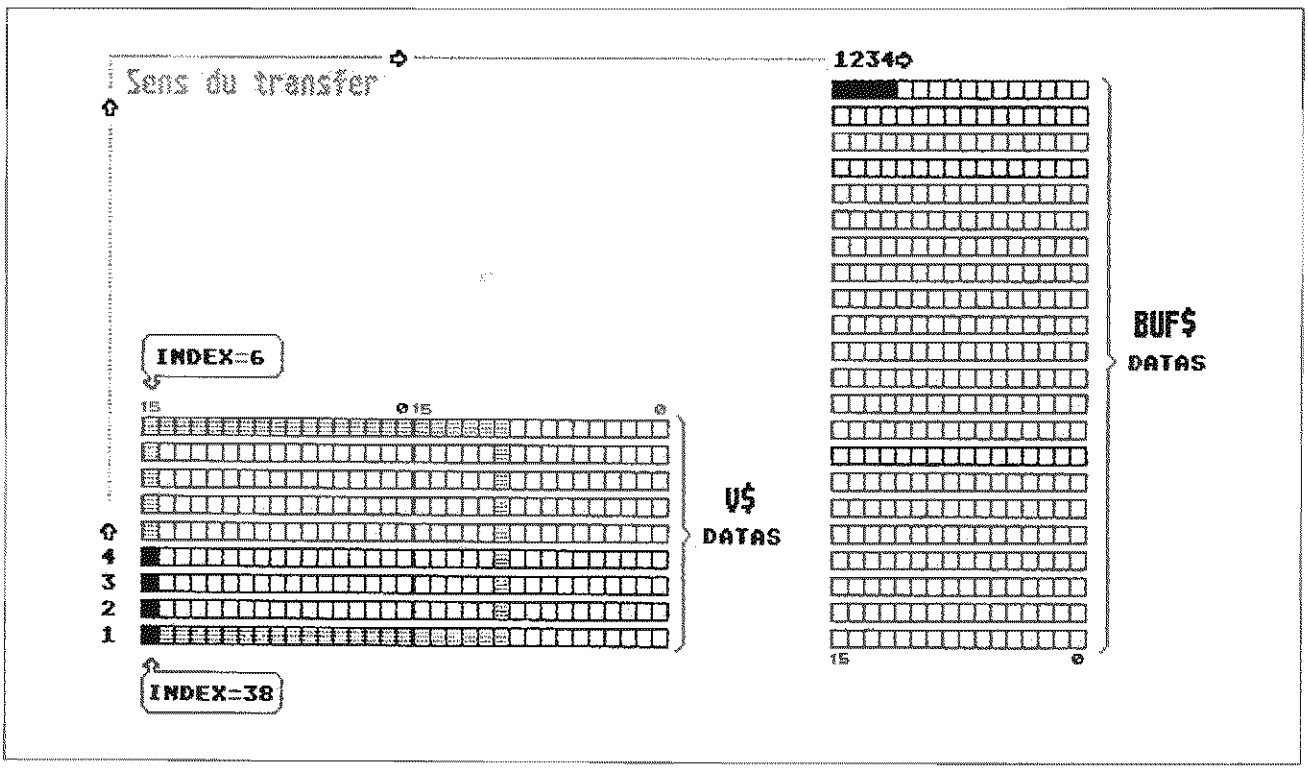


Fig. 7 : Méthode de travail de Rotablock.

Dans la séquence suivante, on teste toujours le 15e bit mais en remontant vers l'index 6 par pas de **-Nbre_Mot** tout en déposant le résultat du test dans le 14e bit et ce jusqu'au bit 0 par pas de valeur **-1**.

Pour bien saisir le processus d'action de Rotablock, voici la traduction littérale en Basic GFA du programme final en assembleur, en vous munissant du listing source désassemblé, vous suivrez clairement et pas à pas le déroulement de la procédure.

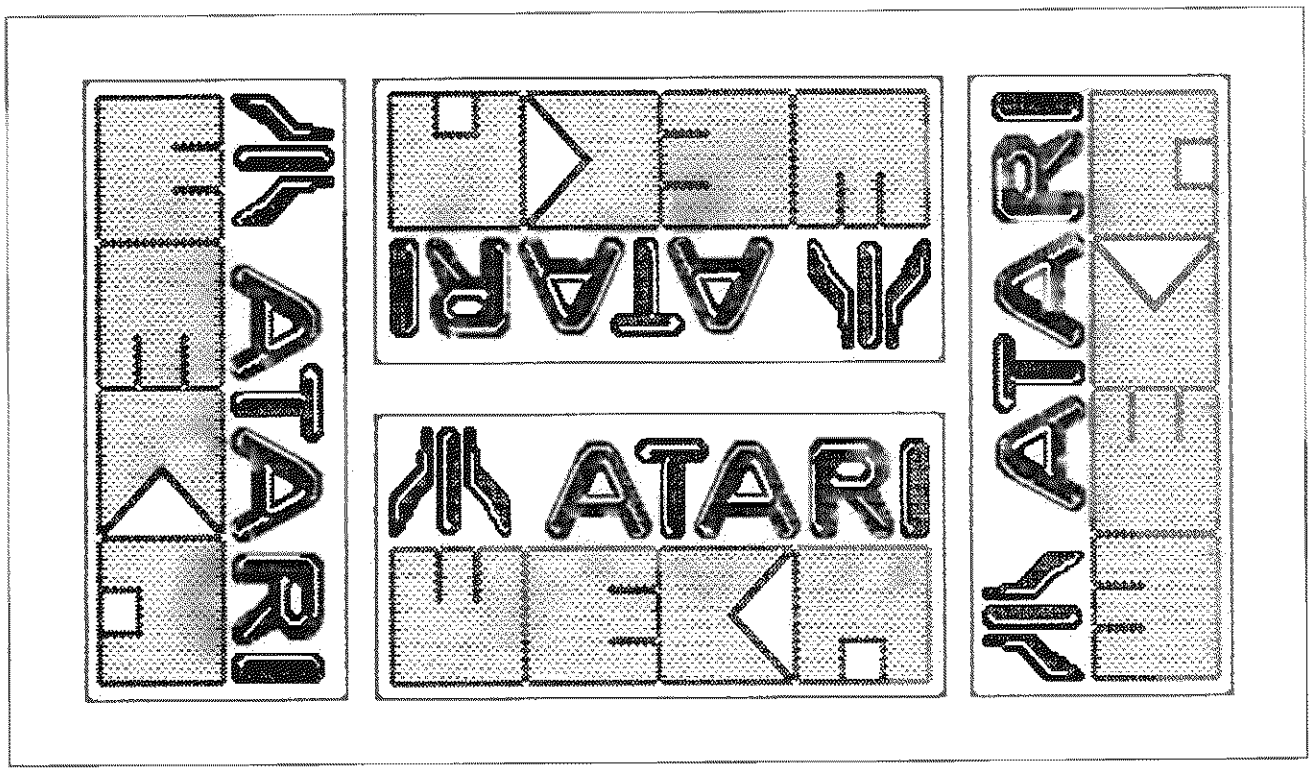
On retrouve ici les lignes responsables du dessin et de la saisie du block initial :

```
Cls
Box 11,2,32,10
Get 11,2,32,10,V$
```

Pour obtenir un résultat, il nous faut un « réceptacle » ! :

```
Buf$=String$ (32000,0)
Cls
```

Après nettoyage de l'écran, on fait appel à la routine Rotablock à laquelle l'on transmet les adresses de **V$** et **Buf$**.



Après plusieurs tours dans Rotablock on obtient la révolution complète du block original....



Gosub Rotablock (Varptr(V$),Varptr(Buf$))

Rotablock envoie en retour la longueur de **buf$** qu'il faudra prendre réellement en compte par la suite :

Buf$ = Left$(Buf$,D0)

Il ne reste plus qu'à afficher le résultat...

Put 0,0,Buf$

Et voilà !

Voyons maintenant la traduction pas à pas de Rotablock :

Procedure Rotablock (Adresse_variable%,Adresse_buffer%)

Si le cœur vous en dit, vous pouvez tester l'efficacité de ce listing, ne serait-ce que pour apprécier la vitesse à laquelle Rotablock en assembleur travaille par rapport en GFA Basic qui jouit pourtant d'une solide réputation à ce niveau.

Nous arrivons au bout de ce long exposé. Il vous suffit de taper les datas qui suivent et de faire tourner les blocks à votre guise. Nous vous avons même concocté un petit listing exemple qui vous permettra de saisir la portion d'écran à torturer de façon fort simple.

```
'*************************************************************************
'*                                                                       *
'*              LISTING SOURCE DES DATAS ROTABLOCKS                      *
'*              ___________________________________                      *
'*                  Auteur : GUISIANO Jm 1990                            *
'*                                                                       *
'*************************************************************************

Data &H4280,&H4281,&H4282,&H206F,&H0004,&H226F,&H0008
Data &H3218,&H3418,&H5488,&H32C2,&H32C1,&H32FC,&H0001
Data &H3F01,&H82FC,&H0010,&H4841,&H3001,&H4241,&H4841
Data &H4A40,&H6702,&H5241,&HC2FC,&H0002,&H3001,&HC2C2
Data &HD1C1,&H2448,&H321F,&H3602,&H7A0F,&H4A45,&H6A06
Data &H548A,&H204A,&H60F4,&H7C0F,&H4244,&H4A46,&H6A04
Data &H32C4,&H60F4,&H3E10,&H0807,&H6702,&H0DC4,&H91C0
Data &H5346,&H51CA,&HFFEA,&H32C4,&H3403,&H204A,&H5345
Data &H51C9,&HFFD0,&H93EF,&H0008,&H2009,&H4E75
'
'
Data FIN
'
'
While Dat$<>"FIN"
  Read Dat$
  If Dat$<>"FIN"
    Let Rotabloc$=Rotabloc$+Mki$(Val(Dat$))
    Add Checksum%,Val(Dat$)
  Endif
Wend
If Checksum%<>120602
  Cls
  Print "ERREUR DANS LA SAISIE DES DATAS !!"
  Print
  Print "       VERIFIEZ SVP."
  End
Endif
Clr Checksum%
Clr Dat$

'  Note  :  Une fois que la checksum% est  correcte,  vous  pouvez
éliminer les lignes figurant en italique avant sauvegarde.

' APPEL DE LA FONCTION :

Buf$=String$(32000,0)
R%=Varptr(Rotabloc$)
Valeur_retournee%=C:R%(L:Varptr(V$),L:Varptr(Buf$))
Buf$=Left$(Buf$,Valeur_retournee%)
```



Exemple d'utilisation de Rotablock

```
'  Ici  doivent  se  trouver  les  Datas  ainsi  que  les  lignes  de
   definition de "Rotablock$".

Repeat
 Mouse X%,Y%,K%
Until K%

Repeat
Until Mousek=0

Graphmode 3
Defline -43691
A1%=X%
B1%=Y%

Box X%,Y%,A1%,B1%

Repeat
 Mouse A%,B%,K%
 If A1%<>A% Or B1%<>B% Or K%
  Box X%,Y%,A1%,B1%
   If K%=0
    Box X%,Y%,A%,B%
    A1%=A%
    B1%=B%
   Endif
 Endif
Until K%

Repeat
Until Mousek=0

Graphmode 1
Defline 1
Get X%,Y%,A%,B%,V$

Gosub Rotablock(Varptr(V$))
Aff:                                  'On sort de la boucle par C/S/A

A%=Mousex
B%=Mousey

Put A%,B%,Buf$,6
```

```
Repeat
 Mouse X%,Y%,K%
 If X%<>A% Or Y%<>B% Or K%
  Put A%,B%,Buf$,6
  If K%=0
   Put X%,Y%,Buf$,6
   A%=X%
   B%=Y%
  Endif
 Endif
Until K%
Put X%,Y%,Buf$
V$=Buf$
Gosub Rotablock(Varptr(V$))
Goto Aff
End

Procedure Rotablock(Av%)
 Buf$=String$(32000,0)
 Ab%=Varptr(Buf$)
 R%=Varptr(Rotabloc$)
 L%=C:R%(L:Av%,L:Ab%)
 Buf$=Left$(Buf$,L%)
Return
```



```
*****************************************************************
*                  ROUTINE ASSEMBLEUR  68000                    *
*                  ------------------------                     *
*                  Auteur : GUISIANO Jm 1990                    *
*                           Rotablock                           *
*****************************************************************

DEBUT ROUTINE :0000

 0 4280              CLR.L   D0
 2 4281              CLR.L   D1
 4 4282              CLR.L   D2
 6 206F0004          MOVEA.L $4(A7),A0          Adresse Variable
 A 226F0008          MOVEA.L $8(A7),A1            Adresse Buffer
 E 3218              MOVE.W  (A0)+,D1                   Largeur
10 3418              MOVE.W  (A0)+,D2                   Hauteur
12 5488              ADDQ.L  #2,A0
14 32C2              MOVE.W  D2,(A1)+         Newlargeur=Hauteur
16 32C1              MOVE.W  D1,(A1)+         Newhauteur=Largeur
18 32FC0001          MOVE.W  #$1,(A1)+                  3° Mot..
1C 3F01              MOVE.W  D1,-(A7)             Sauve Largeur

1E 82FC0010          DIVU    #$10,D1        Calcul de Nbre-mots
22 4841              SWAP    D1
24 3001              MOVE.W  D1,D0
26 4241              CLR.W   D1
28 4841              SWAP    D1
2A 4A40              TST.W   D0
2C 6702              BEQ.S   $30
2E 5241              ADDQ.W  #1,D1

30 C2FC0002          MULU    #$2,D1           Calcul de l'index
34 3001              MOVE.W  D1,D0
36 C2C2              MULU    D2,D1
38 D1C1              ADDA.L  D1,A0
3A 2448              MOVEA.L A0,A2        A2=A0=Pointeur start

3C 321F              MOVE.W  (A7)+,D1         Restitue Largeur
3E 3602              MOVE.W  D2,D3
```

Mot source suivant :

```
40 7A0F          MOVEQ   #$F,D5
42 4A45          TST.W   D5          ⇦ BOUCLE 1
44 6A06          BPL.S   $4C
46 548A          ADDQ.L  #2,A2     si 16 bits testés,restore
48 204A          MOVEA.L A2,A0       pteur source+2 & goto
4A 60F4          BRA.S   $40           Mot_source_suivant.
```

Mot dest suivant :

```
4C 7C0F          MOVEQ   #$F,D6
4E 4244          CLR.W   D4          ⇦ BOUCLE 2
50 4A46          TST.W   D6
52 6A04          BPL.S   $58       si 16 bits positionnés,
54 32C4          MOVE.W  D4,(A1)+      on stoke D4 et goto
56 60F4          BRA.S   $4C             Mot_dest_suivant.

58 3E10          MOVE.W  (A0),D7           D7=mot à tester.
5A 0B07          BTST    D5,D7
5C 6702          BEQ.S   $60                     On teste,
5E 0DC4          BSET    D6,D4             et l'on modifie.

60 91C0          SUBA.L  D0,A0         On remonte d'un pas.
62 5346          SUBQ.W  #1,D6
64 51CAFFEA      DBF     D2,$50              Si les lignes ne

                                     sont pas toutes testées,
                                           goto boucle 2.

68 32C4          MOVE.W  D4,(A1)+        sinon stocker D4,
6A 3403          MOVE.W  D3,D2
6C 204A          MOVEA.L A2,A0                   Restore A0
6E 5345          SUBQ.W  #1,D5
70 51C9FFD0      DBF     D1,$42        Si les colonnes ne
                                     sont pas toutes testées,
                                           goto boucle 1.

74 93EF0008      SUBA.L  $8(A7),A1     Calcul de la valeur
78 2009          MOVE.L  A1,D0                 retournée,
7A 4E75          RTS                     & Retour basic.
```



5/5.5

Julia, Mandelbrot et compagnie

Une des grandes orientations mathématiques de notre temps fut l'étude des fractales associées aux mouvements chaotiques.

Les différentes études menées sur ces objets ont montré que les formes géométriques couramment étudiées en mathématiques sont des cas exceptionnels qui sous-tendent en la régularité des formes de ces objets.

Or, il en est tout autrement : quand on étudie très précisément les formes existantes dans la nature, rares sont celles possédant une régularité parfaite, et pour lesquelles il serait possible de tracer une tangente en chaque point ; de plus, il est possible de définir des dimensions différentes de celles couramment connues : des dimensions intermédiaires entre les dimensions 1, 2 et 3, voir plus.

Avant de nous lancer dans une étude théorique des fractales et du chaos, nous allons pour l'instant essayer de faire comprendre au lecteur la notion de forme fractale par une réflexion et une étude d'ensembles fractals, tout en signalant de suite que les objets réels ne sont pas exactement fractals, mais peuvent souvent être représentés par un modèle fractal.

De la complexité de la Bretagne

Une première réflexion concernant les fractales fut proposée par le mathématicien Benoît Mandelbrot dans son livre *« Les objets fractals »*, en étudiant la mesure d'une côte à l'aspect très irrégulier comme celle de la Bretagne.

Si vous effectuez cette mesure à l'aide d'un mètre, avec beaucoup de courage, vous trouveriez une longueur approximative X.

Le courage aidant, vous recommencez cette mesure avec un double décimètre, pour trouver cette fois-ci une mesure Y telle que cette mesure soit supérieure à X, le contour des grosses pierres entrant en jeu.

Si cette fois-ci, avec beaucoup d'imagination, vous vous placiez dans la peau d'une souris, vous trouveriez une tout autre mesure, tenant compte cette fois-ci des petits cailloux.

Réduisez-vous encore, en passant par la taille d'une mouche, puis d'une puce, puis d'une bactérie. A chaque mesure prise, vous augmenteriez encore la mesure totale de la côte de la Bretagne.

En voyageant sans limite dans le domaine de l'infiniment petit, vous pouvez imaginer que la longueur de la côte de la Bretage tend vers l'infini, de quoi rendre fou un constructeur s'évertuant à créer une digue très précise sur une partie de cette côte.

L'OBJET DE NOTRE ÉTUDE

Comme nous vous l'avons annoncé, nous n'allons pas entrer pour l'instant dans toutes les considérations théoriques concernant les fractales, mais profiter des études mathématiques de trois mathématiciens connus : Pierre Fatou et Gaston Julia (qui ont travaillé sur des ensembles bizarres au début de ce siècle) et Benoît Mandelbrot, chercheur mathématicien plus récent, qui a permis l'accélération des études sur les objets fractals à partir de la parution de son livre « *Les objets fractals* » en 1975.

Le bond fulgurant des fractales fut possible grâce à l'avènement des ordinateurs puissants, et contrairement à ce que l'on pourrait croire, ne pris pas son ampleur dans le domaine scientifique, mais dans le domaine artistique.

Il fut ainsi possible, grâce à de nombreuses études, de créer des paysages, tout en relief, qui, bien que totalement imaginaires, possédaient un réalisme fort troublant. Troublant au point de conforter certains dans leur théorie de génération fractale de notre univers (bien que le chaos y est fortement pour quelque chose, mais nous aurons probablement l'occasion d'en reparler et de découvrir que chaos et fractals sont fortement liés).

Notre première orientation sera ainsi dans ce chapitre essentiellement esthétique, par le tracé de graphismes colorés sur les ensembles dits de Julia et de Mandelbrot.



LE PLAN COMPLEXE

Bien que d'une orientation, dans un premier temps, non scientifique, il est nécessaire de posséder quelques notions de calcul sur les nombres dits complexes et notamment sur le plan complexe.

Le plan complexe

Sans entrer dans la thérorie des nombres complexes, il est possible de comprendre très simplement le fondement du plan complexe, par la représentation de la figure 1.

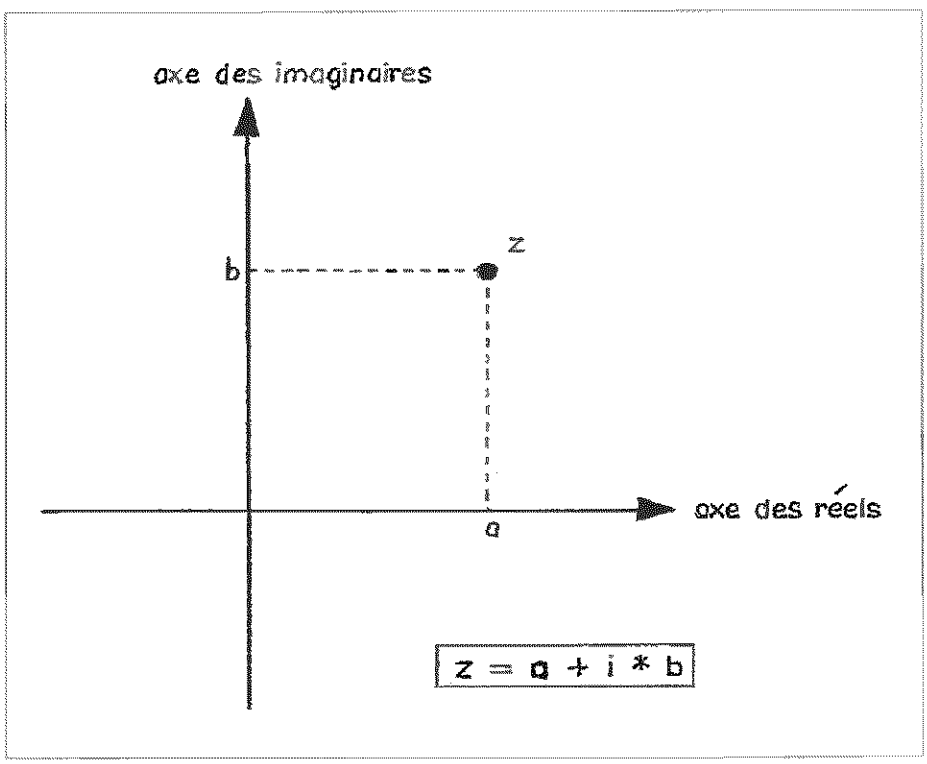


Fig. 1 : Le plan complexe.

Dans le plan usuel, d'un fonctionnement similaire à l'écran de votre Atari, un point quelconque (un pixel de l'écran) est déterminé par ses coordonnées a et b :

z (a, b)

avec : a = abscisse du point z
b = ordonnée du point z

Pour le plan complexe, on associe à ce point un nombre, dit complexe, déterminé par la valeur :

$$z = a + i * b$$

avec a = partie réelle du nombre z
b = partie imaginaire du nombre z

Calculs sur les nombres complexes

Vous avez vu apparaître précédemment la constante **i** (notée très souvent j par les électroniciens, pour ne pas la confondre avec le symbole du courant).

Cette constante est un nombre inventé par les mathématiciens pour les besoins du calcul, et qui a pour particularité, lorsqu'on l'élève au carré, de donner la valeur - 1.

$$i^2 = -1$$

Il est ainsi possible de définir, comme sur les nombres classiques toutes les formes de calcul (addition, multiplication, exponentielles,...), dont nous allons ci-dessous exposer certaines d'entre-elles, sachant que l'on essaye toujours de regrouper les parties réelles, ainsi que les parties imaginaires, entre-elles.

• *L'addition*

Soit deux nombres complexes $z1 = a1 + i * b1$ et $z2 = a2 + i * b2$.

$$\begin{aligned} z1 + z2 &= a1 + i * b1 + a2 + i * b2 \\ &= (a1 + a2) + i * (b1 + b2) \end{aligned}$$

La somme de deux nombres complexes est ainsi un nombre complexe dont la partie réelle est égale à la somme des parties réelles, et la partie imaginaire est égale à la somme des parties imaginaires.

• *La soustraction*

De la même façon, on obtiendra pour la soustraction :

$$z1 - z2 = (a1 - a2) + i * (b1 - b2)$$

• *La multiplication*

En prenant toujours deux nombres complexes z1 et z2, il est possible de les combiner pour en effectuer la multiplication :

$$\begin{aligned} z1 * z2 &= (a1 + i * b1) * (a2 + i * b2) \\ &= a1 * a2 + a1 * i * b2 + i * b1 * a2 + i * b1 * i * b2 \\ &= a1 * a2 + i * (a1 * b2 + a2 * b1) + i^2 * b1 * b2 \end{aligned}$$

N'oublions pas que $i^2 = -1$, ce qui donne :

$$\begin{aligned} z1 * z2 &= a1 * a2 + i * (a1 * b2 + a2 * b1) - b1 * b2 \\ &= (a1 * a2 - b1 * b2) + i * (a1 * b2 + a2 * b1) \end{aligned}$$



On retrouve ici encore un nombre complexe, d'une forme tout de même relativement compliquée (ou complexe ?).

Etudions de suite le cas où z1 = z2 = z = a + i * b :

$$z * z = z^2 = (a * a - b - b * b) + i * (a * b + a * b)$$
$$= (a^2 - b^2) + i * 2 * a * b$$

• *Le conjugué d'un complexe*

Les nombres complexes ont une particularité, celle de posséder un conjugué. Quand on multiplie un complexe par son conjugué, on obtient un nombre réel.

Le conjugué de z = a + i * b est :

$$z' = a - i * b$$

ainsi :

$$z * z' = (a + i * b) * (a - i * b)$$
$$= (a * a - b * (-b)) + i * (a * (-b) + a * b)$$
$$= (a^2 + b^2) + i * (-a * b + a * b)$$
$$= a^2 + b^2$$

• *La division de deux complexes*

En prenant toujours nos deux nombres complexes z1 et z2, nous pouvons effectuer la division de z1 par z2 :

$$\frac{z1}{z2} = \frac{a1 + i * b1}{a2 + i * b2}$$

Afin de pouvoir séparer les parties réelles et les parties imaginaires, il est nécessaire de multiplier le numérateur (z1) et le dénominateur (z2) par le conjugué du dénominateur (z2' = a2 - i * b2), ce qui nous donne :

$$\frac{z1}{z2} = \frac{(a1 + i * b1)(a2 - i * b2)}{(a2 + i * b2)(a2 - i * b2)}$$

$$= \frac{(a1 * a2 + b1 * b2) + i * (a1 * b2 - a2 * b1)}{a2^2 + b2^2}$$

$$= \frac{a1 * a2 + b1 * b2}{a2^2 + b2^2} + i * \frac{a1 * b2 - a2 * b1}{a2^2 + b2^2}$$

• *Le module d'un nombre complexe*

Afin de pouvoir classer les nombres complexes, il a été introduit la notion de module d'un nombre complexe, défini comme la racine carré de la somme des carrés de ses parties réelles et imaginaires :

$$|z| = \sqrt{a^2 + b^2}$$

$$= (a^2 + b^2)^{1/2}$$

Le module d'un nombre complexe correspond en fait à sa distance par rapport à l'origine.

C'est ainsi que tous les nombres complexes possédant le même module se trouvent sur un cercle, centré sur l'origine, et dont le rayon est égal à ce module.

Les ensembles de Julia et domaines de Fatou

Au début de ce siècle, et plus précisément entre 1917 et 1919, les deux mathématiciens Gaston Julia et Pierre Fatou posèrent leurs efforts sur l'étude d'objets mathématiques que l'on appelle maintenant *« Ensembles de Julia »* et *« Domaines de Fatou »*.

Ces ensembles sont obtenus par itérations successives sur une fonction : en considérant une fonction f(x), à partir d'une valeur initiale x0, on calcule la valeur x1 = f(x0), puis x2 = f(x1), puis x3 = f(x2) et ainsi de suite jusqu'à l'infini.

En fonction des résultats obtenus, on tente de tracer l'ensemble en faisant répondre les nombres calculés à certaines conditions.

Une fonction complexe

La fonction complexe déterminant les ensembles dits de Julia est la suivante :

$$f(z) = z^2 + \text{constante complexe}$$

En prenant ainsi une valeur fixe de constante, il suffit d'appliquer l'itération précédemment décrite pour tous les points z du plan complexe, ce qui nous donne l'algorithme suivant :

- DEBUT
  - Donner une valeur complexe à la constante
  - POUR chacun des points du plan complexe
    - ITERER
      - Calculer $z_n = z_{n-1}^2 + \text{constante}$
    - FINITERER
  - FINPOUR
- FIN

On obtiendra ainsi pour les premières itérations :

$z1 = z0^2 + Cte$ (Cte = constante complexe)
$z2 = (z0^2 + Cte)^2 + Cte$
$z3 = ((z0^2 + Cte)^2 + Cte)^2 + Cte$
$z4 =$ ... etc.

Surtout, ne prenez pas de suite vos jambes à votre cou, en vous disant : « Mais je n'aurai pas la patience d'attendre, pour l'infinité des points du plan complexe, l'infinité d'itérations à effectuer, sans compter qu'il existe une infinité d'ensembles de Julia, au vu de l'infinité du nombre de constantes. »

Il n'est pas question d'étudier une infinité de points, pas plus que d'itérer une infinité de fois cette fonction.

Une proposition d'algorithme

En étudiant de près ces itérations, il s'avère que, certaines valeurs de z (pour une constante donnée), après quelques itérations, produisent des nombres complexes très grands, qui, si on continue l'itération, engendrent des parties réelles et imaginaires atteignant très vite des valeurs élevées.

Si l'on se reporte au module des nombres produits, on s'aperçoit que celui-ci tend vers l'infini.

Par contre, certaines valeurs de z produiront des nombres dont le module restera petit, quel que soit le nombre d'itérations, et aura tendance à se cantoner autour d'une certaine valeur.

La valeur de référence du module est fixée usuellement à 2. Ainsi, tout nombre z dont l'itération infinie produit un nombre dont le module reste inférieur à 2 est considéré comme appartenant à l'ensemble ; inversement, dès que le module d'un nombre complexe calculé dépasse la valeur 2, le nombre complexe initial est considéré hors de l'ensemble.

Dans la pratique, on n'attendra jamais l'infinité d'itération, cela se comprend, et on fixera un nombre d'itérations, en général compris entre 100 et 1 000. Sachant que, plus ce nombre est grand, plus la précision de la détermination de l'appartenance ou non d'un point du plan complexe à l'ensemble de Julia sera grande.

En ramenant la représentation du plan complexe à l'écran de votre Atari, il sera possible de colorier chacun des plans en fonction de cette appartenance, par exemple en noir, si le nombre appartient à l'ensemble, en blanc sinon.

La détermination des points initiaux du plan complexe s'effectuera en balayant les coordonnées x et y de chacun du plan, entre deux valeurs extrêmes, pour chacun d'eux.

L'algorithme devient ainsi :

- DEBUT
  - Fixer une valeur de constante (Cte)
  - POUR x0 DE valeur initiale A valeur finale
    - POUR y0 DE valeur initiale A valeur finale
      - TANTQUE module calculé inférieur à 4 et nombre maximum d'itération non atteint
        - Calculer $z_n = z_{n-1}^2 + Cte$
          - CO
            en commençant par z0 = x0 + i * y0
          - FINCO
        - Calculer module de z_n
      - FINTANTQUE
      - SI le module est inférieur à 4
        - Colorier le point correspondant en noir
      - SINON
        - Colorier le point correspondant en blanc
      - FINSI
    - FINPOUR
  - FINPOUR
- FIN

Le calcul de l'itération s'effectuera bien sûr en appliquant les formules donnant les parties réelles et imaginaires, sous la forme suivante :

$$a_n = a_{n-1}^2 - b_{n-1}^2 + x$$
$$b_n = 2 * a_{n-1} + b_{n-1} + y$$

si la constante Cte s'écrit : $x + i * y$.

Vous pouvez de suite appliquer cet algorithme avec un langage structuré, en fixant pour limites de l'ensemble complexe, les valeurs -2 à 2 pour la partie réelle, ainsi que pour la partie réelle des nombres étudiés, et en prenant des valeurs de constantes très petites.

Il suffira ensuite d'introduire un algorithme de conversion de ces points en coordonnées écran.

Mais cette version monochrome reste limitée et ne fournit que ce que l'on appelle une version pointilliste des ensembles de Julia, telle que représentée en figure 2.

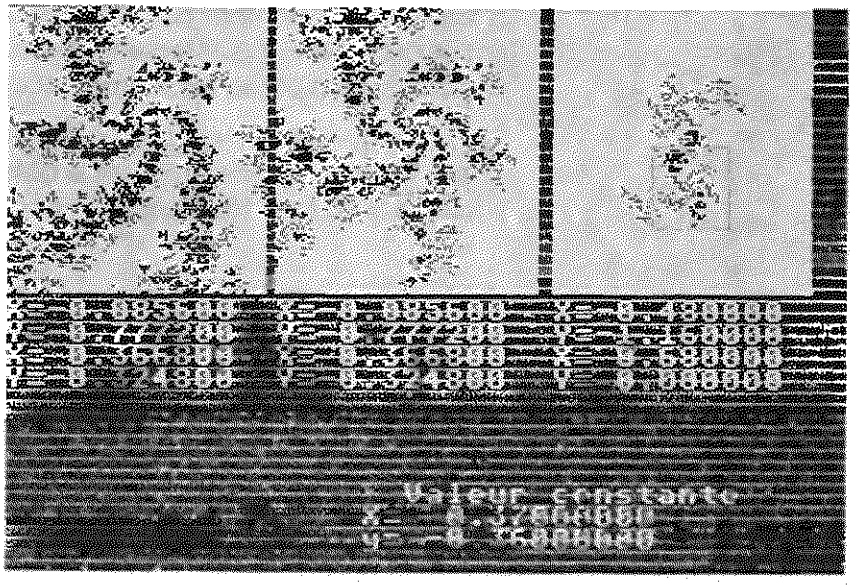


Fig. 2 : Une version pointilliste de Julia pour une précision de 100.

Pour aller plus loin graphiquement

L'étude monochromatique des ensembles de Julia, bien que déterminant très fidèlement les limites de ceux-ci, ne possède tout de même pas un grand attrait graphique.

En étudiant de près les différentes progressions des itérations de chaque points du plan complexe, il s'avère que tous ceux qui s'échappent vers l'infini ne le font pas avec la même rapidité.

Lors d'une itération, les différents modules des différentes valeurs produites peuvent rester longtemps à une valeur inférieure à 2 avant de la dépasser, on appelle cela la « latence » du point. Cette latence peut-être déterminée en connaissant le nombre d'itérations effectuées.

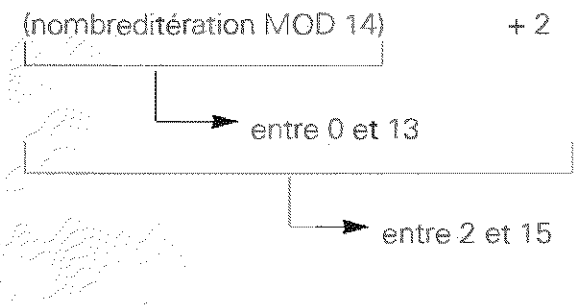
En fonction de cette latence, il sera possible de colorier le point correspondant avec une certaine couleur, différente pour chaque nombre d'itérations.

Le nombre d'itérations pouvant être très grand, et l'Atari ne pouvant afficher qu'un maximum de 16 couleurs, il sera nécessaire de moduler les couleurs en fonction de ce nombre d'itérations.

Moduler... voilà un nom qui rappelle une fonction mathématique : la fonction MODULO.

Cette fonction fournit le reste de la division entière (sans chiffre après la virgule) d'un nombre par un autre. Ainsi, si l'on effectue la division entière du nombre d'itérations effectuées par le nombre de couleurs disponibles, on obtiendra un reste toujours compris entre 0 et 14.

Si l'on désire éliminer les couleurs 0 et 1 (nous nous les réserverons pour le noir, couleur de l'ensemble, et le blanc, couleur d'écriture du texte), et atteindre la couleur 15, il suffira d'effectuer l'opération suivante :



On affectera ainsi la couleur d'affichage du point avec le reste calculé.

Une autre réflexion se pose : comment visualiser, si ce n'est l'infinité, au moins un très grand nombre de points du plan complexe ?

En effet, si nous travaillons sur un carré dont les coordonnées se situent entre - 2 et + 2 pour chacun des axes de coordonnées, nous ne pourrons étudier qu'au maximum, compte tenu du mode d'affichage basse résolution choisi, environ 200 pixels, ce qui nous donne un pas de calcul de :

(+2 - (-2)) / 200 = 4 / 200 = 0,02

Notre programme de recherche

Lors de l'éxécution du programme, vous pourrez vous apercevoir que, aux abords des ensembles, les couleurs se multiplient, et il est alors nécessaire d'utiliser une loupe que nous devrons créer artificiellement.

Pour observer de plus en plus de détails, nous transformerons cette loupe en microscope, les lames et lamelles sur lesquelles reposent les ensembles étant modifiables à l'aide de la souris.

Ayant donc choisi une nouvelle partie du plan complexe à observer, le programme devra de nouveau déterminer le pas de calcul des points à itérer, et appliquer à nouveau l'algorithme.



Dernier problème : il apparaîtra que, selon les goûts de chacun, certaines zones pourront être plus ou moins intéressantes à étudier, or le temps de calcul augmente à la fois avec la précision (nombre d'itérations maximales) et le nombre de points à calculer, ce qui pourrait donner, pour un écran de 200 pixels de côtés et une précision de 1 000 :

200 * 200 * 1 000 = 40 000 000

Ce qui représente un temps non négligeable, même pour un programme en langage machine, surtout pour effectuer des recherches, qui se révèleront quelquefois peu intéressantes.

Aussi, nous avons décidé de limiter notre écran de tracé à un carré de 100 pixels de côtés, visualisé en haut à gauche de l'écran complet (voir la figure 3).

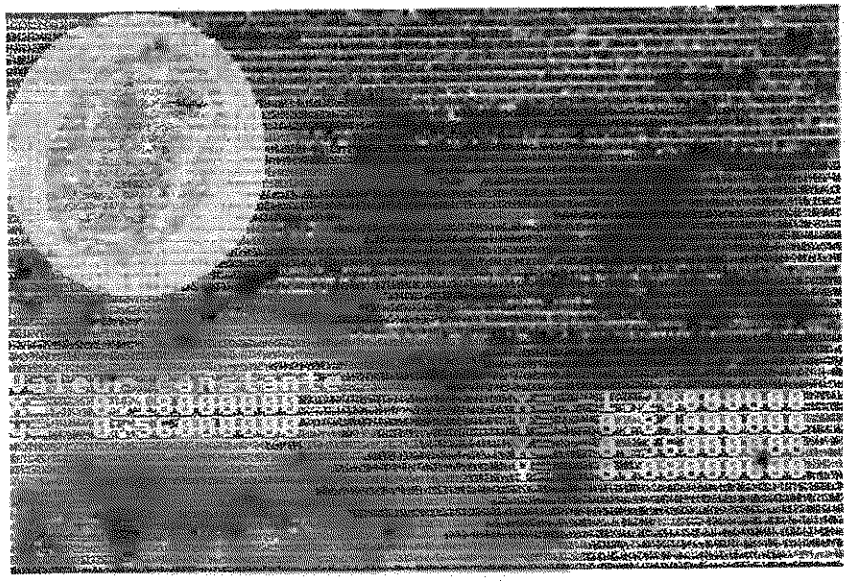


Fig. 3 : Tracé d'un ensemble de Julia pour Cte = 0,18 + *i* * (0,56) et une précision égale à 100.

Sur celui-ci, il sera possible à l'aide de la souris de sélectionner une nouvelle zone de calcul. L'écran précédent sera aussitôt recopié, et les itérations reprendront sur la place libérée. Trois vues pourront ainsi être affichées en même temps, comme représentée sur la figure 4.

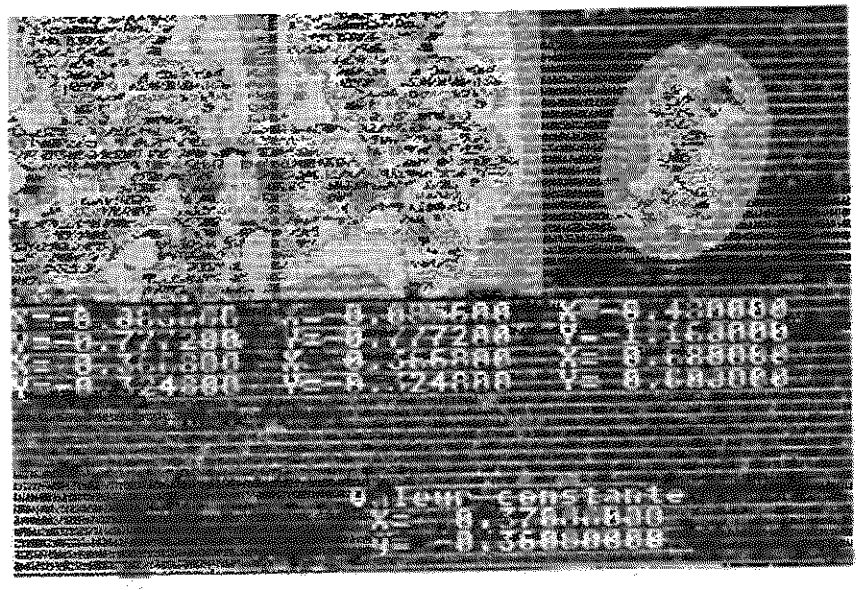


Fig. 4 : Quelques agrandissement d'un ensemble de Julia (précision = 100).

Voici donc, écrit en GFA Basic 3.0, un premier programme qui vous permettra d'effectuer vos premières recherches sur les ensembles de Julia.



```
REM ****************************
REM ** ETUDE DE L'ENSEMBLE **
REM **        DE JULIA        **
REM **        Vers. 2.0       **
REM ****************************
'
'  ----------------
'  initialisations
'  ----------------
'
CLS
'
variable%=XBIOS(4)  ! resolution ecran
IF variable%<>0
  ALERT 3,"Ce programme ne fonctionne qu'en basse resolution
",1,"RETURN",variable%
  EDIT
ENDIF
'
REM precision de calcul
'
REPEAT
  PRINT
  INPUT "Précision du calcul(50 à 1000)";precision%
  PRINT
UNTIL precision%>49 AND precision%<1001
'
REPEAT
  PRINT
  INPUT "nombre de couleurs (2 à 15)";nombre_couleur%
  PRINT
UNTIL nombre_couleur%>1 AND nombre_couleur%<16
'
REM dimensions fenetre d'affichage
'
x_nombre_pixel%=100       ! un carre de
y_nombre_pixel%=100       ! 100 sur 100
'
REM position fenetre d'affichage
'
x_coin_bas_gauche%=0      ! fenetre tout
y_coin_bas_gauche%=100    ! a gauche
'
REM point de depart pour le calcul
'
x_pixel_depart%=x_coin_bas_gauche%-1
y_pixel_depart%=y_coin_bas_gauche%+1
'
'  acquisition de la zone de depart
'
'  ------------------------------------
REM initialisation de la palette
REM de couleur du VDI
'  ------------------------------------
'
IF nombre_couleur%>4
  modification_couleur
```

(suite)

```
ELSE
   IF nombre_couleur%=2
     deux_couleur
   ELSE
     quatre_couleur
   ENDIF
ENDIF
'
'    ------------------------------------
REM acquisition differents parametres
REM          pour les calculs
'    ------------------------------------
'
REM acquisition des coordonnes de depart
'
INPUT "Coordonnee X du point initial";x_initial_julia
INPUT "Coordonnee Y du point initial";y_initial_julia
'
REM acquisition des coordonnees finales
'
INPUT "Coordonnee X du point final   ";x_final_julia
INPUT "Coordonnee Y du point final   ";y_final_julia
'
REM acquisition des coordonnees de la constante
'
INPUT "Constante X ";x_constante
INPUT "Constante Y ";y_constante
'
CLS

DO
   '
   ' verification inferieur-superieur
   '
   IF x_initial_julia>x_final_julia
     SWAP x_initial_julia,x_final_julia
   ENDIF
   IF y_initial_julia>y_final_julia
     SWAP y_initial_julia,y_final_julia
   ENDIF
   '
   ' calcul de proportionnalite
   '
   ratio_x=(x_final_julia-x_initial_julia)/x_nombre_pixel% !
nombre pixels en X
   ratio_y=(y_final_julia-y_initial_julia)/y_nombre_pixel% !
nombre pixels en Y
   '
   ' initialisation sur ecran
   '
   LOCATE 17,22
   PRINT "Valeur constante"
   LOCATE 18,23
   PRINT "X=";USING "###.########",x_constante
   LOCATE 18,24
   PRINT "y=";USING "###.########",y_constante
```



(suite)

```
    '  ------------------------------
    '  calcul de l'ensemble de julia
    '  ------------------------------
    '
    y_julia=y_initial_julia-ratio_y  ! on demarre en dessous
    '                                ! avec rectification dans
    '                                ! la boucle
    '
  FOR y_pixel%=y_pixel_depart% DOWNTO y_pixel_depart%-y_nomb
re_pixel%
    '
      y_julia=y_julia+ratio_y
      x_julia=x_initial_julia-ratio_x  ! on demarre a gauche
      '                                ! pour rectification
      '
      FOR x_pixel%=x_pixel_depart% TO x_pixel_depart%+x_nombre
_pixel%
        x_julia=x_julia+ratio_x
        '
        x_julia_calcul=x_julia
        y_julia_calcul=y_julia
        '
        nombre_iteration%=0
        '
        WHILE nombre_iteration%<precision% AND x_julia_calcul*
x_julia_calcul+y_julia_calcul*y_julia_calcul<4
          '
          x_provisoire=x_julia_calcul*x_julia_calcul-y_julia_c
alcul*y_julia_calcul+x_constante
          '              ! sauvegarde provisoire car la coordonnee
          '              ! x est utilisee pour le calcul de y
          y_julia_calcul=2*x_julia_calcul*y_julia_calcul+y_con
stante
          x_julia_calcul=x_provisoire
          '              ! redonner a x sa valeur
          INC nombre_iteration%
          '              ! une iteration de plus
          '
        WEND
        '
        '  ------------------------
        '  affichage des points
        '  ------------------------
        '
        REM recherche couleur
        '
        IF x_julia_calcul*x_julia_calcul+y_julia_calcul*y_juli
a_calcul<4
          COLOR 0  ! ce point
          '        ! appartient a l'ensemble
        ELSE
          COLOR nombre_iteration% MOD (nombre_couleur%-1)+2
          '        ! le point fuit
          '        ! vers l'infini
        ENDIF
        '
        REM affichage du point calculé
```

(suite)

```
        PLOT x_pixel%,y_pixel%
      NEXT x_pixel%
    NEXT y_pixel%
    REM voir si ESC frappee
    sortie_boucle%=0
    IF INKEY$=CHR$(27)
      ALERT 2,"Arret du calcul en cours?",1,"OUI NON",variable
%
      IF variable%=1
        sortie_boucle%=1
      ELSE
        sortie_boucle%=0
      ENDIF
    ENDIF
    EXIT IF sortie_boucle%=1
    REM sinon nouvelle zone de calcul
    saisie_zone
  LOOP
  ALERT 2,"F= Fin !N= Nouveau ",0," F ! N ",variable%
  CLS
  IF variable%=2
    RUN
  ELSE
    EDIT
  ENDIF
  END
  '
  PROCEDURE saisie_zone
    ' ------------------------------
    ' procedure de saisie d'une
    ' zone carree sur le dessin
    ' ------------------------------
    SHOWM
    DEFMOUSE 5
    COLOR 1
    REPEAT
      ' selection du point initial
      REM mode xor
      GRAPHMODE 3
```



(suite)

```
      MOUSE x_depart%,y_depart%,touche%
      LINE x_depart%,0,x_depart%,199
      LINE 0,y_depart%,319,y_depart%
      REPEAT
        SHOWM
        '
        x_ancien%=x_depart%
        y_ancien%=y_depart%
        '
        MOUSE x_depart%,y_depart%,touche%
        '
        REM reperage coordonnee depart
        '
        LINE x_ancien%,0,x_ancien%,199
        LINE 0,y_ancien%,319,y_ancien%
        LINE x_depart%,0,x_depart%,199
        LINE 0,y_depart%,319,y_depart%
        '
        x_initial_provisoire=x_initial_julia+ratio_x*(x_depart
%-x_coin_bas_gauche%)
        y_initial_provisoire=y_initial_julia-ratio_y*(y_depart
%-y_coin_bas_gauche%)
        '
        REM affichage coordonnee depart
        '
        LOCATE 1,14
        PRINT "X=";USING "##.######",x_initial_provisoire
        LOCATE 1,15
        PRINT "Y=";USING "##.######",y_initial_provisoire
        '
      UNTIL touche%=1
      '
      LINE x_depart%,0,x_depart%,199
      LINE 0,y_depart%,319,y_depart%
      '
      x_ancien%=x_depart%
      y_ancien%=y_depart%
      '
      GRAPHMODE 3
      BOX x_depart%,y_depart%,x_ancien%,y_ancien%
      '
      LOCATE 1,14
      PRINT "X=";USING "##.######",x_initial_provisoire
      LOCATE 1,15
      PRINT "Y=";USING "##.######",y_initial_provisoire
      '
      ' selection du point final
      '
      DELAY 1
      '
      REPEAT
        '
        MOUSE x_fin%,y_fin%,touche%
        '
        y_fin%=y_depart%-(x_fin%-x_depart%)
        x_final_provisoire=x_initial_julia+ratio_x*(x_fin%-x_c
```

(suite)

```
oin_bas_gauche%)
        y_final_provisoire=y_initial_julia-ratio_y*(y_fin%-y_c
oin_bas_gauche%)
        '
        REM affichage coordonnee finale
        '
        LOCATE 1,16
        PRINT "X=";USING "##.######",x_final_provisoire
        LOCATE 1,17
        PRINT "Y=";USING "##.######",y_final_provisoire
        '
        REM dessin zone
        '
        BOX x_depart%,y_depart%,x_ancien%,y_ancien%
        BOX x_depart%,y_depart%,x_fin%,y_fin%
        '
        x_ancien%=x_fin%
        y_ancien%=y_fin%
        '
      UNTIL touche%=1
      '
      IF x_depart%=x_fin% OR y_depart%=y_fin% THEN
        BOX x_depart%,y_depart%,x_fin%,y_fin%
      ENDIF
      '
      DELAY 1
      '
    UNTIL x_depart%<>x_fin% AND y_depart%<>y_fin%
    '
    HIDEM
    '
    REM remettre dans l'ordre les coordonnees
    '
    IF x_fin%<x_depart% THEN
      SWAP x_depart%,x_fin%
    ENDIF
    IF y_fin%<y_depart% THEN
      SWAP y_depart%,y_fin%
    ENDIF
    '
    REM nouvelles coordonnees pour le calcul
    '
    x_initial_julia=x_initial_provisoire
    y_initial_julia=y_initial_provisoire
    x_final_julia=x_final_provisoire
    y_final_julia=y_final_provisoire
    '
    REM mode normal
    '
    GRAPHMODE 1
    '
    REM recopie du dessin
    '
    GET x_nombre_pixel%+1,0,x_nombre_pixel%+x_nombre_pixel%+1,
y_nombre_pixel%+50,dessin$
    PUT x_nombre_pixel%+x_nombre_pixel%+6,0,dessin$
    GET 0,0,x_nombre_pixel%+1,y_nombre_pixel%+50,dessin$
```



(suite)

```
    PUT x_nombre_pixel%+3,0,dessin$
  '
RETURN
'
PROCEDURE modification_couleur
  '
  '  ------------------------------------------
  '  l utilisation de VSETCOLOR permet
  '  d acceder directement aux
  '  registres de couleur du VDI et
  '  ainsi modifier directement sur
  '  les numeros vrais de couleur
  '  ------------------------------------------
  '
  VSETCOLOR 0,0,0,0
  VSETCOLOR 1,15,15,15
  VSETCOLOR 2,0,0,10
  VSETCOLOR 3,0,0,15
  VSETCOLOR 4,15,0,0
  VSETCOLOR 5,0,10,10
  VSETCOLOR 6,0,10,0
  VSETCOLOR 7,0,15,0
  VSETCOLOR 8,10,10,0
  VSETCOLOR 9,15,14,0
  VSETCOLOR 10,15,0,15
  VSETCOLOR 11,0,15,15
  VSETCOLOR 12,10,10,10
  VSETCOLOR 13,15,12,6
  VSETCOLOR 14,15,8,8
  VSETCOLOR 15,15,6,15
  '
RETURN
'
PROCEDURE quatre_couleur
  '
  VSETCOLOR 0,0,0,0
  VSETCOLOR 1,15,15,15
  VSETCOLOR 2,0,0,15
  VSETCOLOR 3,0,15,0
  VSETCOLOR 4,15,0,0
  VSETCOLOR 5,15,0,15
  '
RETURN
'
PROCEDURE deux_couleur
  '
  VSETCOLOR 0,0,0,0
  VSETCOLOR 1,15,15,15
  VSETCOLOR 2,15,15,15
  '
RETURN
'
```

Précisons de suite que ce programme, pour permettre l'accès aux quinze couleurs affichables de l'Atari ne fonctionne qu'en basse résolution.

Le programme vous interroge sur la précision du calcul (entre 50 et 1 000) qui correspond au nombre d'itérations maximum qu'il effectuera avant de décréter si un point du plan complexe appartient à l'ensemble de Julia.

Vous pourrez ensuite préciser le nombre de couleurs affichables aux abords de l'ensemble.

Après initialisation des coordonnées de l'écran d'affichage et de la palette de couleur, il vous est demandé de préciser les coordonnées des points initial et final du plan complexe. Pour un premier calcul sur un nouvel ensemble, vous entrerez dans l'ordre :

- coordonnée X du point initial : - 2
- coordonnée Y du point initial : - 2
- coordonnée X du point final : + 2
- coordonnée Y du point final : + 2

Les ensembles de Julia nécessitant de connaître la valeur d'une constante, celle-ci vous est demandée sous la forme de coordonnées X et Y. Précisons que des valeurs proches de zéro donneront un résultat intéressant.

Le programme vérifie ensuite l'ordre des valeurs données, et si besoin les arrange.

Il effectue ensuite un calcul de proportionnalité pour connaître le pas de calcul correspondant aux coordonnées de l'écran graphique de l'Atari.

La valeur de la constante étant rappelée à l'écran, le calcul commence selon l'algorithme précédemment étudié.

En fin de calcul, le programme teste si la touche ESC à été fappée, pour éventuellement sortir du programme ou effectuer un nouveau calcul.

Sinon, il branche sur une procédure permettant la saisie d'une zone carrée sur l'écran. En cliquant sur le bouton de gauche, vous déterminez la coordonnée du point initial de calcul. Un carré apparaît alors, vous permettant par déplacement de la souris, de choisir le point final de calcul, qui sera validé toujours par le bouton de gauche. Les coordonnées choisies sont affichées sous l'écran de calcul.

Il est alors effectué un décalage sur la droite de deux zones de 100 pixels de côté grâce aux instructions de sauvegarde et de restitution de portion d'écran, respectivement **GET** et **PUT** (à ne pas confondre avec les

instructions de lecture et d'écritures d'enregistrement sur des fichiers à accès aléatoires) dont nous rappelons brièvement le fonctionnement, appliqué à notre programme.

GET xdebut,ydebut,yfin,yfin,variable$** sauvegarde dans la variable alphanumérique **variables$ la portion d'écran dont les limites sont déterminées par **(xdebut,ydebut)** et **(xfin,yfin)**.

PUT xdestination,ydestination,variable$** restitue la portion d'écran sauvegardée dans **variable$ à l'emplacement dont les coordonnées débutent par **(xdestination,ydestination)**.

Une fois les écrans recopiés, le calcul est de nouveau relancé.

En fin de programme se trouvent les procédures de définition de la palette de couleur. On remarquera que les différents registres de couleurs sont affectés à l'aide de l'instruction **VSETCOLOR** qui permet de modifier les numéros vrais des couleurs choisies par l'instruction **COLOR**, car toutes deux travaillent directement avec le VDI, contrairement à **SETCOLOR** qui passe par le TOS.

Pour plus de détails

Lorsque vous aurez déterminé une zone intéressante, comme par exemple sur la figure 5, vous pourrez profiter de tout l'écran pour la visualiser plus en détail. Pour cela, vous devrez noter les coordonnées de visualisation, ainsi que les valeurs des coordonnées de la constante.

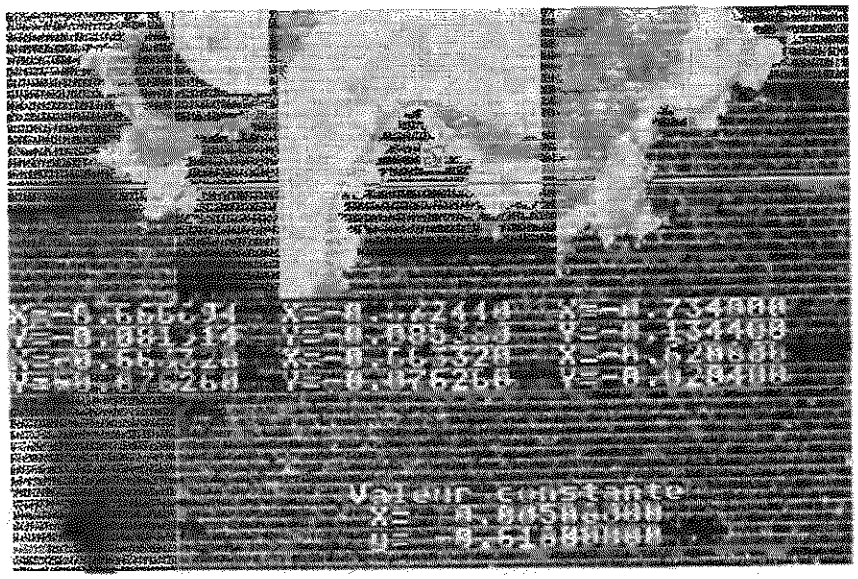


Fig. 5 : De nouveaux détails sur un ensemble de Julia avec une précision égale à 200.

Nous vous proposons donc, ci-après, un programme agrandissant l'image sur un carré de 200 pixels de côté, ce qui vous permettra de visualiser hautement en couleurs des détails tel que celui représenté en figure 6.





```
REM ************************
REM ** ETUDE DE L'ENSEMBLE **
REM **       DE JULIA       **
REM **       Vers. 3.0      **
REM ************************
CLS
variable%=XBIOS(4) ! resolution ecran
IF variable%<>0
  ALERT 3,"Ce programme ne fonctionne qu'en basse resolution
",1,"RETURN",variable%
  EDIT
ENDIF
INPUT "Precision du calcul(50 à 1000)";precision%
x_nombre_pixel%=200
y_nombre_pixel%=200
x_coin_bas_gauche%=70
y_coin_bas_gauche%=190
x_pixel_depart%=x_coin_bas_gauche%-1
y_pixel_depart%=y_coin_bas_gauche%+1
modification_couleur
CLS
INPUT "Coordonnee X du point initial";x_initial_julia
INPUT "Coordonnee Y du point initial";y_initial_julia
INPUT "Coordonnee X du point final  ";x_final_julia
INPUT "Coordonnee Y du point final  ";y_final_julia
INPUT "Constante X ";x_constante
INPUT "Constante Y ";y_constante
CLS
IF x_initial_julia>x_final_julia
  SWAP x_initial_julia,x_final_julia
ENDIF
IF y_initial_julia>y_final_julia
  SWAP y_initial_julia,y_final_julia
ENDIF
ratio_x=(x_final_julia-x_initial_julia)/x_nombre_pixel% ! no
mbre pixels en X
ratio_y=(y_final_julia-y_initial_julia)/y_nombre_pixel% ! no
mbre pixels en Y
y_julia=y_initial_julia-ratio_y
FOR y_pixel%=y_pixel_depart% DOWNTO y_pixel_depart%-y_nombre
_pixel%
  y_julia=y_julia+ratio_y
  x_julia=x_initial_julia-ratio_x
  FOR x_pixel%=x_pixel_depart% TO x_pixel_depart%+x_nombre_p
ixel%
    x_julia=x_julia+ratio_x
    x_julia_calcul=x_julia
    y_julia_calcul=y_julia
    nombre_iteration%=0
    WHILE nombre_iteration%<precision% AND x_julia_calcul*x_
julia_calcul+y_julia_calcul*y_julia_calcul<4
      x_provisoire=x_julia_calcul*x_julia_calcul-y_julia_cal
cul*y_julia_calcul+x_constante
      y_julia_calcul=2*x_julia_calcul*y_julia_calcul+y_const
ante
      x_julia_calcul=x_provisoire
      INC nombre_iteration%
```

(suite)

```
      WEND
      IF x_julia_calcul*x_julia_calcul+y_julia_calcul*y_julia_
calcul<4
        COLOR 0
      ELSE
        COLOR nombre_iteration% MOD 14+2
      ENDIF
      PLOT x_pixel%,y_pixel%
    NEXT x_pixel%
  NEXT y_pixel%
  WHILE INKEY$<>""
  WEND
  REPEAT
  UNTIL INKEY$<>""
  END
  '
  PROCEDURE modification_couleur
    '
    ' ------------------------------------
    ' l'utilisation de VSETCOLOR permet
    ' d'acceder directement aux
    ' registres de couleur du VDI et
    ' ainsi modifier directement sur
    ' les numeros vrais de couleur
    ' ------------------------------------
    '
    VSETCOLOR 0,0,0,0
    VSETCOLOR 1,15,15,15
    VSETCOLOR 2,0,0,10
    VSETCOLOR 3,0,0,15
    VSETCOLOR 4,15,0,0
    VSETCOLOR 5,0,10,10
    VSETCOLOR 6,0,10,0
    VSETCOLOR 7,0,15,0
    VSETCOLOR 8,10,10,0
    VSETCOLOR 9,15,14,0
    VSETCOLOR 10,15,0,15
    VSETCOLOR 11,0,15,15
    VSETCOLOR 12,10,10,10
    VSETCOLOR 13,15,12,6
    VSETCOLOR 14,15,8,8
    VSETCOLOR 15,15,6,15
    '
  RETURN
```



Ce programme vous proposera d'entrer les paramètres dans le même ordre que le précédent, puis effectuera l'affichage de la partie de l'ensemble demandé.

Bien sûr, il vous faudra être patient, surtout si vous avez demandé une précision de calcul élevée.

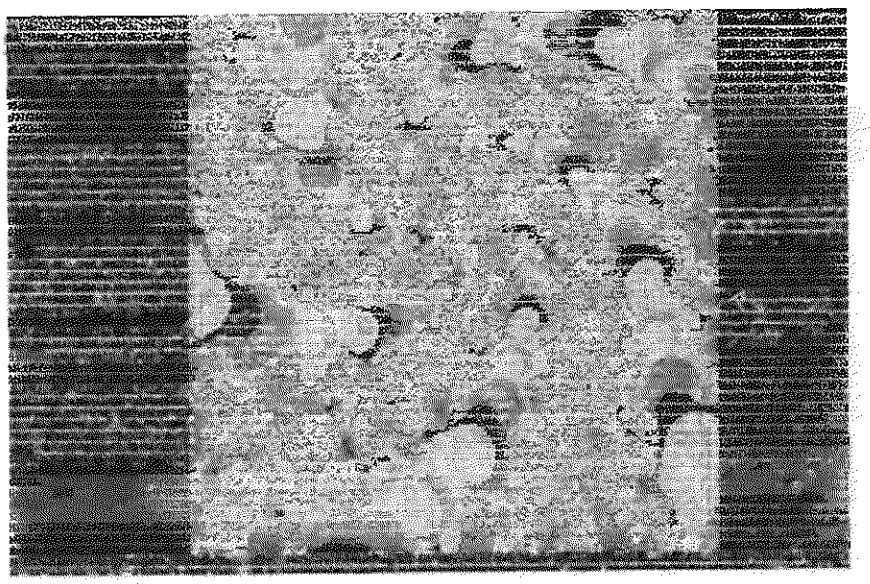


Fig. 6 : Un détail agrandi avec une précision égale à 300.

Dans toutes les images des ensembles de Julia que vous étudierez (les merveilleuxs agencements de couleurs que nous n'avons hélas pu reproduire qu'en noir et blanc), n'oubliez pas que tout point n'apparaissant pas en noir n'appartient pas à l'ensemble de Julia représenté.

LE COMPLEXE ENSEMBLE M

L'ensemble de Mandelbrot (M) fut découvert par le mathématicien français Benoît Mandelbrot, lors de ses recherches sur les formes géométriques dans un centre de recherche de la société IBM aux Etats-Unis.

Cet ensemble est l'un des ensembles les plus extraordinaires qui puissent exister actuellement, au vu de la complexité des formes qu'il engendre.

Bien que différent, puisqu'il est unique, mais à la manière des ensembles de Julia, il est possible de le fouiller par agrandissements

successifs de certaines de ses zones frontières, pour découvrir toujours de nouveaux détails.

Ces détails, en apparence semblable quelquefois à d'autres, recèlent de nouveaux motifs, lesquels apportent encore de nouvelles formes géométriques.

Mais ne tardons pas, et entrons de suite dans l'étude de la fonction le déterminant.

La construction de Mandelbrot

La méthode de construction de l'ensemble M est relativement similaire à celle des ensembles de Julia, car elle se produit aussi dans le plan complexe.

La formule mathématique y ressemble fortement aussi, car il s'agit toujours d'itérer la fonction complexe :

$$z = z^2 + Cte$$

Là où intervient la différence est la détermination de la constante Cte et du point initial de chaque itération z0.

En fait, la constante prendra pour valeur la valeur complexe du point du plan sur lequel on désire effectuer le calcul. Le point initial z0 étant toujours égal à zéro (z0 = 0 + i * 0).

On effectuera ainsi, comme pour les ensembles de Julia, une série d'itération permettant de déterminer la fuite ou non d'un point vers l'infini, ce qui permettra de colorier l'ensemble.

L'algorithme différent de celui des ensembles de Julia se présente sous la forme suivante :

```
– DEBUT
    – POUR x DE valeur initiale A valeur finale
        – POUR y DE valeur initiale A valeur finale
            – Donner la Cte la valeur x + i * y
                – TANTQUE module calculé inférieur à 4 et
                  nombre maximum d'itération non atteint
                    – Calculer z_n = z_n-1 ^2 + Cte
                        – CO
                          en commençant par z0 = 0 + i * 0
                        – FINCO
                    – Calculer module de z_n
                – FINTANTQUE
                – SI le module est inférieur à 4
                – Colorier le point correspondant en noir
```



```
                    - SINON
                        - Colorier le point correspondant en blanc
                    - FINSI
                - FINPOUR
            - FINPOUR
- FIN
```

Cet algorithme permettra de créer une version pointilliste de l'ensemble de Mandelbrot qui possède la forme d'une île qui semble composée d'une cardioïde aux côtés de laquelle seraient greffés des presqu'îles, ou semblant d'îlots de dimensions de plus en plus réduites (voir figure 7).

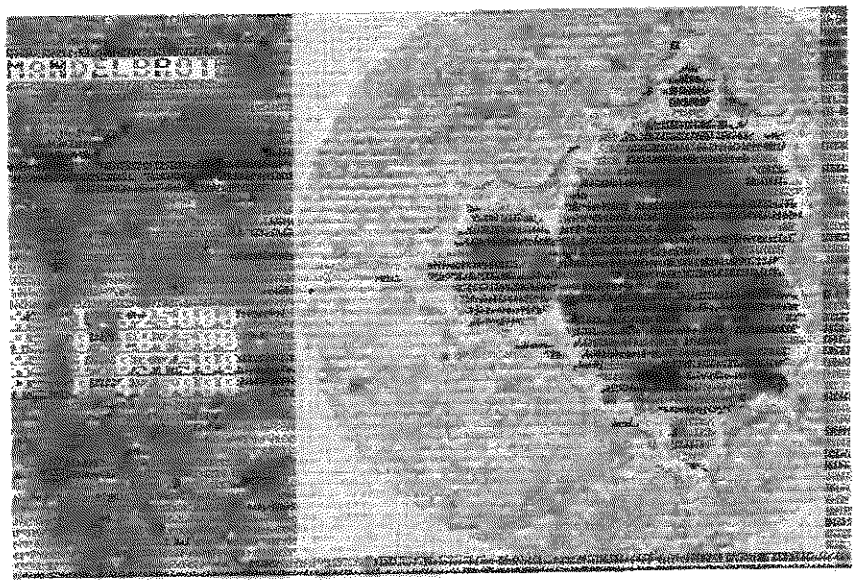


Fig. 7 : Une vue générale de l'île M.

Signalons que cet ensemble se situe approximativement dans la zone du plan complexe dont les coordonnées sont situées :

- en abscisse, entre - 2,5 et + 0,5 (partie réelle) ;
- en ordonnée, entre - 1,25 et + 1,25 (partie imaginaire).

Le programme de recherche

Encore une fois, au vu de la lenteur de calcul pour une zone écran relativement grande, nous vous proposons un programme de recherche des endroits vous intéressant plus particulièrement, et cela en couleurs.

```
REM ***************************
REM ** ETUDE DE L'ENSEMBLE **
REM **     DE MANDELBROT     **
REM **       Vers. 2.0       **
REM ***************************
REM **    version couleur    **
REM **    calcul en modulo   **
REM ***************************
'
CLS
'
PRINT "Etude de l'ensemble de Mandelbrot"
PRINT
PRINT "Version colorée sur trois fenetres"
'
'  ~~~~~~~~~~~~~~~~~~~~
'  initialisations
'  ~~~~~~~~~~~~~~~~~~~~
'
variable%=XBIOS(4) ! resolution ecran
IF variable%<>0
  ALERT 3,"Ce programme ne:fonctionne qu'en:basse resolution
",1,"RETURN",variable%
  EDIT
ENDIF
'
REM precision de calcul
'
REPEAT
  PRINT
  INPUT "Précision ou calcul(50 à 1000)";precision%
  PRINT
UNTIL precision%>49 AND precision%<1001
'
REPEAT
  PRINT
  INPUT "nombre de couleurs (2 à 15)";nombre_couleur%
  PRINT
UNTIL nombre_couleur%>1 AND nombre_couleur%<16
'
'  ~~~~~~~~~~~~~~~~~~~
```



(suite)

```
'
' acquisition de la zone de depart
'
'     ------------------------------------
REM initialisation de la palette
REM de couleur du VDI
'     ------------------------------------
'
IF nombre_couleur%>4
  modification_couleur
ELSE
  IF nombre_couleur%=2
    deux_couleur
  ELSE
    quatre_couleur
  ENDIF
ENDIF
'
'     ------------------------------------------
REM acquisition differents parametres
REM          pour les calculs
'     ------------------------------------------
'
REM acquisition des coordonnes de depart
'
INPUT "Coordonnee X du point initial";x_initial_mandelbrot
INPUT "Coordonnee Y du point initial";y_initial_mandelbrot
'
PRINT
REM acquisition des coordonnees finales
'
INPUT "Coordonnee X du point final  ";x_final_mandelbrot
INPUT "Coordonnee Y du point final  ";y_final_mandelbrot
'
CLS
'
DO
  '
  ' verification inferieur-superieur
  '
  IF x_initial_mandelbrot>x_final_mandelbrot
    SWAP x_initial_mandelbrot,x_final_mandelbrot
  ENDIF
  IF y_initial_mandelbrot>y_final_mandelbrot
    SWAP y_initial_mandelbrot,y_final_mandelbrot
  ENDIF
  '
  ' calcul de proportionnalite
  '
  ratio_x=(x_final_mandelbrot-x_initial_mandelbrot)/x_nombre
_pixel% ! nombre pixels en X
  ratio_y=(y_final_mandelbrot-y_initial_mandelbrot)/y_nombre
_pixel% ! nombre pixels en Y
  '
  ' initialisation sur ecran
  '
  LOCATE 1,19
  FOR i%=1 TO 6
```

(suite)

```
    PRINT CHR$(27);"p";
    PRINT SPACE$(40);
  NEXT i%
  LOCATE 14,21
  PRINT CHR$(27);"q";
  PRINT "L'ile MANDELBROT"
  '
  ' ------------------------------------
  ' calcul de l'ensemble de mandelbrot
  ' ------------------------------------
  '
  y_constante=y_initial_mandelbrot-ratio_y ! on demarre en d
essous
  '                                          ! avec rectificati
on dans
  '                                          ! la boucle
  '
  FOR y_pixel%=y_pixel_depart% DOWNTO y_pixel_depart%-y_nomb
re_pixel%
    y_constante=y_constante+ratio_y
    x_constante=x_initial_mandelbrot-ratio_x ! on demarre a
gauche
    '                                          ! pour rectifica
tion
    '
    FOR x_pixel%=x_pixel_depart% TO x_pixel_depart%+x_nombre
_pixel%
      x_constante=x_constante+ratio_x
      x_mandelbrot=0
      y_mandelbrot=0
      '
      nombre_iteration%=0
      '
      WHILE nombre_iteration%<precision% AND x_mandelbrot*x_
mandelbrot+y_mandelbrot*y_mandelbrot<4
        x_provisoire=x_mandelbrot*x_mandelbrot-y_mandelbrot*
y_mandelbrot+x_constante
        y_mandelbrot=2*x_mandelbrot*y_mandelbrot+y_constante
        x_mandelbrot=x_provisoire
        INC nombre_iteration%
      WEND
      IF x_mandelbrot*x_mandelbrot+y_mandelbrot*y_mandelbrot
<4
        COLOR 0
      ELSE
        COLOR nombre_iteration% MOD (nombre_couleur%)+1
      ENDIF
      PLOT x_pixel%,y_pixel%
      '
    NEXT x_pixel%
    '
  NEXT y_pixel%
  '
  REM voir si ESC frappee
  '
  sortie_boucle%=0
```



(suite)

```
    IF INKEY$=CHR$(27)
      ALERT 2,"Arret du calcul en cours?",1,"OUI:NON",variable
%
      IF variable%=1
        sortie_boucle%=1
      ELSE
        sortie_boucle%=0
      ENDIF
    ENDIF
    EXIT IF sortie_boucle%=1
    '
    REM sinon nouvelle zone de calcul
    '
    saisie_zone
    '
  LOOP
  '
  ALERT 2,"F= Fin :N= Nouveau ",0," F : N ",variable%
  '
  CLS
  IF variable%=2
    RUN
  ELSE
    EDIT
  ENDIF
  '
  END
  '
  '
  PROCEDURE saisie_zone
    '
    ' ------------------------------
    ' procedure de saisie d'une
    ' zone carree sur le dessin
    ' ------------------------------
    '
    GET 0,0,x_nombre_pixel%+1,y_nombre_pixel%+50,dessin2$
    '
    SHOWM
    DEFMOUSE 5
    COLOR 1
    '
    REPEAT
      '
      ' selection du point initial
      '
      REM mode xor
      '
      GRAPHMODE 3
      '
      MOUSE x_depart%,y_depart%,touche%
      LINE x_depart%,0,x_depart%,199
      LINE 0,y_depart%,319,y_depart%
      REPEAT
        SHOWM
        '
        x_ancien%=x_depart%
```

(suite)

```
        y_ancien%=y_depart%
        '
        MOUSE x_depart%,y_depart%,touche%
        '
        REM reperage coordonnee depart
        '
        LINE x_ancien%,0,x_ancien%,199
        LINE 0,y_ancien%,319,y_ancien%
        LINE x_depart%,0,x_depart%,199
        LINE 0,y_depart%,319,y_depart%
        '
        x_initial_provisoire=x_initial_mandelbrot+ratio_x*(x_d
epart%-x_coin_bas_gauche%)
        y_initial_provisoire=y_initial_mandelbrot-ratio_y*(y_d
epart%-y_coin_bas_gauche%)
        '
        REM affichage coordonnee depart
        '
        LOCATE 1,14
        PRINT "X=";USING "##.######",x_initial_provisoire
        LOCATE 1,15
        PRINT "Y=";USING "##.######",y_initial_provisoire
        '
    UNTIL touche%=1

    LINE x_depart%,0,x_depart%,199
    LINE 0,y_depart%,319,y_depart%
    '
    x_ancien%=x_depart%
    y_ancien%=y_depart%
    '
    GRAPHMODE 3
    BOX x_depart%,y_depart%,x_ancien%,y_ancien%
    '
    LOCATE 1,14
    PRINT "X=";USING "##.######",x_initial_provisoire
    LOCATE 1,15
    PRINT "Y=";USING "##.######",y_initial_provisoire
    '
    ' selection du point final
    '
    DELAY 1
    '
    REPEAT
        '
        MOUSE x_fin%,y_fin%,touche%
        '
        y_fin%=y_depart%-(x_fin%-x_depart%)
        x_final_provisoire=x_initial_mandelbrot+ratio_x*(x_fin
%-x_coin_bas_gauche%)
        y_final_provisoire=y_initial_mandelbrot-ratio_y*(y_fin
%-y_coin_bas_gauche%)
        '
        REM affichage coordonnee finale
        '
        LOCATE 1,16
        PRINT "X=";USING "##.######",x_final_provisoire
```



(suite)

```
        LOCATE 1,17
        PRINT "Y=";USING "##.######",y_final_provisoire
        '
        REM dessin zone
        '
        BOX x_depart%,y_depart%,x_ancien%,y_ancien%
        BOX x_depart%,y_depart%,x_fin%,y_fin%
        '
        x_ancien%=x_fin%
        y_ancien%=y_fin%
        '
      UNTIL touche%=1
      '
      IF x_depart%=x_fin% OR y_depart%=y_fin% THEN
        BOX x_depart%,y_depart%,x_fin%,y_fin%
      ENDIF
      '
      DELAY 1
      '
    UNTIL x_depart%<>x_fin% AND y_depart%<>y_fin%
    '
    HIDEM
    '
    REM remettre dans l'ordre les coordonnees
    '
    IF x_fin%<x_depart% THEN
      SWAP x_depart%,x_fin%
    ENDIF
    IF y_fin%<y_depart% THEN
      SWAP y_depart%,y_fin%
    ENDIF
    '
    REM nouvelles coordonnees pour le calcul
    '
    x_initial_mandelbrot=x_initial_provisoire
    y_initial_mandelbrot=y_initial_provisoire
    x_final_mandelbrot=x_final_provisoire
    y_final_mandelbrot=y_final_provisoire
    '
    REM mode normal
    '
    GRAPHMODE 1
    '
    REM recopie du dessin
    '
    GET x_nombre_pixel%+1,0,x_nombre_pixel%+x_nombre_pixel%+1,
y_nombre_pixel%+50,dessin$
    PUT x_nombre_pixel%+x_nombre_pixel%+6,0,dessin$
    GET 0,0,x_nombre_pixel%+1,y_nombre_pixel%+50,dessin2$
    PUT x_nombre_pixel%+3,0,dessin2$
    '
RETURN
'
PROCEDURE modification_couleur
    '
    ' ------------------------------------------
    ' l'utilisation de VSETCOLOR permet
```

(suite)

```
    ' d'acceder directement aux
    ' registres de couleur du VDI et
    ' ainsi modifier directement sur
    ' les numeros vrais de couleur
    ' ____________________________________
    '
   VSETCOLOR 0,0,0,0
   VSETCOLOR 1,15,15,0
   VSETCOLOR 2,15,10,0
   VSETCOLOR 3,15,8,0
   VSETCOLOR 4,12,6,0
   VSETCOLOR 5,15,0,0
   VSETCOLOR 6,15,0,8
   VSETCOLOR 7,12,0,10
   VSETCOLOR 8,10,0,12
   VSETCOLOR 9,0,0,15
   VSETCOLOR 10,0,8,14
   VSETCOLOR 11,0,10,8
   VSETCOLOR 12,0,10,2
   VSETCOLOR 13,8,12,2
   VSETCOLOR 14,8,12,0
   VSETCOLOR 15,10,14,0
   '
RETURN
'
PROCEDURE quatre_couleur
   '
   VSETCOLOR 0,0,0,0
   VSETCOLOR 1,15,15,15
   VSETCOLOR 2,0,0,15
   VSETCOLOR 3,0,15,0
   VSETCOLOR 4,15,0,0
   VSETCOLOR 5,15,0,15
   '
RETURN
'
PROCEDURE deux_couleur
   '
   VSETCOLOR 0,0,0,0
   VSETCOLOR 1,15,15,15
   VSETCOLOR 2,15,15,15
   '
RETURN
'
```



Ce programme fonctionne de façon similaire à celui de recherche sur les ensembles de Julia, aussi, nous vous recommenderons de vous y reporter pour l'utilisation.

En version monochrome, vous pourrez aussi obtenir des versions pointillistes telle que celle représentée en figure 8.

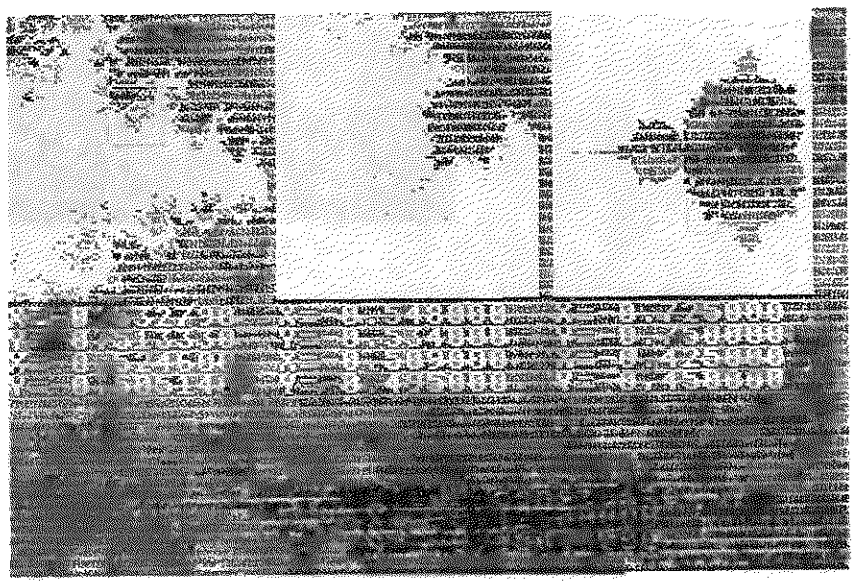


Fig. 8 : Précision égale à 100.

Dans cette représentation, ne vous y fiez pas, les îlots noirs que l'on peut observer dans le tracé du milieu sont en fait des presqu'îles, et il est possible de découvrir un chemin les reliant à l'ensemble, en agrandissant cette partie d'écran.

L'ensemble de Mandelbrot est en effet connexe, c'est-à-dire que tous les points lui appartenant sont reliés ensemble par d'autres points de l'ensemble, cela ayant été démontré par les deux mathématiciens A. Douady et J. Hubbard.

Une étude intéressante se situe à la pointe de M, qui fait découvrir des mini-Mandelbrot, pourtant tous différents les uns des autres reliés entre eux et à la partie mère de l'île, et formant en certains endroits un réseau de type courbe de Peano (figure 9).



Fig. 9 : La pointe de Mandelbrot.

Vers une progressivité des couleurs

Le programme proposé ci-dessus visualise les points en fuite vers l'infini par des zones colorées dépendant de leur latence. Le procédé utilisé étant le reste de la division entière par l'instruction MOD, on retrouve les mêmes couleurs périodiquement.

Pour améliorer l'esthétique de visualisation de certaines zones, nous vous proposons de modifier légèrement le programme afin de rendre progressive l'apparition des couleurs au fur et à mesure que l'on s'approche des frontières de l'ensemble.

La partie de programme à modifier est proposée ci-dessous :

```
REM ***************************
REM **     MODIFICATION      **
REM ** DE LA PROGRESSIVITE **
REM ***************************
'
'
' partie de programme a modifier
' pour un affichage progressif
' des couleurs des differentes
' zones colorees correspondantes
' a des zones de fuites de meme
' ordre de grandeur
'
IF x_mandelbrot*x_mandelbrot+y_mandelbrot*y_mandelbrot<4
  couleur%=0
ELSE
  couleur%=INT(nombre_iteration%*nombre_couleur%/precision%)
+1
ENDIF
PSET x_pixel%,y_pixel%,couleur%
'
```



Dans ce programme, la couleur est calculée en divisant le nombre d'itération du point fuyant, par la précision demandée (nombre maximal d'itération) et multiplié par le nombre de couleur.

Version dégradée de gris

Une autre version intéressante du programme est obtenue en modifiant les procédures de choix des couleurs pour obtenir des effets de dégradé de gris, grâce à la modification de la partie de programme donnée par le listing suivant.

```
REM ****************************
REM **    MODIFICATION DES    **
REM **    AFFECTATIONS DE     **
REM **        COULEURS        **
REM ****************************

PROCEDURE modification_couleur
   '
   ' ------------------------------------------------
   ' l'utilisation de VSETCOLOR permet
   ' d'acceder directement aux
   ' registres de couleur du VDI et
   ' ainsi modifier directement sur
   ' les numeros vrais de couleur
   ' ------------------------------------------------
   '
   '
   ' couleurs donnees pour la
   ' version de programme travaillant
   ' en degrades de gris
   '
   '
  VSETCOLOR 0,0,0,0
  VSETCOLOR 1,15,15,15
  VSETCOLOR 2,2,2,2
  VSETCOLOR 3,3,3,3
  VSETCOLOR 4,4,4,4
  VSETCOLOR 5,5,5,5
  VSETCOLOR 6,6,6,6
  VSETCOLOR 7,7,7,7
  VSETCOLOR 8,8,8,8
  VSETCOLOR 9,9,9,9
  VSETCOLOR 10,10,10,10
  VSETCOLOR 11,11,11,11
  VSETCOLOR 12,12,12,12
  VSETCOLOR 13,13,13,13
  VSETCOLOR 14,14,14,14
  VSETCOLOR 15,15,15,15
   '
RETURN
'
PROCEDURE quatre_couleur

   VSETCOLOR 0,0,0,0
   VSETCOLOR 1,15,15,15
   VSETCOLOR 2,11,11,11
   VSETCOLOR 3,8,8,8
   VSETCOLOR 4,6,6,6
   VSETCOLOR 5,3,3,3
   '
RETURN
'
PROCEDURE deux_couleur
   '
   VSETCOLOR 0,0,0,0
   VSETCOLOR 1,15,15,15
   VSETCOLOR 2,15,15,15
   '
RETURN
```



L'observation des détails et sauvegarde

Après une période de recherche plus ou moins longue, vous pourrez vous permettre grâce au programme qui va suivre de visualiser plus en détail une zone découverte.

Ce programme vous permettra entre autre de continuer à détailler l'ensemble en choisissant une nouvelle zone dans l'écran dessiné sur une fenêtre d'environ 200 pixels de côté, à l'aide de la souris, et même de sauvegarder l'image pour la récupérer dans un programme de votre composition, ou pour continuer l'étude.

```
REM ****************************
REM ** ETUDE DE L'ENSEMBLE **
REM **     DE MANDELBROT      **
REM **        Vers. 3.0       **
REM ****************************
REM **    version couleur     **
REM **    calcul en modulo    **
REM ****************************
'
CLS
'
PRINT "Etude de l'ensemble de Mendelbrot"
PRINT
PRINT "Version colorée"
'
initialisation              ! precision et couleurs
'
fenetre                     ! dimension affichage
'
init_couleur                ! choix des couleurs
'
chargement                  ! voir si ancien calcul
'
DO
   '
   ordonner
   '
   calcul_proportion
   '
   initialiser_ecran
   '
   calculer_mandelbrot
   '
   sauvegarde
   '
LOOP
'
'
PROCEDURE saisie_zone
   '
   ' ----------------------------
   ' procedure de saisie d'une
   ' zone carree sur le dessin
   ' ----------------------------
   '
   '
   SHOWM
   DEFMOUSE 5
   COLOR 2
   '
   REPEAT
      '
      ' selection du point initial
      '
      REM mode xor
      '
      GRAPHMODE 3 ;
```



(suite)

```
      MOUSE x_depart%,y_depart%,touche%
      LINE x_depart%,0,x_depart%,199
      LINE 0,y_depart%,319,y_depart%
      REPEAT
        SHOWM
        '
        x_ancien%=x_depart%
        y_ancien%=y_depart%
        '
        MOUSE x_depart%,y_depart%,touche%
        '
        REM reperage coordonnee depart
        '
        LINE x_ancien%,0,x_ancien%,199
        LINE 0,y_ancien%,319,y_ancien%
        LINE x_depart%,0,x_depart%,199
        LINE 0,y_depart%,319,y_depart%
        '
        x_initial_provisoire=x_initial_mandelbrot+ratio_x*(x_d
epart%-x_coin_bas_gauche%)
        y_initial_provisoire=y_initial_mandelbrot-ratio_y*(y_d
epart%-y_coin_bas_gauche%)
        '
        REM affichage coordonnee depart
        '
        LOCATE 1,14
        PRINT "X=";USING "##.######",x_initial_provisoire
        LOCATE 1,15
        PRINT "Y=";USING "##.######",y_initial_provisoire

      UNTIL touche%=1
      '
      LINE x_depart%,0,x_depart%,199
      LINE 0,y_depart%,319,y_depart%
      '
      x_ancien%=x_depart%
      y_ancien%=y_depart%
      '
      GRAPHMODE 3
      BOX x_depart%,y_depart%,x_ancien%,y_ancien%
      '
      LOCATE 1,14
      PRINT "X=";USING "##.######",x_initial_provisoire
      LOCATE 1,15
      PRINT "Y=";USING "##.######",y_initial_provisoire
      '
      ' selection du point final
      '
      DELAY 1
      '
      REPEAT
        '
        MOUSE x_fin%,y_fin%,touche%
        '
        y_fin%=y_depart%-(x_fin%-x_depart%)
        x_final_provisoire=x_initial_mandelbrot+ratio_x*(x_fin
```

(suite)

```
%-x_coin_bas_gauche%)
        y_final_provisoire=y_initial_mandelbrot-ratio_y*(y_fin
%-y_coin_bas_gauche%)
        '
        REM affichage coordonnee finale
        '
        LOCATE 1,16
        PRINT "X=";USING "##.######",x_final_provisoire
        LOCATE 1,17
        PRINT "Y=";USING "##.######",y_final_provisoire
        '
        REM dessin zone
        '
        BOX x_depart%,y_depart%,x_ancien%,y_ancien%
        BOX x_depart%,y_depart%,x_fin%,y_fin%
        '
        x_ancien%=x_fin%
        y_ancien%=y_fin%
        '
      UNTIL touche%=1
      '
      IF x_depart%=x_fin% OR y_depart%=y_fin% THEN
        BOX x_depart%,y_depart%,x_fin%,y_fin%
      ENDIF
      '
      DELAY 1
      '
    UNTIL x_depart%<>x_fin% AND y_depart%<>y_fin%
    '
    HIDEM
    '
    REM remettre dans l'ordre les coordonnees
    '
    IF x_fin%<x_depart% THEN
      SWAP x_depart%,x_fin%
    ENDIF
    IF y_fin%<y_depart% THEN
      SWAP y_depart%,y_fin%
    ENDIF
    '
    REM nouvelles coordonnees pour le calcul
    '
    x_initial_mandelbrot=x_initial_provisoire
    y_initial_mandelbrot=y_initial_provisoire
    x_final_mandelbrot=x_final_provisoire
    y_final_mandelbrot=y_final_provisoire
    '
    REM mode normal
    '
    GRAPHMODE 1
    '
    REM recopie du dessin
    '
    '
  RETURN
  '
  PROCEDURE modification_couleur
```

(suite)

```
'
' ------------------------------------------------
' l'utilisation de VSETCOLOR permet
' d'acceder directement aux
' registres de couleur du VDI et
' ainsi modifier directement sur
' les numeros vrais de couleur
' ------------------------------------------------
'
VSETCOLOR 0,0,0,0
VSETCOLOR 1,15,15,15
VSETCOLOR 2,15,10,0
VSETCOLOR 3,15,8,0
VSETCOLOR 4,12,6,0
VSETCOLOR 5,15,0,0
VSETCOLOR 6,15,0,8
VSETCOLOR 7,12,0,10
VSETCOLOR 8,10,0,12
VSETCOLOR 9,0,0,15
VSETCOLOR 10,0,8,14
VSETCOLOR 11,0,10,8
VSETCOLOR 12,0,10,2
VSETCOLOR 13,8,12,2
VSETCOLOR 14,8,12,0
VSETCOLOR 15,10,14,0
'
RETURN
'
PROCEDURE quatre_couleur
'
VSETCOLOR 0,0,0,0
VSETCOLOR 1,15,15,15
VSETCOLOR 2,0,0,15
VSETCOLOR 3,0,15,0
VSETCOLOR 4,15,0,0
VSETCOLOR 5,15,0,15
'
RETURN
'
PROCEDURE deux_couleur
'
VSETCOLOR 0,0,0,0
VSETCOLOR 1,15,15,15
VSETCOLOR 2,15,15,15
'
RETURN
'
PROCEDURE frappe_esc
'
ALERT 2,"Arret du:calcul:en cours?",1,"OUI:NON",variable%
IF variable%=1
  ALERT 2,"F= Fin !N= Nouveau ",0," F : N ",variable%
  '
  CLS
  IF variable%=2
    RUN
  ELSE
```

(suite)

```
      EDIT
    ENDIF
  ENDIF
  '
RETURN
'
PROCEDURE calcul_proportion
  '
  ' calcul de proportionnalite
  '
  ratio_x=(x_final_mandelbrot-x_initial_mandelbrot)/x_nombre
_pixel% ! nombre pixels en X
  ratio_y=(y_final_mandelbrot-y_initial_mandelbrot)/y_nombre
_pixel% ! nombre pixels en Y
  '
RETURN
'
PROCEDURE initialisation
  '
  ' ________________
  ' initialisations
  ' ________________
  '
  variable%=XBIOS(4) ! resolution ecran
  IF variable%<>0
    ALERT 3,"Ce programme ne!fonctionne qu'en!basse resoluti
on",1,"RETURN",variable%
    EDIT
  ENDIF
  '
  REM precision de calcul
  '
  REPEAT
    PRINT
    INPUT "Précision du calcul(50 à 1000)";precision%
    PRINT
  UNTIL precision%>49 AND precision%<1001
  '
  REPEAT
    PRINT
    INPUT "nombre de couleurs (2 à 15)";nombre_couleur%
    PRINT
  UNTIL nombre_couleur%>1 AND nombre_couleur%<16
  '
RETURN
'
PROCEDURE fenetre
  '
  '  ____________________________________
  REM dimensions fenetre d'affichage
  '  ____________________________________
  '
  x_nombre_pixel%=190      ! un carre de
  y_nombre_pixel%=190      ! 200 sur 190

  REM position fenetre d'affichage
  '
```



(suite)

```
    x_coin_bas_gauche%=110      ! fenetre tout
    y_coin_bas_gauche%=190      ! a gauche
    '
    REM point de depart pour le calcul
    '
    x_pixel_depart%=x_coin_bas_gauche%-1
    y_pixel_depart%=y_coin_bas_gauche%+1
    '
RETURN
'
PROCEDURE init_couleur
    '
    '   ------------------------------------
    REM initialisation de la palette
    REM de couleur du VDI
    '   ------------------------------------
    '
    IF nombre_couleur%>4
      modification_couleur
    ELSE
      IF nombre_couleur%=2
        deux_couleur
      ELSE
        quatre_couleur
      ENDIF
    ENDIF
    '
RETURN
'
PROCEDURE chargement
    '
    ALERT 2,"Charger|Mandelbrot",2,"OUI|NON",variable%
    '
    IF variable%=1
      SGET ecran$
      OPENW #1,30,30,160,60,&H0
      '
      REPEAT
        CLS
        '
        PRINT "Nom de l'image: "
        FILESELECT "a:\*.ecr","",ecr",nomimage$
        PRINT nomimage$
        '
      UNTIL nomimage$<>""
      '
      OPEN "i",#4,nomimage$
      ecran$=INPUT$(32000,#4)
      CLOSE #4
      '
      nom$=LEFT$(nomimage$,LEN(nomimage$)-4)
      nomdata$=nom$+".dat"
      OPEN "i",#4,nomdata$
      INPUT #4,x_initial_mandelbrot
      INPUT #4,y_initial_mandelbrot
      INPUT #4,x_final_mandelbrot
      INPUT #4,y_final_mandelbrot
```

(suite)

```
      CLOSE #4
      '
      CLOSEW #1
      CLOSEW #0
      '
      SPUT ecran$
      calcul_proportion
      '
      saisie_zone
    ELSE
      '
      coordonnees
      '
    ENDIF
  '
RETURN
'
PROCEDURE coordonnees
  '
  '   ____________________________________________
  REM acquisition differents parametres
  REM          pour les calculs
  '   ____________________________________________
  '
  REM acquisition des coordonnes de depart
  '
  INPUT "Coordonnee X du point initial";x_initial_mandelbrot
  INPUT "Coordonnee Y du point initial";y_initial_mandelbrot
  '
  PRINT
  REM acquisition des coordonnees finales
  '
  INPUT "Coordonnee X du point final   ";x_final_mandelbrot
  INPUT "Coordonnee Y du point final   ";y_final_mandelbrot
  '
  CLS
  '
RETURN
'
PROCEDURE ordonner
  '
  ' verification inferieur-superieur
  '
  IF x_initial_mandelbrot>x_final_mandelbrot
    SWAP x_initial_mandelbrot,x_final_mandelbrot
  ENDIF
  IF y_initial_mandelbrot>y_final_mandelbrot
    SWAP y_initial_mandelbrot,y_final_mandelbrot
  ENDIF
  '
RETURN
'
PROCEDURE initialiser_ecran
  '
  ' initialisation sur ecran
  '
  LOCATE 1,3
```



(suite)

```
    PRINT CHR$(27);"p";
    PRINT "MANDELBROT"
    PRINT CHR$(27);"q"
    '
RETURN
'
PROCEDURE calculer_mandelbrot
    '
    ' ------------------------------------------
    ' calcul de l'ensemble de mandelbrot
    ' ------------------------------------------
    '
   y_constante=y_initial_mandelbrot-ratio_y ! on demarre en d
essous
    '                                       ! avec rectificat
ion dans
    '                                       ! la boucle
    '
   FOR y_pixel%=y_pixel_depart% DOWNTO SUB(y_pixel_depart%,y_
nombre_pixel%)
     ADD y_constante,ratio_y
     x_constante=x_initial_mandelbrot-ratio_x ! on demarre a
gauche
     '                                        ! pour rectific
ation
     '
     FOR x_pixel%=x_pixel_depart% TO ADD(x_pixel_depart%,x_no
mbre_pixel%)
       ADD x_constante,ratio_x
       x_mandelbrot=0
       y_mandelbrot=0
       '
       nombre_iteration%=0
       module=0
       '
       WHILE nombre_iteration%<precision% AND x_mandelbrot*x_
mandelbrot+y_mandelbrot*y_mandelbrot<4
         x_provisoire=x_mandelbrot*x_mandelbrot
         y_provisoire=y_mandelbrot*y_mandelbrot
         y_mandelbrot=2*x_mandelbrot*y_mandelbrot+y_constante
         x_mandelbrot=x_provisoire-y_provisoire+x_constante
         INC nombre_iteration%
       WEND
       IF x_mandelbrot*x_mandelbrot+y_mandelbrot*y_mandelbrot
<4
         couleur%=0
       ELSE
         couleur%=MOD(nombre_iteration%,nombre_couleur%)
       ENDIF
       PSET x_pixel%,y_pixel%,couleur%
       '
       REM voir si ESC est frappe
       '
       IF INKEY$=CHR$(27)
         frappe_esc
       ENDIF
       '
```

(suite)

```
      NEXT x_pixel%
    NEXT y_pixel%
  RETURN
  PROCEDURE sauvegarde
    REM voir sauvegarde
    REM sinon nouvelle zone de calcul
    ALERT 2,"SAUVEGARDE",2,"OUI:NON",variable%
    IF variable%=1
      SGET ecran$
      OPENW #1,30,30,160,60,&H0
      REPEAT
        REPEAT
          CLS
          PRINT "Nom de l'image: "
          FILESELECT "a:\*.ecr","*.ecr",nomimage$
          nomimage$=LEFT$(nomimage$,11)+".ECR"
          PRINT nomimage$
        UNTIL nomimage$<>""
        variable%=0
        IF EXIST(nomimage$)
          ALERT 1,"Fichier existant;Effacement ?",0,"OUI:NON",
variable%
          IF variable%=1
            variable%=0
          ENDIF
        ENDIF
        IF variable%=0
          OPEN "o",#4,nomimage$
          PRINT #4,ecran$
          CLOSE #4
          nom$=LEFT$(nomimage$,LEN(nomimage$)-4)
          nomdata$=nom$+"_dat"
          OPEN "o",#4,nomdata$
          PRINT #4,x_initial_mandelbrot
          PRINT #4,y_initial_mandelbrot
          PRINT #4,x_final_mandelbrot
          PRINT #4,y_final_mandelbrot
          CLOSE #4
        ENDIF
        CLOSEW #1
        CLOSEW #0
        SPUT ecran$
      UNTIL variable%=0
        ENDIF
        saisie_zone
      RETURN
```

Cette version du programme permet d'interrompre à tout moment le calcul par l'appui sur la touche <ESC>, et propose en fin de tracé, la sauvegarde de la nouvelle forme (voir figure 10).

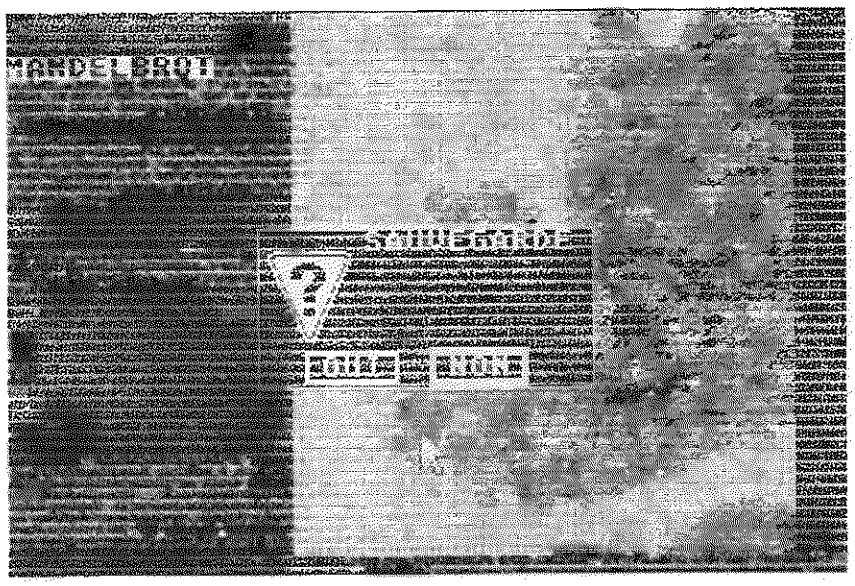


Fig. 10

Il est même possible, lors du lancement de reprendre une image précédemment créée. Une petite précision, toutes les images et leurs données sont placées dans le répertoire principal.



5/5.6

Le mouvement brownien sur Atari

De la découverte ...

C'est au début du XIXe siècle que le botaniste écossais, Robert Brown, fit une première constatation sur des grains de pollen en suspension dans l'eau.

Ceux-ci étaient alors animés de mouvement incessants, microscopiquement parlant entendons-nous. Il en conclut alors que la matière vivante, après sa mort, possédait encore une certaine vitalité qui permettait aux molécules la composant de s'animer. Ceci n'était qu'un phénomène biologique qui paraissait normal.

Quelques temps plus tard, il se mit en possession d'une gouttelette d'eau, enfermée dans un éclat de quartz. Cette gouttelette, restée prisonnière depuis des millions d'années, sans pollution possible par des organismes extérieurs, fut examinée au microscope.

Il y découvrit encore un grand nombre de particules en suspension, toutes animées de mouvements désordonnés et incessants.

Sa conclusion, dès lors, et sans les connaissances actuelles, s'arrêta à l'idée que ces mouvements étaient dus à un phénomène plus physique que biologique.

Nous connaissons actuellement l'explication du phénomène, qui est somme toute assez simple : les particules en suspension dans le liquide (quelle que soit leur nature) sont soumises au «bombardement» incessant des molécules constituant ce liquide. Précisons de suite qu'il en est de même dans un gaz, et fort probablement dans un solide, à une échelle de déplacement moindre.

Le mouvement d'une particule, particulièrement aléatoire, comme par exemple dessiné en figure 1, pourrait être celui d'un grain de pollen en suspension dans l'eau.

En admettant que l'on puisse réaliser l'observation sur une plus petite distance (voir la loupe repésentée), on observerait un mouvement tout aussi désordonné entre deux points qui auparavant semblaient reliés par une ligne continue.

C'est le physicien Jean Perrin qui, en 1912, releva ce type de mouvement.

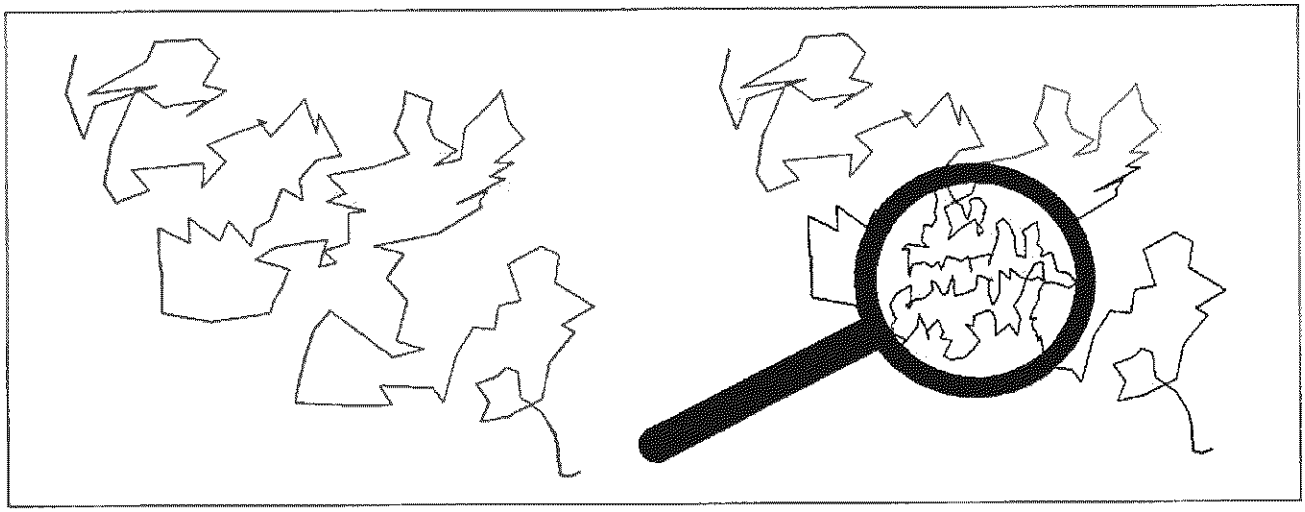


Fig. 1 : Mouvement brownien.

La décomposition quasi-infinie de ces mouvements en d'autres mouvements aussi complexes fait penser au phénomène des fractals, avancé par Benoît Mandelbrot.

... À LA REPRESENTATION SUR ATARI ST

Nous n'irons pas plus loin pour l'instant dans notre approche du mouvement brownien, car celui-ci se réfère, lors de son étude théorique, aux phénomènes probabilistes, qui rebutent déjà grand nombre de mathématiciens en herbe.

Les applications sont aussi nombreuses, notamment dans l'étude des phénomènes de diffusion de particules dans un fluide, de convection, d'agrégation (notons que les notions de volume et températures revêtent d'une grande importance pour l'étude de ces phénomènes).

Arrêtons-nous simplement pour l'intant à la représentation sur l'écran de votre Atari ST de mouvement de type «presque brownien».

L'aléatoire sur ST

Vous avez pu remarquer que nous avons placé entre guillemets l'expression «presque brownien».

En effet, le mouvement brownien appelle la notion de hasard.

Les effets de ce hasard sur Atari ne sont que simulés, et c'est bien normal, par l'utilisation de l'instruction RANDOM. Or cette instruction ne permet que de générer des séquences de nombres pseudo-aléatoires, c'est-à-dire fournis par un algorithme immuable. Dans bien des cas, celui-ci est largement suffisant, car difficilement, pour ne pas dire jamais, l'utilisateur pourra prédire le nombre demandé.



Cette restriction est incompatible avec la théorie et ne permet pas de simuler un mouvement réellement brownien, mais nous devrons nous en contenter. Pour s'approcher du véritable aléatoire, il faudrait, par exemple, demander à l'utilisateur avant chaque tirage de frapper une touche. En attendant la frappe sur la touche, il serait même intéressant d'obliger le générateur pseudo-alétoire à s'exécuter, comme dans le programme suivant :

```
PROCEDURE Aleatoire(nombre_aleatoire,limite)
        LOCAL touche_frappee$,ecran$
        SGET ecran$
        OPENW #1,30,20,235,100,&H0
        TITLEW #1,"GENERATION ALEATOIRE"
        PRINT "Frappez sur une touche"
        PRINT "pour obtenir un nombre aléatoire"
        touche_frappee$ = ""
        REPEAT
                nombre_aleatoire = RANDOM(limite)
                touche_frappee$ = INKEY$
        UNTIL touche_frappee$ <> ""
        CLOSEW #1
        CLOSEW #0
        SPUT ecran$
RETURN
```

La génération devient cette fois-ci «plus aléatoire», car il est impossible de prévoir à quel moment exact l'utilisateur appuiera sur la touche qui permettra de déterminer le nombre aléatoire.

Le programme de simulation de mouvement brownien

- Présentation du programme

Pour simuler ici le mouvement brownien, nous nous contenterons de générer des nombres pseudo-aléatoires, mais préférerons soigner la présentation et les possibilités de visualisation de ce mouvement.

L'utilisation s'effectuera dans les modes basse et moyenne résolution.

Nous avons prévu une présentation sous forme de menu. La barre supérieure présentant les choix suivants :

— BUREAU

— INIT

— VITESSE

— DISTANCE

Le BUREAU est une sélection classique.

INIT permet d'initialiser la page écran, c'est-à-dire la vider, par le choix **initialisation**, sinon de retourner à l'éditeur en sélectionnant **quitter** (voir figure 2).

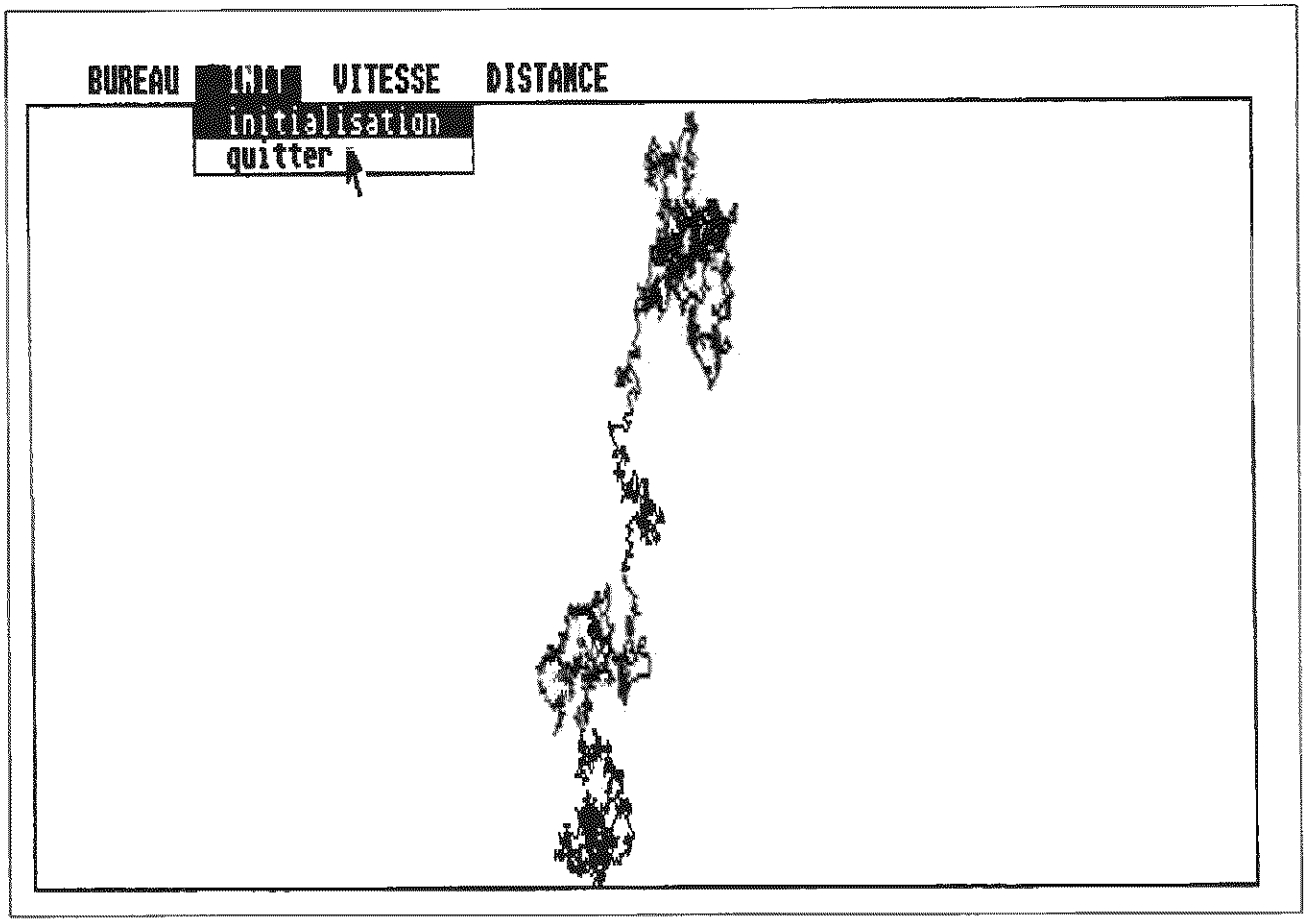


Fig. 2

VITESSE permet de choisir entre trois vitesses d'affichage du mouvement : **petite**, **moyenne**, ou **grande**, ce afin de permettre plus facilement la visualisation (en petite vitesse) ou de visualiser plus rapidement une grande partie du mouvement (voir figure 3).

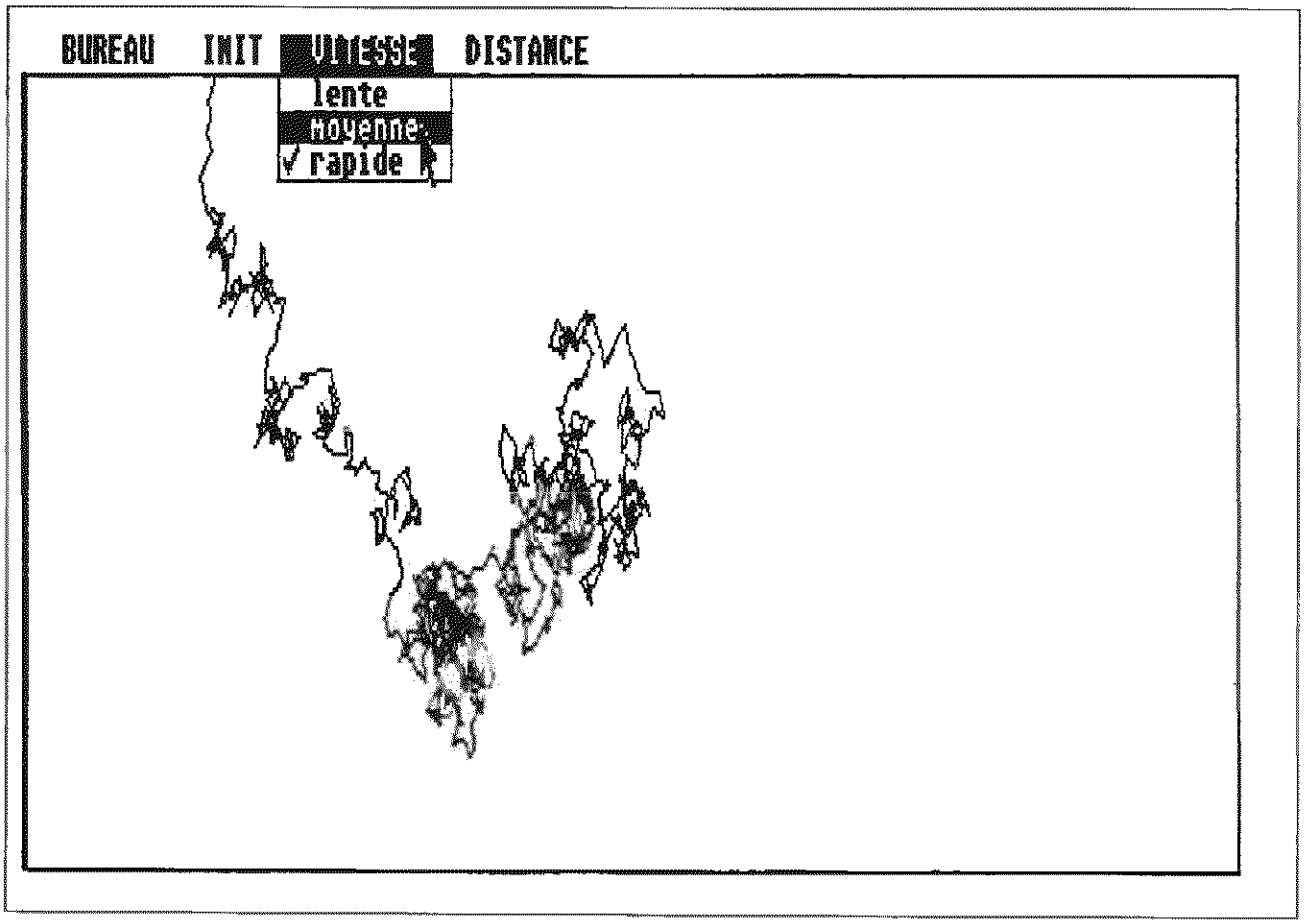


Fig. 3

DISTANCE permet, si l'on peut dire ainsi, de régler la loupe. Selon les choix **petite**, **moyenne** ou **grande**, la particule en mouvement sera visualisée de plus ou moins près, ou encore à des instants plus ou moins rapprochés (voir figure 4).

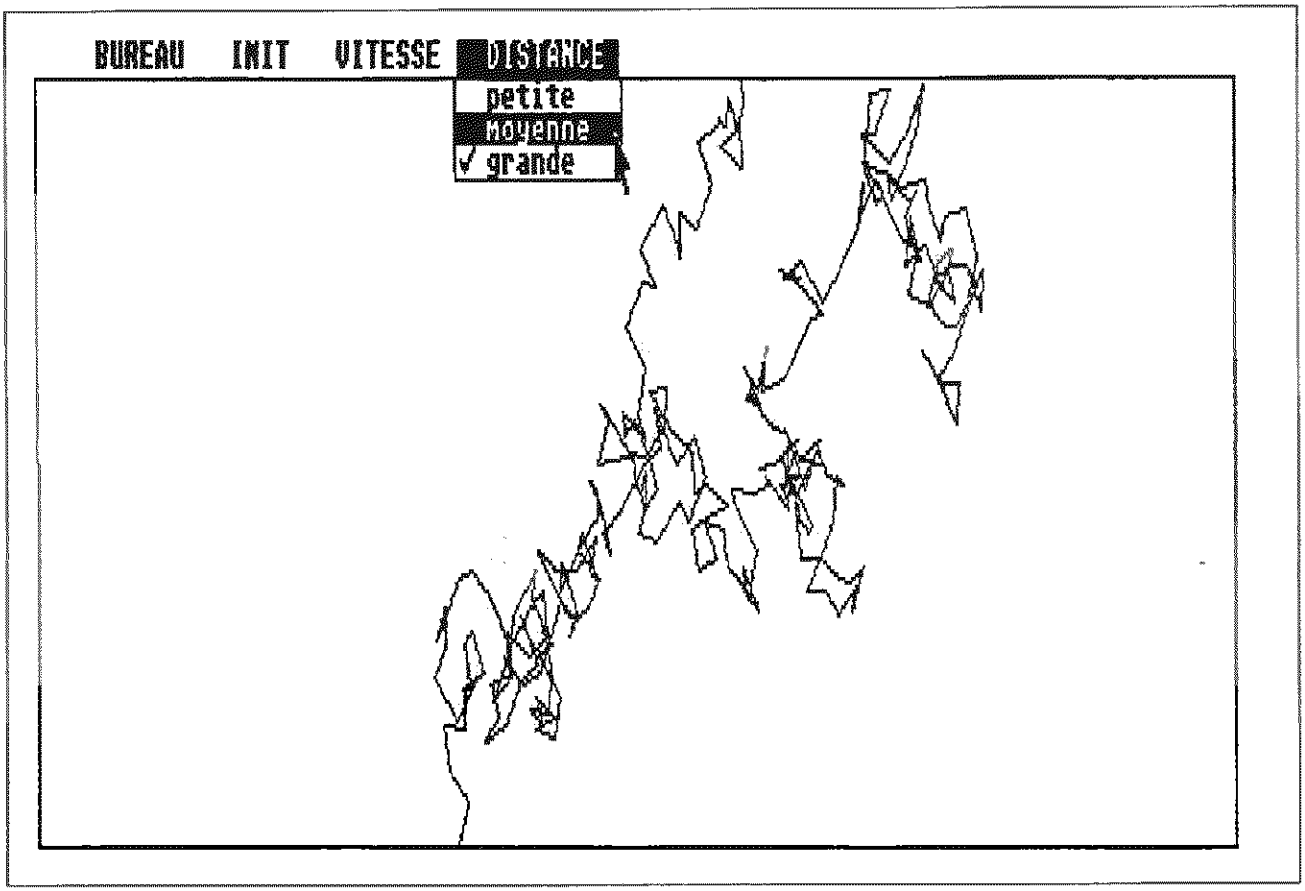


Fig. 4

● Le programme

L'algorithme général du programme étant très simple, nous n'avons pas jugé utile de l'étudier ici, aussi, nous vous proposons de suite le programme en GFA Basic 3.5E, qui sera suivi de quelques explications.

```
REM ***********************************
REM *** SIMULATION DE DEPLACEMENT ***
REM ***  D'UNE PARTICULE ANIMEE   ***
REM ***  D'UN MOUVEMENT BROWNIEN  ***
REM ***********************************
REM
```



(suite)

```
'
' acces a la resolution ecran
'
resolution%=XBIOS(4)
'
' traitement de la resolution
'
IF resolution%=0
  taille_maxi_x%=319
  taille_maxi_y%=199
ELSE
  IF resolution%=1
    taille_maxi_x%=639
    taille_maxi_y%=199
  ELSE
    ALERT 3,"Ce programme est|prevu pour les resolutions|moyennes ou
basses",1,"RETURN",variable%
    EDIT
  ENDIF
ENDIF
'
' dimensionnement et preparation
'     d'un tableau de menu
'
DIM tableau_brown$(30)
'
comptage%=-1
'
RESTORE tableau_brown
'
REPEAT
  INC comptage%
  READ tableau_brown$(comptage%)
UNTIL tableau_brown$(comptage%)="FIN"
'
tableau_brown:
'
DATA  BUREAU,  Informations
DATA ----------------------
DATA "1","2","3","4","5","6"
DATA ""
DATA  INIT,  initialisation  ,  quitter,""
DATA  VITESSE,  lente,  moyenne  ,  rapide,""
DATA  DISTANCE,  petite,  moyenne  ,  grande,""
DATA ""
DATA FIN
'
' affichage du menu
```

(suite)

```
'
OPENW 0          ! reserve la premiere ligne
MENU tableau_brown$()
'
' mise en veille des fonctions
'
FOR i%=13 TO comptage%
  MENU i%,1
NEXT i%
'
' affichage du menu
'
OPENW 0
MENU tableau_brown$()
'
' prepare la surveillance
'
ON MENU GOSUB traitement1
'
' mise en veille des fonctions
'
FOR i%=3 TO comptage%
  MENU i%,2
NEXT i%
'
PRINT "SIMULATION DE MOUVEMENT BROWNIEN"
PRINT
PRINT
PRINT
PRINT
PRINT "Appuyer sur le bouton droit de la souris"
'
REPEAT
  ON MENU
UNTIL MOUSEK=2
'
CLS
'
positionx%=taille_maxi_x%/2
positiony%=taille_maxi_y%/2
'
DRAW positionx%,position_y%
'
position_mediane_x%=positionx%
position_mediane_y%=positiony%
'
FOR i%=11 TO comptage%
  MENU i%,0
```



(suite)

```
  MENU i%,3
NEXT i%
'
MENU 17,1
vitesse%=0
'
MENU 22,1
distance%=6
'
IF resolution%=0
  '
  LINE 0,0,1,188
  DRAW  TO 319,188
  DRAW  TO 319,0
  '
ELSE
  '
  LINE 0,0,1,188
  DRAW  TO 639,188
  DRAW  TO 639,0
  '
ENDIF
'
ON MENU GOSUB traitement
'
' surveillance
'
PAUSE 6
PLOT positionx%,positiony%
'
REPEAT
  ON MENU
  '
  deplacementx%=RANDOM(4*distance%+1)-2*distance%
  deplacementy%=RANDOM(4*distance%+1)-2*distance%
  '
  positionx%=positionx%+deplacementx%
  positiony%=positiony%+deplacementy%
  '
  FOR i%=1 TO vitesse%*5
    ON MENU
  NEXT i%
  '
  DRAW  TO positionx%,positiony%
  IF positionx%>taille_maxi_x%
    positionx%=position_mediane_x%
    positiony%=position_mediane_y%
  ENDIF
```

(suite)

```
  IF positionx%<1
    positionx%=position_mediane_x%
    positiony%=position_mediane_y%
  ENDIF
  IF positiony%>taille_maxi_y%
    positionx%=position_mediane_x%
    positiony%=position_mediane_y%
  ENDIF
  IF positiony%<1
    positionx%=position_mediane_x%
    positiony%=position_mediane_y%
  ENDIF
  PLOT positionx%,positiony%
  '
UNTIL MOUSEK=2
'
CLOSEW 0
'
PROCEDURE traitement1
  '
  MENU OFF
  information
  '
RETURN
'
PROCEDURE traitement
  '
  MENU OFF
  '
  IF MENU(0)>14 AND MENU(0)<18
    '
    FOR j%=15 TO 17
      MENU j%,0
    NEXT j%
    '
    MENU MENU(0),1
    '
    SELECT MENU(0)
    CASE 15
      vitesse%=10
    CASE 16
      vitesse%=5
    CASE 17
      vitesse%=0
    ENDSELECT
    '
  ENDIF
  '
```



(suite)

```
IF MENU(0)>19 AND MENU(0)<23
  '
  FOR j%=20 TO 22
    MENU j%,0
  NEXT j%
  '
  MENU MENU(0),1
  '
  SELECT MENU(0)
  CASE 20
    distance%=1
  CASE 21
    distance%=3
  CASE 22
    distance%=6
  ENDSELECT
  '
ENDIF
'
IF MENU(0)=12
  EDIT
ENDIF
'
IF MENU(0)=11
  CLS
  '
  IF resolution%=0
    '
    LINE 0,0,1,188
    DRAW  TO 319,188
    DRAW  TO 319,0
    '
  ELSE
    '
    LINE 0,0,1,188
    DRAW  TO 639,188
    DRAW  TO 639,0
    '
  ENDIF
  '
  positionx%=position_mediane_x%
  positiony%=position_mediane_y%
  '
  PLOT position_mediane_x%,position_mediane_y%
  '
ENDIF
'
IF MENU(0)=1
```

(suite)

```
    information
  ENDIF
  '
RETURN
'
PROCEDURE information
  '
  SGET ecran$
  '
  IF resolution%=0
    '
    DEFFILL 3,2,9
    PRBOX 100,50,250,150
    LOCATE 14,10
    DEFTEXT 2,2,0,6
    PRINT "MOUVEMENT BROWNIEN"
    '
  ELSE
    '
    DEFFILL 3,2,9
    PRBOX 155,50,500,150
    LOCATE 13,3
    DEFTEXT 2,2,0,32
    PRINT "MOUVEMENT BROWNIEN"
    '
  ENDIF
  '
  REPEAT
  UNTIL MOUSEK<>0
  '
  SPUT ecran$
  '
  DEFTEXT 1,0,0,6
  '
RETURN
```

Dès le début, la résolution de l'écran est recherchée et sauvegardée grâce à la fonction **XBIOS(4)**. Selon celle-ci, le programme est lancé ou non.

Un tableau pour le menu est créé sous le nom **tableau_brown**, celui-ci étant chargé à partir des lignes de DATA.

Les fonctions du menu sont d'abord mise en veille, sauf une pour permettre l'affichage de la (très simple) page de présentation attendant l'appui sur le bouton droit de la souris.

Après quoi le menu est validé, un cadre dessiné selon la résolution, puis la surveillance du menu activée.



Pendant cette surveillance, la particule est mise en mouvement, pseuso-aléatoirement.

On trouve ensuite la première procédure de traitement du menu proposé lors de la présentation (**traitement1**).

Vient la procédure de traitement principale (**traitement**). Selon les différents points du menu, la vitesse ou la distance sont affectées. Retour aussi à l'éditeur si la sélection est égale à 12 (**quitter**), ou réinitialisation de l'écran en cas de sélection 11 (**initialiser**).

En fin de programme se trouve la procédure d'information du logiciel.

EXTENSION DU MOUVEMENT BROWNIEN

Bien qu'intéressant au point de vue physique, le mouvement brownien peut aussi nous être utile pour réaliser quelques figures graphiquement intéressantes.

Nous allons ici vous initier aux possibilités de modification du programme pour obtenir ces figures graphiques.

Aidons nous de Logo

Toujours sous GFA Basic, nous allons utiliser les possibilités offertes par ce langage d'utiliser la célèbre tortue du Logo, pour faire effectuer à nos particules leurs déplacements.

Rappelons d'abord les syntaxes de la tortue utilisable en GFA 3.5.

L'activation de la tortue s'effectue à l'aide de l'instruction **DRAW**, suivie d'une expression alphanumérique.

Dans cette expression seront placés tous les ordres Logo, qui sont:

fd depl (*forward*)	Déplace la tortue de **depl** graduations en avant.
bk depl (*backward*)	Fait reculer la tortue de **depl** graduations en arrière.
sx grad (*scale x*)	Gradue le déplacement horizontal en grad pixels.
sy grad (*scale y*)	Gradue le déplacement vertical en grad pixels.
lt angle (*left turn*)	Oblige la tortue à effectuer une rotation vers la gauche de **angle** degrés.
rt angle (*right turn*)	Oblige la tortue à effectuer une rotation vers la droite de **angle** degrés.
tt angle (*turn to*)	Définie l'angle absolu de la tortue.
ma x,y (*move absolute*)	Oblige la tortue à se positionner en x,y.
mr x,y (*move relative*)	Oblige la tortue à se positionner en x,y, position calculée par rapport à la position courante de la tortue.
da x,y (*draw absolute*)	Dessine une ligne entre la position courante de la tortue et la position x,y d'arrivée.
dr x,y (*draw relative*)	Dessine une ligne entre la position courante de la tortue et la position relative x,y par rapport à cette position courante.

co **couleur** (*COLOR*) — Fixe la couleur du dessin.

pu (*pen up*) — Lève le crayon de la tortue pour interdire l'écriture durant les déplacements.

pd (*pen down*) — Baisse le crayon pour activer l'écriture durant les déplacements.

Exemples de syntaxe :

```
PROCEDURE carre
   DRAW "ma 100,100 pd fd10 lt90 fd10 lt90 fd10 lt90
        fd10 lt90"
RETURN
PROCEDURE bizarre
   DRAW "ma 100,100"
   FOR i% = 1 TO 360
        DRAW "fd5"
        DRAW "lt",i%
   NEXT i%
RETURN
```

Le programme

Pour réaliser notre programme graphique de «tortue brownienne», nous avons utilisé une «graine» graphique, constitué de déplacements qui seront répétés dans le programme (voir figure 5).

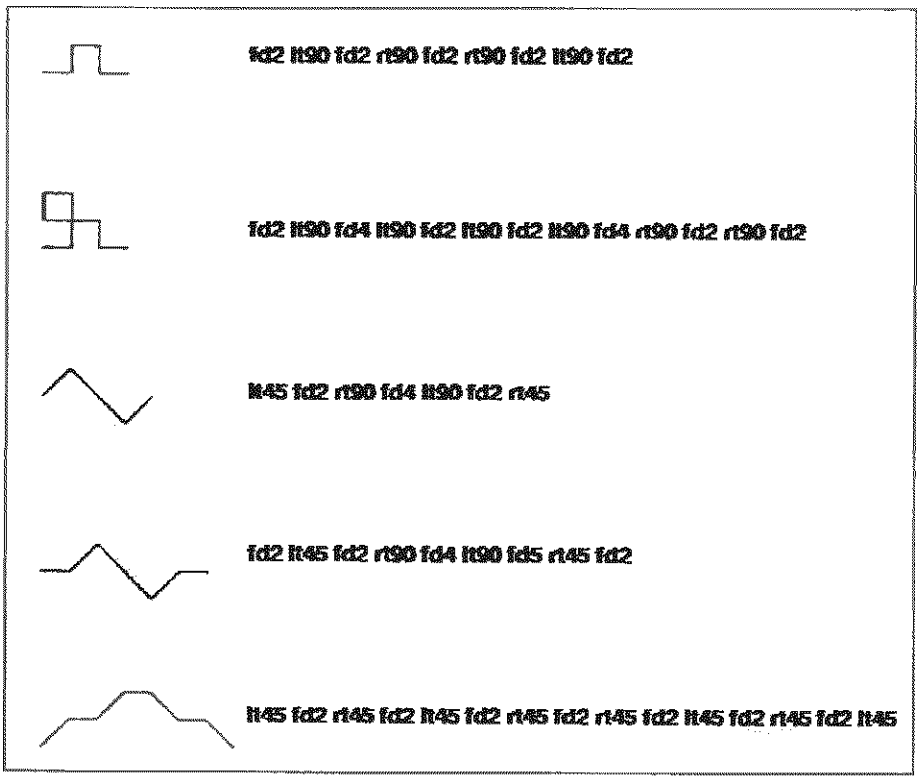


Fig. 5 : Exemples de graines pour le mouvement brownien.



La fonction aléatoire n'affectera plus cette fois la position du point suivant, mais l'angle imposé à la tortue (dans le programme proposé, l'angle pourra être de 0, 90 ou 270 degrés, et ce aléatoirement

Cet angle sera imposé à la tortue juste avant de décrire la graine choisie (voir les lignes de programmes) :

DRAW "lt",angle%/1,4
DRAW "lt45 fd2 rt45 fd2 rt45 fd4 lt45 fd2 lt45 fd2"

Nous avons conservé la même structure de menu que précédemment, sauf pour le choix de la distance de déplacement qui, cette fois-ci, est caduque, la graine étant imposée.

```
  REM ***********************************
  REM *** SIMULATION DE DEPLACEMENT ***
  REM ***  D'UNE PARTICULE ANIMEE   ***
  REM ***  D'UN MOUVEMENT BROWNIEN  ***
  REM ***  PARTIELLEMENT DETERMINE  ***
  REM ***********************************
  REM
  '
  ' acces a la resolution ecran
  '
  resolution%=XBIOS(4)
  '
  ' traitement de la resolution
  '
  IF resolution%=0
    taille_maxi_x%=319
    taille_maxi_y%=199
  ELSE
    IF resolution%=1
      taille_maxi_x%=639
      taille_maxi_y%=199
    ELSE
      ALERT 3,"Ce programme est|prevu pour les resolutions|moyennes ou
basses",1,"RETURN",variable%
      EDIT
    ENDIF
  ENDIF
  '
  ' dimensionnement et preparation
  '     d'un tableau de menu
  '
  DIM tableau_brown$(30)
  '
  comptage%=-1
  '
  RESTORE tableau_brown
```

(suite)

```
'
REPEAT
  INC comptage%
  READ tableau_brown$(comptage%)
UNTIL tableau_brown$(comptage%)="FIN"
'
tableau_brown:
'
DATA  BUREAU,  Informations
DATA ---------------------
DATA "1","2","3","4","5","6"
DATA ""
DATA  INIT,  initialisation  ,  quitter,""
DATA  VITESSE,  lente,  moyenne  ,  rapide,""
DATA ""
DATA FIN
'
' affichage du menu
'
OPENW 0          ! reserve la premiere ligne
MENU tableau_brown$()
'
' mise en veille des fonctions
'
FOR i%=13 TO comptage%
  MENU i%,1
NEXT i%
'
' affichage du menu
'
OPENW 0
'
MENU tableau_brown$()
'
' prepare la surveillance
'
ON MENU GOSUB traitement1
'
' mise en veille des fonctions
'
FOR i%=3 TO comptage%
  MENU i%,2
NEXT i%
'
CLS
'
LOCATE 5,5
PRINT "MOUVEMENT BROWNIEN"
```



(suite)

```
LOCATE 4,8
PRINT "PARTIELLEMENT DETERMINE"
PAUSE 150
'
CLS
'
positionx%=taille_maxi_x%/2
positiony%=taille_maxi_y%/2
'
DRAW positionx%,position_y%
'
position_mediane_x%=positionx%
position_mediane_y%=positiony%
'
FOR i%=11 TO comptage%
  MENU i%,0
  MENU i%,3
NEXT i%
'
MENU 17,1
vitesse%=0
'
IF resolution%=0
  '
  LINE 0,0,1,188
  DRAW  TO 319,188
  DRAW  TO 319,0
  '
ELSE
  '
  LINE 0,0,1,188
  DRAW  TO 639,188
  DRAW  TO 639,0
  '
ENDIF
'
ON MENU GOSUB traitement
'
' surveillance
'
PAUSE 6
'
DRAW "ma",position_mediane_x%,position_mediane_y%
'
REPEAT
  ON MENU
  '
  alea%=0
```

(suite)

```
  REPEAT
    alea%=RANDOM(4)
  UNTIL alea%<>0
  '
  SELECT alea%
  CASE 1
    angle%=0
  CASE 2
    angle%=90
  CASE 3
    angle%=270
  ENDSELECT
  '
  FOR i%=1 TO vitesse%*5
    ON MENU
  NEXT i%
  '
  DRAW "lt",angle%/1,4
  DRAW "lt45 fd2 rt45 fd2 rt45 fd4 lt45 fd2 lt45 fd2"
  '
UNTIL MOUSEK=2
'
CLOSEW 0
'
PROCEDURE traitement1
  '
  MENU OFF
  information
  '
RETURN
'
PROCEDURE traitement
  '
  MENU OFF
  '
  IF MENU(0)>14 AND MENU(0)<18
    '
    FOR j%=15 TO 17
      MENU j%,0
    NEXT j%
    '
    MENU MENU(0),1
    '
    SELECT MENU(0)
    CASE 15
      vitesse%=10
    CASE 16
      vitesse%=5
```



(suite)

```
    CASE 17
      vitesse%=0
    ENDSELECT
    '
  ENDIF
  '
  IF MENU(0)>19 AND MENU(0)<23
    '
    FOR j%=20 TO 22
      MENU j%,0
    NEXT j%
    '
    MENU MENU(0),1
    '
  ENDIF
  '
  IF MENU(0)=12
    EDIT
  ENDIF
  '
  IF MENU(0)=11
    CLS
    '
    DRAW "ma",position_mediane_x%,position_mediane_y%
    '
    IF resolution%=0
      '
      LINE 0,0,1,188
      DRAW  TO 319,188
      DRAW  TO 319,0
      '
    ELSE
      '
      LINE 0,0,1,188
      DRAW  TO 639,188
      DRAW  TO 639,0
      '
    ENDIF
    '
    '
  ENDIF
  '
  IF MENU(0)=1
    information
  ENDIF
  '
RETURN
'
```

(suite)

```
PROCEDURE information
  '
  SGET ecran$
  '
  IF resolution%=0
    '
    DEFFILL 3,2,9
    PBOX 100,50,250,150
    LOCATE 14,10
    DEFTEXT 2,2,0,6
    PRINT "MOUVEMENT BROWNIEN"
    LOCATE 20,12
    PRINT "AVEC"
    LOCATE 15,14
    PRINT "PREDETERMINATION"
    '
  ELSE
    '
    DEFFILL 3,2,9
    PBOX 155,50,500,150
    LOCATE 13,3
    DEFTEXT 2,2,0,32
    PRINT "MOUVEMENT BROWNIEN"
    '
  ENDIF
  '
  REPEAT
  UNTIL MOUSEK<>0
  '
  SPUT ecran$
  '
  DEFTEXT 1,0,0,6
  '
RETURN
```

Nous vous offrons en figures 6, 7 et 8 quelques exemples de graphismes qu'il est possible de réaliser avec différentes graines.

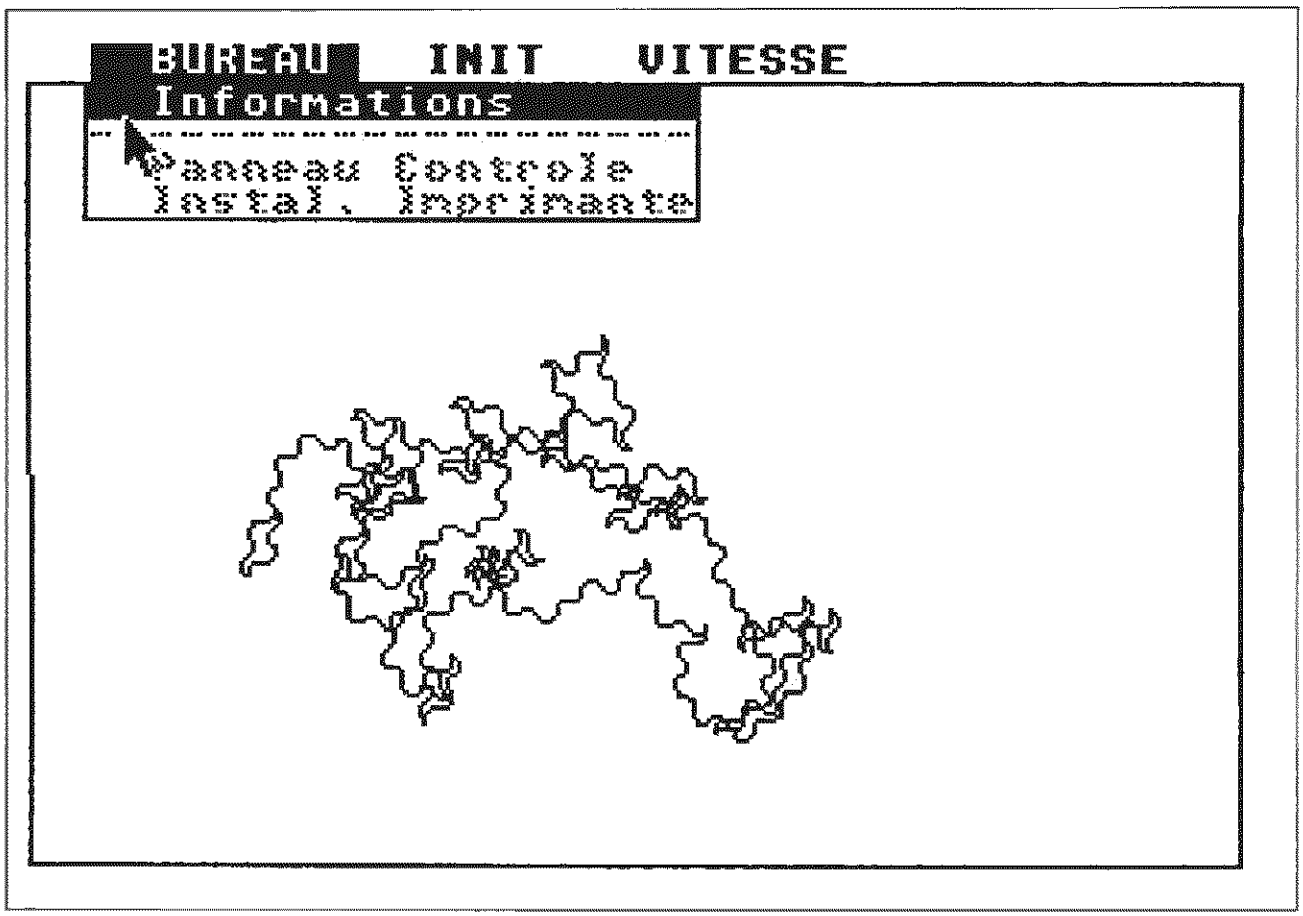


Fig. 6

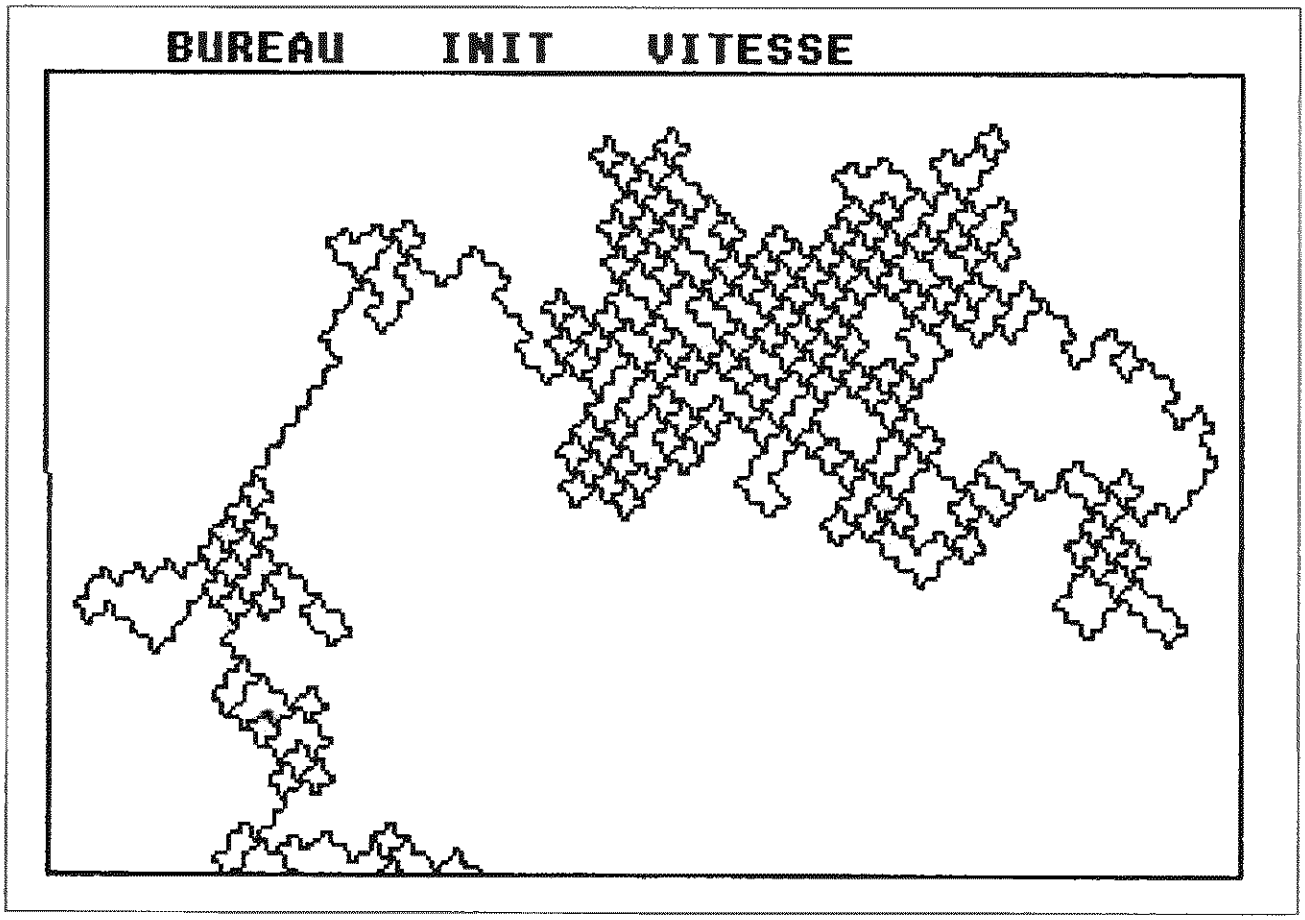


Fig. 7

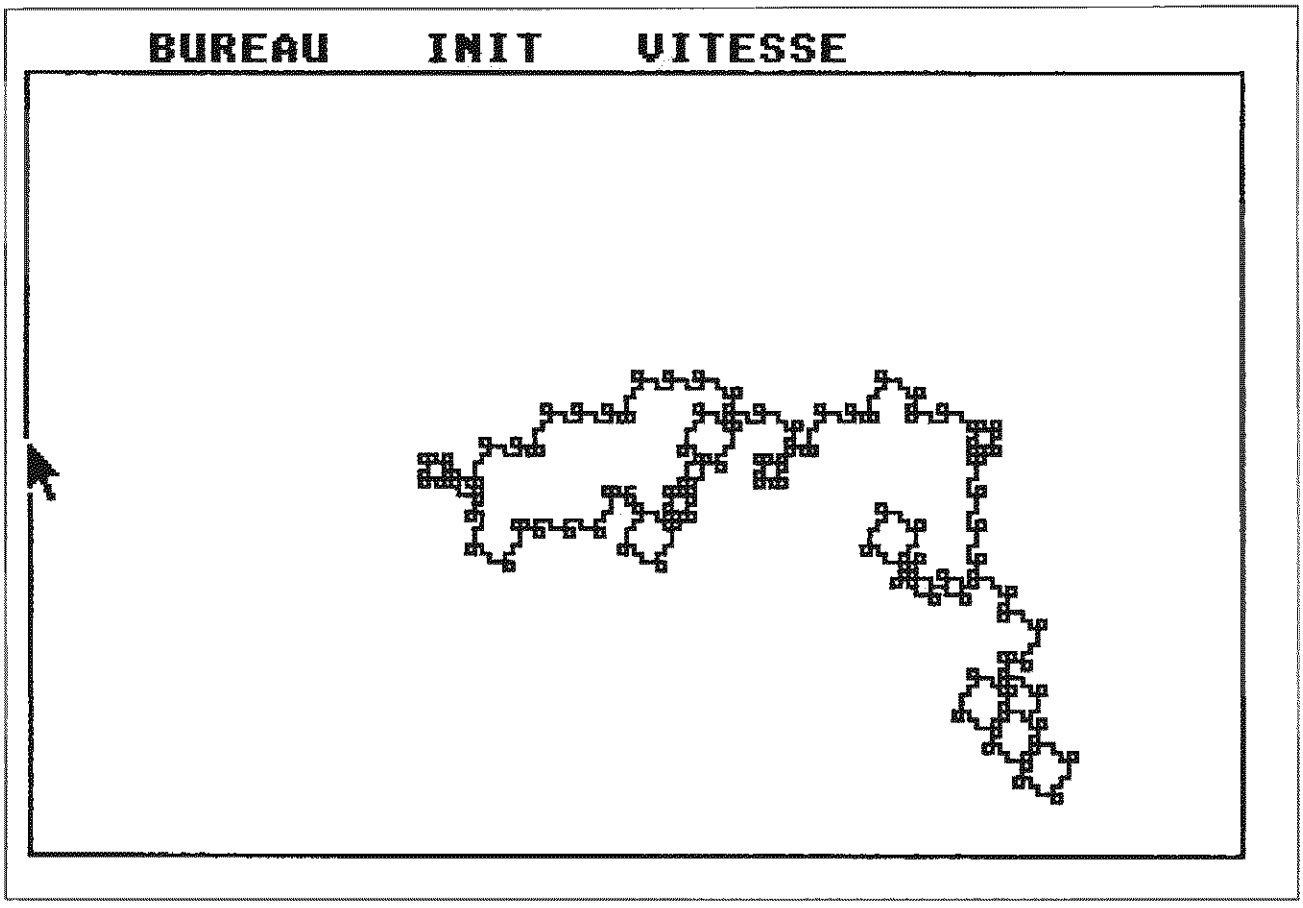


Fig. 8

Nous avons volontairement arrêté tôt les déplacements de la tortue pour les besoins de la copie graphique, mais il est possible, en choisissant bien la graine initiale et après un temps non négligeable, de réaliser des motifs graphiques très originaux (comme pourrait le laisser entrevoir la figure 7).



5/5.7

Agrégation fractale par diffusion limitée

Dans nos précédentes études, nous avons parcouru la complexité de l'ensemble de Mandelbrot, et abordé la simplicité des mouvements de type brownien.

Rappelons que l'ensemble de Mandelbrot, caractérisé par l'itération d'une équation mathématique sur les nombres complexes, est un ensemble appelé fractal.

Nous allons, dans ce chapitre, créer d'autres objets fractals (et non fractaux), en utilisant nos connaissances sur le mouvement brownien.

Nous vous conseillons de relire les pages 1 et 2 de la Partie 5/5.5 pour appréhender la notion de fractal.

L'AGRÉGATION FRACTALE CENTRALE OU COMMENT SE DÉPOSE LE ZINC SUR UNE ÉLECTRODE

Le principe

Le mouvement brownien consiste en un déplacement aléatoire d'une particule dans un fluide. Dans notre étude sur le mouvement brownien, nous avions visualisé le mouvement de ces particules, nous allons cette fois-ci utiliser ce mouvement pour créer des objets fractals, utilisés par les physiciens en tant que modèles mathématiques de phénomènes physiques.

Le modèle que nous vous proposons de tracer ici est dénommé croissance fractale, dite d'agrégation par diffusion limitée; croissance qui, bizarrement, ressemble à la formation d'un dépôt de particules de zinc sur une électrode.

En premier lieu, nous aurons besoin d'une particule centrale, qui sera immobile, représentant l'électrode plongée dans l'électrolyte au zinc.

Ensuite, des particules seront engendrées l'une après l'autre à partir d'un cercle centré sur notre électrode. Ces particules seront animées d'un

mouvement brownien, semblable à la démarche aléatoire d'un individu sortant d'un repas plus qu'arrosé.

Lorsque, à la suite d'un interminable parcours, notre particule rencontre l'électrode centrale, elle s'y dépose pour s'y immobiliser.

Une nouvelle particule est alors engendrée pour un parcours tout aussi erratique.

En supposant que notre particule s'éloigne trop du noyau central, elle sera considérée comme perdue, et une nouvelle particule sera engendrée. Nous prendrons comme éloignement critique, induisant la perte d'une particule, le cercle à partir duquel seront engendrées les particules.

La figure 1 représente le principe de notre croissance fractale.

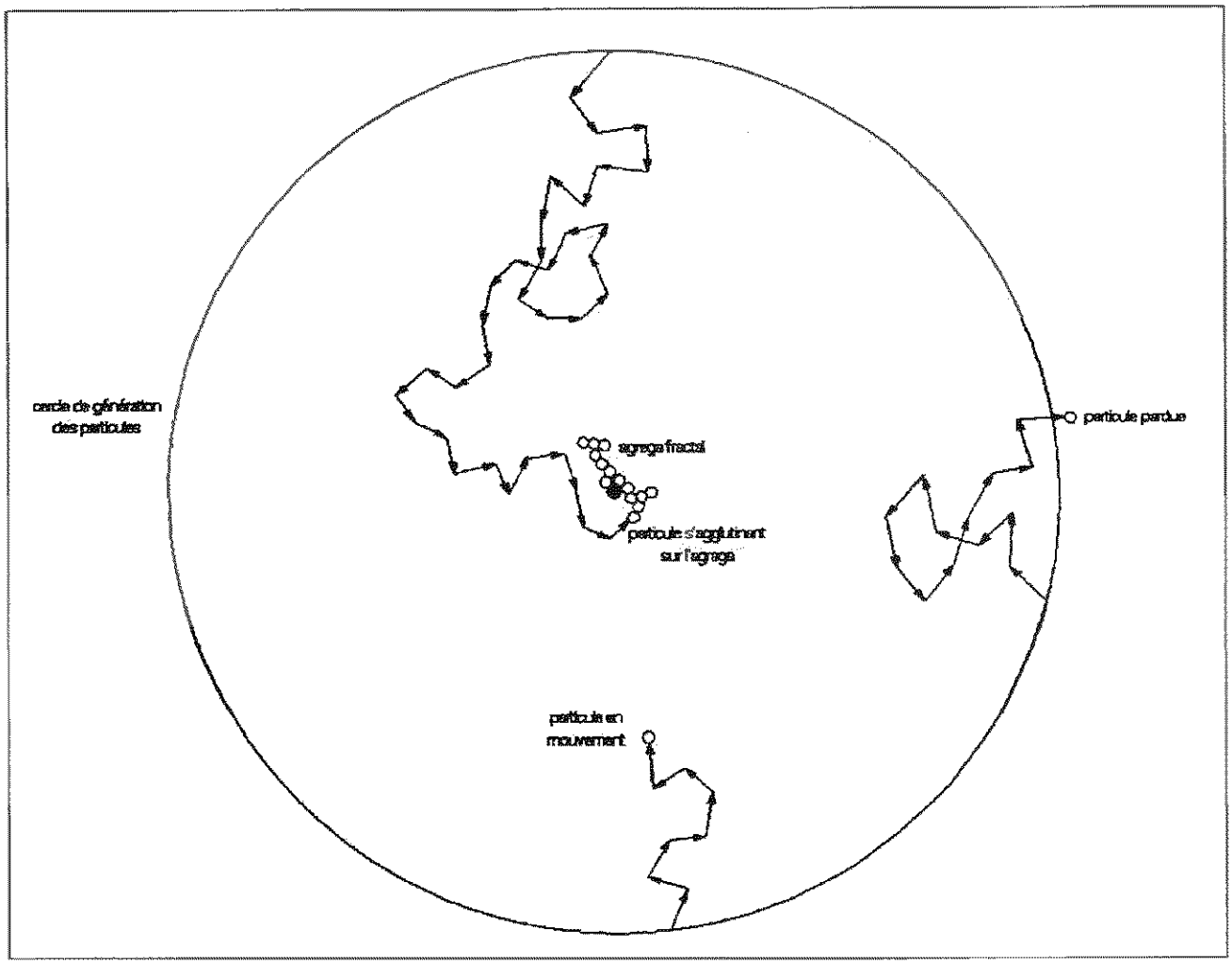


Fig. 1 : Principe de l'agrégation fractale par diffusion limitée.



Le principe, ainsi déterminé, nous permettra de représenter des agrégas fractals ressemblant à ceux des figures 7 et 8. Ces représentations ressemblent fort à celle de l'electrolyse d'une solution de zinc, les particules de zinc se positionnant autour de l'électrode centrale de façon similaire.

La raison de la croissance fractale

Mais pourquoi cette agglutinement de type fractal, plutôt que l'apparition d'une masse informe sans aucune propriété particulière ? Pourquoi ne trouve-t-on pas non plus de redondance dans les formes ?

Aucun raisonnement purement mathématique ne parvient à expliquer ce phénomène.

Par contre, avec un raisonnement qualitatif, et en tenant un discours proche du raisonnement probabiliste et déterministe, on arrive à appréhender le pourquoi de la création de ces objets.

Sur la particule servant d'électrode, la première particule va se déposer au hasard sur l'un des côtés.

Il en sera de même pour les quelques premières particules, qui s'agglutineront indifféremment sur les particules déjà immobilisées.

Ainsi se formera tout autour du groupe immobile une surface contenant de nombreuses aspérités.

Une nouvelle particule s'approchant de cette surface aura plus de chance de se trouver en contact avec une particule saillante du groupe, plutôt qu'avec une particule profondément ancrée.

Chaque nouvelle particule augmentera encore le phénomène, créant ainsi une forme où se développent de plus en plus vite et fortement les parties saillantes de l'objet. Cette particularité est appelé l'instabilité de croissance.

Un terme apparaît aussi dans les études sur ce phénomène de croissance fractal: le bruit des particules. Il s'agit du phénomène produit par l'agrégation des particules sur l'amas, et qui formerait une sorte de champ, que l'on pourrait peut-être appeler «magnétofractal», autour de l'objet fractal, pour mieux piéger les nouvelles particules. Ce bruit est parfois appelé bruit statistique.

Signalons que, de par sa constitution, l'objet fractal créé ressemble à celui de la formation du dépôt de zinc dans une cuve à électrolyse, mais que la création de ce dernier n'a rien à voir avec le mouvement brownien, car ce sont les champs électrostatiques qui provoquent l'agrégation autour de l'électrode centrale.

L'algorithme général pour votre Atari

Nous vous proposons donc de visualiser une croissance fractale sur l'écran de votre Atari (le programme a été réalisé sur 520 STf).

Le programme consistera à positionner, tout d'abord, une particule faisant office d'électrode centrale au milieu de l'écran.

Une particule sera ensuite mise en mouvement à partir d'un cercle le plus éloigné possible de son centre.

Un premier déplacement, de un pixel, s'effectuera dans une direction choisie au hasard.

Si la particule se trouve hors du cercle, une nouvelle particule est engendrée.

S'il y a collage, la particule est arrêtée et une nouvelle particule est engendrée, sinon la particule est de nouveau déplacée, et les étapes de test précédentes à nouveau examinées.

Afin de pouvoir arrêter notre programme, un test sur la touche <ESC> est effectué.

Vous trouverez en figure 2 la représentation sous forme d'algorigramme de cet algorithme.

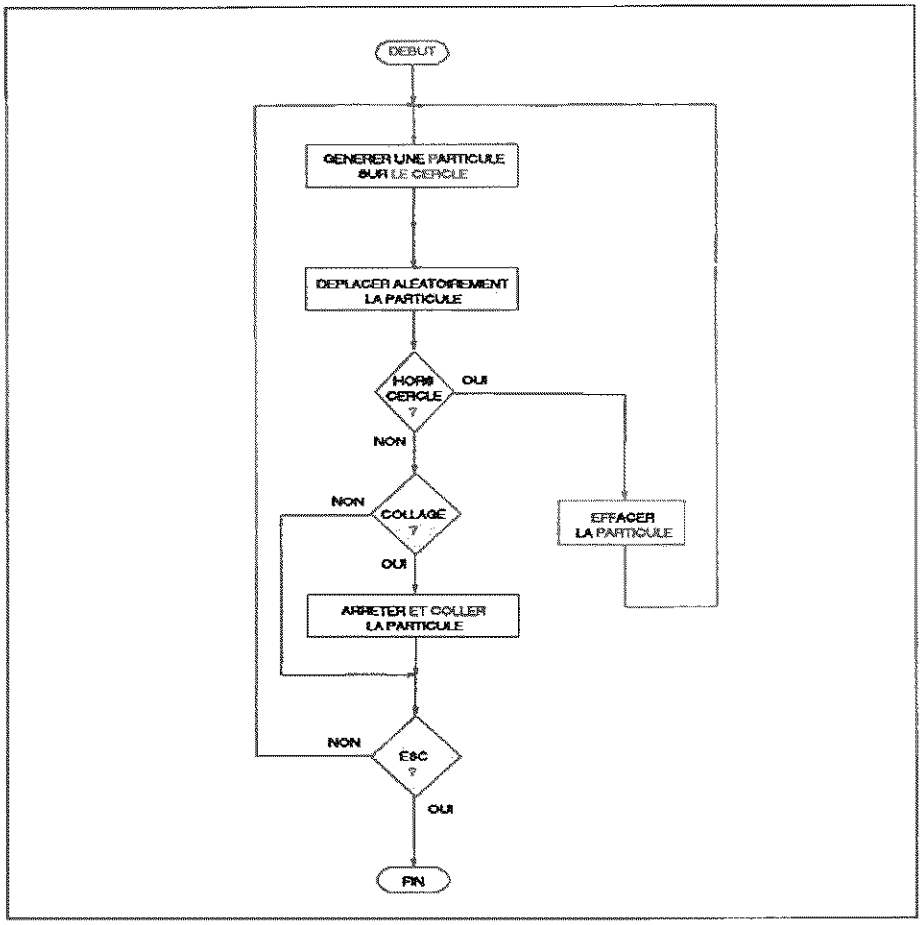


Fig. 2 : Algorithme général d'agrégation fractale.



Le programme et son utilisation

```
REM ****************************
REM ***                          ***
REM *** AGREGATION FRACTALE ***
REM ***                          ***
REM ****************************
REM
REM ------------------------------------------
REM ***   Agregation fractale par  ***
REM *** accumulation de particules ***
REM ***          au  centre          ***
REM ------------------------------------------
'
CLS
IF XBIOS(4)<>0
  ALERT 3,"Ce programme|fonctionne en|basse resolution",1," OK",variable%
  EDIT
ENDIF
'
PRINT TAB(5);"AGREGATION FRACTALE CENTRALE"
PRINT
PRINT
PRINT
PRINT "Ce programme simule une croissance"
PRINT
PRINT "fractale d'un amas de particules,"
PRINT
PRINT "animées d'un mouvement pseudo-brownien,"
PRINT
PRINT "autour d'un noyau central"
'
PRINT
PRINT
PRINT
'
'
REM *** INITIALISATIONS ***
'
' palette de couleur
'
VSETCOLOR 0,0,0,0
VSETCOLOR 1,15,15,15
VSETCOLOR 2,15,0,0
VSETCOLOR 3,0,15,0
VSETCOLOR 4,0,0,15
VSETCOLOR 5,15,15,0
VSETCOLOR 6,11,5,0
VSETCOLOR 7,0,11,0
```

(suite)

```
VSETCOLOR 8,11,0,11
VSETCOLOR 9,0,15,15
'
' tableau de points
'
DIM coordonneex%(5000)
DIM coordonneey%(5000)
DIM couleur_point%(5000)
'
REM * nombre de particules *
'
couleur%=1
nombre_de_particules%=1
'
REM * RECHERCHE PRECEDENTE ? *
'
PRINT "Desirez-vous charger une precedente"
PRINT "recherche (O/N) ?"
'
REPEAT
  reponse$=INKEY$
UNTIL reponse$<>""
'
reponse$=UPPER$(reponse$)
'
IF reponse$="O"
  c.hargement
ENDIF
'
CLS
'
REM * point initial *
'
position_initialex%=160
position_initialey%=100
'
couleur%=1
COLOR couleur%
PLOT position_initialex%,position_initialey%
'
' Initialisation du premier element du tableau
'
coordonneex%(1)=position_initialex%
coordonneey%(1)=position_initialey%
couleur_point%(1)=couleur%
'
REM *  diametre du cercle   *
REM *    pour rattraper la  *
```



(suite)

```
REM *    resolution ecran    *
REM * en moyenne resolution *
'
diametrex%=90    ! il faut doubler ce parametre
diametrey%=90
'
' calcul disance d'echappement
'
distancemaxi%=(diametrex%+diametrey%)*(diametrex%+diametrey%)/4
'
REM * generation aleatoire *
REM *    pour le cercle     *
'
generateur%=360    ! 360 degre du cercle
'
REM *   AFFICHAGE D'UN    *
REM * TABLEAU ENREGISTRE *
'
IF reponse$="O"
  FOR compteur%=1 TO nombre_de_particules%
    COLOR couleur_point%(compteur%)
    PLOT coordonneex%(compteur%),coordonneey%(compteur%)
    PAUSE 3
  NEXT compteur%
ENDIF
'
drapeau$=""
'
REM * DEPLACEMENT DES PARTICULES *
REM ******************************
'
WHILE drapeau$<>CHR$(27)
  '
  LOCATE 1,1
  PRINT "Particules : ";
  PRINT nombre_de_particules%
  '
  REM * determination position de depart *
  '
  angle%=RANDOM(generateur%)
  '
  positionx%=diametrex%*COS(angle%)+position_initialex%
  positiony%=diametrey%*SIN(angle%)+position_initialey%
  '
  COLOR 1
  PLOT positionx%,positiony%
  '
  REM * calcul des deplacements *
```

(suite)

```
'
  drapeau$=""
  '
  WHILE drapeau$<>CHR$(27) AND drapeau$<>"FIN"
    '
    drapeau$=INKEY$
    '
    ancienne_positionx%=positionx%
    ancienne_positiony%=positiony%
    '
    REM * recherche de deplacement *
    '
    recherche%=RANDOM(8)
    '
    SELECT recherche%
      '
    CASE 0
      deplacementx%=-1
      deplacementy%=-1
    CASE 1
      deplacementx%=-1
      deplacementy%=0
    CASE 2
      deplacementx%=-1
      deplacementy%=1
    CASE 3
      deplacementx%=0
      deplacementy%=-1
    CASE 4
      deplacementx%=0
      deplacementy%=1
    CASE 5
      deplacementx%=1
      deplacementy%=-1
    CASE 6
      deplacementx%=1
      deplacementy%=0
    CASE 7
      deplacementx%=1
      deplacementy%=1
      '
    ENDSELECT
    '
    REM * calcul nouvelle position *
    '
    positionx%=positionx%+deplacementx%
    positiony%=positiony%+deplacementy%
    '
```



(suite)

```
      ' effacement ancienne position
      '
      COLOR 0
      PLOT ancienne_positionx%,ancienne_positiony%
      '
      ' affichage nouvelle position
      '
      COLOR 1
      PLOT positionx%,positiony%
      '
      REM * calcul distance du centre *
      '
      distance%=(positionx%-position_initialex%)*(positionx%-
position_initialex%)+(positiony%-position_initialey%)*(positiony%-
position_initialey%)
      '
      REM * recherche si particule autour *
      '
      IF PTST(positionx%-1,positiony%-1)<>0
        drapeau$="FIN"
      ENDIF
      IF PTST(positionx%-1,positiony%)<>0
        drapeau$="FIN"
      ENDIF
      IF PTST(positionx%-1,positiony%+1)<>0
        drapeau$="FIN"
      ENDIF
      IF PTST(positionx%,positiony%-1)<>0
        drapeau$="FIN"
      ENDIF
      IF PTST(positionx%,positiony%+1)<>0
        drapeau$="FIN"
      ENDIF
      IF PTST(positionx%+1,positiony%-1)<>0
        drapeau$="FIN"
      ENDIF
      IF PTST(positionx%+1,positiony%)<>0
        drapeau$="FIN"
      ENDIF
      IF PTST(positionx%+1,positiony%+1)<>0
        drapeau$="FIN"
      ENDIF
      '
      ' si oui, coller la particule
      '
      IF drapeau$="FIN"
        couleur%=INT(nombre_de_particules%/200) MOD 9+1
        COLOR couleur%
```

(suite)

```
      PLOT positionx%,positiony%
      '
      ' une particule de plus
      '
      nombre_de_particules%=nombre_de_particules%+1
      '
      REM ** Sauvegarde en memoire du tableau **
      '
      coordonneex%(nombre_de_particules%)=positionx%
      coordonneey%(nombre_de_particules%)=positiony%
      couleur_point%(nombre_de_particules%)=couleur%
      '
    ENDIF
    '
    ' la particule s'echappe ?
    '
    IF distance%>distancemaxi%
      drapeau$="FIN"
      COLOR 0
      PLOT positionx%,positiony%
    ENDIF
    '
  WEND
  '
WEND
'
REM * TOUCHE <ESC> ENFONCEE *
REM * VOIR POUR SAUVEGARDE  *
REM ***************************
'
SGET ecran$
CLS
'
' FENETRE DE SAUVEGARDE
'
OPENW #1,30,20,235,100,&H0
TITLEW #1,"SAUVEGARDE"
PRINT "Desirez vous sauver"
PRINT "la recherche"
PRINT "(O/N) ?"
'
REPEAT
  a$=INKEY$
UNTIL a$="o" OR a$="n" OR a$="O" OR a$="N"
'
a$=UPPER$(a$)
'
```



(suite)

```
IF a$="O"
  '
  ' selection du fichier
  '
  FILESELECT "\*.frc","",nom_fichier$
  IF INSTR(nom_fichier$,".")=0
    nom_fichier$=nom_fichier$+".FRC"
  ENDIF
  '
  ' affichage de l'etat de sauvegarde
  '
  CLS
  PRINT "Sauvegarde de : "
  PRINT "     ";
  PRINT nom_fichier$
  '
  PRINT
  LOCATE 1,6
  PRINT "Particules ";
  PRINT " X    Y    Couleur"
  PRINT
  OPEN "o",#5,nom_fichier$
  '
  ' sauvegarde effective
  '
  FOR compteur%=1 TO nombre_de_particules%-1
    LOCATE 1,8
    PRINT "   ";compteur%;"     ";
    PRINT coordonneex%(compteur%);"     ";
    PRINT coordonneey%(compteur%);"       ";
    PRINT couleur_point%(compteur%);"     "
    PRINT
    WRITE
#5,coordonneex%(compteur%),coordonneey%(compteur%),couleur_point%(compteur%)
  NEXT compteur%
  '
  CLOSE #5
  '
ENDIF
'
CLOSEW #1
CLOSEW #0
SPUT ecran$
'
' retour au systeme ou interpreteur
'
EDIT
'
```

(suite)

```
REM *** FIN PROGRAMME PRINCIPAL ***
'
REM * CHARGEMENT D'UN FICHIER *
REM *   FRACTAL ENREGISTRE    *
REM ***************************
'
PROCEDURE c.hargement
  '
  CLS
  '
  ' ouverture fenetre de dialogue
  '
  OPENW #1,30,20,235,100,&H0
  TITLEW #1,"CHARGEMENT"
  '
  ' selection du ficher
  '
  FILESELECT "\*.frc","",nom_fichier$
  '
  IF INSTR(nom_fichier$,".")=0
    nom_fichier$=nom_fichier$+".FRC"
  ENDIF
  '
  ' affichage de l'etat de chargement
  '
  CLS
  PRINT "Chargement de : "
  PRINT "      "
  PRINT nom_fichier$
  '
  nombre_de_particules%=1
  LOCATE 1,6
  PRINT "Particules ";
  PRINT " X     Y     Couleur"
  PRINT
  OPEN "i",#5,nom_fichier$
  '
  ' chargement effectif
  '
  WHILE NOT EOF(#5)
    INPUT
#5,coordonneex%(nombre_de_particules%),coordonneey%(nombre_de_particules%),couleur
_point%(nombre_de_particules%)
    LOCATE 1,8
    PRINT "   ";nombre_de_particules%;"      ";
    PRINT coordonneex%(nombre_de_particules%);"     ";
    PRINT coordonneey%(nombre_de_particules%);"        ";
    PRINT couleur_point%(nombre_de_particules%);"     "
```



```
(suite)

    nombre_de_particules%=nombre_de_particules%+1
  WEND
  '
  CLOSE #5
  '
  CLOSEW #1
  CLOSEW #0
  '
RETURN
'
REM **** FIN PROGRAMME AGREGATION ****
```

Signalons de suite que le programme fonctionne uniquement en basse résolution.

Après lancement, le programme propose le chargement d'un fractal déjà créé (pour reprendre un travail en cours). Sur la disquette, lors de sa parution, nous vous proposons deux agrégats fractals situés dans le sous-répertoire \AGREGA\AUCENTRE\ et munis de l'extension .FRC.

Si vous demandez le chargement d'une croissance fractale, celle-ci, après un temps nécessaire à la mémorisation du tableau de données, s'affichera en version colorée, et la génération de nouvelles particules viendra grossir l'objet. Sinon, une particule centrale est positionnée, et les particules périphériques sont engendrées chacune leur tour.

Sachez que pour obtenir une agrégation fractale intéressante à étudier, il vous faudra attendre quelques heures de calcul, aussi, nous avons prévu la sauvegarde d'un calcul en cours. Lors de l'appui sur la touche <ESC> (pour terminer le calcul), il vous sera proposé la sauvegarde du calcul en cours, que vous pourrez accepter ou non.

Vous trouverez sur les figures 3 à 7 la visualisation de la croissance d'un agrégat par diffusion limitée.

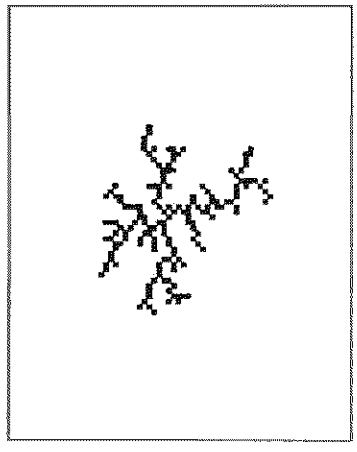


Fig. 3 : 164 particules.

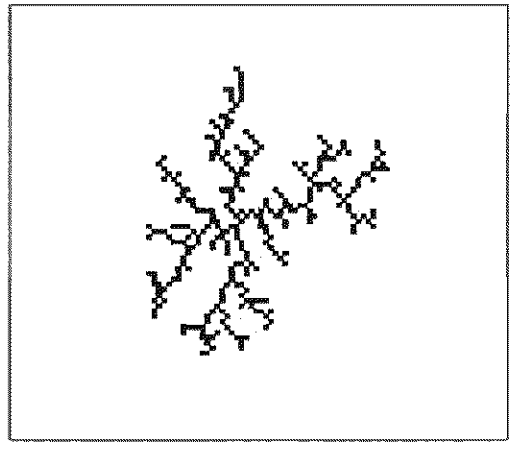


Fig. 4 : 305 particules.

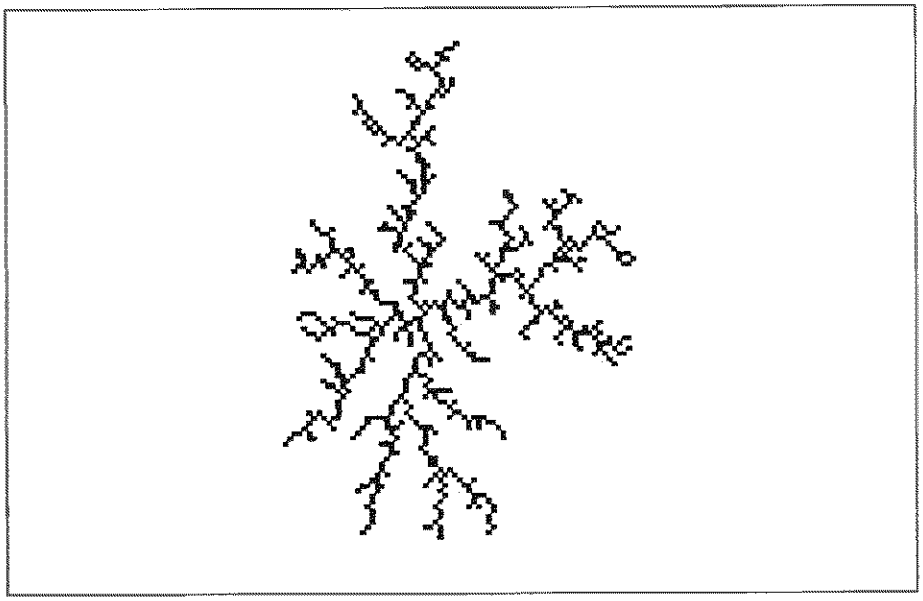


Fig. 5 : 638 particules.

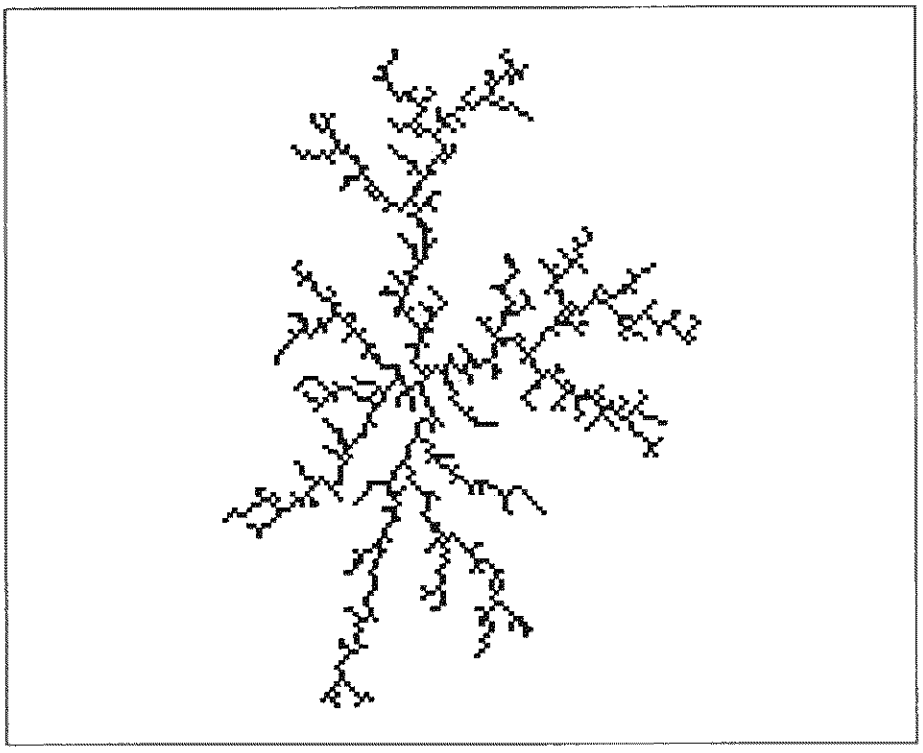


Fig. 6 : 965 particules.

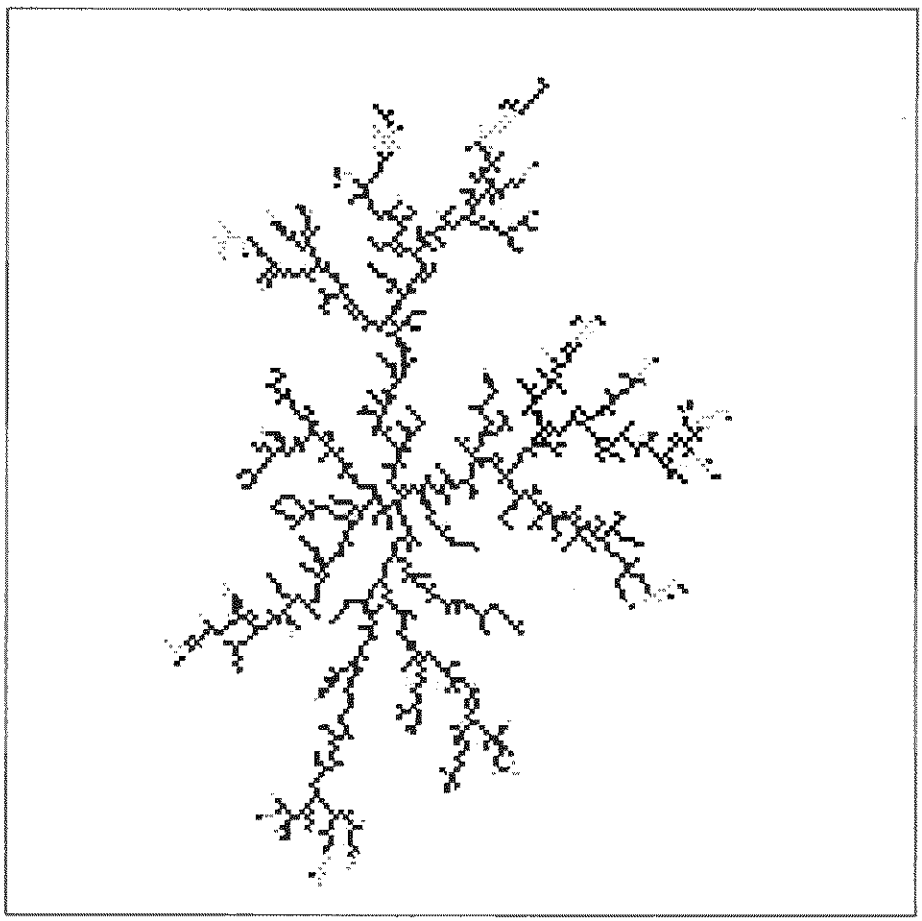


Fig. 7: 1317 particules.

La figure 8 présente la version finale d'un autre agrégat fractal calculé avec le même programme.

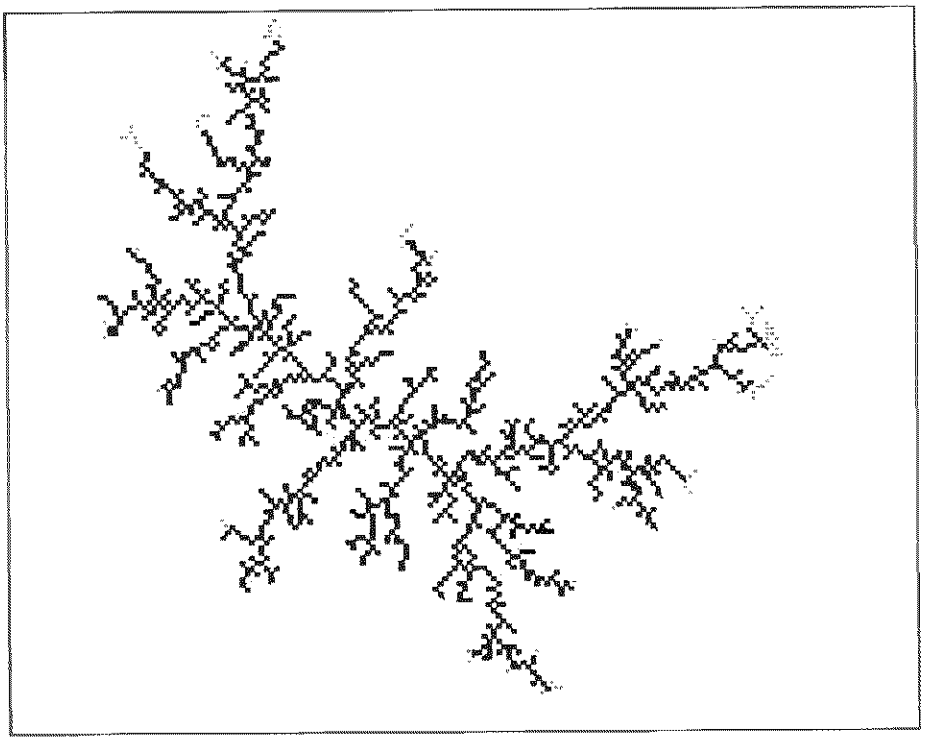


Fig. 8: 1222 particules.

Bien que n'apparaissant pas dans nos figures monochromes, si vous possédez un écran couleur, les particules apparaîtront colorées. Les couleurs sont données en fonction du nombre de particules, celles-ci étant modifiées toutes les 200 particules, ce qui permettra de faire une rapide estimation du nombre de particules en contact.

Au fur et à mesure de leur agrégation, les coordonnées des particules et leurs couleurs sont rangées dans trois tableaux dimensionnés à la valeur 5000 :

— **coordonnée_x%(5000)** pour ranger la coordonnée horizontale ;

— **coordonnée_y%(5000)** pour ranger la coordonnée verticale ;

— **couleur_point%(5000)** pour ranger la couleur de la particule.

Ce tableau sera utilisé pour la sauvegarde sur disquette, et l'affichage ultérieur.

Remarquons que dans la réalité, les particules ne s'ammoncellent pas sur un plan, mais en fait dans l'espace. Ceux qui le désireront, avec du courage, pourront développer le programme en trois dimensions et ainsi représenter des agrégations fractales de type 3-D, comme représenté en figure 9, elles aussi très proches, par exemple, de structures moléculaires existantes.

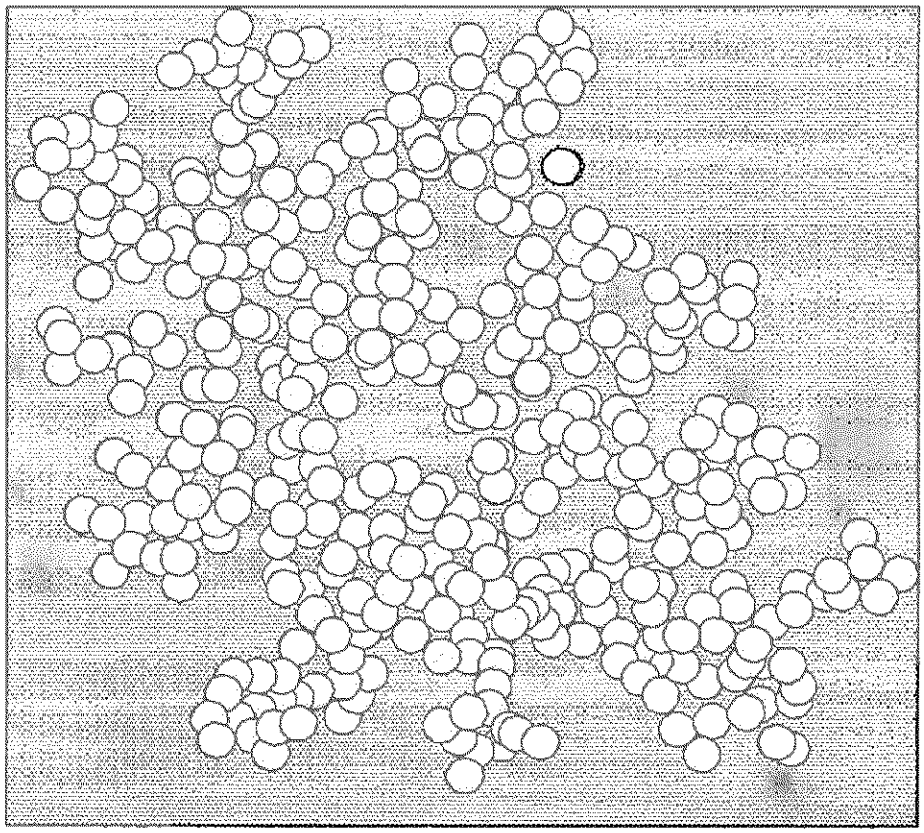


Fig. 9 : Agrégation fractale de type 3-D.

L'AGRÉGATION FRACTALE CIRCULAIRE

Une idée fort simple

Le principe précédent consistait à accumuler les particules engendrées depuis un cercle, sur une particule centrale.

L'idée, maintenant, est d'effectuer l'opération inverse, en considérant que les particules sont émises depuis un point central, et auront tendance à s'accumuler sur un cercle périphérique.

Cette idée pourrait simuler le cheminement de particules de suie dans une cheminée, qui, feux après feux, s'accumulent sur le tubage.

Le programme

Pour la réalisation de ce programme, nous avons repris le programme précédent en effectuant la génération des particules à partir du point central de coordonnées 160,100.

Le déplacement restant de même type, pseudo-aléatoire, les particules seront arrêtées dès qu'elles toucheront le cercle périphérique, ou une particule déjà fixée.

Nous vous proposons donc ci-dessous le listing du programme modifié, et largement simplifié.

```
REM ***************************
REM ***                       ***
REM *** AGREGATION FRACTALE ***
REM ***                       ***
REM ***************************
REM
REM ---------------------------------------
REM ***   Agregation fractale par   ***
REM *** accumulation de particules ***
REM ***       en peripherie         ***
REM ---------------------------------------
'
CLS
IF XBIOS(4)<>0
  ALERT 3,"Ce programme|fonctionne en|basse resolution",1," OK",variable%
  EDIT
ENDIF
'
PRINT TAB(5);"AGREGATION FRACTALE CIRCULAIRE"
PRINT
PRINT
PRINT
PRINT "Ce programme simule une croissance"
PRINT
PRINT "fractale d'un amas de particules,"
PRINT
PRINT "animées d'un mouvement pseudo-brownien,"
PRINT
PRINT "sur un support circulaire"
'
PRINT
PRINT
PRINT
'
'
REM *** INITIALISATIONS ***
'
' palette de couleur
'
VSETCOLOR 0,0,0,0
VSETCOLOR 1,15,15,15
VSETCOLOR 2,15,0,0
```



(suite)

```
VSETCOLOR 3,0,15,0
VSETCOLOR 4,0,0,15
VSETCOLOR 5,15,15,0
VSETCOLOR 6,11,5,0
VSETCOLOR 7,0,11,0
VSETCOLOR 8,11,0,11
VSETCOLOR 9,0,15,15
'
' tableau de points
'
REM * nombre de particules *
'
couleur%=1
nombre_de_particules%=1
'
CLS
'
REM * point initial *
'
position_initialex%=160
position_initialey%=100
'
REM dessin du cercle
'
couleur%=1
COLOR couleur%
CIRCLE 160,100,100
'
' Initialisation du premier element du tableau
'
coordonneex%(1)=position_initialex%
coordonneey%(1)=position_initialey%
couleur_point%(1)=couleur%
'
'
REM ******************************
REM * DEPLACEMENT DES PARTICULES *
REM ******************************
'
WHILE drapeau$<>CHR$(27)
  '
  LOCATE 1,1
  PRINT "Particules : ";
  PRINT nombre_de_particules%
  '
  REM * determination position de depart *
  '
  positionx%=160
```

(suite)

```
positiony%=100
'
COLOR 1
PLOT positionx%,positiony%
'
REM * calcul des deplacements *
'
drapeau$=""
'
WHILE drapeau$<>CHR$(27) AND drapeau$<>"FIN"
  '
  drapeau$=INKEY$
  '
  ancienne_positionx%=positionx%
  ancienne_positiony%=positiony%
  '
  REM * recherche de deplacement *
  '
  recherche%=RANDOM(8)
  '
  SELECT recherche%
    '
  CASE 0
    deplacementx%=-1
    deplacementy%=-1
  CASE 1
    deplacementx%=-1
    deplacementy%=0
  CASE 2
    deplacementx%=-1
    deplacementy%=1
  CASE 3
    deplacementx%=0
    deplacementy%=-1
  CASE 4
    deplacementx%=0
    deplacementy%=1
  CASE 5
    deplacementx%=1
    deplacementy%=-1
  CASE 6
    deplacementx%=1
    deplacementy%=0
  CASE 7
    deplacementx%=1
    deplacementy%=1
    '
  ENDSELECT
```



(suite)

```
'
REM * calcul nouvelle position *
'
positionx%=positionx%+deplacementx%
positiony%=positiony%+deplacementy%
'
' effacement ancienne position
'
COLOR 0
PLOT ancienne_positionx%,ancienne_positiony%
'
' affichage nouvelle position
'
COLOR 1
PLOT positionx%,positiony%
'
REM * recherche si particule autour *
'
IF PTST(positionx%-1,positiony%-1)<>0
  drapeau$="FIN"
ENDIF
IF PTST(positionx%-1,positiony%)<>0
  drapeau$="FIN"
ENDIF
IF PTST(positionx%-1,positiony%+1)<>0
  drapeau$="FIN"
ENDIF
IF PTST(positionx%,positiony%-1)<>0
  drapeau$="FIN"
ENDIF
IF PTST(positionx%,positiony%+1)<>0
  drapeau$="FIN"
ENDIF
IF PTST(positionx%+1,positiony%-1)<>0
  drapeau$="FIN"
ENDIF
IF PTST(positionx%+1,positiony%)<>0
  drapeau$="FIN"
ENDIF
IF PTST(positionx%+1,positiony%+1)<>0
  drapeau$="FIN"
ENDIF
'
' si oui, coller la particule
'
IF drapeau$="FIN"
  couleur%=INT(nombre_de_particules%/200) MOD 9+1
  COLOR couleur%
```

```
(suite)

        PLOT positionx%,positiony%
        '
        ' une particule de plus
        '
        nombre_de_particules%=nombre_de_particules%+1
        '
      ENDIF
      '
    WEND
    '
WEND
'
REM **** FIN PROGRAMME AGREGATION AUTOUR ****
```

Nous n'avons pas fait apparaître dans ce programme les fonctions de sauvegarde et de chargement de précédents fractals, mais les programmeurs en GFA 3 pourront les insérer facilement en se reportant au précédent programme.

Nous vous proposons en figures 10, 11 et 12 un exemple de formation d'une structure fractale sur ce cercle.

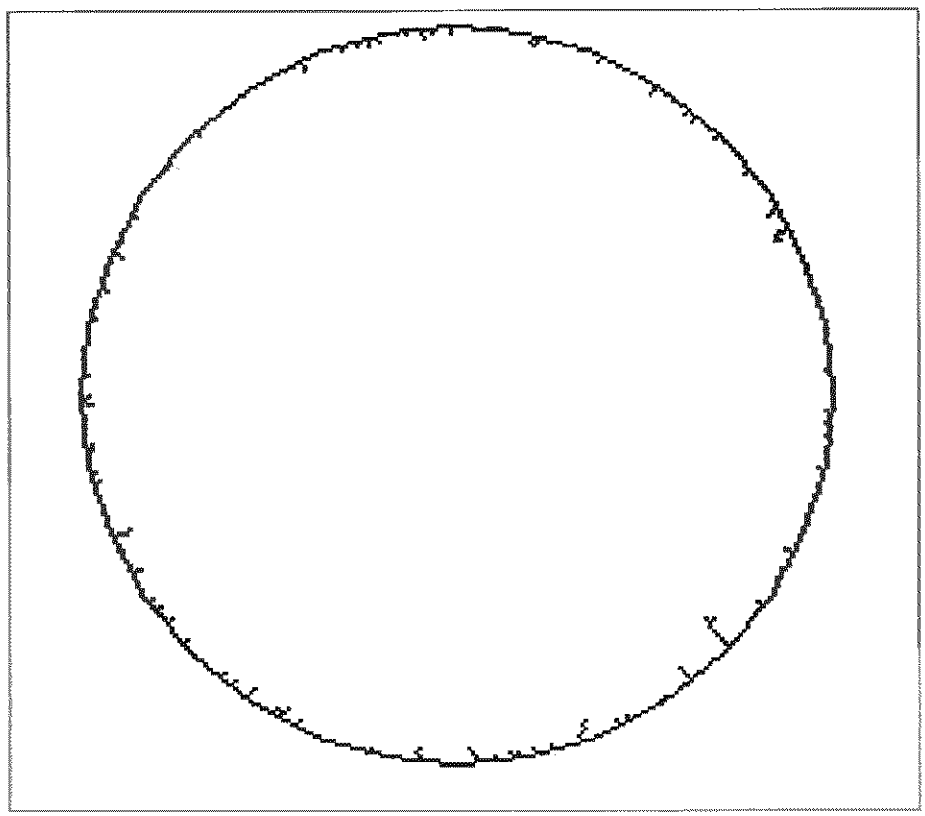


Fig. 10 : 110 particules.

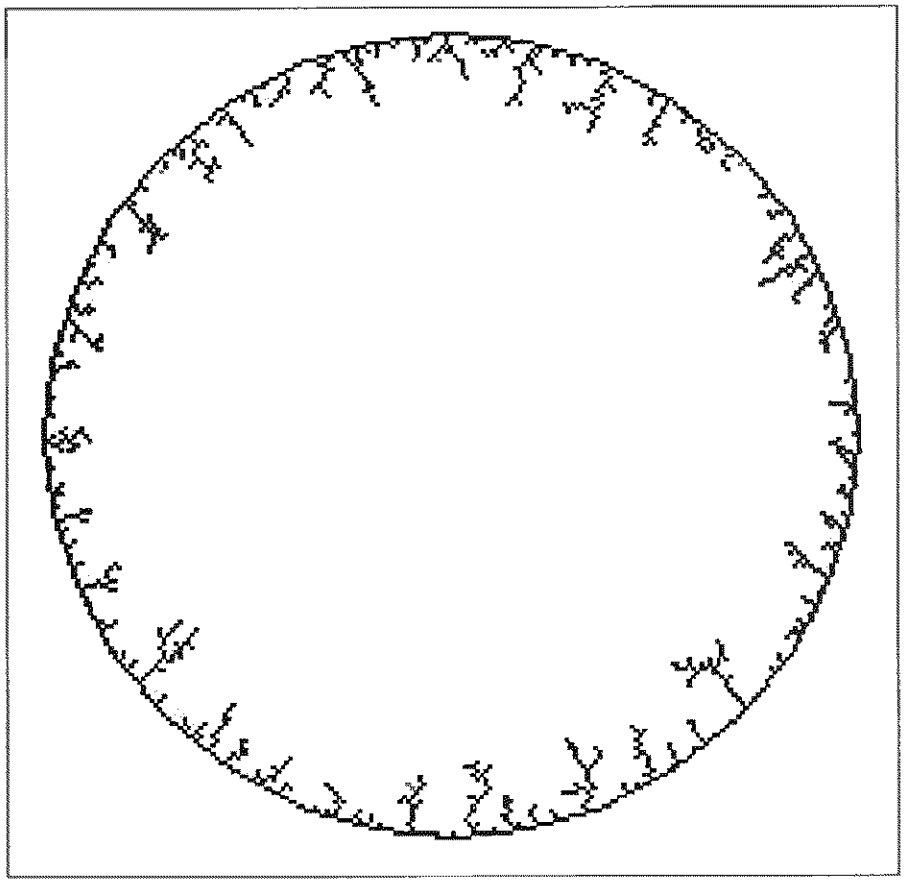


Fig. 11 : 711 particules.

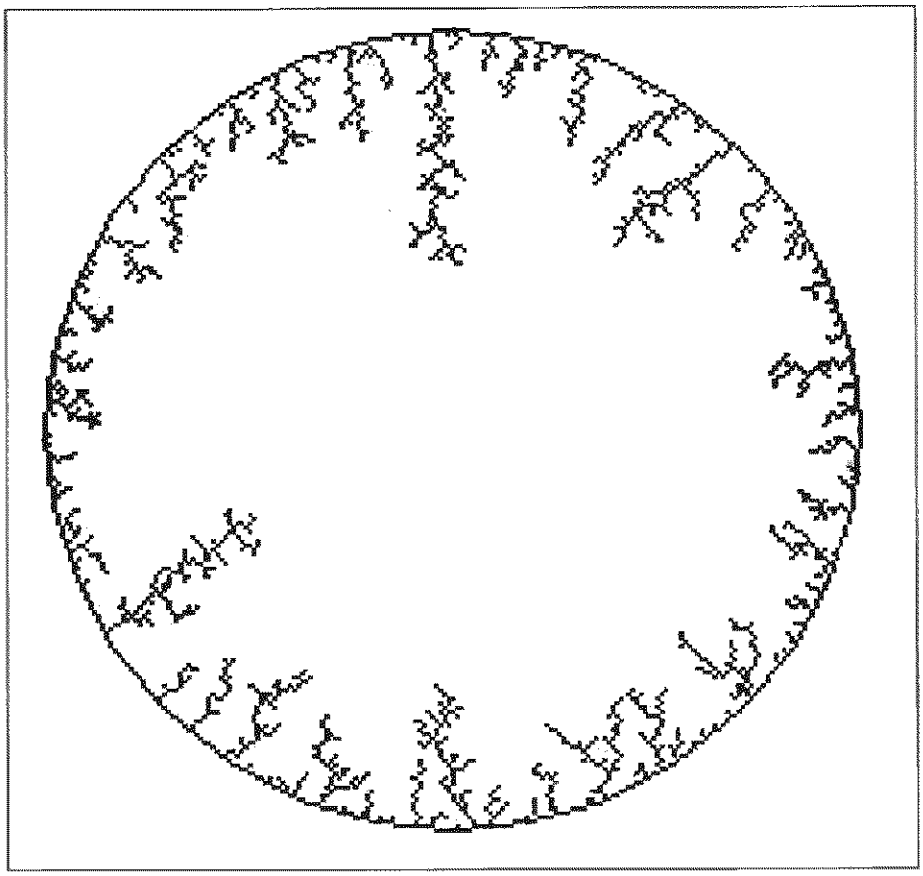


Fig. 12 : 1412 particules.

CARACTÉRISATION D'UNE FRACTALE : LA DIMENSION

Quelques rappels sur les dimensions connues

- La dimension repère

Tout le monde a entendu parler de la dimension, ne serait-ce par le terme 3-D couramment rencontré en matière de logiciel.

Les dimensions connues sont caractérisées par un nombre entier, et très petit, par rapport à tous les nombres existants.

Ainsi, le point est considéré comme faisant partie de la dimension zéro.

Une ligne correspond quant à elle à la dimension un, car, pour repérer la position d'un point sur celle-ci, il suffit d'une seule coordonnée.

Un plan, comme celui d'une table, est considéré de dimension deux, chaque point lui appartenant étant repéré par deux coordonnées, à l'image de l'écran de votre ordinateur.



Dès que l'on arrive dans l'espace à l'intérieur duquel on se déplace, on passe à la troisième dimension, qui repère un point à l'aide de trois coordonnées.

Il est parfaitement possible de créer des espaces avec des dimensions supérieures, et c'est l'un des passe-temps favoris des mathématiciens et physiciens.

Signalons de suite la quatrième dimension, qui, à part son utilisation souvent farfelue dans les films ou romans de science fiction, est utilisée par les chercheurs en informatique, pour notamment la disposition de processeurs de calcul parallèle, en particulier grâce à la notion d'hypercube.

- Les dimensions des objets mathématiques courants

La dimension dont nous venons de parler était celle qui nous permettait de situer un point dans un espace à n dimensions. Mais qu'en est-il des objets qui appartiennent à cet espace ? Pourquoi considère-t-on qu'un carré est de dimension deux, qu'un cube est de dimension trois, qu'un disque est de dimension deux, etc.?

Prenons tout de suite pour référence la figure 13, afin de rechercher la dimension du carré.

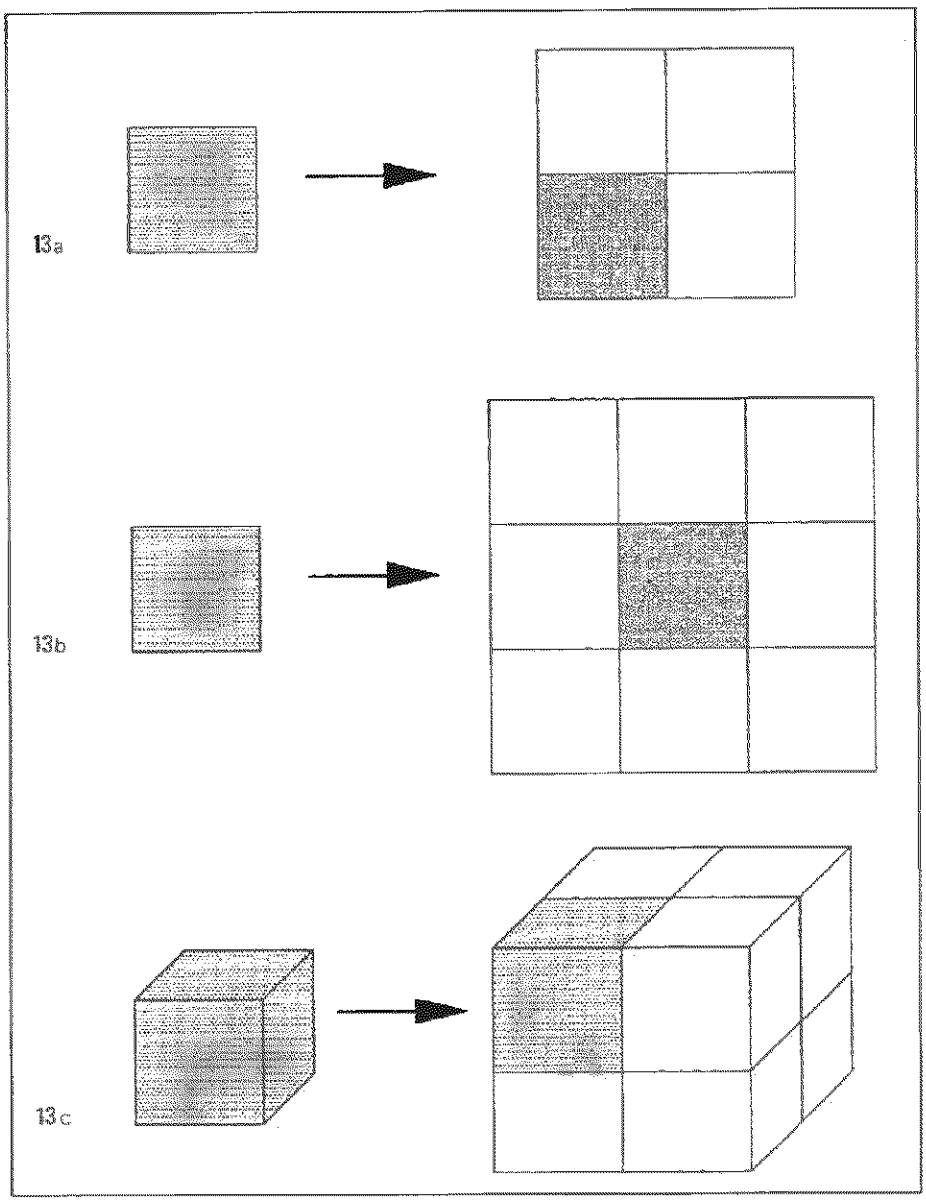


Fig.13 : Dimension du carré.

En figure 13a, nous utilisons un carré pour reproduire un nouveau carré, dont les côtés sont deux fois plus grands, à l'aide de trois carrés supplémentaires ; donc quatre carrés nous permettent de réaliser un carré deux fois plus grand.

Considérons donc, pour l'instant la formule :

$4 = 2^2$ (deux puissance deux)



En 13b, le même carré est utilisé pour réaliser un carré de côtés trois fois plus grand, cette fois-ci à l'aide de neuf carrés, ce qui nous permet de considérer la formule :

$9 = 3^2$ (trois puissance deux)

Il en serait de même pour réaliser un carré quatre fois plus grand, pour lequel il nous faudrait $4^2 = 16$ carrés, et ainsi de suite.

Remarquons que dans ces formules, la puissance utilisée (deux) correspond à la dimension.

Considérons maintenant la figure 13c, qui utilise cette fois-ci un cube.

Pour réaliser un cube dont les arêtes sont deux fois plus grandes, il nous faut huit cubes, or nous pouvons écrire :

$8 = 2^3$ (deux puissance trois)

De même en agrandissant le cube pour obtenir des arêtes trois fois plus grandes, il nous faudrait : $3^3 = 27$ cubes. Il en serait de même selon cette loi en agrandissant comme on le désire le cube initial à 4, 5, 6, ... n fois la longueur des arêtes initiales.

La puissance, cette fois-ci trois, correspond à la dimension du cube.

Nous pouvons déduire la généralité de cette formule qui est :

$$N = A^d$$

avec :

— N, le nombre de morceaux nécesssaires ;

— A, le rapport d'agrandissement, encore appelé rapport d'homotétie ;

— d, la dimension de l'objet.

Prenons cette fois-ci pour exemple le disque.

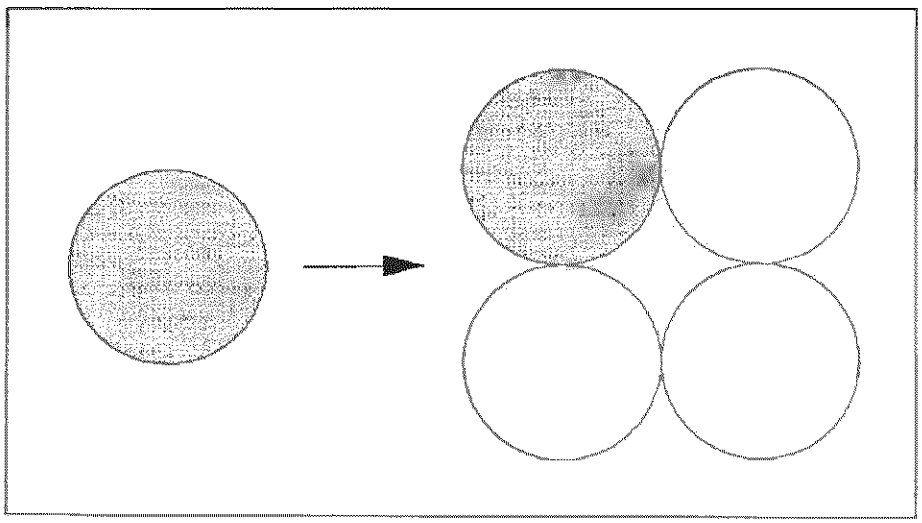


Fig. 14 : Dimension du disque.

Comme nous le montre la figure 14, il est impossible à l'aide du disque de réaliser un nouveau disque plus grand en utilisant plusieurs disques entiers.

Par contre, la figure 15 nous montre qu'un disque contient un carré, donc que sa dimension est supérieure ou égale à deux, dimension du carré.

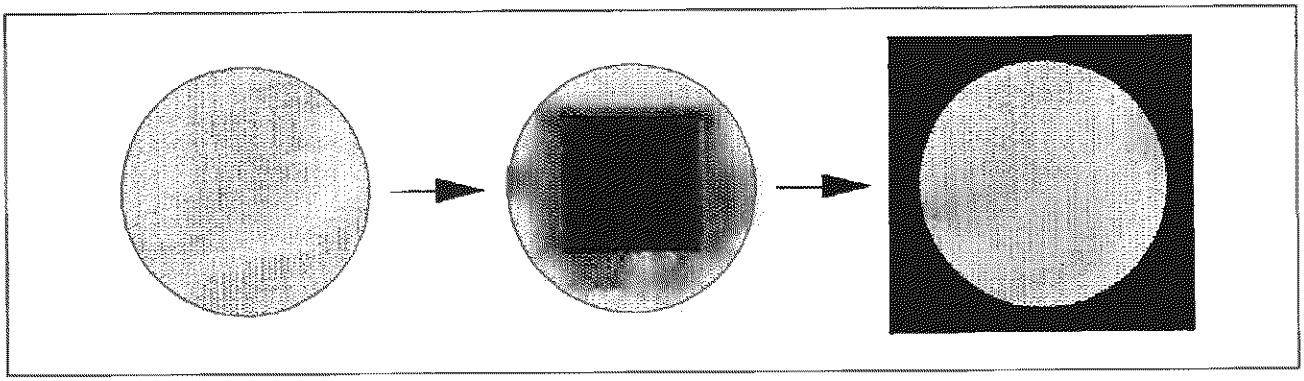


Fig. 15 : Dimension d'un disque contenant un carré.

Cette figure nous montre que ce même disque peut aussi être contenu dans un carré, donc que sa dimension est inférieure ou égale à deux.

Ainsi, si la dimension du disque est à la fois inférieure ou égale et supérieure ou égale à deux, celle-ci vaut exactement deux.

La dimension fractale des agrégats

Les objets fractals réalisés par agrégation ne sont pas des objets mathématiques courants, et il est impossible d'appliquer rigoureusement la formule développée ci-dessus pour déterminer leur dimension.

Si nous reprenons l'objet créé en figure 7 ou 8, nous pouvons donner un encadrement de sa dimension, mais pas plus.

Sachant que notre objet est composé de particules élémentaires, que nous assimilerons au point, celui-ci contient un objet de dimension zéro, donc il est de dimension supérieure ou égale à zéro; par contre, un point ne le contient pas, donc il est de dimension strictement supérieure à zéro.

Il est de plus contenu dans le plan, de dimension deux, donc de dimension inférieure ou égale à deux, mais comme il ne peut pas contenir de carré, de par sa forme, il est de dimension strictement inférieure à deux.

En poussant encore le raisonnement, il est impossible de faire contenir cet objet dans une droite, donc sa dimension sera strictement supérieure à un.

Voilà dans l'état actuel de nos connaissances ce que l'on peut dire : «Notre objet fractal possède une dimension comprise entre 1 et 2.»

Bien que non entière, sachez que la dimension de notre objet est calculable, ce que nous ferons, est qu'elle n'est pas farfelue, les



mathématiciens ayant déterminé que la dimension des objets peut être réelle (au sens où elle appartient à l'ensemble des nombres réels, au même titre que PI = 3,1416..., 1/6 , racine de 2, racine de 5, etc.), et qu'elle est intéressante pour leurs études.

Comment calculer la dimension fractale d'un objet comme le nôtre ?

Nous allons pour cela étendre la formule précédente, en considérant sur le fractal un cercle de rayon R, et en apportant le fait que le nombre de particules originelles dans le disque ainsi créé est proportionnel (en gardant toujours le même centre) au rayon mis à la puissance de la dimension.

$$Nb = Cte \times R^{d}$$

avec :

- Nb, le nombre de particules;
- R, le rayon du cercle de mesure;
- d, la dimension du fractal.

Ainsi, connaissant le nombre de particules dans un cercle de rayon R, la dimension s'écrira :

$$\sqrt[R]{\frac{Nb}{Cte}}$$

«Racine R-ième du rapport Nb sur Cte»

Seule ombre au tableau, comment déterminer la constante, qui a de fortes chances de varier selon la forme du fractal.

Pour cela, nous allons essayer d'éliminer dans notre calcul le terme Cte, en effectuant la détermination du nombre de particules sur deux cercles de rayons différents (voir figure 16).

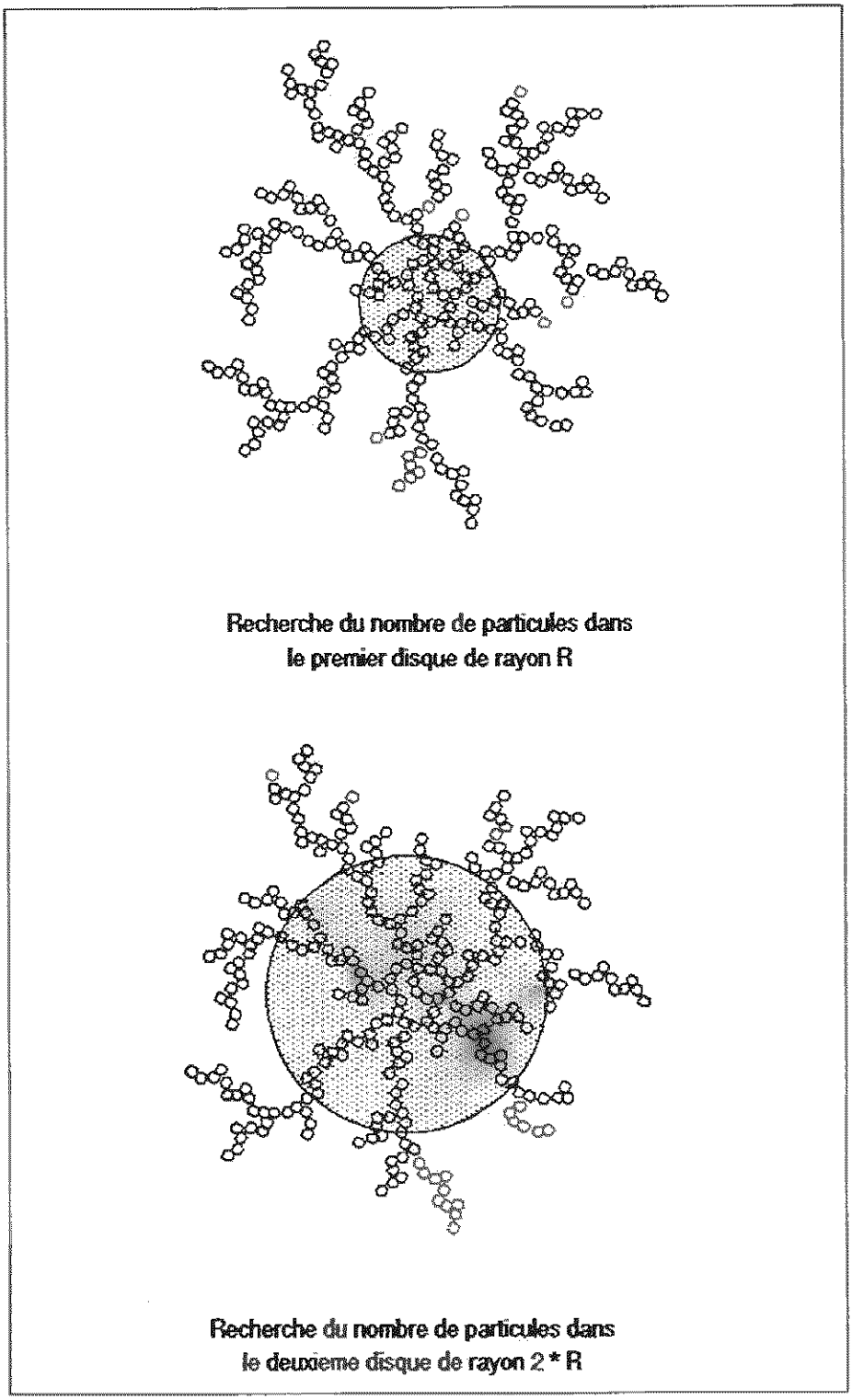
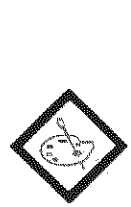


Fig. 16 : Calcul de la dimension d'un fractal de type agrégation par diffusion limitée.



Soient deux cercles de rayons R1 et R2, qui nous donnerons deux quantités respectives Nb1 et Nb2 de particules.

Nous obtiendrons le système suivant :

$$\begin{cases} Nb1 = Cte \times R1^d \\ Nb2 = Cte \times R2^d \end{cases}$$

En effectuant le rapport Nb1 sur Nb2, nous obtiendrons :

$$\frac{Nb1}{Nb2} = \frac{Cte \times R1^d}{Cte \times R2^d} = \frac{R1^d}{R2^d}$$

Passons à la fonction logarithme pour extraire la dimension, en effectuant le Log (népérien de préférence, mais ce n'est pas obligatoire) de chacun des termes de l'égalité :

$$\begin{aligned} Log\left(\frac{Nb1}{Nb2}\right) &= Log\left(\frac{R1^d}{R2^d}\right) \\ &= Log(R1^d) - Log(R2^d) \\ &= d \times Log(R1) - d \times Log(R2) \\ &= d(Log(R1) - Log(R2)) \\ &= d \times Log\left(\frac{R1}{R2}\right) \end{aligned}$$

Ce qui nous donne pour la dimension:

$$d = \frac{Log\left(\frac{Nb1}{Nb2}\right)}{Log\left(\frac{R1}{R2}\right)}$$

Si l'on prend pour cas particulier un cercle R1 de rayon double de celui de rayon R2 (R1 = 2 x R2), on obtient :

$$\begin{aligned} d &= \frac{Log\left(\frac{Nb1}{Nb2}\right)}{Log\left(\frac{2xR2}{R2}\right)} \\ &= \frac{Log\left(\frac{Nb1}{Nb2}\right)}{Log(2)} \end{aligned}$$

Cette formule, valable pour les fractals, est aussi valable pour tous les objets mathématiques dont nous avons dialogué dans le précédent chapitre, *a fortiori* pour une surface plane tel un disque rempli uniformément de particules. Vérifions.

Sur un disque, lorsqu'on double le rayon, la surface croît au carré du rayon, nous obtiendrons donc quatre fois plus de particules :

Nb1 = 4 x Nb2

Ainsi :

$$d = \frac{\mathrm{Log}(\frac{4 \times Nb2}{Nb2})}{\mathrm{Log}(2)}$$

$$= \frac{\mathrm{Log}(4)}{\mathrm{Log}(2)}$$

$$= \frac{\mathrm{Log}(2^2)}{\mathrm{Log}(2)}$$

$$= 2\frac{\mathrm{Log}(2)}{\mathrm{Log}(2)}$$

$$= 2$$

qui est bien la dimension du disque.

Programme de calcul de dimension d'un fractal

Nous vous proposons ici, à l'aide de la méthode décrite ci-dessus, de calculer la dimension des fracals que vous pourrez créer à l'aide du programme d'agrégation centrale.

Pas question, quand on possède un micro-ordinateur de compter le nombre de particules, situées dans une zone circulaire, «à la main». Le travail serait bien trop fastidieux.

Le calcul s'effectuera donc à l'aide de votre ST ou STE.

- Le principe de comptage

Nous avons ici préféré la simplicité de l'algorithme de calcul au détriment de la rapidité d'exécution. Mais, après les quelques heures de calcul passées à la détermination du fractal, que seront les quelques dizaines de secondes passées au comptage !

Il nous faut donc, à partir du point central d'agrégation, compter le nombre de particules dans un cercle centré sur celui-ci, et recommencer le calcul pour un cercle de rayon double.



Pour simplifier l'algorithme, nous avons décidé d'effectuer la recherche sur un carré de côté égal au diamètre du cercle, et d'éliminer tous les points situés, par rapport au centre, à une distance supérieure au rayon.

Comment calculer la distance par rapport au centre ?

Si l'on considère un repère, appelé orthonormé, sur l'écran de l'ordinateur, avec comme unité le pixel, la distance entre deux points est déterminée, en fonction des coordonnées horizontales X et verticales Y, par la formule :

$$\text{distance} = \sqrt{(X - X_R)^2 + (Y - Y_R)^2}$$

avec :

— X , Y les coordonnées du point;

— X_R , Y_R les coordonnées du centre du cercle.

Les point n'appartenant pas au cercle vérifieront l'inégalité suivante :

$$\sqrt{(X - X_R)^2 + (Y - Y_R)^2} > \text{Rayon}$$

que l'on peut encore écrire :

$$(X - X_R)^2 + (Y - Y_R)^2 > \text{Rayon}^2$$

Pour gagner un peu de rapidité de traitement, nous allons encore modifier cette inéquation par :

$$(X - X_R) \times (X - X_R) + (Y - Y_R) \times (Y - Y_R) > \text{Rayon} \times \text{Rayon}$$

L'algorithme devient ainsi :

- DEBUT
  - POUR calculX DE debutducarré A finducarré
    - POUR calculY DE debutducarré A finducarré
      - Calculer le carré de la distance par rapport au centre
      - SI cette distance est inférieure au carré du rayon
      - ALORS
        - SI il y a une particule
        - ALORS
          - Compter cette particule
        - FINSI
      - FINSI
    - FINPOUR
  - FINPOUR
- FIN

● Le programme

```
REM ***********************************
REM ***                               ***
REM ***   CALCUL DE LA DIMENSION      ***
REM *** D'UNE AGREGATION FRACTALE     ***
REM ***                               ***
REM ***********************************
REM
REM ---------------------------------------
REM ***    Calcul de la dimension     ***
REM ***   d'un amas de particules     ***
REM ***  generees par un mouvement    ***
REM ***       pseudo-brownien         ***
REM ***       par la propriete        ***
REM ***  Nb = Cte * D ^ dimension     ***
REM ---------------------------------------
'
CLS
IF XBIOS(4)<>0
  ALERT 3,"Ce programme|fonctionne en|basse resolution",1," OK",variable%
  EDIT
ENDIF
'
REM *** INITIALISATIONS ***
'
' palette de couleur
'
VSETCOLOR 0,0,0,0
VSETCOLOR 1,15,15,15
VSETCOLOR 2,15,0,0
VSETCOLOR 3,0,15,0
VSETCOLOR 4,0,0,15
VSETCOLOR 5,15,15,0
VSETCOLOR 6,11,5,0
VSETCOLOR 7,0,11,0
VSETCOLOR 8,11,0,11
VSETCOLOR 9,0,15,15
'
' tableau de points
'
DIM coordonneex%(5000)
DIM coordonneey%(5000)
DIM couleur_point%(5000)
'
REM * nombre de particules *
'
couleur%=1
nombre_de_particules%=1
```



(suite)

```
'
REM *** CHARGEMENT D'UN FICHIER ***
'
CLS
'
' ouverture fenetre de dialogue
'
OPENW #1,30,20,235,100,&H0
TITLEW #1,"CHARGEMENT"
'
' selection du ficher
'
FILESELECT "\*.frc","",nom_fichier$
'
IF INSTR(nom_fichier$,".")=0
  nom_fichier$=nom_fichier$+".FRC"
ENDIF
'
' affichage de l'etat de chargement
'
CLS
PRINT "Chargement de : "
PRINT "      "
PRINT nom_fichier$
'
nombre_de_particules%=1
LOCATE 1,6
PRINT "Particules ";
PRINT " X     Y     Couleur"
PRINT
OPEN "i",#5,nom_fichier$
'
' chargement effectif
'
WHILE NOT EOF(#5)
  INPUT
#5,coordonneex%(nombre_de_particules%),coordonneey%(nombre_de_particules%),couleur
_point%(nombre_de_particules%)
  LOCATE 1,8
  PRINT "    ";nombre_de_particules%;"      ";
  PRINT coordonneex%(nombre_de_particules%);"   ";
  PRINT coordonneey%(nombre_de_particules%);"      ";
  PRINT couleur_point%(nombre_de_particules%);"     "
  nombre_de_particules%=nombre_de_particules%+1
WEND
'
CLOSE #5
'
```

(suite)

```
CLOSEW #1
CLOSEW #0
'
CLS
'
REM * point initial *
'
position_initialex%=160
position_initialey%=100
'
couleur%=1
COLOR couleur%
PLOT position_initialex%,position_initialey%
'
' Initialisation du premier element du tableau
'
coordonneex%(1)=position_initialex%
coordonneey%(1)=position_initialey%
couleur_point%(1)=couleur%
'
REM *   AFFICHAGE DU TABLEAU   *
'
FOR compteur%=1 TO nombre_de_particules%
  COLOR couleur_point%(compteur%)
  PLOT coordonneex%(compteur%),coordonneey%(compteur%)
NEXT compteur%
'
REM ************************
REM *** BOUCLE DE CALCUL ***
REM ************************
'
REPEAT
  '
  WHILE INKEY$<>""       ! vide
  WEND                   ! buffer clavier
  '
  REM *** CALCUL NOMBRE PARTICULES ***
  REM ***   SUR LE PREMIER CERCLE  ***
  '
  LOCATE 1,1
  PRINT SPC(38)
  LOCATE 1,1
  INPUT "rayon initial (5-40)";rayon_initial%
  LOCATE 1,1
  PRINT SPC(38)
  '
  rayon_petit_cercle%=rayon_initial%
  '
```



(suite)

```
    LOCATE 1,1
    PRINT SPC(38)
    LOCATE 1,1
    PRINT "rayon de comptage : ";
    PRINT rayon_petit_cercle%
    '
    negative_rayon_petit_cercle%=-rayon_petit_cercle%
    distance_maxi%=rayon_petit_cercle%*rayon_petit_cercle%
    nombre_particule_petit_cercle%=0
    '
    FOR compteur1%=negative_rayon_petit_cercle% TO rayon_petit_cercle%
      FOR compteur2%=negative_rayon_petit_cercle% TO rayon_petit_cercle%
        positionx%=160+compteur1%
        positiony%=100+compteur2%
        distance%=(positionx%-160)*(positionx%-160)+(positiony%-100)*(positiony%-
100)
        IF distance%<distance_maxi%
          a%=PTST(positionx%,positiony%)
          PSET positionx%,positiony%,0
          IF a%<>0
            nombre_particule_petit_cercle%=nombre_particule_petit_cercle%+1
            PSET positionx%,positiony%,1
          ENDIF
        ENDIF
      NEXT compteur2%
    NEXT compteur1%
    '
    REM *** CALCUL NOMBRE PARTICULES ***
    REM ***  SUR LE DEUXIEME CERCLE  ***
    '
    rayon_grand_cercle%=2*rayon_petit_cercle%
    '
    LOCATE 1,1
    PRINT SPC(38)
    LOCATE 1,1
    PRINT "rayon de comptage : ";
    PRINT rayon_grand_cercle%
    '
    negative_rayon_grand_cercle%=-rayon_grand_cercle%
    distance2_maxi%=rayon_grand_cercle%*rayon_grand_cercle%
    nombre_particule_grand_cercle%=0
    '
    FOR compteur1%=negative_rayon_grand_cercle% TO rayon_grand_cercle%
      FOR compteur2%=negative_rayon_grand_cercle% TO rayon_grand_cercle%
        positionx%=160+compteur1%
        positiony%=100+compteur2%
        distance%=(positionx%-160)*(positionx%-160)+(positiony%-100)*(positiony%-
100)
```

(suite)

```
      IF distance%<distance2_maxi%
        a%=PTST(positionx%,positiony%)
        PSET positionx%,positiony%,0
        IF a%<>0
          nombre_particule_grand_cercle%=nombre_particule_grand_cercle%+1
          PSET positionx%,positiony%,4
        ENDIF
      ENDIF
    NEXT compteur2%
  NEXT compteur1%
  '
  REM *** CALCUL DE LA DIMENSION ***
  REM ***       DE LA FRACTALE     ***
  '
  LOCATE 1,1
  PRINT SPC(38)
  LOCATE 1,1
  PRINT "Nb1:";
  PRINT nombre_particule_petit_cercle%;
  PRINT "  Nb2:";
  PRINT nombre_particule_grand_cercle%;
  '
  rapport=nombre_particule_grand_cercle%/nombre_particule_petit_cercle%
  dimension=LOG(rapport)/LOG(2)
  '
  PRINT " Dimension:";
  PRINT USING "#.##",dimension
  '
  REPEAT
    flag%=MOUSEK
  UNTIL flag%<>0
  '
UNTIL flag%=2
'
EDIT
'
REM *** FIN PROGRAMME ***
```

Après lancement, le programme ouvre une fenêtre proposant le chargement d'un fractal préalablement calculé, puis demande le rayon du cercle de diamètre le plus petit.

Ce rayon devra être, normalement, inférieur à 44, car les particules sont engendrées depuis un cercle de rayon égal à 90, valeur proche du double.

Il ne faut pas non plus introduire un rayon trop petit, l'erreur de calcul qui s'en suivrait ne permettrait pas une bonne approximation de la dimension.



Nous vous conseillons de choisir un rayon initial compris entre 10 et 20, et d'avoir effectué le calcul sur la génération du fractal le plus longtemps possible (approcher les 5 000 particules est préférable, mais on obtient des résultats acceptables à partir de 1 000 particules).

Rappelez-vous, la dimension de ce type de fractal doit être comprise entre un et deux, sinon, votre cercle de calcul est trop petit (dimension supérieure à deux), ou alors trop grand pour le nombre de particules fixées (dimension inférieure à un).

Sachez aussi que la dimension du dépôt de zinc par electrolyse est de l'ordre de 1,7; mais, son calcul est effectué sur plusieurs milliards de milliards d'atomes.

Enfin, si votre fractal est correctement créé, avec beaucoup de points, vous pourrez remarquer que sa dimension est invariante, quelle que soit sa taille, bien que la densité de particules diminue, comme tous les fractals d'ailleurs.

Bibliographie sommaire :

— Jullien, *Les phénomènes d'agrégation et les agrégats fractals* - «Les annales des télécommunication», tome 41 n° 7 et 8, juillet/août 1986.

— Mandelbrot, *La géométrie fractale dans la nature* - Freeman and Co.

— Perrin, *Les atomes* - Ed. Gallimard.

— Witten et Sander, *Agrégation par diffusion limitée, phénomène de transformation cinétique* - «Physical Review Letters», n°19, novembre 1981, et n°9, mai 1983.

— Stewart, *Les fractals* - «Les chroniques de Rose Polymath» (les fractals en bande dessinée) , Ed. Belin.

— *L'ordre du Chaos* - «Bibliothèque pour la Science», Ed. Belin.



5/6

Traitement d'image

Ce chapitre est le siège des magiciens de l'image. Ici point d'artistes, mais des techniciens seulement.

Les images sur ordinateur se réduisent de toute façon, tôt ou tard, à un fichier où sont rangées savamment toutes les informations nécessaires à leur composition.

Conversion de fichiers en divers formats ou acquisition d'images par les moyens les plus divers, toutes ces techniques sont traitées dans ce chapitre.

5/6.1

Le digitaliseur d'image VIDI-ST

Développée également sur Amiga et compatible PC, cette cartouche de la société ROMBO éditée par Human Technologies est à l'heure actuelle le moins coûteux des systèmes de digitalisation en temps réel pour micro-ordinateur, toutes marques confondues.

SCANNER CONTRE DIGITALISEUR

Un système de digitalisation permet de récupérer sur un écran d'ordinateur, puis sur une disquette, une image provenant d'une source vidéo telle qu'une caméra ou un camescope. L'intérêt d'un tel dispositif par rapport à un scanner vient du fait que des objets en trois dimensions peuvent être ainsi numérisés (alors qu'avec le scanner, il faudrait d'abord en faire une photographie). Mais c'est surtout le fait de pouvoir récupérer des images venant d'un magnétoscope qui est particulièrement attractif, car on bénéficie par l'intermédiaire des différentes chaînes de télévision, vidéo-cassettes ou laser-vidéo d'un réservoir d'images inépuisable : personnages de dessins animés dans différentes attitudes, personnalités célèbres, faits de l'actualité... Images que l'on peut ensuite retravailler pour les imprimer ou les intégrer à un logiciel de sa conception (formats DEGAS, NEOCHROME et GEM pour VIDI-ST, plus RGH et SPECTRUM pour MIXIMAGE).

Quant à la possibilité d'enregistrer en temps réel une succession d'images, elle supprime l'obligation de disposer d'un arrêt sur l'image parfait qui n'est présent que sur les magnétoscopes de haute qualité.

LE PACK VIDI-ST

La cartouche de Human Technologie est fournie avec trois disquettes contenant plusieurs logiciels et de nombreux fichiers exemples ainsi que diverses routines utiles aux programmeurs en GFA Basic. Tous les logiciels peuvent être utilisés sans la cartouche, les options de digitalisation étant bien sûr inopérantes, mais on peut relire et retravailler des séquences préalablement stockées sur disquette.

VIDI-ST

Mémorise une séquence animée de quelques secondes, pas exactement en temps réel (12,5 images/seconde) et en monochrome (ou fausses couleurs), contrairement à ce que laisse croire la publicité, même si une couleur différente peut être attribuée par la suite à chaque niveau de gris.

VIDIMONO

Fonctionne avec un écran haute résolution monochrome.

MIXIMAGE

Digitalise un sujet immobile avec ses vraies couleurs par l'intermédiaire de trois prises de vue au travers de filtres colorés ou de filtres électroniques.

SPSLIDES8

Enchaîne en continu à l'écran des images au format SPECTRUM.

SLIDE

Idem.

ZZ-SCHOW

Idem pour les images 16 couleurs au format RGH avec de nombreux effets de scrolling, volets, fondus, etc.

IMPRIME

Sort sur imprimante couleur les fichiers au format RGH. Les pilotes sont fournis pour :

— CANON PJ-1080 et FP 510

— EPSON LQ 2500

— NEC CP6/CP7

— STAR LC 24-10

— XEROX 4020

DES POSSIBILITES LIMITEES PAR LA CARTE GRAPHIQUE DE L'ATARI ST

VIDI-ST n'est pas destiné à remplacer un magnétoscope, car quelques secondes de digitalisation ont tôt fait de remplir la mémoire de l'Atari qui a intérêt à être la plus étendue possible, sans compter que la sauvegarde doit malheureusement s'effectuer image par image sur un nombre conséquent de disquettes. Comme avec un appareil photo motorisé qui permet d'effectuer des prises de vue en rafale, le fait d'enregistrer automatiquement une série d'images en mémoire permet surtout de sélectionner la meilleure par la suite.

La faible échelle des gris de l'Atari (8 pouvant être étendus à 16 par l'intermédiaire des couleurs) nécessite des réglages acrobatiques pour obtenir une image à peu près présentable ; il est, dans ces conditions, difficile de lutter contre les 256 niveaux de gris des cartes 24 bits sur PC et de certains appareils photos numériques qui, avec davantage de nuances, offrent une qualité nettement supérieure sous une résolution pourtant proche de celle de l'Atari (376 x 256 points ; valeur à comparer tout de même aux 20 millions de points d'une photo argentique). Il est vrai que sur l'Atari, l'image est animée, elle, et les écrans fixes en couleur obtenus par MIXIMAGE sont d'une qualité nettement supérieure.

Enfin, le transfert est à sens unique : s'il est possible d'introduire une image vidéo dans l'ordinateur, il n'est pas possible que la séquence retravaillée sur la machine puisse être retournée vers un téléviseur ou un magnétoscope sans acquisition d'un encodeur et d'un logiciel ad-hoc.



QUELQUES EXEMPLES

Malgré une perte de qualité certaine due au tirage sur imprimante, les exemples qui suivent, tirés du film «Qui veut la peau de Roger Rabbit ?», donnent une idée des résultats qu'il est possible d'obtenir avec VIDI-ST. Un rendu correct des visages est assez difficile à obtenir : la scène doit être bien contrastée, et il est préférable de choisir un gros plan ou un plan moyen pour que les détails ne se perdent pas dans la grisaille.

Dans la scène de rue constituée d'un foisonnement d'éléments, on remarque à nouveau que le handicap vient davantage du manque de nuances que du faible nombre de points qui constituent l'image.



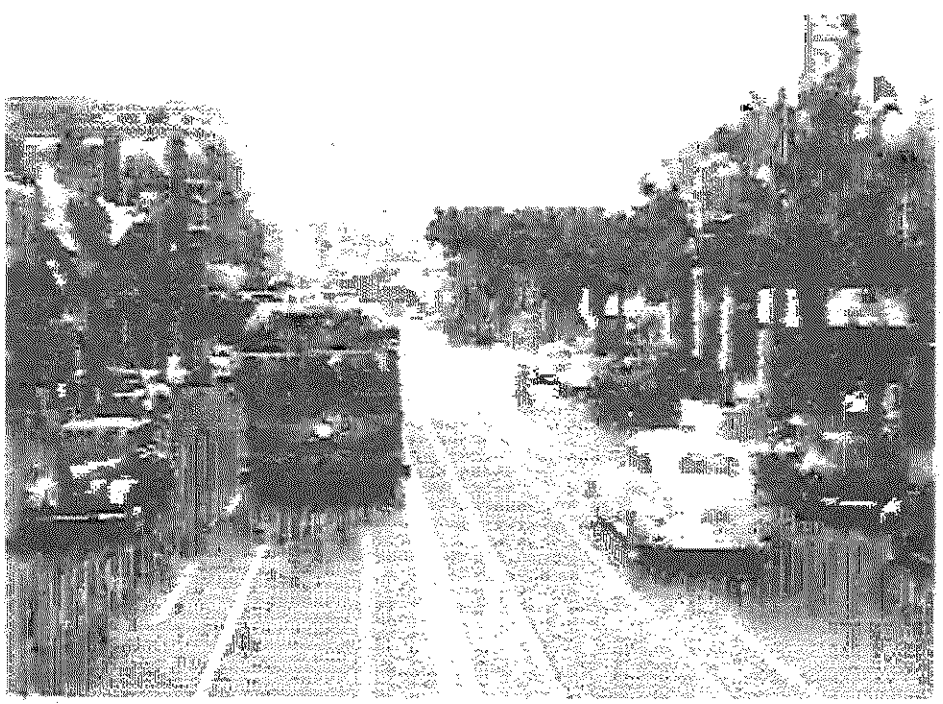
Colère : «il ne faut pas me chercher !»



Inquiète : «Pas de panique, baby !»



Fripon : «Et ça, tu aimes ?»



Toons : «Dans les rues de Toonville.»

UN OUTIL D'AIDE A LA CREATION

Cet ensemble constitue cependant une aide appréciable à un concepteur de logiciels ou à toute personne qui, sans être un graphiste hors pair, aimerait bien se frotter à l'image.Les info-graphistes, eux aussi, utilisent parfois scanneurs et digitaliseurs pour reprendre des esquisses plus faciles à réaliser sur papier et gagner du temps lors du travail sur écran. Il est d'ailleurs intéressant de noter que contrairement aux craintes formulées il y a quelques années, la prolifération des systèmes de digitalisation ainsi que l'accroissement de leurs performances n'ont pas mis cette profession au chômage : on trouve très peu de logiciels de jeux utilisant des images digitalisées, à la fois pour des raisons techniques (la place nécessaire sur le disque, ce qui sera moins problématique avec la généralisation des CD-ROM) et sans doute aussi pour les mêmes raisons qui ont fait le succès de la bande dessinée par rapport au roman photo pourtant moins coûteux à réaliser : rien ne peut remplacer le «coup de patte» de l'artiste pour accentuer la force ou le courage, la douceur ou l'ingénuité, la cruauté ou la perversité de personnages qui, dans tout scénario de jeu, se doivent d'être manichéens à souhait.

CONNEXIONS

La cartouche de digitalisation doit être connectée sur le port de sortie de l'Atari et, par l'intermédiaire d'une prise CINCH, sur la sortie vidéo d'une caméra, d'un camescope ou d'un magnétoscope. Le câble de liaison n'est pas fourni et un adaptateur, généralement du type CINCH - BNC, s'avère nécessaire côté caméra.

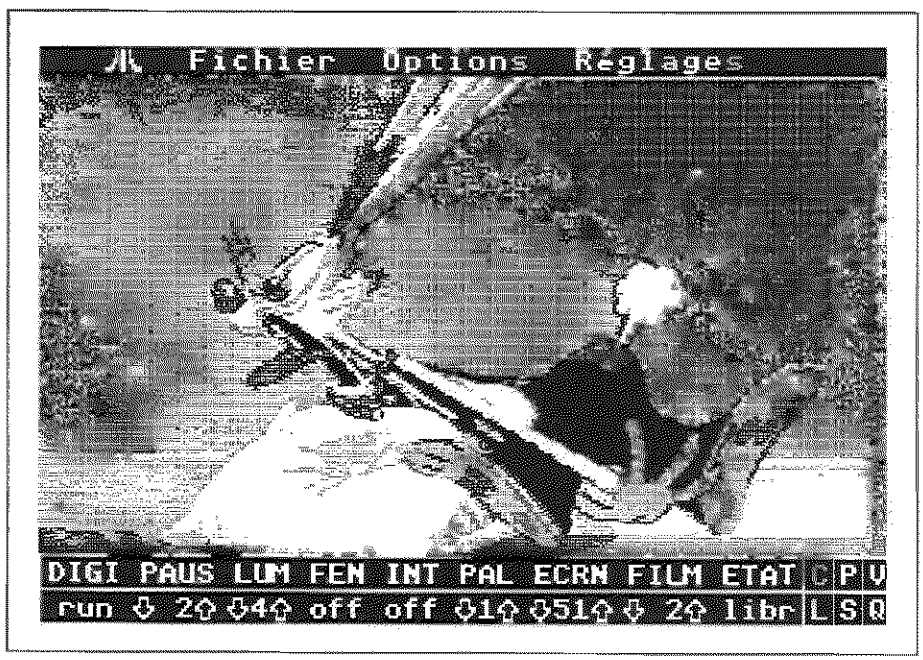
I - Le logiciel VIDI-ST





L'ECRAN D'ACCUEIL

VIDI-ST fonctionne uniquement en basse résolution. La fenêtre de visualisation de l'image est surmontée de la ligne des menus déroulants habituelle dans les programmes GEM. Les options les plus courantes sont rappelées dans une boîte à outils de deux lignes en bas d'écran ce qui en simplifie l'accès.



L'écran de travail.

ENREGISTRER UNE SEQUENCE

Le logiciel n'offre pas d'option de prévisualisation qui permettrait de préparer les réglages ; on ajuste donc la luminosité et le contraste pendant l'enregistrement en mémoire, ce qui ne pose pas de problème puisque celui-ci s'effectue cycliquement et en continu : les images imparfaites obtenues au début des réglages sont écrasées par de nouvelles images dès que la mémoire de l'ordinateur est pleine.

Le menu OPTION/DIGITALISER ou le bouton DIGI de la boîte à outils lance la digitalisation à condition qu'un signal vidéo soit présent à l'entrée de la cartouche : les lignes de menus disparaissent et les images animées apparaissent en plein écran. On en profite pour ajuster le cadrage, la mise au point et le diaphragme, si on travaille avec une caméra, ou seulement la luminosité et le contraste, si on utilise des images déjà stockées sur bande : ces deux derniers réglages s'effectuent grâce aux boutons de la cartouche qui autorisent une large plage de variation avec des rendus très différents ; la qualité du résultat final dépend essentiellement de cet ajustage qui en pratique, doit être modifié à chaque changement d'éclairage ou de sujet.

La digitalisation se poursuit jusqu'à ce qu'on appuie sur une touche de l'ordinateur. Le principe de l'enregistrement cyclique revient à travailler avec une boucle de bande qui repasse sans cesse devant la tête d'enregistrement et seules les dernières images sont effectivement mémorisées. Chaque image, non compressée pour une question de rapidité, occupe 32 000 octets (320 x 200 points multipliés par 4 bits pour coder les 16 couleurs divisées par 8 pour obtenir un résultat en octets : 320 x 200 x 4 / 8 = 32 000). En fonction de la configuration mémoire disponible après chargement du logiciel et des accessoires de bureau on parvient à stocker au mieux :

— 512 K : 9 images;

— 1024 K : 25 images;

— 2048 K : 57 images;

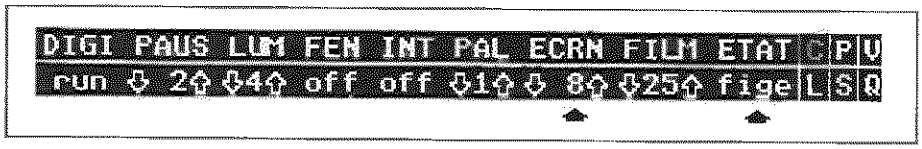
— 4096 K : 124 images.

VISUALISER LA SEQUENCE

Le menu OPTION/ANIMATION ou la touche FILM de la boîte à outils affiche la séquence mémorisée en plein écran. La bande se déroule sans fin de façon cyclique jusqu'à ce qu'un appui sur une touche du clavier gèle l'image courante et redonne les lignes de menus.

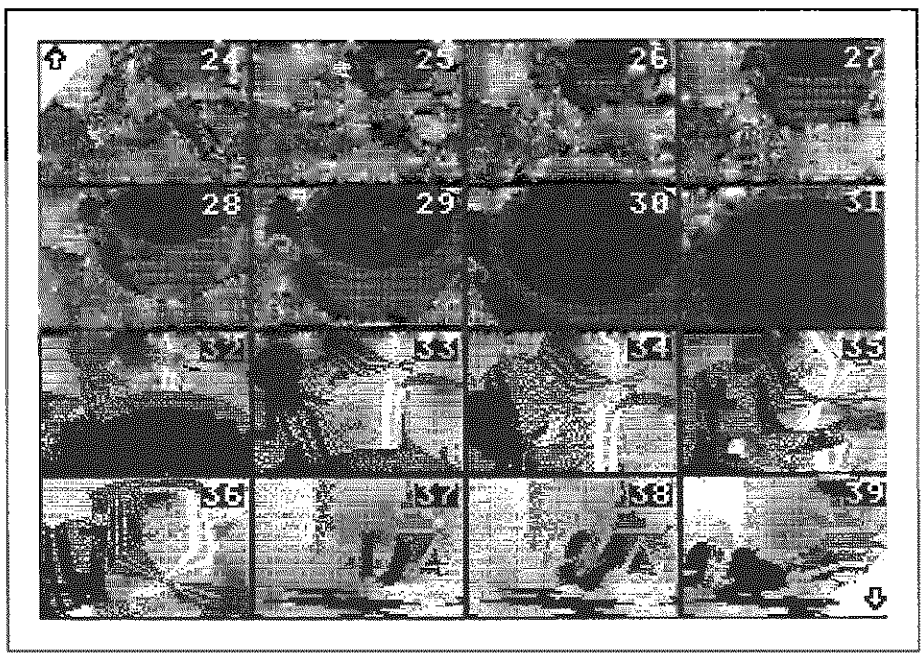
SELECTIONNER UNE VUE FIXE

VIDI-ST n'ayant pas pour vocation de remplacer un magnétoscope, rares sont les situations où on souhaite conserver une séquence animée complète, aussi est-il nécessaire de pouvoir ne sélectionner que la meilleure image de la séquence qu'on vient d'enregistrer.



Pour cela, on fait défiler les images les unes après les autres dans un sens ou dans l'autre en cliquant sur les flèches placées sous ECRN dans la boîte à outils : le numéro de l'image courante est affiché entre les deux flèches.

On peut aussi, par l'intermédiaire du menu OPTION/SELECTIONNER ou en cliquant sur le bouton ECRN de la boîte à outils, faire apparaître une planche de seize vues réduites. La flèche placée sur chacune des vues extrêmes de la planche permet de passer à la planche suivante ou de revenir à la précédente.



Qui va recevoir toutes les casseroles sur la tête ?

Un cadre de sélection suit la souris au fur et à mesure de son déplacement sur les vues réduites, et colorie l'écran en fonction de la palette affectée à chacune ; un clic fait apparaître la vue sélectionnée en plein écran.

Pour conserver la vue en mémoire afin de la retravailler ou de la sauvegarder ultérieurement, on clique sur le mot **LIBR** placé sous le bouton ETAT de la boîte à outils : l'état passe alors à **FIGE** indiquant que cette vue ne sera pas détruite lors des enregistrements suivants. Le fait qu'une ou que plusieurs vues soient figées implique évidemment que la place mémoire disponible pour les prochains enregistrements sera diminuée d'autant.

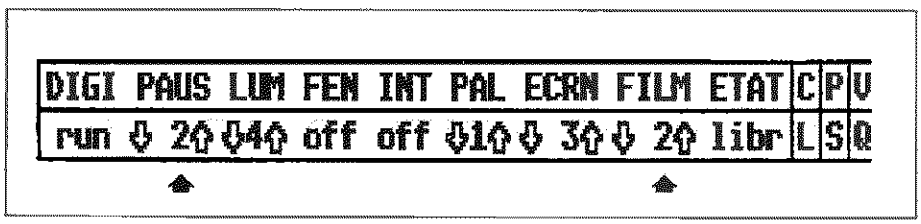
MODIFIER LA VITESSE D'ANIMATION

La vitesse d'animation, et par conséquent la durée d'un cycle, dépend de deux paramètres : la vitesse d'enregistrement et la vitesse de restitution.

Avant de lancer un enregistrement, on peut modifier la temporisation entre chaque prise de vue depuis les flèches situées sous le bouton **PAUS** de la boîte à outils. Si la digitalisation s'effectue bien à 25 images secondes afin de permettre une parfaite synchronisation avec les signaux vidéos en provenance des dispositifs audiovisuels actuels, le nombre d'images enregistrées effectivement en une seconde dans la mémoire de l'ordinateur dépend de la valeur du nombre placé entre les deux flèches de ce bouton : ainsi une valeur de 5 indique qu'on ne mémorise qu'une image sur 5, soit 25/5, c'est-à-dire 5 images à la seconde. Or la plus faible valeur que nous ayons réussi à obtenir (donc la plus grande vitesse) a

été 2, soit 25/2=12,5 images par seconde : VIDI-ST digitalise bien à 25 images secondes, mais ne mémorise qu'à 12,5 images par seconde, ce que la publicité du produit se garde bien de mentionner. L'animation reste toutefois assez souple et les saccades sont imperceptibles dans la plupart des situations ; elle est de toute façon bien meilleure que dans certains dessins animés japonais à 6 images secondes.

L'avantage d'un tel ralentissement forcé de la mémorisation se fait sentir au niveau de la durée du cycle qui atteint ainsi presque 10 secondes à la vitesse maximale au lieu de 5 avec un MEGA ST4.



La vitesse de restitution dépend de la valeur située sous le bouton FILM de la boîte à outils (à condition que le bouton ETAT soit sur LIBR). Une valeur supérieure à la vitesse d'enregistrement provoque un effet de ralenti ou de décomposition image par image d'une scène, alors qu'une valeur inférieure (on peut ici descendre à 1, soit 25 images/seconde) en accélère le déroulement.

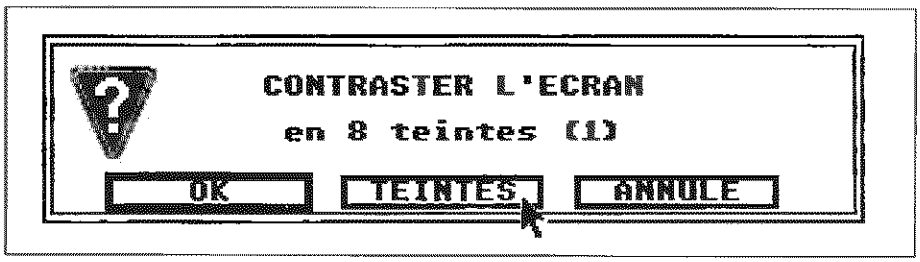
Modifier la palette de couleur

Après digitalisation, l'image apparaît avec 8 niveaux de gris ; il s'agit en réalité de deux fois huit niveaux, certaines zones de même intensité à l'écran étant en réalité codées différemment en mémoire. On parvient à différencier ces zones en modifiant la palette des couleurs grâce aux flèches placées sous le bouton PAL de la boîte à outils. On dispose, en plus de la palette standard, de 9 palettes prédéfinies, et on peut en charger d'autres depuis le menu FICHIER/CHARGER PALETTE : on sélectionne d'abord la palette à écraser depuis le bouton PAL, puis on charge la nouvelle palette, les autres palettes en mémoire ne sont bien sûr pas modifiées. Les palettes se trouvent dans le répertoire PALETTES du disque, et on y trouve notamment une palette qui inverse les gris et fait apparaître l'image en négatif.

Certaines palettes remplacent deux niveaux de gris équivalent par deux couleurs ce qui produit des effets parfois intéressants, d'autres conservent un niveau de gris et remplacent son équivalent par une nuance colorée, ce qui accroît la définition de l'image et lui donne une dominante de couleur.

On peut ainsi colorer indépendamment les unes des autres les images successives d'une même digitalisation.

Le menu OPTION/CONTRASTER offre la possibilité de réduire le nombre de teintes d'une image créant des effets photographiques de tirages durs.



En cliquant sur le bouton TEINTES, on fait défiler les différentes options :

— 8 teintes (x2) : c'est l'option de digitalisation, on ne différencie à l'écran les 16 valeurs que si on charge une palette comme nous l'avons vu ci-dessus;

— 8 teintes (1) : c'est ce qui est affiché à l'écran si on se contente de la palette grise standard;

— 4 teintes

— 2 teintes : uniquement noir et blanc, pratique pour détourer un personnage de dessin animé par exemple.

Attention, la perte d'informations sur une image est irréversible : il est par exemple impossible de revenir de 2 teintes à 8.

Le mode Images Fixes

Si on fige une image en faisant passer le bouton ETAT de LIBR à FIGE, celle-ci ne fait plus partie de l'animation, mais est placée dans le carrousel (par analogie au panier circulaire destiné à recevoir les diapositives d'un montage audio-visuel).

On constate que le nombre 2 placé sous le bouton FILM passe à 25 : il s'agit de la temporisation par défaut entre chaque vue fixe qui peut bien sûr être modifiée par les deux flèches qui encadrent le nombre.

On fait défiler les vues fixes en cliquant sur le bouton FILM alors que le bouton ETAT est sur FIGE ou en choisissant le menu OPTION/CARROUSEL ; on se retrouve alors en présence d'un diaporama cyclique.

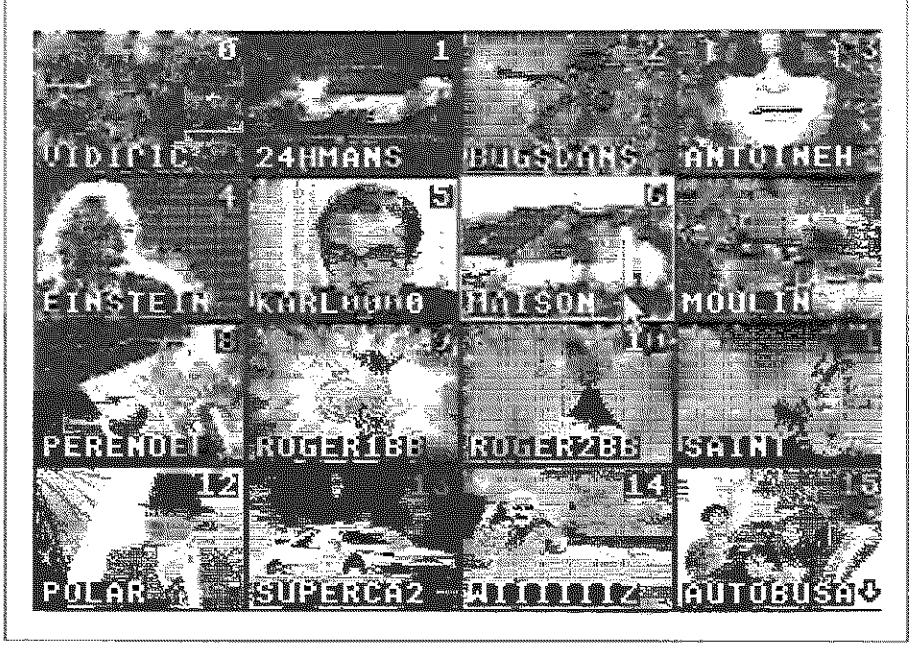
Pour résumer :

— si on active FILM alors que le bouton ETAT est LIBR, on assiste au déroulement de la séquence animée enregistrée;

— si on active FILM alors que ETAT est FIGE, on visionne les vues fixes.

Rappelons que les vues figées ne sont pas effacées lors d'une nouvelle digitalisation de séquence animée.

Le bouton ECRN fait apparaître la planche contact des vues, éventuellement complétées du nom sous lequel elles ont été sauvegardées sur disquette ce qui facilite leur repérage ; ce nom n'apparaît pas en mode plein écran.



Une planche contact.

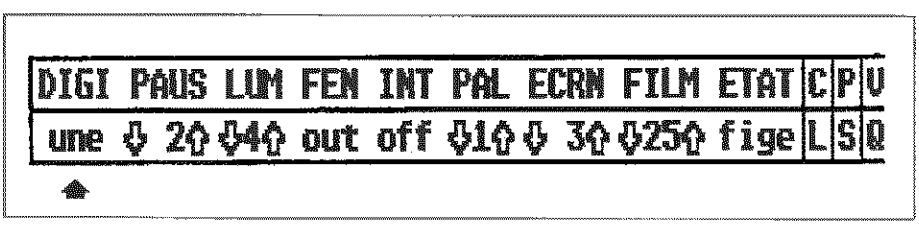
DIGITALISER UNE SEULE IMAGE

Le bouton DIGI offre trois options différentes qui sont affichées sous celui-ci :

RUN — Correspond à l'enregistrement d'une séquence animée.

UNE — Permet de ne digitaliser qu'une seule image sous le numéro indiqué par le bouton ECRN.

VUE — Reproduit sous ce numéro la dernière image digitalisée encore présente dans le tampon mémoire.



Dans le mode UNE, nous avons regretté que l'image disponible à l'entrée de la cartouche et venant d'un magnétoscope ne soit pas reproduite sur l'écran : elle n'apparaît que lorsque la digitalisation est faite, et on travaille

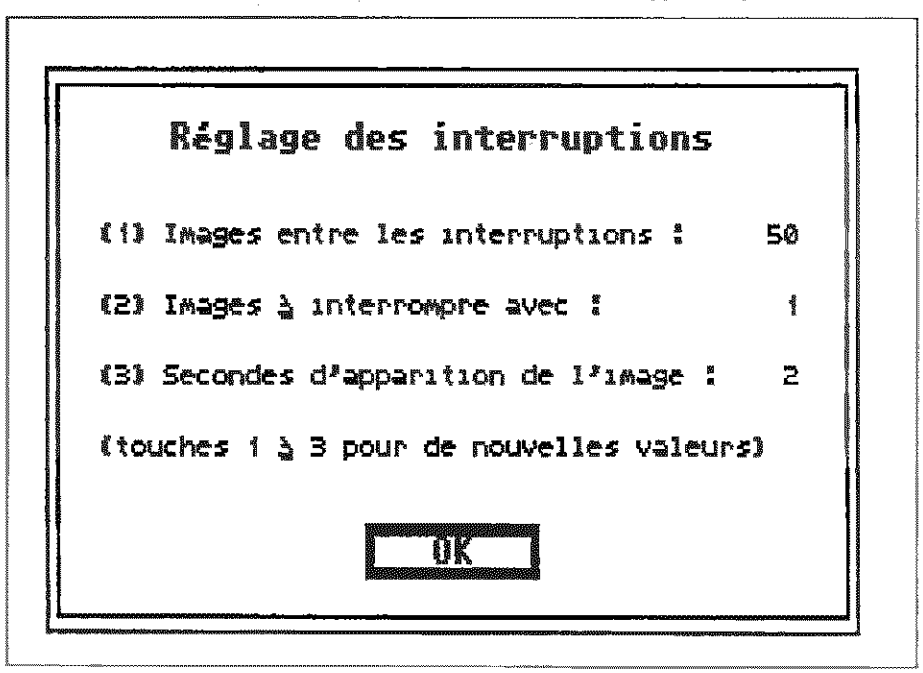


donc «en aveugle». Le problème est évidemment moindre s'il s'agit d'une image fixe prise par une caméra sur pied ou statif.

Le mode Interruption

Ce mode a certainement été créé pour permettre aux revendeurs de réaliser un stand publicitaire pour VIDI-ST et nous n'en voyons guère d'autre utilisation : imaginons la caméra en train de filmer la foule des curieux qui admirent leur image digitalisée en temps réel sur l'écran de l'Atari ; toutes les 3 secondes, un message s'insère entre les images pour expliquer à quel prix ridicule une telle prouesse peut être réalisée grâce à VIDI-ST....

Le menu REGLAGES/INTERRUPTION affiche une boîte qui permet d'ajuster les différents paramètres nécessaires à ce type de présentation :



— combien d'images animées laisse t on entre chaque image fixe ?

— sur combien d'images la présentation fixe doit-elle se dérouler ?

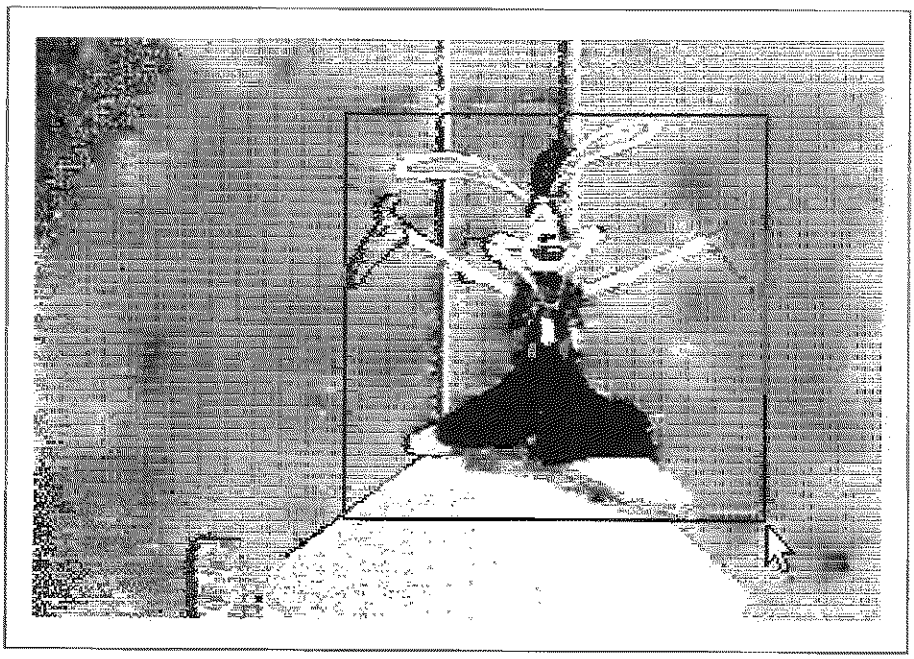
— combien de temps chaque image fixe doit-elle apparaître ?

Il faut bien évidemment préparer les images fixes et penser à les figer afin qu'elles ne soient pas détruites par la digitalisation. On met ensuite à ON l'option située sous le bouton INT de la boîte à outils.

Attention : ce mode ne fonctionne qu'en enregistrement (DIGI) d'une séquence animée (RUN).

DIGITALISER DANS UNE FENETRE

Le menu REGLAGE/FENETRE permet de définir une fenêtre à l'écran par un cliqué-tiré.



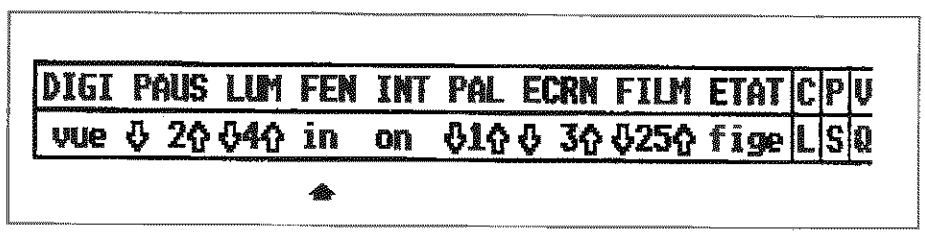
Dans mes bras bébé !

Par la suite la digitalisation peut s'effectuer suivant les trois modes suivants :

— dans la fenêtre, bouton FEN à IN, le reste de l'écran reproduisant une image précédemment digitalisée;

— hors de la fenêtre, bouton FEN à OUT, l'image de la fenêtre n'étant pas altérée;

— en plein écran, bouton FEN à OFF qui est le mode par défaut.



Malheureusement ce mode présente plusieurs défauts :

si on utilise un magnétoscope, on ne peut pas modifier le cadrage ou la taille de l'image qui se trouve dans la fenêtre, ce qui nécessite de créer une fenêtre de grande taille, de préférence au centre de l'écran où se situent généralement les actions les plus intéressantes. Si on digitalise

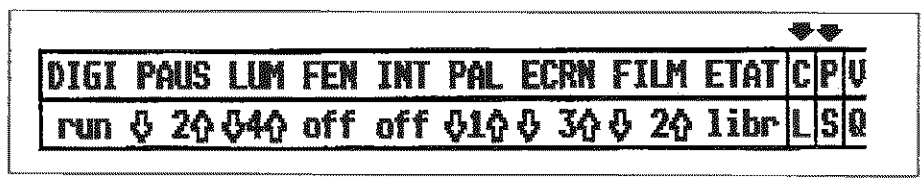
depuis une caméra on peut bien sûr jouer sur le cadrage et le zoom pour que l'image tienne intégralement dans la fenêtre;

— la partie déjà digitalisée à l'extérieur ou à l'intérieur de la fenêtre est obligatoirement fixe : par exemple, on choisit une image, on active la fenêtre, puis on lance la digitalisation qui produit une animation à l'intérieur de la fenêtre ; l'image à l'extérieur reste par contre toujours la même. Si on veut simuler la fonction PIP de certains téléviseurs (deux programmes différents sur le même écran), il faut recommencer la digitalisation dans la fenêtre pour chaque écran.

Le mode FENETRE n'est à nouveau intéressant que pour préparer des écrans de type publicitaire : par exemple, un commentaire fixe apparaît dans la fenêtre alors que le reste de l'écran est animé...

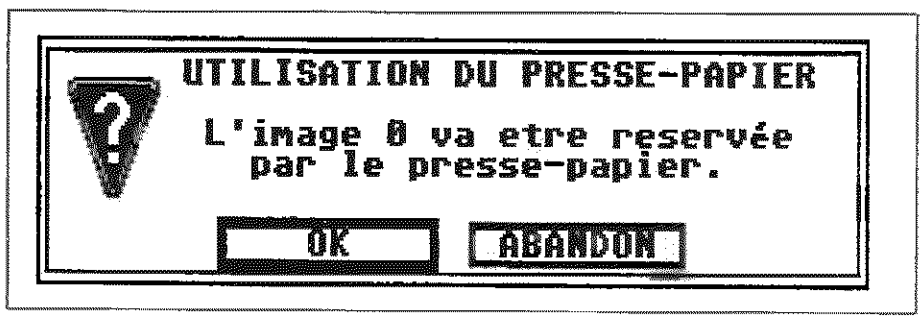
COUPER, COLLER ET PRESSE-PAPIER

Le menu OPTION/COUPER ou le bouton C de la boîte à outils permet de définir à l'aide de la souris une zone rectangulaire qui est mémorisée dans un tampon mémoire.



Cette zone peut être restituée sur le même écran ou sur un autre autant de fois qu'on le souhaite par le menu OPTION/COLLER ou le bouton P (Placer) de la boîte à outils ; une confirmation est demandée, l'écran de travail étant définitivement modifié.

Le menu REGLAGES/PRESSE-PAPIER libère l'écran 0 pour en faire un presse-papier. A partir de ce moment-là, tout élément coupé est rangé dans le presse-papier à la position souhaitée par l'utilisateur. Lorsqu'on choisit l'option COLLER, le contenu du presse-papier apparaît et on choisit d'abord l'élément désiré avant de le placer sur l'écran de travail.



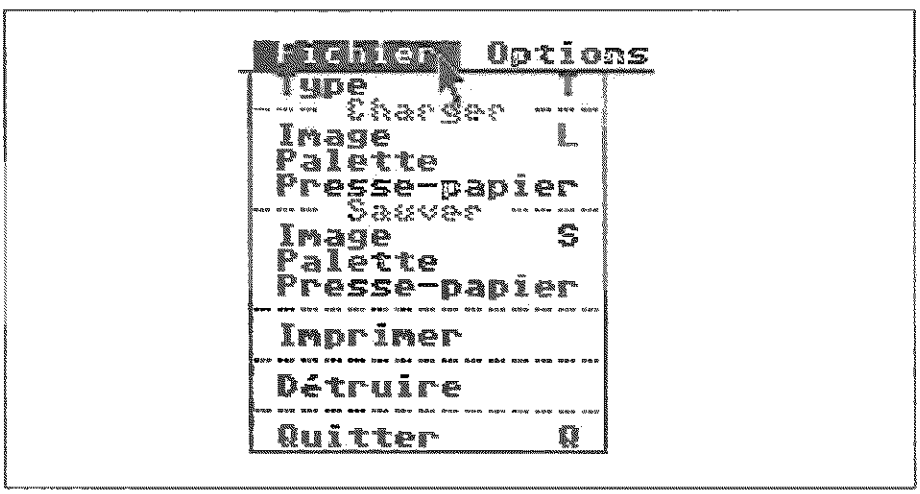
Attention : les éléments à l'intérieur du presse-papier ne doivent pas se chevaucher (sauf effet spécial) et leur nombre est forcément limité par leur taille propre et la taille de l'écran.

LES AUTRES MENUS DE VIDI-ST

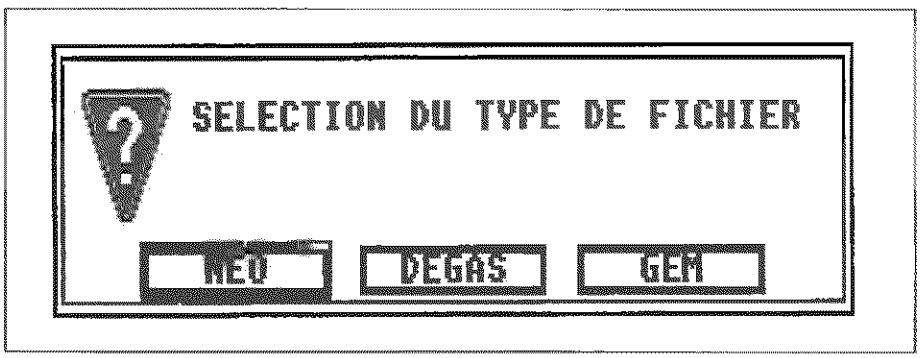
Le bouton LUM

Ce bouton de la boîte à outils vient en complément des boutons matériels de la cartouche. Si la plage de variation de ces derniers n'est pas suffisante, il permet d'augmenter ou de diminuer la lumière de façon logicielle. La valeur de 4 proposée par défaut sera rarement modifiée. L'action de ce bouton ne se fait sentir qu'au moment de la digitalisation et le réglage ne peut donc pas se faire en temps réel puisque le menu disparaît à ce moment là.

Le menu FICHIER



TYPE permet de choisir le format des images tant pour la sauvegarde que pour le chargement.



CHARGER et SAUVEGARDER concernent à la fois les images, les palettes (qu'on peut facilement créer depuis DEGAS si celles qui sont fournies ne suffisent pas) ou le contenu du presse-papier. Ce menu est doublé dans la boîte à outils par les boutons S (Save) et L (Load).

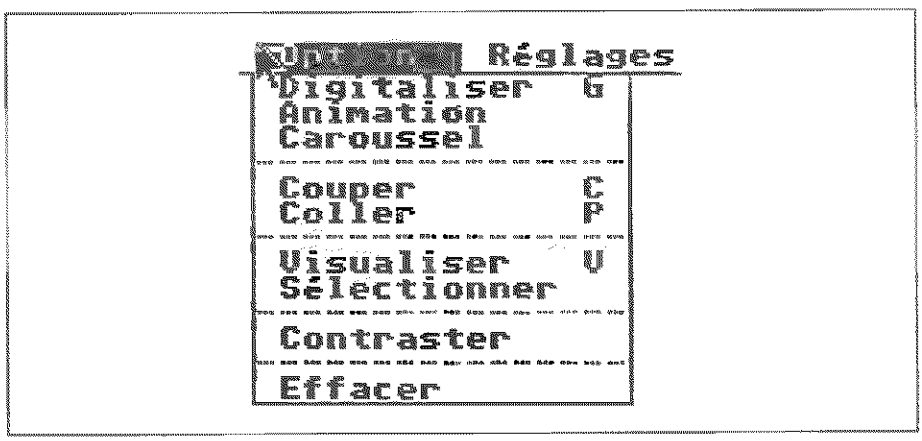


La sauvegarde s'effectue malheureusement image par image ; on ne peut pas mettre sur disque une séquence complète en une seule opération : ainsi sur un MEGA-ST4 la sauvegarde d'une séquence de 10 secondes nécessite 125 appels à ce menu et 6 disquettes pour recevoir toutes les images ; idem pour le chargement...

IMPRIMER réalise une copie d'écran. Pour des sorties définitives, on a intérêt à passer par DEGAS dont le gestionnaire d'impression produit une sortie de meilleure qualité (davantage de niveaux de gris) et de taille supérieure.

DETRUIRE fait de la place sur la disquette lorsqu'on a oublié d'en formatter une pour sauver une géniale production.

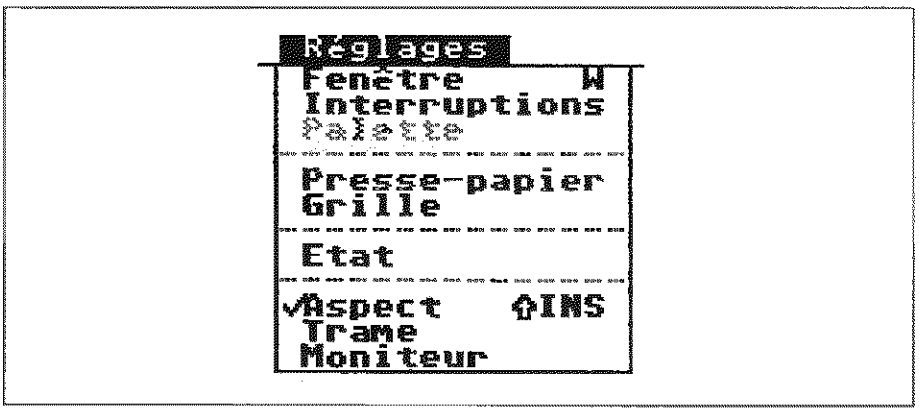
Le menu OPTION



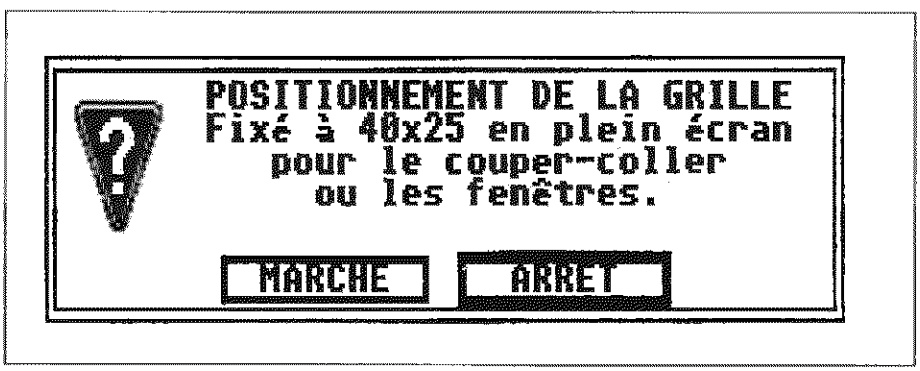
VISUALISER permet d'afficher l'écran de travail en plein écran. On y accède également par le bouton V de la boîte à outils.

EFFACER supprime l'écran courant qui est mis en noir.

Le menu REGLAGES



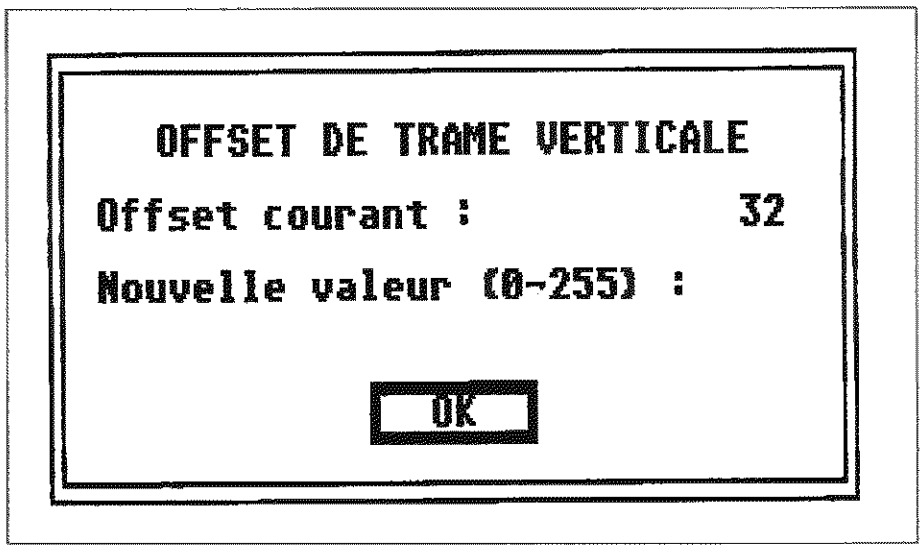
GRILLE active une grille magnétique qui permet de positionner plus précisément les cadres de sélection, que ce soit la fenêtre ou les cadres de COUPER/COLLER.



ETAT affiche les paramètres mémoire de la configuration utilisée.

ASPECT respecte les proportions de l'image, mais perd quelques informations en haut et en bas d'écran, ou la compresse légèrement en hauteur afin qu'elle y figure en totalité.

TRAME retarde ou anticipe le balayage, et décale donc l'image vers le haut ou vers le bas. Un déplacement trop important peut entraîner une perte de synchro et une multiplication des images à l'écran.



MONITEUR passe d'un moniteur 50Hz à un moniteur 60 Hz et inversement.

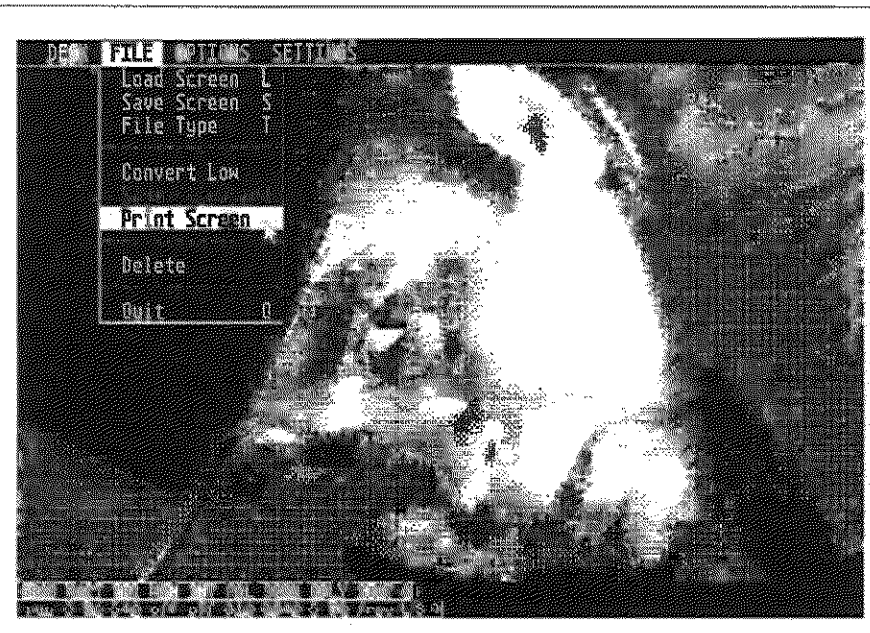
II - Le logiciel VIDIMONO

Dans cette adaptation de l'ancienne version de VIDI-ST (1.2) pour l'écran monochrome haute résolution, les menus ne sont même pas traduits en français, ce qui prouve le désintérêt de l'éditeur pour un logiciel destiné à un marché restreint; les jeux sont en effet plutôt produits en basse résolution.La plupart des menus sont désactivés : pas de palette bien sûr, mais aussi pas de fenêtre, ni d'interruption, ni de planche contact, ni de couper/coller... Quant à la digitalisation en temps réel, elle se produit à un peu moins d'une image seconde, conversion de format oblige...

Les images aux gris tramés sont cependant tout à fait correctes, et certainement aussi bonnes à l'impression que dans la version basse résolution comme le montrent les deux recopies d'écran suivantes.

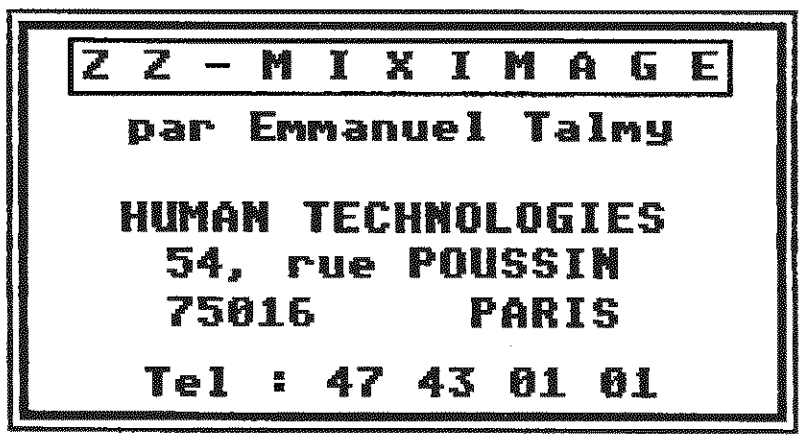


Chèque : en bois ?



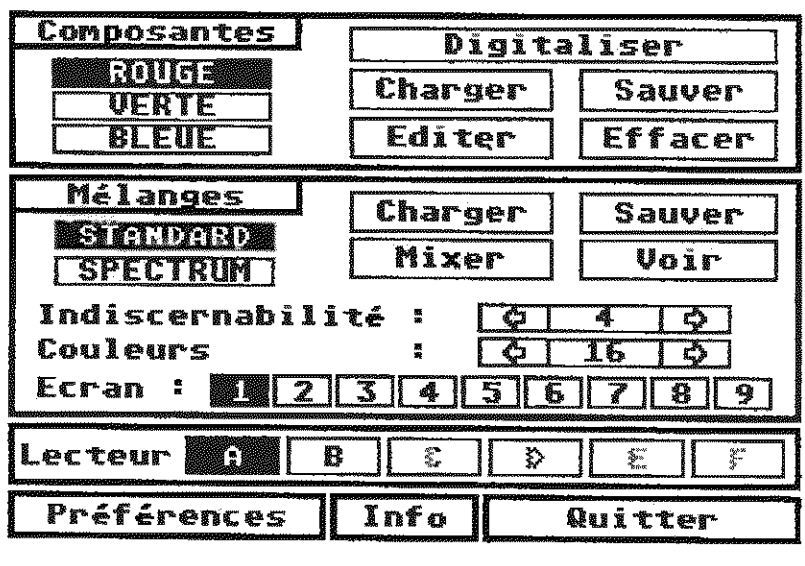
Renversant !

III - Le logiciel MIXIMAGE



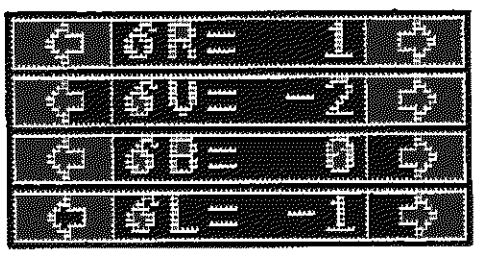
Ce logiciel reproduit les vraies couleurs d'un sujet si celui-ci est digitalisé en trois fois au travers d'un filtre électronique (si on est fortuné) ou de filtres colorés rouge, vert et bleu que l'on doit se procurer auprès d'un photographe au diamètre du porte-filtre de la caméra ou du camescope dont on dispose. Une digitalisation sans bavures nécessite une parfaite immobilité du sujet et de la caméra pendant toute la durée de l'opération, et notamment lors des changements de filtres.

Comme on s'en doute, la première phase consiste à digitaliser le sujet trois fois de suite en plaçant à chaque fois un filtre devant la caméra et en choisissant la composante correspondante depuis l'écran de travail principal.



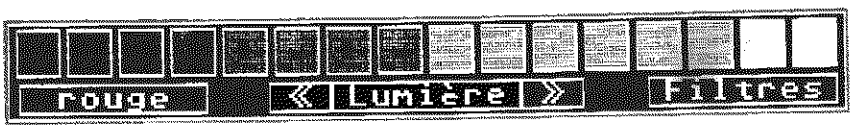
L'écran principal de MIXIMAGE.

Lorsque les trois composantes sont en mémoire et apparaissent précédées d'une coche, un clic sur **MIXER** lance le mélange qui prend quelques secondes. **VOIR** affiche l'image en plein écran et un clic fait apparaître une boîte de réglage qui permet de modifier les quantités de chacune des primaires et de doser l'intensité lumineuse globale.



Dosage des primaires

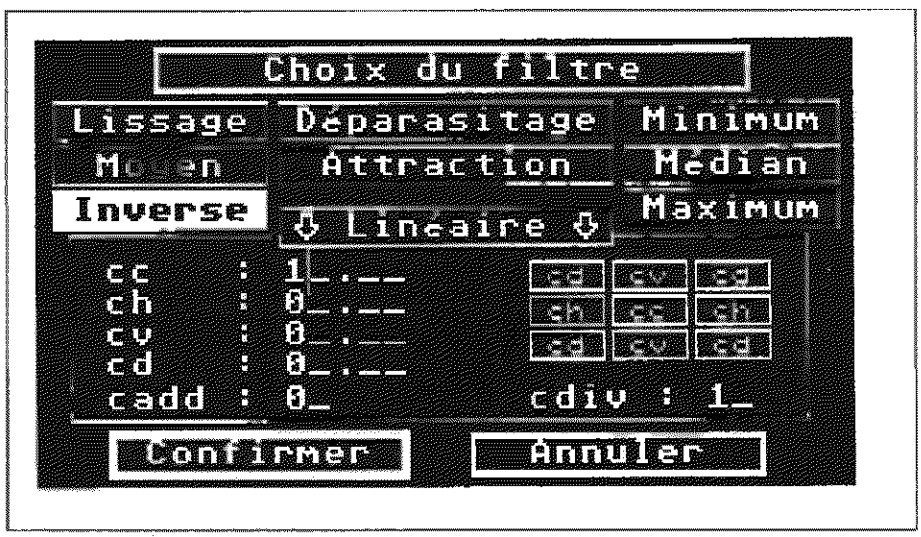
Il est rare que le résultat soit satisfaisant du premier coup ; on recommence l'opération dans un autre écran libre parmi les 9 disponibles et après être passé par le menu **EDITER**. Celui-ci affiche l'image digitalisée dans la primaire correspondante et permet d'intervenir finement sur le dosage des nuances. Un clic sur le sujet fait apparaître une boîte de réglage de la lumière :



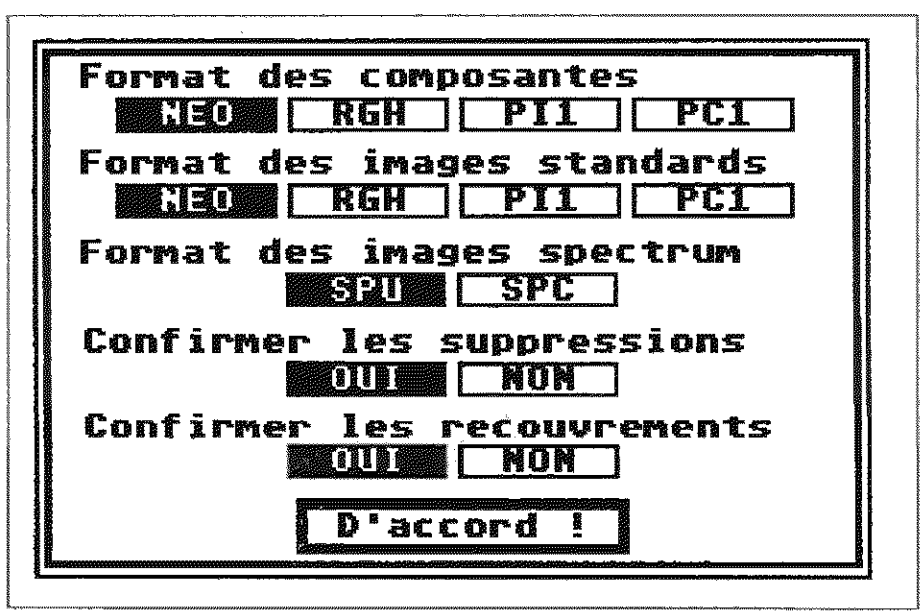
On agit soit sur la totalité des niveaux de gris en cliquant à droite et à gauche du mot **LUMIERE** soit sur le niveau souhaité qu'on éclaircit ou assombrit en cliquant avec le bouton droit, ou le bouton gauche de la souris.

Le bouton **GRIS** permet de mieux différencier les différents niveaux en éclaircissant l'image affichée.

Le bouton **FILTRE** donne accès à une boîte de réglage comprenant de nombreuses options de travail de l'image telles que le lissage des courbes, la suppression des points isolés, le remplissage des petits trous, etc. indispensables pour supprimer les moirages.



Chacune des composantes peut être conservée sur le disque afin de permettre des mixages ultérieurs. Il en est de même de l'image mixée finale, après que l'on ait choisit le format souhaité depuis le bouton PREFERENCES.

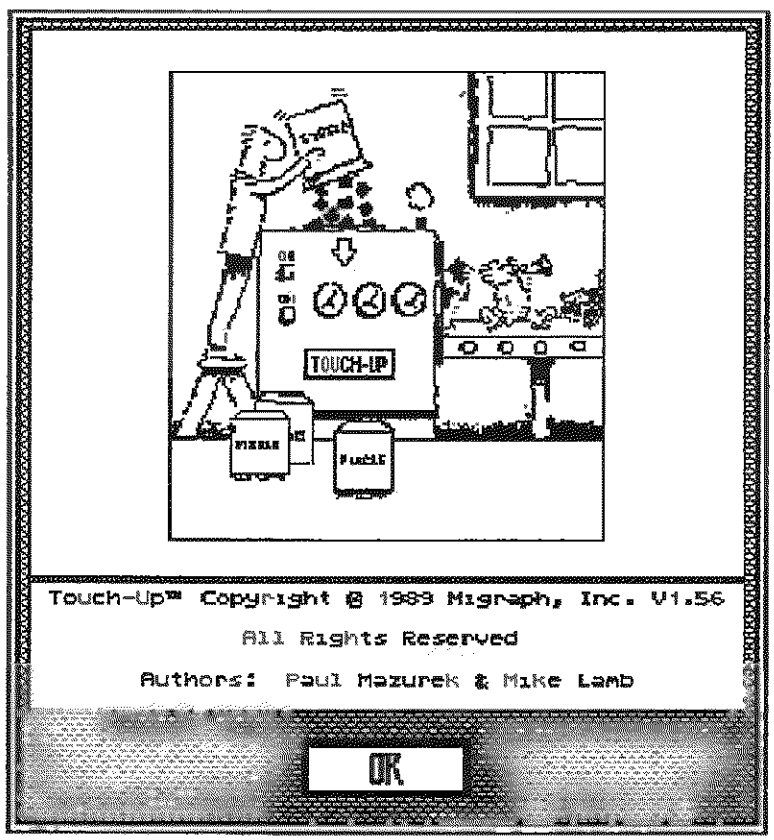




«Happy end» bien sûr !

5/6.2

Le logiciel de retouche Touch-Up et le scanner Golden Image



Cet ensemble distribué par Upgrade fournit à l'amateur de PAO sur écran haute résolution Atari un très bon outil de création, d'acquisition et de retouche d'images monochromes point par point. Le logiciel d'abord distribué sous le nom d'Image Partner figure parmi les meilleurs que nous

ayons essayés dans cette gamme de prix. Il intègre notamment nombre de concepts habituellement présents uniquement dans les logiciels de dessin vectorisés les plus évolués : pré-positionnement des objets, rotation au 1/10e de degré, courbes de Bézier, transparences, etc. Les images sauvées au format .IMG sont sans problème reprises dans Timeworks et l'ensemble des fonctions d'importation et d'exportation suffit à justifier l'achat du logiciel : la plupart des images couleurs provenant des divers logiciels de dessin de l'Atari peuvent être ainsi converties en niveaux de gris de façon beaucoup plus efficace que ne le permet un logiciel de PAO qui a, par ailleurs, bien d'autres problèmes à traiter.

Le scanner souffre certes de sa faible largeur de saisie (10 centimètres), mais on est loin de la dizaine de milliers de francs d'un appareil à plat au format A4.

Quant à la relative mauvaise restitution des gris photographiques sur un écran monochrome, elle constitue en fait un avantage, car elle préfigure bien le document tel qu'il sera imprimé : pas plus l'écran de l'Atari que l'imprimante ne peuvent pour l'instant afficher un point avec d'autres intensités que le blanc pur ou le noir profond.

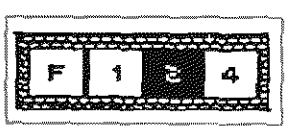
INSTALLATION

Attention ! Le logiciel réclame au minimum un méga-octet de mémoire pour pouvoir fonctionner, deux mégas ne constituent pas un luxe ! L'utilisation est plus agréable sur un écran monochrome (la couleur ne se justifie d'ailleurs pas ici).

Si l'on ne dispose que d'un lecteur de disquette, on peut préparer une disquette formattée destinée à recevoir les sauvegardes des dessins. On y ajoutera une copie du dossier FONTS de la disquette originale si l'on souhaite saisir du texte et éventuellement une copie du dossier GEMSYS qui contient les pilotes d'imprimante pour pouvoir imprimer depuis le logiciel.

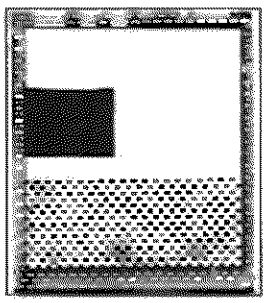
Il est préférable de booter l'ordinateur avec la disquette originale n°1 du programme qui place le GDOS en résidant : le GDOS est indispensable pour les impressions avec les pilotes fournis. Le programme chargé, on introduit enfin la disquette des sauvegardes pour la totalité de la séance de travail.

LES AIDES AU DESSIN



Le zoom

On dispose de quatre niveaux d'agrandissement : pleine page, normal, grossissement deux fois et quatre fois. Certains outils ne fonctionnent qu'au niveau normal. Le mode Flash possède également une loupe permettant de travailler très confortablement au point près.



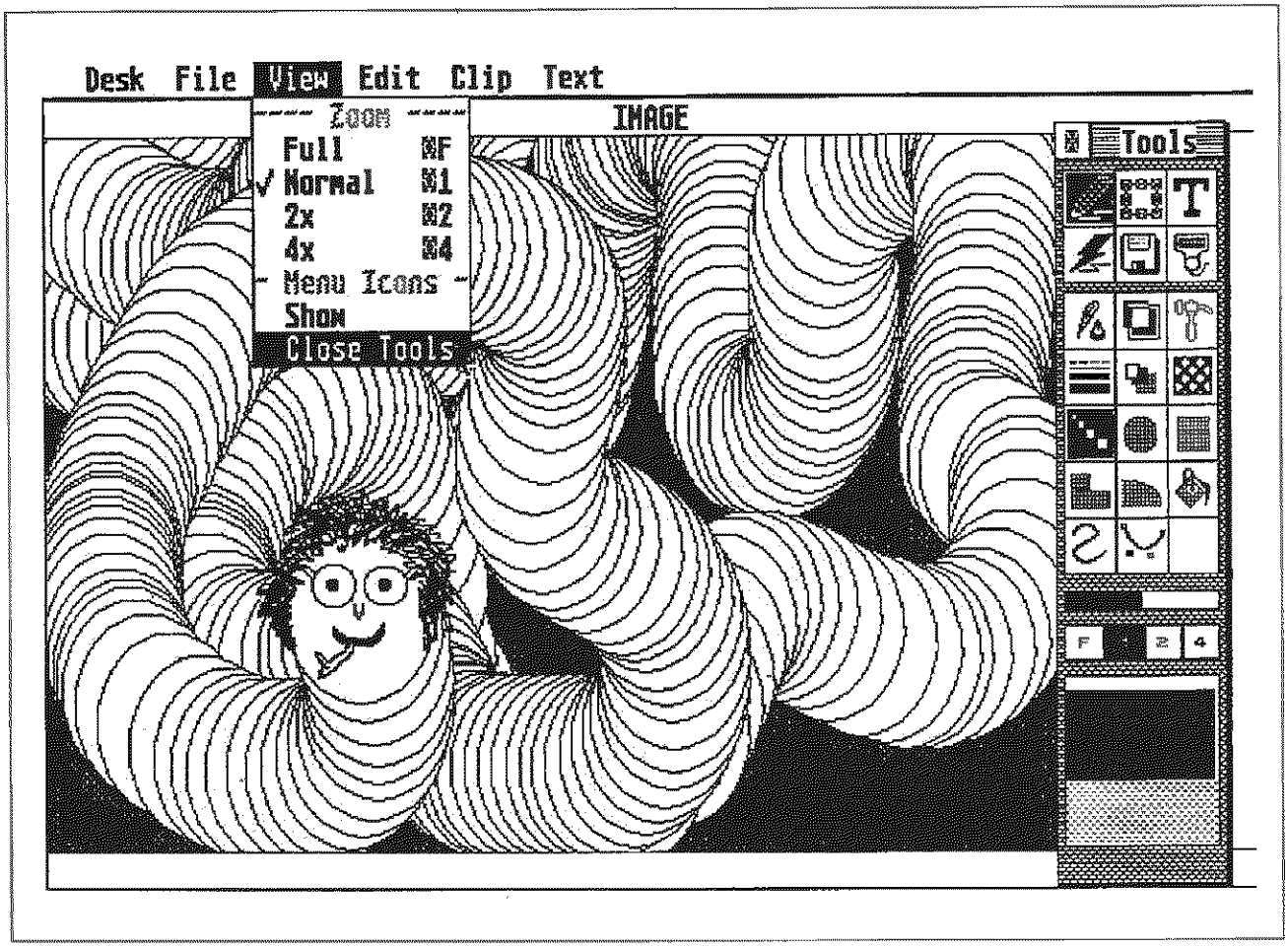
Le repositionneur

Placé sous les icônes de zoom, il représente sous forme d'un rectangle noir la position de l'écran sur la page blanche. Les zones en grisé figurent l'extérieur de la feuille. On peut amener le repositionneur n'importe où sur la page par un cliqué tiré dans l'icône même, c'est plus rapide que les ascenseurs mais moins précis.

Les menus

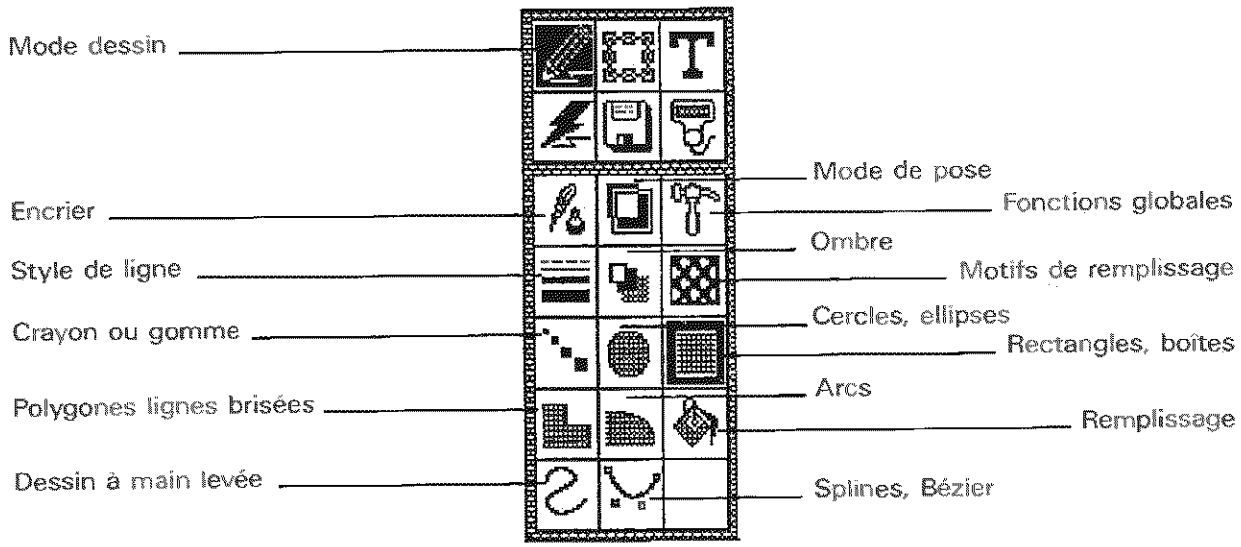
Pratiquement toutes les fonctions disponibles dans les menus du bandeau supérieur sont également accessibles depuis les outils qui apparaissent dans la partie droite de l'écran, aussi ne les décrirons-nous pas sauf quatre qui n'existent nulle part ailleurs :

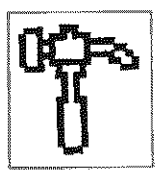
- File/Quit qui permet de sortir du programme.
- File/Outprint qui charge le programme IMPRIME pour une impression en dehors de Touch-Up.
- View/Hide ou Wiew/Show qui cache ou qui montre la boîte à outils à droite de l'écran.
- View/Tear-Aways ou View/Close-Tools qui permet de déplacer la boîte à outils par un simple cliqué tiré sur son bandeau supérieur ou de la remettre à sa place.



Les outils peuvent être déplacés n'importe où sur l'écran.

LE MODE DESSIN





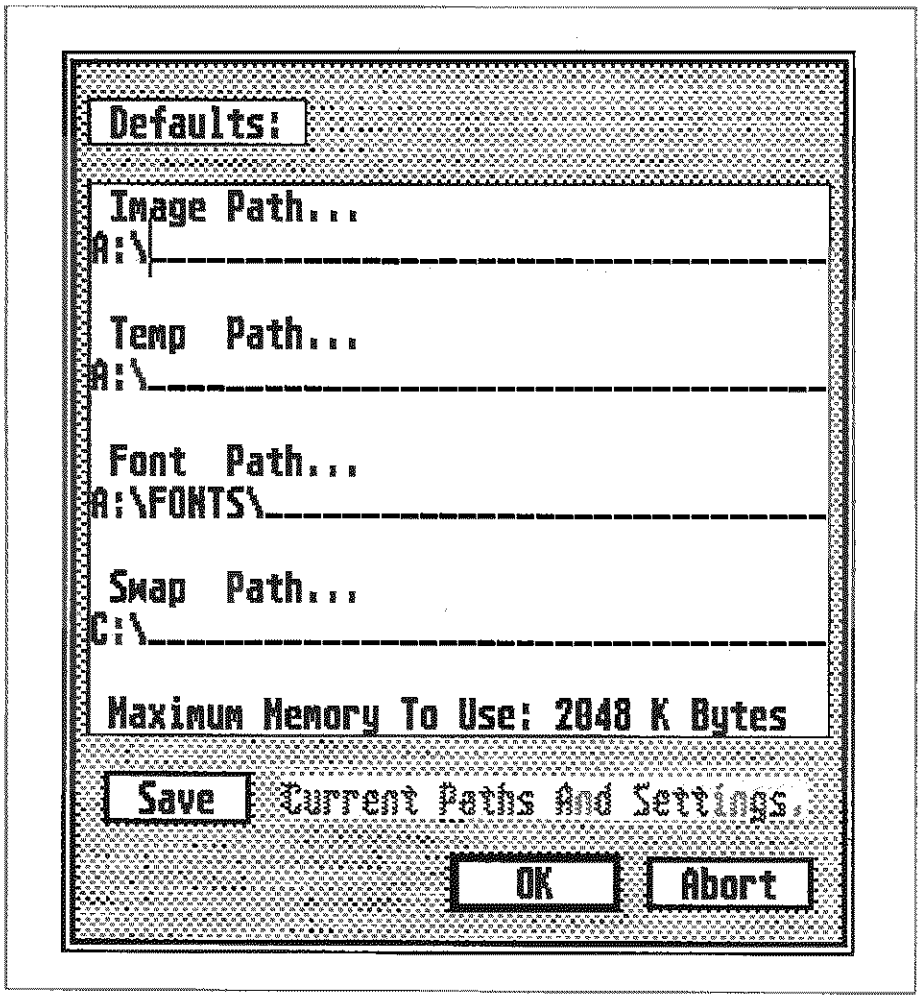
Les fonctions globales

Ces fonctions sont généralement utilisées avant de commencer un dessin.

Misc tools

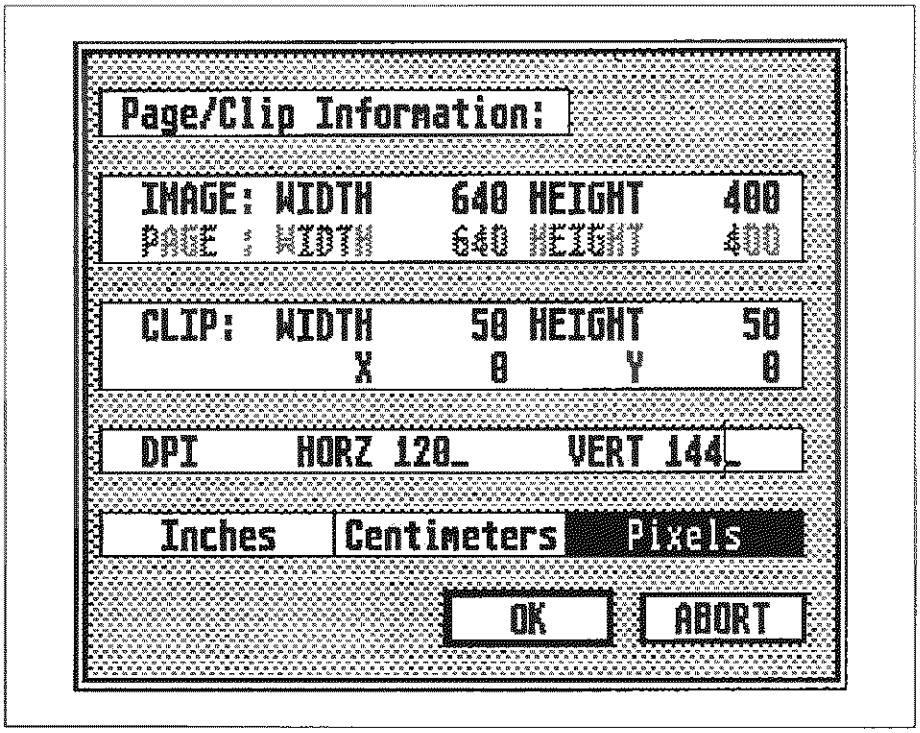
Clear Page	(efface la page)
Invert Page	(inverse les couleurs)
Clip to Page	(sélecteur pleine page)
Defaults	(chemins par défaut)
Page/Clip	(dimensions page...)
Lock	(fixer les objets)

• Defaults fait apparaître une boîte de saisie qui permet de définir les accès par défaut aux images, au presse-papier, aux polices et à la réserve de mémoire (mémoire virtuelle sur disque pour les très grands dessins : à ne pas utiliser sans disque dur). En cas de modification du chemin d'accès aux polices ou de la mémoire maximale, il est nécessaire de sauver la configuration puis de relancer le logiciel, car c'est seulement au chargement que ces informations sont prises en compte. Attention, lors de la sauvegarde, tous les paramètres des menus sont également sauvés.





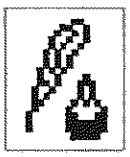
• Page/Clip ouvre une boîte de saisie qui permet d'entrer la taille de l'image ainsi que la taille et la position du sélecteur d'objets (Clip). Si l'on définit une taille d'image supérieure à la taille de la page, cette dernière s'ajuste automatiquement ; le dessin présent à l'écran peut, après confirmation d'une boîte de dialogue, soit être effacé, soit conservé (en étant éventuellement tronqué si la taille de l'image a été réduite). La boîte de saisie permet également d'ajuster la résolution de l'image afin qu'elle corresponde à celle de l'imprimante (DPI HORZ et VERT) ; il est préférable que la taille de l'image soit un multiple de la résolution de l'imprimante afin d'éviter tout effet de moire.



• Lock est par défaut désactivé ce qui fait que les dessins peuvent être repositionnés précisément avant d'être définitivement fixés sur la page. Il peut être intéressant de l'activer lors de la construction de polygones comme nous le verrons par la suite et uniquement dans ce cas-là.

Les réglages

Les effets obtenus à l'aide des différents outils dépendent des réglages préalablement effectués depuis les différents menus.



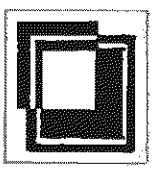
L'encrier

L'activation de cette icône déroule le menu suivant :

Color	
---- *Pen* ----	(*crayon*)
Black	(noir)
White	(blanc)
Opposite	(inverse)
---- *Fill* ----	(*remplissage*)
Black	(noir)
White	(blanc)

On peut dessiner en blanc sur un objet noir ou utiliser un crayon blanc à la manière d'une gomme. Les retouches sont d'ailleurs simplifiées grâce à la fonction **Opposite** qui attribue automatiquement : la couleur blanche au crayon si l'on commence à dessiner sur un point déjà noirci, et la couleur noire si l'on commence sur un point blanc de l'écran. La couleur ainsi attribuée reste active tout au long du tracé tant que le cliqué de la souris est maintenu.

Une méthode rapide de sélection de la couleur consiste à la choisir grâce aux deux boutons noir et blanc situés sous la boîte à outils.



Le mode de pose

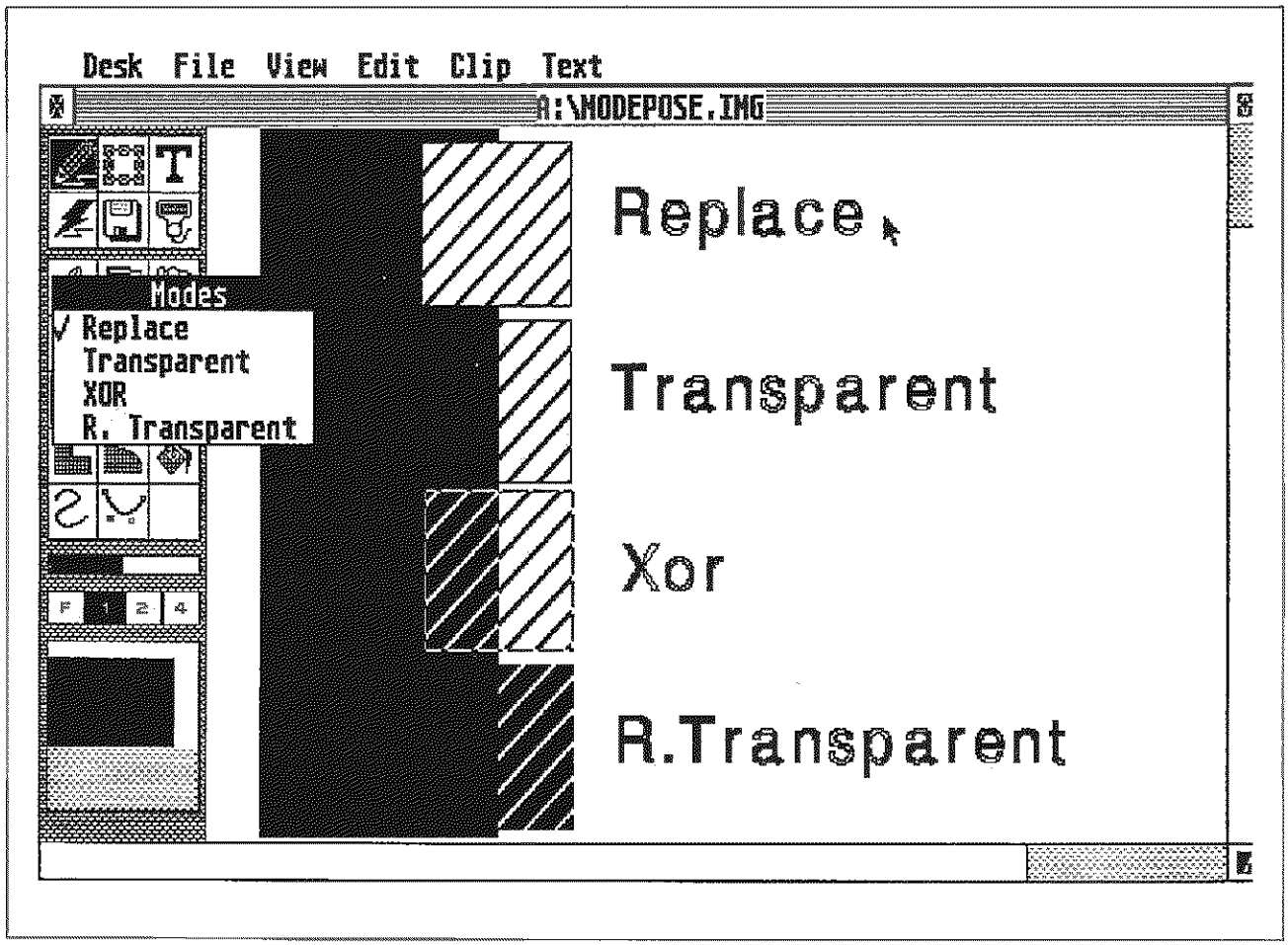
Quatre poses différentes pour un même carré.

Le premier carré recouvre une partie du rectangle noir (**Replace**)

La zone blanche du second carré est transparente et laisse voir le rectangle noir : les zébrures sont donc «noyées» dans le noir du rectangle (**Transparent**)

Le mode **XOR**, bien connu des créateurs de jeux puisqu'il permet de déplacer un sprite sur n'importe quel fond en le laissant visible, effectue un OU logique exclusif entre chaque point du dessin de fond et chaque point du dessin qu'on lui superpose : deux points noirs apparaissent donc en blanc ; c'est le cas des zébrures sur le rectangle noir, mais c'est aussi le cas de ces mêmes zébrures de remplissage avec le tour du carré (on voit un point blanc à l'endroit du contact). Il est préférable d'éviter le mode **XOR** avec une figure contenant un motif et des traits larges.

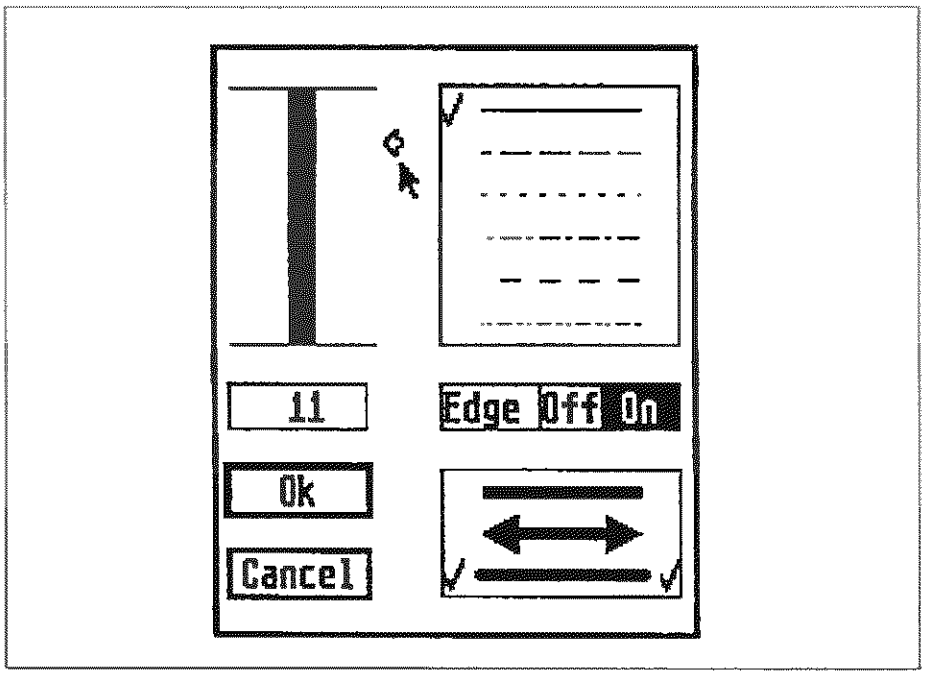
Le mode **R.Transparent** inverse d'abord la couleur de l'objet, puis laisse les zones blanches transparentes.



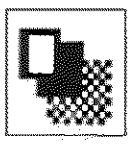
Le mode de pose.



Le style de ligne

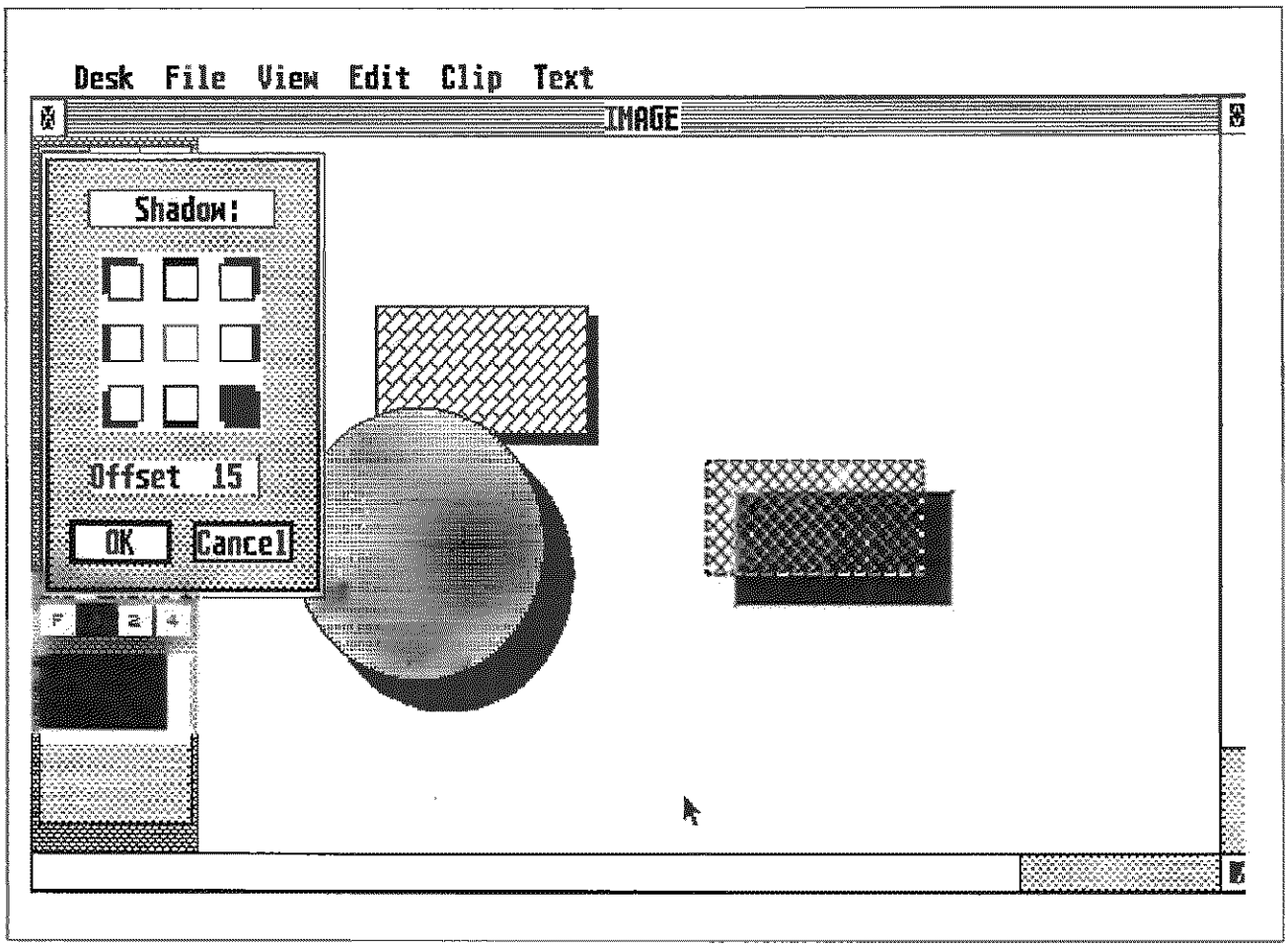


On choisit le style du trait (pointillé, continu...), sa largeur (de 1 à 61 points) mais uniquement pour un trait continu, la forme de ses extrémités.

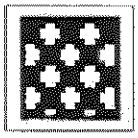


L'ombre

On détermine la position de l'ombre et le déplacement (**Offset**) en points (appuyer sur **[ESC]** pour effacer l'ancienne valeur et entrer la nouvelle). Si l'objet est transparent, on voit la totalité de l'ombre qui se trouve normalement par dessous comme le montre notre recopie d'écran qui présente un rectangle avec un offset de 5, un cercle avec un offset de 15 et un rectangle «résillé» en mode XOR (voir page suivante).

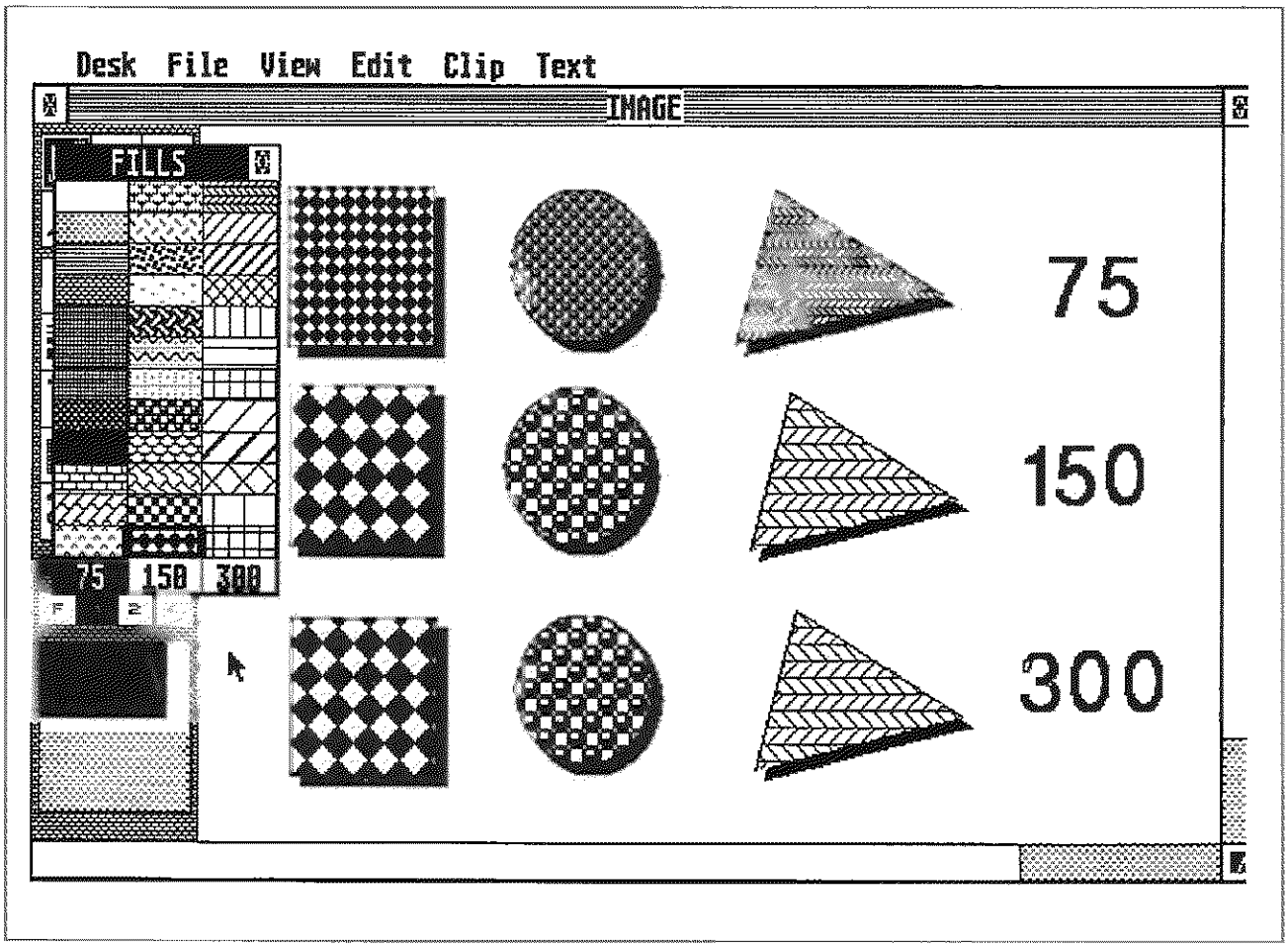


Positionnement de l'ombre.

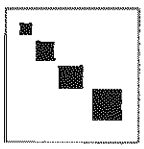


Les motifs de remplissage

Trente-six motifs sont proposés dont 7 niveaux de gris. Les boutons 75, 150 et 300 doivent être choisis au plus près de la résolution de l'imprimante dont on dispose afin que les gris soient correctement échelonnés : plus la résolution de l'imprimante est élevée, plus le dessin de l'écran se trouve réduit et plus le motif de remplissage doit être gros (on voit cependant que de 150 à 300 les motifs sont les mêmes, ce n'est pas le cas des grisés).



La résolution doit être ajustée au plus près de celle de l'imprimante.



Les outils de tracé

Le crayon (ou gomme)

Contrairement aux autres outils de tracé de ce mode, les points sont définitivement placés ou effacés à l'écran. Bien qu'il permette également le tracé à main levée, le crayon associé à un grossissement important est plus particulièrement adapté aux fines retouches sur un dessin. On sélectionne la taille de la mine de 1 à 4 depuis les nombres obtenus à l'aide de la touche [SHIFT] sur le clavier alphanumérique : pavé de 1 point, 3 points, cinq points ou sept points de côté. Très pratique, la touche [SHIFT] qui accompagne un cliqué-tiré et qui force un tracé horizontal alors que la touche [ALT] force, elle, un tracé vertical.

Si la valeur White ou Opposite est active dans le menu Encrier, le crayon peut être utilisé comme une gomme de précision.

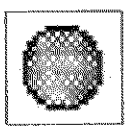


Les outils permettant un positionnement

Les outils qui vont suivre offrent tous la possibilité de positionner précisément par un cliqué-tiré l'objet qu'on vient de réaliser puis de le fixer définitivement par un double-clic-droit ou de l'annuler par un clic-droit/clic-gauche simultané.

Dans le menu placé sous l'icône **Outils divers** l'état **Lock** annule la possibilité de repositionnement des objets avant fixation définitive, ce qui facilite les tracés à l'aide de l'outil **Polygone** (la possibilité d'annulation du dessin en cours par un clic simultané des deux boutons est cependant conservée).

Pour la plupart des outils les attributs réglés par défaut (aspect de la bordure, du remplissage....) n'apparaissent que lorsque l'objet est définitivement fixé à l'aide du double-clic-droit.

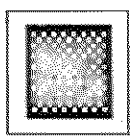


Les cercles et ellipses

Le pointeur étant sur la page, un cliqué-tiré agrandit l'ellipse qui se transforme en cercle si l'on maintient la touche [ALT].

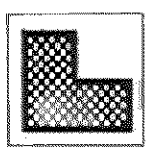
Après avoir relâché le clic, on peut encore déplacer l'objet pour le positionner précisément.

Un clic-gauche le fixe définitivement.



Les rectangles et boîtes

Un cliqué-tiré agrandit le rectangle suivant une de ses diagonales. Un appui simultané sur [ALT] force un carré, alors que [SHIFT] produit une boîte aux coins arrondis.



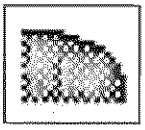
Les polygones et lignes brisées

Attention : si l'on souhaite simplement obtenir une ligne brisée, il faut veiller à ce que le motif de remplissage actif soit blanc et à ce que le mode de pose soit transparent ; dans le cas contraire, on obtient une surface qui masque ce qu'elle recouvre.

Un cliqué-tiré permet d'obtenir un premier côté ou un premier segment, un appui sur [CONTROL] force son inclinaison à 45°. Si l'on relâche le clic, on peut encore déplacer ce segment sur l'écran. Un clic-droit le fixe provisoirement afin que le suivant puisse être «tiré» depuis le point que l'on vient de placer, etc.

Un double-clic droit fixe la figure définitivement.

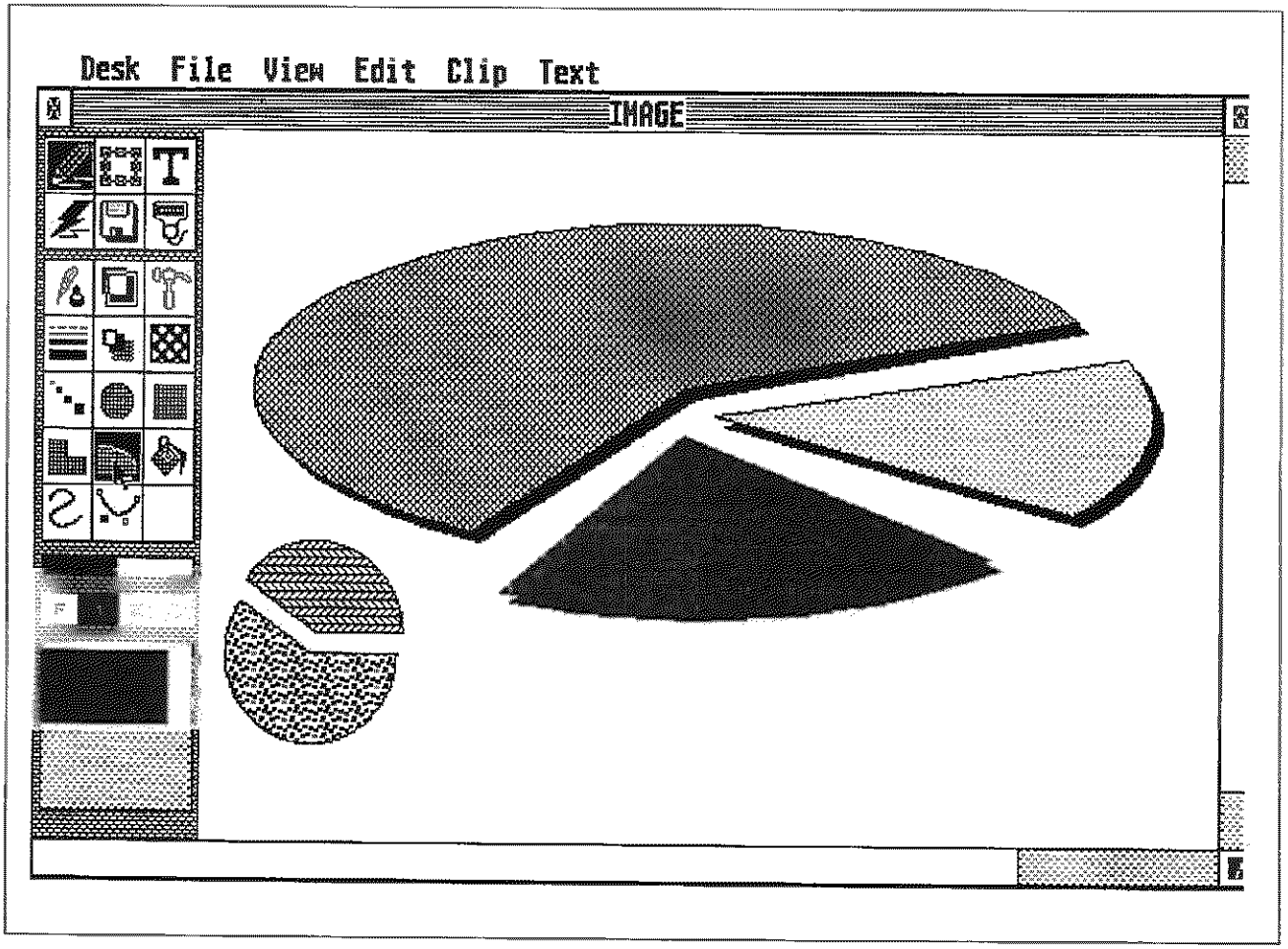
La possibilité de déplacement de la figure entre chaque segment fait que la procédure est un peu déroutante la première fois qu'on l'utilise ; si l'on éprouve par la suite des difficultés de réalisation, il est préférable d'activer l'état **Lock** du menu **Outils divers**.



Les arcs

Cet outil est particulièrement bien conçu dans Touch-Up.

Un cliqué-tiré agrandit un arc correspondant à un quart d'ellipse. La touche [ALT] le force au quart de cercle. Après relâchement du clic, on modifie la valeur de l'angle à l'aide des flèches du clavier par pas de 5° ou d'un demi-degré si l'on appuie simultanément sur [SHIFT]. Un clic-droit fixe simplement l'arc alors qu'un appui simultané sur [SHIFT] remplit en plus le secteur angulaire à l'aide du motif courant (portion de camembert).



Classiques ou originaux : des effets faciles à obtenir.

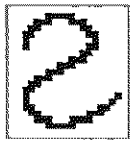
A noter : il est préférable de n'utiliser l'ombre que pour une «portion de camembert», sinon on obtient deux arcs légèrement décalés (l'arc et son ombre).



Le remplissage

Le remplissage d'une figure s'effectue avec le motif sélectionné par défaut.

Cet outil requiert les précautions habituelles : sauvegarde préalable du dessin, vérification des fuites possibles au niveau des traits de contour. En cas de fuite, toute la page se remplit et l'annulation est impossible (l'annulation existe dans le menu **Flash** qu'il est préférable d'utiliser pour cette fonction).



Le dessin à main levée

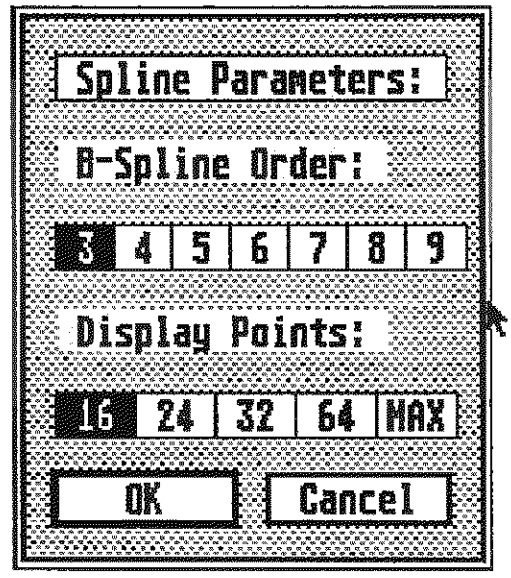
On dessine tant que le cliqué-tiré est maintenu et que la figure n'excède pas 128 points de long, ce qui n'autorise pas des réalisations très complexes.

On déplace la figure, puis on la fixe par un clic-droit ; elle apparaît remplie du motif courant.



B-SPlines et courbes de Bézier

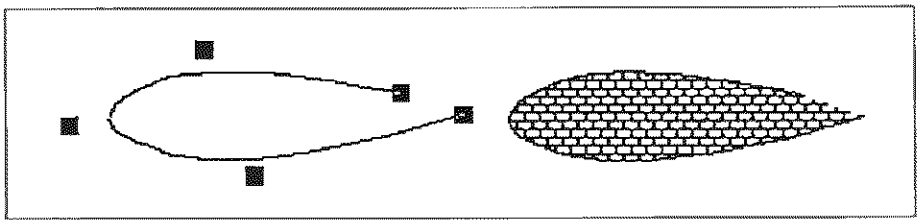
Voici deux outils très puissants habituellement réservés aux logiciels de dessin vectorisé. Les paramètres de la **SPline** sont définis depuis une boîte de réglage accessible depuis le menu **Parameters**.



- B-SPline

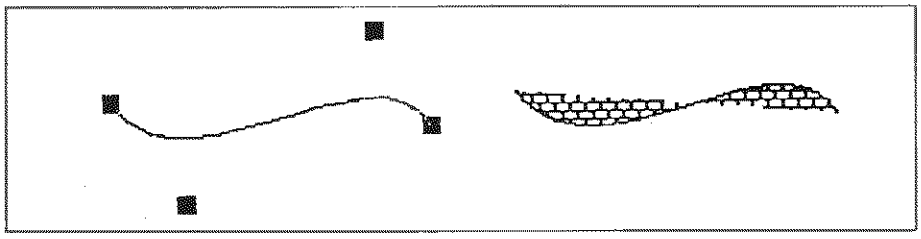
On place le point de départ puis les différents points de contrôle de la courbe. Un cliqué-tiré sur un point le déplace et modifie la courbe en conséquence. Un clic-droit sur un point le supprime. Un [SHIFT] cliqué-tiré-droit déplace l'ensemble de la figure. Un double-clic-droit la fixe

définitivement et la remplit du motif actif. La dernière figure ainsi créée peut être dupliquée par un [SHIFT] clic-gauche qui fait réapparaître les points de contrôle ; on déplace ensuite le double par un [SHIFT] cliqué-tiré.

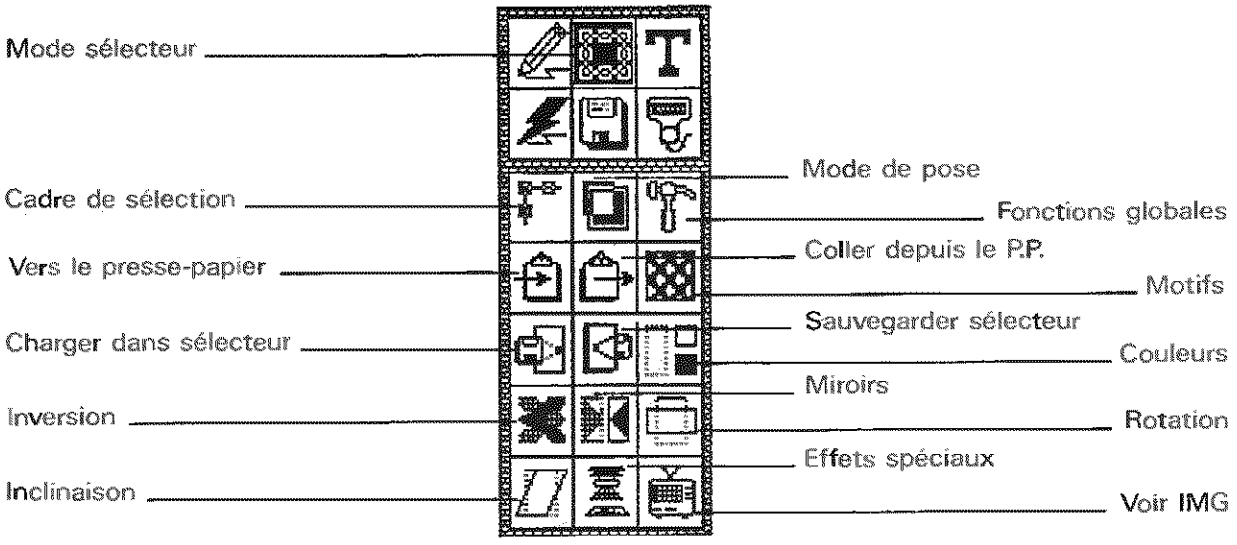


- Courbe de Bézier

Elle n'est définie que par quatre points. On place le point de départ puis deux points de contrôle et enfin le point d'arrivée. Les possibilités de déformations, de déplacements et de duplication en combinaison avec les touches du clavier sont les mêmes que pour la SPline.



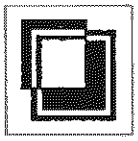
LE MODE SELECTEUR (OU MODE CLIP)



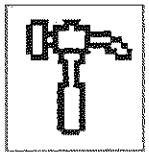
Toute zone sélectionnée à l'écran peut subir diverses modifications telles que déplacement, duplication, rotation, inversion, miroir...

Les fonctions communes aux outils de tracé

Elles ne seront pas à nouveau détaillées ici.

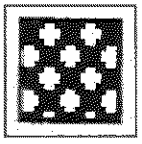


Le mode de pose



Les fonctions globales

Rappelons que l'option «Clip To Page» agrandit la boîte de sélection à la totalité de la page alors que «Page/Clip» ajuste précisément sa taille et sa position.



Les motifs

Travail sur une zone

Sélection d'une zone

Lors du premier appel, la boîte de sélection apparaît dans l'angle supérieur gauche de la page. On la déplace par un cliqué-tiré en son centre et on la déforme par un cliqué-tiré sur une des huit poignées qui la bordent.

On peut plus simplement entourer l'objet à sélectionner en le saisissant au lasso par un cliqué-tiré en diagonale qui aura pour objet de créer une nouvelle boîte de sélection et de détruire l'ancienne.

Déplacement, duplication d'une zone

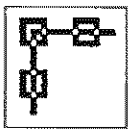
[Shift] au relâchement du cliqué-tiré déplace la zone à la position du sélecteur ; elle vient éventuellement recouvrir une autre zone suivant le mode de pose actif.

[Control] au relâchement du cliqué-tiré duplique la zone de départ ; c'est la zone de destination qui est alors sélectionnée.

[Control][Alt] dans les mêmes conditions la duplique, mais c'est la zone de départ qui reste sélectionnée.

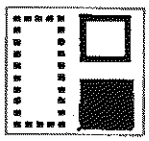
Déformation d'une zone

[Shift] cliqué-tiré sur une des poignées de déformation redimensionne l'objet à l'intérieur du sélecteur : si des trames grisées ont été utilisées, des moires apparaissent.



Le cadre de sélection

Un clic sur l'icône fait apparaître le rectangle de sélection à l'écran, un autre clic le fait disparaître.

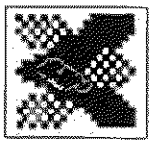


Les couleurs

Color	
White	(blanc)
Black	(noir)
Invert	(inversion)

Les couleurs choisies n'affectent que la boîte de sélection. Attention, White ou Black effacent tout ce qui s'y trouvait.

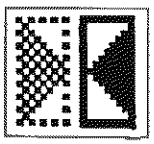
Deux inversions successives rétablissent l'état initial du sélecteur.



L'inversion

Flip	
Horizontal	
Vertical	
Both	(les deux)

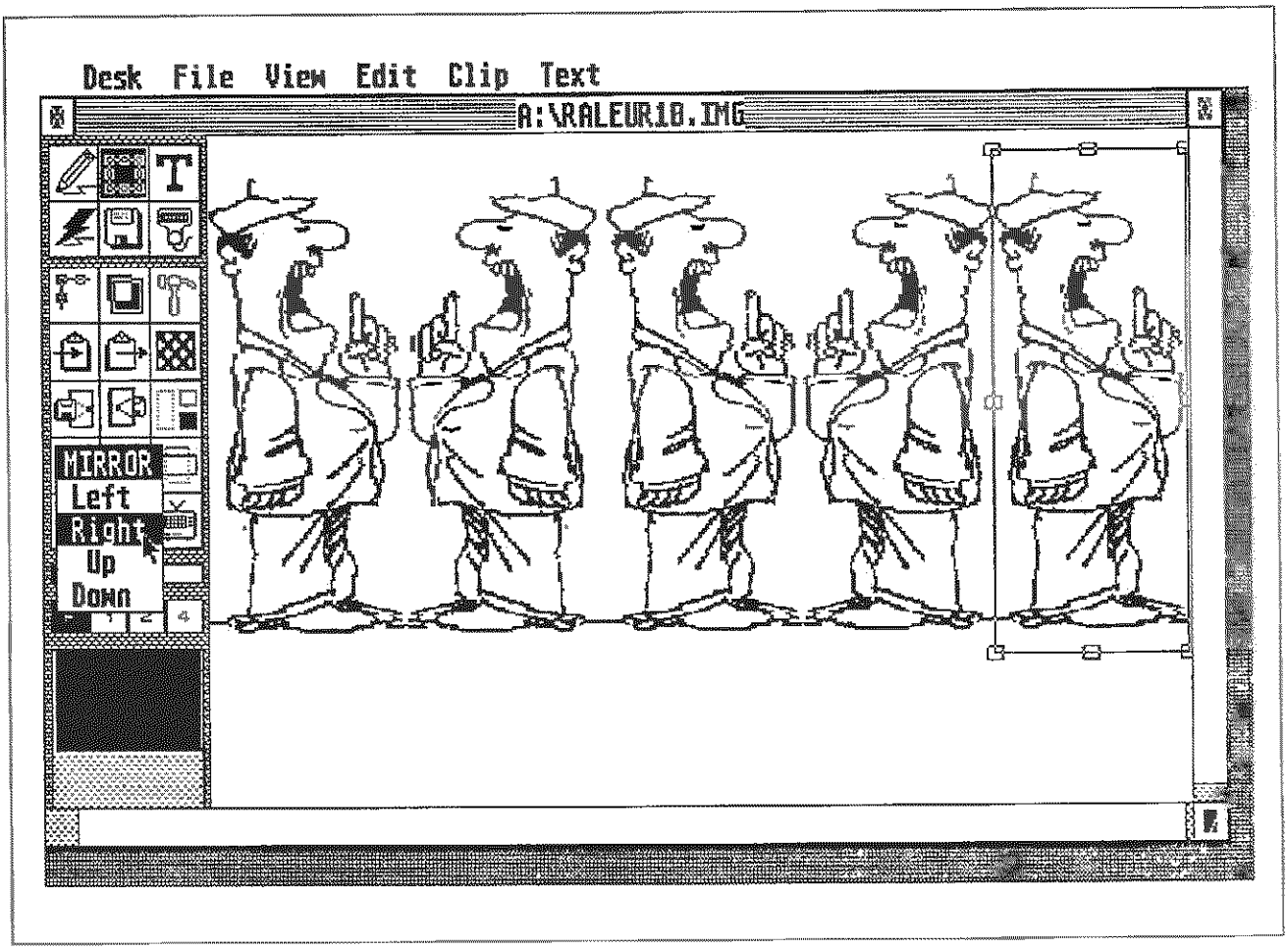
L'image est retournée sur elle même. "Both" correspond à une rotation de 180°.



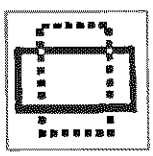
Les miroirs

Mirror	
Left	(gauche)
Right	(droit)
Up	(haut)
Down	(bas)

Il est facile de réaliser des frises à l'aide de la fonction **Miroir** : si l'on appuie sur **[Shift]** au moment du choix de la direction (par exemple Right), le cadre de sélection se trouve sur l'objet reproduit qui est ainsi prêt à être reproduit à son tour.



Une frise rapide à réaliser grâce aux miroirs.

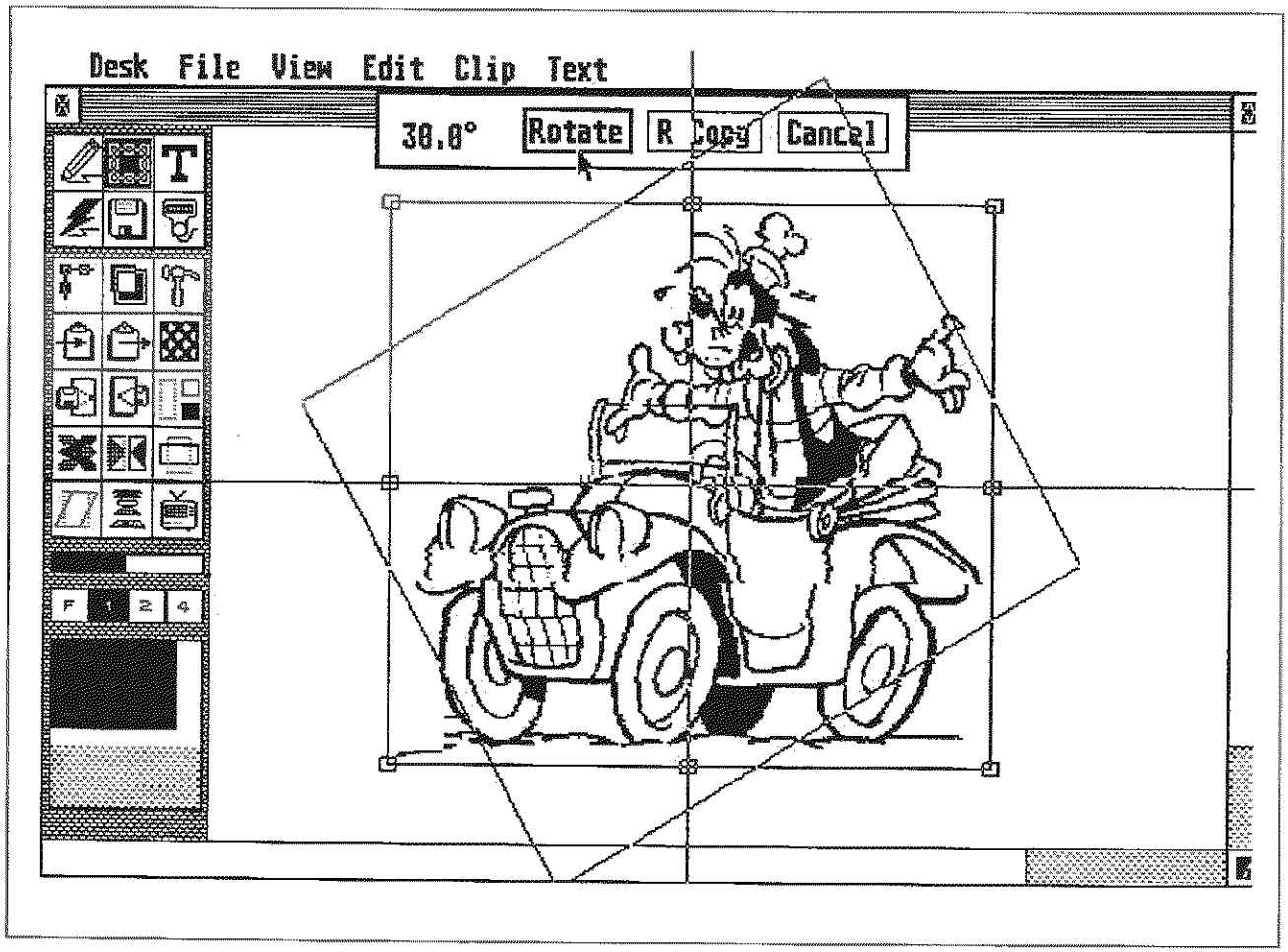


La rotation

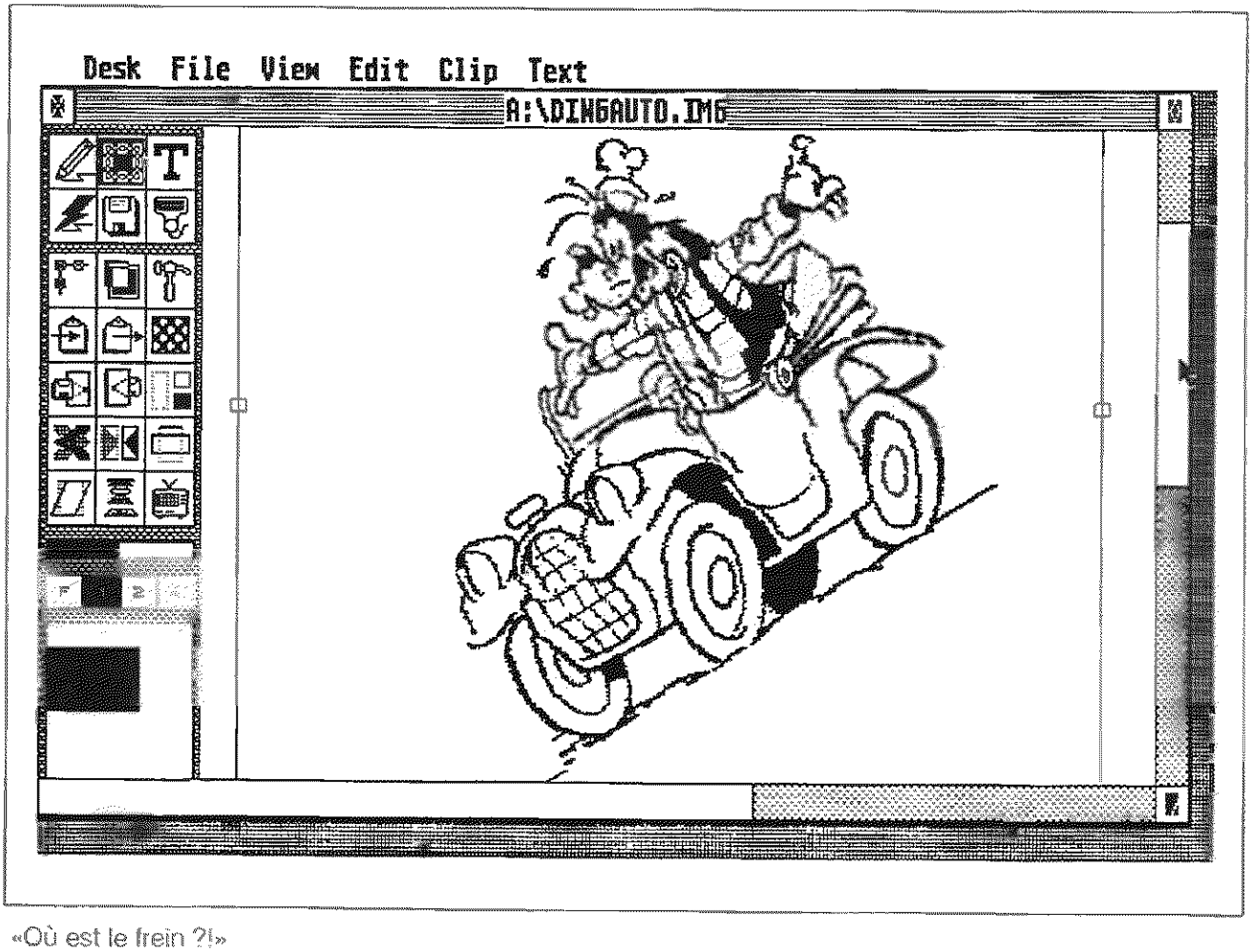
Les possibilités sont intéressantes puisqu'on peut travailler au 1/10e de degré près autour d'un point qu'on a défini. Avec des choix d'angles «tordus», le temps de calcul n'est cependant pas négligeable.

Rotate	
Clip Degrees	(par degré)
Clip Left	(sélection à droite)
Clip Right	(sélection à gauche)
Clip 180	(demi-tour)
Page Left	(page à gauche)
Page Right	(page à droite)

• Clip Degree place un réticule en plein centre de la boîte de sélection. Un cliqué-tiré à l'intersection des deux droites déplace l'axe de rotation sur la page ([Home] replace les axes au centre de la boîte, [F1] les replace à leur position précédente, [F2] les met au centre de l'écran) ; un cliqué-tiré sur une des droites provoque le basculement du cadre virtuel de sélection : la valeur de la rotation s'affiche en degrés en haut d'écran ; les flèches autorisent un ajustage au degré près et même au 1/10e de degré si la touche [Shift] est enfoncée simultanément. On a ensuite le choix d'effectuer une rotation simple ou une recopie (vérifier alors le mode de pose : Transparent, XOR...).

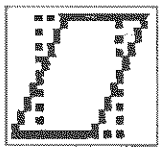


Le drame se prépare.



«Où est le frein ?!»

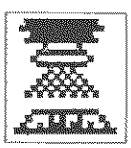
- Page Left et Page Right proposent de basculer soit la totalité de la page (Full Page) dont les dimensions changent, soit uniquement la zone de dessin (**Image Area**).



L'inclinaison

Elle peut s'effectuer soit horizontalement, soit verticalement. Un cliqué-tiré-droit déforme la boîte et un cliqué-gauche fixe l'image.





Les effets spéciaux

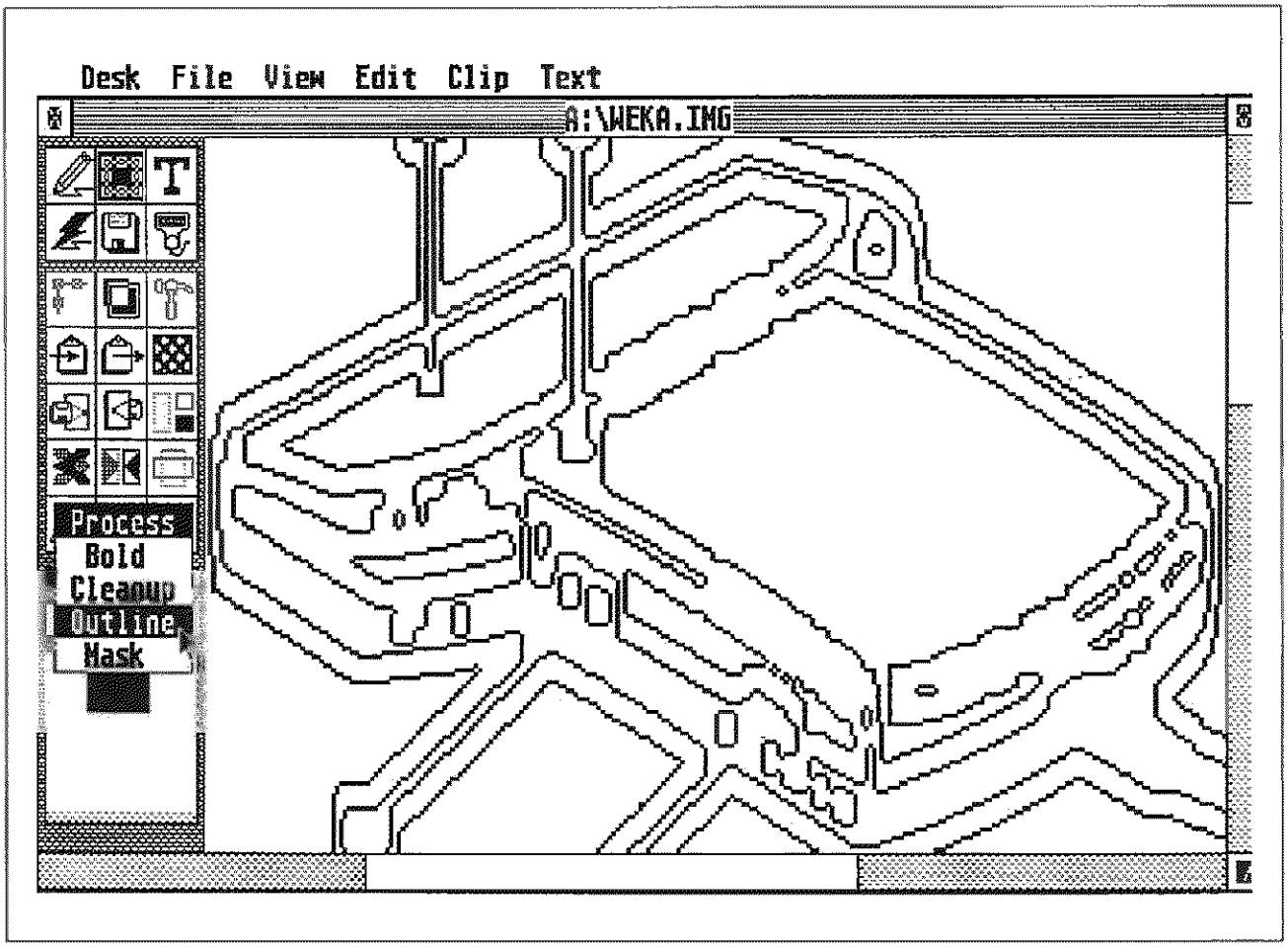
En plus de leur aspect pratique de traitement automatique des images, les outils disponibles ici permettent d'obtenir des effets graphiques parfois imprévus, souvent originaux ou esthétiques.

Process	
Bold	(gras)
Cleanup	(nettoyage)
Outline	(détourage)
Mask	(masquage)

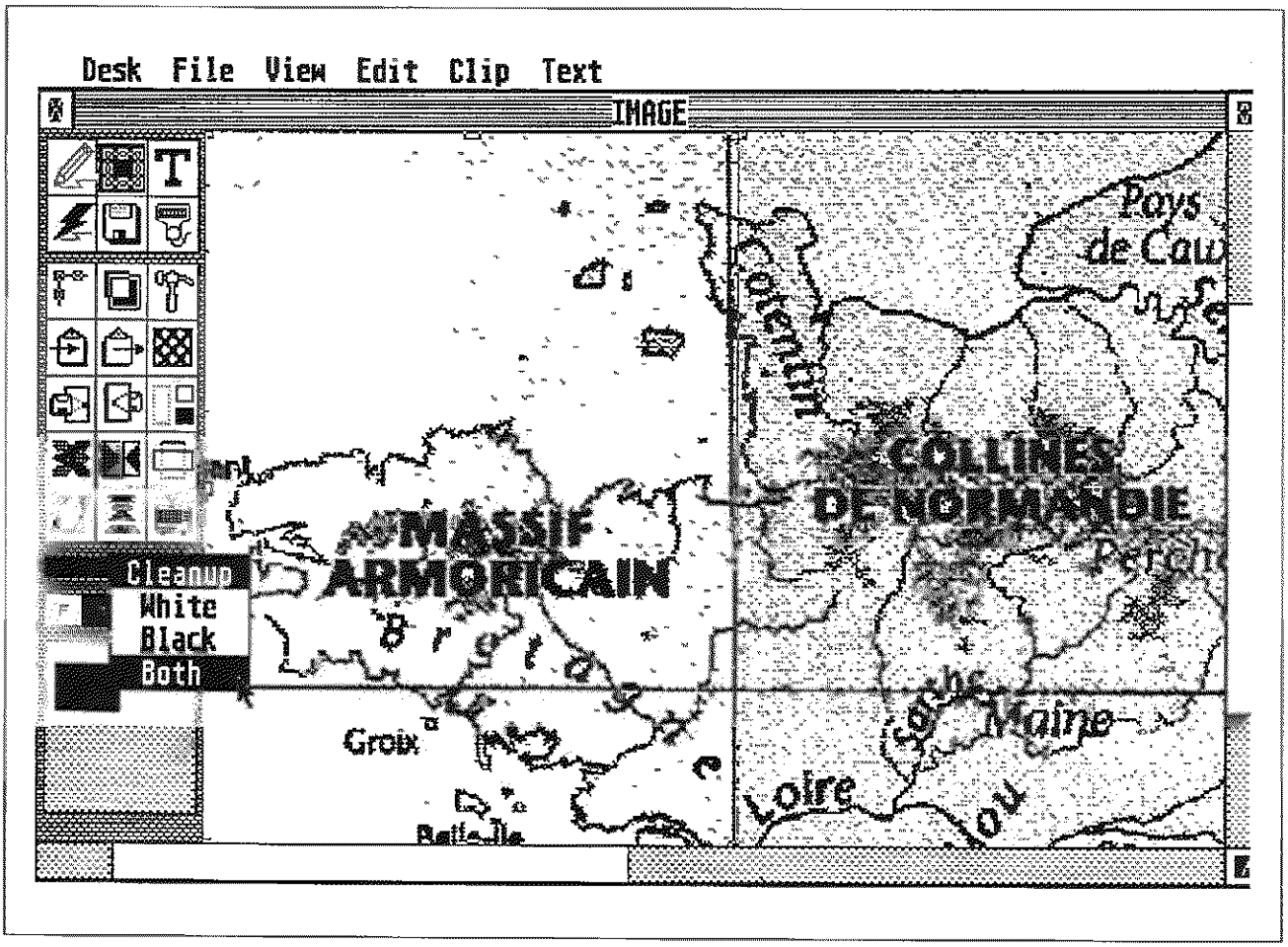


• Bold épaissit les traits en ajoutant des pixels au choix : soit en haut et en bas, soit à droite et à gauche, soit dans toutes les directions.

• Outline vide les surfaces pleines et ne conserve que leur contour : on peut ainsi vider une «silhouette» qui apparaît en aplat noir pour la recolorer. Dans notre exemple, nous avons repris le dessin du scanneur sur lequel nous avons appliqué une succession de mises en gras suivies de détourages afin de parvenir à un objet pratiquement méconnaissable.



• Cleanup supprime au choix tous les points noirs ou tous les points blancs isolés. Très utile pour rendre sa lisibilité à une image venant du scanneur comme on peut le voir sur cette carte de France dont seule la partie gauche a subit le traitement.

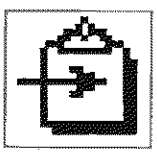


Efficace nettoyage sur la Bretagne.

- Mask permet d'éclaircir la zone d'écran située sous la boîte de sélection. On choisit d'abord une trame grise proportionnelle au degré d'affaiblissement souhaité, puis on choisit un masque blanc ou un masque noir (dans le cas du masque noir, le motif est inversé avant d'être appliqué).

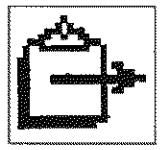
Le fonctionnement s'apparente à l'utilisation d'un ET logique sur le dessin : si un point noir de la trame rencontre un point noir du dessin, ce dernier est mis à blanc.

Les fonctions disque



Vers le presse-papier

La zone sélectionnée est sauvée dans un fichier nommé CLIPBOARD.TMP. Le presse-papier ne contient qu'une image à la fois et chaque sauvegarde détruit la précédente.



Coller depuis le presse-papier

Le sélecteur qui apparaît sur la page est de la taille de l'image sauvée dans le presse-papier. On le positionne en déplaçant la souris. On fixe l'image par un clic-gauche.



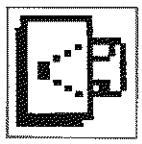
Charger un fichier dans le sélecteur

Load clip
.IMG
Reload
.GEM

Lors du chargement d'un fichier .IMG ou .GEM on peut décider de conserver les proportions du dessin originel (il est probable qu'une zone restera blanche à droite ou en bas du dessin) ou au contraire de le déformer (ignorer) pour remplir la totalité du sélecteur.

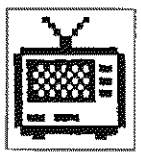
Reload ramène le même fichier .IMG dans les mêmes conditions (duplication rapide d'un objet sauvé sur disque à différents endroits de la page).

Le chargement d'un dessin vectorisé de suffixe .GEM produit une image point par point ; les textes éventuellement présents dans le fichier de départ sont ignorés.



Sauvegarder le contenu du sélecteur

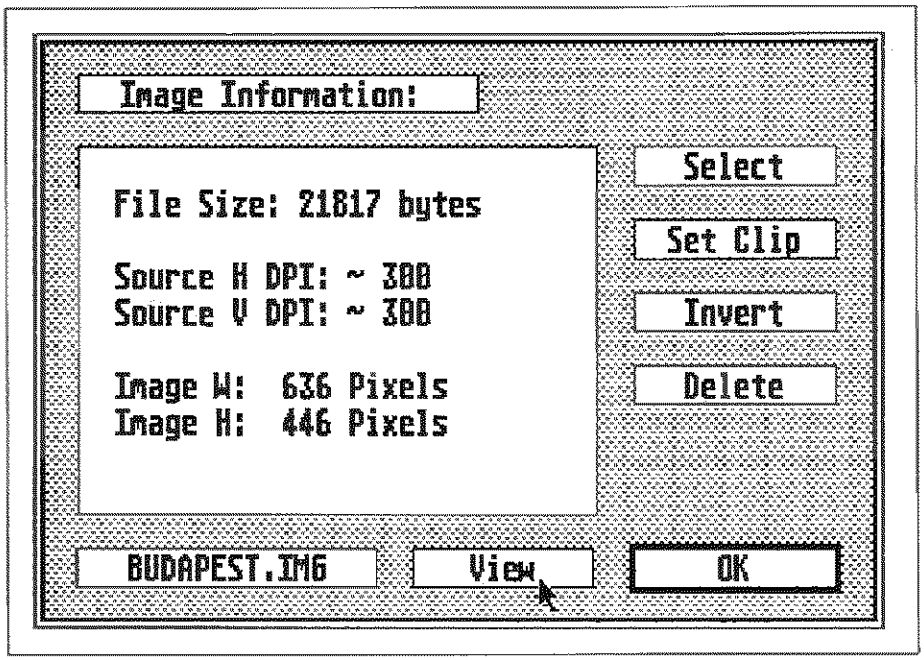
Même principe que pour le presse-papier, mais ici on nomme le fichier qui portera le suffixe .IMG



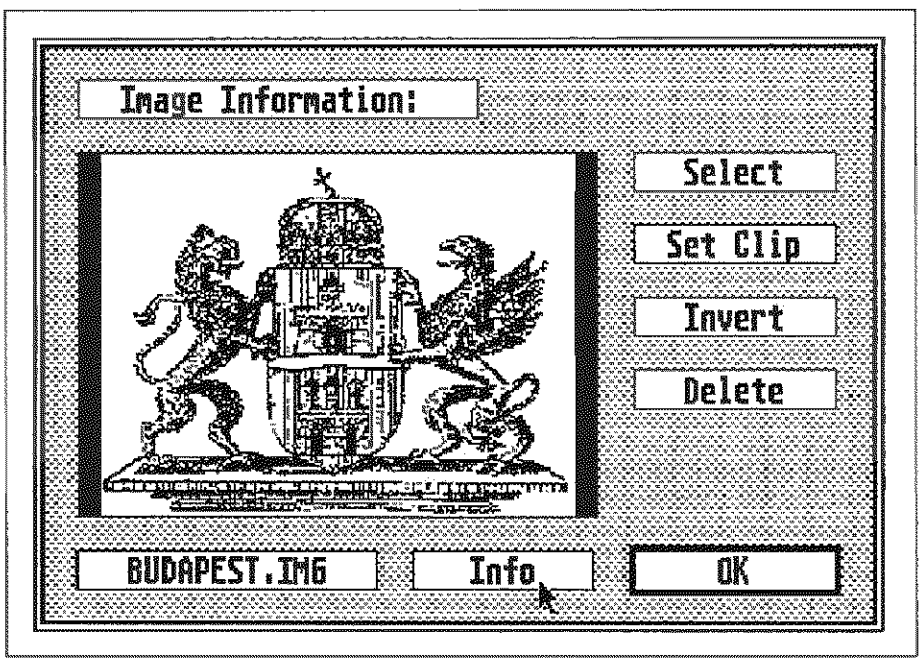
Voir le contenu d'un fichier .IMG

On obtient ainsi diverses informations sur le fichier sans avoir à le charger définitivement : taille en octet, résolution, dimensions. On peut même le visualiser dans une petite fenêtre.

- Set Clip dimensionne la boîte de sélection à la taille exacte de l'image ce qui permet de la charger par la suite à l'intérieur sans déformation.



Caractéristiques du fichier ...



... et rapide coup d'oeil sur son contenu.

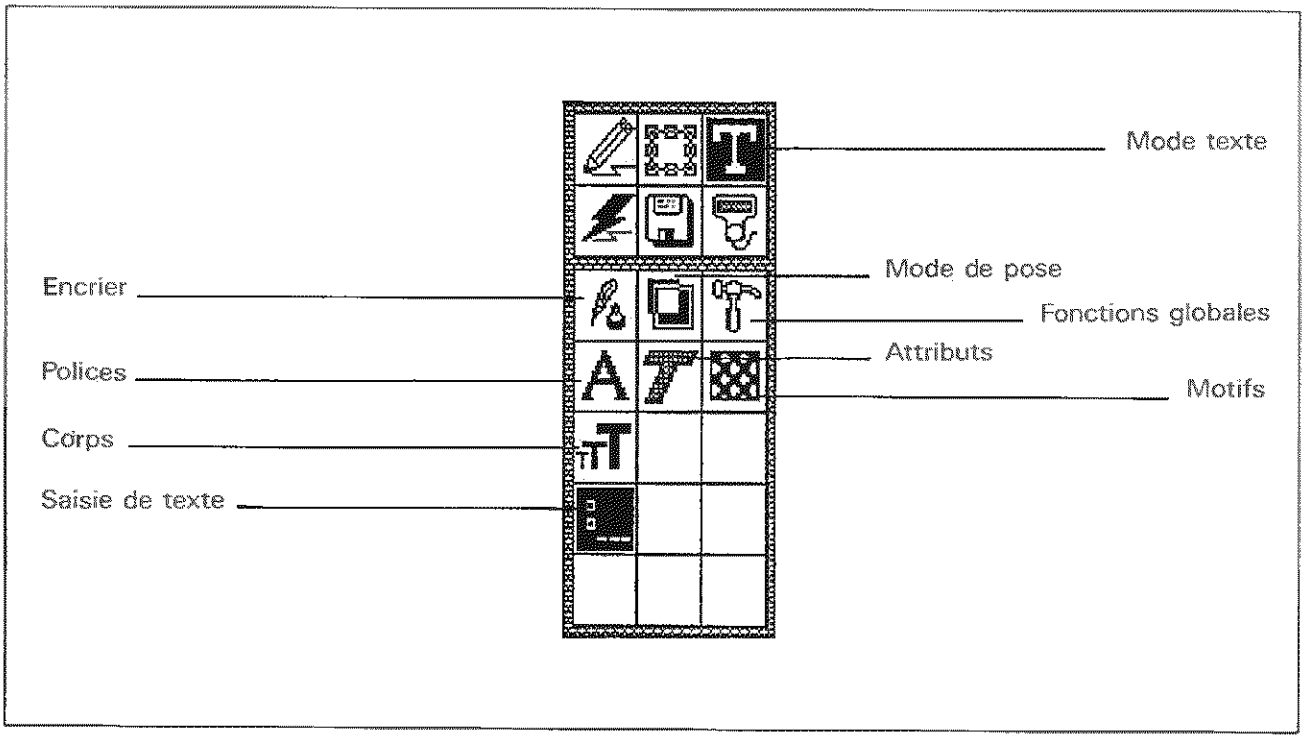
LE MODE TEXTE

La qualité d'un logiciel de dessin peut souvent être évaluée uniquement à partir des possibilités de traitement des caractères qu'il autorise. Avec 10 polices déclinables dans 9 styles sous n'importe quelle taille avec 36



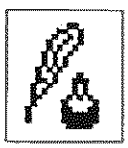
motifs de remplissages différents, Touch-Up est à même de combler le graphiste souvent confronté aux problèmes de lisibilité ou d'impact des affiches qu'on lui demande de réaliser.

Seul problème : les lettres accentuées ne figurent pas dans les polices : il faut placer les accents en les dessinant à la main !



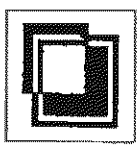
Les fonctions communes aux outils de tracé

Elles ne seront pas à nouveau détaillées ici. Certaines auraient pu être supprimées puisqu'elles sont sans effet sur le mode texte.



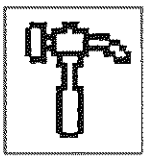
L'encrier

Sans effet ici.



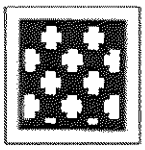
Le mode de pose

Sans effet ici puisque le texte est toujours transparent.

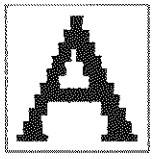


Les fonctions globales

Le menu **Defaults** permet de modifier le chemin d'accès aux polices. C'est utile si l'on a placé un répertoire des polices de nom différent sur la disquette destinée à recevoir les fichiers de sauvegarde des images.

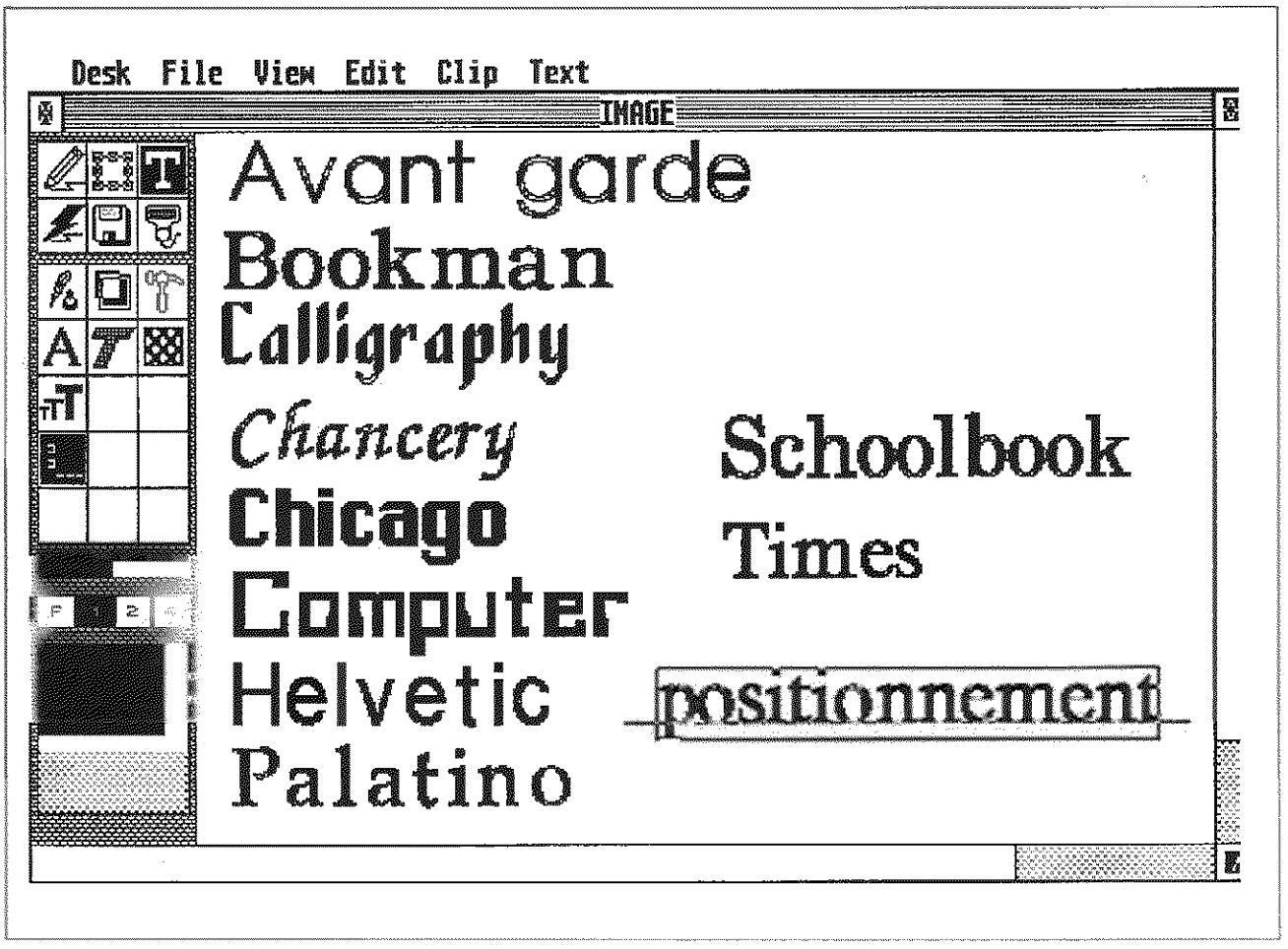


Les motifs

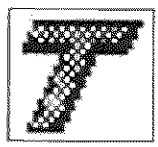


Les polices

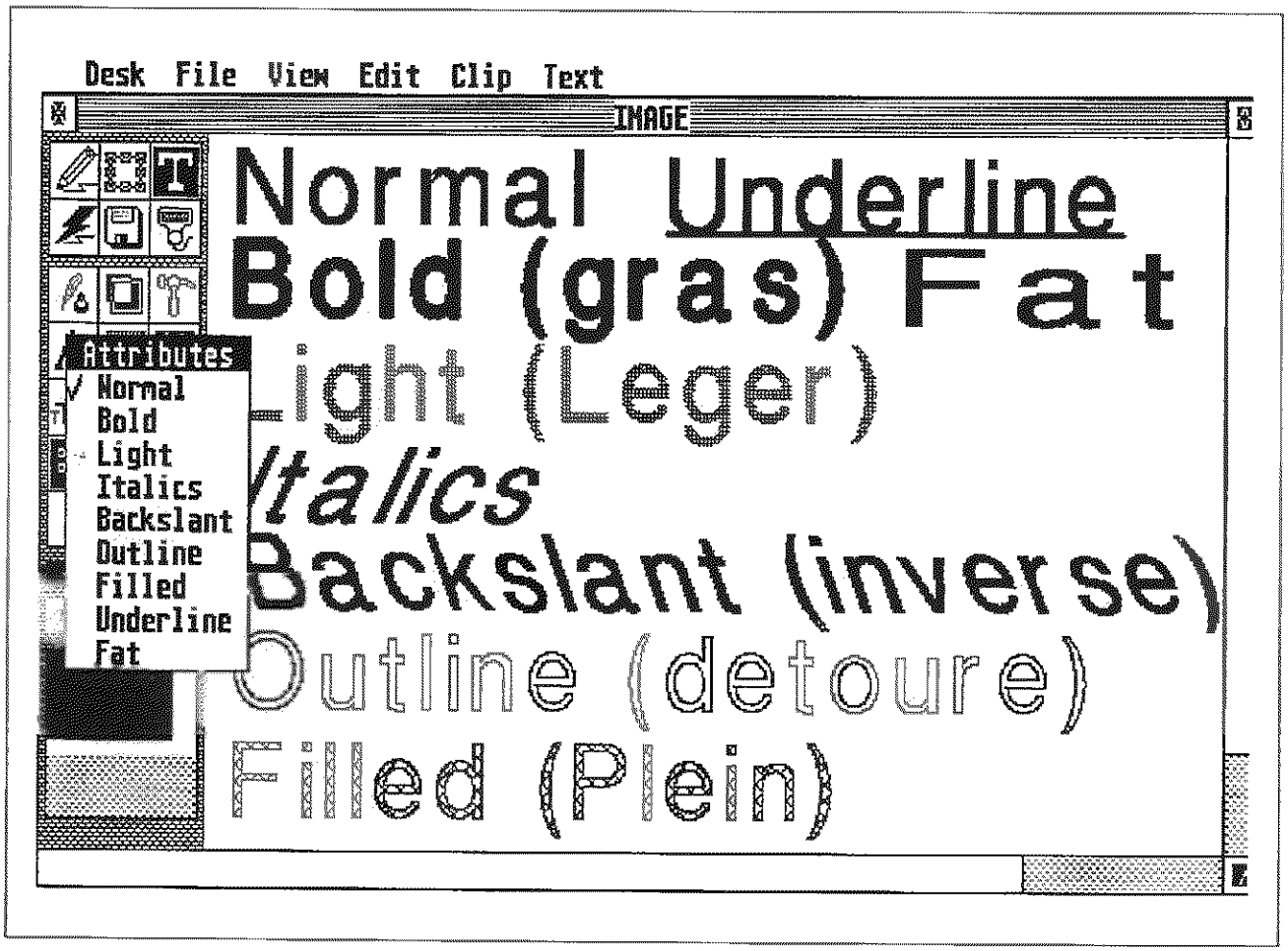
Le descripteur des polices est constitué lors du chargement du logiciel.



Dix polices à disposition ...



Les attributs



... conjuguées dans neuf styles.

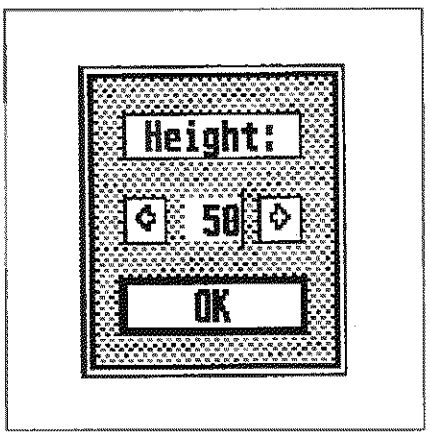
La plupart des différents attributs peuvent se cumuler (gras + italique + souligné + détouré par exemple).

Quelques particularités :

- **Backslant** est un italique inversé.
- **Filled** permet le remplissage des caractères à l'aide d'un des 36 motifs.
- **Fat** (gros) double la largeur des caractères.

Le corps

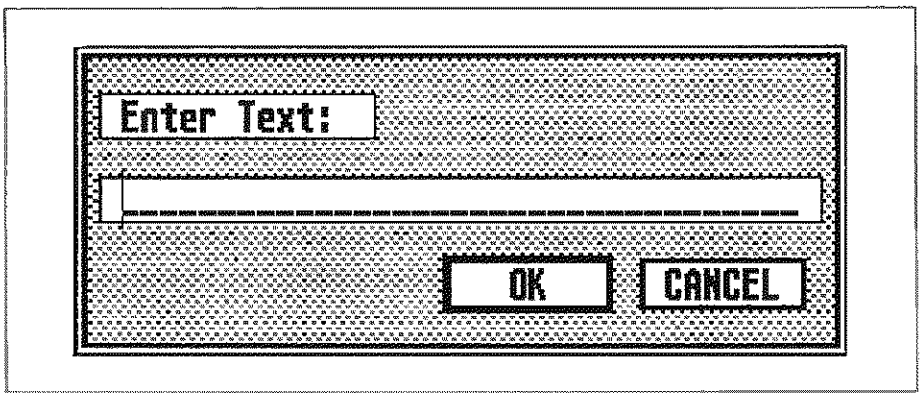
On peut entrer dans la boîte de saisie n'importe quelle valeur de 1 à 999 points. Les corps inférieurs à 30 points sont affichés en définition fine, les autres en définition épaisse.



Si l'on clique sur les flèches de modification, [Shift] incrémente ou décrémente la valeur par dizaine, [Alt] par centaine.

La saisie de texte

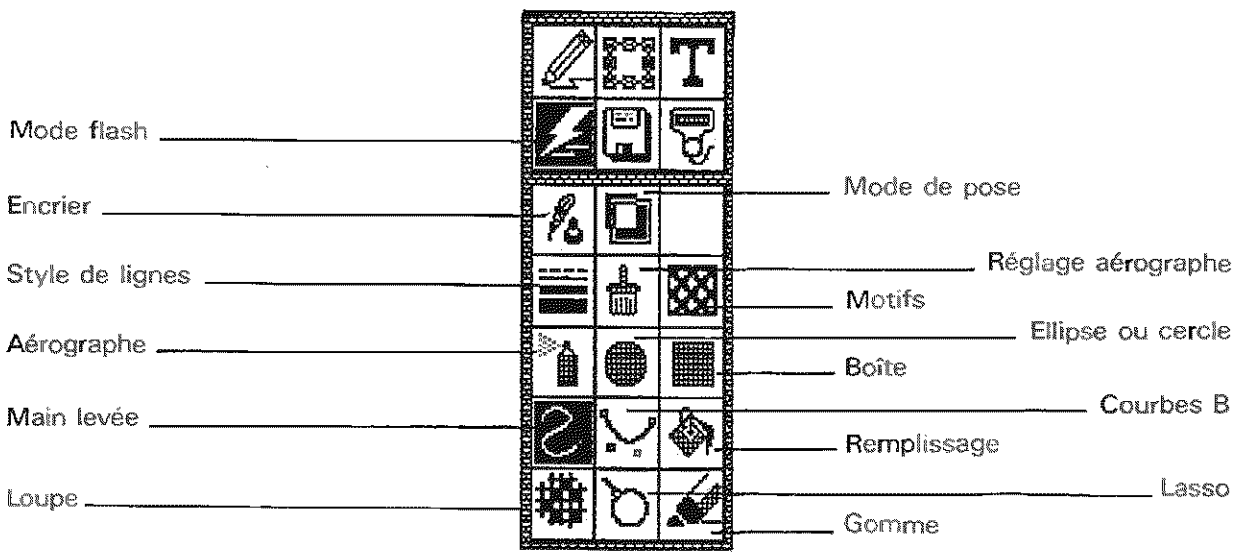
Le texte de 35 caractères maximum est saisi dans une boîte d'édition. La boîte mémorise le texte jusqu'à la saisie suivante.



Le Positionnement

Lorsqu'on valide la boîte d'édition, un rectangle de positionnement apparaît à l'écran : on le place à l'aide de la souris et on le fixe par un clic-droit. Un clic gauche annule le rectangle et permet de revenir au menu.

LE MODE FLASH



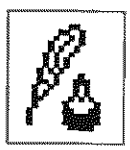
L'ensemble des outils de ce mode permettent un travail limité à la zone affichée sur l'écran (on peut cependant accéder à d'autres zones de la feuille grâce au **Repositionneur**). Les ascenseurs et la ligne supérieure des menus disparaissent augmentant légèrement la surface de travail.

L'intérêt de ce mode vient du fait qu'à chaque opération, la page précédente est mémorisée : en cas d'erreur, la touche **[Undo]** restitue le dessin à l'étape précédente (une seule étape est mémorisée, un second appui sur **[Undo]** ne change rien!). Une «fuite» de remplissage, par exemple, n'affecte que l'écran et non pas la totalité de la page et un appui sur **[Undo]** restitue le dessin initial.

En plus d'une rapidité accrue, on dispose d'outils nouveaux tels que la loupe, le lasso, l'aérographe, une vraie gomme, une infinité de brosses...

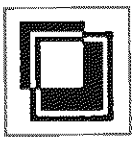
Les fonctions communes aux outils de tracé

Elles ne seront pas à nouveau détaillées ici.



L'encrier

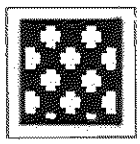
L'option **Opposite** n'est pas active puisqu'on ne possède pas de crayon en mode **Flash**.



Le mode de pose



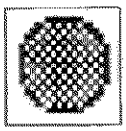
Les styles de lignes



Les motifs

Les fonctions ayant subi quelques variantes

Ce sont les outils géométriques de base qui peuvent servir de brosse. Une fois définie (taille, motif de remplissage...) la brosse est mémorisée jusqu'à ce qu'on en fabrique une nouvelle. [ESC] annule la brosse personnalisée et restitue les paramètres par défaut.



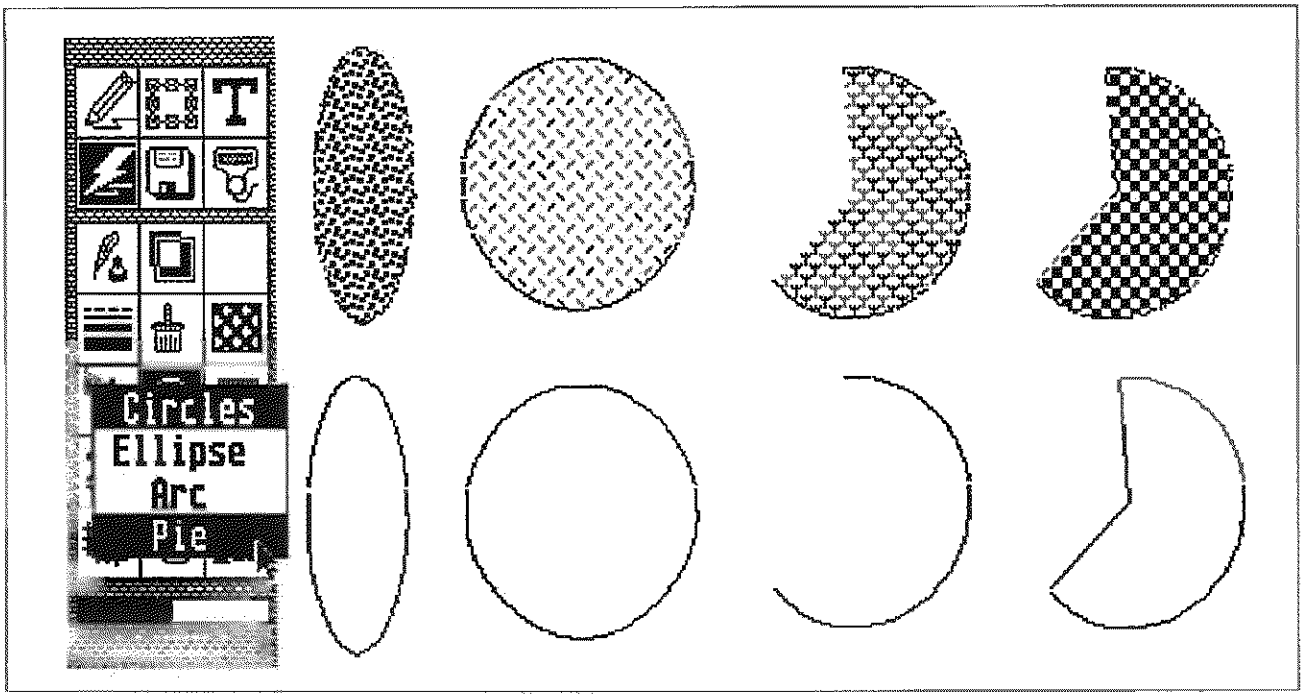
L'ellipse ou cercle

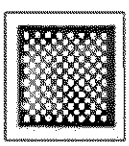
Cette icône ouvre un menu de sélection donnant accès à trois formes différentes.

[Shift] cliqué-tiré modifie la taille de l'objet.

[Alt] force l'ellipse (ou l'arc) en cercle.

Arc et Pie peuvent voir leur angle modifié grâce aux flèches du clavier. La différence entre Arc et Pie est visible lorsqu'on n'utilise pas de motif de remplissage.



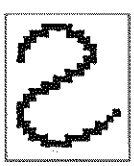


La boîte

[Shift] cliqué-tiré modifie sa taille.

[Shift][Alt] cliqué-tiré force une boîte carrée.

[Alt][R] arrondit les coins de la boîte tant qu'elle n'est pas fixée. Un nouvel appui sur [Alt][R] annule l'arrondi.



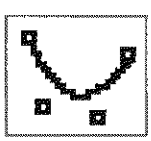
La «main levée»

On a le choix entre «dessin à main levée» (Sketch) et «polygone ou ligne brisée» (Polyline). Dans les deux cas, la taille maximale est de 128 points ou 127 segments.

La ligne brisée subit les mêmes contraintes de réalisation que le polygone vu dans le mode dessin.

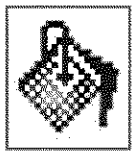
Un clic-droit en fin de création de l'objet (double-clic-droit dans le cas d'une ligne brisée) fait apparaître un carré noir de sélection. On peut alors commencer à peindre avec le nouvel objet par un cliqué-tiré-gauche.

[ESC] annule l'objet courant et permet la réalisation d'un autre.



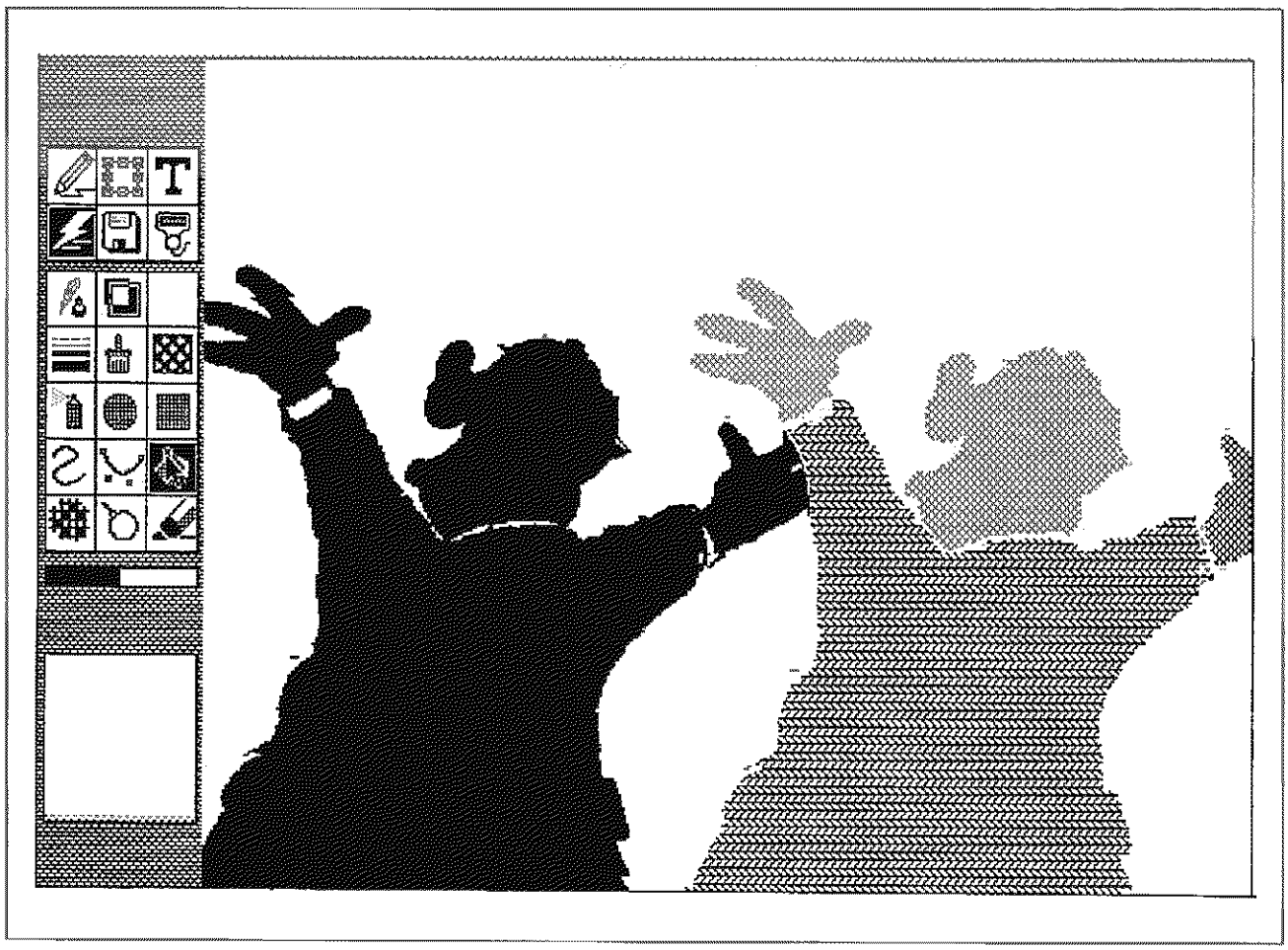
Courbes B et courbes de Bézier

La construction s'effectue comme dans le menu des outils de dessin. L'objet créé est ensuite transformé en brosse.



Le remplissage

L'action du remplissage est limitée à la portion de dessin présente à l'écran. Le remplissage s'arrête lorsque le motif rencontre un point de couleur opposée au point de départ, ce qui permet de modifier le remplissage de zones en aplat noir.



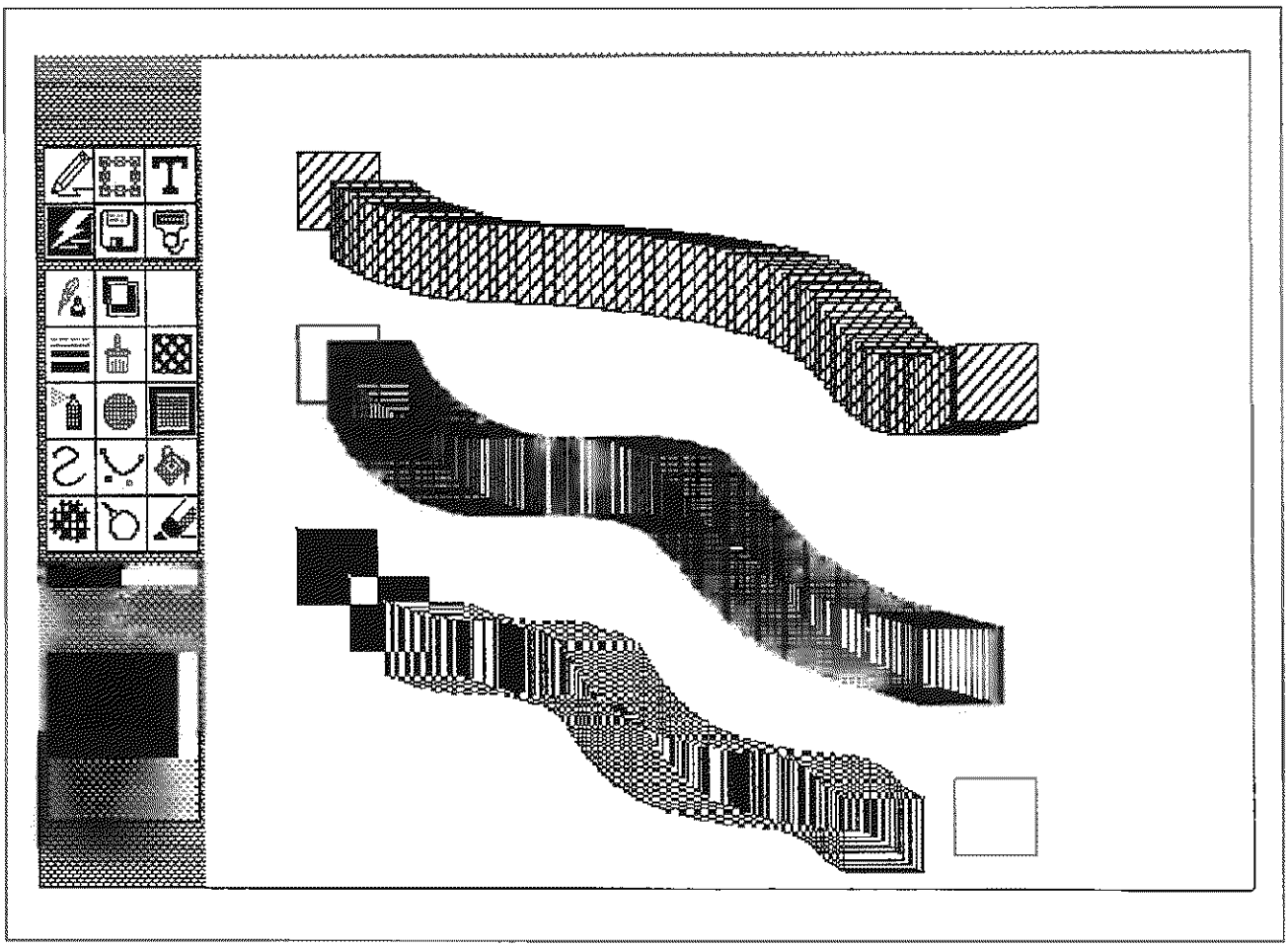
Remplissage d'une zone noire par un motif.

Les fonctions nouvelles

Les brosses

N'importe quel objet prédéterminé (boîte, cercle...) ou créé par l'utilisateur (courbe, dessin à main levée...) peut être transformé en brosse.

On sélectionne l'objet, on modifie éventuellement sa taille par un **[Shift]** cliqué-tiré s'il s'agit d'une forme prédéterminée. Par la suite, un simple clic fixe la forme à l'écran alors qu'un cliqué-tiré permet de «peindre» avec. En fonction du mode de pose choisi on obtient des effets différents.



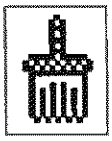
Brosse avec Replace, Transparent et XOR.

L'annulation ([Undo])

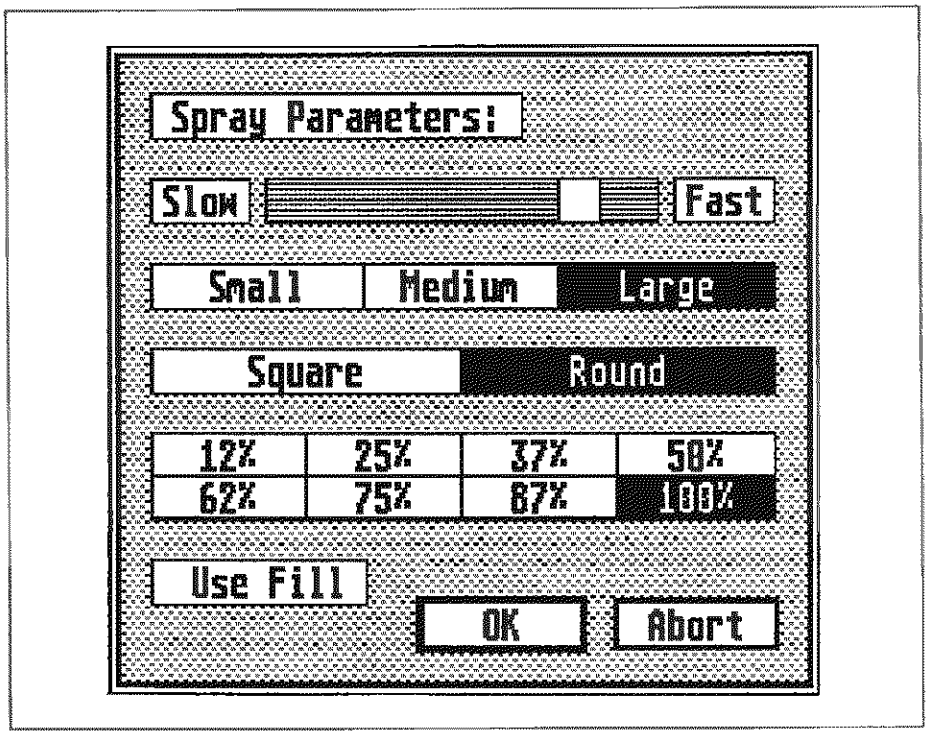
[Undo] annule la dernière modification réalisée.

[Shift][Undo] annule tout le travail réalisé en mode **flash**.

[F10] fixe définitivement les informations sur la page (**[Undo]** et **[Shift][Undo]** sont désormais sans effet pour ces informations qui ne peuvent plus être perdues). Les informations sont également fixées si l'on utilise le **Repositionneur**.



Le réglage de l'aérographe

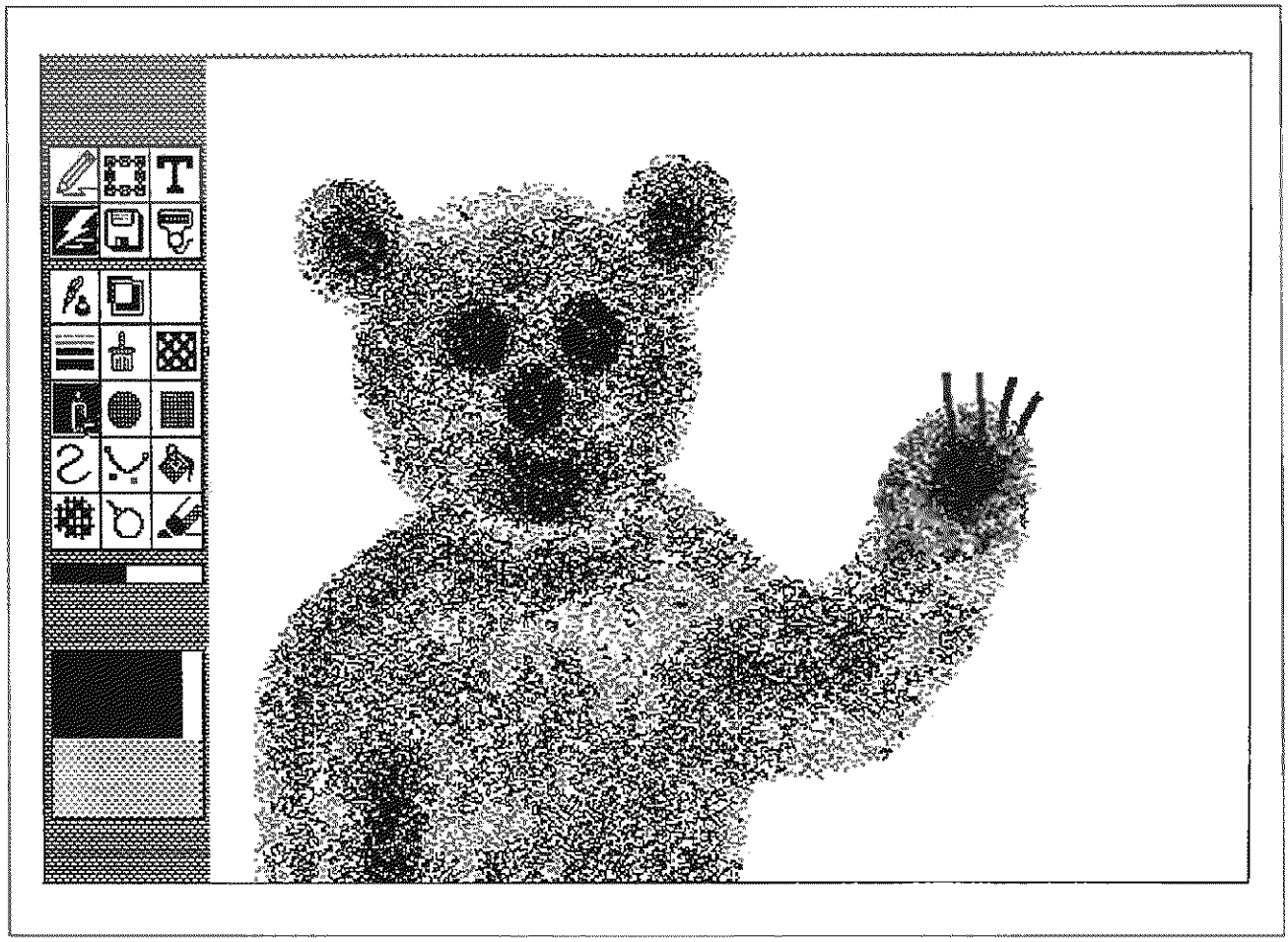


On ajuste le débit de peinture, la taille (petit, moyen ou gros) et la forme du jet (rond ou carré), la saturation maximale (de 17% à 100%). **Use Fill** (utiliser le motif) donne l'équivalent d'un travail au pochoir, le remplissage par défaut constituant le masque.

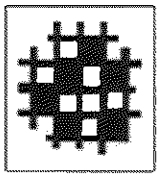


L'aérographe ou la bombe à peinture

Il permet de réaliser des dégradés, de créer des ombres, d'estomper des motifs si l'on utilise une couleur de remplissage blanche. Si l'on clique sur la page sans déplacer la souris, la zone visée se remplit progressivement en quelques secondes jusqu'à la saturation définie dans le menu de réglage. L'aérographe s'utilise plutôt en mouvement, **[Shift]** force la réalisation de traits horizontaux alors que **[Alt]** force les traits verticaux.



L'aérographe permet des créations rapides.



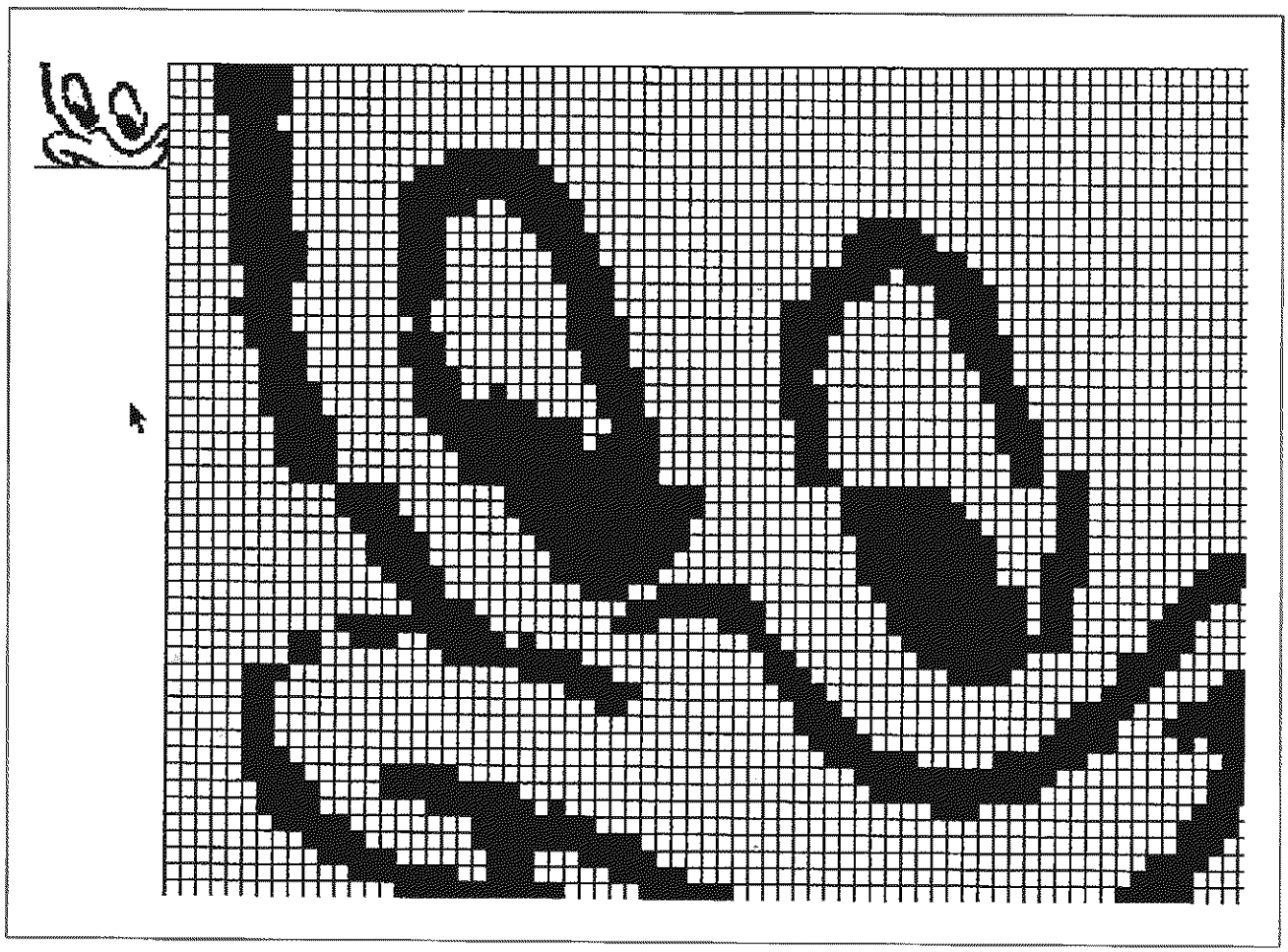
La loupe

On sélectionne l'icône puis on positionne le rectangle correspondant à la loupe sur la partie du dessin à agrandir. On peut alors noircir les points blancs de l'écran et blanchir les points noirs pour faire des modifications de détail. La portion de dessin en taille réelle apparaît dans l'angle supérieur gauche de l'écran.

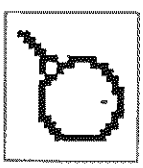
On ne peut malheureusement pas déplacer la loupe sur le dessin sans revenir en vue normale, ni modifier la taille de la zone agrandie.

Un clic droit ramène à la taille normale.

[ESC] également mais sans valider les corrections.



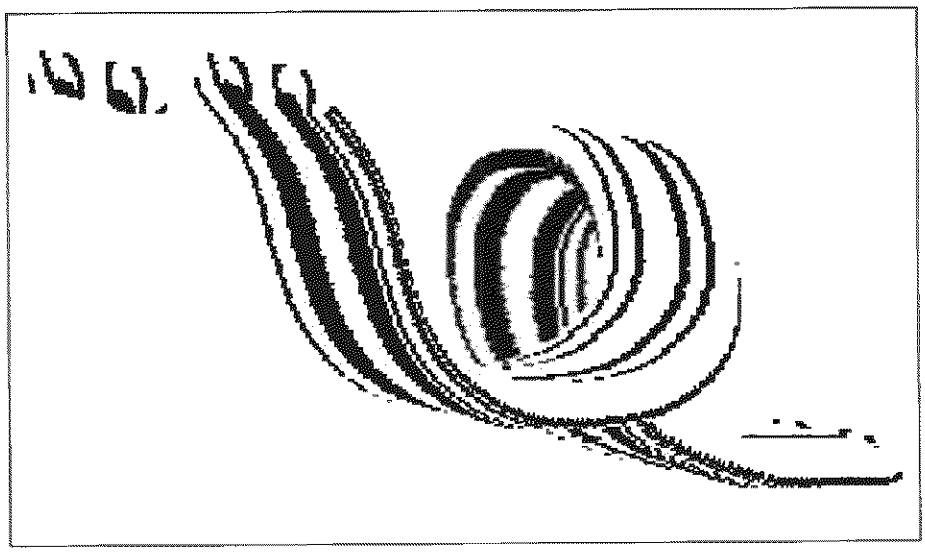
Une loupe puissante pour des yeux fatigués.



Le lasso

On sélectionne une portion d'écran soit à main levé soit à l'intérieur d'un polygone ; le choix dépend de l'outil sélectionné dans le menu Main levée.

La zone ainsi saisie devient une brosse dont les effets dépendent du mode de pose actif.



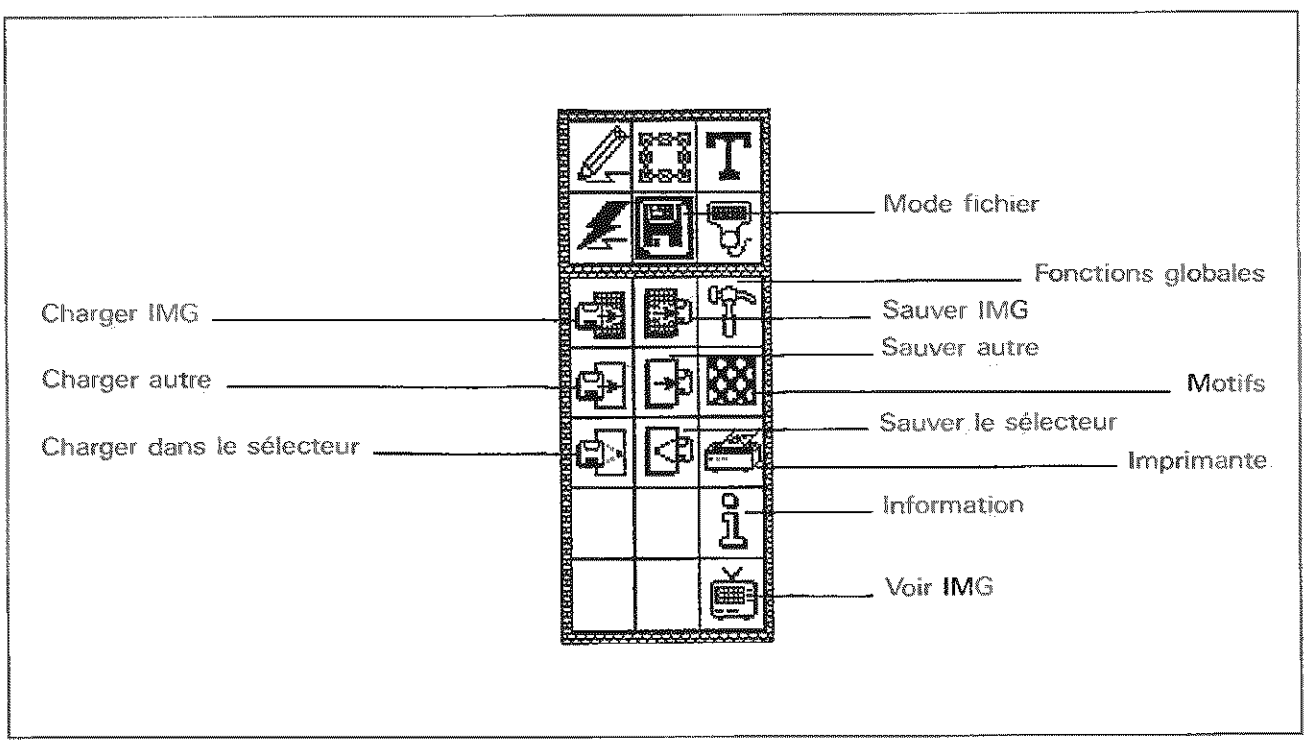
Une brosse attrapée au lasso.



La gomme

Le carré qui apparaît et qui figure la gomme voit sa taille modifiée par un **[Shift]** cliqué-tiré. Elle peut être positionnée n'importe où sur l'écran, puis activée par un cliqué-tiré gauche.

LE MODE FICHIER

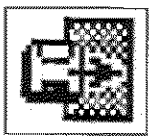


Commandes communes au mode sélecteur et au mode dessin

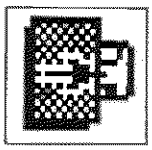
Ces commandes ne seront pas détaillées à nouveau.



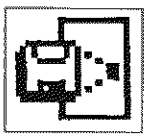
Les fonctions globales



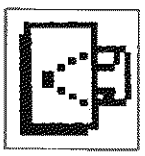
Charger IMG



Sauver IMG

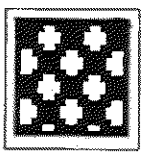


Charger un fichier dans le sélecteur



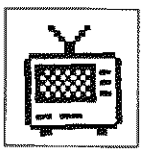
Sauver le contenu du sélecteur

Le suffixe du fichier peut éventuellement être modifié afin d'assurer une exportation (voir les conversions ci-dessous).

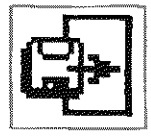


Le motif

Ce menu sert uniquement à définir la résolution souhaitée pour les trames lorsqu'on importe un fichier vectorisé de type .GEM.



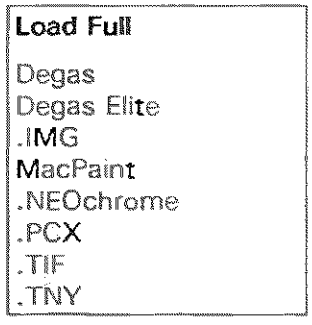
Voir .IMG



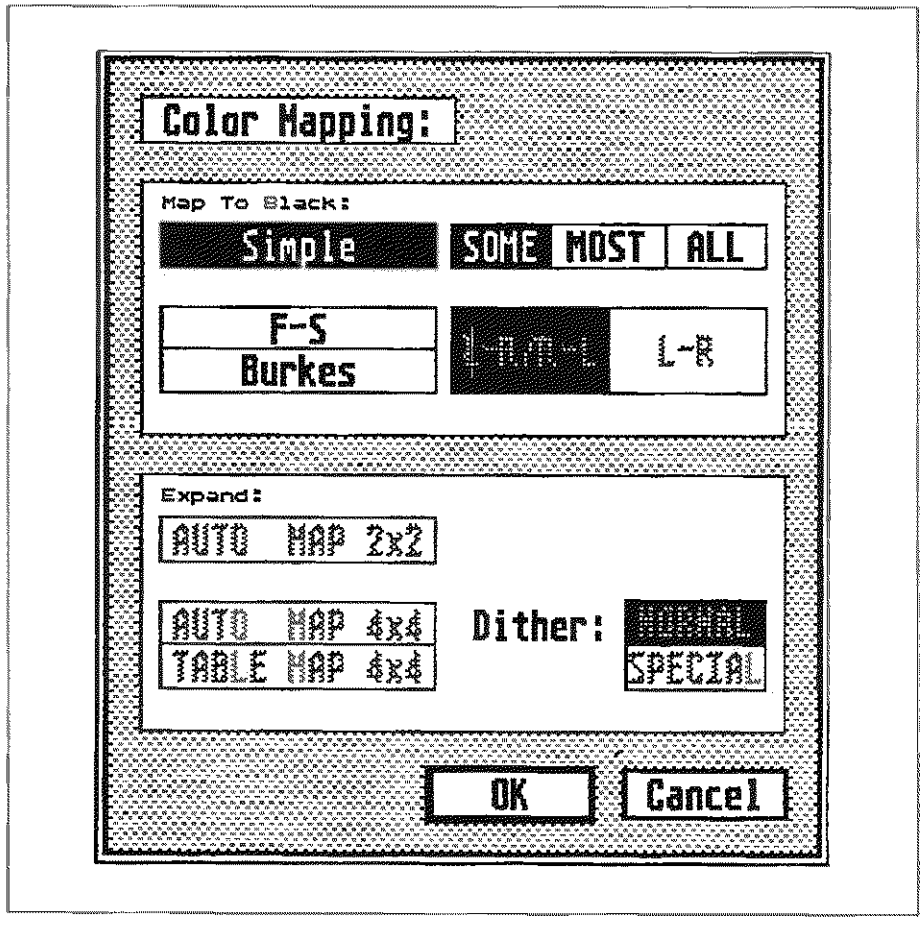
Commandes nouvelles

Charger autre

La liste des différents formats disponibles apparaît.

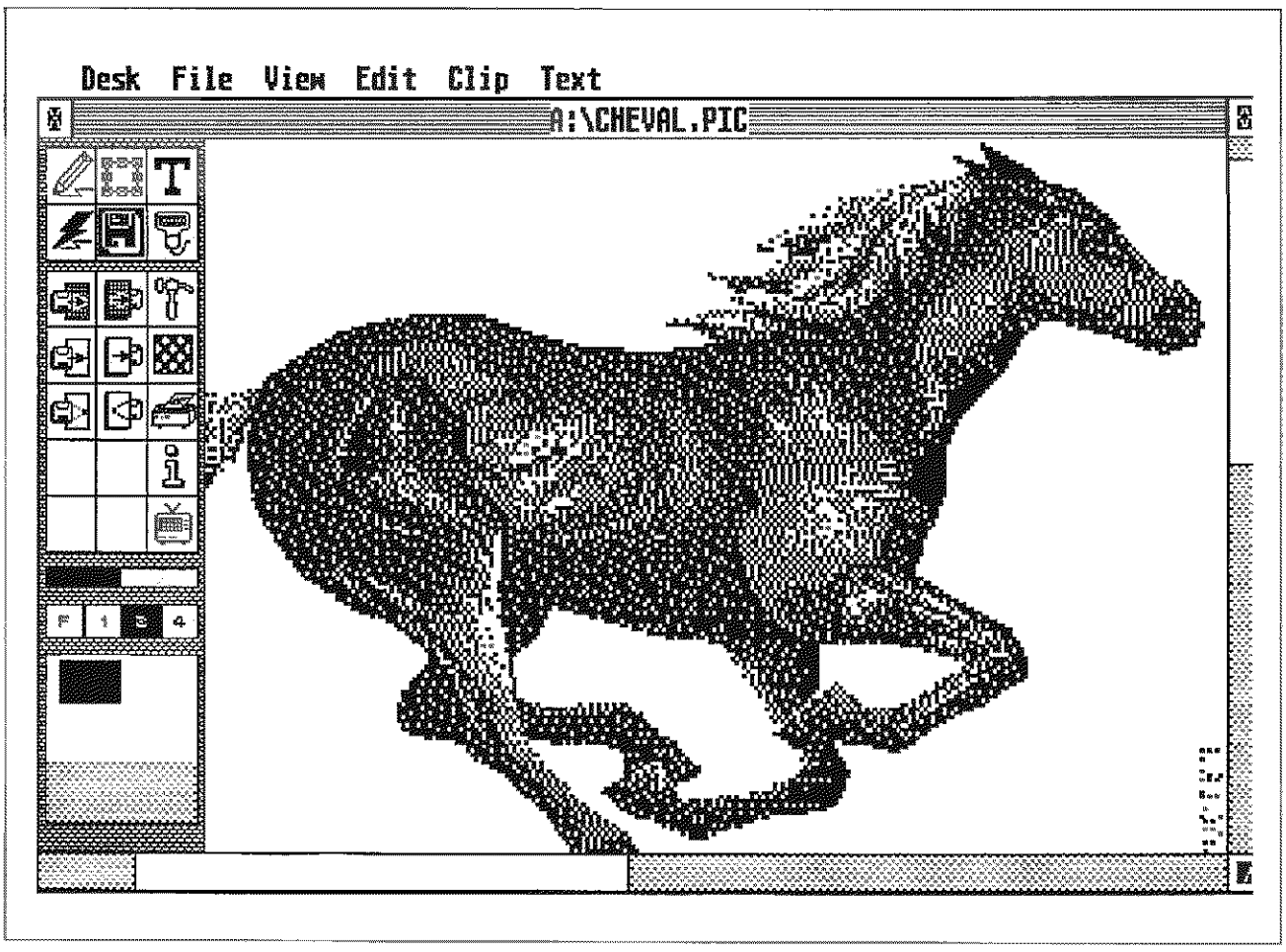


Le chargement d'une image couleur provenant d'un autre logiciel nécessite la conversion des couleurs en trames. Dans ce cas, après avoir sélectionné le fichier, une boîte de saisie apparaît à l'écran.

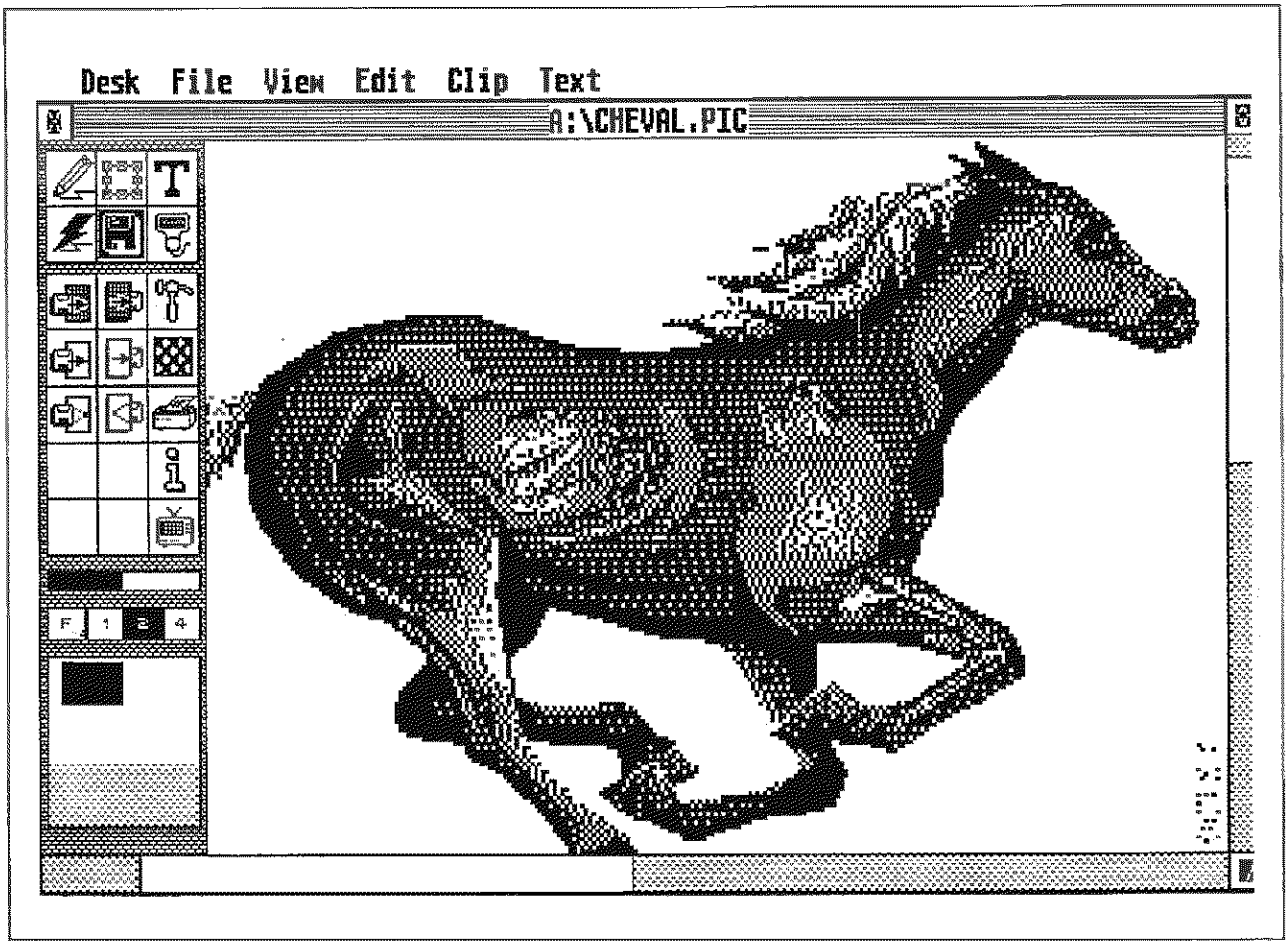


- Simple est une méthode très rapide qui ne restitue que des aplats noirs et propose trois options de réglage : Some, Most et All qui définissent à partir de quel seuil les couleurs sont transformées en noir alors que le reste apparaît en blanc ; Some à partir de 70% de noir (couleurs sombres), Most avec au moins 30% de noir (couleurs moyennes), All transforme toutes les surfaces même très faiblement colorées en aplat noir. Nous n'avons obtenu avec cette méthode que des images grossières, découpées en gros carrés noirs et blancs et parfaitement inexploitables.

- F-S (Floyd-Steinberg) et Burke : ces algorithmes fonctionnent soit de gauche à droite (L-R) soit par aller et retour (L-R/R-L) et transforment les couleurs en textures d'intensités variables.

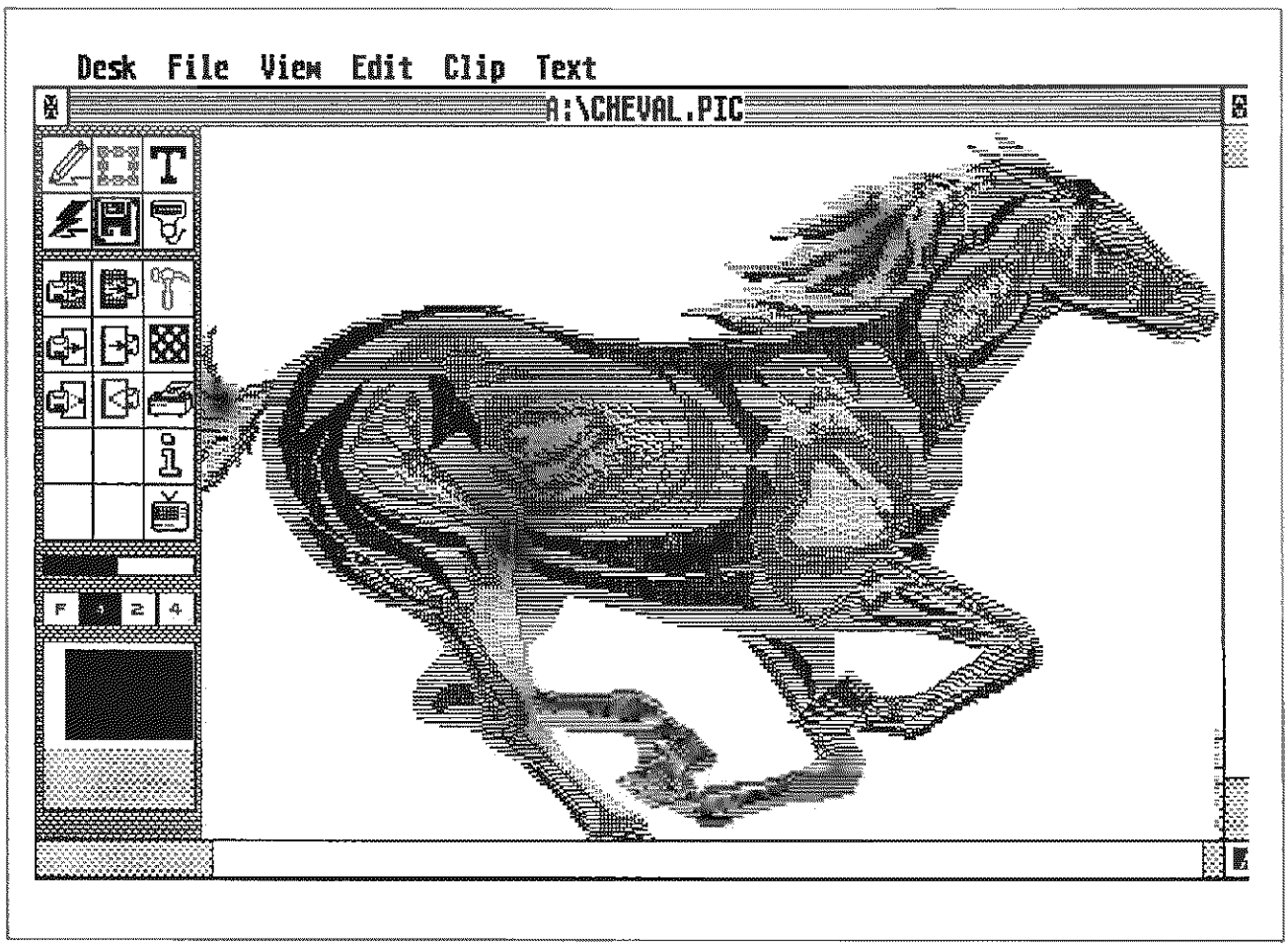


Algorithme de Floy-Steinberg, zoom x 2, fichier néochrome couleur.



Algorithme de Burke, zoom x 2.

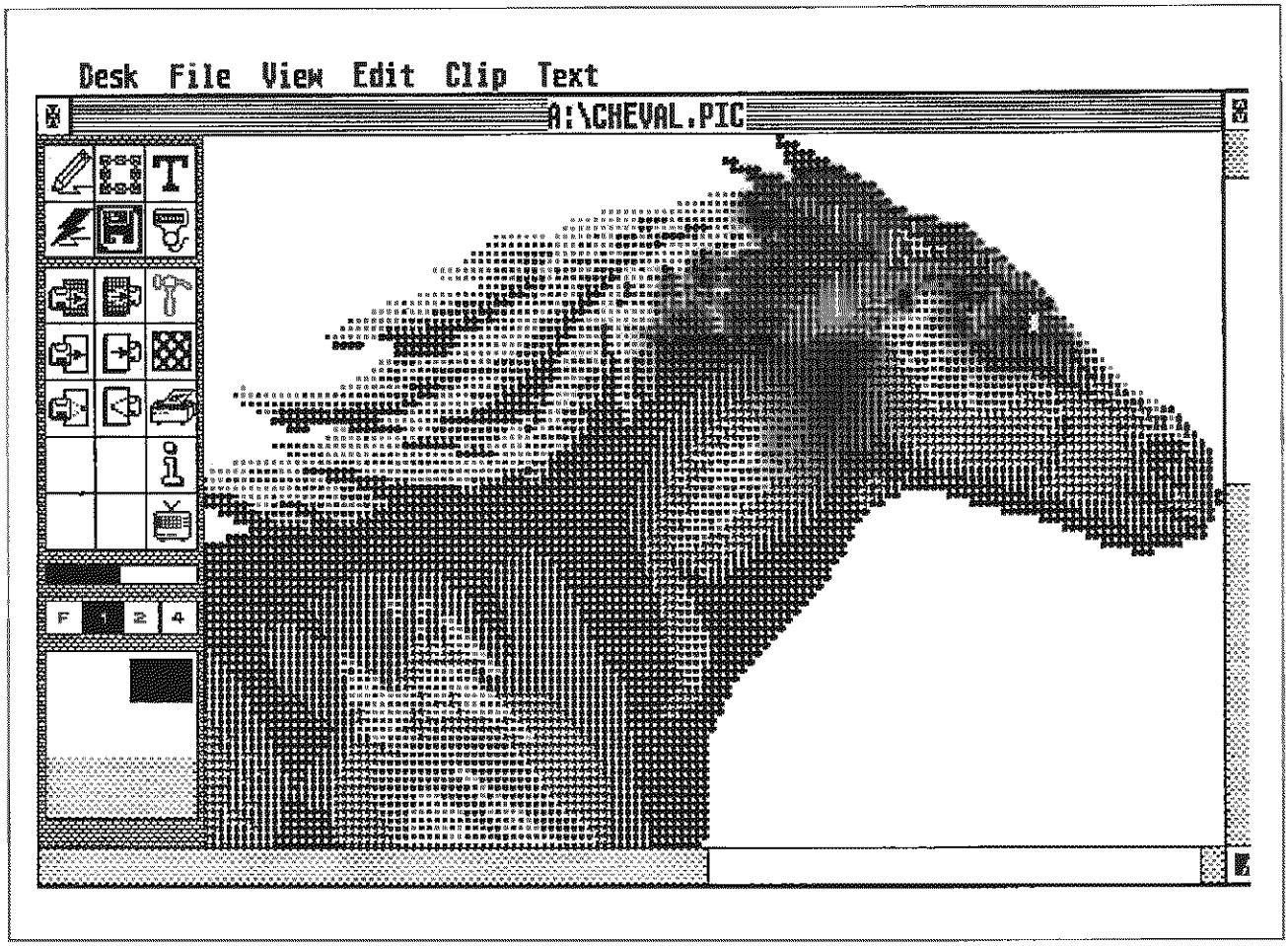
- Auto-Map 2x2 convertit un pixel en une matrice 2x2. L'image résultante occupe donc une surface quadruplée.



Auto-Map 2x2, taille normale.



• Auto-Map 4x4 convertit un pixel en une matrice 4x4. Avec 1 Mo, l'image risque de ne pas tenir en mémoire. Avec une matrice de 4x4 on dispose de nombreuses combinaisons : l'utilisateur a le choix entre Normal et Special qui produisent des effets légèrement différents.



Auto-Map 4x4, taille normale.

• Table Map 4x4 laisse à l'utilisateur le choix des motifs correspondant à chacune des couleurs. Il faut, bien évidemment, utiliser cette fonction sur un moniteur couleur en basse résolution afin de voir les 16 couleurs originelles.

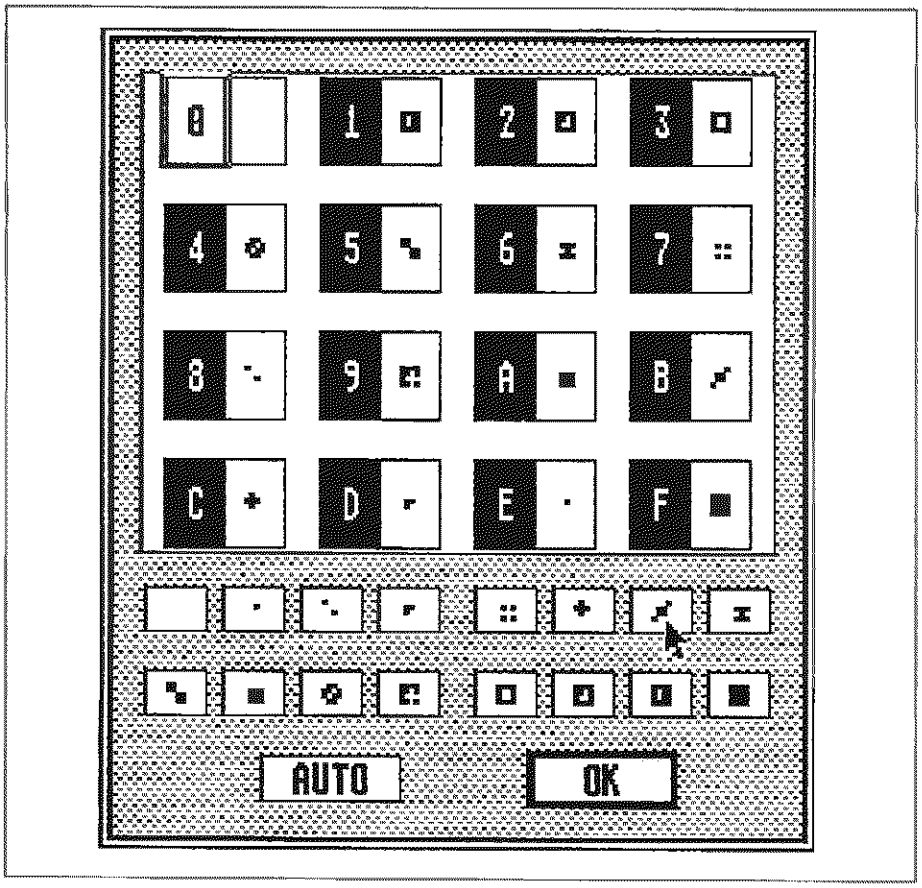
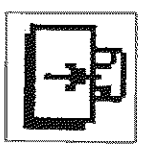


Table à utiliser avec un écran couleur.



Sauver autre

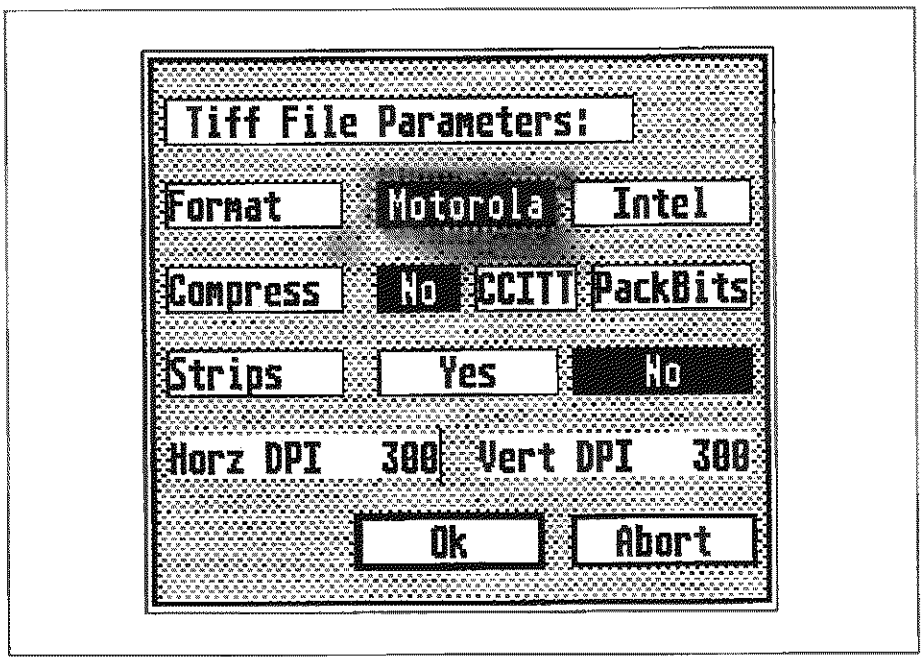
On choisit le format dans la liste qui s'affiche :

Save Full
.GIF
.IMG
.PCX
.TIF
Degas
IFF-ILBM
Mac Paint

- .IMG : au moment de la sauvegarde, **[Alt][P]** sauvegarde en une largeur et **[Alt][M]** en trois.
- .PCX : si l'on appuie sur **[Shift]** pendant la sauvegarde, les dimensions de l'image sont modifiées pour être multiples de 16 pixels.



• .TIF : on choisit le format Mac ou IBM, la compression ou non, sous forme de bandes verticales ou pas, ainsi que le nombre de points par pouce.



• Degas : le fichier étant de taille fixe, la taille maximale de l'image est de 640x400 (1 écran). L'image est éventuellement tronquée. [ALT][D] place le sélecteur avec une taille de 640x400 sur le dessin, on peut ainsi amener dans l'angle supérieur gauche de la page la partie à conserver.

• MacPaint : le fichier étant également de taille fixe, la taille de l'image est de 586x720. [Alt][Q] place le sélecteur à la taille voulue.



L'imprimante

Une boîte de choix propose soit l'imprimante laser SLM804, soit un périphérique défini sous GDOS qui doit donc être chargé en résidant (par défaut le n° 21). Cette option ne fonctionne correctement que si la mémoire de l'ordinateur est suffisamment vaste pour contenir le tampon destiné à recevoir la page entière avant impression.

Les pilotes disponibles sont :

— 21 : FX80 matricielle en 120x144 ;

— 23 : LQ800 matricielle en 180x180 ;

— 29 : Laser Atari (il est plus simple de choisir SLM804 dans la boîte).

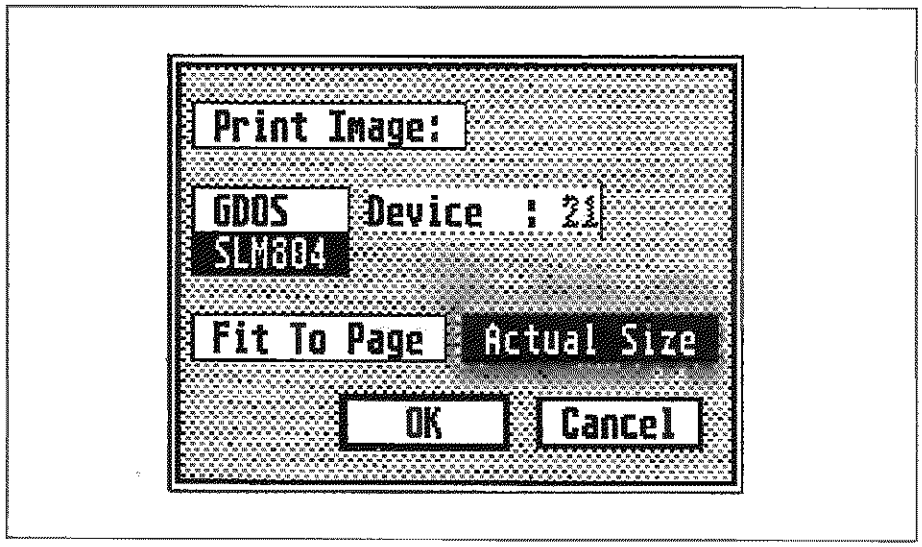
On peut acquérir les pilotes suivants (dommage qu'ils ne soient pas livrés en standard) :

— 24 : Toshiba matricielle 180x180 ;

— 25 : HP Laserjet 150x150 ;

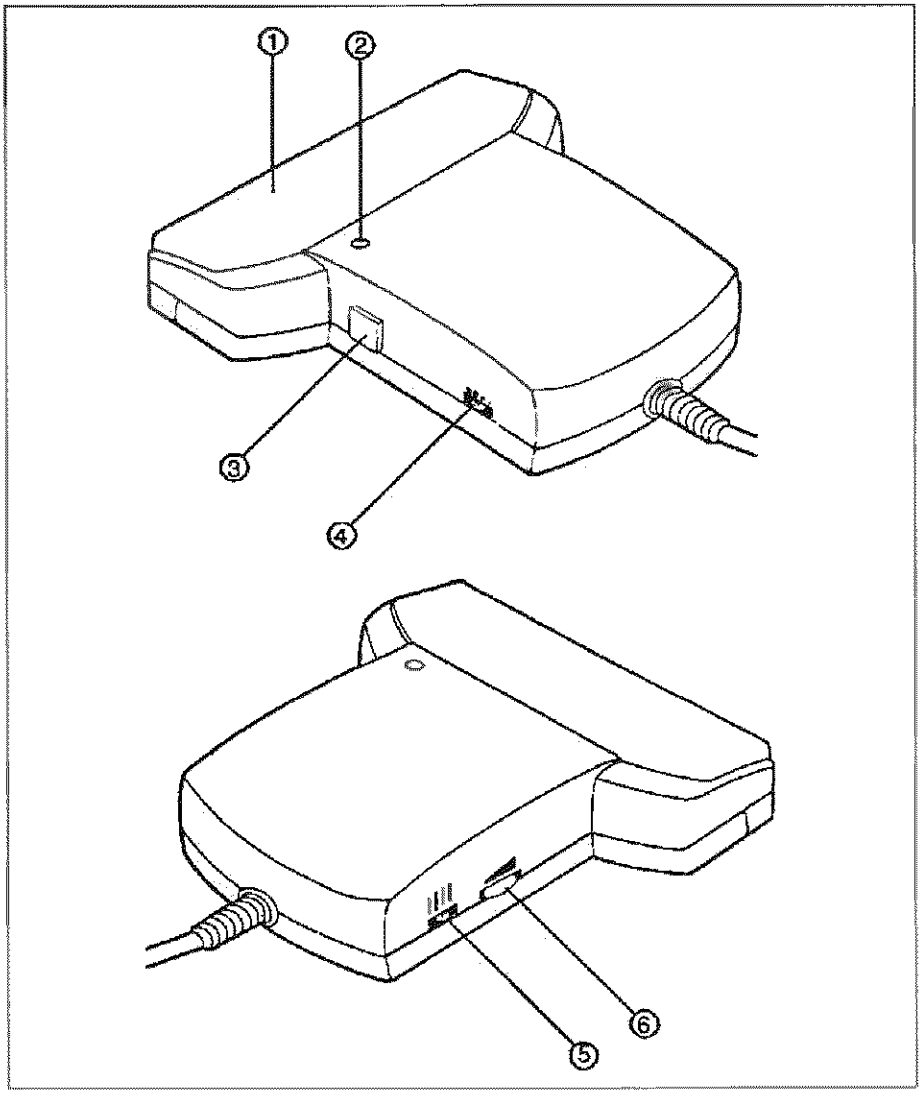
— 27 : HP Laserjet 300x300.

- **Fit to Page** adapte l'image au format de la page.
- **Actual Size** l'imprime dans sa taille actuelle en tronquant éventuellement les éléments qui dépassent.



L'impression requiert le GDOS.

LE MODE SCANNEUR



Le scanneur Golden Image comporte les éléments suivants :

① Fenêtre de balayage qui s'allume au cours de la numérisation et qui laisse voir le document traité.

② Témoin de vitesse de numérisation ; le témoin clignote lorsque la vitesse limite est atteinte et s'éteint lorsqu'elle est trop élevée (dans ce cas la numérisation s'effectue quand même mais des défauts risquent d'apparaître).

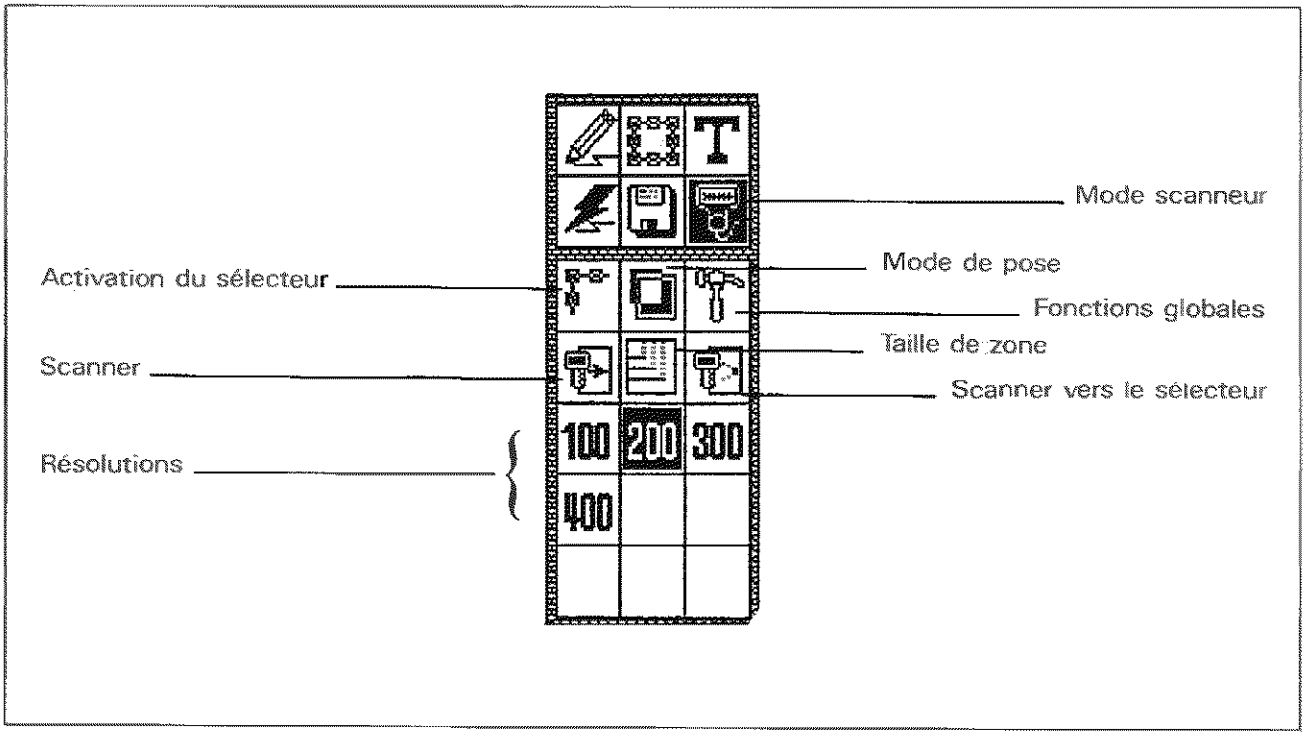
③ Bouton de numérisation qui doit être maintenu pendant tout le temps de l'opération (comme le cliqué-tiré de la souris).

④ Commutateur de dégradé qui permet de sélectionner en mode PHOTO trois tailles de points pour les trames de la plus fine à la plus

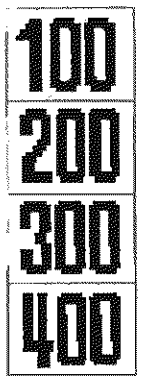
grossière ainsi qu'un mode TEXT qui convient très bien pour les dessins constitués d'aplats noirs comme ceux qu'on trouve dans certaines bandes dessinées.

⑤ Commutateur offrant les résolutions 100, 200, 300 et 400 points par pouce.

⑥ Une molette de luminosité qui modifie le contraste de l'image à l'écran : plusieurs essais sont nécessaires en fonction des documents à reproduire ; généralement, en mode PHOTO, on positionne la molette le plus près possible du blanc alors qu'en mode TEXT les résultats les meilleurs sont obtenus au maximum de noir (aplats parfaitement noirs).



Ce menu ne peut être activé que si un scanneur est connecté sur le port cartouche de l'Atari. On y retrouve l'icône des fonctions globales, la case d'activation du sélecteur ainsi que le mode de pose.



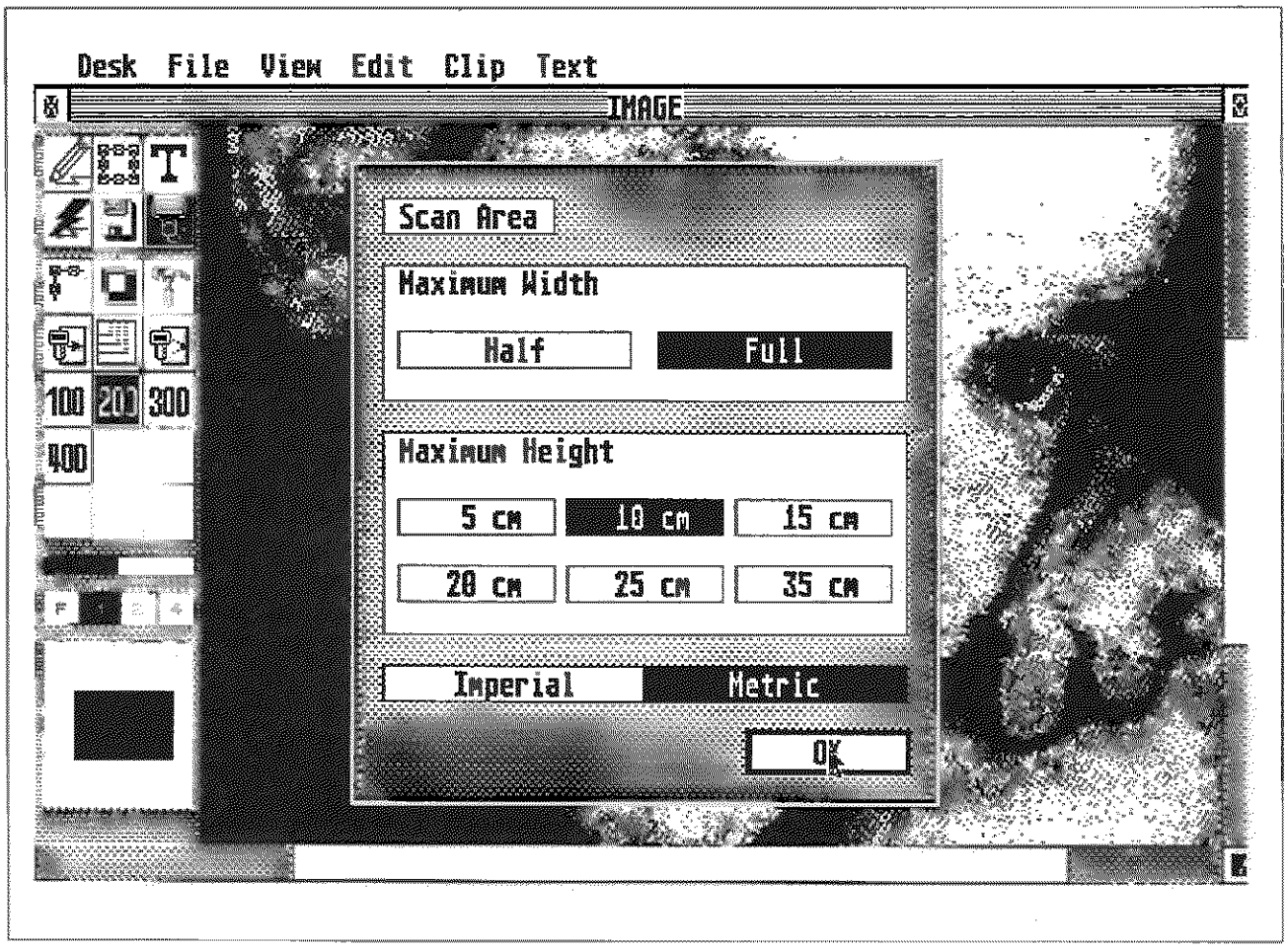
La résolution

On choisit la résolution correspondant à l'interrupteur situé sur le scanneur. Plus la résolution est grande, plus une même image occupe de place, à la fois dans le fichier de sauvegarde et à l'écran. Si la résolution du scanneur correspond exactement à celle de l'imprimante (ex : 300x300 avec une laser), le dessin sera imprimé à la même taille que l'original. Une résolution de 100 points donne de médiocres résultats, surtout avec des documents comportant des niveaux de gris (photos). En 400 points, le grain de l'image est très fin mais celle-ci n'est pas facile à travailler vu sa grande taille (le mode vue globale du zoom donne alors de piètres résultats) et occupe une place énorme sur le disque. Les résultats en 200 points sont satisfaisants et ce format est recommandé avec la plupart des imprimantes à aiguilles.

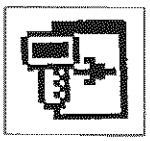


La taille de la zone

La largeur de la zone saisie par le scanneur mesure une dizaine de centimètres. On peut décider de ne saisir que la moitié gauche de la zone. La longueur peut être ajustée entre 5 et 35 cm : on s'approchera au plus près de la longueur du document à reproduire.

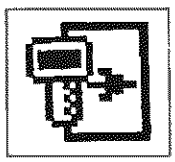


Dimension d'un document à numériser.



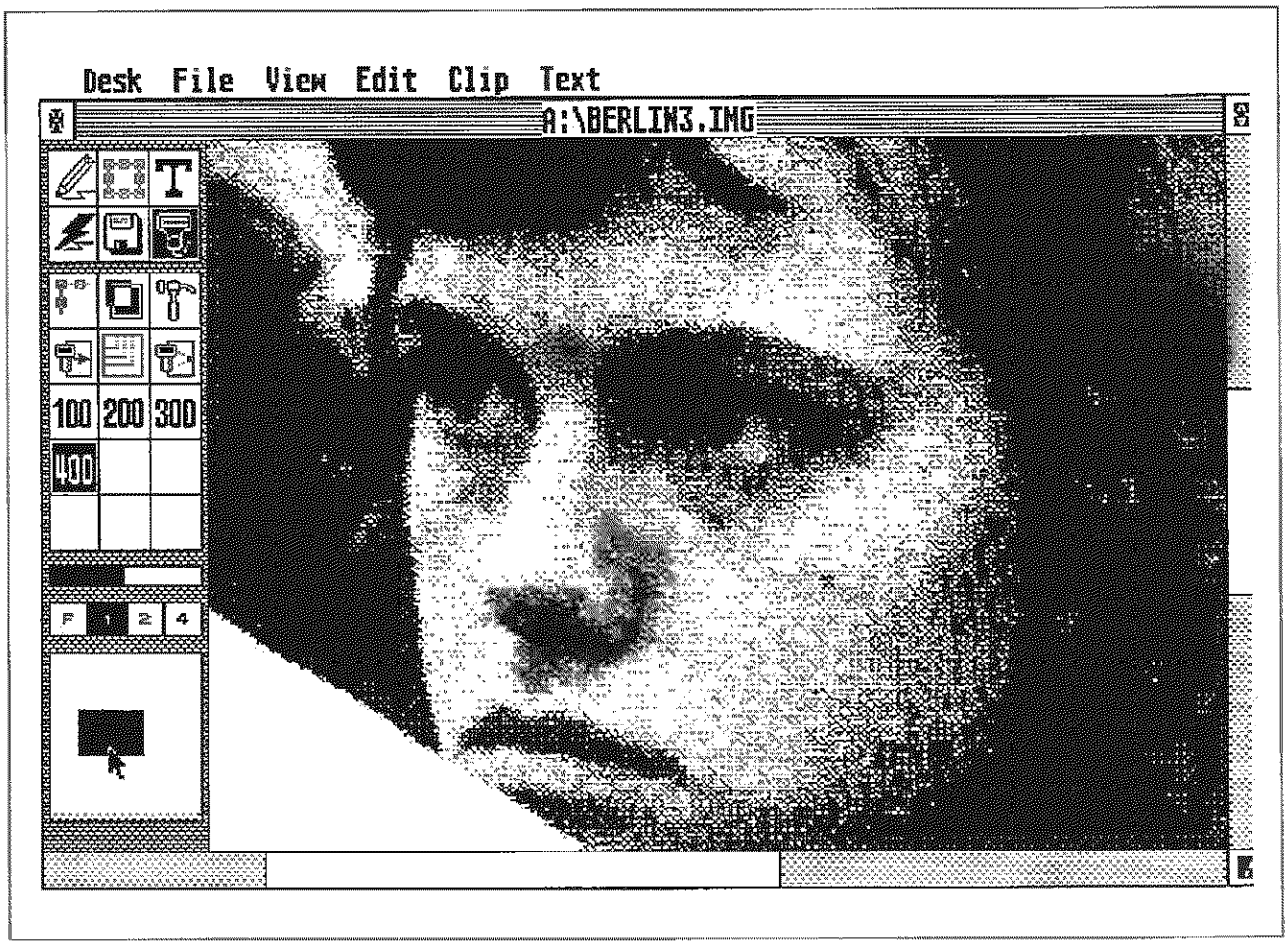
Le scanneur

On clique sur l'icône; le scanneur s'allume et on dispose de 5 secondes après la confirmation de l'effacement de la page pour commencer à balayer l'image à reproduire. Il faut déplacer le scanneur de façon rectiligne du haut de l'image vers le bas (on peut s'aider d'une règle épaisse pour le guider) et suffisamment lentement pour éviter de faire clignoter ou s'éteindre le témoin de vitesse de numérisation. Si la limite de longueur de zone est atteinte au cours de l'opération, le scanneur s'éteint et l'image apparaît à l'écran; dans le cas contraire, il suffit d'attendre quelques secondes pour que l'appareil s'éteigne de lui-même.



Scanner dans le sélecteur

Le sélecteur étant correctement dimensionné, l'opération se déroule comme pour une page complète, sauf qu'elle s'interrompt dès que le sélecteur est plein ; le reste de la page n'est pas effacé et en fonction du mode de pose, l'image peut très bien apparaître superposée aux éléments déjà présents sur la page (mode Transparent ou XOR). On peut utiliser le sélecteur pour lire un document plus large que la fenêtre du scanneur en le traitant en plusieurs bandes verticales : l'ajustage final n'est certes pas des plus simple !



La restitution des photos est correcte à 400 points.



6

Sommaire

6/1 Le son sur l'Atari ST
6/1.1 Comment produire un son avec un ordinateur
6/1.2 Le générateur de son programmable
I. La présentation de l'AY-3-8910
II. La programmation de l'AY-3-8910
III. Le programme d'enveloppe de son en C
IV. Un éditeur de son
6/1.3 Programmation du générateur sonore sous interruption
6/1.3.1 Utilisation des interruptions sur Atari ST
6/1.3.2 Programmation du générateur sonore
6/1.3.3 Programmation d'un délai et d'un écho
6/1.3.4 Utilisation du timer A pour générer une suite de notes sous interruption
6/1.3.5 Arpégiateur sous VBL
6/1.3.6 Arpégiateur sous timer B et VBL
6/1.3.7 Activation polyphonique du générateur sonore
6/1.3.8 Créer un éditeur de partitions
6/1.3.9 Saisie graphique de morceaux de musique
6/1.3.10 Reproduction d'échantillons sur le ST
6/1.3.11 Ecran sonore
6/1.3.12 Séquenceur rythmique
6/1.4 Création d'enveloppe à points multiples
6/1.5 Générateur d'accompagnement automatique
6/1.6 Jeux musicaux en GFA

6/2 L'environnement MIDI
6/2.1 Généralités
6/2.2 Le câblage MIDI
6/2.3 Les codes MIDI
I. Les codes message canal
II. Les codes message mode
III. Les codes message temps réel

IV. Les codes message système commun
V. Les codes message système exclusif

6/2.4 Programmation MIDI
6/2.4.1 Programme de détection du running statut
6/2.4.2 Arpégiateur harmonique sur le générateur sonore commandé depuis un clavier MIDI
6/2.4.3 Boîte d'effets MIDI
6/2.4.4 Analyse d'accords
6/2.4.4.1 Analyse des positions incomplètes d'accords
6/2.4.5 Dictée musicale
6/2.4.6 Principe et construction d'un éditeur de sons
I. Editeur des paramètres sonores
II. ADSR à cinq points avec affichage des paramètres
III. Transfert des données exclusives
6/2.4.7 Dictée rythmique

6/3 Plus loin
6/3.1 Les possibilités sonores de l'Atari STE

6/4 Les grands logiciels musicaux
6/4.1 FM Melody Maker
6/4.2 PRO12 de Steinberg
6/4.3 ST Replay 4, logiciel d'échantillonnage sonore

6/1

Le son sur l'Atari ST

Le son est une des grandes spécialités de l'Atari ST. Les ingénieurs d'Atari ont eu la brillante idée de proposer sur la même machine un processeur sonore performant susceptible d'être utilisé dans des jeux et ainsi qu'une interface MIDI destinée à connecter tout un environnement musical professionnel à base de synthétiseurs. Nous reparlerons de cette dernière un peu plus tard. Voyons d'abord les possibilités offertes par le processeur sonore de l'Atari ST pour produire directement des sons de haute qualité.

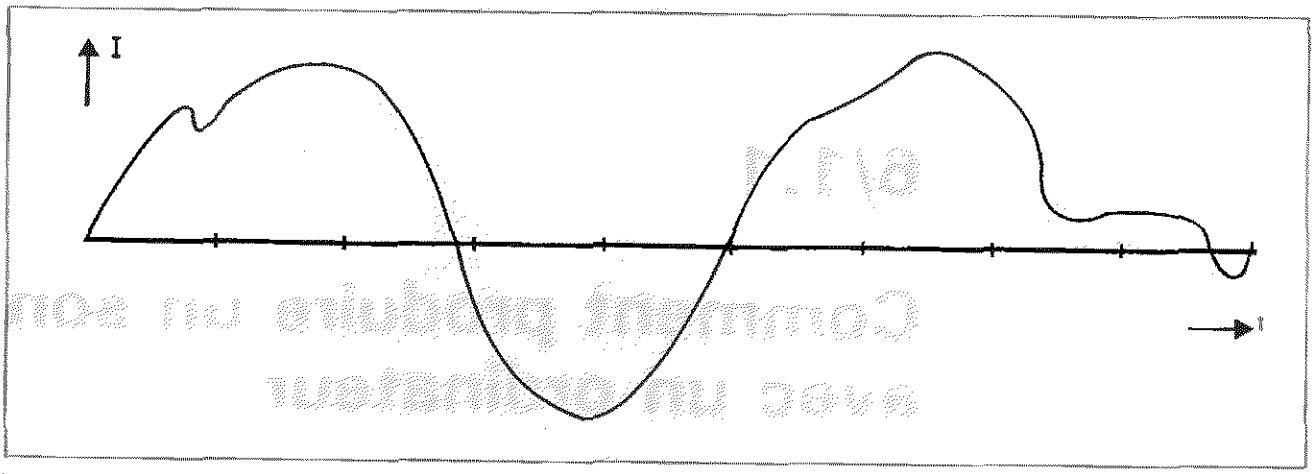
6/1.1

Comment produire un son avec un ordinateur

Pour créer un son, il faut fabriquer un signal électrique périodique que l'on envoie ensuite vers un haut parleur. La membrane du haut parleur se met alors à vibrer conformément aux variations de ce même signal électrique. La vitesse à laquelle vibrera la membrane conditionnera la hauteur du son. Un son grave aura une fréquence faible alors qu'un son aigu aura une fréquence élevée. C'est ce qui va, en musique, différencier un do d'un la. Un son audible possède une fréquence comprise entre 20 Hz

et 15 000 Hz. Ces limites sont, bien entendu, fonction de l'auditeur et varie fortement avec son âge. D'un autre côté, chacun sait qu'un piano produira un la totalement différent du la d'un saxophone. C'est là que les choses se compliquent. Sans se plonger dans de sombres théories, on peut simplement signaler qu'un second paramètre important concerne l'enveloppe du son. Ce paramètre permet de définir les variations d'intensité entre l'apparition de ce son et sa disparition complète. Ce paramètre influe considérablement sur l'aspect sonore du son. Pour prendre un exemple très simple, un son de flûte est maintenu très longtemps, tandis que celui d'une guitare sera généralement plus bref. De plus, mais nous n'aborderons pas vraiment cet aspect, un son de fréquence ou plutôt de hauteur donnée est en fait composé d'harmoniques qui altéreront la pureté du son. Pour reprendre l'exemple précédent, le son de la flûte est très pur et une analyse plus fouillée nous montrerait que le signal est très proche de la sinusoïde. A l'opposé, le saxophone est constitué d'un signal beaucoup plus complexe qui n'a plus rien à avoir avec celui de la flûte. Mais nous ne donnerons pas ici un cours sur la théorie des sons. Nous n'avons la prétention que d'apprendre à en créer avec l'Atari ST.

Pour produire un son sur ordinateur, il faut donc construire un signal périodique en tenant compte des paramètres dont nous venons de parler. Cela prendrait beaucoup de temps à l'ordinateur s'il devait calculer à chaque microseconde une nouvelle valeur pour construire ce signal périodique.



Exemple de signal sonore.

Heureusement, il n'en n'est rien, car l'Atari ST est équipé d'un générateur de son programmable, le AY-3-8910. Celui-ci libère le 68000 de cette tâche très gourmande en temps machine.

6/1.2

Le générateur de son programmable

I - La présentation de l'AY-3-8910

(Vous pouvez également vous reporter à la Partie 2, chapitre 1.4. II)

Le PSG (*Programmable Sound Generator*) est en mesure de générer une grande variété de sons sur 3 voies indépendantes. Il contient également un générateur de bruit blanc qui peut être mélangé à n'importe laquelle des voies. De plus, le son produit peut être assorti d'une enveloppe qui élargit considérablement la variété des sons possibles. Sa programmation se fait par l'intermédiaire de 16 registres spécialisés de taille 8 bits. Sur l'Atari ST, le PSG est cadencé par une horloge de 2 MHz. Nous verrons plus loin que cette valeur conditionne la gamme des sons possibles.

II - La programmation de l'AY-3-8910

Le PSG possède donc 3 canaux A, B et C qui produisent chacun un signal carré de fréquence programmable. Chaque canal est configurable par l'utilisateur par l'intermédiaire de 3 registres. Deux registres permettent de programmer la fréquence du canal correspondant. En fait de fréquence, on programme plutôt la période du son désiré. Cette période (exprimée en cycles horloge) est calculée en fonction de la fréquence de cadencement du PSG qui est de 2 MHz. La formule à utiliser est :

PERIODE = FREQ HORLOGE/(16*FREQ VOULUE)

Ainsi la période d'un La (440 Hz) est :

2000000/16*440=284

Il faut donc séparer cette valeur en deux parties pour l'écrire dans les deux registres de période :

R0 = 284 MODULO 256 = 28
R1 = 284 / 256 = 1

Les registres R0 et R1 contrôlent le canal A, R2 et R3 contrôlent le canal B, et R4 et R5, le canal C.

Le volume est programmable à l'aide d'un registre par canal contenant 5 bits utilisables. Si le bit 4 est à 0, seuls les 4 bits de poids faible sont

pris en compte, ce qui permet d'obtenir 16 niveaux différents de volume. Si le bit 4 est à 1, c'est le générateur d'enveloppe, que nous étudierons plus loin, qui détermine le volume à un instant donné.

Le générateur de bruit est contrôlé par un registre de période (R6) sur seulement 5 bits. Le calcul est identique à celui que nous avons déjà vu pour les canaux A, B et C.

La validation de un ou plusieurs canaux se fait par le registre R7 qui contrôle le mélange des sons en provenance des différents générateurs. Les bits B0 à B2 valident ou non le son sur un canal donné. Attention, le canal est validé si le bit est à zéro.

Par exemple, si B0=0, B1=1 et B2=0, seuls les canaux A et C seront actifs. Les bits B3 à B5 contrôlent le mélange du bruit sur un ou plusieurs canaux. Les bits B6 et B7 contrôlent les ports E/S, A et B.

		B7	B6	B5	B4	B3	B2	B1	B0
R0	Période	8 BITS FAIBLES DE LA PERIODE DU CANAL A							
R1	Canal A					4 BITS FORT CANAL A			
R2	Période	8 BITS FAIBLES DE LA PERIODE DU CANAL B							
R3	Canal B					4 BITS FORTS CANAL B			
R4	Période	8 BITS FAIBLES DE LA PERIODE DU CANAL C							
R5	Canal C					4 BITS FORTS CANAL C			
R6	Pér. Bruit				PERIODE GENERATEUR DE BRUIT				
R7	Validation	ES/A	ES/B	BRUIT CANAUX A/B/C			SON CANAUX A/B/C		
R8	Volume A				MODE	VOLUME CANAL A (0..15)			
R9	Volume B				MODE	VOLUME CANAL B (0..15)			
RA	Volume C				MODE	VOLUME CANAL B (0..15)			
RB	Période enveloppe	8 BITS FAIBLES DE LA PERIODE ENVELOPPE							
RC		8 BITS FORTS DE LA PERIODE ENVELOPPE							
RD	Forme env					CONT	ATT	ALT	RET
RE	Port E/S A	PORT PARALELLE 8 BITS E/S A							
RF	Port E/S B	PORT PARALELLE 8 BITS E/S B							

Les registres du générateur sonore.

Abordons maintenant l'étude de l'enveloppe. Nous avons vu plus haut que l'enveloppe était représentative de la manière dont le son évoluait au cours du temps. Le AY-8910 possède 3 registres de contrôle de l'enveloppe. Les deux premiers (R11 et R12) contiennent la période de l'enveloppe tandis que le second (R13) contrôle le type d'enveloppe par le biais de 4 bits. Cette période est également calculée en fonction de la fréquence de cadencement du PSG qui est de 2 MHz, dans ce cas, nous disposons de 16 bits pour coder cette période. La formule à utiliser est :

PERIODE = FREQ HORLOGE/(256*FREQ VOULUE)

A partir d'une fréquence d'horloge de 2 MHz, la période d'horloge minimale sera donc 1, correspondant à la fréquence maximale :

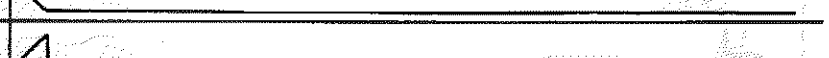
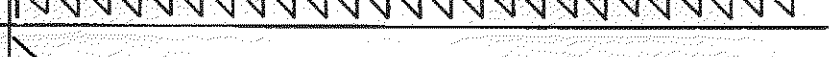
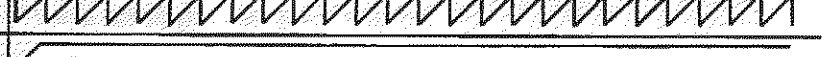
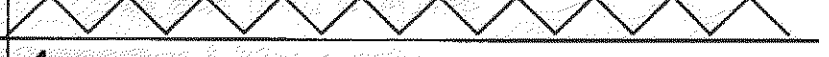
FREQ MAXI = 200000/256 = 7812.5 Hz

Alors que la fréquence minimale sera basée sur une période maximale de 65535 :

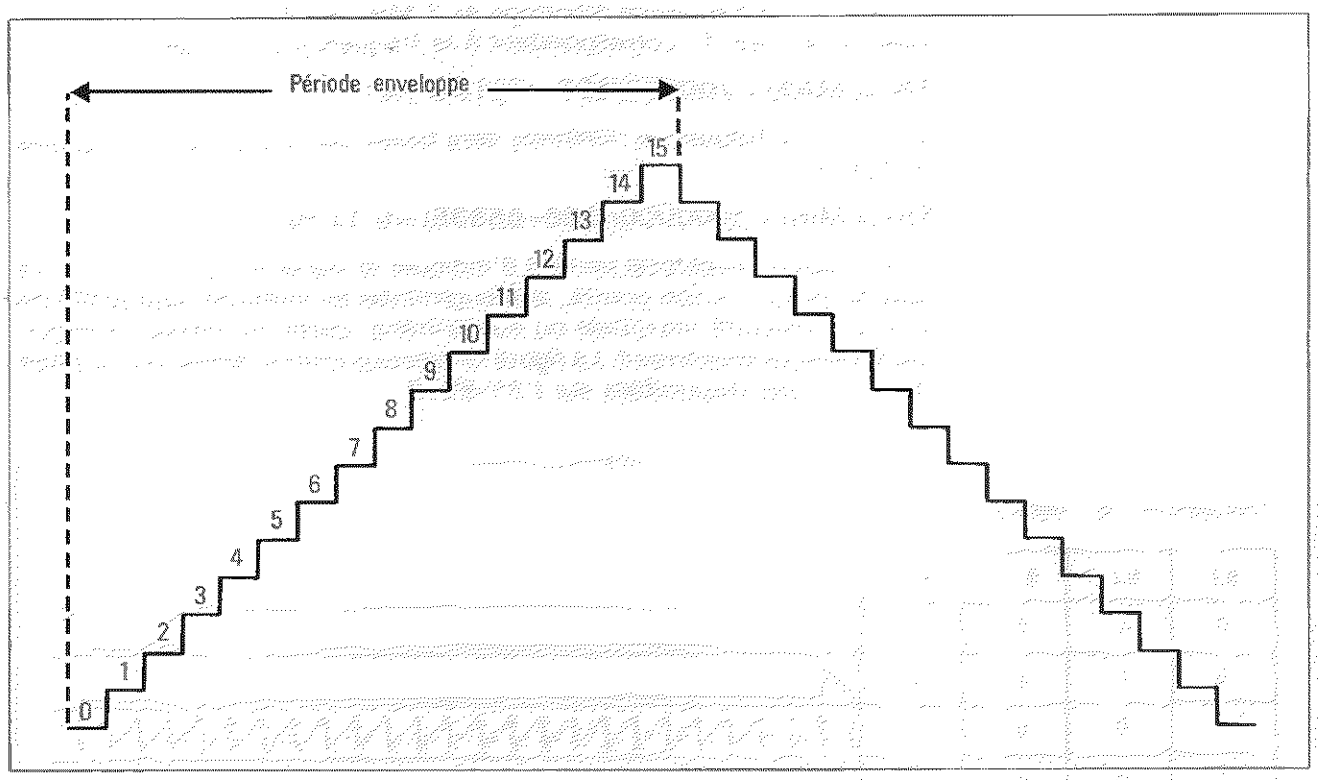
FREQ MINI = 2000000/(256*65535) = 0.12 Hz

La période d'enveloppe permet d'indiquer la durée du son. Pendant ce laps de temps, le son pourra, soit augmenter en intensité pour atteindre à la fin l'intensité maximale ou au contraire, partir du maximum et disparaître progressivement. La figure ci-dessous montre les différents types d'enveloppes disponibles sur l'AY-8910.

Configuration du registre R13

B3	B2	B1	B0	
0	0	X	X	
0	1	X	X	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Différentes enveloppes possibles sur l'AY-8910.



Exemple d'enveloppe d'un son.

Voyons quelques exemples de son qu'il est possible de produire avec l'AY-8910 :

— remplir tous les registres avec 0×00 pour initialiser le PSG ;

— définir la fréquence du canal A à 440 Hz par la valeur 0×11C (0×1C dans R0 et 0×01 dans R1) ;

— valider le canal A en écrivant 0×3E sur le registre R7 ;

— mettre le volume du canal A au maximum en écrivant 0×0F dans le registre R8 ;

— attendre 350 ms par une boucle quelconque ;

— définir la fréquence du canal A à 187 Hz en écrivant 0×9c dans R0 et 0×02 dans R1 ;

— attendre 350 ms avant de continuer ;

— stopper le canal A en écrivant 0×0 dans le registre de volume R8.

Ce son ressemble à un bruit de sirène d'usine.

Le système d'exploitation de l'Atari ST comprend une fonction dédiée au contrôle du son sous interruption. Cette fonction fait partie du XBIOS (*eXtended Basic Input Output System*) (voir Partie 3, chapitre 3) et



s'appelle DOSOUND. Elle permet de se libérer des problèmes de temporisation en envoyant simplement une liste de commandes qui sera exécutée sous interruption. Etudions la structure de la liste de commandes.

Cette liste de commandes est constituée d'octets possédant chacun son rôle propre.

Les numéros de 0×00 à 0×0F ont un octet d'argument et permettent d'écrire dans un registre du PSG. Par exemple : 0×08 0×0C permet de mettre le volume du canal A à la valeur 12.

La commande 0×80 place l'argument dans un registre temporaire.

La commande comprend 3 arguments. Le premier représente le numéro de registre à charger en utilisant le contenu du registre temporaire. Le deuxième argument est une valeur à ajouter au registre temporaire et le dernier est la valeur de fin. Ce type de commande permet de réaliser des effets de balayage de fréquence ou d'intensité assez intéressants.

Les commandes 0×82 à 0×FF demandent un paramètre qui représente le nombre d'interruptions système (chacune dure 20 ms) avant la prochaine mise à jour. Si le paramètre est égal à zéro, le son est stoppé immédiatement.

Dans le cas de l'exemple précédent, le son pourra être déclenché par l'appel de la fonction DOSOUND avec la liste suivante :

```
char SoundList[ ]= {
0×08, 0×00,          /* Volume canal A à zéro   */
0×09, 0×00,          /* Volume canal B à zéro   */
0×0A, 0×00,          /* Volume canal C à zéro   */
0×00, 0×1C,
0×01, 0×01           /* 440 Hz en canal A       */
0×07, 0×3E,          /* Autorise le canal A     */
0×08, 0×0F,          /* Volume 15 en canal A    */
0×82, 0×11,          /* 17×20 ms = 340 ms       */
0×00, 0×9C,
0×01, 0×02,          /* 187 Hz en canal A       */
0×82, 0×11,          /* Attend 350 ms           */
0×08, 0×00,          /* Stoppe le canal A       */
0×FF, 0×00           /* Fin de la liste         */
};
```

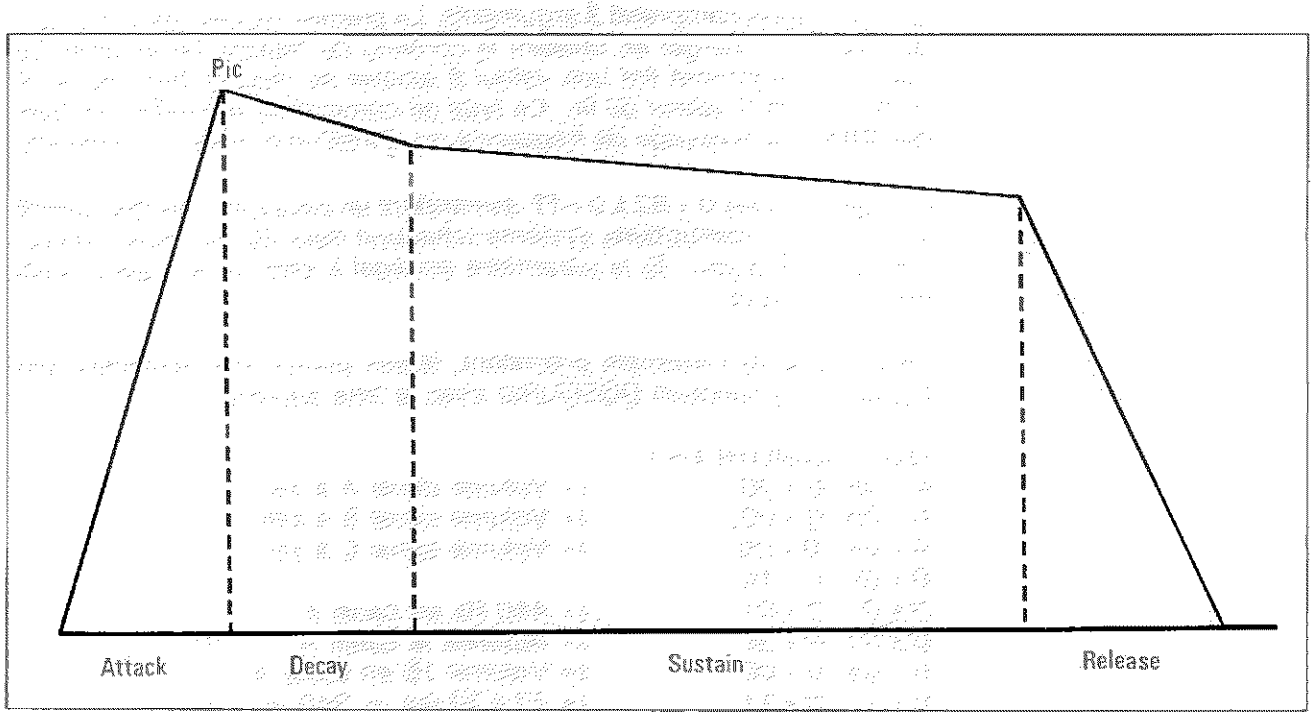
Nous allons maintenant étudier un autre moyen de produire des sons avec le AY-8910. Comme nous l'avons vu précédemment, la génération d'enveloppes est limitée à seulement quelques choix. Il existe, dans le domaine musical, un type de description des enveloppes. Il porte le nom ADSR. Ces quatre lettres sont synonymes des quatre mots anglais *Attack Decay Sustain Release*. Cela permet de décrire la façon dont un son évoluera dans le temps. Dans le cas des enveloppes ADSR, cette description est très poussée.

Attack indique la façon dont le son va démarrer. Il peut évoluer doucement depuis une intensité nulle vers un pic qui sera l'intensité maximale du son. Il peut aussi apparaître brutalement et atteindre son pic en quelques millisecondes.

Decay indique la façon dont le son va évoluer depuis son pic vers un régime de croisière.

Sustain représente ce régime de croisière et peut durer un laps de temps très court pour un piano (note piquée). Pour un orgue, le son sera maintenu pendant un temps un peu plus long.

Release est relatif à la disparition du son, l'intensité du son en fin de cette phase est bien entendu nulle.



Les quatre zones descriptives d'un son.

III - Le programme d'enveloppe de son en C

Voici un programme en C qui utilise une routine sous interruptions pour contrôler une enveloppe de type ADSR. Dans notre cas, seul le volume sera enveloppé. Plus tard, nous étudierons un module complet de gestion d'une enveloppe ADSR, mais aussi bien pour la fréquence que pour l'intensité.



```
        extern long gemdos();
        extern long bios();
        extern long xbios();

    /*  Module de gestion d'enveloppe de type ADSR
        WEKA 1989
    */
    /*  Définitions des accès directs à la mémoire
    */
    #define PEEK(x) *(char *)(x)
    #define POKE(x,y) *(char *)(x)=y
    #define DPEEK(x) *(int *)(x)
    #define DPOKE(x,y) *(int *)(x)=y
    #define LPEEK(x) *(long *)(x)
    #define LPOKE(x,y) *(long *)(x)=y

    /* Définition des registres du AY-8910 */
    #define AY_8910 0xFF8800L

    #define AY_Per_Al 0
    #define AY_Per_Ah 1
    #define AY_Per_Bl 2
    #define AY_Per_Bh 3
    #define AY_Per_Cl 4
    #define AY_Per_Ch 5
    #define AY_Per_Br 6
    #define AY_Activ 7
    #define AY_Vol_A 8
    #define AY_Vol_B 9
    #define AY_Vol_C 10
    #define AY_Per_El 11
    #define AY_Per_Eh 12
    #define AY_For_Env 13

    /* Calcul d'une nouvelle valeur de volume à partir  */
    /* d'un volume courant, d'un volume final et de     */
    /* temps courants et temps final                    */
    /* Le calcul est (V-Vf)/(T-Tf).                     */
    CalcVol(CurVol,FinVol,CurTime,FinTime)
    int CurVol,FinVol,CurTime,FinTime;
    {   int DV=(FinVol-CurVol)*10;

        if (FinTime==CurTime) return DV/10;
        DV /= (FinTime-CurTime);
        if (DV<0) DV-=5;
        else DV+=5;
        DV/=10;
        return DV;
```

(suite)

```
/* Structure des données utilisées pour la gestion  */
/* d'enveloppe sur un canal                         */
typedef struct
{
    int Freq;    /* Fréquence actuelle du canal   */
    int Vol;     /* Volume actuel du canal        */
    int Time;    /* Moment du canal               */
    int Etat;    /* Etat en cours 0,1,2,3,4       */
                 /* 0 : Inactif                   */
                 /* 1 : Attaque                   */
                 /* 2 : decay                     */
                 /* 3 : sustain                   */
                 /* 4 : release                   */
    int Tp;      /* Moment du peak                */
    int Td;      /* Moment du decay               */
    int Ts;      /* Moment du sustain             */
    int Tr;      /* Moment du release             */
    int Vp;      /* Volume du peak                */
    int Vd;      /* Volume du decay               */
    int Vs;      /* Volume du release             */
} CANAL;

CANAL Canal[3];      /* Pour trois canaux */

/* Ecriture d'un registre sur le AY8910
*/
WriteReg(Reg,Value)
int Reg;
int Value;
{
    POKE(AY_8910,Reg);       /* Sélectionne le registre  */
    POKE(AY_8910+2,Value);   /* Ecrit le registre        */
}

/* Lecture d'un registre du AY8910
*/
ReadReg(Reg)
int Reg;
{
    POKE(AY_8910,Reg);       /* Sélectionne le registre  */
    return PEEK(AY_8910+2); /* Lit le registre           */
}

/* Routine de gestion de l'enveloppe                    */
/* Cette fonction sera exécutée à chaque VBL            */
/* Chaque canal contient une variable d'état qui permet */
```

(suite)

```
/* de savoir si ce canal est en attaque, en decay, en    */
/* sustain ou en release. Cet état permet de savoir quel*/
/* calcul il faut effectuer.                             */
Enveloppe()
{
     int i;
     int DeltaVol;
     for (i=0;i<3;i++)
     {
          switch (Canal[i].Etat) {
          case 0 :    /* Canal inactif, on passe au canal */
                      /* suivant si il y en a un.         */
               continue;

          case 1 :    /* Période d'attaque, on calcule    */
                      /* quelle est la variation d'intens.*/
                      /* à appliquer pour atteindre le pic*/
               DeltaVol=CalcVol(Canal[i].Vol,
                                Canal[i].Vp,
                                Canal[i].Time,
                                Canal[i].Tp);
               Canal[i].Vol+=DeltaVol;
                    /* Si on est au pic, on peut changer */
                    /* l'état et passer au decay         */
               if (++Canal[i].Time==Canal[i].Tp)
                    Canal[i].Etat++;
               break;

          case 2 :    /* Période de decay */
               DeltaVol=CalcVol(Canal[i].Vol,
                                Canal[i].Vd,
                                Canal[i].Time,
                                Canal[i].Td);
               Canal[i].Vol+=DeltaVol;
               if (++Canal[i].Time==Canal[i].Td)
                    Canal[i].Etat++;
               break;

          case 3 :    /* Période de sustain   */
               DeltaVol=CalcVol(Canal[i].Vol,
                                Canal[i].Vs,
                                Canal[i].Time,
                                Canal[i].Ts);
               Canal[i].Vol+=DeltaVol;
               if (++Canal[i].Time==Canal[i].Ts)
                    Canal[i].Etat++;
               break;
```

(suite)

```
                case 4 :     /* Période de release    */
                    DeltaVol=CalcVol(Canal[i].Vol,
                                     0,
                                     Canal[i].Time,
                                     Canal[i].Tr);
                    Canal[i].Vol+=DeltaVol;
                    if (++Canal[i].Time==Canal[i].Tr)
                    {   /* On met l'état à zéro pour désactiver */
                        /* le son mais si on mettait 1 le son   */
                        /* recommencerait                       */
                        Canal[i].Etat=0;
                        Canal[i].Time=0;
                        Canal[i].Vol=0;
                    }
                    break;
                }
                /* On fait évoluer le volume en fonction de
                   de DeltaVol.
                */
                WriteReg(AY_Vol_A+i,Canal[i].Vol);
            }
        }

        /* Sélection de la période pour un canal
            il faut donner la période et le canal voulu
        */
        SetFreq(Periode,Canal)
        int Periode;
        int Canal;
        {
            long OldSSP=gemdos(0x20,0L);
            WriteReg(AY_Per_Al+2*Canal,Periode&0xFF);
            WriteReg(AY_Per_Ah+2*Canal,(Periode>>8)&0x0F);
            gemdos(0x20,OldSSP);
        }

        /* Installation de la routine de gestion d'enveloppe     */
        /* sur VBL. En 456 se trouve un pointeur sur une liste   */
        /* de pointeurs sur des routines qui seront exécutées à  */
        /* chaque interruption VBL. Cette interruption a lieu à  */
        /* chaque retour trame. Pour un moniteur monochrome,     */
        /* c'est 70 fois par seconde et sur un moniteur couleur  */
        /* c'est 50 fois par seconde. C'est un moyen très simple */
        /* d'exécuter régulièrement une routine. La méthode de   */
        /* l'installation consiste à tester les pointeurs de la  */
        /* liste et mettre l'adresse de la routine à installer   */
        /* dès que l'on trouve un 0.                             */
```



(suite)

```
InstallVBL()
{
     long OldSSP=gemdos(0x20,0L);          /* Passe en superviseur */
     int NbVBL=DPEEK(0x454);       /* Lit le maxi de VBLs  */
     long *VblList=LPEEK(0x456); /* Liste des routines   */

     while (NbVBL--)
     {
          if (*VblList++==0)
          {
               *--VblList=&Enveloppe;
               gemdos(0x20,OldSSP);
               return 0;
          }
     }
     gemdos(0x20,OldSSP);
     return -1;
}

/* Lorsque l'on quitte le programme, il faut remettre   */
/* la liste dans l'état initial. Dans le cas contraire  */
/* la routine continuerait à etre exécutée alors que le */
/* programme n'est plus présent en mémoire. En clair, il*/
/* y aurait quelques bombes dès que l'on chargerait un  */
/* nouveau programme.                                   */
SupprimeVBL()
{
     long OldSSP=gemdos(0x20,0L);          /* Passe en superviseur */
     int NbVBL=DPEEK(0x454);       /* Lit le maxi de VBLs  */
     long *VblList=LPEEK(0x456); /* Liste des routines   */

     while (NbVBL--)
     {
          if (*VblList++==&Enveloppe)
          {
               *--VblList=0L;
               gemdos(0x20,OldSSP);
               return 0;
          }
     }
     gemdos(0x20,OldSSP);
     return -1;
}

main()
{    int i;
```

(suite)

```
/* Initialise les 3 canaux à l'état 0 */
for (i=0;i<3;i++) Canal[i].Etat=0;

InstallVBL();

/* On met le canal A à 440 Hz */
SetFreq(0x11C,0);

/* On met 2 (40 ms) en temps d'attaque et
   15 en volume de peak
*/
Canal[0].Tp=2; Canal[0].Vp=15;

/* On met 5 (100 ms) en moment de decay et
   12 en volume
*/
Canal[0].Td=5;  Canal[0].Vd=12;

/* On met 55 (1,1 s) en moment de sustain et
   12 en volume de sustain
*/
Canal[0].Ts=55; Canal[0].Vs=12;

/* Et 60 (1.2 s) en moment de release
*/
Canal[0].Tr=60;

/* On lance le son */
Canal[0].Time=Canal[0].Vol=0;
Canal[0].Etat=1;
bios(2,2);       /* Attend une touche au clavier */
SupprimeVBL();
```



IV - Un éditeur de son

LES REGISTRES DU PSG

Les nombreuses possibilités du générateur de son YM-2149 font de celui-ci un des composants sonores les plus répandus. Pour permettre une utilisation très large de ce composant, un grand nombre de paramètres sont directement accessibles par la programmation. Ces paramètres sont manipulés à l'aide de 14 registres dont 1 est multifonctionnel.

Les caractéristiques du YM-2149 sont :

— 3 voix indépendantes (3 oscillateurs) ;

— 1 générateur de bruit associé à une voix (bruit blanc) ;

— 1 générateur d'enveloppe (contrôle de la variation de puissance du son dans le temps) qui peut à partir d'une certaine vitesse d'exécution influencer considérablement le timbre du son.

Descriptif des fonctions de chaque registre :

— Registres 0 et 1 : ils fixent la période du premier oscillateur, et déterminent la hauteur de la note. La période est uniquement codée sur 12 bits, elle utilise les huit bits du registre 0 et les 4 bits de poids faible du registre 1. Plus la valeur attribuée aux registres est forte, plus le son est grave. Un tableau vous permet de programmer facilement ces registres.

— Registres 2 et 3 : leurs fonctions sont identiques à celles des registres 0 et 1, les valeurs gérant l'oscillateur de la voix 2.

— Registres 4 et 5 : leurs fonctions sont également identiques à celles des registres 0 et 1, les valeurs gérant l'oscillateur de la voix 3.

— Registre 6 : il gère le générateur de bruit blanc. Seuls les 5 bits de poids faible sont utilisés pour en faire varier la fréquence. Plus la valeur attribuée est haute, plus le son est grave.

— Registre 7 : il gère la mise hors/en fonction des différents oscillateurs. Chacun des bits gère une opération.

Bit	Voie	Opération
Bit 0	voie 1	0 = ef / 1 = hf
Bit 1	voie 2	0 = ef / 1 = hf
Bit 2	voie 3	0 = ef / 1 = hf
Bit 3	bruit sur 1	0 = ef / 1 = hf
Bit 4	bruit sur 2	0 = ef / 1 = hf
Bit 5	bruit sur 3	0 = ef / 1 = hf
Bit 6	port A	0 = entrée / 1 = sortie
Bit 7	port B	0 = entrée / 1 = sortie
hf = hors fonction, ef = en fonction		

Les bits 6 et 7 ne sont pas utilisés par la partie sonore, et sont habituellement positionnés à 1.

— Registre 8 : il permet de fixer le volume sonore de la voix 1. Le réglage s'effectue sur 15 pas de 0 à 15 (15 = maximum). Les 4 bits de poids faible sont utilisés à cette tâche. Le bit 4 a une action particulière, s'il est positionné à 0, le volume de sortie de la voix 1 est contrôlé par la valeur des 4 bits de poids faible, si le bit 4 est à 1 (valeur décimale 16) le volume de la voix est en relation avec l'enveloppe choisie par le registre 13, les 4 bits de poids faible sont alors totalement ignorés.

— Registre 9 : même action que le registre 8 pour la voix 2.

— Registre 10 : même action que le registre 8 pour la voix 3.

— Registres 11 et 12 : les 16 bits de ces registres permettent le réglage de la vitesse de défilement de l'enveloppe choisie à l'aide du registre 13. Le registre 11 est l'octet de poids faible et le registre 12 le poids fort. Les valeurs s'échelonnent de 0 à 65535. Plus la valeur est faible et plus la fréquence de l'enveloppe s'élève.

— Registre 13 : il gère le type d'enveloppe utilisée par les voix qui ont sélectionné ce mode de jeux à l'aide du bit 4 du volume (volume à 16). Seuls les 4 bits de poids faible sont utilisés. Un tableau détaillé des différents types d'enveloppe vous est donné en Partie 6, chapitre 1 à 1.2 page 5.

— Registres 14 et 15 : ces registres, bien qu'appartenant au générateur sonore, sont utilisés pour les entrées/sorties des ports A et B. Ils n'interviennent à aucun moment dans la programmation du générateur sonore.

Pour bien comprendre le fonctionnement de chacun des registres, nous allons programmer quelques exemples puis fabriquer un éditeur de son.

Deux solutions s'offrent à nous pour accéder aux registres, soit utiliser la fonction POKE du GFA, soit utiliser les fonctions SOUND et WAVE. Dans le programme suivant les deux solutions sont utilisées en parallèle.

Pour les programmes en assembleur, l'accès aux adresses du générateur de son s'effectue en mode superviseur.

• *Utilisation de POKE*

Pour programmer le générateur sonore il faut toujours lui envoyer deux valeurs : le numéro du registre suivi de la valeur à lui attribuer. La fonction POKE nécessite de connaître l'adresse du générateur sonore. Ces adresses sont :

adresse du n° de registre $FF8800

adresse de donnée $FF8802,



Exemple :

Changer le volume de la voix 2 :

```
POKE $FF8800,8
POKE $FF8802,0      ! Volume à 0
Traitement X ou attente
POKE $FF8800,8
POKE $FF8802,15     ! Volume au maximum 15
```

• *Utilisation de SOUND et WAVE*

L'instruction SOUND peut être suivie de 4 ou 5 paramètres :

SOUND canal, vol, note, octave, délai
SOUND canal, vol, #période, délai

La fonction **délai** n'appartient pas au générateur de son mais effectue une pause n avant la prochaine instruction.

Canal : sélectionne la voix, mais ne fonctionne pas de la même manière que le registre 7.

vol : registre 8 à 10.

note/octave : registre 0 à 6. Le GFA calcule lui-même la période correspondant à la note et l'octave voulu.

#période : registre 0 à 6. La période est envoyée directement sur 12 bits.

Exemple :

Changer le volume de la voix 2 :

```
SOUND Y,0,X,X,X     Volume à 0
                    Y = 1, 2 ou 3
SOUND Y,15,X,X,X    Volume au maximum 15
```

WAVE voix, env, forme, durée, délai

voix : registre 7

env : registres 8 à 10 bit 4

forme : registre 13

durée : registres 11 et 12

délai : pause avant l'instruction suivante

Exemples :

1) Modification de la hauteur de la note.

Voix 1, montée des 12 notes de la gamme à raison d'une note par seconde :

```
FOR A=1 TO 12
SOUND 1,15,A,2,50
NEXT A
```

2) Même exemple mais la voix 2 descend simultanément la gamme :

```
FOR A=1 TO 12
SOUND 1,15,A,2
SOUND 2,15,13-A,2
WAVE 3,0,0,0,50
NEXT A
```

L'instruction WAVE est obligatoire pour permettre la polyphonie (plusieurs sons joués simultanément). L'omission de cette instruction empêche la polyphonie, seule la deuxième instruction SOUND serait exécutée. L'instruction SOUND contient uniquement l'ouverture de la voix définie par le canal, la dernière instruction SOUND élimine les précédentes.

L'étude détaillée des petits programmes comme ceux présentés en exemple vous donnera certainement de nouvelles idées d'utilisation du générateur sonore.

PROGRAMMATION D'UN ÉDITEUR DE SON

L'éditeur de sons vous permet de mieux comprendre, en temps réel, le rôle de chacun des registres du générateur de son. Le programme est écrit en GFA 3, pour moyenne et haute résolution.

UTILISATION DU PROGRAMME

Pour activer un curseur, cliquer avec le bouton gauche de la souris, la valeur du réglage apparaît en dessous du curseur. Certains curseurs sont doubles car la valeur du paramètre utilise deux registres.

Pour le registre 7, les deux boutons de la souris sont utilisés, si l'on clique le bouton gauche le bit correspondant se met à 1, et il s'annule si l'on clique à droite. La remise à zéro permet de reprendre depuis le début la programmation des sons. Les bits 6 et 7 sont inaccessibles (comptez les bits de 7 à 0), car indépendants de la gestion des sons.

Pour sortir du programme, il suffit d'appuyer sur la barre d'espace.

Notez les valeurs de chacun des registres afin d'utiliser vos propres sons dans vos programmes.



```
'Programme de test des performances du générateur
'sonore
'
'
'            Programme en GFA basic 3
'
'
'
'   ---------- INITIALISATION DE L'ATARI ----------
DEFINT "a-z"
CLS
DEFMOUSE 0
SHOWM
'
'       ---------- DEBUT DU PROGRAMME ---------'
'
PRINT AT(20,1);"--- PROGRAMMATION DU GENERATEUR
SONORE ---"
PRINT AT(1,3);"REG N°    0,1     2,3     4,5     6
8         9      10      11,12     13"
PRINT AT(35,19);"REGISTRE 7"
'
' -------recherche de la résolution -----'
'
res=XBIOS(4)      ! res=1 en moyenne résolution
IF res=0          ! res=2 en haute résolution
  PRINT AT(50,50);"Mauvaise résolution !!"
ENDIF
'
'initialisation de la puce son et affichage --'
init
' -------- programme principale  ---------------'
DO
  EXIT IF INKEY$=" "
  MOUSE x,y,k
  IF k<>0
    FOR a=65 TO 580 STEP 60
      IF x>a AND y>25*res AND x<a+30 AND y<100*res
        ON ((a-65)/60)+1 GOSUB per1,per2,per3,per4,
                               vol1,vol2,vol3,per_env,form_env
      ENDIF
    NEXT a
    IF x>200 AND y>125*res AND x<440 AND y<140*res
      reg7
    ENDIF
```

(suite)

```
      IF x>200 AND y>165*res AND x<440 AND y<180*res
        init
      ENDIF
    ENDIF
  LOOP
  '
  '   ---------- procedures----------'
  '
  '
  '     dessin de l'ecran de travail
  '
  PROCEDURE dessine
    DEFFILL 1,0
    FOR a=65 TO 580 STEP 60
      PBOX a,25*res,a+30,100*res
    NEXT a
    FOR a=80 TO 240 STEP 60
      LINE a,25*res,a,100*res
    NEXT a
    LINE 500,25*res,500,100*res
    FOR a=200 TO 410 STEP 30
      PBOX a,125*res,a+30,140*res
      TEXT a+10,(125*res)+(10*res),"1"
    NEXT a
    PBOX 200,165*res,440,180*res
  RETURN
  '   procedure de traitement de la période de
  '   chacune des voix et du bruit
  '
  PROCEDURE per1
    periode(0,1,7,14)
  RETURN
  PROCEDURE per2
    periode(2,3,15,14)
  RETURN
  PROCEDURE per3
    periode(4,5,23,14)
  RETURN
  PROCEDURE per4
    resu=32-((y-(25*res))/4.8)
    SPOKE &HFF8800,6
    SPOKE &HFF8802,resu
    vase
    affi(32,14)
  RETURN
```

(suite)

```
'
' procedure de réglage des volumes
'
PROCEDURE vol1
  resu=17-((y-(25*res))/9)
  SPOKE &HFF8800,8
  SPOKE &HFF8802,resu
  vase
  affi(39,14)
RETURN
PROCEDURE vol2
  resu=17-((y-(25*res))/9)
  SPOKE &HFF8800,9
  SPOKE &HFF8802,resu
  vase
  affi(46,14)
RETURN
PROCEDURE vol3
  resu=17-((y-(25*res))/9)
  SPOKE &HFF8800,10
  SPOKE &HFF8802,resu
  vase
  affi(53,14)
RETURN
'
' Période de l'enveloppe
'
PROCEDURE per_env
  resu=257-((y-(25*res))/0.58)
  IF x<a+15
     SPOKE &HFF8800,11
    c=60
  ELSE
    SPOKE &HFF8800,12
    c=64
    a=a+15
  ENDIF
  SPOKE &HFF8802,resu
  vase1
  affi(c,14)
RETURN
'
'  Choix de la forma d'enveloppe
'
```

(suite)

```
PROCEDURE form_env
  resu=16-((y-(25*res))/10)
  SPOKE &HFF8800,13
  SPOKE &HFF8802,resu
  vase
  affi(69,14)
RETURN
'
'  traitement spéciale pour le registre 7
'
PROCEDURE reg7
  aa=5
  FOR a=260 TO 410 STEP 30
    IF x>a AND x<a+30
      IF k=1
        registre7=BSET(registre7,aa)
        TEXT a+10,(125*res)+(10*res),"1"
      ELSE
        registre7=BCLR(registre7,aa)
        TEXT a+10,(125*res)+(10*res),"0"
      ENDIF
      SPOKE &HFF8800,7
      SPOKE &HFF8802,registre7
    ENDIF
    DEC aa
  NEXT a
RETURN
'
'intialisation ou remise à zero de tous les registres
'
PROCEDURE init
  FOR a=0 TO 13
    IF a<>7
      SPOKE &HFF8800,a
      SPOKE &HFF8802,0
    ENDIF
  NEXT a
  registre7=255
  SPOKE &HFF8800,7
  SPOKE &HFF8802,registre7
  dessine
  SHOWM
  resu=0
  FOR a=7 TO 25 STEP 8
```

(suite)

```
      affi(a,14)
      affi(a+3,14)
    NEXT a
    FOR a=32 TO 60 STEP 7
      affi(a,14)
    NEXT a
    affi(64,14)
    affi(69,14)
    PRINT AT(33,22);" RETOUR A ZERO "
  RETURN
  '
  '  traitement de la période
  '
  PROCEDURE periode(b,h,c,l)
    IF x<a+15
      resu=257-((y-(25*res))/0.58)
      SPOKE &HFF8800,b
      SPOKE &HFF8802,resu
      vase1
    ELSE
      resu=16-((y-(25*res))/10)
      SPOKE &HFF8800,h
      SPOKE &HFF8802,resu
      a=a+15
      vase1
      ADD c,3
    ENDIF
    affi(c,1)
  RETURN
  '
  '     affichage d'un curseur simple
  '
  PROCEDURE vase1
    DEFFILL 1,1
    PBOX a,y,a+15,100*res
    DEFFILL 0,0
    PBOX a+1,(25*res)+1,a+14,y
  RETURN
  '
  '     affichage d'un double curseur
  '
```

(suite)

```
PROCEDURE vase
  DEFFILL 1,1
  PBOX a,y,a+30,100*res
  DEFFILL 0,0
  PBOX a+1,(25*res)+1,a+29,y
RETURN
'
'    affichage des réglages de chacun des curseurs
'
PROCEDURE affi(c,l)
  PRINT AT(c,l);USING "###",resu
RETURN
```

6/1.3

Programmation du générateur sonore sous interruption

6/1.3.1

Utilisation des interruptions sur Atari ST

Le principe d'une interruption est d'interrompre le déroulement normal d'un programme. En général, c'est un traitement spécial qui est effectué à ce moment-là, au moyen d'un programme spécifique appelé routine d'interruption.

Cette routine fonctionne en partie comme un sous-programme puisque l'on doit revenir après traitement à l'endroit où l'interruption s'était produite. Il est évident par contre qu'on ne peut interrompre un programme à n'importe quel moment et n'importe quel endroit et s'attendre à ce que tout refonctionne à nouveau. Il faut donc sauvegarder des registres entiers du processeur, registres de données, registres d'adresses, registres d'état pour que l'opération soit possible. Pour qu'aucun problème n'apparaisse, la routine d'interruption doit rendre la main au programme interrompu exactement dans l'état où elle a trouvé ce dernier. Le retour au programme s'effectuera par un RTE et non un RTS comme ce serait le cas pour un sous-programme classique.

Un autre principe fondamental dans la gestion des interruptions est la possibilité d'interdire ou activer toute source d'interruption ; le programmeur pourra donc choisir les interruptions qu'il permettra et les moments où ces interruptions pourront se produire ; il est bien évident aussi qu'on ne peut impunément interdire n'importe quelle interruption pendant le temps que l'on veut, sans porter atteinte à la bonne marche du système.

Par exemple, le vertical blank est une routine d'interruption qui s'occupe entre autres du clignotement du curseur, de vérifier le type de moniteur connecté, ceci chaque fois que l'image totale de l'écran est dessinée (1/70 de seconde pour un moniteur monochrome haute résolution) ; si vous empêchez le travail de cette routine d'interruption, vous courez au plantage pur et simple de la machine ; si vous décidez d'interrompre cette interruption pendant un certain temps, vous risquez d'avoir des ennuis d'images sur l'écran et de sauvegardes sur le lecteur de disquette ; précaution à prendre, ne pas empêcher ce traitement pendant trop longtemps, car il y a certains événements auxquels on doit répondre immédiatement sous peine d'entraver la bonne marche du système.

Afin de mieux expliciter le mode de fonctionnement d'une interruption, prenons un exemple simple ; soit le programme suivant en GFA Basic qui ne fait que minuter une boucle sur un ordinateur A:

```
t=TIMER
FOR i=1 TO 1000000
NEXT i
PRINT TIMER-t
```

Le résultat est 10325.

A l'aide d'un second ordinateur B et d'un cordon MIDI, branchez le MIDI OUT de l'ordinateur B dans le MIDI IN de l'ordinateur A ; le programme généré par l'ordinateur B est un simple envoi de code MIDI:

```
DO
  OUT 3,120
LOOP
```

Le résultat donné par le programme de l'ordinateur A est maintenant 20412.

Cela veut dire que le même programme a tourné plus lentement à cause de l'interruption MIDI générée par le second ordinateur. Dans ce cas précis, cette donnée MIDI n'intéresse pas le programme de l'ordinateur A; il faudrait désactiver l'interruption MIDI pendant le déroulement du programme A pour optimiser ce dernier.

Dans un cadre plus général, il est indispensable de connaître les sources d'interruptions qui peuvent se manifester pendant un traitement particulier. Il est bon de savoir celles que vous pouvez désactiver, et pendant combien de temps sans prendre trop de risques.

D'autre part, l'Atari possède un système de priorités d'interruptions. Ce système permet à une routine d'interruption d'en interrompre une autre ; mais pour que n'importe quelle routine ne puisse interrompre n'importe quelle autre, on a introduit dans l'Atari un ordre de priorité. Il serait en effet impensable que sur ordinateur toutes les fonctions aient le même degré d'importance au vu du fonctionnement du système ; un événement ne peut en interrompre un autre que s'il a un degré de priorité supérieur à ce dernier.



Reprenons le fonctionnement du processeur 68000: il a la possibilité de gérer des interruptions avec 7 degrés différents de priorité ; par hardware, dès qu'un événement survient, le processeur connaît la priorité de ce dernier ; la routine correspondante ne sera déclenchée que si aucun traitement de priorité supérieure n'est en cours à ce moment là ; pour connaître la priorité du traitement actuel il suffit de lire les trois bits des masques d'interruption dans le registre d'état (SR), décrits dans la table suivante :

priorité	bits des masques d'interruptions dans le registre d'état			
Indicateur de niveau de priorité IPL = *(Indicator of Priority Level)*				
	IPL2	IPL1	IPL0	
7	1	1	1	NMI interruption non masquable
6	1	1	0	
5	1	0	1	
4	1	0	0	
3	0	1	1	
2	0	1	0	
1	0	0	1	
0	0	0	0	Pas d'interruption

L'adresse de la routine qui se charge de gérer l'interruption, c'est-à-dire le vecteur d'interruption, doit être connue du composant périphérique du 68000 qui provoque l'interruption. Avec le système d'auto-vectorisation, le processeur peut déduire directement le vecteur du niveau d'interruption par la correspondance suivante :

Niveau	Numéro du vecteur	Adresse du vecteur
1	25	$64
2	26	$68
3	27	$6C
4	28	$70
5	29	$74
6	30	$78
7	31	$7C

Sur l'Atari STF, l'indicateur de niveau IPL0 est maintenu en permanence à zéro pour ne permettre que les niveaux pairs de priorité, à savoir 2, 4, 6.

Leurs fonctions sont les suivantes :

niveau 2 : retour de balayage horizontal (HBL)

niveau 4 : retour de balayage vertical (VBL)

niveau 6 : interruptions du MFP 68901.

Le niveau normal pour une interruption du processeur est 3 ; c'est la valeur que l'on trouve dans les bits du registre d'état (*State Register* ou SR) ; ceci afin d'empêcher l'interruption HBL de niveau 2 de se produire à chaque retour de ligne. Si c'était le cas, la perte de temps occasionnée par le HBL serait de 10 % en utilisation couleur et 22 % en monochrome, de quoi regarder votre Atari d'un mauvais œil ! Par défaut, le HBL remet le masque à 3 et revient au programme par un RTE. Cette interruption est générée toutes les 50 ou 64 microsecondes suivant le moniteur et peut être utilisée pour réaliser des effets visuels spéciaux puisque l'on connaît exactement la ligne d'écran qui vient d'être représentée ; le seul problème est le temps dont on dispose au cours d'une telle interruption et bien souvent on préfère utiliser le VBL qui assure déjà par ailleurs une multitude de tâches ; l'œil est en effet moins sensible à un décalage de temps sur une image écran que sur une visualisation de lignes successives qui risquent de déformer l'image.

Sous le niveau d'interruption 6, le MFP 68901 peut générer lui-même des vecteurs d'interruption. Le MFP 68901 gère à la fois les demandes de ses propres interruptions ainsi que celles de tous les périphériques. Il peut superviser 16 interruptions de priorités différentes, et ceci indépendamment de l'Atari ; 15 d'entre-elles sont disposées sur l'Atari, mais seulement 7 sont actives ; elles se trouvent toutes au niveau de priorité maximum (6) par rapport à l'Atari, mais ont des priorités classées à l'intérieur du MFP suivant le tableau :

Niveau	Actif	Fonction
15		Détecteur de moniteur monochrome
14		Ring Indicator de la RS232
13		Timer A du MFP
12	*	Tampon de réception de la RS232 plein
11	*	Erreur de réception de la RS232
10	*	Tampon d'émission de la RS232 vide
9	*	Erreur d'émission de la RS232
8		Compteur HBL, TIMER B
7		Contrôleur floppy et DMA
6	*	ACIAS du clavier et de la MIDI
5	*	fréquence 200 hz, TIMER C
4		Générateur de la fréquence d'émission et de réception de la RS232, TIMER D
3		inutilisé
2	*	CTS de la RS232
1		DCD de la RS232
0		Busy du Centronics

Les interruptions actives, sauf deux d'entre-elles, sautent à des adresses stockées dans un domaine attribué aux vecteurs d'exception non-autonomes dont voici les adresses :

Niveau	Vecteur	adr.routine	Fonction
2	66	$FC26B2/85540	CTS RS232
5		$FC2F78/8E06	TMER C
6		$FC281C/286AA	ACIAS
9	73	$FC2718/85A6	RS232 erreur émission
10	74	$FC2666/84F4	RS232 tampon émission vide
11	75	$FC26FA/8588	RS232 erreur réception
12	76	$FC2596/8424	RS232 tampon récept plein

Il nous reste maintenant à décrire les interruptions actives pour le système d'exploitation :

— NIVEAU 2, CTS (clear to send) de la RS232

Chaque fois qu'un récepteur connecté peut recevoir des données supplémentaires, la ligne CTS prévient l'interface RS232. La routine envoie le caractère suivant par le tampon d'émission de la RS232.

— NIVEAU 5, 200 HZ/timer C

Ce timer fonctionne sur une fréquence de 200 HZ ; la routine incrémente le compteur situé à l'adresse $4BA. C'est d'ailleurs ce que fait l'instruction T=TIMER de l'exemple du début, lire le mot long situé à cette adresse : faites t=LPEEK(&H4BA) et comparez. Tous les quatre appels de cette routine, donc sur une fréquence de 50 HZ, on appelle une routine qui traite le son et gère la répétition des touches enfoncées, puis on appelle la routine event-timer du GEM, initialisée par le vecteur $400.

— NIVEAU 6, Clavier et Midi

Les deux ACIAS sont placés sur le même niveau d'interruption. Pour savoir lequel des deux a demandé une interruption, les bits de demande d'interruption du registre d'état de l'ACIA sont testés et l'octet mémorisé (adresses de $FFFC00 à $FFFC06) ; si la donnée provient de la MIDI, l'octet est placé dans le tampon de réception ; si la donnée provient du clavier, le code de lecture est changé en code ASCII par le tableau de correspondance du clavier ; le traitement est analogue si les données proviennent de la souris ou de la manette.

— NIVEAU 9, Erreur d'émission de la RS232

Si au cours de l'émission de données de la RS232, on rencontre une erreur, le registre d'état de l'émetteur est lu et la description de cet état est transmise au bloc de paramètres de la RS232 afin de régénérer une validation éventuelle des données.

— NIVEAU 10, Tampon d'émission de la RS232 vide.

Dès qu'un octet a été émis par le MFP, le tampon se retrouve vide et cette interruption est exécutée ; le MFP est prêt à envoyer l'octet suivant.

— NIVEAU 11, Erreur de réception de la RS232.

Si au cours de la réception de données de la RS232, on rencontre une erreur, la routine efface le registre d'état du récepteur et revient à l'état initial. Les erreurs les plus communément rencontrées sont des erreurs de parité ou de dépassement de capacité.

— NIVEAU 12, Tampon de réception de la RS232 plein.

Lorsque le MFP a reçu un octet, on peut prendre le caractère correspondant et l'inscrire dans le tampon de réception, s'il y a de la place. On peut aussi instaurer un mode de Handshake avec accusés de réception en pré et post transferts avec XON/XOFF et RTS/CTS.

6/1.3.2

Programmation du générateur sonore

L'utilisation de mélodies, servant à la sonorisation de programme, nécessite l'emploi de fonctions ne ralentissant peu ou pas le traitement du programme principal. La fonction ~XBIOS (32,L:VARPTR(Chaine$)) permet de gérer une chaîne de données sous interruption (Voir Partie 3, chapitre 3, page 22). Cette chaîne est gérée automatiquement par l'ordinateur et n'entraîne donc aucune programmation particulière à l'intérieur d'un programme.

PROGRAMME DE CRÉATION DE MÉLODIES AVEC XBIOS (32)

Ce programme permet de créer une mélodie monophonique à l'aide du canal 1 du générateur sonore. Cette mélodie peut être sauvegardée sur disquette pour être ensuite utilisée à l'intérieur d'un autre programme, à la fin de ce programme, une démonstration pour le chargement des données vous est proposée.

La fonction XBIOS(32) n'a besoin que d'un seul paramètre, c'est l'adresse de la chaîne à exécuter. L'adresse est obtenue par VARPTR(chaîne$). L'adresse est codée sur 4 octets (mot long), l'instruction L: devant VARPTR est nécessaire.

Les données employées par l'instruction XBIOS(32) dans ce programme sont de deux sortes. Chacune des informations est constituée de deux octets.

Le premier octet est le type de la commande, le deuxième octet est la valeur attribuée à cette commande.

— Commande de 0 à 13

La commande représente le numéro de registre auquel l'octet suivant va être affecté.

Exemple :

07,255

Registre 07 à 255, tous les canaux sont fermés. L'équivalent avec la fonction SPOKE nous donnerait ceci :

```
SPOKE &Hff8800,07
SPOKE &Hff8802,255
```

Avec la fonction WAVE

```
WAVE 0
```

— Commande 130

Cette commande nous permet de gérer le temps d'exécution entre deux instructions. La valeur de l'attente est indiquée en 1/50 de seconde.

La valeur 130 peut être remplacée par n'importe quelle valeur comprise entre 130 et 255.

Si l'attente est égale à 0, le son est coupé, c'est la fin de la chaîne.

Exemple :

08,15,130,100,08,0,130,0

08,15 Volume du canal 1 à 15

130,100 Attente de 2 secondes avant la prochaine instruction

08,0 Volume du canal 1 à 0

130,0 Fin de la chaîne

```
'
'                        XBIOS(32)
'
'
DIM note$(12)
FOR a%=0 TO 12
  READ note$(a%)
  PRINT AT((a%+1)*6,10);note$(a%)
NEXT a%
DATA DO,DO#,RE,RE#,MI,FA,FA#,SOL,SOL#,LA,LA#,SI,DO
DIM haut(12)
FOR a%=0 TO 12
  READ haut(a%)
NEXT a%
DATA 856,807,762,719,679,641,605,571,539,508,480,453,428
DIM dureee$(4)
FOR a%=0 TO 4
  READ dureee$(a%)
  PRINT AT((a%+1)*10,14);dureee$(a%)
NEXT a%
DATA RONDE,BLANCHE,NOIRE,CROCHE,D. CROCHE
PRINT AT(15,18);"VALIDATION          JOUER          SILENCE"
affi=2
tempo=100
sui=0
SPOKE &HFF8800,8
SPOKE &HFF8802,15
SPOKE &HFF8800,7
SPOKE &HFF8802,255
piste$=CHR$(7)+CHR$(254)
DO
```

(suite)

```
    MOUSE x%,y%,k%
    IF k%=1
      IF y%>140
        IF y%>140 AND y%<160
          note
        ENDIF
        IF y%>200 AND y%<230
          durees
        ENDIF
        IF y%>270 AND y%<290
          IF x%>100 AND x%<190
            valider
          ENDIF
          IF x%>270 AND x%<320
            jouer
          ENDIF
          IF x%>400 AND x%<485
            silence
          ENDIF
        ENDIF
      ENDIF
      IF y%<40
        PRINT AT(30,2);"* SAUVEGARDER LA MELODIE *"
        sauvegarder
        PRINT AT(30,2);SPC(26)
      ENDIF
      REPEAT
      UNTIL MOUSEK=0
    ENDIF
    IF y%>40 AND y%<85 AND k%<>0
      modification
    ENDIF
    IF k%=2
      SPOKE &HFF8800,7
      SPOKE &HFF8802,255
    ENDIF
  LOOP
  PROCEDURE note
    flag_silence=0
    note=INT((x%-30)/48)
    IF flag_modification=0
      PRINT AT(20,20);note$(note);"  "
    ELSE
      buffer_affi=affi
      affi=(modifier*10)+2
      MID$(piste$,((sui+modifier)*6)+3,2)=CHR$(0)+CHR$(haut(note) AND 255)
      MID$(piste$,((sui+modifier)*6)+5,2)=CHR$(1)+CHR$(INT(haut(note)/256))

      MID$(piste$,((sui+modifier)*6)+7,2)=CHR$(133)+CHR$(tempo/(2^duree))
      affichage
      affi=buffer_affi
    ENDIF
    entendre_la_note
```

(suite)

```
RETURN
PROCEDURE durees
  duree=INT((x%-60)/80)
  IF flag_modification=0
    PRINT AT(20,22);dureeS$(duree);"          "
  ELSE
    buffer_affi=affi
    affi=(modifier*10)+2
    MID$(piste$,((sui+modifier)*6)+3,2)=CHR$(0)+CHR$(haut(note) AND 255)
    MID$(piste$,((sui+modifier)*6)+5,2)=CHR$(1)+CHR$(INT(haut(note)/256))

    MID$(piste$,((sui+modifier)*6)+7,2)=CHR$(133)+CHR$(tempo/(2^duree))
    affichage
    affi=buffer_affi
  ENDIF
RETURN
PROCEDURE valider
  IF flag_silence=0
    piste$=piste$+CHR$(0)+CHR$(haut(note) AND 255)
    piste$=piste$+CHR$(1)+CHR$(INT(haut(note)/256))
    piste$=piste$+CHR$(133)+CHR$(tempo/(2^duree))
  ELSE
    piste$=piste$+CHR$(7)+CHR$(255)
    piste$=piste$+CHR$(133)+CHR$(tempo/(2^duree))
    piste$=piste$+CHR$(7)+CHR$(254)
  ENDIF
  IF affi>80
    affi=2
    ADD sui,8
    PRINT AT(2,4);SPC(78)
    PRINT AT(2,5);SPC(78)
    COLOR 0
    LINE buffer_a_modifier*80,85,(buffer_a_modifier+1)*80,85
    modification=0
  ENDIF
  affichage
  ADD affi,10
RETURN
PROCEDURE entendre_la_note
  SPOKE &HFF8800,0
  SPOKE &HFF8802,(haut(note) AND 255)
  SPOKE &HFF8800,1
  SPOKE &HFF8802,INT(haut(note)/256)
  SPOKE &HFF8800,7
  SPOKE &HFF8802,254
RETURN
PROCEDURE jouer
  a_jouer$=piste$+CHR$(7)+CHR$(255)+CHR$(255)+CHR$(0)
  ~XBIOS(32,L:VARPTR(a_jouer$))
RETURN
```



(suite)

```
PROCEDURE silence
  flag_silence=1
  IF flag_modification=0
    PRINT AT(20,20);"SILENCE"
  ELSE
    buffer_affi=affi
    affi=(modifier*10)+2
    MID$(piste$,((sui+modifier)*6)+3,2)=CHR$(7)+CHR$(255)
    MID$(piste$,((sui+modifier)*6)+5,2)=CHR$(133)+CHR$(tempo/(2^duree))
    MID$(piste$,((sui+modifier)*6)+7,2)=CHR$(7)+CHR$(254)
    affichage
    affi=buffer_affi
  ENDIF
  SPOKE &HFF8800,7
  SPOKE &HFF8802,255
RETURN
PROCEDURE affichage
  IF flag_silence=0
    PRINT AT(affi,4);note$(note);SPC(10-(LEN(note$(note))))
  ELSE
    PRINT AT(affi,4);"SILENCE"
  ENDIF
  PRINT AT(affi,5);durees$(duree);SPC(10-(LEN(durees$(duree))))
RETURN
PROCEDURE modification
  modifier=INT(x%/80)
  IF MID$(piste$,((sui+modifier)*6)+3,1)=CHR$(7)
    silence=1
  ENDIF
  IF k%=1 AND sui+modifier<(LEN(piste$)-2)/6
    COLOR 0
    LINE buffer_a_modifier*80,85,(buffer_a_modifier+1)*80,85
    COLOR 1
    LINE modifier*80,85,(modifier+1)*80,85
    buffer_a_modifier=modifier
    SPOKE &HFF8800,0
    SPOKE &HFF8802,ASC(MID$(piste$,((sui+modifier)*6)+4,1))
    SPOKE &HFF8800,1
    SPOKE &HFF8802,ASC(MID$(piste$,((sui+modifier)*6)+6,1))
    SPOKE &HFF8800,7
    SPOKE &HFF8802,254
    flag_modification=1
  ENDIF
  IF k%=2
    COLOR 0
    LINE buffer_a_modifier*80,85,(buffer_a_modifier+1)*80,85
    flag_modification=0
  ENDIF
RETURN
PROCEDURE sauvegarder
  FILESELECT "*.*","",le_fichier$
  IF le_fichier$<>"\" AND le_fichier$>""
    a_jouer$=piste$+CHR$(7)+CHR$(255)+CHR$(255)+CHR$(0)
    BSAVE le_fichier$,VARPTR(a_jouer$),LEN(a_jouer$)
  ENDIF
RETURN
```

La présentation du logiciel est volontairement très sommaire afin de ne pas compliquer le programme.

Aucune des instructions SOUND ou WAVE ne sont utilisées, elles sont remplacées par des SPOKE.

LES VARIABLES

— Le tableau note$()

Il permet un affichage de la hauteur de note par indice.

— Le tableau haut()

Les valeurs utilisées correspondent à l'octave 3 de l'instruction SOUND. Pour transposer d'un octave vers le haut, il faut diviser chacune des données par 2, pour 2 octaves, diviser par 4.

Pour transposer d'un octave vers le bas, il faut multiplier chacune des valeurs par 2.

— Le tableau durée$()

Il permet le choix d'une durée par multiple de deux. La ronde est la plus grande durée, la blanche vaut la moitié de la ronde, la noire la moitié de la blanche etc.

D. croche veut dire double croche.

— Affi = 2

Affi est le décalage pour l'affichage des notes à l'écran, 2 est la deuxième colonne, à chaque nouvel affichage, affi augmente de 10.

— Tempo = 100

Tempo est la variable gérant la vitesse d'exécution de la mélodie. Plus la variable tempo est élevée, plus la mélodie sera jouée lentement.

— Sui = 0

La variable sui est le décalage dû au passage à la page suivante pour entrer les informations de notes. La variable sui augmente de 8 à chaque page.

Initialisation du générateur sonore :

```
SPOKE &HFF8800,8
SPOKE &HFF8802,15
SPOKE &HFF8800,7
SPOKE &HFF8802,255
```

Volume voix 1 à 15, toutes les voix coupées.

— piste$ = CHR$(7) + CHR$(254)

Ouverture de la voix 1 à l'aide du registre 7.

— DO__LOOP

Entre les instructions do/loop sont gérés les différents branchements en fonction des zones de cliquage.

— PROCEDURE note

Affichage de la note si le flag__modification est à 0. Sinon affichage en haut de la note sélectionnée.

Remplacement des données dans la chaîne piste$ avec sui pour la page et modifier pour le numéro de la note sélectionnée. La multiplication par 6 permet un accès à chacune des notes, +2 nous donne le décalage occasionné par l'initialisation de piste$ au début du programme.

— PROCEDURE durees

Affichage de la durée si le flag__modification est à 0. Sinon affichage en haut de la durée sélectionnée.

— PROCEDURE valider

Mise au bout de la chaîne piste$ des nouvelles données sélectionnées par note et durée. Si un silence est sélectionné, la construction de la chaîne est différente, arrêt de la voix 1, durée du silence, ouverture de la voix 1.

Si affi > 80 (bout de ligne), sui est augmenté de 8, affi = 2, effacement des données précédentes.

— PROCEDURE entendre_la_note

Envoi des données vers le générateur sonore, la hauteur de la note étant codée sur deux octets distincts, un calcul nous oblige à séparer le poids fort du poids faible.

— PROCEDURE jouer

Envoi de la chaîne dans XBIOS(32), à la chaîne piste$ s'ajoute des valeurs qui nous servent à couper le son du générateur sonore, puis signifier la fin de la chaîne.

— PROCEDURE silence

Entrée des données pour un silence, puis suppression du son.

— PROCEDURE affichage

Affichage des données en haut de l'écran.

— PROCEDURE modification

Recherche de la note à modifier en fonction de sui et de la position de x%.

— PROCEDURE sauvegarder

Sauvegarde du fichier avec création d'une fin de chaîne.

PROGRAMME DE CHARGEMENT DES DONNÉES

Ce programme permet de charger des données fabriquées avec le programme précédent. La fonction TRIM$() permet d'ajuster la longueur de la chaîne au nombre de données chargées.

La chaîne fin$ permet d'interrompre le déroulement du morceau. CIRCLE est le programme principal.

```
'
'          Chargement de fichiers pour XBIOS(32)
'
'
piste$=SPACE$(32676)
fin$=CHR$(7)+CHR$(255)+CHR$(255)+CHR$(0)
FILESELECT "\*.*","",s$
BLOAD s$,VARPTR(piste$)
piste$=TRIM$(piste$)+CHR$(0)
SPOKE &HFF8800,8
SPOKE &HFF8802,15
~XBIOS(32,L:VARPTR(piste$))
DO
  IF MOUSEK=1
    ~XBIOS(32,L:VARPTR(piste$))
  ENDIF
  IF MOUSEK=2
    ~XBIOS(32,L:VARPTR(fin$))
  ENDIF
  CIRCLE RANDOM(640),RANDOM(399),RANDOM(50)
LOOP
```

6/1.3.3

Programmation d'un délai et d'un écho

Le délai fait partie de ce que l'on appelle couramment en musique, un effet. L'effet de délai se caractérise par un décalage de l'événement dans le temps.

Ce premier exemple non musical vous permet d'apprécier cet effet sur les touches du clavier.

```
DO
  IF INP?(2)
    a| =INP(2)
    PRINT a| ;
    PAUSE 20
    PRINT CHR$(a| )
  ENDIF
LOOP
```

Après analyse du port d'entrée clavier, le code ASCII est sorti sur l'écran, puis le caractère ASCII du code s'affiche après un temps de pause.

Dans notre exemple la pause interdit l'usage du clavier avant la deuxième instruction **PRINT**, ce qui en limite l'effet.

L'effet de délai implique obligatoirement la mémorisation de l'événement principal. Deux systèmes de mémorisation sont possibles.

PREMIER SYSTÈME DE MÉMORISATION

Le stockage des données s'effectue soit par l'intermédiaire d'une chaîne de caractères ou d'un tableau suivant le format des éléments à mémoriser.

Dans le délai la notion de temps est importante, c'est pourquoi il faut mémoriser non seulement la donnée mais aussi l'heure de l'événement. Le timer de l'Atari peut nous aider à cela.

Pour ce programme nous allons utiliser une chaîne de caractères pour les données et un tableau pour le timer.

Ce programme autorise le mélange des données entre l'entrée et la lecture des données.

Deux pointeurs sont nécessaires pour gérer le délai :

• Un pointeur pour le numéro d'entrée (**pointeur_entrée**) qui mémorise la donnée dans **mem$** et le timer dans **temp()**, ce pointeur étant incrémenté après chaque entrée.

• Un pointeur sur le prochain événement à interpréter par l'effet (**pointeur_delai**), ce pointeur étant incrémenté après chaque lecture.

La variable délai gère le temps de répétition des événements.

```
mem$=SPACE$(50)
DIM temps(50)
pointeur_entree=1
pointeur_delai=1
delai=200
DO
  IF INP?(2)
    a| =INP(2)
    MID$(mem$,pointeur_entree,1)=CHR$(a| )
    PRINT a| ''
    temps(pointeur_entree)=TIMER
    INC pointeur_entree
  ENDIF
  IF pointeur_delai<pointeur_entree
    IF TIMER-temps(pointeur_delai)>delai
      PRINT MID$(mem$,pointeur_delai,1)''
      INC pointeur_delai
    ENDIF
  ENDIF
LOOP
```

Le problème de ce programme est la limite par la taille de la chaîne de caractères et du tableau, après 50 entrées, l'interpréteur vous donne un magnifique **index de champ trop grand** causé par le dépassement de longueur de la chaîne de caractères. Ce système est donc limitatif dans son exploitation, la taille maximale d'une chaîne étant de 32 767 éléments. Une des possibilités est la réorganisation de la chaîne et du tableau dès qu'une valeur X est atteinte.

L'exemple suivant est musical, il est identique au précédent mais effectue une réorganisation dès que l'événement délai est exécuté, ce qui nous donne un buffer maximal de 49 données. Au-delà de cette limite le message d'erreur apparaît.

L'incrément du **pointeur_delai** est supprimé puisque sa valeur sera toujours à 1.

L'entrée des notes s'effectue par le clavier de l'Atari.

L'instruction **DELETE** temps(0) effectue un décalage de 1 vers la gauche sur tout le tableau à partir de l'indice 0.

L'instruction **LSET mem$=RIGHT$(mem$,49)** déplace vers la gauche le contenu de la chaîne **mem$** de 1 du deuxième caractère au cinquantième.

Les instructions **PRINT** permettent de comparer l'état de la chaîne des données.

```
mem$=SPACE$(50)
DIM temps(50)
pointeur_entree=1
delai=200
DO
  IF INP?(2)
    a| =INP(2)
    MID$(mem$,pointeur_entree,1)=CHR$(a| )
    WAVE 1
    SOUND 1,15,MOD(a| ,11)+1,3,2
    WAVE 0
    temps(pointeur_entree)=TIMER
    INC pointeur_entree
  ENDIF
  IF pointeur_entree>1
    IF TIMER-temps(1)>delai
      WAVE 1
      SOUND 1,15,MOD(ASC(MID$(mem$,1,1)),11)+1,3
      DELETE temps(1)
      DEC pointeur_entree
      PRINT mem$
      LSET mem$=RIGHT$(mem$,49)
      PRINT mem$
      WAVE 0
    ENDIF
  ENDIF
LOOP
```

DEUXIÈME SYSTÈME DE MÉMORISATION

Le deuxième système utilise un buffer cyclique de même type que celui du clavier ou de l'entrée MIDI de l'Atari.

Principe du buffer :

Une zone mémoire est réservée afin de fixer l'emplacement du buffer et des pointeurs. Cette mémoire est réservée par l'instruction **MALLOC(tail%)**,

x% est l'adresse de début de buffer, la variable tail% est la longueur du buffer.

Trois pointeurs sont nécessaires à la gestion du buffer, le pointeur d'adresse de base du buffer **x%**, le pointeur d'entrée des données **p_in%**, le pointeur de sortie des données **p_out%**.

La variable posi% ainsi que toutes les lignes marquées d'un ! servent uniquement à l'affichage des données.

Dans ce programme c'est l'entrée des données dans le buffer qui est utilisée, aucune lecture n'est prévue.

```
tail%=10
x%=MALLOC(tail%)
p_in%=x%
p_out%=x%
posi%=0                                    !
DO
  IF INP?(2)
    POKE p_in%,INP(2)
    INC p_in%
    IF p_in%>=x%+tail%
      p_in%=x%
    ENDIF
    PRINT AT(6,12);SPACE$(70)              !
    PRINT AT((posi%+1)*6,12);"^"           !
    INC posi%                              !
    posi%=MOD(posi%,tail%)
  ENDIF
  FOR a%=0 TO tail%-1                      !
    PRINT AT((a%+1)*6,10);PEEK(x%+a%)'     !
  NEXT a%                                  !
LOOP
```

Dans le programme suivant, la taille du buffer est de 240 octets, soit 40 emplacements mémoire. Les données sont codées sur six octets, les deux premiers octets représentent la donnée, alors que les quatre suivants représentent le timer.

Dans ce programme, les pointeurs ne sont pas des adresses mémoire mais des offsets par rapport à l'adresse de base du buffer. Une quatrième variable est **nb_out**, c'est le nombre de données qu'il reste à interpréter.

A la sortie du programme, la procédure fin libère la place mémoire réservée.

Le temps de délai est programmable par la souris.

```
'
'              Delai avec gestion
'
'                 d'un buffer
'
'               générateur ATARI
'
'
tail=240                ! taille du buffer pour 40 emplacements notes
x%=MALLOC(tail)         ! adresse du buffer mémoire
p_in=0                  ! pointeur d'entrée
nb_out=0                ! pointeur nombre de données à jouer
p_out=0                 ! pointeur adresse à jouer
delai=600
PRINT AT(10,2);"Montez le volume du moniteur et appuyez sur les
touches"
PRINT AT(10,4);"Pour modifier le temps de délai, actioner la souris"
PRINT AT(20,10);"Délai en secondes:    ";USING "###.##",delai/200
ON BREAK GOSUB fin
DO
  k|=MOUSEK
  IF k|=1
    DEC delai
    delai=MAX(delai,10)
    PRINT AT(41,10);USING "###.##",delai/200
  ENDIF
  IF k|=2
    INC delai
    delai=MIN(delai,3000)   ! 15 secondes
    PRINT AT(41,10);USING "###.##",delai/200
  ENDIF
  lecture
  IF INP?(2)
    IF nb_out*6<tail
      DPOKE x%+p_in,INP(2)
      LPOKE x%+p_in+2,TIMER
      WAVE 1
      SOUND 1,15,MOD(DPEEK(x%+p_in),12),4,1
      WAVE 0
      ADD p_in,6
      INC nb_out
      p_in=MOD(p_in,tail)
    ENDIF
  ENDIF
LOOP
```

(suite)

```
PROCEDURE lecture
  IF nb_out<>0
    p_out=MOD(p_out,tail)
    IF LPEEK(x%+p_out+2)+delai<TIMER
      WAVE 1
      SOUND 1,15,MOD(DPEEK(x%+p_out),12),4,1
      WAVE 0
      ADD p_out,6
      DEC nb_out
    ENDIF
  ENDIF
RETURN
PROCEDURE fin
  g%=MFREE(x%)
  END
RETURN
```

Ce programme de délai nous offre la possibilité de créer un effet d'écho. Pour cela il faut multiplier les différents pointeurs ainsi que les procédures lecture. En plus de la répétition des informations, le programme d'écho atténue le volume de chaque répétition.

Les deux derniers programmes peuvent facilement être adaptés à l'une de vos créations. Outre le délai des notes, il est facile de changer la hauteur des notes répétées, en modifiant la hauteur note dans l'instruction **SOUND** de chacune des procédures lecture.

Exemple :

```
SOUND 1,9,MOD(DPEEK(x% + p_out2) + modif_hauteur,12),4,1
```

La variable **modif_hauteur** effectue la transposition de la note.

```
'
'               Echo avec gestion
'
'               d'un buffer
'
'               générateur ATARI
'
'
'
'
tail=240                          ! taille du buffer pour 40
emplacements notes
x%=MALLOC(tail)                   ! adresse du buffer mémoire
p_in=0                            ! pointeur d'entrée
nb_out=0                          ! pointeur nombre de données à jouer
p_out=0                           ! pointeur adresse à jouer
p_in2=0                           ! pointeur d'entrée
nb_out2=0                         ! pointeur nombre de données à jouer
p_out2=0                          ! pointeur adresse à jouer
p_in3=0                           ! pointeur d'entrée
nb_out3=0                         ! pointeur nombre de données à jouer
p_out3=0                          ! pointeur adresse à jouer
delai=100
PRINT AT(10,2);"Montez le volume du moniteur et appuyez sur les
touches"
PRINT AT(10,4);"Pour modifier le temps de délai, actioner la souris"
PRINT AT(21,10);"Echo en secondes:   ";USING "###.##",delai/200
ON BREAK GOSUB fin
DO
  k|=MOUSEK
  IF k|=1
    DEC delai
    delai=MAX(delai,10)
    PRINT AT(41,10);USING "###.##",delai/200
  ENDIF
  IF k|=2
    INC delai
    delai=MIN(delai,300)   ! 15 secondes
    PRINT AT(41,10);USING "###.##",delai/200
  ENDIF
  lecture1
  lecture2
  lecture3
  IF BIOS(1,2)
    IF nb_out*6<tail
      DPOKE x%+p_in,INP(2)
      LPOKE x%+p_in+2,TIMER
      WAVE 1
      SOUND 1,15,MOD(DPEEK(x%+p_in),12),4,1
      WAVE 0
      ADD p_in,6
      INC nb_out
```

(suite)

```
      p_in=MOD(p_in,tail)
      ADD p_in2,6
      INC nb_out2
      p_in2=MOD(p_in2,tail)
ADD p_in3,6
      INC nb_out3
      p_in2=MOD(p_in3,tail)
    ENDIF
  ENDIF
LOOP
PROCEDURE lecture1
  IF nb_out<>0
    p_out=MOD(p_out,tail)
    IF LPEEK(x%+p_out+2)+delai<TIMER
      WAVE 1
      SOUND 1,12,MOD(DPEEK(x%+p_out),12),4,1
      WAVE 0
      ADD p_out,6
      DEC nb_out
    ENDIF
  ENDIF
RETURN
PROCEDURE lecture2
  IF nb_out2<>0
    p_out2=MOD(p_out2,tail)
    IF LPEEK(x%+p_out2+2)+delai+delai<TIMER
      WAVE 1
      SOUND 1,9,MOD(DPEEK(x%+p_out2),12),4,1
      WAVE 0
      ADD p_out2,6
      DEC nb_out2
    ENDIF
  ENDIF
RETURN
PROCEDURE lecture3
  IF nb_out3<>0
    p_out3=MOD(p_out3,tail)
    IF LPEEK(x%+p_out3+2)+delai+delai+delai<TIMER
      WAVE 1
      SOUND 1,5,MOD(DPEEK(x%+p_out3),12),4,1
      WAVE 0
      ADD p_out3,6
      DEC nb_out3
    ENDIF
  ENDIF
RETURN
PROCEDURE fin
  g%=MFREE(x%)
  END
RETURN
```

6/1.3.4

Utilisation du timer A pour générer une suite de notes sous interruption

Le timer A est l'un des quatre timers disponibles sur le MFP 68091 ; les timers A et B sont potentiellement identiques et contiennent chacun un diviseur de fréquence programmable et un compteur de 8 bits fonctionnant par décrémentation ; ces deux valeurs de départ sont mises respectivement dans les octets d'appellation d'origine **tacr** et **tadr** ; la valeur 7 est mise dans **tacr**, ce qui configure le timer A dans le mode Delay et une prédivision de fréquence de 200 ; la valeur maximale est aussi attribuée au compteur, soit 255, ce qui fixera le timer A sur sa vitesse minimale. Sur la base de sa fréquence, le timer A générera une interruption chaque fois que son compteur atteindra 0, la décrémentation de ce compteur s'effectuant à chaque impulsion de fréquence. Vous pouvez vous amuser à modifier ces valeurs en restant dans le mode Delay (valeurs de 1 à 7 pour **tacr**) pour accélérer la génération des notes, jusqu'à éventuellement planter le système si cette interruption revient trop fréquemment et si son traitement est trop long.

Dans l'initialisation du timer A, il faut ensuite mettre à un le bit 5 des registres **iera**, **ipra**, **isra**, **imra**, ce numéro de bit correspondant au timer A, afin de valider l'interruption timer A, permettre son traitement prioritaire par software, masquer et démasquer les interruptions moins prioritaires et pouvoir démasquer séparément la source d'interruption timer A.

Dans le sous-programme d'initialisation, on positionne le volume du canal 1 de façon définitive à sa valeur maximale et on transfère l'adresse du vecteur d'interruption correspondant au niveau 13 du MFP (timer A) à l'adresse **timaint**, étiquette du traitement de l'interruption du timer A. Tout ceci doit évidemment se faire en mode superviseur.

L'interruption elle-même masque les bits du registre d'état, incrémente un compteur qui sert à comptabiliser le nombre de passages dans l'interruption. Le compteur plus, quant à lui, sert à générer une suite de hauteurs de notes décroissantes de façon cyclique. La fin du traitement ne fait que démasquer les bits d'interruption du timer A et repositionner les bits de priorité.

Ce petit programme de génération sonore fonctionnant sous interruption, il est nécessaire qu'il puisse interrompre un programme principal, qui n'est ici que pour la forme et qui inscrit des A sur l'écran jusqu'à ce qu'on soit passé un certain nombre de fois, ici 500, dans **timaint**.

Une fois le traitement effectué, il faut désactiver l'interruption, remettre à zéro le bit 5 des registres **iera**, **imra**, **ipra**, **isra**, repositionner **tacr** et **tadr** à leur valeur d'origine et mettre le volume à zéro sur le canal 1, avant de rendre la main au système.

```
                .DATA
;initialisation du timer A en mode superviseur

tima:       clr.l       -(sp)
            move.w      #$20,-(sp)
            trap        #1
            addq.l      #6,sp
            move.l      d0,sassp

            move.b      #8,$ff8800
            move.b      #15,$ff8802

            move.b      #7,tacr             ;mode et fréquence
                                           ;du ttimer A
            move.b      #255,tadr           ;data du timer A
                                            ;modulo
            bset.b      #5,iera             ; init des bits
                                            ;interrupt
            bset.b      #5,imra
            bset.b      #5,ipra
            bset.b      #5,isra

            move.l      #timaint,tavec      ;transfert de
                                            ;l'adresse vecteur
                                            ; interruption

            move.l      sassp,-(sp)         ;retour du mode
                                            ;superviseur
            move.w      #$20,-(sp)
            trap        #1
            addq.l      #6,sp

            rts


ipra        equ $fffa0b
iera        equ $fffa07
isra        equ $fffa0f
imra        equ $fffa13
tacr        equ $fffa19
tadr        equ $fffa1f
tavec       equ $134
```

(suite)

```
timaint:  ori.w     #$600,sr            ;masquage des bits
                                        ;interrupt
          addq.l    #1,compteur
          addq.b    #1,plus
          cmpi.b    #255,plus
          bne.s     suite
          clr.b     plus
suite:    move.b    #0,$ff8800
          move.b    plus,$ff8802
          bclr.b    #5,ipra
          bclr.b    #5,isra
          ori.b     #$df,ipra           ;démasquage des bits
                                        ;interr a
          ori.b     #$df,isra
          andi.w    #$f3ff,sr           ;reposition des bits
                                        ;de priorité
          rte



compteur: .DC.l 0
plus:     .DC.b 0
          .EVEN


          .BSS

sassp:    .DS.l 1


          .TEXT

          bsr       tima

boucle:   move.w    #65,-(sp)
          move.w    #2,-(sp)
          trap      #1
          addq.l    #4,sp
vboucle:  cmpi.l    #500,compteur
          bls       boucle
```

(suite)

```
;---------------DESACTIVATION DE L'INTERRUPTION---------

timoff:   clr.l     -(sp)
          move.w    #$20,-(sp)
          trap      #1
          addq.l    #6,sp
          move.l    d0,sassp
          move.b    #0,tacr
          move.b    #2,tadr
          bclr.b    #5,iera
          bclr.b    #5,imra
          bclr.b    #5,ipra
          bclr.b    #5,isra

          move.b    #8,$ff8800
          move.b    #0,$ff8802

          move.l    sassp,-(sp)
          move.w    #$20,-(sp)
          trap      #1
          addq.l    #6,sp

fin:      move.w    #0,-(sp)            ;rendre la main
          trap      #1
          .END
```

Le chargeur hexadécimal Basic du programme
Génération suite notes timer A

```
$682C6:[ 6100 007e                    ]      bsr     $68346
$682CA:[ 3f3c 0041                    ]      move.w  #$41,-(a7)
$682CE:[ 3f3c 0002                    ]      move.w  #$2,-(a7)
$682D2:[ 4e41                         ]      trap    #$1
$682D4:[ 588f                         ]      addq.l  #$4,a7
$682D6:[ 0cb9 000001f4 00068408       ]      cmpi.l  #$1f4,$68408
$682E0:[ 63e8                         ]      bls.s   $682ca
$682E2:[ 42a7                         ]      clr.l   -(a7)
$682E4:[ 3f3c 0020                    ]      move.w  #$20,-(a7)
$682E8:[ 4e41                         ]      trap    #$1
$682EA:[ 5c8f                         ]      addq.l  #$6,a7
$682EC:[ 23c0 0006840e                ]      move.l  d0,$6840e
$682F2:[ 13fc 0000 00fffa19           ]      move.b  #$0,$fffa19
$682FA:[ 13fc 0002 00fffa1f           ]      move.b  #$2,$fffa1f
$68302:[ 08b9 0005 00fffa07           ]      bclr    #$5,$fffa07
$6830A:[ 08b9 0005 00fffa13           ]      bclr    #$5,$fffa13
$68312:[ 08b9 0005 00fffa0b           ]      bclr    #$5,$fffa0b
$6831A:[ 08b9 0005 00fffa0f           ]      bclr    #$5,$fffa0f
$68322:[ 13fc 0008 00ff8800           ]      move.b  #$8,$ff8800
$6832A:[ 13fc 0000 00ff8802           ]      move.b  #$0,$ff8802
$68332:[ 2f39 0006840e                ]      move.l  $6840e,-(a7)
$68338:[ 3f3c 0020                    ]      move.w  #$20,-(a7)
$6833C:[ 4e41                         ]      trap    #$1
$6833E:[ 5c8f                         ]      addq.l  #$6,a7
$68340:[ 3f3c 0000                    ]      move.w  #$0,-(a7)
$68344:[ 4e41                         ]      trap    #$1
$68346:[ 42a7                         ]      clr.l   -(a7)
$68348:[ 3f3c 0020                    ]      move.w  #$20,-(a7)
$6834C:[ 4e41                         ]      trap    #$1
$6834E:[ 5c8f                         ]      addq.l  #$6,a7
$68350:[ 23c0 0006840e                ]      move.l  d0,$6840e
$68356:[ 13fc 0008 00ff8800           ]      move.b  #$8,$ff8800
$6835E:[ 13fc 000f 00ff8802           ]      move.b  #$f,$ff8802
$68366:[ 13fc 0007 00fffa19           ]      move.b  #$7,$fffa19
$6836E:[ 13fc 00ff 00fffa1f           ]      move.b  #$ff,$fffa1f
$68376:[ 08f9 0005 00fffa07           ]      bset    #$5,$fffa07
$6837E:[ 08f9 0005 00fffa13           ]      bset    #$5,$fffa13
$68386:[ 08f9 0005 00fffa0b           ]      bset    #$5,$fffa0b
$6838E:[ 08f9 0005 00fffa0f           ]      bset    #$5,$fffa0f
$68396:[ 23fc 000683b0 00000134       ]      move.l  #$683b0,$134
$683A0:[ 2f39 0006840e                ]      move.l  $6840e,-(a7)
$683A6:[ 3f3c 0020                    ]      move.w  #$20,-(a7)
$683AA:[ 4e41                         ]      trap    #$1
$683AC:[ 5c8f                         ]      addq.l  #$6,a7
$683AE:[ 4e75                         ]      rts
$683B0:[ 007c 0600                    ]      ori.w   #$600,sr
$683B4:[ 52b9 00068408                ]      addq.l  #$1,$68408
$683BA:[ 5239 0006840c                ]      addq.b  #$1,$6840c
$683C0:[ 0c39 00ff 0006840c           ]      cmpi.b  #$ff,$6840c
$683C8:[ 6606                         ]      bne.s   $683d0
```

(suite)

```
$683CA:[ 4239 0006840c          ]      clr.b    $6840c
$683D0:[ 13fc 0000 00ff8800     ]      move.b   #$0,$ff8800
$683D8:[ 13f9 0006840c 00ff8802 ]      move.b   $6840c,$ff8802
$683E2:[ 08b9 0005 00fffa0b     ]      bclr     #$5,$fffa0b
$683EA:[ 08b9 0005 00fffa0f     ]      bclr     #$5,$fffa0f
$683F2:[ 0039 00df 00fffa0b     ]      ori.b    #$df,$fffa0b
$683FA:[ 0039 00df 00fffa0f     ]      ori.b    #$df,$fffa0f
$68402:[ 027c f3ff              ]      andi.w   #-$c01,sr
$68406:[ 4e73                   ]      rte
$68408:[ 0000 0000              ]      ori.b    #$0,d0
$6840C:[ 0000 0000              ]      ori.b    #$0,d0
$68410:[ 0000 0000              ]      ori.b    #$0,d0
```

6/1.3.5

Arpégiateur sous VBL

Le principe de ce petit programme est d'utiliser une interruption déjà existante, le **VBL**, et de lui adjoindre un traitement supplémentaire.

La fréquence du **VBL** étant, suivant les moniteurs branchés, de 50 hz ou 70 hz, il est nécessaire, sauf volonté de vouloir faire jouer des arpèges très rapides, de ne pas générer un son à chaque appel de la routine d'interruption ; ici, c'est tous les huit appels du **VBL** que le son est produit par l'intermédiaire du compteur **compt1**.

Le compteur **compt2** permet de générer des hauteurs croissantes grâce à une décrémentation itérative de valeur 8 de ce compteur dans le registre 0 de la puce son.

La fonction **Setexec,** (Bios(5)) modifie le vecteur n° 28, c'est-à-dire celui qui correspond au **VBL** et place à la même adresse, l'adresse de la nouvelle routine **vblir**. L'ancienne adresse est sauvegardée dans **oldvbl** et il suffira de la placer sur la pile superviseur en fin de traitement pour pouvoir réaliser les anciennes fonctions de l'interruption **VBL**.

Le programme principal interrompu est simplement une boucle de comptage simple.

Notez bien que la vitesse maximale de l'arpège correspond à la période du **VBL** et que le temps séparant deux notes ne peut être qu'une valeur multiple entière de cette période.

En fin de programme il est nécessaire de repositionner le vecteur **VBL** en son adresse originale au moyen de la même fonction setexec et de remettre le volume du canal 1 de la puce son à zéro pour ne pas devoir arrêter le son en frappant une touche du clavier.

```
            .DATA
vblvec      equ 28
compt:      .DC.l 0
compt1:     .DC.b 0
compt2:     .DC.b 0

vblir:      movem.l   d0-d2/a0-a2,-(sp)    ;sauvegarde des
                                           ;registres
            addq.b    #1,compt1
            cmpi.b    #8,compt1
            beq.s     suite
            bra.s     suite1
suite:      move.b    #0,compt1
            move.b    #8,$ff8800
            move.b    #15,$ff8802
            move.b    #0,$ff8800
            move.b    compt2,$ff8802
            subq.b    #8,compt2
suite1:     movem.l   (sp)+,d0-d2/a0-a2
            move.l    oldvbl,-(sp)         ;se brancher sur
                                           ;l'ancienneroutine
                                           ;VBL
            rts

;placer la nouvelle adresse du traitement

init:       move.l    #vblir,-(sp)
            move.w    #vblvec,-(sp)
            move.w    #5,-(sp)
            trap      #13
            addq.l    #8,sp
            move.l    d0,oldvbl            ;sauvegarde de
                                           ;l'ancien vecteur
            rts

            .BSS

oldvbl:     .DS.l 1
sassp:      .DS.l 1

            .TEXT

            bsr       init

boucle:     addq.l    #1,compt
            cmpi.l    #1000000,compt
            bls.s     boucle
```

(suite)

```
;-----------------repositionner l'ancienne adresse:

move.l    oldvbl,   -(sp)
          move.w    #vblvec,-(sp)
          move.w    #5,-(sp)
          trap      #13
          addq.l    #8,sp

; passer en mode superviseur

          clr.l     -(sp)
          move.w    #$20,-(sp)
          trap      #1
          addq.l    #6,sp
          move.l    d0,sassp

          move.b    #8,$ff8800
          move.b    #0,$ff8802

          move.l    sassp,-(sp)          ;retour du mode
                                         ;superviseur
          move.w    #$20,-(sp)
          trap      #1
          addq.l    #6,sp

fin:      move.w    #0,-(sp)             ;rendre la main
          trap      #1                   ;au système
          .END
```

Le chargeur hexadécimal Basic du programme Arpégiateur sous VBL

```
$682C2:[ 6100 00b2                ]        bsr       $68376
$682C6:[ 52b9 0006831e            ]        addq.l    #$1,$6831e
$682CC:[ 0cb9 000f4240 0006831e   ]        cmpi.l    #$f4240,$6831e
$682D6:[ 63ee                     ]        bls.s     $682c6
$682D8:[ 2f39 00068390            ]        move.l    $68390,-(a7)
$682DE:[ 3f3c 001c                ]        move.w    #$1c,-(a7)
$682E2:[ 3f3c 0005                ]        move.w    #$5,-(a7)
$682E6:[ 4e4d                     ]        trap      #$d
$682E8:[ 508f                     ]        addq.l    #$8,a7
$682EA:[ 42a7                     ]        clr.l     -(a7)
$682EC:[ 3f3c 0020                ]        move.w    #$20,-(a7)
$682F0:[ 4e41                     ]        trap      #$1
$682F2:[ 5c8f                     ]        addq.l    #$6,a7
$682F4:[ 23c0 00068394            ]        move.l    d0,$68394
$682FA:[ 13fc 0008 00ff8800       ]        move.b    #$8,$ff8800
$68302:[ 13fc 0000 00ff8802       ]        move.b    #$0,$ff8802
$6830A:[ 2f39 00068394            ]        move.l    $68394,-(a7)
$68310:[ 3f3c 0020                ]        move.w    #$20,-(a7)
$68314:[ 4e41                     ]        trap      #$1
$68316:[ 5c8f                     ]        addq.l    #$6,a7
$68318:[ 3f3c 0000                ]        move.w    #$0,-(a7)
$6831C:[ 4e41                     ]        trap      #$1
$6831E:[ 0000 0000                ]        ori.b     #$0,d0
$68322:[ 0000 48e7                ]        ori.b     #$e7,d0
$68326:[ e0e0                     ]        asr       -(a0)
$68328:[ 5239 00068322            ]        addq.b    #$1,$68322
$6832E:[ 0c39 0008 00068322       ]        cmpi.b    #$8,$68322
$68336:[ 6702                     ]        beq.s     $6833a
$68338:[ 6030                     ]        bra.s     $6836a
$6833A:[ 13fc 0000 00068322       ]        move.b    #$0,$68322
$68342:[ 13fc 0008 00ff8800       ]        move.b    #$8,$ff8800
$6834A:[ 13fc 000f 00ff8802       ]        move.b    #$f,$ff8802
$68352:[ 13fc 0000 00ff8800       ]        move.b    #$0,$ff8800
$6835A:[ 13f9 00068323 00ff8802   ]        move.b    $68323,$ff8802
$68364:[ 5139 00068323            ]        subq.b    #$8,$68323
$6836A:[ 4cdf 0707                ]        movem.l   (a7)+,d0-d2/a0-a2
$6836E:[ 2f39 00068390            ]        move.l    $68390,-(a7)
$68374:[ 4e75                     ]        rts
$68376:[ 2f3c 00068324            ]        move.l    #$68324,-(a7)
$6837C:[ 3f3c 001c                ]        move.w    #$1c,-(a7)
$68380:[ 3f3c 0005                ]        move.w    #$5,-(a7)
$68384:[ 4e4d                     ]        trap      #$d
$68386:[ 508f                     ]        addq.l    #$8,a7
$68388:[ 23c0 00068390            ]        move.l    d0,$68390
$6838E:[ 4e75                     ]        rts       ;
$68390:[ 0000 0000                ]        ori.b     #$0,d0
$68394:[ 0000 0000                ]        ori.b     #$0,d0
$68398:[ 0000 0000                ]        ori.b     #$0,d0
```

(suite)

```
$68328:[ 5239 00068322          ]      addq.b    #$1,$68322
$6832E:[ 0c39 0008 00068322     ]      cmpi.b    #$8,$68322
$68336:[ 6702                   ]      beq.s     $6833a
$68338:[ 6030                   ]      bra.s     $6836a
$6833A:[ 13fc 0000 00068322     ]      move.b    #$0,$68322
$68342:[ 13fc 0008 00ff8800     ]      move.b    #$8,$ff8800
$6834A:[ 13fc 000f 00ff8802     ]      move.b    #$f,$ff8802
$68352:[ 13fc 0000 00ff8800     ]      move.b    #$0,$ff8800
$6835A:[ 13f9 00068323 00ff8802 ]      move.b    $68323,$ff8802
$68364:[ 5139 00068323          ]      subq.b    #$8,$68323
$6836A:[ 4cdf 0707              ]      movem.l   (a7)+,d0-d2/a0-a2
$6836E:[ 2f39 00068390          ]      move.l    $68390,-(a7)
$68374:[ 4e75                   ]      rts
$68376:[ 2f3c 00068324          ]      move.l    #$68324,-(a7)
$6837C:[ 3f3c 001c              ]      move.w    #$1c,-(a7)
$68380:[ 3f3c 0005              ]      move.w    #$5,-(a7)
$68384:[ 4e4d                   ]      trap      #$d
$68386:[ 508f                   ]      addq.l    #$8,a7
$68388:[ 23c0 00068390          ]      move.l    d0,$68390
$6838E:[ 4e75                   ]      rts
$68390:[ 0000 0000              ]      ori.b     #$0,d0
$68394:[ 0000 0000              ]      ori.b     #$0,d0
$68398:[ 0000 0000              ]      ori.b     #$0,d0
```

6/1.3.6

Arpégiateur sous timer B et VBL

Ce programme utilise deux interruptions en même temps. L'une utilise le timer B dans le mode event count et prend donc en compte l'interruption **HBL**, et l'autre utilise l'interruption **VBL** pour lui adjoindre un traitement supplémentaire, avant de rendre la main au processus normal de cette interruption. Voyons un peu plus avant le détail de ces interruptions : c'est grâce aux timers de l'Atari que nous allons pouvoir utiliser le retour de ligne horizontale ou **HBL** sans devoir inhiber les autres interruptions et risquer de planter le système ; en effet, le registre d'état est positionné par défaut sur la valeur trois ce qui ne permet pas dans un traitement normal d'utiliser le **HBL** dont le niveau de priorité est de deux ; par contre, sur les quatre timers de l'Atari, deux ont la possibilité de compter des signaux extérieurs. Le timer A est en rapport pour l'utilisateur avec Centronics-Busy et le timer B peut compter tous les **HBL** dans le mode qu'on appelle *event-count* ou comptage d'événements. Il suffit de lui donner une valeur à l'avance, ici 160 après l'avoir réglé sur le mode adéquat, et d'autoriser l'interruption par le timer B dans le MFP (niveau 8) ; l'ordinateur sautera à la routine d'interruption timerb dès que le nombre de **HBL** est égal à **nbrhbl**. Cela est effectué par la fonction **XBIOS(31)** ; cette réinitialisation du timer B est effectuée aussi à chaque passage dans l'interruption **VBL** pour réactualiser le timer B et sa fonction avec la nouvelle valeur de **nbrhbl** au départ. Cette valeur peut être incrémentée et décrémentée respectivement par les touches curseur haut et curseur bas. La sortie du programme se fait au moyen de la touche **Escape**. Pour donner un effet un peu rythmique à l'arpège, l'interruption **VBL** joue la note de fréquence **nbrhbl** au volume 15 et l'interruption **timerb** joue cette même hauteur régénérée en positionnant à zéro le volume sur le canal 1 immédiatement après. Si vous désirez avoir un effet d'arpèges descendant plus fluides, vous pouvez court-circuiter l'interruption timerb en ne validant que le rte de la dernière instruction ; vous aurez alors plus facilement accès à la note de départ de la série de notes, par l'intermédiaire des touches curseur haut et bas.

Dans l'exemple, c'est la même valeur qui fixe la hauteur jouée et la fréquence d'appel de **timerb**, mais vous avez tout le loisir d'introduire des variables différentes pour les hauteurs de notes jouées dans **timerb** et **vblir** ainsi que pour **nbrhbl**. Vous pourrez faire varier facilement ces paramètres en suivant le même modèle que le corps du programme qui lit les scan codes des touches du clavier : prenez par exemple les touches des curseurs gauche et droit, dont les scan codes à tester sont respectivement $4B et $4D.

```
            .DATA

vblvec      equ 28
compt:      .DC.l 0
nbrhbl:     .DC.w 160
compt1:     .DC.b 0
compt2:     .DC.b 0

            .EVEN

timerb:     move.b    #1,$ff8800
            move.b    nbrhbl,$ff8802
            move.b    #8,$ff8800
            move.b    #0,$ff8802
            rte

vblir:      movem.l   d0-d2/a0-a2,-(sp)    ;sauvegarde des
                                           ;registres
            move.l    #timerb,-(sp)
            move.w    nbrhbl,-(sp)         ; nbre de hbl
            move.w    #$8,-(sp)            ; mode event
            move.w    #1,-(sp)             ;timer B
            move.w    #31,-(sp)
            trap      #14
            adda.l    #12,sp

;on ne génère un son que tous les huit passages dans
;l'interruption VBL, mais c'est un autre paramètre
;intéressant sur lequel travailler pour réaliser des effets

            addq.b    #1,compt1
            cmpi.b    #8,compt1
            beq.s     suite
            bra.s     suite1
suite:      move.b    #0,compt1
            move.b    #8,$ff8800
            move.b    #15,$ff8802
            move.b    #1,$ff8800
            move.b    nbrhbl,$ff8802

;la valeur  125 qui décrémente compt2 est aussi un paramètre
;intéressant dans l'effet sonore obtenu puisqu'il représente
;des sauts itératifs de fréquence; pour agrémenter
```



(suite)

```
;l'affaire, vous pouvez si vous le désirez, essayer de faire
;varier cette valeur au moyen d'une lecture de touche
;clavier, générer des sauts aléatoires ou utiliser une
;fonction mathématique style suites itératives au moyen d'un
;petit sous-programme.

            move.w      #1,-(sp)
            move.w      #31,-(sp)
            trap        #14
            adda.l      #12,sp
            move.l      #vblir,-(sp)
            move.w      #vblvec,-(sp)
            move.w      #5,-(sp)
            trap        #13
            addq.l      #8,sp
            move.l      d0,oldvbl
            rts

            .BSS

oldvbl:     .DS.l 1
sassp:      .DS.l 1

            .TEXT

            bsr         init

;----------corps du programme-------

boucle0:    move.w      #1,-(sp)
            trap        #1
            addq.l      #2,sp
            swap.w      d0
            cmpi.w      #$48,d0
            bne         boucle1
            subq.b      #1,nbrhbl
boucle1:    cmpi.w      #$50,d0
            bne         boucle2
            addq.b      #1,nbrhbl
boucle2:    cmpi.w      #$1,d0
            bne         boucle0
```

(suite)

```
;----------désactive l'interruption timerb-------

          move.w    #8,-(sp)
          move.w    #26,-(sp)
          trap      #14
          addq.l    #4,sp

;---------------remet l'ancien vecteur en place------

          move.l    oldvbl,-(sp)
          move.w    #vblvec,-(sp)
          move.w    #5,-(sp)
          trap      #13
          addq.l    #8,sp

;-------rend la main au système---------

fin:      move.w    #0,-(sp)
          trap      #1
          .END
```



Le chargeur hexadécimal Basic du programme
Arpégiateur sous timer B et VBL

```
$682C6:[ 6100 0104                ]     bsr       $683cc
$682CA:[ 3f3c 0001                ]     move.w    #$1,-(a7)
$682CE:[ 4e41                     ]     trap      #$1
$682D0:[ 548f                     ]     addq.l    #$2,a7
$682D2:[ 4840                     ]     swap      d0
$682D4:[ 0c40 0048                ]     cmpi.w    #$48,d0
$682D8:[ 6600 0008                ]     bne       $682e2
$682DC:[ 5339 0006831e            ]     subq.b    #$1,$6831e
$682E2:[ 0c40 0050                ]     cmpi.w    #$50,d0
$682E6:[ 6600 0008                ]     bne       $682f0
$682EA:[ 5239 0006831e            ]     addq.b    #$1,$6831e
$682F0:[ 0c40 0001                ]     cmpi.w    #$1,d0
$682F4:[ 66d4                     ]     bne.s     $682ca
$682F6:[ 3f3c 0008                ]     move.w    #$8,-(a7)
$682FA:[ 3f3c 001a                ]     move.w    #$1a,-(a7)
$682FE:[ 4e4e                     ]     trap      #$e
$68300:[ 588f                     ]     addq.l    #$4,a7
$68302:[ 2f39 00068406            ]     move.l    $68406,-(a7)
$68308:[ 3f3c 001c                ]     move.w    #$1c,-(a7)
$6830C:[ 3f3c 0005                ]     move.w    #$5,-(a7)
$68310:[ 4e4d                     ]     trap      #$d
$68312:[ 508f                     ]     addq.l    #$8,a7
$68314:[ 3f3c 0000                ]     move.w    #$0,-(a7)
$68318:[ 4e41                     ]     trap      #$1
$6831A:[ 0000 0000                ]     ori.b     #$0,d0
$6831E:[ 00a0 000013fc            ]     ori.l     #$13fc,-(a0)
$68324:[ 0001 00ff                ]     ori.b     #$ff,d1
$68328:[ 8800                     ]     or.b      d0,d4
$6832A:[ 13f9 0006831e 00ff8802   ]     move.b    $6831e,$ff8802
$68334:[ 13fc 0008 00ff8800       ]     move.b    #$8,$ff8800
$6833C:[ 13fc 0000 00ff8802       ]     move.b    #$0,$ff8802
$68344:[ 4e73                     ]     rte
$68346:[ 48e7 e0e0                ]     movem.l   d0-d2/a0-a2,-(a7)
$6834A:[ 2f3c 00068322            ]     move.l    #$68322,-(a7)
$68350:[ 3f39 0006831e            ]     move.w    $6831e,-(a7)
$68356:[ 3f3c 0008                ]     move.w    #$8,-(a7)
$6835A:[ 3f3c 0001                ]     move.w    #$1,-(a7)
$6835E:[ 3f3c 001f                ]     move.w    #$1f,-(a7)
$68362:[ 4e4e                     ]     trap      #$e
$68364:[ dffc 0000000c            ]     adda.l    #$c,a7
$6836A:[ 5239 00068320            ]     addq.b    #$1,$68320
$68370:[ 0c39 0008 00068320       ]     cmpi.b    #$8,$68320
$68378:[ 6702                     ]     beq.s     $6837c
$6837A:[ 6044                     ]     bra.s     $683c0
$6837C:[ 13fc 0000 00068320       ]     move.b    #$0,$68320
$68384:[ 13fc 0008 00ff8800       ]     move.b    #$8,$ff8800
$6838C:[ 13fc 000f 00ff8802       ]     move.b    #$f,$ff8802
$68394:[ 13fc 0001 00ff8800       ]     move.b    #$1,$ff8800
$6839C:[ 13f9 0006831e 00ff8802   ]     move.b    $6831e,$ff8802
```

(suite)

```
$683A6:[ 0439 007d 00068321      ]       subi.b    #$7d,$68321
$683AE:[ 13fc 0000 00ff8800      ]       move.b    #$0,$ff8800
$683B6:[ 13f9 00068321 00ff8802  ]       move.b    $68321,$ff8802
$683C0:[ 4cdf 0707               ]       movem.l   (a7)+,d0-d2/a0-a2
$683C4:[ 2f39 00068406           ]       move.l    $68406,-(a7)
$683CA:[ 4e75                    ]       rts
$683CC:[ 2f3c 00068322           ]       move.l    #$68322,-(a7)
$683D2:[ 3f39 0006831e           ]       move.w    $6831e,-(a7)
$683D8:[ 3f3c 0008               ]       move.w    #$8,-(a7)
$683DC:[ 3f3c 0001               ]       move.w    #$1,-(a7)
$683E0:[ 3f3c 001f               ]       move.w    #$1f,-(a7)
$683E4:[ 4e4e                    ]       trap      #$e
$683E6:[ dffc 0000000c           ]       adda.l    #$c,a7
$683EC:[ 2f3c 00068346           ]       move.l    #$68346,-(a7)
$683F2:[ 3f3c 001c               ]       move.w    #$1c,-(a7)
$683F6:[ 3f3c 0005               ]       move.w    #$5,-(a7)
$683FA:[ 4e4d                    ]       trap      #$d
$683FC:[ 508f                    ]       addq.l    #$8,a7
$683FE:[ 23c0 00068406           ]       move.l    d0,$68406
```

6/1.3.7

Activation polyphonique du générateur sonore

Comme vous le savez certainement, le générateur sonore de l'Atari comporte trois voix activables simultanément. Dans les paragraphes qui suivent, nous allons voir comment, en GFA Basic et en assembleur.

Mais avant d'en arriver là, nous vous proposons un petit programme écrit en GFA Basic qui donne diverses informations concernant les notes des sept octaves connues du générateur sonore.

CODAGE DES NOTES DES SEPT OCTAVES DU GÉNÉRATEUR SONORE

Comment utiliser le programme

Lancez le programme. Les données des pages suivantes s'affichent sur l'écran.

Comme vous le voyez, les notes sont classées par octaves. Pour chaque note apparaissent :

— le nom de la note ;

— la fréquence correspondante ;

— la période de la note, en décimal ;

— le poids fort de la période de la note en hexadécimal ;

— le poids faible de la période de la note, en hexadécimal.

Par exemple, pour Fa# –2 :

```
Fa#     92.50    1351      5     47
```

Les trois dernières données sont particulièrement importantes, car elles permettront d'initialiser correctement les registres du générateur sonore en GFA et en assembleur.

****
* Octave -3 *
****

Note	Fréquence (Hz)	Période (Déc)	MSB	LSB
Do	32.70	3822	E	EE
Do#	34.65	3607	E	17
Ré	36.71	3405	D	4D
Ré#	38.89	3214	C	8E
Mi	41.20	3033	B	D9
Fa	43.65	2863	B	2F
Fa#	46.25	2702	A	8E
Sol	49.00	2551	9	F7
Sol#	51.91	2407	9	67
La	55.00	2272	8	E0
La#	58.27	2145	8	61
Si	61.74	2024	7	E8

****
* Octave -2 *
****

Note	Fréquence (Hz)	Période (Déc)	MSB	LSB
Do	65.41	1911	7	77
Do#	69.30	1803	7	8
Ré	73.42	1702	6	A6
Ré#	77.78	1607	6	47
Mi	82.41	1516	5	EC
Fa	87.31	1431	5	97
Fa#	92.50	1351	5	47
Sol	98.00	1275	4	FB
Sol#	103.83	1203	4	B3
La	110.00	1136	4	70
La#	116.54	1072	4	30
Si	123.47	1012	3	F4

****
* Octave -1 *
****

Note	Fréquence (Hz)	Période (Déc)	MSB	LSB
Do	130.81	955	3	BB
Do#	138.59	901	3	85
Ré	146.83	851	3	53
Ré#	155.56	803	3	23
Mi	164.81	758	2	F6
Fa	174.61	715	2	CB
Fa#	185.00	675	2	A3
Sol	196.00	637	2	7D
Sol#	207.65	601	2	57
La	220.00	568	2	38
La#	233.08	536	2	18
Si	246.94	506	1	FA

(suite)

****
* Octave 0 *
****

Note	Fréquence (Hz)	Période (Déc)	MSB	LSB
Do	261.63	477	1	DD
Do#	277.18	450	1	C2
Ré	293.66	425	1	A9
Ré#	311.13	401	1	91
Mi	329.63	379	1	7B
Fa	349.23	357	1	65
Fa#	369.99	337	1	51
Sol	392.00	318	1	3E
Sol#	415.30	300	1	2C
La	440.00	284	1	1C
La#	466.16	268	1	C
Si	493.88	253	0	FD

****
* Octave 1 *
****

Note	Fréquence (Hz)	Période (Déc)	MSB	LSB
Do	523.25	238	0	EE
Do#	554.37	225	0	E1
Ré	587.33	212	0	D4
Ré#	622.25	200	0	C8
Mi	659.26	189	0	BD
Fa	698.46	178	0	B2
Fa#	739.99	168	0	A8
Sol	783.99	159	0	9F
Sol#	830.61	150	0	96
La	880.00	142	0	8E
La#	932.33	134	0	86
Si	987.77	126	0	7E

****
* Octave 2 *
****

Note	Fréquence (Hz)	Période (Déc)	MSB	LSB
Do	1046.50	119	0	77
Do#	1108.73	112	0	70
Ré	1174.66	106	0	6A
Ré#	1244.51	100	0	64
Mi	1318.51	94	0	5E
Fa	1396.91	89	0	59
Fa#	1479.98	84	0	54
Sol	1567.98	79	0	4F
Sol#	1661.22	75	0	4B
La	1760.00	71	0	47
La#	1864.66	67	0	43
Si	1975.53	63	0	3F

(suite)

****
* Octave 3 *
****

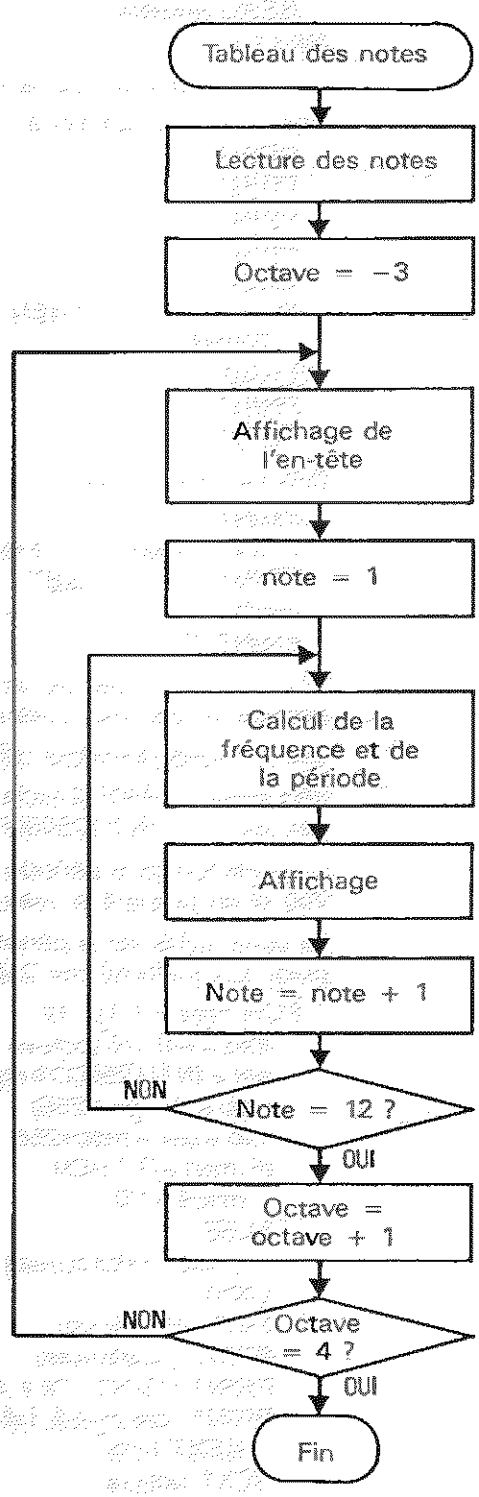
Note	Fréquence (Hz)	Période (Déc)	MSB	LSB
Do	2093.00	59	0	3B
Do#	2217.46	56	0	38
Ré	2349.32	53	0	35
Ré#	2489.02	50	0	32
Mi	2637.02	47	0	2F
Fa	2793.83	44	0	2C
Fa#	2959.96	42	0	2A
Sol	3135.96	39	0	27
Sol#	3322.44	37	0	25
La	3520.00	35	0	23
La#	3729.31	33	0	21
Si	3951.07	31	0	1F

****
* Octave 4 *
****

Note	Fréquence (Hz)	Période (Déc)	MSB	LSB
Do	4186.01	29	0	1D
Do#	4434.92	28	0	1C
Ré	4698.64	26	0	1A
Ré#	4978.03	25	0	19
Mi	5274.04	23	0	17
Fa	5587.65	22	0	16
Fa#	5919.91	21	0	15
Sol	6271.93	19	0	13
Sol#	6644.88	18	0	12
La	7040.00	17	0	11
La#	7458.62	16	0	10
Si	7902.13	15	0	F

Le programme en détail

La logique du programme apparaît dans l'ordinogramme suivant :



Après l'initialisation du tableau note$:

```
DIM note$(12)
FOR i=1 TO 12
  READ note$(i)
NEXT i
```

Le programme affiche le numéro de l'octave :

```
FOR octave= -3 TO 4
  PRINT
  PRINT
  PRINT
  PRINT ''                                    **************''
  PRINT ''                                    * Octave '';octave;
  IF octave >=0 THEN
    PRINT '' '';
  ENDIF
  PRINT ''*''
  PRINT ''                                    **************''
```

puis l'en-tête du tableau :

```
  PRINT
  PRINT ''Note      Fréquence (Hz)      Période (Déc)      MSB'';
  PRINT ''      LSB''
  PRINT ''---       ---------------     -------------      ---'';
  PRINT ''      ---''
```

Chacune des notes est ensuite affichée, suivie de sa fréquence et de sa période en décimal, puis en hexadécimal.

Les formules de calcul utilisées sont les suivantes :

fréquence = 440*(2^(octave+((note-10)/12)))
période = INT(125000/fréquence)

Le poids fort de la période est extrait en divisant la valeur de la période par 256 et en prenant la valeur entière du résultat.

Le poids faible de la période est égal à la valeur de la période diminué du poids fort multiplié par 256 :

```
  FOR note=1 TO 12
    freq=440*(2^(octave+((note-10)/12)))
    per=INT(125000/freq)
    msb=INT(per/256)
    lsb=per-msb*256
    IF msb=0 THEN
      msb$=''0''
    ELSE
      msb$=HEX$(msb)
    ENDIF
    lsb$=HEX$(lsb),
    PRINT note$(note),
    PRINT USING ''#####.##'',freq;
    PRINT ,per,msb$,lsb$
      NEXT note
    NEXT octave
```

Le listing du programme est le suivant :

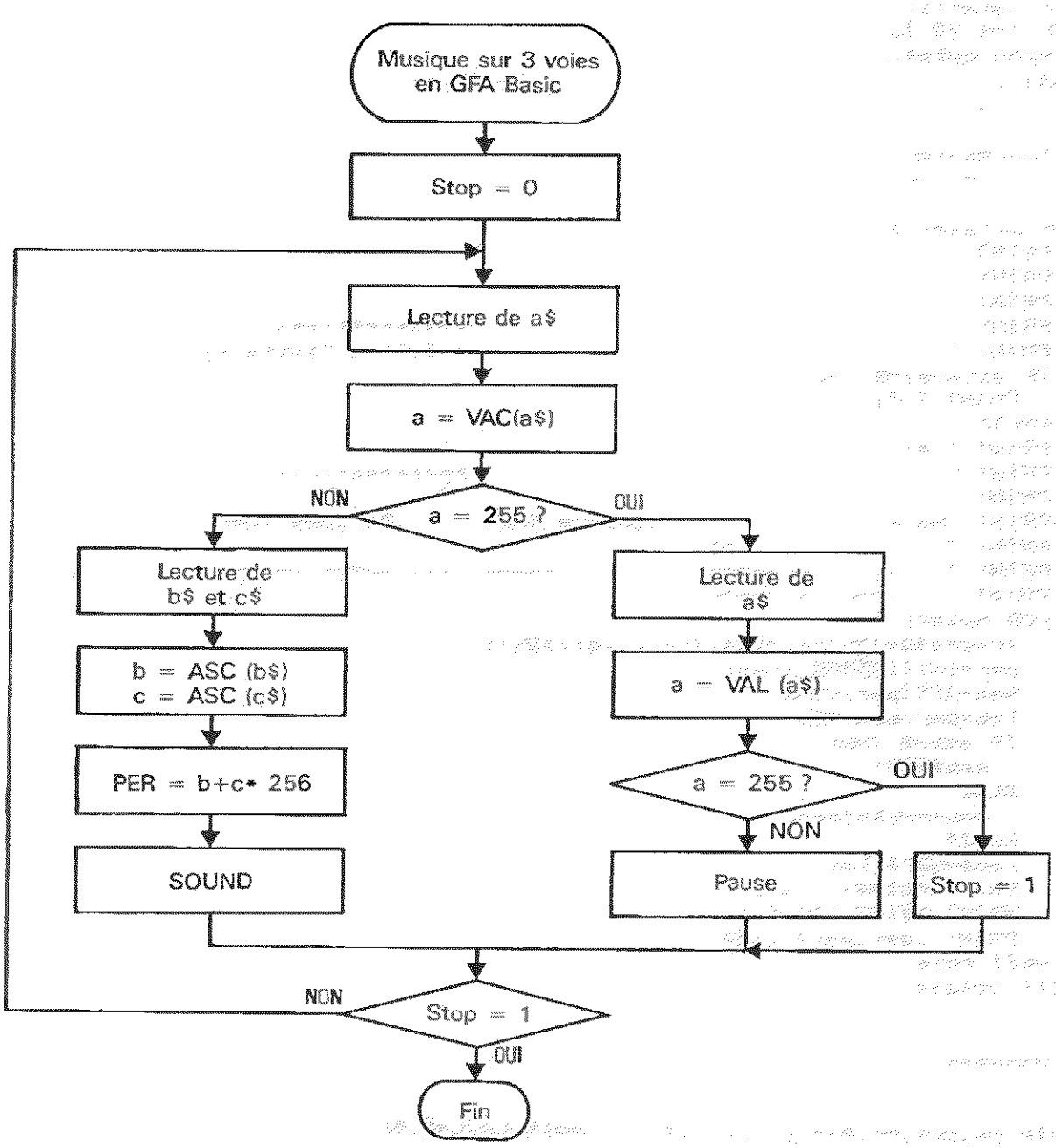
```
' ------------------------------------------------------
' Impression du tableau des fréquences et des notes
' ------------------------------------------------------
'
' ---------------
' Initialisation
' ---------------
'
DIM note$(12)
FOR i=1 TO 12
  READ note$(i)
NEXT i
'
' -----------
' Impression
' ---------
'
FOR octave=-3 TO 4
  PRINT
  PRINT
  PRINT
  PRINT "                                    *************"
  PRINT "                                    * Octave ";octave;
  IF octave>=0 THEN
    PRINT " ";
  ENDIF
  PRINT " *"
  PRINT "                                    *************"
  PRINT
  PRINT "Note              Fréquence (Hz)      Période (Déc)     MSB";
  PRINT "                LSB"
  PRINT "-------------------------------------------------------------";
  PRINT "-------------------"
  FOR note=1 TO 12
    freq=440*(2^(octave+((note-10)/12)))
    per=INT(125000/freq)
    msb=INT(per/256)
    lsb=per-msb*256
    IF msb=0 THEN
      msb$="0"
    ELSE
      msb$=HEX$(msb)
    ENDIF
    lsb$=HEX$(lsb)
    PRINT note$(note),
    PRINT USING "####.##",freq;
    PRINT ,per,msb$,lsb$
  NEXT note
NEXT octave
'
' --------
' Données
' --------
'
DATA Do,Do#,Ré,Ré#,Mi,Fa,Fa#,Sol,Sol#,La,La#,Si
```

ACTIVATION DE MUSIQUE SUR TROIS VOIX EN GFA BASIC

Les deux programmes proposés ici sont repris en assembleur. Vous pourrez ainsi effectuer toutes les comparaisons souhaitables dans le but d'apprendre le langage assembleur.

Premier exemple :

La logique du programme apparaît dans l'ordinogramme suivant :





Le programme se divise en deux parties. La première contient la logique d'activation du générateur sonore. La seconde contient le codage des diverses notes qui composent le morceau.

Comme le précise l'en-tête du programme, les données peuvent avoir trois significations différentes :

1) Si la première donnée est différente de $FF, elle spécifie un numéro de voix. Dans ce cas, les deux suivantes représentent (respectivement) le poids faible et le poids fort de la période de la note à jouer.

2) Si la première donnée est égale à $FF et la seconde différente de $FF, la seconde donnée contient une temporisation en 1/50 secondes.

3) Deux données consécutives égales à $FF matérialisent la fin du morceau.

Le programme débute par l'initialisation de la variable stop à zéro. Cette variable sera mise à 1 lorsque le code terminateur ($FF, $FF) sera rencontré dans les données.

```
stop=0 !Indicateur de fin de morceau
```

Les données sont ensuite lues à l'intérieur d'une boucle REPEAT UNTIL qui prend fin lorsque la variable stop vaut 1, c'est-à-dire lorsque le terminateur a été identifié.

```
REPEAT
...
UNTIL stop=1
```

La première donnée est lue à l'aide d'une instruction READ. Elle est stockée dans la variable a$:

```
READ a$
```

Si cette donnée est égale à 255 :

```
a=VAL(a$)
IF a=255 THEN !Temporisation ou fin de morceau
```

la donnée suivante est lue :

```
READ a$
```

Si elle est égale à 255, le morceau est terminé. La variable stop est initialisée à 1 :

```
a=VAL(a$)
IF a=255 THEN
  stop=1 !Fin du morceau
```

Dans le cas contraire, le programme marque une pause dont la durée est égale à la valeur de la seconde donnée multipliée par 8 :

```
ELSE
  WAVE 7,0,0,0,1 !Forme d'onde
  PAUSE a*8      !Temporisation
ENDIF
```

Dans le cas où la première donnée n'est pas égale à 255, il s'agit d'une note à jouer. Le poids faible et le poids fort de la période de la note sont lus à l'aide de deux instructions READ :

```
ELSE
  READ b$
  READ c$
```

Ces données sont converties en numérique pour calculer la valeur de la période :

```
b=VAL(b$)
c=VAL(c$)
per=b+c*256        !Période de la note jouer
```

La note ainsi décodée est jouée grâce à une instruction **sound** :

```
SOUND a,12,#per,0   !Activation de la note
```

Lorsque le code terminateur a été détecté, le générateur sonore est désactivé :

```
SOUND 1,0,0,0,0   !Arrêt du générateur sonore
```

La seconde partie du programme contient les notes codées comme il vient d'être dit :

```
DATA 1,$86,0,2,$a8,0,3,$c8,0,$ff,4,2,0,0,3,0,0,1,$c8,0,$ff,1
...
DATA 1,0,0,2,0,0,3,0,0,$ff,$ff
```

Le listing du programme est le suivant :

```
' ------------------------------------
' Activation de musique sur 3 Voies
' ------------------------------------
' Former Voie+Frequence
'        FF+Tempo(1 à 4)
'        FF+FF (Terminateur)
' ------------------------------------
'
stop=0 !Indicateur de fin de morceau
REPEAT
  READ a$
  a=VAL(a$)
  IF a=255 THEN !Temporisation ou fin de morceau
    READ a$
    a=VAL(a$)
    IF a=255 THEN
      stop=1 !Fin du morceau
    ELSE
      WAVE 7,0,0,0,1   !Forme d'onde
      PAUSE a*8        !Temporisation
    ENDIF
  ELSE
    READ b$
    READ c$
    b=VAL(b$)
    c=VAL(c$)
    per=b+c*256        !Période de la note à jouer
    SOUND a,12,#per,0 !Activation de la note
  ENDIF
UNTIL stop=1
SOUND 1,0,0,0,0 !Arret du générateur sonore
END
'
' ------------------
' Données musicales
' ------------------
'
DATA 1,$86,0,2,$a8,0,3,$c8,0,$ff,4,2,0,0,3,0,0,1,$c8,0,$ff,1
DATA 1,$a8,0,$ff,1,1,$86,0,$ff,1,1,$a8,0,$ff,1,1,$96,0,2,$c8,0
DATA 3,$bb,3,$ff,2,2,0,0,3,0,0,1,$ee,0,$ff,2,1,$c8,0,$ff,2,1,$a8,0,$ff,1
DATA 1,$96,0,$ff,1,1,$96,0,2,$b2,0,3,$c8,0,$ff,4,1,0,0,2,0,0,3,0,0
DATA $ff,1,1,$b2,0,$ff,1,1,$96,0,$ff,1,1,$b2,0,$ff,1,1,$e1,0,2,$c,1
DATA $ff,2,1,$51,1,2,0,0,$ff,2,1,$a8,0,2,$e1,0,3,$c,1,$ff,2,1,$51,1
DATA 2,0,0,3,0,0,$ff,2
DATA 1,$a8,0,2,$c8,0,3,$c,1,$ff,4,1,$a8,0,2,$c8,0,3,$23,3,$ff,2
DATA 1,$e1,0,2,0,0,3,0,0,$ff,1,1,$ee,0,$ff,1,1,$ee,0
DATA 2,$38,2,$ff,2,1,$dd,1,2,0,0,$ff,2,1,$a8,0,2,$ee,0,3,$1c,1
DATA $ff,2,1,$b2,0,2,0,0,3,0,0,$ff,2,1,$b2,0,2,$c,1,3,$18,2
DATA $ff,1,1,$e1,0,2,0,0,3,0,0,$ff,1,1,$c,1,$ff,1,1,$65,1,$ff,1
DATA 1,$b2,0,2,$ee,0,$ff,1,1,$c8,0,2,0,0,$ff,1,1,$ee,0,$ff,1
DATA 1,$1c,1,$ff,1,1,$b2,0,2,$c,1,$ff,2,1,$65,1,2,0,0,3,0,0,$ff,2
DATA 1,$e1,0,2,$c,1,$ff,2,1,$65,1,2,0,0,$ff,2
DATA 1,$86,0,2,$a8,0,3,$c8,0,$ff,4,2,0,0,3,0,0,1,$c8,0,$ff,1
DATA 1,$a8,0,$ff,1,1,$86,0,$ff,1,1,$a8,0,$ff,1,1,$96,0,2,$c8,0
```

(suite)

```
DATA 3,$bb,3,$ff,2,2,0,0,3,0,0,1,$ee,0,$ff,2,1,$c8,0,$ff,2,1,$a8,0,$ff,1
DATA 1,$96,0,$ff,1,1,$96,0,2,$b2,0,3,$c8,0,$ff,4,1,0,0,2,0,0,3,0,0
DATA $ff,1,1,$b2,0,$ff,1,1,$96,0,$ff,1,1,$b2,0,$ff,1,1,$e1,0,2,$c,1
DATA $ff,2,1,$51,1,2,0,0,$ff,2,1,$a8,0,2,$e1,0,3,$c,1,$ff,2,1,$51,1
DATA 2,0,0,3,0,0,$ff,2
DATA 1,$a8,0,2,$c8,0,3,$c,1,$ff,4,1,$a8,0,2,$c8,0,3,$23,3
DATA $ff,2,1,$c,1,2,0,0,3,0,0,$ff,2,1,$b2,0,2,$ee,0,3,$59,2
DATA $ff,3,1,$b2,0,2,$ee,0,3,0,0,$ff,2,1,$c,1,2,0,0,$ff,2
DATA 1,$b2,0,2,$c,1,3,$18,2,$ff,1,1,$e1,0,2,0,0,3,0,0,$ff,1
DATA 1,$c,1,$ff,1,1,$65,1,$ff,1,1,$3e,1,2,$23,3,$ff,1,1,$c8,0
DATA 2,0,0,$ff,1,1,$c,1,$ff,1,1,$3e,1,$ff,1,1,$e1,0,2,$c,1,3,$65,1
DATA $ff,3,1,0,0,2,0,0,3,0,0,$ff,2,1,$e1,0,$ff,1,1,$c8,0,$ff,1
DATA 1,$b2,0,2,$e1,0,3,$c,1,$ff,4,1,$86,0,2,$b2,0,3,$e1,0
DATA $ff,2,1,$96,0,2,$cb,2,3,0,0,$ff,1,1,$86,0,2,0,0,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,2,1,$cb,2,2,0,0,3,0,0,$ff,2
DATA 1,$e1,0,2,$c,1,$ff,2,1,$e1,0,2,$cb,2,$ff,1,1,$c8,0,2,0,0,$ff,1
DATA 1,$b2,0,2,$c8,0,3,$ee,0,$ff,2,1,$77,0,2,$cb,2,3,0,0,$ff,2
DATA 1,$77,0,2,$c8,0,3,$ee,0,$ff,2,1,$86,0,2,$cb,2,3,0,0,$ff,1
DATA 1,$77,0,2,0,0,$ff,1,1,$b2,0,2,$ee,0,3,$bb,3,$ff,2
DATA 1,$70,0,2,$cb,2,3,0,0,$ff,2,1,$c8,0,2,$ee,0,$ff,2,1,$cb,2,2,0,0
DATA $ff,2,1,$b2,0,2,$e1,0,3,$c,1,$ff,2,1,$86,0,2,$cb,2,3,0,0
DATA $ff,2,1,$b2,0,2,$e1,0,3,$c,1,$ff,2,1,$86,0,2,$b2,0,3,$ee,0
DATA $ff,2,1,$cb,2,2,0,0,3,0,0,$ff,2,1,$8e,0,2,$cb,2,3,0,0
DATA $ff,1,1,$86,0,2,0,0,$ff,1
DATA 1,$86,0,2,$b2,0,3,$c,1,$ff,1,1,$e1,0,2,0,0,3,0,0,$ff,1
DATA 1,$c,1,$ff,1,1,$65,1,$ff,1,1,$b2,0,2,$ee,0,3,$dd,1,$ff,1
DATA 1,$c8,0,2,0,0,3,0,0,$ff,1,1,$ee,0,$ff,1,1,$1c,1,$ff,1
DATA 1,$b2,0,2,$c,1,3,$c2,1,$ff,1,1,$e1,0,2,0,0,3,0,0,$ff,1
DATA 1,$c,1,$ff,1,1,$65,1,$ff,1,1,$b2,0,2,$ee,0,3,$bb,3,$ff,1
DATA 1,$c8,0,2,0,0,3,0,0,$ff,1,1,$ee,0,$ff,1,1,$c8,0,2,$1c,1
DATA $ff,1,1,$b2,0,2,$e1,0,3,$c,1,$ff,4,1,$86,0,2,$e1,0,3,$c,1
DATA $ff,2,1,$96,0,2,$65,1,3,0,0,$ff,1,1,$86,0,2,0,0,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,2,1,$65,1,2,0,0,3,0,0,$ff,2
DATA 1,$e1,0,2,$c,1,$ff,2,1,$e1,0,2,$65,1,$ff,1,1,$c8,0,2,0,0
DATA $ff,1,1,$b2,0,2,$c8,0,3,$ee,0,$ff,2,1,$77,0,2,$65,1
DATA 3,0,0,$ff,2,1,$77,0,2,$c8,0,3,$ee,0,$ff,2
DATA 1,$86,0,2,$65,1,3,0,0,$ff,1,1,$77,0,2,0,0,$ff,1
DATA 1,$77,0,2,$b2,0,3,$ee,0,$ff,2,1,$70,0,2,$65,1,3,0,0
DATA $ff,2,1,$c8,0,2,$ee,0,$ff,2,1,$cb,2,2,0,0,$ff,2
DATA 1,$b2,0,2,$e1,0,3,$c,1,$ff,2,1,$86,0,2,$cb,2,3,0,0
DATA $ff,2,1,$86,0,2,$e1,0,3,$c,1,$ff,2,1,$86,0,2,$cb,2
DATA 3,0,0,$ff,2,1,$86,0,2,$ee,0,3,$2c,1,$ff,2
DATA 1,$96,0,2,$cb,2,3,0,0,$ff,2,1,$96,0,2,$2c,1,$ff,2
DATA 1,$86,0,2,$cb,2,$ff,2,1,$86,0,2,$b2,0,3,$c,1,$ff,1
DATA 1,$e1,0,2,0,0,3,0,0,$ff,1,1,$c,1,$ff,1
DATA 1,$65,1,$ff,1,1,$b2,0,2,$ee,0,3,$bb,3,$ff,1
DATA 1,$c8,0,2,0,0,3,0,0,$ff,1,1,$ee,0,$ff,1,1,$1c,1,$ff,1
DATA 1,$b2,0,2,$e1,0,3,$c,1,$ff,2,1,$cb,2,2,0,0,3,0,0,$ff,2
DATA 1,$e1,0,2,$c,1,3,$18,2,$ff,2,1,$cb,2,2,0,0,3,0,0,$ff,2
DATA 1,$70,0,2,$86,0,3,$a8,0,$ff,2,1,$54,0,2,$c8,0,3,0,0
DATA $ff,2,1,$86,0,2,$a8,0,$ff,2,1,$77,0,2,$86,0,$ff,1
DATA 1,$86,0,2,0,0,$ff,1,1,$a8,0,2,$c8,0,3,$bb,3,$ff,2
DATA 1,$ee,0,2,0,0,3,0,0,$ff,2,1,$c8,0,$ff,2,1,$96,0,$ff,2
DATA 1,$96,0,2,$b2,0,$ff,2,1,$96,0,2,$e1,0,$ff,2,1,$59,0
```

(suite)

```
DATA 2,$b2,0,$ff,1,1,$4b,0,2,0,0,$ff,1,1,$70,0,2,$96,0,$ff,2
DATA 1,$70,0,2,$a8,0,3,$e1,0,$ff,2,1,$c,1,2,0,0,3,0,0,$ff,2
DATA 1,$e1,0,$ff,2,1,$a8,0,$ff,2
DATA 1,$a8,0,2,$c8,0,$ff,2,1,$64,0,2,$c,1,$ff,2,1,$64,0
DATA 2,$a8,0,3,$c8,0,$ff,2,1,$70,0,2,$c,1,3,0,0,$ff,1
DATA 1,$77,0,2,0,0,$ff,1,1,$77,0,2,$b2,0,3,$ee,0,$ff,2
DATA 1,$b2,0,2,$cb,2,3,0,0,$ff,2,1,$54,0,2,$a8,0,3,$ee,0,$ff,2
DATA 1,$59,0,2,$b2,0,3,0,0,$ff,2,1,$59,0,2,$b2,0,3,$c,1,$ff,1
DATA 1,$e1,0,2,0,0,3,0,0,$ff,1,1,$c,1,$ff,1,1,$65,1,$ff,1
DATA 1,$b2,0,2,$ee,0,$ff,1,1,$c8,0,2,0,0,$ff,1
DATA 1,$ee,0,$ff,1,1,$1c,1,$ff,1,1,$b2,0,2,$e1,0,$ff,2,1,$65,1,2,0,0
DATA $ff,2,1,$e1,0,$ff,2,1,$cb,2,$ff,2,1,$54,0,2,$86,0,3,$a8,0
DATA $ff,2,1,$86,0,2,$c8,0,3,0,0,$ff,2,1,$86,0,2,$a8,0,$ff,2,1,$77,0
DATA 2,$c8,0,$ff,1,1,$86,0,2,0,0,$ff,1,1,$96,0,2,$c8,0,3,$bb,3
DATA $ff,2,1,$ee,0,2,0,0,3,0,0,$ff,2,1,$96,0,2,$c8,0,$ff,2
DATA 1,$96,0,2,$ee,0,$ff,2,1,$59,0,2,$96,0,3,$b2,0,$ff,2
DATA 1,$4b,0,2,$e1,0,3,0,0,$ff,2,1,$70,0,2,$96,0,3,$b2,0,$ff,2
DATA 1,$70,0,2,$e1,0,3,0,0,$ff,2,1,$70,0,2,$a8,0,3,$e1,0,$ff,2
DATA 1,$c,1,2,0,0,3,0,0,$ff,2,1,$a8,0,2,$e1,0,$ff,2
DATA 1,$c,1,2,0,0,$ff,1,1,$70,0,$ff,1,1,$64,0,2,$a8,0,3,$c8,0
DATA $ff,2,1,$86,0,2,$c,1,3,0,0,$ff,2,1,$a8,0,2,$ee,0,$ff,2
DATA 1,$96,0,2,$c,1,$ff,2,1,$96,0,2,$b2,0,3,$ee,0,$ff,2
DATA 1,$2c,1,2,0,0,3,0,0,$ff,2,1,$b2,0,2,$ee,0,$ff,1
DATA 1,$86,0,2,0,0,$ff,1,1,$77,0,2,$2c,1,$ff,1,1,$86,0
DATA 2,0,0,$ff,1,1,$86,0,2,$b2,0,$ff,1
DATA 1,$e1,0,2,0,0,$ff,1,1,$86,0,2,$c,1,$ff,1
DATA 1,$65,1,2,0,0,$ff,1,1,$3e,1,2,$91,1,$ff,1
DATA 1,$c8,0,2,0,0,$ff,1,1,$c,1,$ff,1,1,$3e,1,$ff,1
DATA 1,$e1,0,2,$c,1,$ff,3,1,0,0 ,2,0,0
DATA $ff,2,1,0,0,$ff,1
DATA 1,0,0,2,0,0,3,0,0,$ff,3,1,0,0,2,0,0,3,0,0,$ff,3
DATA 1,0,0,2,0,0,3,0,0,$ff,$ff
```

Deuxième exemple :

Le second programme fonctionne comme le premier. Seules les données changent.

Le listing du programme est le suivant :

```
'
' ------------------------------------
' Activation de musique sur 3 Voies
' ------------------------------------
' Forme: Voie+Frequence
'        FF+Tempo(1 à 4)
'        FF+FF (Terminateur)
' ------------------------------------
'
stop=0 !Indicateur de fin de morceau
REPEAT
  READ a$
  a=VAL(a$)
  IF a=255 THEN !Temporisation ou fin de morceau
    READ a$
    a=VAL(a$)
    IF a=255 THEN
      stop=1 !Fin du morceau
    ELSE
      WAVE 7,0,0,0,1 !Forme d'onde
      PAUSE a*8      !Temporisation
    ENDIF
  ELSE
    READ b$
    READ c$
    b=VAL(b$)
    c=VAL(c$)
    per=b+c*256      !Période de la note à jouer
    SOUND a,12,#per,0 !Activation de la note
  ENDIF
UNTIL stop=1
SOUND 1,0,0,0,0 !Arret du générateur sonore
END
'
' ---------------------
' Données musicales
' ---------------------
'
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,0,0,2,0,0,3,$18,2,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,2,0,0,3,$18,2,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,0,0,2,0,0,3,$18,2,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,2,0,0,3,$18,2,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,2,0,0,3,$18,2,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,2,0,0,3,$18,2,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,2,0,0,3,$18,2,$ff,1
DATA 1,$77,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,2,0,0,3,$18,2,$ff,1
DATA 1,$77,0,2,$96,0,3,$65,1,$ff,1,1,$96,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$96,0,2,$ee,0,3,$65,1,$ff,1,1,$96,0,2,0,0,3,$cb,2,$ff,1
```

(suite)

```
DATA 1,$96,0,2,$ee,0,3,$65,1,$ff,1,1,$96,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$b2,0,2,$ee,0,3,$65,1,$ff,1,1,$b2,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$96,0,2,$ee,0,3,$65,1,$ff,1,1,$86,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$86,0,2,$ee,0,3,$65,1,$ff,1,1,$86,0,2,0,0,3,$cb,2,$ff,1
DATA 1,0,0,2,$96,0,3,$65,1,$ff,1,2,0,0,3,$cb,2,$ff,1
DATA 2,$96,0,3,$65,1,$ff,1,2,0,0,3,$cb,2,$ff,1
DATA 2,$c,1,3,$91,1,$ff,1,2,0,0,3,$23,3,$ff,1
DATA 2,$c,1,3,$91,1,$ff,1,2,0,0,3,$23,3,$ff,1
DATA 2,$c,1,3,$91,1,$ff,1,2,0,0,3,$23,3,$ff,1
DATA 1,$a8,0,2,$c,1,3,$91,1,$ff,1,1,$a8,0,2,0,0,3,$23,3,$ff,1
DATA 1,$a8,0,2,$c,1,3,$91,1,$ff,1,1,$a8,0,2,0,0,3,$23,3,$ff,1
DATA 1,$b2,0,2,$c,1,3,$91,1,$ff,1,1,$b2,0,2,0,0,3,$23,3,$ff,1
DATA 1,$b2,0,2,$c,1,3,$91,1,$ff,1,1,$b2,0,2,0,0,3,$23,3,$ff,1
DATA 1,$c8,0,2,$c,1,3,$91,1,$ff,1,1,$c8,0,2,0,0,3,$23,3,$ff,1
DATA 1,$b2,0,2,$c,1,3,$65,1,$ff,1,1,$b2,0,2,0,0,3,$91,1,$ff,1
DATA 1,$b2,0,2,$c,1,3,$65,1,$ff,1,1,$b2,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$b2,0,2,$c,1,3,$65,1,$ff,1,1,$b2,0,2,0,0,3,$91,1,$ff,1
DATA 1,$b2,0,2,$c,1,3,$65,1,$ff,1,1,$b2,0,2,0,0,3,$cb,2,$ff,1
DATA 1,0,0,2,$2c,1,3,$65,1,$ff,1,2,0,0,3,$cb,2,$ff,1
DATA 2,$2c,1,3,$65,1,$ff,1,2,0,0,3,$cb,2,$ff,1
DATA 2,$2c,1,3,$65,1,$ff,1,2,0,0,3,$cb,2,$ff,1
DATA 1,$b2,0,2,$2c,1,3,$65,1,$ff,1,1,$b2,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,2,0,0,3,$18,2,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,2,0,0,3,$18,2,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,2,0,0,3,$18,2,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,2,0,0,3,$18,2,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,2,0,0,3,$18,2,$ff,1
DATA 1,0,0,2,$e1,0,3,$c,1,$ff,1,2,0,0,3,$18,2,$ff,1
DATA 1,$77,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,2,0,0,3,$18,2,$ff,1
DATA 1,$77,0,2,$2c,1,3,$65,1,$ff,1,1,$77,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$77,0,2,$2c,1,3,$65,1,$ff,1,1,$77,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$77,0,2,$2c,1,3,$65,1,$ff,1,1,$96,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$b2,0,2,$2c,1,3,$65,1,$ff,1,1,$b2,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$96,0,2,$2c,1,3,$65,1,$ff,1,1,$96,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$86,0,2,$2c,1,3,$65,1,$ff,1,1,$86,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$86,0,2,$2c,1,3,$65,1,$ff,1,1,$86,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$86,0,2,$2c,1,3,$65,1,$ff,1,1,$86,0,2,0,0,3,$cb,2,$ff,1
DATA 1,0,0,2,$c,1,3,$91,1,$ff,1,2,$2c,1,3,$23,3,$ff,1
DATA 2,$c,1,3,$91,1,$ff,1,2,$2c,1,3,$23,3,$ff,1
DATA 2,$c,1,3,$91,1,$ff,1,2,$2c,1,3,$23,3,$ff,1
DATA 1,$a8,0,2,$51,1,3,$91,1,$ff,1,1,$a8,0,2,0,0,3,$23,3,$ff,1
DATA 1,$a8,0,2,$51,1,3,$91,1,$ff,1,1,$a8,0,2,0,0,3,$23,3,$ff,1
DATA 1,$a8,0,2,$51,1,3,$91,1,$ff,1,1,$a8,0,2,0,0,3,$23,3,$ff,1
DATA 1,$b2,0,2,$51,1,3,$91,1,$ff,1,1,$b2,0,2,0,0,3,$23,3,$ff,1
DATA 1,$c8,0,2,$51,1,3,$91,1,$ff,1,1,$b2,0,2,0,0,3,$23,3,$ff,1
DATA 1,$b2,0,2,$2c,1,3,$65,1,$ff,1,1,$b2,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$b2,0,2,$2c,1,3,$65,1,$ff,1,1,$b2,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$b2,0,2,$2c,1,3,$65,1,$ff,1,1,$b2,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$b2,0,2,$2c,1,3,$65,1,$ff,1,1,$b2,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$70,0,2,$2c,1,3,$65,1,$ff,1,1,$70,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$86,0,2,$2c,1,3,$65,1,$ff,1,1,$86,0,2,0,0,3,$cb,2,$ff,1
```

(suite)

```
DATA 1,$96,0,2,$2c,1,3,$65,1,$ff,1,1,$96,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$86,0,2,$2c,1,3,$65,1,$ff,1,1,$86,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,2,0,0,3,$18,2,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,2,0,0,3,$18,2,$ff,1
DATA 1,0,0,2,$e1,0,3,$c,1,$ff,1,2,0,0,3,$18,2,$ff,1
DATA 1,$b2,0,2,$e1,0,3,$c,1,$ff,1,1,$b2,0,2,0,0,3,$18,2,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,2,0,0,3,$18,2,$ff,1
DATA 1,$77,0,2,$e1,0,3,$c,1,$ff,1,1,$77,0,2,0,0,3,$18,2,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,2,0,0,3,$18,2,$ff,1
DATA 1,$77,0,2,$e1,0,3,$c,1,$ff,1,1,$77,0,2,0,0,3,$18,2,$ff,1
DATA 1,$77,0,2,$1c,1,3,$65,1,$ff,1,1,$77,0,2,0,0,3,$dd,1,$ff,1
DATA 1,0,0,2,$1c,1,3,$65,1,$ff,1,2,0,0,3,$dd,1,$ff,1
DATA 2,$1c,1,3,$65,1,$ff,1,2,0,0,3,$dd,1,$ff,1
DATA 1,$77,0,2,$1c,1,3,$65,1,$ff,1,1,$77,0,2,0,0,3,$dd,1,$ff,1
DATA 1,$70,0,2,$1c,1,3,$65,1,$ff,1,1,$70,0,2,0,0,3,$dd,1,$ff,1
DATA 1,$77,0,2,$1c,1,3,$65,1,$ff,1,1,$77,0,2,0,0,3,$dd,1,$ff,1
DATA 1,$70,0,2,$1c,1,3,$65,1,$ff,1,1,$70,0,2,0,0,3,$dd,1,$ff,1
DATA 1,$70,0,2,$e1,0,3,$c,1,$ff,1,1,$70,0,2,0,0,3,$18,2,$ff,1
DATA 1,$70,0,2,$e1,0,3,$c,1,$ff,1,1,$70,0,2,0,0,3,$18,2,$ff,1
DATA 1,0,0,2,$e1,0,3,$c,1,$ff,1,2,0,0,3,$18,2,$ff,1
DATA 2,$e1,0,3,$c,1,$ff,1,2,0,0,3,$18,2,$ff,1
DATA 1,$70,0,2,$e1,0,3,$c,1,$ff,1,1,$70,0,2,0,0,3,$18,2,$ff,1
DATA 1,$64,0,2,$e1,0,3,$c,1,$ff,1,1,$64,0,2,0,0,3,$18,2,$ff,1
DATA 1,$59,0,2,$e1,0,3,$c,1,$ff,1,1,$59,0,2,0,0,3,$18,2,$ff,1
DATA 1,$54,0,2,$e1,0,3,$c,1,$ff,1,1,$54,0,2,0,0,3,$18,2,$ff,1
DATA 1,$54,0,2,$51,1,3,$91,1,$ff,1,1,$54,0,2,0,0,3,$23,3,$ff,1
DATA 1,$54,0,2,$51,1,3,$91,1,$ff,1,1,$54,0,2,0,0,3,$23,3,$ff,1
DATA 1,$59,0,2,$51,1,3,$91,1,$ff,1,1,$59,0,2,0,0,3,$23,3,$ff,1
DATA 1,$64,0,2,$51,1,3,$91,1,$ff,1,1,$64,0,2,0,0,3,$23,3,$ff,1
DATA 1,$64,0,2,$1c,1,3,$65,1,$ff,1,1,$64,0,2,0,0,3,$91,1,$ff,1
DATA 1,$59,0,2,$1c,1,3,$65,1,$ff,1,1,$59,0,2,0,0,3,$91,1,$ff,1
DATA 1,$59,0,2,$1c,1,3,$65,1,$ff,1,1,$59,0,2,0,0,3,$91,1,$ff,1
DATA 1,$70,0,2,$1c,1,3,$65,1,$ff,1,1,$77,0,2,0,0,3,$91,1,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,2,0,0,3,$18,2,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,2,0,0,3,$18,2,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,2,0,0,3,$18,2,$ff,1
DATA 1,0,0,2,$e1,0,3,$c,1,$ff,1,2,0,0,3,$18,2,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,2,0,0,3,$18,2,$ff,1
DATA 1,$77,0,2,$e1,0,3,$c,1,$ff,1,1,$77,0,2,0,0,3,$18,2,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,2,0,0,3,$18,2,$ff,1
DATA 1,$77,0,2,$e1,0,3,$c,1,$ff,1,1,$77,0,2,0,0,3,$18,2,$ff,1
DATA 1,$77,0,2,$1c,1,3,$65,1,$ff,1,1,$77,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$77,0,2,$1c,1,3,$65,1,$ff,1,1,$77,0,2,0,0,3,$dd,1,$ff,1
DATA 1,$77,0,2,$1c,1,3,$65,1,$ff,1,1,0,0,2,0,0,3,$cb,2,$ff,1
DATA 2,$1c,1,3,$65,1,$ff,1,2,0,0,3,$dd,1,$ff,1
DATA 1,$77,0,2,$1c,1,3,$65,1,$ff,1,1,$77,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$70,0,2,$1c,1,3,$65,1,$ff,1,1,$70,0,2,0,0,3,$dd,1,$ff,1
DATA 1,$64,0,2,$1c,1,3,$65,1,$ff,1,1,$70,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$64,0,2,$1c,1,3,$65,1,$ff,1,1,$70,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$70,0,2,$e1,0,3,$c,1,$ff,1,1,$70,0,2,0,0,3,$18,2,$ff,1
DATA 1,$70,0,2,$e1,0,3,$c,1,$ff,1,1,$70,0,2,0,0,3,$18,2,$ff,1
```



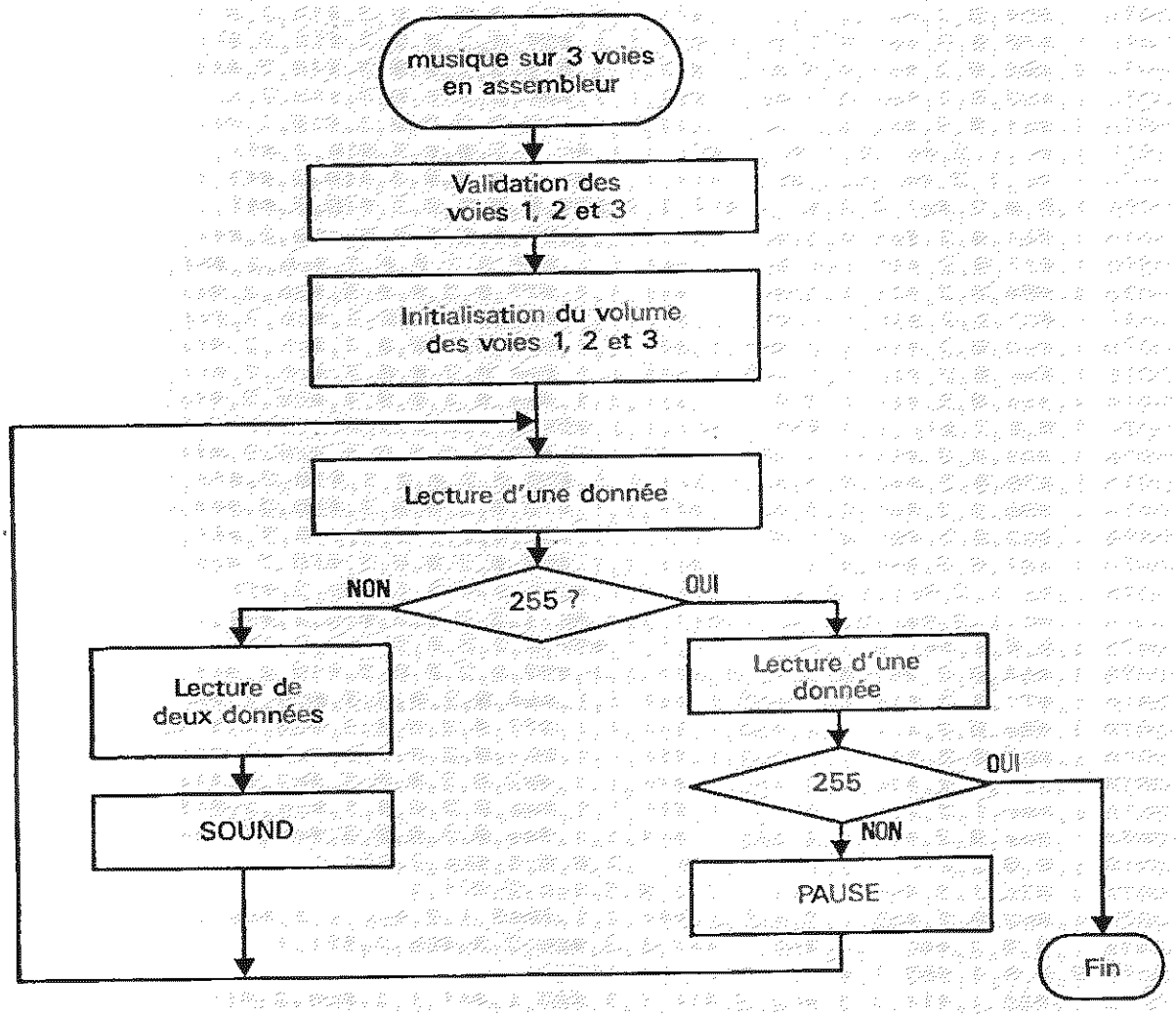
(suite)

```
DATA 1,$70,0,2,$e1,0,3,$c,1,$ff,1,1,0,0,2,0,0,3,$18,2,$ff,1
DATA 2,$e1,0,3,$c,1,$ff,1,2,0,0,3,$18,2,$ff,1
DATA 1,$70,0,2,$e1,0,3,$c,1,$ff,1,1,$70,0,2,0,0,3,$18,2,$ff,1
DATA 1,$64,0,2,$e1,0,3,$c,1,$ff,1,1,$64,0,2,0,0,3,$18,2,$ff,1
DATA 1,$59,0,2,$e1,0,3,$c,1,$ff,1,1,$59,0,2,0,0,3,$18,2,$ff,1
DATA 1,$54,0,2,$e1,0,3,$c,1,$ff,1,1,$54,0,2,0,0,3,$18,2,$ff,1
DATA 1,$70,0,2,$e1,0,3,$c,1,$ff,1,1,$59,0,2,0,0,3,$18,2,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$70,0,2,0,0,3,$18,2,$ff,1
DATA 1,$b2,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,2,0,0,3,$18,2,$ff,1
DATA 1,$e1,0,2,$e1,0,3,$c,1,$ff,1,1,$b2,0,2,0,0,3,$18,2,$ff,1
DATA 1,$c,1,2,$e1,0,3,$c,1,$ff,1,1,$c,1,2,0,0,3,$18,2,$ff,1
DATA 1,$c,1,2,$e1,0,3,$c,1,$ff,1,1,$c,1,2,0,0,3,$18,2,$ff,1
DATA 1,0,0,2,$e1,0,3,$c,1,$ff,1,1,$59,0,2,0,0,3,$18,2,$ff,1
DATA 1,$64,0,2,$e1,0,3,$c,1,$ff,1,1,$59,0,2,0,0,3,$18,2,$ff,1
DATA 1,$77,0,2,$1c,1,3,$65,1,$ff,1,1,$64,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$8e,0,2,$1c,1,3,$65,1,$ff,1,1,$77,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$b2,0,2,$1c,1,3,$65,1,$ff,1,1,$8e,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$c8,0,2,$1c,1,3,$65,1,$ff,1,1,$b2,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$ee,0,2,$1c,1,3,$65,1,$ff,1,1,$ee,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$ee,0,2,$1c,1,3,$65,1,$ff,1,1,$ee,0,2,0,0,3,$cb,2,$ff,1
DATA 1,0,0,2,$1c,1,3,$65,1,$ff,1,1,$59,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$64,0,2,$1c,1,3,$65,1,$ff,1,1,$59,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$70,0,2,$e1,0,3,$c,1,$ff,1,1,$59,0,2,0,0,3,$18,2,$ff,1
DATA 1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$70,0,2,0,0,3,$18,2,$ff,1
DATA 1,$b2,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,2,0,0,3,$18,2,$ff,1
DATA 1,$e1,0,2,$e1,0,3,$c,1,$ff,1,1,$b2,0,2,0,0,3,$18,2,$ff,1
DATA 1,$c,1,2,$e1,0,3,$c,1,$ff,1,1,$c,1,2,0,0,3,$18,2,$ff,1
DATA 1,$c,1,2,$e1,0,3,$c,1,$ff,1,1,$c,1,2,0,0,3,$18,2,$ff,1
DATA 1,0,0,2,$e1,0,3,$c,1,$ff,1,1,$59,0,2,0,0,3,$18,2,$ff,1
DATA 1,$64,0,2,$e1,0,3,$c,1,$ff,1,1,$59,0,2,0,0,3,$18,2,$ff,1
DATA 1,$77,0,2,$1c,1,3,$65,1,$ff,1,1,$64,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$8e,0,2,$1c,1,3,$65,1,$ff,1,1,$77,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$b2,0,2,$1c,1,3,$65,1,$ff,1,1,$8e,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$c8,0,2,$1c,1,3,$65,1,$ff,1,1,$b2,0,2,0,0,3,$b2,0,$ff,1
DATA 1,$ee,1,2,$1c,1,3,$65,1,$ff,1,1,$ee,0,2,0,0,3,$cb,2,$ff,1
DATA 1,$ee,0,2,$1c,1,3,$65,1,$ff,1,1,$ee,0,2,0,0,3,$cb,2,$ff,1
DATA 1,0,0,2,$1c,1,3,$65,1,$ff,1,2,0,0,3,$cb,2,$ff,1
DATA 2,$1c,1,3,$65,1,$ff,1,2,0,0,3,$cb,2,$ff,1
DATA 1,$b2,0,2,$65,1,3,$65,1,$ff,1,1,$b65,1,2,$cb,2,3,$cb,2,$ff,1
DATA 1,0,0,2,$65,1,3,$65,1,$ff,1,2,$cb,2,3,$cb,2,$ff,1
DATA 2,0,0,3,$65,1,$ff,1,3,$cb,2,$ff,1
DATA 3,$65,1,$ff,1,3,$cb,2,$ff,1,3,$65,1,$ff,1,3,$cb,2,$ff,1
DATA 1,0,0,2,0,0,3,0,0,$ff,$ff
```

ACTIVATION DE MUSIQUE SUR TROIS VOIX EN ASSEMBLEUR

Premier exemple :

La logique du programme apparaît dans l'ordinogramme suivant :





Comme pour le programme écrit en GFA Basic, le programme assembleur se divise en deux parties. La première contient la logique d'activation du générateur sonore. La seconde contient le codage des diverses notes qui composent le morceau.

L'étiquette de début de programme est muse :

```
muse:
```

Dans un premier temps, les voix 1, 2 et 3 sont validées. Pour ce faire, la fonction GIACCESS #28 du XBIOS est activée. Rappelons le format d'appel de cette fonction :

• Accès en écriture :

```
move.w    #88,-(sp)          ;Accès en écriture du registre 8
move.w    #5,-(sp)           ;Valeur 5
move.w    #28,-(sp)          ;Fonction #28
trap      #14                ;Accès au XBIOS
addq.w    #6,sp              ;Rétablissement de la pile
```

• Accès en lecture :

```
move.w    #5,-(sp)           ;Accès en écriture du registre 8
clr.w     -(sp)              ;Valeur quelconque
move.w    #28,-(sp)          ;Fonction #28
trap      #14                ;Accès au XBIOS
addq.w    #6,sp              ;Rétablissement de la pile
```

Reportez-vous à la Partie 3, Chapitre 3, pages 19 et 20 pour avoir plus de détails sur la fonction GIACCESS.

L'accès au registre 7 en écriture se fait donc comme suit :

```
move  #$80+7,-(sp)
move  #$f8,-(sp) ;Voix 1, 2 et 3 Validees
move  #28,  -(sp)
trap  #14
addq.l  #6,sp
```

Les trois accès suivants à la fonction GIACCESS permettent de définir le volume de chaque voix à la valeur 12 :

```
move  #$80+8,-(sp)
move  #12,-(sp) ;Volume Voix 1
move  #28,-(sp)
trap  #14
addq.l  #6,-(sp)
move  #$80+9,-(sp)
move  #12,-(sp) ;Volume Voix 2
move  #28,-(sp)
trap  #14
addq.l  #6,sp
move  #$80+10,-(sp)
move  #12,-(sp) ;Volume Voix 3
move  #28,-(sp)
trap  #14
addq.l  #6,sp
```

L'adresse de la première donnée musicale est ensuite placée dans le registre a5 :

```
move.l #data,a5 ;donnees musicales
```

La boucle de lecture principale a pour nom muse2 :

```
muse2:
```

Dans le cas où la donnée lue est égale à 255, le programme se débranche à l'étiquette traispe :

```
move.b (a5)+,d0 ;dans D0
cmp.b #$ff,d0 ;Code Special ?
beq traispe ;Oui
```

La prochaine donnée est lue. Si elle est égale à 255, le programme se débranche à l'étiquette finmuse qui redonne le contrôle au système :

```
Traispe: ;FF detecte
move.b (a5)+,d0 ;Code suivant
cmp.b #$ff,d0
beq finmuse

;-------------------------------------------------------------------
;Fin du morceau
;-------------------------------------------------------------------
finmuse;
rts
```

Dans le cas où la seconde donnée lue n'est pas égale à 255, elle correspond à une temporisation. En fonction de sa valeur, le programme se débranche à l'étiquette pause1, pause2, pause3 ou pause4 :

```
cmp.b #1,d0
beq pause1
cmp.b #2,d0
beq pause2
cmp.b #3,d0
beq pause3
cmp.b #4,d0
beq pause4
```

Les programmes situés aux étiquettes pauseX initialisent le registre d6 à différentes valeurs. pause1 stocke la plus petite valeur dans d6 et pause4 la plus grande. Quelle que soit l'étiquette accédée (pause1 à pause4), le programme se débranche à l'étiquette preloop après l'initialisation du registre d6 :

```
pause1: move.l #$ffff,d6 ;Pause la + courte
jmp preloop
pause2: move.l #$1ffff,d6
jmp preloop
pause3: move.l #$2ffff,d6
jmp preloop
pause4: move.l #$3ffff,d6 ;Pause la + longue
```



Cette étiquette pointe sur un programme de temporisation qui :

— initialise le registre **d7** à 0,

— incrémente le registre **d7**,

— compare sa valeur à celle de d6. Si d7< >d6, le registre **d7** est à nouveau incrémenté. Ainsi de suite... Si d7=d6, le programme se débranche en **muse2**. La temporisation est terminée et la prochaine note peut être lue :

```
preloop: move.l #$0,d7
loop: addq.l #1,d7
cmp.l d6,d7
bne loop
jmp muse2
```

Dans le cas où la donnée lue n'est pas égale à 255, les poids faible et fort sont lus et stockés dans les registres du générateur sonore :

```
sub.b #1,d0
add.b d0,d0
add.b #$80,d0
move.b d0,d1
move d0,-(sp)
move.b (a5)+,d0 ;Note LSB
move.w d0,-(sp)
move.w #28,(sp)
trap #14
addq.l #6,sp
move.b d1,d0
add.w #1,d0
move.w d0,-(sp)
move.b (a5)+,d0 ;Note MSB
move.w d0,-(sp)
move.w #28,-(sp)
trap #14
addq.l #6,sp
```

Le programme se débranche alors en muse2 pour lire la prochaine note :

```
jmp muse2
```

Les données musicales sont placées à partir de l'étiquette **data**. Elles sont de même nature que celles du programme GFA Basic correspondant :

```
data:
dc.b 1,$86,0,2,$a8,0,3,$c8,0,$ff,4,2,0,0,3,0,0,1,$c8,0,$ff,1
...
dc.b 1,0,0,2,0,0,3,0,0,$ff,$ff
```

Le listing du programme est le suivant :

```
                        ; Activation de musique sur 3 Voies
                        ;-------------------------------------------------------
                        ;Forme: Voie+Frequence OU FF+Tempo(1 a 4)
                        ;       OU FF+FF (Terminateur)
                        ;-------------------------------------------------------
                        muse1:
000000 3F3C0087                 move    #$80+7,-(sp)
000004 3F3C00F8                 move    #$f8,-(sp)  ;Voies 1, 2 et 3 Validees
000008 3F3C001C                 move    #28,-(sp)
00000C 4E4E                     trap    #14
00000E 5C8F                     addq.l  #6,sp
000010 3F3C0088                 move    #$80+8,-(sp)
000014 3F3C000C                 move    #12,-(sp)   ;Volume Voie 1
000018 3F3C001C                 move    #28,-(sp)
00001C 4E4E                     trap    #14
00001E 5CA7                     addq.l  #6,-(sp)
000020 3F3C0089                 move    #$80+9,-(sp)
000024 3F3C000C                 move    #12,-(sp)   ;Volume Voie 2
000028 3F3C001C                 move    #28,-(sp)
00002C 4E4E                     trap    #14
00002E 5C8F                     addq.l  #6,sp
000030 3F3C008A                 move    #$80+10,-(sp)
000034 3F3C000C                 move    #12,-(sp)   ;Volume Voie 3
000038 3F3C001C                 move    #28,-(sp)
00003C 4E4E                     trap    #14
00003E 5C8F                     addq.l  #6,sp
000040 2A7C000000EE             move.l  %data,a5    ;donnees musicales
                        muse2:
000046 101D                     move.b  (a5)+,d0    ;dans D0
000048 0C0000FF                 cmp.b   #$ff,d0     ;Code Special ?
00004C 67000036                 beq     traispe     ;Oui
000050 04000001                 sub.b   #1,d0
000054 D000                     add.b   d0,d0
000056 06000080                 add.b   #$80,d0
00005A 1200                     move.b  d0,d1
00005C 3F00                     move    d0,-(sp)
00005E 101D                     move.b  (a5)+,d0    ;Note LSB
000060 3F00                     move.w  d0,-(sp)
000062 3F3C001C                 move.w  #28,-(sp)
000066 4E4E                     trap    #14
000068 5C8F                     addq.l  #6,sp
00006A 1001                     move.b  d1,d0
00006C 06400001                 add.w   #1,d0
000070 3F00                     move.w  d0,-(sp)
000072 101D                     move.b  (a5)+,d0    ;Note MSB
000074 3F00                     move.w  d0,-(sp)
000076 3F3C001C                 move.w  #28,-(sp)
00007A 4E4E                     trap    #14
00007C 5C8F                     addq.l  #6,sp
00007E 4EF900000046             jmp     muse2
                        Traispe: ;FF    detecte
000084 101D                     move.b  (a5)+,d0    ;Code suivant
000086 0C0000FF                 cmp.b   #$ff,d0
00008A 67000060                 beq     finmuse
                        ;-------------------------------------------------------
                        ;Tempo entre deux notes
                        ;-------------------------------------------------------
00008E 0C000001                 cmp.b   #1,d0
000092 6700001A                 beq     pause1
000096 0C000002                 cmp.b   #2,d0
```



(suite)

```
 61 0C3B9A 6700001E              beq     pause2
 62 00009E 0C000003              cmp.b   #3,d0
 63 0000A2 67000022              beq     pause3
 64 0000A6 0C000004              cmp.b   #4,d0
 65 0000AA 67000026              beq     pause4
 66 0000AE 2C3C0000FFFF pause1:  move.l  #$ffff,d6   ;Pause la + courte
 67 0000B4 4EF900000008          jmp     preloop
 68 0000BA 2C3C0001FFFF pause2:  move.l  #$1ffff,d6
 69 0000C0 4EF900000008          jmp     preloop
 70 0000C6 2C3C0002FFFF pause3:  move.l  #$2ffff,d6
 71 0000CC 4EF900000008          jmp     preloop
 72 0000D2 2C3C0003FFFF pause4:  move.l  #$3ffff,d6  ;Pause la + longue
 73 0000D8 2E3C00000000 preloop: move.l  #$0,d7
 74 0000DE 5287         loop:    addq.l  #1,d7
 75 0000E0 BE86                  cmp.l   d6,d7
 76 0000E2 6600FFFA              bne     loop
 77 0000E6 4EF900000046          jmp     fin3e2
 78                      ;-------------------------------------------------
 79                      ;Fin du morceau
 80                      ;-------------------------------------------------
 81                      finmuse:
 82 0000EC 4E75                  rts
 83 0000EE 01B60002A800 data:    dc.b    1,$86,0,2,$a8,0,3,$c8,0,$ff,4,2,0,0,3
 84 000104 01A800FF0101          dc.b    1,$a8,0,$ff,1,1,$86,0,$ff,1,1,$a8,0,$
 85 000119 03BB03FF0202          dc.b    3,$bb,3,$ff,2,2,0,0,3,0,0,1,$ee,0,$ff
 86 000133 019600FF0101          dc.b    1,$96,0,$ff,1,1,$96,0,2,$b2,0,3,$c8,0
 87 00014C FF0101B200FF          dc.b    $ff,1,1,$b2,0,$ff,1,1,$96,0,$ff,1,1,$
 88 000163 FF0201510102          dc.b    $ff,2,1,$51,1,2,0,0,$ff,2,1,$a8,0,2,$
 89 00017B 02000003000           dc.b    2,0,0,3,0,0,$ff,2
 90 000183 01A80002C800          dc.b    1,$a8,0,2,$c8,0,3,$c,1,$ff,4,1,$a8,0,
 91 000199 01E100020000          dc.b    1,$e1,0,2,0,0,3,0,0,$ff,1,1,$ee,0,$ff
 92 0001AC 0238 02FF0201         dc.b    2,$38,2,$ff,2,1,$dd,1,2,0,0,$ff,2,1,$
 93 0001C2 FF0201B20002          dc.b    $ff,2,1,$b2,0,2,0,0,3,0,0,$ff,2,1,$b2
 94 0001D8 FF0101E10002          dc.b    $ff,1,1,$e1,0,2,0,0,3,0,0,$ff,1,1,$c,
 95 0001EF 01B20002EE00          dc.b    1,$b2,0,2,$ee,0,$ff,1,1,$c8,0,2,0,0,$
 96 000204 011C01FF0101          dc.b    1,$1c,1,$ff,1,1,$b2,0,2,$c,1,$ff,2,1,
 97 00021C 01E100C20C01          dc.b    1,$e1,0,2,$c,1,$ff,2,1,$65,1,2,0,0,$f
 98 000232C 0186 0002A800        dc.b    1,$86,0,2,$a8,0,3,$c8,0,$ff,4,2,0,0,3
 99 000242 01A800FF0101          dc.b    1,$a8,0,$ff,1,1,$86,0,$ff,1,1,$a8,0,$
100 000257 03BB03FF0202          dc.b    3,$bb,3,$ff,2,2,0,0,3,0,0,1,$ee,0,$ff
101 000271 019600FF0101          dc.b    1,$96,0,$ff,1,1,$96,0,2,$b2,0,3,$c8,0
102 00028A FF0101B200FF          dc.b    $ff,1,1,$b2,0,$ff,1,1,$96,0,$ff,1,1,$
103 0002A1 FF0201510102          dc.b    $ff,2,1,$51,1,2,0,0,$ff,2,1,$a8,0,2,$
104 0002B9 02000003000           dc.b    2,0,0,3,0,0,$ff,2
105 0002C1 01A80002C800          dc.b    1,$a8,0,2,$c8,0,3,$c,1,$ff,4,1,$a8,0,
106 0002D5 FF0201 0C0102         dc.b    $ff,2,1,$c,1,2,0,0,3,0,0,$ff,2,1,$b2,
107 0002EB FF0301B20002          dc.b    $ff,3,1,$b2,0,2,$ee,0,3,0,0,$ff,2,1,$
108 000300 01B200020C01          dc.b    1,$b2,0,2,$c,1,3,$1B,2,$ff,1,1,$e1,0,
109 000316 010C01FF0101          dc.b    1,$c,1,$ff,1,1,$65,1,$ff,1,1,$3e,1,2,
110 00032B 020000FF0101          dc.b    2,0,0,$ff,1,1,$c,1,$ff,1,1,$3e,1,$ff,
111 000343 FF0301000002          dc.b    $ff,3,1,0,0,2,0,0,3,0,0,$ff,2,1,$e1,0
112 00035A 01B20002E100          dc.b    1,$b2,0,2,$e1,0,3,$c,1,$ff,4,1,$86,0,
113 00036E FF0201960002          dc.b    $ff,2,1,$96,0,2,$cb,2,3,0,0,$ff,1,1,$
114 000383 01860002E100          dc.b    1,$86,0,2,$e1,0,3,$c,1,$ff,2,1,$cb,2,
115 000399 01E100020C01          dc.b    1,$e1,0,2,$c,1,$ff,2,1,$e1,0,2,$cb,2,
116 0003B1 01B200 22C800         dc.b    1,$b2,0,2,$c8,0,3,$ee,0,$ff,2,1,$77,0
117 0003C7 01770002C800          dc.b    1,$77,0,2,$c8,0,3,$ee,0,$ff,2,1,$86,0
118 0003DD 01770002 0000         dc.b    1,$77,0,2,0,0,$ff,1,1,$b2,0,2,$ee,0,3
119 0003F0 01700002CB02          dc.b    1,$70,0,2,$cb,2,3,0,0,$ff,2,1,$c8,0,2
120 000409 FF0201B20002          dc.b    $ff,2,1,$b2,0,2,$e1,0,3,$c,1,$ff,2,1,
```

(suite)

```
121 00041F FF0201B20C32          dc.b    $ff,2,1,$b2,0,2,$e1,0,3,$c,1,$ff,2,1,
122 000435 FF0201CB0202          dc.b    $ff,2,1,$cb,2,2,0,0,3,0,0,$ff,2,1,$8e
123 00044B FF0101860002          dc.b    $ff,1,1,$86,0,2,0,0,$ff,1
124 000455 01860002B200          dc.b    1,$86,0,2,$b2,0,3,$c,1,$ff,1,1,$e1,0,
125 00046B 010C01FF0101          dc.b    1,$c,1,$ff,1,1,$65,1,$ff,1,1,$b2,0,2,
126 000480 01C800020000          dc.b    1,$c8,0,2,0,0,3,0,0,$ff,1,1,$ee,0,$ff
127 000495 01B200020C01          dc.b    1,$b2,0,2,$c,1,3,$c2,1,$ff,1,1,$e1,0,
128 0004AB 010C01FF0101          dc.b    1,$c,1,$ff,1,1,$65,1,$ff,1,1,$b2,0,2,
129 0004C0 01C800020000          dc.b    1,$c8,0,2,0,0,3,0,0,$ff,1,1,$ee,0,$ff
130 0004D6 FF0101B20002          dc.b    $ff,1,1,$b2,0,2,$e1,0,3,$c,1,$ff,4,1,
131 0004EC FF0201960002          dc.b    $ff,2,1,$96,0,2,$65,1,3,0,0,$ff,1,1,$
132 000501 01860002E100          dc.b    1,$86,0,2,$e1,0,3,$c,1,$ff,2,1,$65,1,
133 000517 01E100020C01          dc.b    1,$e1,0,2,$c,1,$ff,2,1,$e1,0,2,$65,1,
134 00052D FF0101B20002          dc.b    $ff,1,1,$b2,0,2,$c8,0,3,$ee,0,$ff,2,1
135 000540 030000FF0201          dc.b    3,0,0,$ff,2,1,$77,0,2,$c8,0,3,$ee,0,$
136 000550 0186000265O1          dc.b    1,$86,0,2,$65,1,3,0,0,$ff,1,1,$77,0,2
137 000563 0177000B200           dc.b    1,$77,0,2,$b2,0,3,$ee,0,$ff,2,1,$7C,0
138 000577 FF0201C80002          dc.b    $ff,2,1,$c8,0,2,$ee,0,$ff,2,1,$cb,2,2
139 000589 01B20002E100          dc.b    1,$b2,0,2,$e1,0,3,$c,1,$ff,2,1,$86,0,
140 00059D FF0201B60002          dc.b    $ff,2,1,$86,0,2,$e1,0,3,$c,1,$ff,2,1,
141 0005B0 030000FF0201          dc.b    3,0,0,$ff,2,1,$86,0,2,$ee,0,3,$2c,1,$
142 0005C0 01960002CB02          dc.b    1,$96,0,2,$cb,2,3,0,0,$ff,2,1,$96,0,2
143 0005D3 01860002CB02          dc.b    1,$86,0,2,$cb,2,$ff,2,1,$86,0,2,$b2,0
144 0005E6 01E100020000          dc.b    1,$e1,0,2,0,0,3,0,0,$ff,1,1,$c,1,$ff,
145 0005F6 016501FF0101          dc.b    1,$65,1,$ff,1,1,$b2,0,2,$ee,0,3,$bb,3
146 000606 01C800020000          dc.b    1,$c8,0,2,0,0,3,0,0,$ff,1,1,$ee,0,$ff
147 00061B 01B20002E100          dc.b    1,$b2,0,2,$e1,0,3,$c,1,$ff,2,1,$cb,2,
148 000631 01E100020C01          dc.b    1,$e1,0,2,$c,1,3,$18,2,$ff,2,1,$cb,2,
149 000647 0170002B600           dc.B    1,$70,0,2,$86,0,3,$a8,0,$ff,2,1,$54,0
150 00065B FF0201B60002          dc.b    $ff,2,1,$86,0,2,$a8,0,$ff,2,1,$77,0,2
151 00066D 0186000220000         dc.b    1,$86,0,2,0,0,$ff,1,1,$a8,0,2,$c8,0,3
152 000680 01EE00020000          dc.b    1,$ee,0,2,0,0,3,0,0,$ff,2,1,$c8,0,$ff
153 000695 0196000B200           dc.b    1,$96,0,2,$b2,0,$ff,2,1,$96,0,2,$e1,0
154 0006A8 02B200FF0101          dc.b    2,$b2,0,$ff,1,1,$4b,0,2,0,0,$ff,1,1,$
155 0006BD 0170002A800           dc.b    1,$70,0,2,$a8,0,3,$e1,0,$ff,2,1,$c,1,
156 0006D3 01E100FF0201          dc.b    1,$e1,0,$ff,2,1,$a8,0,$ff,2
157 0006DD 01A80002CB00          dc.b    1,$a8,0,2,$c8,0,$ff,2,1,$64,0,2,$c,1,
158 0006F0 02A8000BC800          dc.b    2,$a8,0,3,$c8,0,$ff,2,1,$70,0,2,$c,1,
159 000703 0177000220000         dc.b    1,$77,0,2,0,0,$ff,1,1,$77,0,2,$b2,0,3
160 000716 01B20002CB02          dc.b    1,$b2,0,2,$cb,2,3,0,0,$ff,2,1,$54,0,2
161 00072C 0159000B200           dc.b    1,$59,0,2,$b2,0,3,0,0,$ff,2,1,$59,0,2
162 000742 01E100020000          dc.b    1,$e1,0,2,0,0,3,0,0,$ff,1,1,$c,1,$ff,
163 000757 01B2000EE00           dc.b    1,$.2,0,2,$ee,0,$ff,1,1,$c8,0,2,0,0,$
164 000767 01EE00FF0101          dc.b    1,$ee,0,$ff,1,1,$1c,1,$ff,1,1,$b2,0,2
165 00077F FF0201E100FF          dc.b    $ff,2,1,$e1,0,$ff,2,1,$cb,2,$ff,2,1,$
166 000794 FF0201860002          dc.b    $ff,2,1,$86,0,2,$c8,0,3,0,0,$ff,2,1,$
167 0007AC 02CB00FF0101          dc.b    2,$c8,0,$ff,1,1,$86,0,2,0,0,$ff,1,1,$
168 0007C2 FF0201EE0002          dc.b    $ff,2,1,$ee,0,2,0,0,3,0,0,$ff,2,1,$96
169 0007D7 0196000EE00           dc.b    1,$96,C,2,$ee,0,$ff,2,1,$59,0,2,$96,0
170 0007EA C1$B0002E100          dc.b    1,$4b,0,2,$e1,0,3,0,0,$ff,2,1,$70,0,2
171 000800 0170000E100           dc.b    1,$70,0,2,$e1,0,3,0,0,$ff,2,1,$70,0,2
172 000816 010C01020000          dc.b    1,$c,1,2,0,0,3,0,0,$ff,2,1,$a8,0,2,$e
173 000829 010C01020000          dc.b    1,$c,1,2,0,0,$ff,1,1,$70,0,$ff,1,1,$6
174 00083F FF0201B60002          dc.b    $ff,2,1,$86,0,2,$c,1,3,0,0,$ff,2,1,$a
175 000854 0196000020C01         dc.b    1,$96,0,2,$c,1,$ff,2,1,$96,0,2,$b2,0,
176 000867 012C01020000          dc.b    1,$2c,1,2,0,0,3,0,0,$ff,2,1,$b2,0,2,$
177 00C37A 0186000020000         dc.b    1,$86,0,2,0,0,$ff,1,1,$77,0,2,$2c,1,$
178 00088D 020000FF0101          dc.b    2,0,0,$ff,1,1,$86,0,2,$b2,0,$ff,1
179 00089A 01E100020000          dc.b    1,$e1,0,2,0,0,$ff,1,1,$86,0,2,$c,1,$f
180 0008AA 016501020000          dc.b    1,$65,1,2,0,0,$ff,1,1,$3e,1,2,$91,1,$
```

(suite)

```
  181 0008BA 01C800020000          dc.b    1,$c8,0,2,0,0,$ff,1,1,$c,1,$ff,1,1,$3
  182 0008CC 01E100020C01          dc.b    1,$e1,0,2,$c,1,$ff,3,1,0,0 ,2,0,0
  183 0008DA FF0201000FF           dc.b    $ff,2,1,0,0,$ff,1
  184 0008E1 010000020000          dc.b    1,0,0,2,0,0,3,0,0,$ff,3,1,0,0,2,0,0,3
  185 0008F7 010000020000          dc.b    1,0,0,2,0,0,3,0,0,$ff,$ff
No Errors

--SYMBOL TABLE--

DATA......000000EE+ FINMUSE...00000CCC+ LOOP......000000DE+ MUSE......00000000+
MUSE2.....00000046+ PAUSE1....000000AE+ PAUSE2....000000BA+ PAUSE3....000000C6+
PAUSE4....000000D2+ PRELOOP...000000D8+ TRAISPE...000000B4+
```

Deuxième exemple :

Le second programme fonctionne comme le premier. Seules les données changent.

Le listing du programme est le suivant :

```
                        ; Activation de musique sur 3 Voies
                        ;-------------------------------------------------
                        ;Forme: Voie+Frequence ! FF+Tempo(1 a 4^)
                        ;       ! FF+FF (Terminateur)
                        ;-------------------------------------------------
                        muse:
000902 3F3C0087                 move    #$80+7,-(sp)
000906 3F3C00F8                 move    #$f8,-(sp)  ;Voies 1, 2 et 3 Validees
00090A 3F3C001C                 move    #28,-(sp)
00090E 4E4E                     trap    #14
000910 5C8F                     addq.l  #6,sp
000912 3F3C0088                 move    #$80+8,-(sp)
000916 3F3C000C                 move    #12,-(sp)   ;Volume Voie 1
00091A 3F3C001C                 move    #28,-(sp)
00091E 4E4E                     trap    #14
000920 5CA7                     addq.l  #6,-(sp)
000922 3F3C0089                 move    #$80+9,-(sp)
000926 3F3C000C                 move    #12,-(sp)   ;Volume Voie 2
00092A 3F3C001C                 move    #28,-(sp)
00092E 4E4E                     trap    #14
000930 5C8F                     addq.l  #6,sp
000932 3F3C008A                 move    #$80+10,-(sp)
000936 3F3C000C                 move    #12,-(sp)   ;Volume Voie 3
00093A 3F3C001C                 move    #28,-(sp)
00093E 4E4E                     trap    #14
000940 5C8F                     addq.l  #6,sp
000942 2A7C000009F0             move.l  #data,a5    ;donnees musicales
                        muse2:
000948 101D                     move.b  (a5)+,d0    ;dans D0
00094A 0C0000FF                 cmp.b   #$ff,d0     ;Code Special ?
00094E 67000036                 beq     traispe     ;Oui
000952 04000001                 sub.b   #1,d0
000956 D000                     add.b   d0,d0
000958 06000080                 add.b   #$80,d0
00095C 1200                     move.b  d0,d1
00095E 3F00                     move    d0,-(sp)
000960 101D                     move.b  (a5)+,d0    ;Note LSB
000962 3F00                     move.w  d0,-(sp)
000964 3F3C001C                 move.w  #28,-(sp)
000968 4E4E                     trap    #14
00096A 5C8F                     addq.l  #6,sp
00096C 1001                     move.b  d1,d0
00096E 06400001                 add.w   #1,d0
000972 3F00                     move.w  d0,-(sp)
000974 101D                     move.b  (a5)+,d0    ;Note MSB
000976 3F00                     move.w  d0,-(sp)
000978 3F3C001C                 move.w  #28,-(sp)
00097C 4E4E                     trap    #14
00097E 5C8F                     addq.l  #6,sp
000980 4EF900000948             jmp     muse2
```

(suite)

```
                        Traispe: ;FF      dtect
000986 101D                      move.b   (a5)+,d0     ;Code suivant
000988 0C0000FF                  cmp.b    #$ff,d0
00098C 67000250                  beq      finmuse
                        ;-------------------------------------------
                        ;Tempo entre deux notes
                        ;-------------------------------------------
000990 0C000001                  cmp.b    #1,d0
000994 6700001A                  beq      pause1
000998 0C000002                  cmp.b    #2,d0
00099C 6700001E                  beq      pause2
0009A0 0C000003                  cmp.b    #3,d0
0009A4 67000022                  beq      pause3
0009A8 0C000004                  cmp.b    #4,d0
0009AC 67000026                  beq      pause4
0009B0 2C3C0000FFFF pause1:      move.l   #$ffff,d6    ;Pause la + courte
0009B6 4EF900000 9DA             jmp      preloop
0009BC 2C3C0001FFFF pause2:      move.l   #$1ffff,d6
0009C2 4EF900000 9DA             jmp      preloop
0009C8 2C3C0002FFFF pause3:      move.l   #$2ffff,d6
0009CE 4EF900000 9DA             jmp      preloop
0009D4 2C3C0003FFFF pause4:      move.l   #$3ffff,d6   ;Pause la + longue
0009DA 2E3C00000000 preloop:     move.l   #$0,d7
0009E0 5287         loop:        addq.l   #1,d7
0009E2 BE86                      cmp.l    d6,d7
0009E4 6600FFFA                  bne      loop
0009E8 4EF900000948              jmp      muse2
                        ;-------------------------------------------
                        ;Fin du morceau
                        ;-------------------------------------------
                        finmuse:
0009EE 4E75                      rts

0009F0 01860002E100 data:        dc.b     1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,0,0,2,
000A06 01860002E100              dc.b     1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,
000A1C 01860002E100              dc.b     1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,0,0,2,
000A32 01860002E100              dc.b     1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,
000A48 01860002E100              dc.b     1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,
000A5E 01860002E100              dc.b     1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,
000A74 01860002E100              dc.b     1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,
000A8A 01770002E100              dc.b     1,$77,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,
000AA0 01770002960 0             dc.b     1,$77,0,2,$96,0,3,$65,1,$ff,1,1,$96,0
000AB6 01960002EE00              dc.b     1,$96,0,2,$ee,0,3,$65,1,$ff,1,1,$96,0
000ACC 01960002EE00              dc.b     1,$96,0,2,$ee,0,3,$65,1,$ff,1,1,$96,0
000AE2 01B20002EE00              dc.b     1,$b2,0,2,$ee,0,3,$65,1,$ff,1,1,$b2,0
000AF8 01960002EE00              dc.b     1,$96,0,2,$ee,0,3,$65,1,$ff,1,1,$86,0
000B0E 01860002EE03              dc.b     1,$86,0,2,$ee,0,3,$65,1,$ff,1,1,$86,0
000B24 01000002960 0             dc.b     1,0,0,2,$96,0,3,$65,1,$ff,1,2,0,0,3,$
000B37 02960003650 1             dc.b     2,$96,0,3,$65,1,$ff,1,2,0,0,3,$cb,2,$
```

(suite)

```
107 000B47 020C01039101          dc.b    2,$c,1,3,$91,1,$ff,1,2,0,0,3,$23,3,$f
108 000B57 020C01039101          dc.b    2,$c,1,3,$91,1,$ff,1,2,0,0,3,$23,3,$f
109 000B67 020C01039101          dc.b    2,$c,1,3,$91,1,$ff,1,2,0,0,3,$23,3,$f
110 000B77 01A800020C01          dc.b    1,$a8,0,2,$c,1,3,$91,1,$ff,1,1,$a8,0,
111 000B8D 01A800020C01          dc.b    1,$a8,0,2,$c,1,3,$91,1,$ff,1,1,$a8,0,
112 000BA3 01B200020C01          dc.b    1,$b2,0,2,$c,1,3,$91,1,$ff,1,1,$b2,0,
113 000BB9 01B200020C01          dc.b    1,$b2,0,2,$c,1,3,$91,1,$ff,1,1,$b2,0,
114 000BCF 01C800020C01          dc.b    1,$c8,0,2,$c,1,3,$91,1,$ff,1,1,$c8,0,
115 000BE5 01B200020C01          dc.b    1,$b2,0,2,$c,1,3,$65,1,$ff,1,1,$b2,0,
116 000BFB 01B200020C01          dc.b    1,$b2,0,2,$c,1,3,$65,1,$ff,1,1,$b2,0,
117 000C11 01B200020C01          dc.b    1,$b2,0,2,$c,1,3,$65,1,$ff,1,1,$b2,0,
118 000C27 01B200020C01          dc.b    1,$b2,0,2,$c,1,3,$65,1,$ff,1,1,$b2,0,
119 000C3D 010000022C01          dc.b    1,0,0,2,$2c,1,3,$65,1,$ff,1,2,0,0,3,$
120 000C50 022C01036501          dc.b    2,$2c,1,3,$65,1,$ff,1,2,0,0,3,$cb,2,$
121 000C60 022C01036501          dc.b    2,$2c,1,3,$65,1,$ff,1,2,0,0,3,$cb,2,$
122 000C70 01B200022C01          dc.b    1,$b2,0,2,$2c,1,3,$65,1,$ff,1,1,$b2,0
123 000C86 01860002E100          dc.b    1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,
124 000C9C 01860002E100          dc.b    1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,
125 000CB2 01860002E100          dc.b    1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,
126 000CC8 01860002E100          dc.b    1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,
127 000CDE 01860002E100          dc.b    1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,
128                              dc.b
129 000CF4 01000002E100          dc.b    1,0,0,2,$e1,0,3,$c,1,$ff,1,2,0,0,3,$1
130 000D07 01770002E100          dc.b    1,$77,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,
131 000D1D 017700022C01          dc.b    1,$77,0,2,$2c,1,3,$65,1,$ff,1,1,$77,0
132 000D33 017700022C01          dc.b    1,$77,0,2,$2c,1,3,$65,1,$ff,1,1,$77,0
133 000D49 017700022C01          dc.b    1,$77,0,2,$2c,1,3,$65,1,$ff,1,1,$96,0
134 000D5F 01B200022C01          dc.b    1,$b2,0,2,$2c,1,3,$65,1,$ff,1,1,$b2,0
135 000D75 019600022C01          dc.b    1,$96,0,2,$2c,1,3,$65,1,$ff,1,1,$96,0
136 000D8B 018600022C01          dc.b    1,$86,0,2,$2c,1,3,$65,1,$ff,1,1,$86,0
137 000DA1 018600022C01          dc.b    1,$86,0,2,$2c,1,3,$65,1,$ff,1,1,$86,0
138 000DB7 018600022C01          dc.b    1,$86,0,2,$2c,1,3,$65,1,$ff,1,1,$86,0
139 000DCD 010000020C01          dc.b    1,0,0,2,$c,1,3,$91,1,$ff,1,2,$2c,1,3,
140 000DE0 020C01039101          dc.b    2,$c,1,3,$91,1,$ff,1,2,$2c,1,3,$23,3,
141 000DF0 020C01039101          dc.b    2,$c,1,3,$91,1,$ff,1,2,$2c,1,3,$23,3,
142 000E00 01A800025101          dc.b    1,$a8,0,2,$51,1,3,$91,1,$ff,1,1,$a8,0
143 000E16 01A800025101          dc.b    1,$a8,0,2,$51,1,3,$91,1,$ff,1,1,$a8,0
144 000E2C 01A800025101          dc.b    1,$a8,0,2,$51,1,3,$91,1,$ff,1,1,$a8,0
145 000E42 01B200025101          dc.b    1,$b2,0,2,$51,1,3,$71,1,$ff,1,1,$b2,0
146 000E58 01C800025101          dc.b    1,$c8,0,2,$51,1,3,$91,1,$ff,1,1,$b2,0
147 000E6E 01B202022C01          dc.b    1,$b2,0,2,$2c,1,3,$65,1,$ff,1,1,$b2,0
148 000E84 01B200022C01          dc.b    1,$b2,0,2,$2c,1,3,$65,1,$ff,1,1,$b2,0
149 000E9A 01B200022C01          dc.b    1,$b2,0,2,$2c,1,3,$65,1,$ff,1,1,$b2,0
150 000EB0 01B200022C01          dc.b    1,$b2,0,2,$2c,1,3,$65,1,$ff,1,1,$b2,0
151 000EC6 017000022C01          dc.b    1,$70,0,2,$2c,1,3,$65,1,$ff,1,1,$70,0
152 000EDC 018600022C01          dc.b    1,$86,0,2,$2c,1,3,$65,1,$ff,1,1,$86,0
153 000EF2 019600022C01          dc.b    1,$96,0,2,$2c,1,3,$65,1,$ff,1,1,$96,0
154 000F08 018600022C01          dc.b    1,$86,0,2,$2c,1,3,$65,1,$ff,1,1,$86,0
155 000F1E 01860002E100          dc.b    1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,
156 000F34 01860002E100          dc.b    1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,
157 000F4A 01000002E100          dc.b    1,0,0,2,$e1,0,3,$c,1,$ff,1,2,0,0,3,$1
158 000F5D 01B20002E100          dc.b    1,$b2,0,2,$e1,0,3,$c,1,$ff,1,1,$b2,0,
159 000F73 01860002E100          dc.b    1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,
160 000F89 01770002E100          dc.b    1,$77,0,2,$e1,0,3,$c,1,$ff,1,1,$77,0,
161 000F9F 01860002E100          dc.b    1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,
162 000FB5 01770002E100          dc.b    1,$77,0,2,$e1,0,3,$c,1,$ff,1,1,$77,0,
```

(suite)

```
163 000FCB 0177000021C01        dc.b    1,$77,0,2,$1c,1,3,$65,1,$ff,1,1,$77,0
164                             dc
165 000FE1 0100000021C01        dc.b    1,0,0,2,$1c,1,3,$65,1,$ff,1,2,0,0,3,$
166 000FF4 021C01036501         dc.b    2,$1c,1,3,$65,1,$ff,1,2,0,0,3,$dd,1,$
167 001004 0177000021C01        dc.b    1,$77,0,2,$1c,1,3,$65,1,$ff,1,1,$77,0
168 00101A 0170000021C01        dc.b    1,$70,0,2,$1c,1,3,$65,1,$ff,1,1,$70,0
169 001030 0177000021C01        dc.b    1,$77,0,2,$1c,1,3,$65,1,$ff,1,1,$77,0
170 001046 0170000021C01        dc.b    1,$70,0,2,$1c,1,3,$65,1,$ff,1,1,$70,0
171 00105C 017000002E100        dc.b    1,$70,0,2,$e1,0,3,$c,1,$ff,1,1,$70,0,
172 001072 017000002E100        dc.b    1,$70,0,2,$e1,0,3,$c,1,$ff,1,1,$70,0,
173 001088 010000002E100        dc.b    1,0,0,2,$e1,0,3,$c,1,$ff,1,2,0,0,3,$1
174 00109B 02E10003C01          dc.b    2,$e1,0,3,$c,1,$ff,1,2,0,0,3,$16,2,$f
175 0010AB 017000002E100        dc.b    1,$70,0,2,$e1,0,3,$c,1,$ff,1,1,$70,0,
176 0010C1 016400002E100        dc.b    1,$64,0,2,$e1,0,3,$c,1,$ff,1,1,$64,0,
177 0010D7 015900002E100        dc.b    1,$59,0,2,$e1,0,3,$c,1,$ff,1,1,$59,0,
178 0010ED 015400002E100        dc.b    1,$54,0,2,$e1,0,3,$c,1,$ff,1,1,$54,0,
179 001103 01540002510          dc.b    1,$54,0,2,$51,1,3,$91,1,$ff,1,1,$54,0
180 001119 01540002510          dc.b    1,$54,0,2,$51,1,3,$91,1,$ff,1,1,$54,0
181 00112F 01590002510          dc.b    1,$59,0,2,$51,1,3,$91,1,$ff,1,1,$59,0
182 001145 01640002510          dc.b    1,$64,0,2,$51,1,3,$91,1,$ff,1,1,$64,0
183 00115B 0164000021C01        dc.b    1,$64,0,2,$1c,1,3.$65,1,$ff,1,1,$64,0
184 001171 0159000021C01        dc.b    1,$59,0,2,$1c,1,3,$65,1,$ff,1,1,$59,0
185 001187 0159000021C01        dc.b    1,$59,0,2,$1c,1,3,$65,1,$ff,1,1,$59,0
186 00119D 0170000021C01        dc.b    1,$70,0,2,$1c,1,3,$65,1,$ff,1,1,$77,0
187 0011B3 016600002E100        dc.b    1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,
188 0011C9 016600002E100        dc.b    1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,
189 0011DF 016600002E100        dc.b    1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,
190 0011F5 010000002E100        dc.b    1,0,0,2,$e1,0,3,$c,1,$ff,1,2,0,0,3,$1
191 001208 016600002E100        dc.b    1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,
192 00121E 017700002E100        dc.b    1,$77,0,2,$e1,0,3,$c,1,$ff,1,1,$77,0,
193 001234 016600002E100        dc.b    1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,
194 00124A 017700002E100        dc.b    1,$77,0,2,$e1,0,3,$c,1,$ff,1,1,$77,0,
195 001260 0177000021C01        dc.b    1,$77,0,2,$1c,1,3,$65,1,$ff,1,1,$77,0
196 001276 0177000021C01        dc.b    1,$77,0,2,$1c,1,3,$65,1,$ff,1,1,$77,0
197 00128C 0177000021C01        dc.b    1,$77,0,2,$1c,1,3,$65,1,$ff,1,1,0,0,2
198 0012A2 021C01036501         dc.b    2,$1c,1,3,$65,1,$ff,1,2,0,0,3,$dd,1,$
199 0012B2 0177000021C01        dc.b    1,$77,0,2,$1c,1,3,$65,1,$ff,1,1,$77,0
200 0012C8 0170000021C01        dc.b    1,$70,0,2,$1c,1,3,$65,1,$ff,1,1,$70,0
201 0012DE 0164000021C01        dc.b    1,$64,0,2,$1c,1,3,$65,1,$ff,1,1,$70,0
202 0012F4 0164000021C01        dc.b    1,$64,0,2,$1c,1,3,$65,1,$ff,1,1,$70,0
203 00130A 017000002E100        dc.b    1,$70,0,2,$e1,0,3,$c,1,$ff,1,1,$70,0,
204 001320 017000002E100        dc.b    1,$70,0,2,$e1,0,3,$c,1,$ff,1,1,$70,0,
205 001336 017000002E100        dc.b    1,$70,0,2,$e1,0,3,$c,1,$ff,1,1,0,0,2,
206 00134C 02E10003C01          dc.b    2,$e1,0,3,$c,1,$ff,1,2,0,0,3,$18,2,$f
207 00135C 017000002E100        dc.b    1,$70,0,2,$e1,0,3,$c,1,$ff,1,1,$70,0,
208 001372 016400002E100        dc.b    1,$64,0,2,$e1,0,3,$c,1,$ff,1,1,$64,0,
209 001388 015900002E100        dc.b    1,$59,0,2,$e1,0,3,$c,1,$ff,1,1,$59,0,
210 00139E 015400002E100        dc.b    1,$54,0,2,$e1,0,3,$c,1,$ff,1,1,$54,0,
211 0013B4 017000002E100        dc.b    1,$70,0,2,$e1,0,3,$c,1,$ff,1,1,$59,0,
212 0013CA 018600002E100        dc.b    1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$70,2,
213 0013E0 01B200002E100        dc.b    1,$b2,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,
214 0013F6 01E100002E100        dc.b    1,$e1,0,2,$e1,0,3,$c,1,$ff,1,1,$b2,0,
215 00140C 010C0102E100         dc.b    1,$c,1,2,$e1,0,3,$c,1,$ff,1,1,$c,1,2,
216 001422 010C0102E100         dc.b    1,$c,1,2,$e1,0,3,$c,1,$ff,1,1,$c,1,2,
217 001438 010000002E100        dc.b    1,0,0,2,$e1,0,3,$c,1,$ff,1,1,$59,0,2,
218 00144E 016400002E100        dc.b    1,$64,0,2,$e1,0,3,$c,1,$ff,1,1,$59,0,
```

(suite)

```
 219 001464 017700021C01          dc.b    1,$77,0,2,$1c,1,3,$65,1,$ff,1,1,$64,0
 220 00147A 018E00021C01          dc.b    1,$8e,0,2,$1c,1,3,$65,1,$ff,1,1,$77,0
 221 001490 01B200021C01          dc.b    1,$b2,0,2,$1c,1,3,$65,1,$ff,1,1,$8e,0
 222 0014A6 01C800021C01          dc.b    1,$c8,0,2,$1c,1,3,$65,1,$ff,1,1,$b2,0
 223 0014BC 01EE00021C01          dc.b    1,$ee,0,2,$1c,1,3,$65,1,$ff,1,1,$ee,0
 224 0014D2 01EE00021C01          dc.b    1,$ee,0,2,$1c,1,3,$65,1,$ff,1,1,$ee,0
 225 0014E8 010000021C01          dc.b    1,0,0,2,$1c,1,3,$65,1,$ff,1,1,$59,0,2
 226 0014FE 016400021C01          dc.b    1,$64,0,2,$1c,1,3,$65,1,$ff,1,1,$59,0
 227 001514 017000002E100         dc.b    1,$70,0,2,$e1,0,3,$c,1,$ff,1,1,$59,0,
 228 00152A 018600002E100         dc.b    1,$86,0,2,$e1,0,3,$c,1,$ff,1,1,$70,0,
 229 001540 01B200002E100         dc.b    1,$b2,0,2,$e1,0,3,$c,1,$ff,1,1,$86,0,
 230 001556 01E100002E100         dc.b    1,$e1,0,2,$e1,0,3,$c,1,$ff,1,1,$b2,0,
 231 00156C 010C0102E100          dc.b    1,$c,1,2,$e1,0,3,$c,1,$ff,1,1,$c,1,2,
 232 001582 010C0102E100          dc.b    1,$c,1,2,$e1,0,3,$c,1,$ff,1,1,$c,1,2,
 233 001598 010000002E100         dc.b    1,0,0,2,$e1,0,3,$c,1,$ff,1,1,$59,0,2,
 234 0015AE 016400002E100         dc.b    1,$64,0,2,$e1,0,3,$c,1,$ff,1,1,$59,0,
 235 0015C4 017700021C01          dc.b    1,$77,0,2,$1c,1,3,$65,1,$ff,1,1,$64,0
 236 0015DA 018E00021C01          dc.b    1,$8e,0,2,$1c,1,3,$65,1,$ff,1,1,$77,0
 237 0015F0 01B200021C01          dc.b    1,$b2,0,2,$1c,1,3,$65,1,$ff,1,1,$8e,0
 238 001606 01C200021C01          dc.b    1,$c8,0,2,$1c,1,3,$65,1,$ff,1,1,$b2,0
 239 C0161C 01EE01021C01          dc.b    1,$ee,1,2,$1c,1,3,$65,1,$ff,1,1,$ee,0
 240 001632 01EE00021C01          dc.b    1,$ee,0,2,$1c,1,3,$65,1,$ff,1,1,$ee,0
 241 00164B 010000021C01          dc.b    1,0,0,2,$1c,1,3,$65,1,$ff,1,2,0,0,3,$
 242 00165B 021C01036501          dc.b    2,$1c,1,3,$65,1,$ff,1,2,0,0,3,$cb,2,$
 243 00166B 01B200026501          dc.b    1,$b2,0,2,$65,1,3,$65,1,$ff,1,1,$b65,
 244 001681 010000026501          dc.b    1,0,0,2,$65,1,3,$65,1,$ff,1,2,$cb,2,3
 245 001694 020000036501          dc.b    2,0,0,3,$65,1,$ff,1,3,$cb,2,$ff,1
 246 0016A1 036501FF0103          dc.b    3,$65,1,$ff,1,3,$cb,2,$ff,1,3,$65,1,$
 247
 248
 249
 250
 251
 252 0016B5 010000020000          dc.b    1,0,0,2,0,0,3,0,0,$ff,$ff
 253
No Errors

--SYMBOL TABLE--

DATA.....00C309F0+ FINMUSE...000009EE+ LOOP.....000009E0+ MUSE.....00000902+
MUSE2....00000948+ PAUSE1....000009E0+ PAUSE2....000009BC+ PAUSE3....000009C8+
PAUSE4....000009D4+ PRELOOP...000009DA+ TRAISPE...00000986+
```

6/1.3.8

Créer un éditeur de partitions

La programmation d'un éditeur de partitions n'est pas une chose très simple, en effet l'éditeur allie la musique, notion de hauteur et de durée, à la notion de graphisme, position écran. De plus ce graphisme doit être rapide et la sélection d'objets aisée. Le déplacement dans la partition doit se faire conjointement à l'écran et dans la chaîne à jouer afin de pouvoir modifier ou supprimer des éléments.

L'éditeur proposé ici est très simpliste comparé à un logiciel professionnel, mais vous apprécierez ses performances de simplicité et de rapidité.

DESCRIPTIF GÉNÉRAL DU PROGRAMME

Séquenceur pas à pas utilisant la fonction XBIOS(32) avec affichage de la partition.

Construction de la séquence à l'aide de la souris.

Correction à l'aide de la souris, insertion, suppression.

Affichage en clé de FA et clé de SOL.

Affichage des dièses.

Cinq durées de notes de la ronde à la double croche, ainsi que les silences correspondants.

Scrolling de la partition.

Sauvegarde au format XBIOS(32) pour une utilisation à l'intérieur de vos programmes.

A ajouter si vous le désirez :

L'affichage des signes clé de FA et clé de SOL

Changement de tempo

Barre de mesures (attention aux liaisons)

PROGRAMMATION

Codage des chaînes

Un problème se pose quant au choix de codage. Pour jouer la chaîne avec XBIOS(32), il faut absolument respecter le codage imposé par le construc

teur. Ce codage n'est malheureusement pas le plus performant pour l'affichage, puisque avant de retrouver la hauteur d'affichage de la note, il faudrait reconstituer la fréquence à partir de XBIOS(32), puis faire un test d'octave, puis le nom de la note, puis la correspondance graphique. Tous ces tests ralentissent considérablement la vitesse de traitement graphique.

Autre solution, créer une chaîne destinée à une gestion rapide du graphisme, et lorsque l'on désire entendre le résultat, convertir la chaîne graphique au format XBIOS(32). Mais le problème, bien que moindre, existe toujours un certain temps de conversion est nécessaire. Ceci est insupportable!!, car un ordinateur se doit d'être rapide.

Il ne reste qu'une seule solution, en fait la plus souple, créer deux chaînes distinctes, l'une au format XBIOS(32) pour la partie sonore, l'autre à un format permettant une gestion rapide de l'affichage.

Dans le programme, la chaîne servant à la fonction XBIOS(32) se nomme **piste$** et la chaîne gérant l'affichage se nomme **affi$**. Le codage de la chaîne **piste$** vous est connu puisqu'il a été utilisé précédemment. A vide piste$ possède deux caractères, à chaque nouvelle donnée, **piste$** augmente de 6 octets.

Le codage de la chaîne **affi$** est le suivant :

A chaque note ou silence, la chaîne augmente de trois octets.

Descriptif de la chaîne :

1er octet : N° du graphisme pour la_note$( ).

De 0 à 4, ronde, blanche, noire, croche, double croche.

De 5 à 9, pause, demi-pause, soupir, demi-soupir, quart de soupir.

2e octet : Position en x de l'affichage.

3e octet : Test pour l'affichage d'un dièse, si 0 normal, sinon affichage du symbole du dièse.

Lors d'un déplacement d'une note dans la partition, le pointeur de piste$ s'incrémentera de 6, alors que celui de affi$ s'incrémentera de 3.

Chaque chaîne possède son propre pointeur. Ce pointeur bien que différent désigne le même emplacement, c'est la première note située à gauche de l'écran.

Pour la chaîne **piste$**, le pointeur **p_chaîne%**.

Pour la chaîne **affi$**, le pointeur **p_affi%**.

Sauvegarde et chargement

La sauvegarde et le chargement des fichiers sont doublés par l'existence de deux chaînes distinctes. Il est possible de réunir les deux chaînes en une seule en ajoutant des pointeurs, mais cette solution complique l'implantation dans vos programmes de la seule chaîne **piste$** pour XBIOS(32).



Au chargement, les deux fichiers avec les suffixes .AFF et X32 doivent être présents au même niveau. Attention, le logiciel ne fait pas la gestion des erreurs.

Graphisme

Pour l'affichage des notes nous disposons de sprites.

Malheureusement, les sprites proposés par l'Atari ne font que 16*16 points, ce qui est trop peut pour notre programme. Une définition sur 24 points de hauteur est plus adaptée. De plus le sprite est lent et occupe beaucoup plus de place mémoire, le masque et la position étant sauvegardés.

Pour remplacer les sprites, nous utilisons la fonction **PUT** qui est logiquement issue de la fonction GET.

Lorsque vous définissez une portion de l'écran avec **GET**, le résultat est dans une chaîne de caractères. En scrutant la chaîne, on s'aperçoit que les six premiers caractères sont réservés à la description du motif, et les caractères suivants définissent le motif.

Les quatre premiers octets de la chaîne définissent la longueur et la hauteur de la portion écran sous forme de mots. Dans le programme nous construisons de toutes pièces une fonction PUT sans utiliser de **GET**. La chaîne **entete$** définit la longueur et la hauteur du sprite, suivent les motifs sous format binaire.

Chaque sprite est ensuite mis en mémoire dans le tableau de chaîne **la_note$( )**.

Position d'affichage

La position de chacune des notes est fonction pour les x de la valeur contenue dans **affi$**, pour les y c'est la variable **position%** qui gère l'emplacement. Après un offset de 60 qui représente le bord de l'écran, s'ajoute à cette variable 320/(2^le type de la note), modulo 5 pour que les silences soient égaux aux notes.

Sélection et suppression de la note sur la partition

La sélection de la note s'effectue par comparaison de la position souris et du constant calcul de position des notes à l'affichage. La note est repérée avant qu'un cliquage de souris n'intervienne, ce qui accélère le traitement.

Scrolling

Le scrolling s'effectue dans les deux sens. Les pointeurs **p_affi%** et **p_piste%** sont incrémentés respectivement de 3 et 6 pour avancer, et décrémentés de 3 et 6 pour reculer.

Affichage

L'affichage complet de la partition à partir des pointeurs de chaînes s'effectue à chaque changement : l'ajout d'une note, l'insertion d'une note, la suppression d'une note et le scrolling.

DESCRIPTIF DU LISTING

Procédures charge_sprites

Cette procédure permet le chargement des 10 sprites représentant les notes et silences dans le tableau de chaîne la_note$( ), ainsi que le sprite diese$. Chacun des sprites, après ajout de l'entête enteteS, est dans un format compatible avec la fonction PUT. Le format binaire pour la définition des sprites permet de visualiser directement le résultat du sprite durant sa programmation. Vous pouvez aussi améliorer le graphisme, en utilisant le mode texte 8 du GFA pour créer de nouveaux sprites.

Procédure charge_les_variables

Chargement des variables :

option$( ) : Tableau de chaîne pour les commandes du logiciel.

haut_note$: Tableau de chaîne pour l'affichage de la hauteur de la note durant l'ajout ou l'insertion. L'affichage apparaît à la place de l'option QUIT.

fre% : Tableau de mots pour la conversion haut de la note et gestion de la puce son. Ces données sont ensuite converties sur deux octets.

hauteur_affi% : Tableau contenant la hauteur d'affichage réelle de la note, sans tenir compte des dièses.

note_affi% : Tableau contenant la hauteur de la note avant l'affichage, ce tableau permet de différencier une note normale d'une note diésée. S'il y a inégalité entre hauteur_affi% et note_affi%, c'est que la note est diésée.

Procédure fond

Cette procédure dessine les lignes de portées ainsi que la présentation générale du logiciel. Les lignes sont sauvegardées avec un GET afin d'être envoyées avant chaque affichage.

Procédure instal_note

Procédure principale du programme, elle permet la construction de la séquence. Dès l'entrée dans cette procédure la hauteur de la note est constamment testée, après relâchement de la souris, l'on teste si le relâchement s'est effectué dans la zone d'affichage.

Si oui, test de dièse pour les notes, test si c'est une insertion ou un ajout, test si dépassement en longueur pour l'affichage, si oui scrolling avant.

Procédure construire_la_chaine

Rajoute des données en bout des deux chaînes.

Procédure insérer_la_chaîne

Insère les nouvelles données dans les deux chaînes.

Procédure supprimer_dans_la_chaîne

Supprime une note et reconditionne la longueur des chaînes.

Procédure jouer_la_chaîne

Envoie de la chaîne piste$ vers XBIOS(32).

Procédure avance_d_un

Scrolling de la partition vers l'avant, les pointeurs de chaînes sont incrémentés.

Procédure recul_d_un

Scrolling de la partition vers l'arrière, les pointeurs de chaînes sont décrémentés.

Procédure affi_déplacement

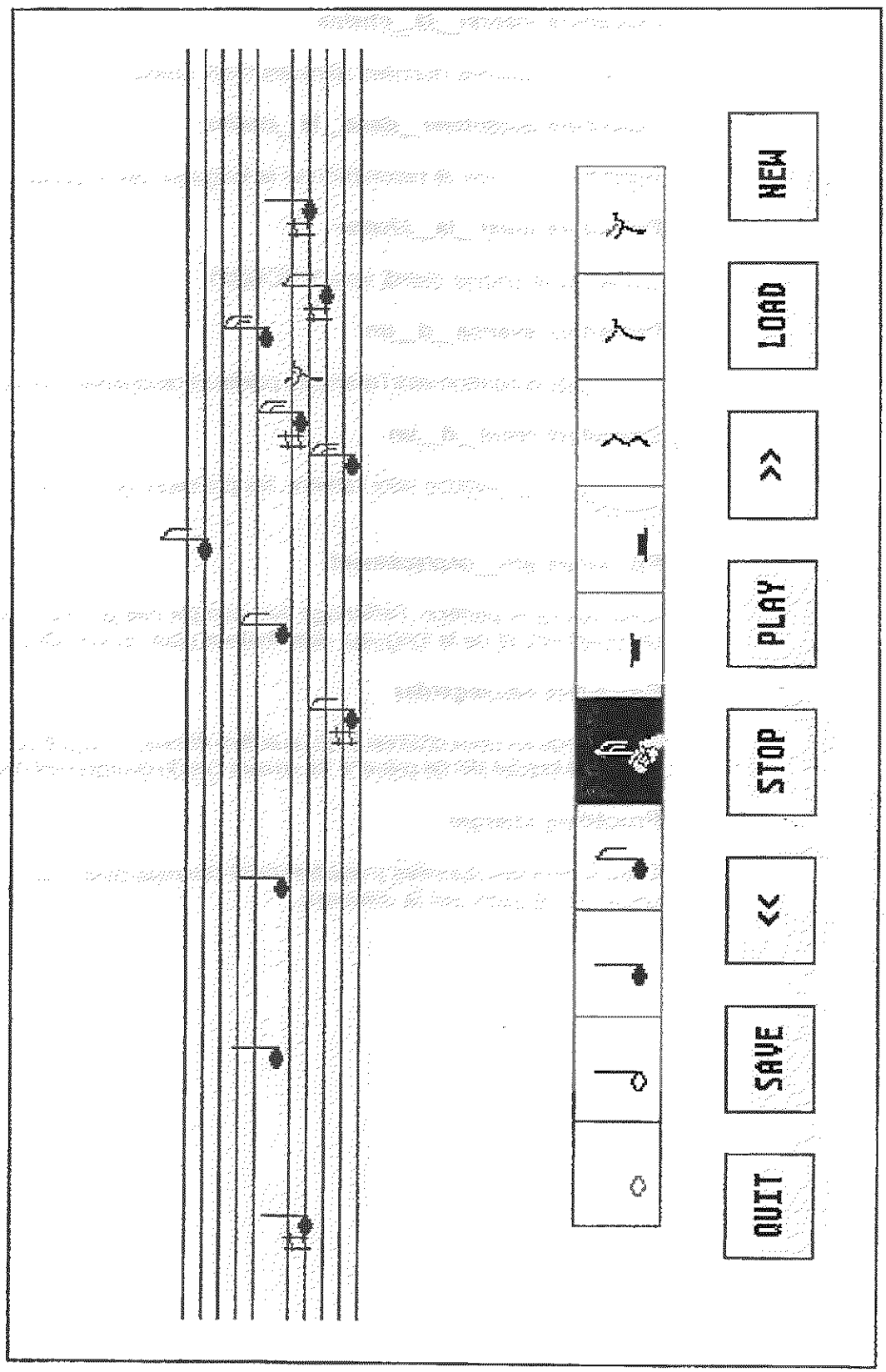
Affichage de la partition, l'affichage tient compte des pointeurs de chaînes, de position% et de la longueur des chaînes pour ne pas afficher du vide.

Procédure sauvegarder

Sauvegarde les deux chaînes avec un suffixe différent. Les suffixes sont imposés par le logiciel afin de prévenir les erreurs lors du chargement des données.

Procédure charger

Chargement des données précédemment sauvegardées, sans test de présence des fichiers sur la disquette.


NEW
LOAD
>>
PLAY
STOP
<<
SAVE
QUIT

```
'
'       Editeur de partitions
'
'       pour la
'
'       fonction
'
'       Dosound     XBIOS(32)
'
'       Emploi d'une chaine pour Dosound et d'une
'       chaîne différente pour l'affichage
'
'
charge_sprites
charge_variables
fond
DO
  MOUSE x%,y%,k|
  IF y%<200          ! selection d'une note écrite
    b&=60
    FOR p%=p_affi% TO LEN(affi$) STEP 3
      WHILEx%>b&-5ANDx%<b&+15ANDMOUSEY<200ANDLEN(affi$)<>0
        x%=MOUSEX
        IF MOUSEK=2
          supprimer_dans_chaine
          position%=60
          affi_deplacement
          REPEAT
          UNTIL MOUSEK=0
          x%=0
        ENDIF
      WEND
      b&=b&+(320/2^MOD(ASC(MID$(affi$,p%,1)),5))
    NEXT p%
  ENDIF
  IF k|=1
    IF y%>260 AND y%<300 AND x%>65 AND x%<550
      instal_note
    ENDIF
    FOR a&=50 TO 560 STEP 70
      IF x%>a& AND y%>330 AND x%<a&+50 AND y%<370
        SELECT INT((x%-50)/70)
        CASE 0
          END
        CASE 1
          sauvegarder
        CASE 2
          recul_d_un
        CASE 3
          SPOKE &HFF8800,7
          SPOKE &HFF8802,255
        CASE 4
          jouer_la_chaine
        CASE 5
          avance_d_un
        CASE 6
```

(suite)

```
            charger
          CASE 7
            position%=60
            affi$=""
            p_affi%=1
            piste$=CHR$(7)+CHR$(254)
            PUT 40,45,ligne$
          ENDSELECT
        ENDIF
      NEXT a&
    ENDIF
  LOOP
  PROCEDURE charge_sprites
    DIM la_note$(9)
    entete$=MKI$(15)+MKI$(23)+MKI$(1)
    ronde$=entete$
    FOR a%=1 TO 24
      READ z%
      ronde$=ronde$+MKI$(z%)
    NEXT a%
    DATA %0000000000000000
    DATA %0000000000000000
    DATA %0000000000000000
    DATA %0000000000000000
    DATA %0000000000000000
    DATA %0000000000000000
    DATA %0000000000000000
    DATA %0000000000000000
    DATA %0000000000000000
    DATA %0000000000000000
    DATA %0000000000000000
    DATA %0000000000000000
    DATA %0000000000000000
    DATA %0000000000000000
    DATA %0000000000000000
    DATA %0000000000000000
    DATA %0000000000000000
    DATA %0001110000000000
    DATA %0110001100000000
    DATA %1000000011000000
    DATA %1000000001000000
    DATA %1100000011000000
    DATA %0011001100000000
    DATA %0001110000000000
    la_note$(0)=ronde$
    blanche$=entete$
    FOR a%=1 TO 24
      READ z%
      blanche$=blanche$+MKI$(z%)
    NEXT a%
    DATA %0000000001000000
    DATA %0000000001000000
    DATA %0000000001000000
    DATA %0000000001000000
    DATA %0000000001000000
    DATA %0000000001000000
```

(suite)

```
  DATA %0000000001000000
  DATA %0000000001000000
  DATA %0000000001000000
  DATA %0000000001000000
  DATA %0000000001000000
  DATA %0000000001000000
  DATA %0000000001000000
  DATA %0000000001000000
  DATA %0000000001000000
  DATA %0000000001000000
  DATA %0001110001000000
  DATA %0110001101000000
  DATA %1000000011000000
  DATA %1000000001000000
  DATA %1100000011000000
  DATA %0011001100000000
  DATA %0001110000000000
  la_note$(1)=blanche$
  noire$=entete$
  FOR a%=1 TO 24
    READ z%
    noire$=noire$+MKI$(z%)
  NEXT a%
  DATA %0000000001000000
  DATA %0000000001000000
  DATA %0000000001000000
  DATA %0000000001000000
  DATA %0000000001000000
  DATA %0000000001000000
  DATA %0000000001000000
  DATA %0000000001000000
  DATA %0000000001000000
  DATA %0000000001000000
  DATA %0000000001000000
  DATA %0000000001000000
  DATA %0000000001000000
  DATA %0000000001000000
  DATA %0001110001000000
  DATA %0111111101000000
  DATA %1111111111000000
  DATA %1111111111000000
  DATA %1111111111000000
  DATA %0111111100000000
  DATA %0001110000000000
  la_note$(2)=noire$
  croche$=entete$
  FOR a%=1 TO 24
    READ z%
    croche$=croche$+MKI$(z%)
  NEXT a%
  DATA %0000000001100000
  DATA %0000000001100000
```

(suite)

```
DATA %0000000001011000
DATA %0000000001000100
DATA %0000000001000110
DATA %0000000001000010
DATA %0000000001000010
DATA %0000000001000010
DATA %0000000001000010
DATA %0000000001000010
DATA %0000000001000010
DATA %0000000001000010
DATA %0000000001000010
DATA %0000000001000000
DATA %0000000001000000
DATA %0000000001000000
DATA %0001110001000000
DATA %0111111101000000
DATA %1111111111000000
DATA %1111111111000000
DATA %1111111111000000
DATA %0111111100000000
DATA %0001110000000000
la_note$(3)=croche$
d_croche$=entete$
FOR a%=1 TO 24
  READ z%
  d_croche$=d_croche$+MKI$(z%)
NEXT a%
DATA %0000000001100000
DATA %0000000001100000
DATA %0000000001011000
DATA %0000000001000100
DATA %0000000001000110
DATA %0000000001000010
DATA %0000000001100010
DATA %0000000001010010
DATA %0000000001001010
DATA %0000000001001010
DATA %0000000001001010
DATA %0000000001001010
DATA %0000000001001010
DATA %0000000001000010
DATA %0000000001000000
DATA %0000000001000000
DATA %0000000001000000
DATA %0001110001000000
DATA %0111111101000000
DATA %1111111111000000
DATA %1111111111000000
DATA %1111111111000000
DATA %0111111100000000
DATA %0001110000000000
la_note$(4)=d_croche$
pause$=entete$
FOR a%=1 TO 24
  READ z%
```

(suite)

```
  pause$=pause$+MKI$(z%)
NEXT a%
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %1111111111111111
DATA %0001111111111000
DATA %0001111111111000
DATA %0001111111111000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
la_note$(5)=pause$
d_pause$=entete$
FOR a%=1 TO 24
  READ z%
  d_pause$=d_pause$+MKI$(z%)
NEXT a%
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0001111111111000
DATA %0001111111111000
DATA %0001111111111000
DATA %1111111111111111
DATA %0000000000000000
```

(suite)

```
la_note$(6)=d_pause$
soupir$=entete$
FOR a%=1 TO 24
  READ z%
  soupir$=soupir$+MKI$(z%)
NEXT a%
DATA %0000000000000000
DATA %0000000000000000
DATA %0000100000000000
DATA %0000110000000000
DATA %0000011000000000
DATA %0000011000000000
DATA %0000000100000000
DATA %0000000011000000
DATA %0000000110000000
DATA %0000000100000000
DATA %0000001100000000
DATA %0000011000000000
DATA %0000010000000000
DATA %0000010000000000
DATA %0000110000000000
DATA %0000001000000000
DATA %0000001100000000
DATA %0000000110000000
DATA %0000000110000000
DATA %0000000110000000
DATA %0000001100000000
DATA %0000011000000000
DATA %0000111000000000
DATA %0000100000000000
la_note$(7)=soupir$
d_soupir$=entete$
FOR a%=1 TO 24
  READ z%
  d_soupir$=d_soupir$+MKI$(z%)
NEXT a%
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0001100000000000
DATA %0000110000000000
DATA %0000010000000000
DATA %0000001100000100
DATA %0000000010111100
DATA %0000000000100000
DATA %0000000000100000
DATA %0000000011000000
DATA %0000001000000000
DATA %0000001100000000
DATA %0000001000000000
DATA %0000010000000000
DATA %0000110000000000
DATA %0000110000000000
```

(suite)

```
DATA %0000110000000000
DATA %0000010000000000
DATA %0000000000000000
DATA %0000000000000000
la_note$(8)=d_soupir$
q_soupir$=entete$
FOR a%=1 TO 24
  READ z%
  q_soupir$=q_soupir$+MKI$(z%)
NEXT a%
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0001100000000000
DATA %0000110000000000
DATA %0000010000000000
DATA %0000001100000100
DATA %0010000010111100
DATA %0001000000100000
DATA %0000111000100000
DATA %0000001111000000
DATA %0000000100000000
DATA %0000001100000000
DATA %0000001000000000
DATA %0000011000000000
DATA %0000010000000000
DATA %0000010000000000
DATA %0000110000000000
DATA %0000110000000000
DATA %0000110000000000
DATA %0000010000000000
DATA %0000000000000000
DATA %0000000000000000
la_note$(9)=q_soupir$
diese$=entete$
FOR a%=1 TO 24
  READ z%
  diese$=diese$+MKI$(z%)
NEXT a%
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000000000000
DATA %0000000010000000
DATA %0001000010000000
DATA %0001000010000000
DATA %0001001111100000
DATA %0011110010000000
DATA %0001000010000000
```

(suite)

```
    DATA %0001000010000000
    DATA %0001000010000000
    DATA %0001000010000000
    DATA %0001001111100000
    DATA %0011110010000000
    DATA %0001000010000000
    DATA %0001000000000000
    DATA %0000000000000000
  RETURN
  PROCEDURE charge_variables
    DIM option$(7)
    FOR a%=0 TO 7
      READ option$(a%)
    NEXT a%
    DATA QUIT,SAVE, <<,STOP,PLAY, >>,LOAD,NEW
    DIM haut_note$(12)
    FOR a%=0 TO 12
      READ haut_note$(a%)
    NEXT a%
    DATA DO ,DO#,RE,RE#,MI,FA,FA#,SOL,SOL#,LA,LA#,SI,DO
    DIM fre%(12)
    FOR a%=0 TO 12
      READ fre%(a%)
    NEXT a%
    DATA 856,807,762,719,679,641,605,571,539,508,480,453,428
    DIM hauteur_affi%(13)
    FOR a%=1 TO 13
      READ hauteur_affi%(a%)
    NEXT a%
    DATA 2,6,6,10,10,14,14,18,22,22,26,26,28
    DIM note_affi%(13)
    FOR a%=1 TO 13
      READ note_affi%(a%)
    NEXT a%
    DATA 2,4,6,8,10,12,14,18,20,22,24,26,28
    position%=60
    affi$=""
    p_affi%=1
    SPOKE &HFF8800,8
    SPOKE &HFF8802,15
    SPOKE &HFF8800,7
    SPOKE &HFF8802,255
    piste$=CHR$(7)+CHR$(254)
    tempo=100
    p_piste%=3
  RETURN
  PROCEDURE fond
    FOR a&=80 TO 116 STEP 8
      LINE 20,a&,620,a&
      LINE 20,a&+48,620,a&+48
    NEXT a&
    GET 40,45,639,205,ligne$
    FOR a&=65 TO 540 STEP 50
      BOX a&,260,a&+50,300
      PUT a&+15,270,la_note$((a&-50)/50)
    NEXT a&
```

(suite)

```
    FOR a&=50 TO 560 STEP 70
      BOX a&,330,a&+50,370
      TEXT a&+10,355,option$((a&-50)/70)
    NEXT a&
  RETURN
  PROCEDURE instal_note
    al=INT((x%-65)/50)
    GRAPHMODE 3
    DEFMOUSE 3
    boitel=al
    PBOX (boitel*50)+65,260,(boitel*50)+115,300
    GRAPHMODE 1
    REPEAT
      MOUSE x%,y%,kl
      IF y%<190 AND y%>59 AND al<5
        FOR a%=0 TO 12
          IF     MOD(y%-66,28)>=note_affi%(a%)     AND
MOD(y%-66,28)<note_affi%(a%+1)
            h$=haut_note$(12-a%)
            TEXT 60,355,h$+SPACE$(4-LEN(h$))
            octave%=INT(y%-66)/28
            aa%=a%
          ENDIF
        NEXT a%
        SHOWM
      ENDIF
    UNTIL kl=0
    TEXT 60,355,"QUIT"
    IF y%<190 AND y%>60
      die%=0
      pos%=66+(28*octave%)+hauteur_affi%(aa%)-20
      IF note_affi%(aa%)<>hauteur_affi%(aa%)
        die%=1
      ENDIF
      b&=60
      insert=0
      FOR p%=p_affi% TO LEN(affi$)-6 STEP 3
        b&=b&+(320/2^MOD(ASC(MID$(affi$,p%,1)),5))
        IF x%<b&
          inserer_la_chaine
          p%=LEN(affi$)
          insert=1
        ENDIF
      NEXT p%
      IF insert=0
        construire_la_chaine
      ENDIF
      ADD position%,320/(2^MOD(al,5))
      IF position%>610
        ADD p_piste%,3
        avance_d_un
      ELSE
        position%=60
        affi_deplacement
      ENDIF
    ENDIF
```

(suite)

```
    GRAPHMODE 3
    PBOX (boitel*50)+65,260,(boitel*50)+115,300
    GRAPHMODE 1
    DEFMOUSE 0
  RETURN
  PROCEDURE construire_la_chaine
    IF al<5
      frequence%=fre%(12-aa%)*(2^(octave%-1))
      piste$=piste$+CHR$(0)+CHR$(frequence% AND 255)
      piste$=piste$+CHR$(1)+CHR$(INT(frequence%/256))
      piste$=piste$+CHR$(133)+CHR$(tempo/(2^al))
    ELSE
      piste$=piste$+CHR$(7)+CHR$(255)
      piste$=piste$+CHR$(133)+CHR$(tempo/(2^(al-5)))
      piste$=piste$+CHR$(7)+CHR$(254)
    ENDIF
    affi$=affi$+CHR$(al)+CHR$(pos%)+CHR$(die%)!type,position,diese
  RETURN
  PROCEDURE inserer_la_chaine
    IF al<5
      frequence%=fre%(12-aa%)*(2^(octave%-1))
      don$=CHR$(0)+CHR$(frequence% AND 255)
      don$=don$+CHR$(1)+CHR$(INT(frequence%/256))
      don$=don$+CHR$(133)+CHR$(tempo/(2^al))
    ELSE
      don$=CHR$(7)+CHR$(255)
      don$=don$+CHR$(133)+CHR$(tempo/(2^(al-5)))
      don$=don$+CHR$(7)+CHR$(254)
    ENDIF

  piste$=LEFT$(piste$,(p%*2)+6)+don$+RIGHT$(piste$,LEN(piste$)-((p%*2)
  +6))
    daffi$=CHR$(al)+CHR$(pos%)+CHR$(die%)!type,position,diese
    affi$=LEFT$(affi$,p%+2)+daffi$+RIGHT$(affi$,LEN(affi$)-(p%+2))
  RETURN
  PROCEDURE supprimer_dans_chaine
    piste$=LEFT$(piste$,(p%*2))+RIGHT$(piste$,LEN(piste$)-((p%*2)+6))
    affi$=LEFT$(affi$,p%-1)+RIGHT$(affi$,LEN(affi$)-(p%+2))
  RETURN
  PROCEDURE jouer_la_chaine
    a_jouer$=piste$+CHR$(7)+CHR$(255)+CHR$(255)+CHR$(0)
    ~XBIOS(32,L:VARPTR(a_jouer$))
  RETURN
  PROCEDURE avance_d_un
    position%=60
    ADD p_affi%,3
    p_affi%=MIN(p_affi%,LEN(affi$)-2)
    p_affi%=MAX(0,p_affi%)
    affi_deplacement
  RETURN
  PROCEDURE recul_d_un
    position%=60
    SUB p_affi%,3
    p_affi%=MAX(p_affi%,1)
    affi_deplacement
  RETURN
```

(suite)

```
PROCEDURE affi_deplacement
  PUT 40,45,ligne$
  FOR a%=p_affi% TO LEN(affi$)-2 STEP 3
    al=ASC(MID$(affi$,a%))
    IF al<5
      PUT position%,ASC(MID$(affi$,a%+1)),la_note$(al),7
      IF ASC(MID$(affi$,a%+2))=1
        PUT position%-10,ASC(MID$(affi$,a%+1)),diese$,7
      ENDIF
    ELSE
      PUT position%,121,la_note$(al),7
      SUB al,5
    ENDIF
    ADD position%,320/(2^al)
    EXIT IF position%>610
  NEXT a%
RETURN
PROCEDURE sauvegarder
  FILESELECT "\*.X32","",le_fichier$
  IF le_fichier$<>"\" AND le_fichier$>""
    a_jouer$=piste$+CHR$(7)+CHR$(255)+CHR$(255)+CHR$(0)
    IF INSTR(le_fichier$,".")=0
      le_fichier$=le_fichier$+".X32"
    ENDIF
    IF RIGHT$(le_fichier$)<>".X32"
      p_p=INSTR(le_fichier$,".")
      le_fichier$=LEFT$(le_fichier$,p_p)+"X32"
    ENDIF
    BSAVE le_fichier$,VARPTR(a_jouer$),LEN(a_jouer$)
    MID$(le_fichier$,LEN(le_fichier$)-2,3)="AFF"
    BSAVE le_fichier$,VARPTR(affi$),LEN(affi$)
  ENDIF
RETURN
PROCEDURE charger
  FILESELECT "\*.X32","",le_fichier$
  IF le_fichier$<>"\" AND le_fichier$>""
    IF INSTR(le_fichier$,".")=0
      le_fichier$=le_fichier$+".X32"
    ENDIF
    IF RIGHT$(le_fichier$)<>".X32"
      p_p=INSTR(le_fichier$,".")
      le_fichier$=LEFT$(le_fichier$,p_p)+"X32"
    ENDIF
    piste$=SPACE$(32767)
    BLOAD le_fichier$,VARPTR(piste$)
    piste$=TRIM$(piste$)
    piste$=LEFT$(piste$,LEN(piste$)-4)
    MID$(le_fichier$,LEN(le_fichier$)-2,3)="AFF"
    affi$=SPACE$(32767)
    BLOAD le_fichier$,VARPTR(affi$)
    affi$=TRIM$(affi$)
    position%=60
    p_affi%=1
    PUT 40,45,ligne$
    affi_deplacement
  ENDIF
RETURN
```

6/1.3.9

Saisie graphique de morceaux de musique

Ce programme s'adresse à tous les mélomanes qui désirent avoir des copies propres de leurs œuvres. Il permet de saisir et d'imprimer facilement et rapidement des notes de musique comprises entre le LA GRAVE et le DO AIGU de la clé de sol.

Les notes saisies peuvent bien entendu comporter des bémols et des dièses.

Leur durée peut être l'une des suivantes :

— TC : triple croche,

— DC : double croche,

— Cr : croche,

— No : noire,

— Bl : blanche,

— Ro : ronde.

Il est également possible d'utiliser des pauses égales à 1/4, 1/2, 1, 2 et 4 temps.

Les morceaux saisis peuvent être sauvegardés sur disque dans un fichier ASCII et affichés sur l'écran. L'appui simultané sur les touches <Alternate> et <Help> du clavier de l'ATARI vous permettent d'imprimer les notes affichées sur l'écran.

COMMENT UTILISER LE PROGRAMME

Le programme est écrit en GFA Basic. Voici son listing :

```
REM ========================================
REM Saisie graphique de morceaux de musique
REM ========================================
REM
REM Dimentionnement des tableaux du programme
DIM no(200),alt(200),dur(200),paus(200)
boucle_principale:
REM ------------------------------------
REM Affichage de l'écran de saisie
REM ------------------------------------
CLS
LINE 10,110,360,110
LINE 10,116,360,116
LINE 10,122,360,122
LINE 10,128,360,128
LINE 10,134,360,134
LINE 15,140,25,140
LINE 15,146,25,146
LINE 35,140,45,140
LINE 35,146,45,146
LINE 55,140,65,140
LINE 295,104,305,104
LINE 315,104,325,104
LINE 315,98,325,98
LINE 335,104,345,104
LINE 335,98,345,98
FOR i=1 TO 17
  PCIRCLE 20*i,149-i*3,3
NEXT i
LOCATE 12,5
PRINT "Dièse    Bémol    Bécarre"
LOCATE 14,5
PRINT "TC DC Cr No Bl Ro                    VALID
E   EFFACE"
LOCATE 16,5
PRINT "Pause  1/4  1/2  1    2    4"
LOCATE 20,25
PRINT "CHARGE   SAUVE   AFFICHE"
LOCATE 22,35
PRINT "FIN"
REM
REM ------------------------
REM Acquisition des notes
REM ------------------------
index=1 !Index dans le tableau des données
LOCATE 12,45
PRINT "Note 1 :"
REPEAT
  fin=FALSE
  ok=FALSE
  REPEAT
  UNTIL MOUSEK=1 !Attente de l'appui sur le bouton gauche
de la souris
  REPEAT
```



(suite)

```
  UNTIL MOUSEK=0 !Attente du relachement du bouton gauche de la
souris
  note=0
  REM ------------------------------------------------------
  REM Action en fonction du choix de l'utilisateur
  REM ------------------------------------------------------
  IF (MOUSEX>272) AND (MOUSEX<296) AND (MOUSEY>336) AND (MOUSEY<
352) THEN
    ok=TRUE
    fin=TRUE !L'option "FIN" a été sélectionnée
  ENDIF
  FOR i=1 TO 17
    IF (MOUSEX>20*i-5) AND (MOUSEX<20*i+5) AND (MOUSEY>144-i*3)
AND (MOUSEY<154-i*3) THEN
      note=i
      ok=TRUE
    ENDIF
  NEXT i
  IF note<>0 THEN
    paus(index)=0 !Pas de pause
    LOCATE 12,56
    PRINT SPACE$(30)
    no(index)=18-note !Mémorisation de la note
    LOCATE 12,56
    SELECT note
    CASE 1,8,15
      PRINT "LA "
    CASE 2,9,16
      PRINT "SI "
    CASE 3,10,17
      PRINT "DO "
    CASE 4,11
      PRINT "RE "
    CASE 5,12
      PRINT "MI "
    CASE 6,13
      PRINT "FA "
    CASE 7,14
      PRINT "SOL"
    ENDSELECT
  ENDIF
  IF (MOUSEY>176) AND (MOUSEY<192) THEN
    LOCATE 12,59
    IF (MOUSEX>32) AND (MOUSEX<72) THEN
      ok=TRUE
      PRINT "#"
      alt(index)=1
    ENDIF
    IF (MOUSEX>104) AND (MOUSEX<144) THEN
      ok=TRUE
      PRINT "b"
      alt(index)=2
    ENDIF
    IF (MOUSEX>176) AND (MOUSEX<232) THEN
```

(suite)

```
      ok=TRUE
      PRINT " "
      alt(index)=0
    ENDIF
  ENDIF
  IF (MOUSEY>208) AND (MOUSEY<224) THEN
    LOCATE 12,61
    IF (MOUSEX>32) AND (MOUSEX<48) THEN
      ok=TRUE
      PRINT "TC"
      dur(index)=1
    ENDIF
    IF (MOUSEX>56) AND (MOUSEX<72) THEN
      ok=TRUE
      PRINT "DC"
      dur(index)=2
    ENDIF
    IF (MOUSEX>80) AND (MOUSEX<96) THEN
      ok=TRUE
      PRINT "Cr"
      dur(index)=3
    ENDIF
    IF (MOUSEX>104) AND (MOUSEX<120) THEN
      ok=TRUE
      PRINT "No"
      dur(index)=4
    ENDIF
    IF (MOUSEX>128) AND (MOUSEX<144) THEN
      ok=TRUE
      PRINT "Bl"
      dur(index)=5
    ENDIF
    IF (MOUSEX>152) AND (MOUSEX<168) THEN
      ok=TRUE
      PRINT "Ro"
      dur(index)=6
    ENDIF
    IF (MOUSEX>416) AND (MOUSEX<464) THEN
      ok=TRUE
      LOCATE 12,56
      PRINT SPACE$(30)
      index=index+1
      LOCATE 12,45
      PRINT "Note ";index;" : "
    ENDIF
    IF (MOUSEX>488) AND (MOUSEX<536) THEN
      ok=TRUE
      LOCATE 12,56
      PRINT SPACE$(30)
    ENDIF
  ENDIF
  IF (MOUSEY>240) AND (MOUSEY<256) THEN
    no(index)=0
    alt(index)=0
```

(suite)

```
      dur(index)=0
      LOCATE 12,56
      PRINT "Pause : ";
      IF (MOUSEX>88) AND (MOUSEX<112) THEN
        ok=TRUE
        PRINT "1/4 temps"
        paus(index)=1
      ENDIF
      IF (MOUSEX>128) AND (MOUSEX<152) THEN
        ok=TRUE
        PRINT "1/2 temps"
        paus(index)=2
      ENDIF
      IF (MOUSEX>168) AND (MOUSEX<176) THEN
        ok=TRUE
        PRINT "1 temps  "
        paus(index)=3
      ENDIF
      IF (MOUSEX>208) AND (MOUSEX<216) THEN
        ok=TRUE
        PRINT "2 temps  "
        paus(index)=4
      ENDIF
      IF (MOUSEX>248) AND (MOUSEX<256) THEN
        ok=TRUE
        PRINT "4 temps  "
        paus(index)=5
      ENDIF
    ENDIF
    IF (MOUSEY>304) AND (MOUSEY<320) THEN
      IF (MOUSEX>192) AND (MOUSEX<240) THEN
        ok=TRUE
        CLS
        INPUT "Entrez le nom du fichier : ";nf$
        OPEN "I",#1,nf$
        INPUT #1,index
        sauv_index=index
        FOR i=1 TO index
          INPUT #1,no(i)
          INPUT #1,alt(i)
          INPUT #1,dur(i)
          INPUT #1,paus(i)
        NEXT i
        CLOSE #1
        GOTO boucle_principale
      ENDIF
      IF (MOUSEX>264) AND (MOUSEX<304) THEN
        ok=TRUE
        CLS
        INPUT "Entrez le nom du fichier : ";nf$
        OPEN "O",#1,nf$
        PRINT #1,index-1
        FOR i=1 TO index-1
          PRINT #1,no(i)
```

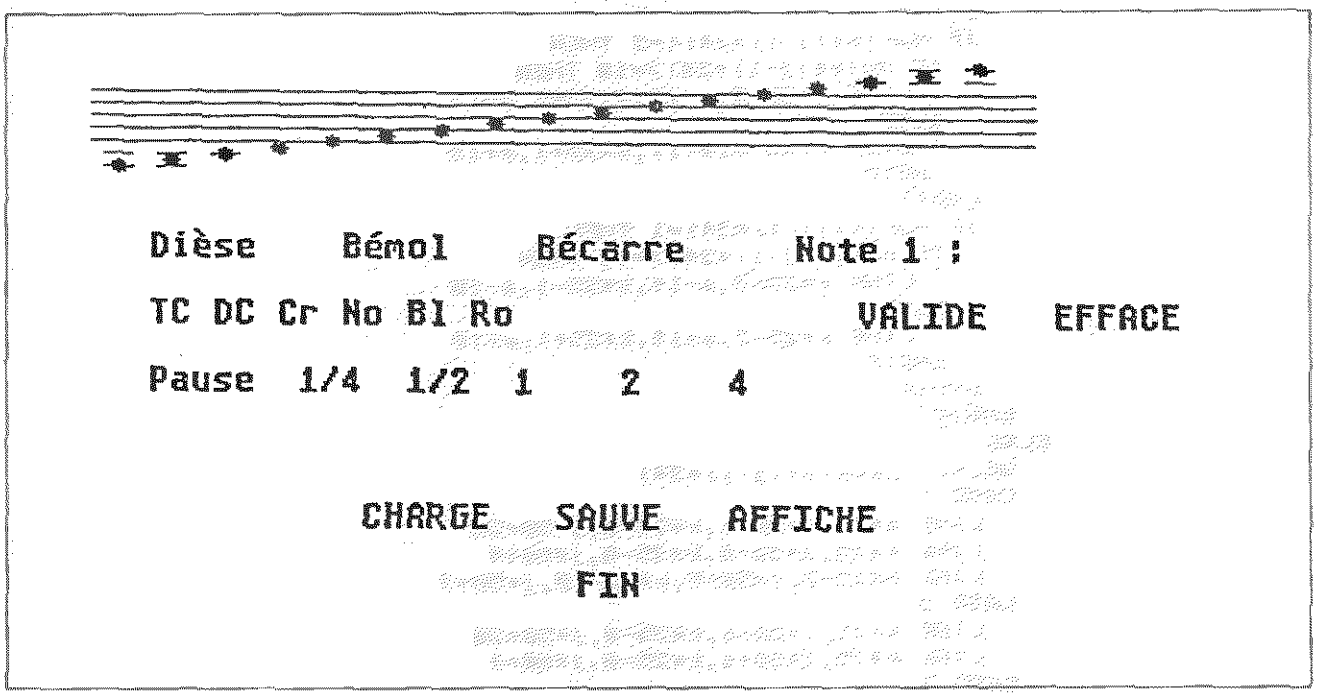
(suite)

```
      PRINT #1,alt(i)
      PRINT #1,dur(i)
      PRINT #1,paus(i)
    NEXT i
    CLOSE #1
    GOTO boucle_principale
  ENDIF
  IF (MOUSEX>328) AND (MOUSEX<384) THEN
    ok=TRUE
    CLS
    j=0
    FOR z=1 TO MAX(index-1,sauv_index-1) STEP 24
      j=j+1
      LINE 10,j*50,610,j*50
      LINE 10,j*50+6,610,j*50+6
      LINE 10,j*50+12,610,j*50+12
      LINE 10,j*50+18,610,j*50+18
      LINE 10,j*50+24,610,j*50+24
      FOR k=1 TO 24
        IF paus(k+(j-1)*24)=0 THEN
          IF alt(k+(j-1)*24)=2 THEN
            TEXT k*25-14,no(k+(j-1)*24)*3+(j-1)*50+40,"b"
          ENDIF
          IF alt(k+(j-1)*24)=1 THEN
            TEXT k*25-14,no(k+(j-1)*24)*3+(j-1)*50+42,"#"
          ENDIF
          IF (k+(j-1)*24)<=MAX(index,sauv_index) THEN
            IF no(k+(j-1)*24)>14 THEN
              LINE k*25-5,(j-1)*50+80,k*25+5,(j-1)*50+80
              LINE k*25-5,(j-1)*50+86,k*25+5,(j-1)*50+86
            ENDIF
            IF no(k+(j-1)*24)<4 THEN
              LINE k*25-5,(j-1)*50+38,k*25+5,(j-1)*50+38
              LINE k*25-5,(j-1)*50+44,k*25+5,(j-1)*50+44
            ENDIF
            IF dur(k+(j-1)*24)>5 THEN
              CIRCLE k*25,no(k+(j-1)*24)*3+(j-1)*50+35,3
            ELSE
              PCIRCLE k*25,no(k+(j-1)*24)*3+(j-1)*50+35,3
            ENDIF
            IF (no(k+(j-1)*24)<10) AND (dur(k+(j-1)*24)<>6) THEN
              a=no(k+(j-1)*24)*3+(j-1)*50+35
              LINE k*25-3,a,k*25-3,a+20
            ENDIF
            IF (no(k+(j-1)*24)>=10) AND (dur(k+(j-1)*24)<>6) THEN
              a=no(k+(j-1)*24)*3+(j-1)*50+35
              LINE k*25+3,a,k*25+3,a-20
            ENDIF
            IF dur(k+(j-1)*24)<=3 THEN
              IF no(k+(j-1)*24)>=10 THEN
                LINE k*25+3,a-20,k*25-1,a-16
              ELSE
                LINE k*25-3,a+20,k*25+1,a+16
              ENDIF
```

(suite)

```
                ENDIF
                IF dur(k+(j-1)*24)<=2 THEN
                  IF no(k+(j-1)*24)>=10 THEN
                    LINE k*25+3,a-17,k*25-1,a-13
                  ELSE
                    LINE k*25-3,a+17,k*25+1,a+13
                  ENDIF
                ENDIF
                IF dur(k+(j-1)*24)=1 THEN
                  IF no(k+(j-1)*24)>=10 THEN
                    LINE k*25+3,a-14,k*25-1,a-10
                  ELSE
                    LINE k*25-3,a+14,k*25+1,a+10
                  ENDIF
                ENDIF
              ENDIF
            ELSE
              SELECT paus(k+(j-1)*24)
              CASE 1
                LINE k*25,j*50+6,k*25-8,j*50+20
                LINE k*25,j*50+6,k*25-8,j*50+4
                LINE k*25-2,j*50+9,k*25-10,j*50+7
              CASE 2
                LINE k*25,j*50+6,k*25-8,j*50+20
                LINE k*25,j*50+6,k*25-8,j*50+4
              CASE 3
                LINE k*25,j*50+6,k*25+3,j*50+9
                DRAW  TO k*25,j*50+12
                DRAW  TO k*25+3,j*50+15
                DRAW  TO k*25,j*50+18
              CASE 4
                LINE k*25,j*50+3,k*25+10,j*50+3
                PBOX k*25+3,j*50+4,k*25+7,j*50+7
              CASE 5
                LINE k*25,j*50+5,k*25+10,j*50+5
                PBOX k*25+3,j*50+1,k*25+7,j*50+4
              ENDSELECT
            ENDIF
          NEXT k
        NEXT z
        REPEAT
        UNTIL INKEY$<>""
        GOTO boucle_principale
      ENDIF
    ENDIF
    IF NOT ok THEN !Aucun option valide l'a été sélectionnée
      SOUND 1,15,10,4,5
      SOUND 1,0,0,0,0
    ENDIF
  UNTIL fin
  END
```

Entrez soigneusement les lignes du listing et lancez le programme. Un écran de saisie du type suivant apparaît :



Toutes les informations affichées sur l'écran peuvent être accédées à l'aide du bouton gauche de la souris. Elles peuvent être réparties en trois groupes.

Premier groupe

Notes :

— les notes sur un peu plus de deux octaves ;

— les altérations dièse, bémol et bécarre ;

— les durées TC (triple croche), DC (double croche), Cr (croche), No (noire), Bl (blanche) et Ro (ronde) ;

— les pauses de 1/4, 1/2, 1, 2 et 4 temps.

Second groupe

Validation/Annulation d'une note :

— l'option VALIDE enregistre la note courante et positionne le pointeur de note sur la prochaine note ;

— l'option EFFACE efface la note courante. Le pointeur de note n'est pas modifié.

Troisième groupe

Affichage et disque

— les notes saisies peuvent être sauvegardées dans un fichier disque à l'aide de l'option SAUVE ;

— inversement, elles peuvent être chargées en mémoire à l'aide de l'option CHARGE ;

— l'option AFFICHE affiche sur l'écran les notes en mémoire ;

— enfin, l'option FIN termine le programme et redonne le contrôle au TOS.

Pour saisir un morceau, suivez les étapes suivantes :

— pour saisir une note :

1) cliquez sur la note désirée dans la portée. Le nom de la note cliquée apparaît en clair sur l'écran. Si vous cliquez sur une autre note, elle remplace la précédente.

2) cliquez éventuellement sur l'altération de la note (dièse ou bémol). Cliquez sur **Bécarre** pour annuler une altération erronée. Par défaut les notes saisies ne sont pas altérées.

3) cliquez sur la durée de la note (TC, DC, Cr, No, Bl ou Ro).

4) validez la note entrée en cliquant sur l'option **VALIDE**. Si vous avez entré une mauvaise note, vous pouvez également cliquer sur l'option **EFFACE**.

— pour saisir une pause :

1) cliquez sur la durée désirée (1/4, 1/2, 1, 2 ou 4 temps) ;

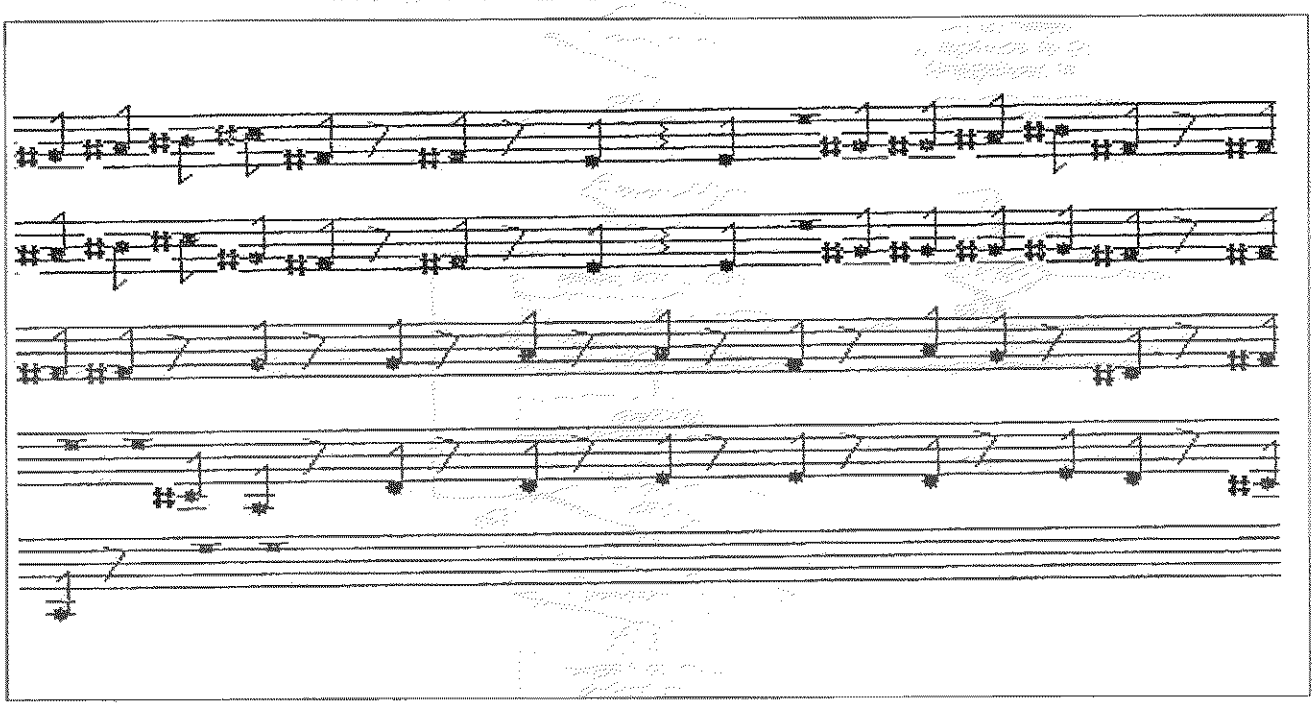
2) validez la pause en cliquant sur **VALIDE**. L'option **EFFACE** peut également être utilisée pour effacer la pause entrée.

— lorsque tout le morceau a été saisi, vous pouvez :

- le sauvegarder sur disque en cliquant sur l'option **SAUVE** et en entrant le nom du fichier de sauvegarde. Prochainement, nous verrons comment utiliser ce fichier pour activer le générateur sonore.

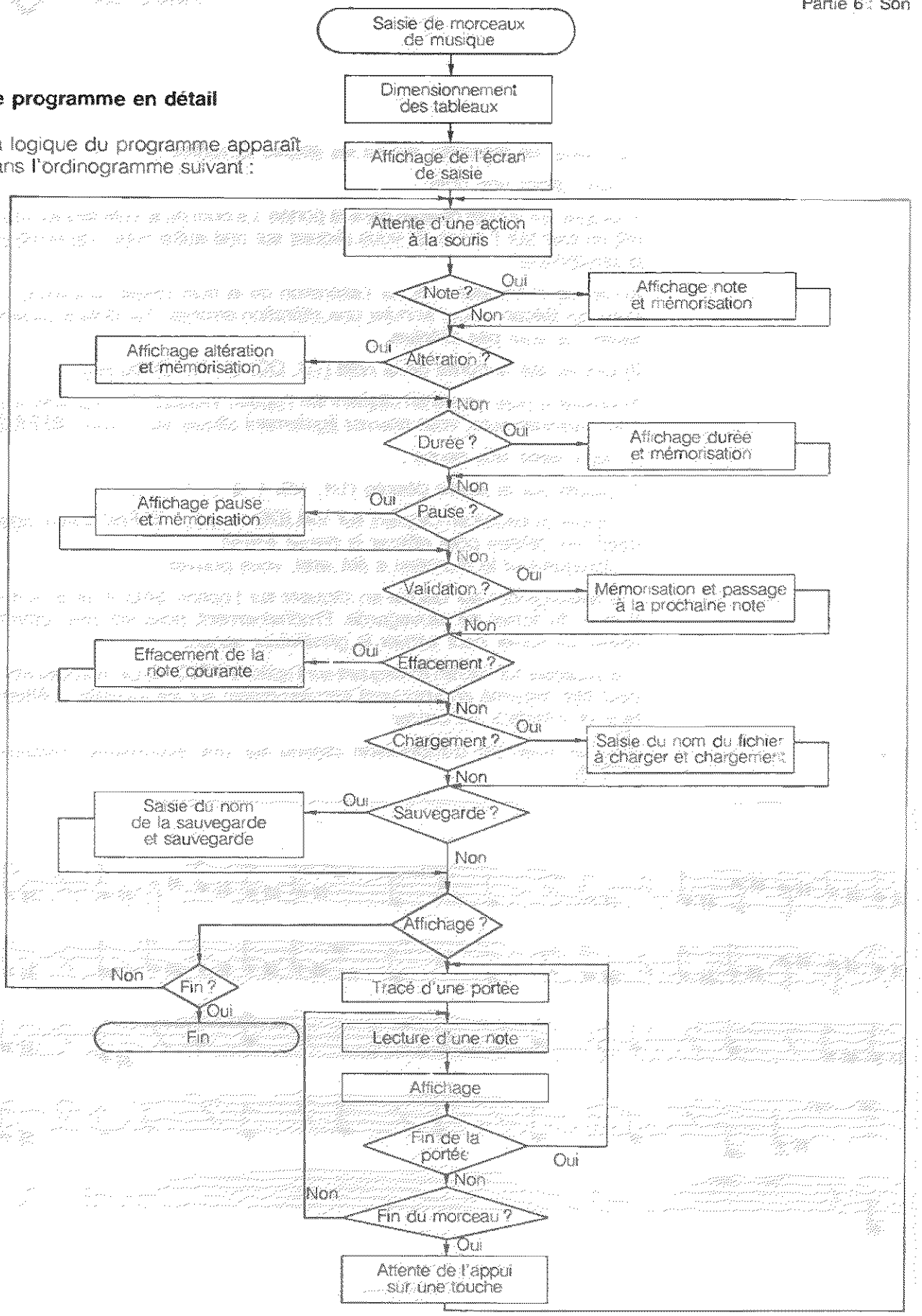
- le visualiser sur l'écran en cliquant sur l'option **AFFICHE**. Le morceau affiché peut être imprimé en appuyant simultanément sur les touches **<Alternate>** et **<Help>** du clavier.

Voici un exemple d'impression obtenu sur une imprimante compatible EPSON :



Le programme en détail

La logique du programme apparaît dans l'ordinogramme suivant :





Le programme débute par le dimensionnement des tableaux qu'il utilise :

```
DIM no(200),alt(200),dur(200),paus(200)
```

no est le tableau des notes ;
alt est le tableau des altérations ;
dur est le tableau des durées ;
paus est le tableau des pauses.

Il se poursuit par la boucle principale.

Dans un premier temps, l'écran de saisie est affiché. Tout d'abord la portée :

```
LINE 10,110,360,110
...
LINE 335,98,345,98
```

puis les notes :

```
FOR i=1 TO 17
  PCIRCLE 20*i,149-i*3,3
NEXT i
```

Les diverses options sont enfin affichées à l'aide d'une série de LOCATE/PRINT :

```
LOCATE 12,5
PRINT "Dièse    Bémol    Bécarre"
LOCATE 14,5
PRINT "TC DC Cr No Bl Ro                    VALIDE    EFFACE"
LOCATE 16,5
PRINT "Pause    1/4    1/2    1    2    4"
LOCATE 20,25
PRINT "CHARGE    SAUVE    AFFICHE"
LOCATE 22,35
PRINT "FIN"
```

Le programme se met en attente d'un appui sur le bouton gauche de la souris :

```
REPEAT
UNTIL MOUSEK=1 !Attente de l'appui sur le bouton gauche de la souris
```

puis de son relâchement :

```
REPEAT
UNTIL MOUSEK=0 !Attente du relâchement du bouton gauche de la souris
```

Si le curseur de la souris se trouvait sur l'option FIN lors du clic :

```
IF (MOUSEX>272) AND (MOUSEX<296) AND (MOUSEY>336) AND (MOUSEY<352) THEN
```

le choix est considéré valide, et la variable **fin** est initialisée à TRUE, ce qui mettra fin à la boucle principale :

```
ok=TRUE
fin=TRUE !L'option ''FIN'' a été sélectionnée
```

Si le curseur de la souris se trouvait sur l'une des notes de la portée lors du clic :

```
FOR i=1 TO 17
  IF (MOUSEX>20*i-5) AND (MOUSEX<20*i+5) AND (MOUSEY
  >144-i*3)
AND (MOUSEY<154-i*3) THEN
```

Le choix est considéré valide, et la variable note est initialisée en conséquence :

```
    note=i
    ok=TRUE
```

La note sélectionnée est affichée en clair dans la partie droite de l'écran, à l'aide d'une structure SELECT CASE :

```
no(index)=18-note !Mémorisation de la note
LOCATE 12,56
SELECT note
CASE 1,8,15
  PRINT ''LA''
...
CASE 7,14
  PRINT ''SOL''
ENDSELECT
```

Si le curseur de la souris se trouvait sur une altération lors du clic, le choix est considéré valide, l'altération est mémorisée et affichée dans la partie droite de l'écran :

```
IF (MOUSEY>176) AND (MOUSEY<192) THEN
  LOCATE 12,59
  IF (MOUSEX>32) AND (MOUSEX<72) THEN
    ok=TRUE
    PRINT ''#''
    alt(index)=1
  ENDIF
  IF (MOUSEX>104) AND (MOUSEX<144) THEN
    ok=TRUE
    PRINT ''b''
    alt(index)=2
  ENDIF
  IF (MOUSEX>176) AND (MOUSEX<232) THEN
    ok=TRUE
    PRINT '' ''
    alt(index)=0
  ENDIF
ENDIF
```



Si le curseur de la souris se trouvait sur une des options de durée lors du clic, le choix est considéré valide, cette durée est mémorisée dans la variable **dur**. Par exemple pour une triple croche :

```
IF (MOUSEY>208) AND (MOUSEY < 224) THEN
  LOCATE 12,61
  IF (MOUSEX>32) AND (MOUSEX<48) THEN
    ok=TRUE
    PRINT "TC"
    dur(index)=1
  ENDIF
```

Si le curseur de la souris se trouvait sur l'option **VALIDE** lors du clic, le choix est considéré valide, l'espace d'affichage de la note ou de la pause courante est effacé, et le pointeur de note est incrémenté :

```
  IF (MOUSEX>416) AND (MOUSEX<464) THEN
    ok=TRUE
    LOCATE 12,56
    PRINT SPACE$(30)
    index=index+1
    LOCATE 12,45
    PRINT "Note ";index;" : "
  ENDIF
```

Si le curseur de la souris se trouvait sur l'option **EFFACE** lors du clic, le choix est considéré valide, et la zone d'affichage de la note ou de la pause est effacée :

```
  IF (MOUSEX>488) AND (MOUSEX<536) THEN
    ok=TRUE
    LOCATE 12,56
    PRINT SPACE$(30)
  ENDIF
```

Si le curseur de la souris se trouvait sur une durée de pause lors du clic, le choix est considéré valide et la durée est affichée en clair dans la partie droite de l'écran. Par exemple, pour une pause d'1/4 de temps :

```
IF (MOUSEY>240) AND (MOUSEY<256) THEN
  no(index)=0
  alt(index)=0
  dur(index)=0
  LOCATE 12,56
  PRINT "Pause : ";
  IF (MOUSEX>88) AND (MOUSEX<112) THEN
    ok=TRUE
    PRINT "1/4 temps"
    paus(index)=1
  ENDIF
```

Si le curseur de la souris se trouvait sur l'option **CHARGE** lors du clic :

```
IF (MOUSEY>304) AND (MOUSEY<320) THEN
  IF (MOUSEX>192) AND (MOUSEX<240) THEN
```

le choix est considéré valide, et le programme demande d'entrer le nom du fichier à charger :

```
ok=TRUE
CLS
INPUT "Entrez le nom du fichier : ";nf$
```

Ce fichier doit avoir été créé par le programme, sans quoi, les données qu'il contient risquent fort de ne pas correspondre au format attendu. Dans le cas où le fichier existe et où son format est correct, les données qu'il contient sont chargées dans les tableaux **no, alt, dur** et **paus** :

```
OPEN "I",#1,nf$
INPUT #1,index
sauv_index=index
FOR i=1 TO index
    INPUT #1,no(i)
    INPUT #1,alt(i)
    INPUT #1,dur(i)
    INPUT #1,paus(i)
NEXT i
CLOSE #1
```

Dans le cas où le curseur de la souris se trouvait sur l'option **SAUVE** lors du clic :

```
IF (MOUSEX>264) AND (MOUSEX<304) THEN
```

le choix est considéré valide, et le programme demande d'entrer un nom du fichier :

```
ok=TRUE
CLS
INPUT "Entrez le nom du fichier : ";nf$
```

Les données en mémoire sont sauvegardées dans ce fichier à partir des tableaux **no, alt, dur** et **paus** :

```
OPEN "O",#1,nf$
PRINT #1,index-1
FOR i=1 TO index-1
    PRINT #1,no(i)
    PRINT #1,alt(i)
    PRINT #1,dur(i)
    PRINT #1,paus(i)
NEXT i
CLAUSE #1
```

Enfin, si le curseur se trouvait sur l'option **AFFICHE** lors du clic :

```
IF (MOUSEX>328) AND (MOUSEX<384) THEN
```

le choix est considéré valide :

```
ok=TRUE
```



Les 24 premières notes sont affichées à l'aide d'une boucle FOR NEXT :

```
FOR z=1 TO MAX(index-1,sauv_index-1) STEP 24
```

La boucle débute par l'affichage d'une portée :

```
LINE 10,j*50,610,j*50
LINE 10,j*50+6,610,j*50+6
LINE 10,j*50+12,610,j*50+12
LINE 10,j*50+18,610,j*50+18
LINE 10,j*50+24,610,j*50+24
```

Chacune des 24 premières notes en mémoire est ensuite décodée et affichée sur la portée.

L'affichage commence par l'altération (bémol ou dièse) :

```
IF alt(k+(j-1)*24)=2 THEN
  TEXT k*25-14,no(k+(j-1)*24)*3+(j-1)*50+40,"b"
ENDIF
IF alt(k+(j-1)*24)=1 THEN
  TEXT k*25-14,no(k+(j-1)*24)*3+(j-1)*50+42,"#"
ENDIF
```

Il se poursuit par l'affichage de la note. Si nécessaire, la portée est prolongée vers le bas :

```
IF (k+(j-1)*24)<=MAX(index,sauv_index) THEN
  IF no(k+(j-1)*24)>14 THEN
    LINE k*25-5,(j-1)*50+80,k*25+5,(j-1)*50+80
    LINE k*25-5,(j-1)*50+86,k*25+5,(j-1)*50+86
  ENDIF
```

ou vers le haut :

```
IF no(k+(j-1)*24)<4 THEN
  LINE k*25-5,(j-1)*50+38,k*25+5,(j-1)*50+38
  LINE k*25-5,(j-1)*50+44,k*25+5,(j-1)*50+44
ENDIF
```

La note est représentée par un cercle vide si sa durée est égale à une blanche ou à une ronde :

```
IF dur(k+(j-1)*24)>5 THEN
  CIRCLE k*25,no(k+(j-1)*24)*3+(j-1)*50+35,3
```

La note est représentée par un cercle plein dans tous les autres cas :

```
ELSE
  PCIRCLE k*25,no(k+(j-1)*24)*3+(j-1)*50+35,3
ENDIF
```

Selon sa durée, la note est complétée d'un ou de plusieurs traits

```
IF (no(k+(j-1)*24)<10) AND (dur(k+(j-1)*24)<>6)
THEN
'''
  ELSE
    LINE k*25-3,a+14,k*25+1,a+10
```

```
      ENDIF
    ENDIF
```

Dans le cas où la donnée courante représente une pause, un signe particulier est affiché en fonction de sa durée. Par exemple pour une durée d'1/4 de temps :

```
    SELECT paus(k+(j-1)*24)
    CASE 1
      LINE k*25,j*50+6,k*25-8,j*50+20
      LINE k*25,j*50+6,k*25-8,j*50+4
      LINE k*25-2,j*50+9,k*25-10,j*50+7
```

Lorsque toutes les notes en mémoire ont été représentées, le programme se met en attente de l'appui sur une des touches du clavier :

```
  REPEAT
  UNTIL INKEY$<>""
```

Il rend alors le contrôle à la boucle principale :

```
  GOTO boucle_principale
```

Si le choix de l'utilisateur n'est pas considéré valide (OK=FALSE), un bip sonore est émis :

```
IF NOT ok THEN !Aucune option valide n'a été sélectionnée
  SOUND 1,15,10,4,5
  SOUND 1,0,0,0,0
END
```

Le programme prend fin lorsque la variable fin vaut TRUE, c'est-à-dire lorsque l'option FIN a été sélectionnée :

```
UNTIL fin
END
```

6/1.3.10

Reproduction d'échantillons sur le ST

Le programme que nous vous présentons est découpé en trois parties.

La première partie a trait à la présentation et la réalisation de deux échantillons de synthèse classiques. Ce sont deux formes d'onde qui sont générées, l'une de fréquence fixe (hauteur constante à l'audition) et d'amplitude variable qui sera répertoriée sous le signe **am** (pour *modulation d'amplitude*) et l'autre d'amplitude constante et de fréquence variable qui sera répertoriée sous le sigle **fm** (pour *modulation de fréquence*).

Après la présentation graphique générale des deux formes d'onde, les tableaux de 6 000 points sont remplis et sauvegardés sur disquette (malgré la rapidité proverbiale du GFA, cela devrait prendre tout de même quelques 30 secondes).

La seconde partie est l'écriture du programme de reproduction d'échantillon en assembleur. Ce programme est prévu pour être appelé par le Basic GFA, donc toutes les adresses sont relatives par rapport à A6. Le listing désassemblé vous est aussi fourni si vous ne disposez pas d'assembleur. Le programme de conversion fonctionne directement avec les convertisseurs de la puce son, donc en 4 bits, ce qui est assurément loin de la qualité CD, mais permet de bien comprendre le principe de la conversion. Ce programme fontionne aussi sur le STE, mais étant donné que ce dernier possède un convertisseur, il est préférable de sortir directement l'échantillon par cet intermédiaire. Pour ne garder que les 4 bits, il suffit de réaliser quatre rotations à droite sur l'octet échantillon et placer le résultat dans les registres 8, 9 et 10 pour avoir le plus « gros » son possible sur le moniteur.

Les paramètres d'appel du programme en GFA sont dans l'ordre :

- l'adresse de l'échantillon (mot long) ;
- la longueur de l'échantillon (mot long) ;
- le nombre de répétitions de l'échantillon (mot) ;
- la vitesse de l'échantillon (mot).

C'est le rôle des premières instructions du programme que de dépiler ces paramètres.

La suite logique des opérations est :

- passer en mode superviseur ;
- interrompre les transmissions IKBD ;
- désactiver les interruptions ;
- convertir et jouer l'échantillon ;
- réactiver les interruptions ;
- connecter l'IKBD ;
- passer en mode utilisateur.

La troisième partie ne fait qu'appeler la routine.

Quelques remarques :

L'échantillon est joué du début à la fin, mais vous pouvez très bien indiquer des adresses qui sont comprises entre les **adr%** et **adrfin%** réelles des fichiers si vous ne voulez écouter qu'une portion de l'échantillon.

La longueur de l'échantillon n'est pratiquement pas limitée, mais pensez à tester la mémoire disponible (**MALLOC**) si vous voulez faire jouer un échantillon de 500 K.

La vitesse devrait théoriquement correspondre à la vitesse d'échantillonnage à l'enregistrement ; ce paramètre vous permet de ralentir la vitesse des échantillons du programme GFA. Une valeur supérieure à 150 vous prolongera le son et amènera quelques fréquences hautement indésirables à se superposer aux belles formes d'onde d'origine.



Programme version Basic GFA :

```
' VERSION BASIC GFA avec programme assembleur en INLINE
DIM tab|(6000),tabam|(640),tabfm|(640)
' Les tableaux tabam| et tabfm| ne servent qu'à visualiser
' les courbes de modulation d'amplitude et de fréquence
' les paramètres de modulatiuon sont réduits pour permettre
' la visualisation de toute la courbe
FOR i&=0 TO 640
  tabam|(i&)=127+50*COS(i&/5)*SIN(i&/200)
  tabfm|(i&)=127+50*COS(i&/5+100*SIN(i&/100))
NEXT i&
TEXT 260,40,"MODULATION D'AMPLITUDE"
TEXT 260,240,"MODULATION DE FREQUENCE"
FOR i&=0 TO 639
  DRAW i&,tabam|(i&) TO i&+1,tabam|(i&+1)
  DRAW i&,200+tabfm|(i&) TO i&+1,200+tabfm|(i&+1)
NEXT i&
~INP(2)
CLS
'Lorsque vous aurez remplis les deux fichiers sur disquette
' il faudra supprimer le remplissage des tableaux ci-dessous
adr%=V:tab|(0)
adrfin%=adr%+6000
FOR i&=0 TO 6000
  tab|(i&)=127+127*COS(i&/10)*SIN(i&/2000)
NEXT i&
BSAVE "mod_amp",adr%,6000
FOR i&=0 TO 6000
  tab|(i&)=127+127*COS(i&/10+100*SIN(i&/1000))
NEXT i&
BSAVE "mod_fre",adr%,6000
~INP(2)
' Le programme utilitaire commencera ici en version définitive
' n'oubliez pas de conserver le tableau tab| adr% et adrfin%
BLOAD "mod_amp",adr%
INLINE echantillon%,&H150
~C:echantillon%(L:adr%,L:adrfin%,W:4,W:100)
~INP(2)
BLOAD "mod_fre",adr%
~C:echantillon%(L:adr%,L:adrfin%,W:4,W:100)
~INP(2)
```

Programme version assembleur :

```
;programme assembleur échantillon sur puce son

        .TEXT
; on récupère les paramètres d'appel sur la pile

        lea.l   VARIABLE(pc),a6
        move.l  4(sp),DEB_ECH(a6)
        move.l  8(sp),FIN_ECH(a6)
        move.w  12(sp),NBOUCLES(a6)
        move.w  14(sp),VITESSE(a6)

;mode superviseur
        clr.l   -(sp)
        move.w  #$20,-(sp)
        trap    #1
        addq.l  #$6,sp
        move.l  d0,PILESUP(a6)

;déconnecter le clavier

        move.b  #$13,CLAV(a6)
        move.l  a6,-(sp)
        move.w  #$0,-(sp)
        move.w  #25,-(sp)
        trap    #14
        addq.l  #$8,sp

;masquer les interruptions

        move.w  sr,REGETAT(a6)
        ori.w   #$700,sr

;initialiser la puce son
; charger adresse de la table des paramètres
        lea.l   VALDEP(pc),a0
INIT_BOUCLE:
        move.w  (a0)+,d0
        bmi     INIT_FIN
        move.b  d0,$ff8800
        move.w  (a0)+,d0
        move.b  d0,$ff8802
        bra     INIT_BOUCLE
INIT_FIN:
        move.b  #$7,$ff8800
        move.b  $ff8800,d0
        andi.b  #$c0,d0
        move.w  (a0)+,d1
        or.b    d1,d0
        move.b  d0,$ff8802
```

(suite)

```
;PROGRAMME DE CONVERSION
                move.w      NBOUCLES(a6),d4
                ori.w       #$1,d4
DEBUT:
                movea.l     DEB_ECH(a6),a2
                movea.l     FIN_ECH(a6),a3
JOUE:
                move.w      VITESSE(a6),d6
ATTENTE:
                dbf         d6,ATTENTE
                move.b      (a2)+,d5
                andi.w      #$fc,d5
                lsr.w       #4,d5
; on fait jouer les trois registres
                move.b      #$8,$ff8800
                move.b      d5,$ff8802
                move.b      #$9,$ff8800
                move.b      d5,$ff8802
                move.b      #$a,$ff8800
                move.b      d5,$ff8802
                cmpa.l      a2,a3
                bgt         JOUE

FIN:
                dbf         d4,DEBUT
;re activer les interruptions

                move.w      REGETAT(a6),sr

;reconnecter le clavier

                move.b      #$11,CLAV(a6)
                move.l      a6,-(sp)
                move.w      #$0,-(sp)
                move.w      #25,-(sp)
                trap        #14
                addq.l      #$8,sp

;repasser en mode utilisateur

                move.l      PILESUP(a6),-(sp)
                move.w      #$20,-(sp)
                trap        #1
                addq.l      #6,sp

                rts
```

(suite)

```
;valeurs de variables et réservations (par raport à A6)

VALDEP: .DC.w 0,255,1,255,2,255,3,255,4,255,5,255,6,0,-1
        .DC.w $3f
VARIABLE:
CLAV:   .DC.w 0
PILESUP: .DC.l 0
REGETAT:.DC.w 0
DEB_ECH:.DC.l 0
FIN_ECH: .DC.l 0
NBOUCLES:
        .DC.w 0
VITESSE: .DC.w 0
        .END
```

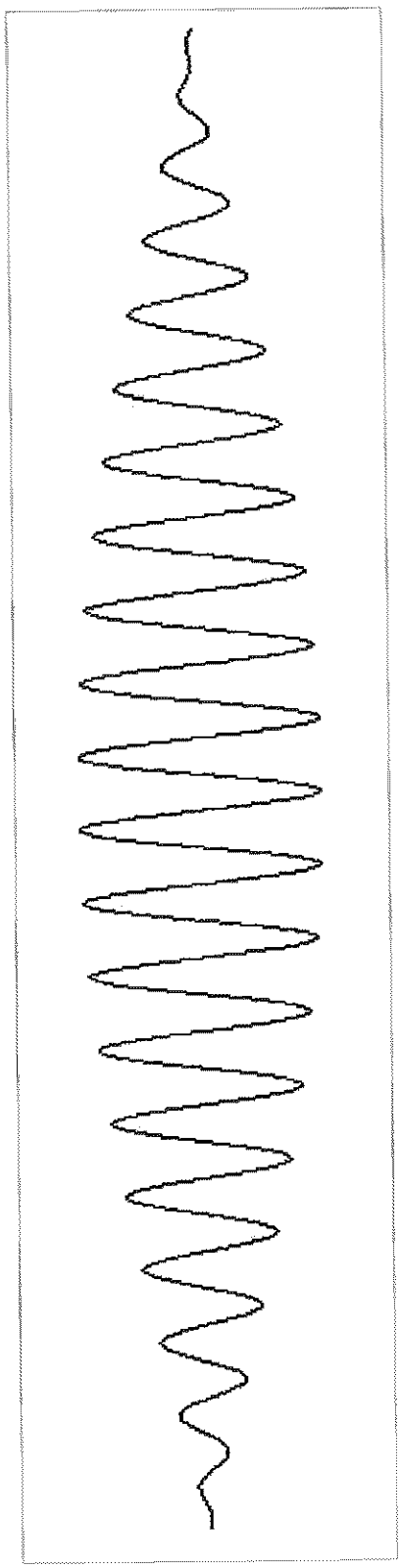


Listing pour chargeur :

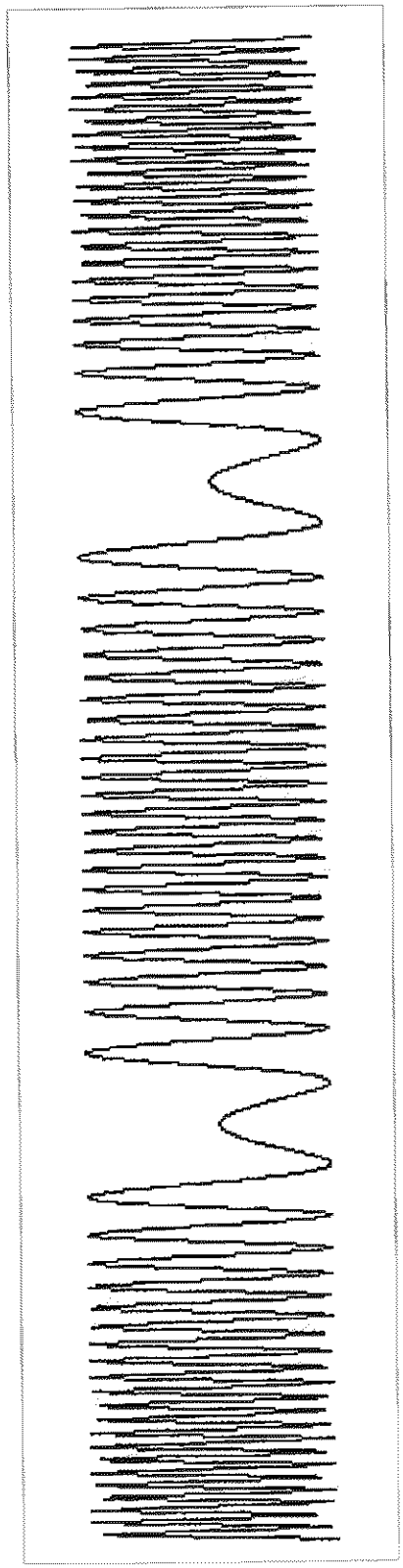
```
$43FAA:[ 4dfa 0112                ]        lea.l    $440be(pc),a6
$43FAE:[ 2d6f 0004 011c           ]        move.l   $4(a7),$11c(a6)
$43FB4:[ 2d6f 0008 0120           ]        move.l   $8(a7),$120(a6)
$43FBA:[ 3d6f 000c 0124           ]        move.w   $c(a7),$124(a6)
$43FC0:[ 3d6f 000e 0126           ]        move.w   $e(a7),$126(a6)
$43FC6:[ 42a7                     ]        clr.l    -(a7)
$43FC8:[ 3f3c 0020                ]        move.w   #$20,-(a7)
$43FCC:[ 4e41                     ]        trap     #$1
$43FCE:[ 5c8f                     ]        addq.l   #$6,a7
$43FD0:[ 2d40 0116                ]        move.l   d0,$116(a6)
$43FD4:[ 1d7c 0013 0114           ]        move.b   #$13,$114(a6)
$43FDA:[ 2f0e                     ]        move.l   a6,-(a7)
$43FDC:[ 3f3c 0000                ]        move.w   #$0,-(a7)
$43FE0:[ 3f3c 0019                ]        move.w   #$19,-(a7)
$43FE4:[ 4e4e                     ]        trap     #$e
$43FE6:[ 508f                     ]        addq.l   #$8,a7
$43FE8:[ 40ee 011a                ]        move.w   sr,$11a(a6)
$43FEC:[ 007c 0700                ]        ori.w    #$700,sr
$43FF0:[ 41fa 00ac                ]        lea.l    $4409e(pc),a0
$43FF4:[ 3018                     ]        move.w   (a0)+,d0
$43FF6:[ 6b00 0012                ]        bmi      $4400a
$43FFA:[ 13c0 00ff8800            ]        move.b   d0,$ff8800
$44000:[ 3018                     ]        move.w   (a0)+,d0
$44002:[ 13c0 00ff8802            ]        move.b   d0,$ff8802
$44008:[ 60ea                     ]        bra.s    $43ff4
$4400A:[ 13fc 0007 00ff8800       ]        move.b   #$7,$ff8800
$44012:[ 1039 00ff8800            ]        move.b   $ff8800,d0
$44018:[ 0200 00c0                ]        andi.b   #$c0,d0
$4401C:[ 3218                     ]        move.w   (a0)+,d1
$4401E:[ 8001                     ]        or.b     d1,d0
$44020:[ 13c0 00ff8802            ]        move.b   d0,$ff8802
$44026:[ 382e 0124                ]        move.w   $124(a6),d4
$4402A:[ 0044 0001                ]        ori.w    #$1,d4
$4402E:[ 246e 011c                ]        movea.l  $11c(a6),a2
$44032:[ 266e 0120                ]        movea.l  $120(a6),a3
$44036:[ 3c2e 0126                ]        move.w   $126(a6),d6
$4403A:[ 51ce fffe                ]        dbra     d6,$4403a
$4403E:[ 1a1a                     ]        move.b   (a2)+,d5
$44040:[ 0245 00fc                ]        andi.w   #$fc,d5
$44044:[ e84d                     ]        lsr.w    #$4,d5
$44046:[ 13fc 0008 00ff8800       ]        move.b   #$8,$ff8800
$4404E:[ 13c5 00ff8802            ]        move.b   d5,$ff8802
$44054:[ 13fc 0009 00ff8800       ]        move.b   #$9,$ff8800
$4405C:[ 13c5 00ff8802            ]        move.b   d5,$ff8802
$44062:[ 13fc 000a 00ff8800       ]        move.b   #$a,$ff8800
$4406A:[ 13c5 00ff8802            ]        move.b   d5,$ff8802
$44070:[ b7ca                     ]        cmpa.l   a2,a3
$44072:[ 6ec2                     ]        bgt.s    $44036
$44074:[ 51cc ffb8                ]        dbra     d4,$4402e
$44078:[ 46ee 011a                ]        move.w   $11a(a6),sr
$4407C:[ 1d7c 0011 0114           ]        move.b   #$11,$114(a6)
$44082:[ 2f0e                     ]        move.l   a6,-(a7)
```

(suite)

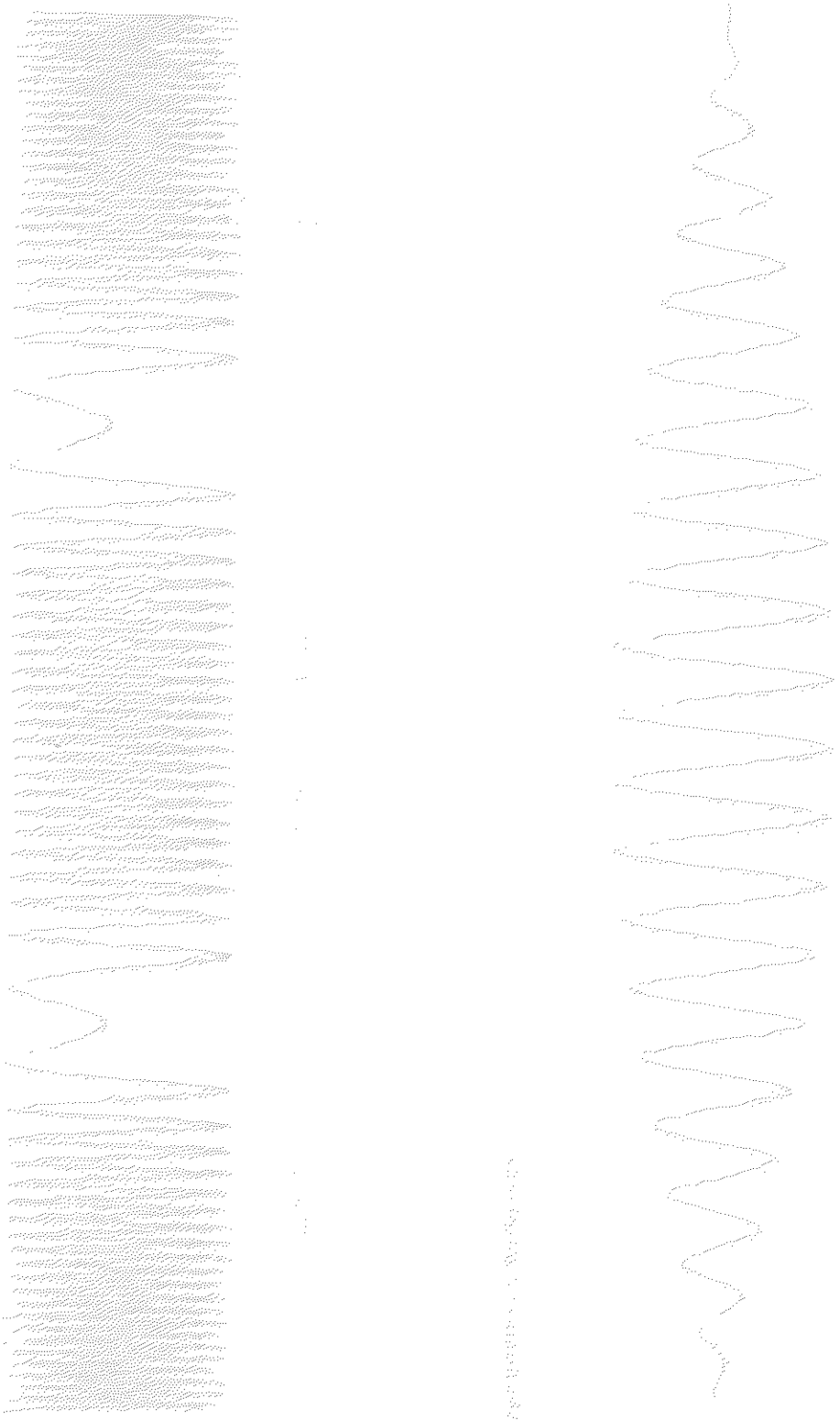
```
$44084:[ 3f3c 0000          ]        move.w    #$0,-(a7)
$44088:[ 3f3c 0019          ]        move.w    #$19,-(a7)
$4408C:[ 4e4e               ]        trap      #$e
$4408E:[ 508f               ]        addq.l    #$8,a7
$44090:[ 2f2e 0116          ]        move.l    $116(a6),-(a7)
$44094:[ 3f3c 0020          ]        move.w    #$20,-(a7)
$44098:[ 4e41               ]        trap      #$1
$4409A:[ 5c8f               ]        addq.l    #$6,a7
$4409C:[ 4e75               ]        rts
$4409E:[ 0000 00ff          ]        ori.b     #$ff,d0
$440A2:[ 0001 00ff          ]        ori.b     #$ff,d1
$440A6:[ 0002 00ff          ]        ori.b     #$ff,d2
$440AA:[ 0003 00ff          ]        ori.b     #$ff,d3
$440AE:[ 0004 00ff          ]        ori.b     #$ff,d4
$440B2:[ 0005 00ff          ]        ori.b     #$ff,d5
$440B6:[ 0006 0000          ]        ori.b     #$0,d6
$440BA:[ ffff               ]
$440BC:[ 003f               ]
$440BE:[ 0000 0000          ]        ori.b     #$0,d0
$440C2:[ 0000 0000          ]        ori.b     #$0,d0
$440C6:[ 0000 0000          ]        ori.b     #$0,d0
$440CA:[ 0000 0000          ]        ori.b     #$0,d0
$440CE:[ 0000 0000          ]        ori.b     #$0,d0
```



Modulation d'amplitude.



Modulation de fréquence.



6/1.3.11

Ecran sonore

INTRODUCTION

Le but de ce programme est de définir graphiquement une zone sur laquelle un clic de souris pourra jouer des notes qui suivront une succession d'intervalles définie comme l'un des paramètres d'appel. On peut ainsi définir trois notes sur tout l'écran ou un grand nombre de notes sur une portion réduite d'écran, en ayant le choix de dessiner ou pas les barres horizontales des zones correspondant aux intervalles sonores.

La sortie de son se fait par l'intermédiaire du générateur sonore mais le codage des hauteurs de notes est immédiatement accessible, si vous le désirez, par le système MIDI. Il suffit d'introduire l'offset de la première note référencée dans le tableau, C1 de code MIDI 36 et d'indice de tableau 0, pour récupérer le code de hauteur MIDI.

Après l'initialisation du générateur sonore, du tableau des noms et des hauteurs de notes et des gammes ou suites d'intervalles utilisés, le programme entre dans sa boucle principale qui teste les modifications éventuelles de valeurs de paramètres et de leur validation ainsi que les clics de souris à l'intérieur de la zone son. Cette dernière opération est le fait de la procédure **zoneson**.

LE PROGRAMME EN DETAILS

L'initialisation

C'est le groupe d'instructions qui s'étend du début du programme au **DO LOOP**. Les tableaux **param$()** et **param&()** contiennent respectivement les appellations et les valeurs des paramètres de la zone à définir.

Il vous sera facile de transformer ce programme en procédure pour l'inclure dans un autre programme en récupérant les sept paramètres d'appel. Chaque paramètre possède une valeur minimale et une valeur maximale qui sont respectivement codées dans les tableaux **minparam&()** et **maxparam&()**. Il y a une exception à cette règle, c'est celle

des gammes puisque vous pouvez très bien décider d'avoir trois choix de gammes à trois sons et cinq choix de gammes à huit sons. C'est pourquoi il faut noter dans deux tableaux supplémentaires ces nombres de choix minimums et maximums relatifs aux gammes dans les tableaux **gamminl()** et **gammaxl()**. Si vous désirez changer les data d'origine il faut prendre garde à bien respecter les valeurs que vous notez en **gammaxl()** au risque de décaler toutes vos séquences de gammes. La suite des intervalles des gammes est codée dans le tableau **gamme$()**, le premier indice étant le nombre d'intervalles à un offset d'indice de tableau près et le second indice la référence de la gamme parmi celles comprenant ce même nombre d'intervalles.

Le tableau **tonique$()** code les appellations des notes jouées, le premier indice correspondant au choix des dièses ou des bémols dans l'affichage.

Le tableau **registrel()** permet de coder sur deux octets les valeurs à envoyer dans les registres de hauteur du générateur sonore. On peut ainsi faire jouer sur le canal voulu, la note dont la hauteur correspond à l'indice du tableau (C1 correspond à l'indice 0).

Les tests de paramètres

Ces tests sont tous compris dans la boucle **DO LOOP**. On incrémente la valeur de **param&()** par le clic gauche et on décrémente par le clic droit. Chaque paramètre doit rester majoré par sa valeur maximum comprise dans **maxparam&()** et minoré par sa valeur minimum comprise dans **minparam&()**. La seule exception est évidemment celle des gammes qui correspondent à **param&(6)** et qu'il faut donc réinitialiser par les valeurs de **gamminl()** et **gammaxl()**. Ce problème n'interviendrait pas si vous aviez décidé de distribuer le même nombre de gammes quel que soit le nombre d'intervalles. Dans la version actuelle du programme, il faut cliquer sur **valider** (appel de la procédure **valider**) pour répercuter les nouvelles valeurs des paramètres affichés sur l'image et le son. Vous pouvez en décider autrement en incluant cette procédure **valider** après le traitement de toute modification de paramètre.

La procédure zoneson

Cette procédure est appelée dès qu'on clique dans les zones d'écran qu'on veut sonoriser. Le nombre de zones est contenu dans **param&(4)**. Il suffit de connaître l'indice de la zone dans laquelle on a cliqué pour retrouver l'intervalle correspondant dans le tableau **gamme$()**. On récupère le code de la note en corrigeant par l'offset de la tonique et le code de C1 (**tonil** et **tonioffsetl**). L'affichage du nom de la note se fait par défaut mais il est simple d'introduire un flag logique pour permettre ou interdire cette éventualité.



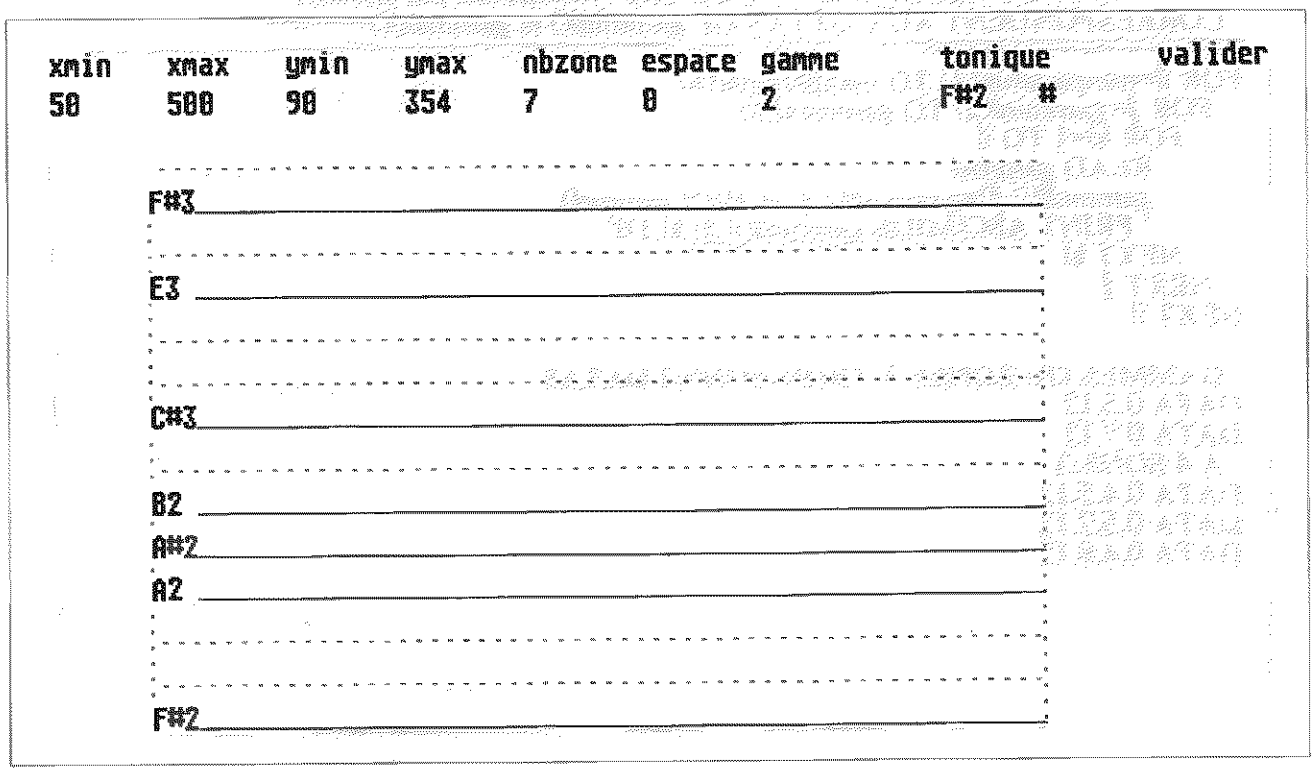
Les procédures

Procedure ***valider***

Cette procédure réalise le nouvel affichage au moyen des procédures **cadreblanc** qui efface l'ancien, **tracecadre** qui affiche le nouveau cadre extérieur et **tracezone** qui délimite toutes les zones différentes d'intervalles. Ces deux dernières procédures pourraient aussi n'effectuer leur affichage qu'avec un positionnement de flags logiques sur la valeur TRUE.

Procédure ***jf_hauteur***

Cette procédure est utilisée dès que l'on veut faire jouer une note de hauteur et volume connus sur l'un des trois canaux sonores du générateur. Le tableau **registrel()** est exploré suivant son indice et la hauteur correspondante est envoyée sur les registres hauteurs du générateur sonore. Pour des raisons de rapidité d'exécution, vous pouvez tout à fait supprimer la boucle itérative FOR NEXT et recopier les deux lignes de programme qui écrivent dans les registres du générateur sonore.



Page représentative de l'écran sonore.

```
' Découpage de l'écran en espaces sonores sur des choix de gammes
'
DIM param$(6),param&(6),minparam&(6),maxparam&(6),gammin|(12),gammax|(12)
offx|=60
tonioffset|=36
zonemax|=12                    ! nombre de zones par défaut on peut le changer
suitemax|=20
DIM ordonnee&(13),gamme$(zonemax|,suitemax|)
DIM tonique$(1,48),alte$(1)
FOR i|=0 TO 6
  READ param$(i|),minparam&(i|),maxparam&(i|)
  param&(i|)=minparam&(i|)
NEXT i|
DATA xmin,50,200,xmax,400,630,ymin,55,200,ymax,250,350,nbzone,3,12,espace,0,0
DATA gamme,0,0
FOR i|=minparam&(4) TO maxparam&(4)
  READ  gammin|(i|),gammax|(i|)          !nbre min et max de gammes par zone
NEXT i|
DATA 1,2,1,3,1,4,1,2,1,2,1,1,1,3,1,2,1,2,1,1
' MAINTENANT CE SONT LES DATAS DES GAMMES OU SUITES D'INTERVALLES
' ---------ATTENTION!!!!!!!!
' --------IL FAUT ABSOLUMENT QUE LES NOMBRES DE CHOIX --------
' CORRESPONDENT AUX DATAS DE gammin|() et gammax|()---------
' ------------------------
FOR i|=minparam&(4) TO maxparam&(4)
  FOR j|=gammin|(i|) TO gammax|(i|)
    FOR k|=1 TO i|
      READ gamme|
      gamme$(i|,j|)=gamme$(i|,j|)+CHR$(gamme|)
      ' PRINT ASC(MID$(gamme$(i|,j|),k|,1))'
    NEXT k|
  NEXT j|
NEXT i|
'
' GAMMES OU ZONES À TROIS SONS:2 DATAS
DATA 0,5,12
DATA 0,7,12
' A 4 SONS:3
DATA 0,4,5,12
DATA 0,5,7,12
DATA 0,4,8,12
```

(suite)

```
' A 5 SONS:4
DATA 0,4,5,7,12
DATA 0,3,6,9,12
DATA 0,2,5,7,12
DATA 0,3,5,7,12
' A 6 SONS :2
DATA 0,2,5,6,8,12
DATA 0,3,5,8,9,12
' A 7 SONS: 2
DATA 0,2,3,5,7,8,12
DATA 0,3,4,5,7,10,12
' A8 SONS 1
DATA 0,2,4,6,7,8,10,12
' A9 SONS 3
DATA 0,2,4,5,7,8,9,10,12
DATA 0,1,3,5,6,8,9,11,12
DATA 0,2,3,5,6,8,9,10,12
' A 10 SONS2
DATA 0,1,3,4,5,6,8,9,11,12
DATA 0,2,3,4,5,6,7,10,11,12
' A11 SONS 2
DATA 0,1,2,3,5,6,7,8,9,10,12
DATA 0,2,3,4,5,6,8,9,10,11,12
' A 12 SONS 1
DATA 0,1,2,3,4,5,6,8,9,10,11,12
'  ----------ON PEUT ICI----------
' METTRE UN TABLEAU À DEUX DIMENSIONS POUR GÉRER
' LES BÉMOLS ET LES DIESES
' SUIVANT LA POSITION D'UN FLAG LOGIQUE----------
'
FOR j|=0 TO 1
   FOR i|=0 TO 48
      READ tonique$(j|,i|)
   NEXT i|
NEXT j|
DATA C1,C#1,D1,D#1,E1,F1,F#1,G1,G#1,A1,A#1,B1
DATA C2,C#2,D2,D#2,E2,F2,F#2,G2,G#2,A2,A#2,B2
DATA C3,C#3,D3,D#3,E3,F3,F#3,G3,G#3,A3,A#3,B3
DATA C4,C#4,D4,D#4,E4,F4,F#4,G4,G#4,A4,A#4,B4
DATA C5
DATA C1,Db1,D1,Eb1,E1,F1,Gb1,G1,Ab1,A1,Bb1,B1
DATA C2,Db2,D2,Eb2,E2,F2,Gb2,G2,Ab2,A2,Bb2,B2
DATA C3,Db3,D3,Eb3,E3,F3,Gb3,G3,Ab3,A3,Bb3,B3
DATA C4,Db4,D4,Eb4,E4,F4,Gb4,G4,Ab4,A4,Bb4,B4
DATA C5
```

(suite)

```
' ----------registres son----------
'
' valeur des registres
DIM registre|(97)      !4 octaves * 12 demi-tons + 1
'  poids fort à droite                  msb
'  poids faible à gauche                lsb
FOR octave&=-2 TO 2
  FOR note|=1 TO 12
    EXIT IF octave&=2 AND note|=2
    freq=440*(2^(octave&+((note|-10)/12)))
    per&=INT(125000/freq)
    msb|=INT(per&/256)
    lsb|=per&-msb|*256
    k|=(octave&+2)*12+note|-1
    registre|(2*k|)=lsb|
    registre|(2*k|+1)=msb|
  NEXT note|
  EXIT IF octave&=2 AND note|=2
NEXT octave&
alte$(0)="#"
alte$(1)="b"
GET minparam&(0),minparam&(2)-10,maxparam&(1),maxparam&(3)+10,cadreblanc$
param&(1)=500
param&(2)=90
param&(3)=354
FOR i|=0 TO 5
  TEXT offx|*i|,20,param$(i|)
  TEXT offx|*i|,40,param&(i|)
NEXT i|
param&(6)=gammin|(param&(4))
TEXT offx|*6,20,param$(6)
TEXT offx|*6,40,param&(6)
TEXT 560,20,"valider"
TEXT 450,20,"tonique"
toni|=24
TEXT 450,40,tonique$(alte|,toni|)
TEXT 500,40,alte$(alte|)
DO
  MOUSE x&,y&,k|
  IF k|=1
    FOR i|=0 TO 6
```

(suite)

```
      IF x&>offxl*il AND x&<offxl*il+40 AND y&>20 AND y&<40
        INC param&(il)
        IF il<6
          param&(il)=MIN(param&(il),maxparam&(il))
          IF il=4
            param&(6)=gammin(param&(4))
            TEXT  offxl*6,40,"        "
            TEXT offxl*6,40,param&(6)
          ENDIF
        ELSE
          param&(il)=MIN(param&(il),gammax(param&(4)))
        ENDIF
        TEXT  offxl*il,40,"        "
        TEXT offxl*il,40,param&(il)
        PAUSE 3
      ENDIF
    NEXT il
    IF x&>560 AND x&<600 AND y&<20
      @valider
    ENDIF
    IF x&>450 AND x&<480 AND y&>20 AND y&<40
      INC tonil
      tonil=MIN(tonil,36)
      TEXT 450,40,"    "
      TEXT 450,40,tonique$(altel,tonil)
      PAUSE 3
    ENDIF
    IF x&>500 AND x&<520 AND y&>20 AND y&<40
      altel=1
      TEXT 450,40,"    "
      TEXT 450,40,tonique$(altel,tonil)
      TEXT 500,40,"    "
      TEXT 500,40,alte$(altel)
      PAUSE 3
    ENDIF
    @zoneson
  ENDIF
  IF kl=2
    FOR il=0 TO 6
      IF x&>offxl*il AND x&<offxl*il+40 AND y&>20 AND y&<40
        DEC param&(il)
        IF il<6
          param&(il)=MAX(param&(il),minparam&(il))
```

(suite)

```
            IF il=4
              param&(6)=gammin(param&(4))
              TEXT offxl*6,40,"      "
              TEXT offxl*6,40,param&(6)
            ENDIF
          ELSE
            param&(il)=MAX(param&(il),gammin(param&(4)))
          ENDIF
          TEXT offxl*il,40,"      "
          TEXT offxl*il,40,param&(il)
          PAUSE 3
        ENDIF
      NEXT il
      IF x&>450 AND x&<480 AND y&>20 AND y&<40
        tonil=MAX(tonil,1)
        DEC tonil
        TEXT 450,40,"   "
        TEXT 450,40,tonique$(altel,tonil)
        PAUSE 3
      ENDIF
      IF x&>500 AND x&<520 AND y&>20 AND y&<40
        altel=0
        TEXT 450,40,"   "
        TEXT 450,40,tonique$(altel,tonil)
        TEXT 500,40,"   "
        TEXT 500,40,alte$(altel)
        PAUSE 3
      ENDIF
    ENDIF
    EXIT IF kl=3
  LOOP
  PROCEDURE zoneson
    ' nb est le nombre de zones sons
    ' gaml est la référence de la suite d'intervalles à partir de la tonique
    ' tonl est le code de la tonique tonl=60 équivaut à C3
    ' espacel est la référence de la spacialisation des zones (en escalier,
    ' alignées à gauche ou à droite etc.....
    SELECT param&(5)
    CASE 0
      FOR il=1 TO param&(4)-1
        IFx&>param&(0)ANDx&<param&(1)ANDy&>ordonnee&(il+1)ANDy&<ordonnee&(
il)
```

(suite)

```
            ' il EST LE NUMÉRO DE LA ZONE DANS LAQUELLE ON A CLIQUE
            pitch|=tonioffset|+toni|+ASC(MID$(gamme$(param&(4),param&(6)),il,1))
            TEXT  param&(0),ordonnee&(il)," "
            TEXT param&(0),ordonnee&(il),tonique$(alte|,pitch|-tonioffset|)
            jf_hauteur(1,pitch|-tonioffset|,16)
          ENDIF
        NEXT il
        IFx&>param&(0)ANDx&<param&(1)ANDy&>ordonnee&(il)-hauteur&ANDy&<ordon
nee&(il)
          ' il EST LE NUMÉRO DE LA ZONE DANS LAQUELLE ON A CLIQUE
          pitch|=tonioffset|+toni|+ASC(MID$(gamme$(param&(4),param&(6)),il,1))
          TEXT  param&(0),ordonnee&(il)," "
          TEXT param&(0),ordonnee&(il),tonique$(alte|,pitch|-tonioffset|)
          jf_hauteur(1,pitch|-tonioffset|,16)
        ENDIF
      CASE 1
        ' Ici on peut intoduire un découpage différent en X
    ENDSELECT
  RETURN
  PROCEDURE valider
    valide!=TRUE
    @cadreblanc
    @tracecadre
    @tracezone
  RETURN
  PROCEDURE cadreblanc
    PUT minparam&(0),minparam&(2)-10,cadreblanc$
  RETURN
  PROCEDURE tracecadre
    DEFLINE 3,1,0,0
    BOX param&(0),param&(2),param&(1),param&(3)
    SELECT param&(5)
    CASE 0
      hauteur&=(param&(3)-param&(2)) DIV 12
      ordonnee&=param&(3)
      FOR il=1 TO 13
        ordonnee&=param&(3)-hauteur&*il
        LINE param&(0),ordonnee&,param&(1),ordonnee&
      NEXT il
    CASE 1
    ENDSELECT
  RETURN
```

(suite)

```
PROCEDURE tracezone
  ' IL FAUT TRACER EN FONCTION DE LA SUITE D'INTERVALLES
  ' AUTANT DE ORDONNEE&() QUE DE DEMI-TONS
  DEFLINE 1,1,0,0
  SELECT param&(5)
  CASE 0
    hauteur&=(param&(3)-param&(2)) DIV 12
    ordonnee&(0)=param&(3)
    FOR i|=1 TO param&(4)
      ordonnee&(i|)=param&(3)-hauteur&*ASC(MID$(gamme$(param&(4),param&(6)),i|,
1))
      LINE param&(0),ordonnee&(i|),param&(1),ordonnee&(i|)
    NEXT i|
  CASE 1
  ENDSELECT
RETURN
' PROCEDURE alert
' ALERT 3,"vous avez peut| etre oublié| de cliquer sur|VALIDER",1,"OUI JOHN",pl
usjamais&
' RETURN
PROCEDURE jf_hauteur(canal_pucel,note1|,volume1|)
  SPOKE &HFF8800,7
  SPOKE &HFF8802,BCLR(&HFF,canal_pucel-1)
  SPOKE &HFF8800,13
  SPOKE &HFF8802,1
  SPOKE &HFF8800,11
  SPOKE &HFF8802,20
  SPOKE &HFF8800,12
  ' ici la valeur qui suit peut réduire
  ' ou augmenter considérablement le release
  SPOKE &HFF8802,80
  FOR k1|=0 TO 1
    SPOKE &HFF8800,2*(canal_pucel-1)+k1|
    SPOKE &HFF8802,registrel(2*note1|+k1|)
  NEXT k1|
  SPOKE &HFF8800,8+canal_pucel-1          !vol
  SPOKE &HFF8802,volume1|
RETURN
PROCEDURE jf_silence
  SPOKE &HFF8800,7
  SPOKE &HFF8802,&HFF
RETURN
```

6/1.3.12

Séquenceur rythmique

Le programme en général

Ce programme se présente sous la forme d'une procédure assez générale que vous pourrez facilement inclure dans un autre programme et qui peut jouer sur le générateur sonore une séquence rythmique éditable au moyen d'une grille comparable dans le principe aux **grid edit** de multiples séquenceurs. Les possesseurs d'instrument MIDI n'auront que peu de choses à rajouter puisque les notes sont codées de façon à pouvoir passer facilement du code hauteur au code MIDI. Notez aussi qu'il est facile de passer d'une notion de séquenceur deux pistes (une pour jouer et une pour le métronome) à un séquenceur huit pistes ou plus, bien que le Basic GFA posera sans compilation quelques problèmes de mise en place. Le programme se sépare en deux parties, la première traitant de l'affichage **(jf_testrythme)** et la seconde **(jf_playrythme)** de la fonction jouer.

Le programme en détail

L'initialisation

Les tableaux définis **joue%()** et **jouemetro%()** sont la base du séquenceur final et contiendront les codes des notes à jouer. Par une simplification volontaire, les notes seront codées sur 4 octets, deux octets pour le temps, en supposant donc une musique tenant sur une page écran, un octet pour la hauteur et un octet pour le volume (ou la vélocité dans le cas du MIDI). Sur un morceau durant plus longtemps, il suffit de coder le temps sur 4 octets et reprendre le même principe de fonctionnement.

Les procédures qui suivent initialisent les valeurs du tableau des hauteurs de notes et les registres du générateur sonore.

*Procédure **jf_registre** :*

Cette procédure initialise les valeurs du tableau **registrel()** des codes de hauteurs sur quatre octaves. Les codes vont de 0 à 48 de la note C1 à C4. Pour le MIDI il suffit de rajouter 36 à l'indice du premier élément puisque ce nombre est le code de C1 ou premier élément du

tableau. Notez que les fréquences des notes référencées par leurs indices dans les tableaux sont codées sur deux octets.

*Procédure **jf_soninit** :*

Cette procédure ouvre les trois canaux du générateur sonore.

*Procédure **jf_desact_porta** :*

Cette procédure désactive le bit 1 du port A.

*Procédure **jf_silence** :*

Cette procédure remet à zéro les volumes des trois canaux et ferme ces canaux. La seule fermeture des canaux provoque quand même des bruits très brefs lorsqu'on envoie des notes à jouer sur un canal fermé, ce qui n'est pas logique dans le principe mais est dû au fonctionnement du générateur sonore.

*Procédure **jf_hauteur** :*

Cette procédure est utilisée dès que l'on veut faire jouer une note de hauteur et volume connus sur l'un des trois canaux sonores du générateur. Le tableau **registrel()** est exploré suivant son indice et la hauteur correspondante est envoyée sur les registres hauteurs du générateur sonore. Pour des raisons de rapidité d'exécution, vous pouvez tout à fait supprimer la boucle itérative **FOR NEXT** de degré deux et recopier les deux lignes de programme qui écrivent dans les registres du générateur sonore.

Initialisation de l'édition

La partie édition du programme débute avec l'appel de la procédure **jf_dessin_rythme()** qui fixe et dessine la zone d'écriture de la grille rythmique. C'est cette procédure générale qui appellera les procédures d'édition **(jf_test_rythme)** et de séquence **(jf_play_rythme)**.

Les appellations des paramètres d'appels de cette procédure sont assez explicites ; ces derniers fixent les zones d'affichage de la grille (des lignes verticales), le nombre de pulsations à considérer ainsi que la division des pulsations. Il est possible ici de donner n'importe quelle valeur à cette division et avoir des principes binaires, ternaires ou des quintolets ou septolets ; l'édition proprement dite en sera un peu plus compliquée et ce n'est pas le but de cette première approche. La variable **curseurl** affiche ou pas le curseur du métronome au moyen d'un tiret sous la pulsation **(IF il MOD div_pulsl=0)**.

Comme il est possible de donner des valeurs d'accent dynamique aux notes, des boîtes d'accent sont dessinées au-dessus des lignes de grilles rythmiques. La variable **jf_hautbasel** définit la hauteur du trait de pulsation qui est réduite pour une division de cette pulsation qui ne soit



pas un temps. Les valeurs de trente et de dix (adaptées à l'écran monochrome) seront certainement excessives si votre écran est en couleur.

L'édition proprement dite

*Procédure **jf_test_rythme** :*

Cette procédure est un aiguillage sur les différents traitements. Le corps principal de cette procédure réalise l'écriture des notes sur la grille et leur modification. Le double clic de souris appelle la procédure **jf_play_rythme** qui met en forme et joue la chaîne éditée. La sortie de la fonction peut se faire par la frappe de la touche espace.

Les notes sont dessinées sous forme de cercles noirs proportionnels à la taille du dessin de grille. La logique veut que l'on puisse accentuer des notes existantes et uniquement les débuts des notes dont la longueur excède une division de pulsation. Pour augmenter une longueur de note, il suffit de cliquer à gauche sur une note déjà existante pour augmenter cette longueur d'une division. Le clic à droite supprime la note éventuelle ou l'accent. La partie la plus délicate consiste à traiter les clics droits au milieu de zones représentant plusieurs divisions de pulsations. Il faut alors couper la note, générer un silence sur la position cliquée et créer une nouvelle note sur la position suivante. Pour simplifier le problème, on suppose ici que la longueur maximum est égale à une pulsation, mais il est facile de déduire une procédure de traitement général de coupe de note de longueur quelconque. Dès qu'une note est coupée et qu'il y a régénération de nouvelle grille rythmique, et non simplement pose de note sur un silence ou élongation de note, il faut redessiner la grille au moyen de la procédure **jf_redessin**.

*Procédure **jf_redessin** :*

Cette procédure redessine la grille rythmique à partir des données contenues dans les tableaux **jf_notel()** et **jf_accentl()** qui contiennent respectivement les longueurs de notes en multiples de divisions de pulsations et les accents éventuels sur chaque début de note. Ces tableaux sont explorés de la première à la **jf_nbnote** case en prenant soin de sauter les codages des cases de **jf_notel()** qui suivent la longueur de la note **(ADD jl,jf_notel(jl)-1)**.

Le séquenceur

C'est le rôle de la procédure **jf_playrythme**.

Les tableaux utilisés par le séquenceur sont les tableaux d'entiers (4 octets) **joue%() et jouemetro%()**. Le tableau **joue%()** est rempli consécutivement par tous les événements de début ou fin de note, comme une succession de **Note on** et **Note off** en MIDI puisque il est

inutile de faire jouer des silences et donc de les coder. Dès qu'on trouve une information de codage de début de note dans le tableau **jf_notel()**, on positionne l'indicateur logique **play!** sur **TRUE** pour ne pas monopoliser le générateur sonore sur des silences et on génère une fin de note en rajoutant la durée de la note au temps de début de note. Dans les autres cas, on comptabilise le nombre de divisions de pulsations de silences dans **compt0&** pour rajouter cette valeur au temps de la première note qui suit.

Le tableau du métronome est volontairement rempli afin que vous puissiez facilement utiliser ce séquenceur pour une autre piste.

L'octet de hauteur est ici arbitrairement toujours le même, mais il est très facile de faire jouer une mélodie différente à chacune des pistes en codant les hauteurs de 0 à 48. Si cet ambitus ne vous suffit pas ou si vous voulez transformer la sortie vers le MIDI, il faut le faire à partir de la procédure **jf_registre** et la dimension du tableau **registrel()**. Les tableaux **joue%()** et **jouemetro%()** sont préparés et transformés afin que l'opération de sortie sonore ne comporte plus aucun calcul en temps réel. Si vous décidez de faire jouer une chaîne plus longue et que vous trouvez que le temps de calcul préparatoire est trop important, il suffit de prévoir un buffer de préparation de plusieurs mesures et n'effectuer les calculs que lorsque aucun événement sonore n'est à réaliser, en ayant ainsi toujours quelques mesures d'avance sur la fonction jouer.

La variable **delail** n'est utilisée que pour permettre au générateur sonore de reprendre son souffle lorsque plusieurs notes consécutives sont jouées rapidement. Ce problème n'apparaîtra pas du tout sur le système MIDI. Suivant votre tempo et la répétition des notes un délai de 1 à 5 cinquantièmes de secondes peut suffire en tenant compte des temps d'interprétation des instructions. En fin de parcours, (dernière note de chaîne jouées) les canaux utilisés sont mis à zéro et le port A désactivé.

```
'          PROGRAMME SEQUENCEUR RYTHMIQUE AVEC METRONOME
' Chaque note est codee sur 4 octets:
'     2 premiers: temps
'     3eme        : hauteur (pour code MIDI, rajouter 36)
'     4eme        : volume  (pour MIDI
'
' ------ tableau du rythme à jouer
DIM joue%(256)          ! si on ne dépasse pas 64 notes
' -------- tableau metro (peut devenir une autre piste)
DIM jouemetro%(256)          ! si on ne dépasse pas 64 notes
DIM jf_posmetro&(64)
'
' ------------
tempo|=140
jf_registre
jf_soninit
jf_desact_porta
jf_dessin_rythme(50,500,300,8,4,1)
jf_silence
jf_desact_porta
PROCEDURE jf_hauteur(canal_puce|,note|,volume|)
   SPOKE &HFF8800,8+canal_puce|-1              !vol
   SPOKE &HFF8802,volume|
   FOR k|=0 TO 1
      SPOKE &HFF8800,2*(canal_puce|-1)+k|
      SPOKE &HFF8802,registre|(2*note|+k|)
   NEXT k|
RETURN
PROCEDURE jf_silence
   SPOKE &HFF8800,8
   SPOKE &HFF8802,&H0
   SPOKE &HFF8800,9
   SPOKE &HFF8802,&H0
   SPOKE &HFF8800,10
   SPOKE &HFF8802,&H0
   SPOKE &HFF8800,7
   SPOKE &HFF8802,&HFF
RETURN
PROCEDURE jf_soninit
   SPOKE &HFF8800,7
   SPOKE &HFF8802,&HF8
RETURN
PROCEDURE jf_activ_porta
   ~XBIOS(29,&X11111101) ! bit 1 port A activé
RETURN
```

(suite)

```
PROCEDURE jf_desact_porta
   XBIOS(30,&X10)          ! bit 1 port A désactivé
RETURN
' ---------- trace cases pulsations et divisions +  accents
PROCEDURE jf_dessin_rythme(x_deb&,x_fin&,y_ligne&,nbr_puls|,div_puls|,curseur|)
   GET x_deb&,y_ligne&-200,x_fin&+10,y_ligne&,jf_blanc$
   jf_nbnotes|=nbr_puls|*div_puls|
   jf_longdiv&=(x_fin&-x_deb&) DIV jf_nbnotes|
   IF (jf_longdiv& MOD 2*jf_nbnotes|<>0)
      REPEAT
         DEC x_fin&
         jf_longdiv&=(x_fin&-x_deb&) DIV jf_nbnotes|
      UNTIL jf_longdiv& MOD 2*jf_nbnotes|<>0
   ENDIF
   DIM jf_note|(jf_nbnotes|),jf_pos&(jf_nbnotes|),jf_accent|(jf_nbnotes|)
   jj|=1

   jf_hautbase|=30
   FOR i|=0 TO jf_nbnotes|
      IF i| MOD div_puls|=0
         jf_hautpuls|=jf_hautbase|
      ELSE
         jf_hautpuls|=jf_hautbase|-10
      ENDIF
      LINE x_deb&+i|*jf_longdiv&,y_ligne&,x_deb&+i|*jf_longdiv&,y_ligne&-jf_hautp
uls|
      IF i|<>jf_nbnotes|
         BOX x_deb&+i|*jf_longdiv&,y_ligne&-2*jf_hautbase|,x_deb&+(i|+1)*jf_longdi
v&,y_ligne&-1.5*jf_hautbase|
      ENDIF
      jf_pos&(i|)=x_deb&+i|*jf_longdiv&
      IF i| MOD div_puls|=0
         jf_posmetro&(jj|)=jf_pos&(i|)
         INC jj|
      ENDIF
   NEXT i|
   GET x_deb&,y_ligne&-jf_hautpuls|,x_fin&,y_ligne&,jf_affrythme$
   IF curseur|
      affichmetro!=TRUE
   ELSE
      affichmetro!=FALSE
   ENDIF
   jf_test_rythme
   EDIT
RETURN
```



(suite)

```
' ------ remplissage tableau par souris
PROCEDURE jf_test_rythme
  DO
    IF MOUSEK=3
      ' ------- procedure transforme edition en tableau pour jouer les notes
      jf_playrythme
    ENDIF
    EXIT IF INKEY$=" "
    MOUSE x&,y&,k|
    IF k|=1
      DEFFILL 1,1
      FOR i|=0 TO jf_nbnotes|-1
        IFx&>jf_pos&(i|)ANDx&<jf_pos&(i|+1)ANDy&<y_ligne&ANDy&>y_ligne&-
jf_hautbase|
          IF jf_note|(i|) AND jf_note|(i|)<div_puls|
            IF jf_note|(i|+jf_note|(i|))=0
              FOR ii|=0 TO jf_longdiv&
                PCIRCLE (jf_pos&(i|)+jf_pos&(i|+1)) DIV 2+ii|*jf_note|(i|),y_li
gne&,jf_longdiv& DIV 2
              NEXT ii|
              INC jf_note|(i|)
              jf_note|(i|+jf_note|(i|)-1)=9+jf_note|(i|)
              REPEAT
              UNTIL MOUSEK=0
            ENDIF
          ENDIF
          IF jf_note|(i|)=0
            PCIRCLE (jf_pos&(i|)+jf_pos&(i|+1)) DIV 2,y_ligne&,jf_longdiv& DIV 2
            jf_note|(i|)=1
            REPEAT
            UNTIL MOUSEK=0
          ENDIF
        ENDIF
        IFx&>x_deb&+i|*jf_longdiv&ANDy&>y_ligne&-2*jf_hautbase|ANDx&<x_deb
&+(i|+1)*jf_longdiv& AND y&<y_ligne&-1.5*jf_hautbase|
          IF jf_accent|(i|)=0 AND jf_note|(i|)>0 AND jf_note|(i|)<5
            PBOX x_deb&+i|*jf_longdiv&,y_ligne&-2*jf_hautbase|,x_deb&+(i|+1)*jf
_longdiv&,y_ligne&-1.5*jf_hautbase|
            jf_accent|(i|)=1
          ENDIF
        ENDIF
      NEXT i|
    ENDIF
```

(suite)

```
      IF k|=2
        DEFFILL 0,1
        jf_ellipse!=FALSE
        FOR i|=0 TO jf_nbnotes|-1
          ' on efface si écrit en position simple
          IFx&>jf_pos&(i|)ANDx&<jf_pos&(i|+1)ANDy&<y_ligne&ANDy&>y_ligne&-
jf_hautbase| AND jf_note|(i|)>1
            ' si c'est le dernier de la chaine
            IF jf_note|(i|+1)<10
              jf_reste|=jf_note|(i|)-10
              DEC jf_note|(i|-jf_reste|)
              REPEAT
              UNTIL MOUSEK=0
              jf_note|(i|)=0
              jf_redessin
            ELSE
              ' ce n'est pas le dernier il faut créer un silence et des notes
              jf_corpos&=jf_note|(i|)-10              !numéro de la division
              SELECT jf_corpos&
              CASE 1              !deuxième
                jf_note|(i|+1)=1
                IF jf_note|(i|-1)=4
                  jf_note|(i|+2)=11
                  INC jf_note|(i|+1)
                ENDIF
                jf_note|(i|-1)=1
              CASE 2              ! troisième
                jf_note|(i|-2)=2
                jf_note|(i|-1)=11
                jf_note|(i|+1)=1
              DEFAULT             !premier
                IF jf_note|(i|)>2
                  DEC jf_note|(i|+2)
                ENDIF
                IF jf_note|(i|)>3
                  DEC jf_note|(i|+3)
                ENDIF
                jf_note|(i|+1)=jf_note|(i|)-1
                IF jf_accent|(i|)
                  jf_accent|(i|)=0
                  jf_accent|(i|+1)=1
                ENDIF
              ENDSELECT
```

(suite)

```
            REPEAT
            UNTIL MOUSEK=0
            jf_note(il)=0                          !silence
            jf_redessin
          ENDIF
        ENDIF
        IFx&>jf_pos&(il)ANDx&<jf_pos&(il+1)ANDy&<y_ligne&ANDy&>y_ligne&-
jf_hautbase AND jf_note(il)=1
          REPEAT
          UNTIL MOUSEK=0
          IF jf_accent(il)
            jf_accent(il)=0
          ENDIF
          jf_note(il)=0
          jf_redessin
        ENDIF
        IFx&>x_deb&+il*jf_longdiv&ANDy&>y_ligne&-2*jf_hautbaseANDx&<x_deb
&+(il+1)*jf_longdiv& AND y&<y_ligne&-1.5*jf_hautbase
          IF jf_accent(il)=1
            PBOX x_deb&+il*jf_longdiv&,y_ligne&-2*jf_hautbase,x_deb&+(il+1)*jf
_longdiv&,y_ligne&-1.5*jf_hautbase
            jf_accent(il)=0
          ENDIF
        ENDIF
      NEXT il
    ENDIF
  LOOP
RETURN
PROCEDURE jf_registre
  DIM registre(97)                                 !4 octaves * 12 demi-tons + 1
  FOR octave&=-2 TO 2
    FOR note=1 TO 12
      EXIT IF octave&=2 AND note=2
      freq=440*(2^(octave&+((note-10)/12)))
      per&=INT(125000/freq)
      msb=INT(per&/256)
      lsb=per&-msb*256
      k=(octave&+2)*12+note-1
      registre(2*k)=lsb
      registre(2*k+1)=msb
    NEXT note
    EXIT IF octave&=2 AND note=2
  NEXT octave&
RETURN
```

(suite)

```
PROCEDURE jf_redessin
  DEFFILL 1,1
  CLS
  PUT x_deb&,y_ligne&-jf_hautpuls|,jf_blanc$
  PUT x_deb&,y_ligne&-jf_hautpuls|,jf_affrythme$
  FOR j|=0 TO jf_nbnotes|-1
    IF jf_note|(j|)=1
      PCIRCLE (jf_pos&(j|)+jf_pos&(j|+1)) DIV 2,y_ligne&,jf_longdiv& DIV 2
      IF jf_accent|(j|)=1
        PBOX x_deb&+j|*jf_longdiv&,y_ligne&-2*jf_hautbase|,x_deb&+(j|+1)*jf_lon
gdiv&,y_ligne&-1.5*jf_hautbase|
      ENDIF
    ENDIF
    IF jf_note|(j|)>1
      IF jf_accent|(j|)=1
        PBOX x_deb&+j|*jf_longdiv&,y_ligne&-2*jf_hautbase|,x_deb&+(j|+1)*jf_lon
gdiv&,y_ligne&-1.5*jf_hautbase|
      ENDIF
      PCIRCLE (jf_pos&(j|)+jf_pos&(j|+1)) DIV 2,y_ligne&,jf_longdiv& DIV 2
      FOR ii|=0 TO jf_longdiv&
        PCIRCLE (jf_pos&(j|)+jf_pos&(j|+1)) DIV 2+ii|*(jf_note|(j|)-1),y_ligne&
,jf_longdiv& DIV 2
      NEXT ii|
      ADD j|,jf_note|(j|)-1
    ENDIF
  NEXT j|
  x&=0
  y&=0
RETURN
PROCEDURE jf_playrythme
  ' ------- remplir tout le tableau avec des 0
  ARRAYFILL joue%(),0
  ARRAYFILL jouemetro%(),0
  play!=FALSE
  j&=1
  h|=10                !hauteur par défaut
  v|=12                ! volume par défaut
  compt0&=0
  ' -Si on veut une 2eme piste reutiliser cette boucle avec autres tableaux
  FOR i%=0 TO jf_nbnotes|-1
    ' --- si note a jouer
```



(suite)

```
    IF jf_note|(i%)
      play!=TRUE
      ' ----- duree (2 octets)
      DPOKE V:joue%(j&),DPEEK(V:joue%(j&-1))+compt0&
      ' ------ hauteur(+36 POUR midi)
      POKE V:joue%(j&)+2,h|
      ' ---- volume (pour MIDI, 5*V|+20*JF_ACCENT|)
      POKE V:joue%(j&)+3,v|+2*jf_accent|(i%)
      ' ----- fin de la note (note off)
      DPOKE V:joue%(j&+1),DPEEK(V:joue%(j&))+jf_note|(i%)
      POKE V:joue%(j&+1)+2,h|
      POKE V:joue%(j&+1)+3,0
      ' ------------------------------------------------
      compt0&=0
      ' ---- sauter la duree de la note
      ADD i%,jf_note|(i%)-1
      ' ------ chaque note = on-off(= 2 elements du tableau)
      ADD j&,2
    ELSE
      ' ---- sinon, silence, donc incrementer compteur de silence
      INC compt0&
    ENDIF
  NEXT i%
  ' ---------- si au moins une note a jouer
  IF play!=TRUE
    PUT x_deb&,y_ligne&+10,jf_blanc$  !effacer le metronome
    PRINT AT(5,5);"play"
    ' ------ transformation de temps en secondes par rapport au tempo
    FOR i&=1 TO j&
      DPOKE (V:joue%(i&)),(DPEEK(V:joue%(i&))*12000) DIV div_puls|*tempo|
    NEXT i&
    j1&=1
    ' ------ compte le nombre de silences
    compt0&=0
    ' ------- METRONOME
    FOR i%=0 TO 128                ! si pas plus de 64 notes
      IF i% MOD div_puls|=0
        ' ------- NOTE ON
        DPOKE V:jouemetro%(j1&),DPEEK(V:jouemetro%(j1&-1))+compt0&
        POKE V:jouemetro%(j1&)+2,h|+12
        POKE V:jouemetro%(j1&)+3,v|-1
```

(suite)

```
        ' ------ NOTE OFF
        DPOKE V:jouemetro%(j1&+1),DPEEK(V:jouemetro%(j1&))+1
        POKE V:jouemetro%(j1&+1)+2,h|+12
        POKE V:jouemetro%(j1&+1)+3,0
        compt0&=0
        ADD j1&,2
      ELSE
        INC compt0&
      ENDIF
    NEXT i%
    FOR i&=1 TO j1&
      DPOKE(V:jouemetro%(i&)),(DPEEK(V:jouemetro%(i&))*12000)DIVdiv_puls|*te
mpo|
    NEXT i&
    ' DEC j&
    ' ------ REINITISALISER LES DEUX INDICES (1 PAR CHAINE A JOUER)
    i&=1          !chaine joue
    i1&=1         ! metronome
    ' ------------------ jouer
    t%=TIMER
    REPEAT
      ' ------ mesure a vide(4 pulsations)
      IF TIMER-t%>=DPEEK(V:jouemetro%(i1&))
        jf_hauteur(2,PEEK(V:jouemetro%(i1&)+2),PEEK(V:jouemetro%(i1&)+3))
        INC i1&
      ENDIF
    UNTIL i1&=10
    ' ------ reinitialiser
    i1&=1
    t%=TIMER
    REPEAT
      ' --- pour chaine joue, si quelque chose pas de delai
      IF PEEK(V:joue%(i&)+3)
        delai|=0
        ' ------- si silence, ecourter la note
      ELSE
        '             delai|=5
        delai|=3
      ENDIF
```



(suite)

```
            ' ------- SEQUENCEUR_____________
            '
            ' ------ CHAINE 1
            IF TIMER-t%>=DPEEK(V:joue%(i&))-delai!
              jf_hauteur(1,PEEK(V:joue%(i&)+2),PEEK(V:joue%(i&)+3))
              INC i&
            ENDIF
            ' ------ CHAINE 2
            IF TIMER-t%>=DPEEK(V:jouemetro%(i1&))
              ' si on veut afficher le métronome
              IF affichmetro!=TRUE
                TEXT jf_posmetro&(i1& DIV 2+i1& MOD 2),y_ligne&+20,"_"
              ENDIF
              jf_hauteur(2,PEEK(V:jouemetro%(i1&)+2),PEEK(V:jouemetro%(i1&)+3))
              INC i1&
            ENDIF
            ' SI 2 ou 3 EME CHAINE, RAJOUTER ICI IDEM, SUR CANAL 2 et 3
          UNTIL i&>j&+1 OR PEEK(&HFFFC02)=57  !JUSAU'AU BOUT DES NOTES
          jf_hauteur(2,0,0)
          jf_hauteur(1,0,0)
          jf_desact_porta
          PRINT AT(5,5);"fini"
          PAUSE 50
        ENDIF
      RETURN
```

6/1.4

Création d'enveloppe à points multiples

Ce programme permet de réaliser une enveloppe à points multiples, le nombre maximum de points dépendant de votre bon vouloir et des dimensions du cadre rectangulaire choisi pour représenter cette enveloppe.

Par définition, une enveloppe est une courbe qui guide les variations d'un des paramètres du son, le plus communément l'amplitude, mais aussi la hauteur ou la fréquence de coupure d'un filtre ; ici nous nous limiterons à appliquer cette courbe à l'amplitude afin de pouvoir utiliser la puce son Atari, mais rien ne vous empêche d'utiliser ce programme de façon externe à l'Atari.

Afin de lever toute ambiguïté quant aux différences entre enveloppe et forme d'onde, il convient de noter qu'une forme d'onde est de façon générale cyclique même si le phénomène n'est pas périodique, alors que l'enveloppe est souvent liée à un jeu du clavier et ne contient pas de points de bouclage ; le propos du programme qui va suivre permet de jongler entre ces deux définitions, le support technique qu'offre l'Atari permettant de simuler aussi bien une enveloppe qu'une forme d'onde.

Le principe général est simple : disposer d'une courbe à points multiples, ici une ligne brisée à segments linéaires, et pouvoir choisir les points de bouclage du son. Il nous faut donc pouvoir réaliser plusieurs fonctions :

— Affichage de la courbe avec Polyline.

— Marquage des points avec Polymark.

— Suppression et addition de points.

— Lecture des pentes et des segments de droites.

— Transcription des coordonnées des points de la courbe en valeurs de volume (de 0 à 15 ici).

— Modification des paramètres de lecture et de jeu des tableaux de sons.

— Simulation du release par cliquage de la souris.

— Opérations conventionnelles de sauvegardes, chargement de banques de sons et copies d'enveloppes.

Dans un premier temps, il nous faut réaliser une présentation du tableau de travail : le cadre externe de l'enveloppe, les boîtes de paramètres et la visualisation d'une enveloppe initiale choisie par défaut à quatre points, attaque rapide et sans release.

Noter que les paramètres **susg%** et **susd%** indiqueront en permanence les numéros des points de bouclage gauche et droit de l'enveloppe : il suffira qu'ils soient confondus pour pouvoir simuler une enveloppe sans sustain.

ENVELOPPE MULTI-POINTS

Initialisation des variables

```
sr%=1
trait$="|"
tiret$="-------"
```

Variables chaînes de caractères pour les boîtes d'alerte

```
text1$="voulez-vous remplacer | le timbre N°"
text2$="voulez-vous changer le nom | du timbre N°"
bout$="oui | non"
flag2!=FALSE
dimmax%=50!NBRE MAXIMUM DE POINTS D'ENVELOPPE
maxpermis%=500
```

Tableaux de valeurs des ordonnées des points graphiques et des indices correspondant aux quinze valeurs de volume ; ceci permettra de générer directement et sans calculs le release à partir de la valeur du volume au moment du cliquage de la souris.

```
DIM tabjoue%(1000),tabrelease%(5000) ! sons à jouer
DIM kk%(15),pente(dimmax%)! pentes de l'enveloppe
```

Tableaux des points de bouclage de la forme d'onde :

```
DIM loopx%(1),loopy%(1)!
```

Tableaux chaînes de caractères de la bibliothèque de sons :

```
DIM transfert$(8),long%(8)
FOR i%=1 TO 8
  transfert$(i%)=STRING$(500,254)
NEXT i%
```



Variable chaîne de caractères pour les sauvegardes et chargements des sons :

```
transfert$=STRING$(12000,254)
GET 0,0,120,15,bloncS
BOX 5,15,50,40
BOX 51,15,100,40
BOX 101,15,180,40
BOX 181,15,240,40
BOX 241,15,300,40
BOX 301,15,360,40
BOX 361,15,420,40
BOX 421,15,460,40
BOX 461,15,500,40
PRINT AT(2,2);"jouer"
PRINT AT(8,2);"vol"
PRINT AT(14,2);"temps"
PRINT AT(24,2);"rel"
PRINT AT(32,2);"pas"
PRINT AT(39,2);"susg"
PRINT AT(47,2);"susd"
PRINT AT(54,2);"save"
PRINT AT(59,2);"load"
DIM xa%(dimmax%),ya%(dimmax%)
```

Initialisation de la représentation graphique de l'enveloppe

- distance minimum en pixels entre deux points d'enveloppe **d% = 6**
- valeur absolue du périmètre de saisie d'un point d'enveloppe **dd% = 8**
- abscisse du point haut de la boîte enveloppe **x0% = 100**
- ordonnée du point haut de la boîte enveloppe **y0% = 100**
- longueur de la boîte enveloppe **x1% = 400**
- hauteur de la boîte enveloppe **y1% = 200**
- abscisse du point bas de la boîte enveloppe **x1% = x0% + x1%**
- ordonnée du point bas de la boîte enveloppe **y1% = y0% + y1%**
- pour ramener les ordonnées entre 0 et 15 **coeff = 15/(y1% - 2*d%)**

Boîtes blanches pour effacer les titres et les enveloppes ultérieurement

```
GET x0%,y0%,x0%+x1%,y0%+y1%,blanc$
GET dd%+1,y0%,x0%-2,y0%-2+(y1%+2)/8,nomblanc$
```

Boîtes de noms de sons

```
FOR i%=1 TO 8
  toi%=y0%-1+i%*(yl%+2)/8
  BOX dd%,y0%-1+(i%-1)*(yl%+2)/8,x0%-1,toi%
  BOX x1%+1,y0%-1+(i%-1)*(yl%+2)/8,x1%-dd%+x0%-1,toi%
NEXT i%
```

Fenêtre de l'enveloppe :

```
BOX X0%-1,Y0%-1,X0%+Xl%+1,YO%+Yl%+1
```

ENVELOPPES DE DEPART :

```
FOR te%=1 TO 8
  MID$(transfert$(te%),1)=CHR$(86)
  MID$(transfert$(te%),2)=CHR$(79)
  MID$(transfert$(te%),3)=CHR$(73)
  MID$(transfert$(te%),4)=CHR$(88)
  MID$(transfert$(te%),5)=CHR$(66)
  MID$(transfert$(te%),6)=CHR$(65)
  MID$(transfert$(te%),7)=CHR$(83)
  MID$(transfert$(te%),8)=CHR$(69)
 • nombre de points de l'enveloppe saisis
  ICI 4 PAR DEFAUT
nbre%=4
```

• facteur multiplicatif du temps d'attente d'un point au suivant sur le release de l'enveloppe **rel% = 4**

• volume de la piste **vol% = 15**

• on fixe le pas de lecture des tableaux d'enveloppe ici on lira point par point, si le pas était de deux, on lirait un point sur deux, etc. **pas% = 1**

• fixation des points de bouclage **susg% = 2**

• le point droit de bouclage sera toujours l'avant dernier point de l'enveloppe **susd% = nbre% - 1**

• temps d'attente dans la scrutation de la boucle son **temps% = 1**

```
xa%(0)=x0%+2*dd%
  ya%(0)=y1%-d%
  xa%(1)=xa%(0)+6
  ya%(1)=120
  xa%(2)=450
  ya%(2)=120
```



```
    xa%(3)=456
    ya%(3)=y1%-d%
    sauvertimbre(te%)
    TEXT dd%+1,y0%-4+te%*(y1%+2)/8,MID$(transfert$(te%),1,8)
  NEXT te%
```

On vient de remplir la bibliothèque de départ avec huit sons identiques initiaux.

On appelle maintenant les procédures d'affichage des paramètres de jeux et de l'enveloppe conformément aux valeurs entrées dans les tableaux xa et ya.

```
Paramètres
enveloppe
TEXT 230,60,"INIT"
TEXT 120,60,"BIBLIOTHEQUE"
TEXT 120,80,"TIMBRE"
TEXT 20,80,"SONS"
TEXT 520,80,"SAUVER"
```

On affiche par défaut le premier timbre de la bibliothèque

```
chargertimbre (1)
```

——————— ici commence le programme principal——————— après une modification éventuelle des paramètres de jeu, on peut demander de jouer le son qui correspond à l'enveloppe affichée ; le programme calcule tous les points et génère les indices de prise en compte du tableau de release de façon systématique ; dans l'optique d'une utilisation de banques de sons sauvegardées sur disquette, il conviendra de court-circuiter cette phase de calculs et lire simplement les tableaux remplis. Dans la phase normale de jeu, la procédure jouer calcule tous les points des tableaux, la procédure son joue les ordonnées des points jusqu'au point de sustain droit, la procédure sonsus boucle le son entre les deux points éventuels de bouclage, la procédure sonrelease joue le son à partir de la valeur de l'indice au moment du cliquage.

```
DO
  MOUSE x%,y%,k%
  IF k%=1
```

On teste le clic gauche dans la boîte jouer :

```
IF x%>15 AND x%<50 AND y%>10 AND y%<40
    HIDEM
    jouer
    son
    sonsus
    sonrelease(tabjoue%(json%))
    final
    SHOWM
  ENDIF
```

On teste si on clique à gauche dans une des boîtes afin d'incrémenter les valeurs des paramètres.

```
IF x%>85 AND x%<95 AND y%<30 AND y%>18
  INC vol%
  vol%=MIN(vol%,15)
  PAUSE 4
  PRINT AT(11,2);USING "##",vol%
ENDIF
IF x%>135 AND x%<165 AND y%<30 AND y%>18
  INC temps%
  temps%=MIN(temps%,150)
  PAUSE 4
  PRINT AT(19,2);USING "###",temps%
ENDIF
IF x%>215 AND x%<235 AND y%<30 AND y%>18
  INC rel%
  rel%=MIN(rel%,99)
  PAUSE 4
  PRINT AT(28,2);USING "##",rel%
ENDIF
IF x%>280 AND x%<295 AND y%<30 AND y%>18
  INC pas%
  pas%=MIN(pas%,50)
  PAUSE 4
  PRINT AT(36,2);USING "##",pas%
ENDIF
```

Dès qu'on modifie le point de bouclage gauche il faut refaire le marquage des points de l'enveloppe.

```
IF x%>345 AND x%<360 AND y%<30 AND y%>18
    susg1%=susg%+1
    susg1%=MIN(susg1%,susd%)
    IF susg1%<>susg%
      susg%=susg1%
      PAUSE 4
      enveloppe

    ENDIF
  ENDIF
```

Ici on se branche sur les procédures de sauvegarde et chargement des bibliothèques de timbre.

```
IF x%>425 AND x%<455 AND y%>18 AND y%<30
  sauversons
ENDIF
IF x%>465 AND x%<495 AND y%>18 AND y%<30
  chargersons
ENDIF
```

Si on clique dans la boîte de l'enveloppe

```
IF x%>xa%(0) AND x%<xa%(nbre%-1)+dd%
      IF y%<y1%-d% AND y%>y0%+d%
        flag1!=FALSE
```

Si on attrape un point existant de l'enveloppe

```
FOR i%=1 TO nbre%-2
  IF ABS(x%-xa%(i%))<dd% AND ABS(y%-ya%(i%))<dd%
    flag1!=TRUE
    REPEAT
      xa%=MOUSEX
      ya%=MOUSEY
      xa%=MAX(xa%,xa%(i%-1)+dd%)
      xa%=MIN(xa%,xa%(i%+1)-dd%)
      ya%=MIN(ya%,y1%-d%)
      ya%=MAX(ya%,y0%+d%)
```

on ne redessine l'enveloppe que si le nouveau point a une position différente de la précédente

```
        IF ABS(xa%-xa%(i%))>d% OR ABS(ya%-ya%(i%))>d%
          xa%(i%)=xa%
          ya%(i%)=ya%
          enveloppe
        ENDIF
      UNTIL MOUSEK=0
    ENDIF
  NEXT i%
```

Si on attrape le dernier point de l'enveloppe, il faut traiter ce cas à part puisque ce point n'a pas de suivant, mais sa position est limitée par la fenêtre de l'enveloppe.

```
IF ABS(x%-xa%(nbre%-1))<dd% AND ABS(y%-ya%(nbre%-1))<dd%
          flag1!=TRUE
          REPEAT
            xa%=MOUSEX
            xa%=MAX(xa%,xa%(nbre%-2)+dd%)
            xa%=MIN(xa%,x1%-dd%)
            IF ABS(xa%-xa%(nbre%-1))>d%
              xa%(nbre%-1)=xa%
              enveloppe
            ENDIF
          UNTIL MOUSEK=0
        ENDIF
```

Si on crée un nouveau point d'enveloppe, il suffit de tester si on clique dans le cadre de l'enveloppe en dehors d'un des points existant précédemment ; il faut bien sûr tenir compte de la taille des motifs de marquage des points de l'enveloppe, pour dire que deux points sont disctincts.

Autres précautions à prendre :

— on ne peut rajouter de point à gauche du point de départ,

— il faut modifier les valeurs de susg% et susd% lorsqu'on insère un point ; ces modifications dépendent de la position de ce dernier par rapport aux points de bouclage et impliquent un décalage gauche ou droit des indices des tableaux xa%() et ya%(),

— lorsqu'un nouveau point est créé, on a évidemment la possibilité de déplacer ce point comme s'il existait précédemment, jusqu'au relâchement de la souris ;

```
IF flag1!=FALSE AND flag2!=FALSE
          IF nbre%<dimmax%
            FOR i%=0 TO nbre%-2
              IF x%>xa%(i%)+dd% AND x%<xa%(i%+1)-dd%
                INC nbre%
                IF x%<xa%(susd%-1)
                  INC susd%
                ENDIF
                IF x%<xa%(susg%-1)
                  INC susg%
                ENDIF
                FOR j%=nbre%-1 DOWNTO i%+2
                  xa%(j%)=xa%(j%-1)
                  ya%(j%)=ya%(j%-1)
                NEXT j%
                xa%(i%+1)=x%
```



(suite)

```
            ya%(i%+1)=y%
            flag2!=TRUE
          ENDIF
          EXIT IF flag2!=TRUE
        NEXT i%
        IF flag2!=TRUE
          enveloppe
          REPEAT
            xa%=MOUSEX
            ya%=MOUSEY
            xa%=MAX(xa%,xa%(i%)+dd%)
            xa%=MIN(xa%,xa%(i%+2)-dd%)
            ya%=MIN(ya%,y1%-d%)
            ya%=MAX(ya%,y0%+d%)
            xaa%= ABS(xa%-xa%(i%+1))
            yaa%=ABS(ya%-ya%(i%+1))
            IF xaa%>d% OR yaa%>d%
              xa%(i%+1)=xa%
              ya%(i%+1)=ya%
              enveloppe
            ENDIF
            UNTIL MOUSEK=0
          ENDIF
          flag2!=FALSE
        ELSE
```

ceci uniquement pour le cas où le nombre de points maximum que vous avez fixé est dépassé.

```
            PRINT "alert"
            PRINT AT(12,12);"nbre%= ";nbre%
          ENDIF
        ENDIF
      ENDIF
    ENDIF
    tre%=0
```

Test de cliquage dans les boîtes de noms de timbre :

Le timbre qui correspond à l'enveloppe affichée a sa boîte visualisée en gras.

Pour afficher un timbre, il suffit de tirer la souris dans la fenêtre de l'enveloppe ; la procédure chargertimbre se charge d'afficher le timbre sélectionné.

Pour copier une enveloppe il suffit de tirer la souris sur une autre boîte de timbre.

Pour sauver un timbre édité il faut cliquer sur la boîte située au même niveau à droite en regard sous le titre SAUVER ; nous aurons l'apparition de boîtes d'alarme pour éventuellement sauvegarder ce timbre suivant son indice dans le tableau chaîne de caractères transfert$ et changer son appellation ;

```
FOR te%=1 TO 8
      IF x%>dd% AND y%>y0%-1+(te%-1)*(yl%+2)/8
        IF x%<x0%-1 AND y%<y0%-1+te%*(yl%+2)/8
          DEFMOUSE 4
          REPEAT
          UNTIL MOUSEK=0
          IF MOUSEX>x0% AND MOUSEX<xl%
           IF  MOUSEY>y0% AND MOUSEY<yl%
            chargertimbre(te%)
           ENDIF
          ENDIF
          FOR ii%=1 TO 8
            IF MOUSEX>dd% AND MOUSEY>y0%-1+(ii%-1)*(yl%+2)/8
              IF MOUSEX<x0%-1 AND MOUSEY<y0%-1+ii%*(yl%+2)/8
                IF ii%<>te%
                  transfert$(ii%)=transfert$(te%)
                  impr$=mid$(transfert$(ii%),1,8)
                  TEXT dd%+1,y0%-4+ii%*(yl%+2)/8,impr$
                  IF ii%=sr%
                    chargertimbre(ii%)
                  ENDIF
                ENDIF
              ENDIF
            ENDIF
          NEXT ii%
          DEFMOUSE 0
        ENDIF
      ENDIF
```

(suite)

```
      IF x%>x1% AND y%>y0%-1+(te%-1)*(y1%+2)/8
        IF x%<x1%-dd%+x0%-1 AND y%<y0%-1+te%*(y1%+2)/8
          text1$=text1$+STR$(te%)
          ALERT 2,text1$,1,bout$,rep1%
          text1$=LEFT$(text1$,34)
          IF rep1%=1
            text2$=text2$+STR$(te%)
            COLOR 0
            ydeb%=y0%+(sr%-1)*(y1%+2)/8
            yfin%=y0%-2+sr%*(y1%+2)/8
            BOX dd%+1,ydeb%,x0%-2,yfin%
            COLOR 1
            ydeb%=y0%+(te%-1)*(y1%+2)/8
            yfin%=y0%-2+te%*(y1%+2)/8
            BOX dd%+1,ydeb%,x0%-2,yfin%
            sr%=te%
            ALERT 2,text2$,1,bout$,rep2%
            text2$=LEFT$(text2$,39)
            IF rep2%=1
              PUT dd%+1,y0%+(te%-1)*(y1%+2)/8,nomblanc$
              TEXT dd%+1,y0%-6+te%*(y1%+2)/8,tiret$
              ydeb%=y0%+(te%-1)*(y1%+2)/8
              yfin%=y0%-2+te%*(y1%+2)/8
              BOX dd%+1,ydeb%,x0%-2,yfin%
              PUT 230,69,nomblanc$
              car%=0
              tre%=0
              WHILE car%<=7
                IF INP?(2)
                  tre%=INP(2)
                  IF tre%<>8
                    INC car%
                    EXIT IF tre%=13
                    ypos%=y0%-6+te%*(y1%+2)/8
                    TEXT dd%+3+8*car%,ypos%,trait$
                    TEXT dd%+2+8*(car%-1),ypos%,CHR$(tre%)
                    MID$(transfert$(te%),car%)=CHR$(tre%)
                  ELSE
                    IF car%>0
                      ypos%=y0%-6+te%*(y1%+2)/8
                      TEXT dd%+2+8*(car%),ypos%,CHR$(32)
                      TEXT dd%+3+8*(car%-1),ypos%,trait$
                      MID$(transfert$(te%),car%)=CHR$(32)
                      DEC car%
                    ENDIF
                  ENDIF
                ENDIF
```

(suite)

```
                EXIT IF tre%=13
              WEND
              IF car%<9
               SP$=SPACE$(9-car%)
               MID$(transfert$(te%),car%,9-car%)=SP$
              ENDIF
              ypos1%=y0%+(te%-1)*(y1%+2)/8
              ypos2%=y0%-2+te%*(y1%+2)/8
              PUT dd%+1,ypos1%,nomblanc$
              BOX dd%+1,ypos1%,x0%-2,ypos2%
              TEXT 230,80,MID$(transfert$(te%),1,8)
              TEXT dd%+2,ypos2%-4,MID$(transfert$(te%),1,8)
            ENDIF
            sauvertimbre(te%)
          ENDIF
        ENDIF
      ENDIF
    NEXT te%
  ENDIF
```

Ici commencent les tests de décrémentation des paramètres de jeu :

```
IF k%=2
  IF x%>85 AND x%<95 AND y%<30 AND y%>18
    DEC vol%
    vol%=MAX(vol%,0)
    PAUSE 5
    PRINT AT(11,2);USING "##",vol%
  ENDIF
  IF x%>135 AND x%<165 AND y%<30 AND y%>18
    DEC temps%
    temps%=MAX(temps%,1)
    PAUSE 4
    PRINT AT(19,2);USING "###",temps%
  ENDIF
  IF x%>215 AND x%<235 AND y%<30 AND y%>18
    DEC rel%
    rel%=MAX(rel%,1)
    PAUSE 4
    PRINT AT(28,2);USING "##",rel%
  ENDIF
  IF x%>280 AND x%<295 AND y%<30 AND y%>18
```

(suite)

```
    DEC pas%
    pas%=MAX(pas%,1)
    PAUSE 4
    PRINT AT(36,2);USING "##",pas%
  ENDIF
  IF x%>345 AND x%<360 AND y%<30 AND y%>18
    susg1%=susg%-1
    susg1%=MAX(susg1%,2)
    IF susg1%<>susg%
      susg%=susg1%
      PAUSE 4
      enveloppe
    ENDIF
  ENDIF
```

Si on supprime un point de l'enveloppe, on peut supprimer un point d'enveloppe en cliquant sur ce dernier avec la touche dte de la souris. L'enveloppe conservera un minimum de trois points ; les réajustements des valeurs de susg% et susd% se font suivant le même principe que lors de l'addition de points ;

```
    IF nbre%>3
      FOR i%=1 TO nbre%-2
        IF ABS(x%-xa%(i%))<dd% AND ABS(y%-ya%(i%))<dd%
          IF x%<xa%(susg%-1)
            DEC susg%
          ENDIF
          FOR j%=i% TO nbre%-2
            xa%(j%)=xa%(j%+1)
            ya%(j%)=ya%(j%+1)
          NEXT j%
          DEC nbre%
          enveloppe
        ENDIF
      NEXT i%
    ENDIF
  ENDIF
LOOP
final
```

La procédure enveloppe joint tous les points de l'enveloppe à partir des tableaux xa%() et ya%() et marque les deux points de bouclage correspondant au sustain du son par l'appel de la procédure susenvelop ;

```
PROCEDURE enveloppe
  PUT x0%,y0%,blanc$
  POLYLINE nbre%,xa%(),ya%()
  DEFMARK 1,4
  POLYMARK nbre%,xa%(),ya%()
  susenvelop
RETURN

PROCEDURE susenvelop
  DEFMARK 1,2
  susd%=nbre%-1
  susg%=MIN(susg%,susd%)
  susg%=MAX(susg%,2)
  PRINT AT(44,2);USING "##",susg%
  PRINT AT(51,2);USING "##",susd%
  loopx%(0)=xa%(susg%-1)
  loopx%(1)=xa%(susd%-1)
  loopy%(0)=ya%(susg%-1)
  loopy%(1)=ya%(susd%-1)
  POLYMARK 2,loopx%(),loopy%()
RETURN
```

Après avoir calculé tous les points de l'enveloppe de façon linéaire par l'intermédiaire des pentes des segments de droites, les ordonnées sont rangées dans le tableau tabjoue% après avoir été ramenées dans l'intervalle 0-15 ; cette procédure appelle la procédure tabrelease qui s'occupe de générer la droite de même pente que le dernier segment de l'enveloppe.





```
PROCEDURE jouer
  mise en tableau de l'ordonnée enveloppe
  FOR ii%=0 TO nbre%-2
    pente(ii%)=(ya%(ii%)-ya%(ii%+1))/(xa%(ii%+1)-xa%(ii%))
  NEXT ii%
  FOR ii%=0 TO nbre%-3
    FOR jj%=xa%(ii%) TO xa%(ii%+1)
     stk%=coeff*(y0%+yl%-ya%(ii%)+pente(ii%)*(jj%-xa%(ii%)))
     tabjoue%(jj%-x0%-2*dd%)=stk%
     IF tabjoue%(jj%-x0%-2*dd%)>15
       tabjoue%(jj%-x0%-2*dd%)=15
     ENDIF
     IF tabjoue%(jj%-x0%-2*dd%)<0
       tabjoue%(jj%-x0%-2*dd%)=0
     ENDIF
   NEXT jj%
 NEXT ii%
  indice1%=jj%-x0%
  FOR jj%=xa%(nbre%-2)+1 TO xa%(ii%+1)
    st1%=yl%-ya%(nbre%-2)+pente(nbre%-2)*(jj%-xa%(nbre%-2))
    st%=coeff*(y0%+st1%)
    tabjoue%(jj%-x0%-2*dd%)=st%
    IF tabjoue%(jj%-x0%-2*dd%)>15
      tabjoue%(jj%-x0%-2*dd%)=15
    ENDIF
    IF tabjoue%(jj%-x0%-2*dd%)<0
      tabjoue%(jj%-x0%-2*dd%)=0
    ENDIF
  NEXT jj%
  tabrelease
RETURN
```

Cette procédure joue le son sur le canal 1 avec une note de hauteur constante du point de départ au point de bouclage gauche.

```
PROCEDURE son
  FOR json%=2 TO xa%(susg%-1)-x0%-2*dd% STEP pas%
    SOUND 1,tabjoue%(json%)*vol%/15,10,2
    EXIT IF MOUSEK=3
    REPEAT
      EXIT IF MOUSEK=3
      INC compteur%
    UNTIL compteur% MOD temps%=0
    EXIT IF MOUSEK=3
  NEXT json%
RETURN
```

Cette procédure effectue un travail similaire mais sur la partie bouclée de l'enveloppe, le pas de lecture du tableau permet d'atténuer les pics de différence de niveaux ; il est à noter que suivant la vitesse de lecture que vous pouvez préciser par temps%, le timbre du son peut changer complètement de caractéristique si vous prenez une valeur du pas qui vous fasse sauter des parties importantes de segments de droite.

```
PROCEDURE sonsus
  WHILE MOUSEK=0
    EXIT IF susd%=susg%
    gg%=xa%(susg%-1)-x0%-2*dd%+1
    FOR json%=gg% TO xa%(susd%-1)-2*dd%-x0% STEP pas%
      SOUND 1,tabjoue%(json%)*vol%/15,10,2
      EXIT IF MOUSEK=3
      REPEAT
        EXIT IF MOUSEK=3
        INC compteur%
      UNTIL compteur% MOD temps%=0
      EXIT IF MOUSEK=3
    NEXT json%
  WEND
RETURN
PROCEDURE final
  SOUND 1,0,0,0,0
  PUT 0,50,bloncS
RETURN
```

La procédure tabrelease calcule les points du segment de droite correspondant au release de l'enveloppe, ici le dernier segment par défaut (mais libre à vous de détourner cette définition communément admise).

— Vu le nombre réduit de pas de volume de la puce son, nous limitons le nombre maximum de points à lire ainsi que la pente minimum à considérer pour le release (ici −0.25) ;

— D'autre part, dès que le niveau du volume à jouer est nul, on arrête la génération des points à lire dans le tableau tabrelease%() ;

— Le tableau KK%() permet de stocker les valeurs d'indices du tableau tabrelease%() qui correspondent à chaque changement de valeur ; par exemple, si KK%(5)=89, cela voudra dire que tabrelease%(89) et que tabrelease%(88)=6 ; cette définition du tableau KK%() permet de se brancher immédiatement au bon indice pour amorcer la phase release de l'enveloppe et simuler un ADSR complexe en temps réel.

```
PROCEDURE tabrelease
  kk%=0
  pente(nbre%-2)=MIN(pente(nbre%-2),-0.25)
  REPEAT
    tabrelease%(kk%)=y1%+pente(nbre%-2)*kk%
    INC kk%
  UNTIL tabrelease%(kk%-1)<=0
  IF kk%<>maxpermis%
    DEC kk%
    FOR yu%=0 TO kk%
      tabrelease%(yu%)=coeff*tabrelease%(yu%)
      EXIT IF tabrelease%(yu%)=0
    NEXT yu%
    kk%=yu%
    kk%(15)=0
    i%=14
    j%=0
    REPEAT
      REPEAT
        INC j%
      UNTIL tabrelease%(j%)<>tabrelease%(kk%(i%+1))
      stip%=i%-tabrelease%(kk%(i%+1))+tabrelease%(j%)+1
      FOR i1%=i% DOWNTO stip%
        kk%(i1%)=j%
      NEXT i1%
      SUB i%,-tabrelease%(j%)+tabrelease%(kk%(i%+1))
    UNTIL i%<0
  ELSE
    EDIT
  ENDIF
  PRINT AT(2,4);"je vais jouer"
RETURN
```

La procédure sonrelease considère le même pas de lecture de tableau que la procédure son, mais dispose d'un coefficient supplémentaire multiplicateur du temps d'attente.

Il serait toutefois facile de pouvoir disposer d'un pas de lecture independant pour le release en rajoutant une boîte supplémentaire, ou pourquoi pas, un pas spécifique à chacun des segments de droite de l'enveloppe en optant bien sûr pour une présentation différente, vu le nombre important de paramètres qu'il faudrait afficher.

```
PROCEDURE sonrelease(niv%)
  compteur%=0
  FOR i%=kk%(niv%) TO kk%!pas de pas pour le release
    SOUND 1,tabrelease%(i%)*vol%/15,10,2
    REPEAT
      INC compteur%
    UNTIL compteur% MOD rel%*temps%=0
  NEXT i%
RETURN
```

La procédure paramètres réalise l'affichage simple des paramètres de jeu.

```
PROCEDURE parametres
  PRINT AT(11,2);USING "##",vol%
  PRINT AT(19,2);USING "###",temps%
  PRINT AT(28,2);USING "##",rel%
  PRINT AT(36,2);USING "##",pas%
  PRINT AT(44,2);USING "##",susg%
  PRINT AT(51,2);USING "##",susd%
RETURN
```

La procédure sauversons sauvegarde sur disquette les huit sons présentés dans la colonne de gauche ; il faut bien se souvenir que chaque timbre a une longueur de chaîne différente stockée en début de sa chaîne, suivie par les valeurs des paramètres et des tableaux. Les huit chaînes timbres sont concaténées pour former la chaîne transfert$ qui code toute la bibliothèque.



```
PROCEDURE sauversons
  long%=0
  FOR te%=1 TO 8
    long%(te%)=8*ASC(MID$(transfert$(te%),9))+14
    nn%=long%(te%)
    MID$(transfert$,long%+1)=MID$(transfert$(te%),1,nn%)
    ADD long%,long%(te%)
  NEXT te%
  FILESELECT "A:\*.CMI","",nom$
  IF nom$="" OR RIGHT$(nom$)="\"
    GOTO ff1
  ENDIF
  IF INSTR(nom$,".")<>0
    nom$=LEFT$(nom$,INSTR(nom$,"."))+"CMI"
  ELSE
    nom$=nom$+".CMI"
  ENDIF
  BSAVE nom$,VARPTR(transfert$),long%
  ff1:
RETURN
```

Cette procédure fait l'opération inverse, à savoir charger la bibliothèque dans le tableau chaîne de caractères, à partir de la disquette ; en fin de procédure, le premier timbre de la bibliothèque est affiché par défaut.

```
PROCEDURE chargersons
  FILESELECT "A:\*.CMI","",nom$
  IF nom$="" OR RIGHT$(nom$)="\"
    GOTO ff2
  ENDIF
  BLOAD nom$,VARPTR(transfert$)
  traz$=transfert$
  FOR te%=1 TO 8
    long%=8*ASC(MID$(traz$,9))+14
    MID$(transfert$(te%),1,long%)=MID$(traz$,1,long%)
    traz$=MID$(traz$,long%+1)
    PUT dd%+1,y0%+(te%-1)*(y1%+2)/8,nomblanc$
    TEXT dd%+1,y0%-4+te%*(y1%+2)/8,MID$(transfert$(te%),1,8)
  NEXT te%
  PUT 230,45,bloncs
  TEXT 230,60,MID$(nom$,4,INSTR(nom$,".")-4)
  chargertimbre(1)
  ff2:
RETURN
```

C'est la procédure de sauvegarde d'un timbre qui est appelée lorsqu'on clique dans la boîte correspondante sous SAUVER et que l'on a choisi l'option de sauvegarde dans le tableau transfert$().

```
PROCEDURE sauvertimbre(te%)
  MID$(transfert$(te%),9)=CHR$(nbre%)
  MID$(transfert$(te%),10)=CHR$(rel%)
  MID$(transfert$(te%),11)=CHR$(vol%)
  MID$(transfert$(te%),12)=CHR$(pas%)
  MID$(transfert$(te%),13)=CHR$(temps%)
  MID$(transfert$(te%),14)=CHR$(susg%)
  FOR i%=0 TO nbre%-1
    MID$(transfert$(te%),4*i%+15,4)=MKL$(xa%(i%))
    MID$(transfert$(te%),4*i%+15+4*nbre%,4)=MKL$(ya%(i%))
  NEXT i%
RETURN
```

La procédure chargertimbre réalise l'affichage de l'enveloppe du timbre sélectionné à partir du tableau transfer$(), ainsi que l'affectation et l'affichage des valeurs des paramètres.

```
PROCEDURE chargertimbre(te%)
  nbre%=ASC(MID$(transfert$(te%),9))
  rel%=ASC(MID$(transfert$(te%),10))
  vol%=ASC(MID$(transfert$(te%),11))
  pas%=ASC(MID$(transfert$(te%),12))
  temps%=ASC(MID$(transfert$(te%),13))
  susg%=ASC(MID$(transfert$(te%),14))
  FOR i%=0 TO nbre%-1
    xa%(i%)=CVL(MID$(transfert$(te%),4*i%+15,4))
    ya%(i%)=CVL(MID$(transfert$(te%),4*i%+15+4*nbre%,4))
  NEXT i%
  PUT 230,70,nomblanc$
  TEXT 230,80,MID$(transfert$(te%),1,8)
  COLOR 0
  BOX dd%+1,y0%+(sr%-1)*(y1%+2)/8,x0%-2,y0%-2+sr%*(y1%+2)/8
  COLOR 1
  BOX dd%+1,y0%+(te%-1)*(y1%+2)/8,x0%-2,y0%-2+te%*(y1%+2)/8
  sr%=te%
  parametres
  enveloppe
RETURN
```

6/1.5

Générateur d'accompagnement automatique

But du programme

Ce programme se propose de réaliser un schéma simple d'accompagnement automatique à partir d'une grille d'accords. Dans un premier temps, les accords choisis ne dépasseront pas quatre sons, puisque ce programme devra fonctionner sur le générateur sonore de l'Atari, de polyphonie limitée à trois voix, aussi bien que sur le système MIDI ;

Les fonctions essentielles que doit assurer ce programme sont les suivantes :

— initialisation de grille d'accords, sauvegarde et chargement ;

— remplissage et modification d'une grille ;

— calcul d'un accompagnement automatique ;

— réalisation sonore de cet accompagnement suivant certains paramètres de style, cycle, tempo et affectation de sortie audio.

Les accompagnements se feront sous forme d'arpèges pouvant être générés sur un schéma allant de un à quatre temps, mais il est facile avec le principe d'analyse exposé, d'étendre ces possibilités de génération à un nombre supérieur de temps. De façon classique, les choix de fonctions et des valeurs de paramètres se feront à l'aide du clavier Atari et de la souris.

Le programme en général

La première partie du programme, jusqu'à l'instruction DO LOOP, se charge d'initialiser les tableaux et les affichages de choix de fonctions, de grille initiale et de valeurs de paramètres de jeu. La première partie du traitement de la boucle DO LOOP teste la lecture du clavier Atari, par BIOS (1, 2), et permet de remplir une grille d'accords en notation américaine.

La seconde partie du traitement teste l'état de la souris par BIOS(1, 3) et permet de réaliser plusieurs fonctions :

— choisir l'accord à positionner dans la grille ;

— fixer des valeurs de tempo, cycle, affectation de sortie audio, numéro de canal MIDI, nombre de répétitions et style d'accompagnement ;

— appeler les fonctions de sauvegarde, chargement et initialisation de grille d'accords.

LE PROGRAMME EN DÉTAIL

Initialisation

Les variables de jeu

La variable **transfert$** est utilisée pour les opérations de chargement et sauvegarde sur disquette. La variable **joue$** est la chaîne de caractères qui contiendra les données à jouer ; elle est initialisée par défaut sur une longueur de 500 caractères, mais pensez à augmenter sa taille si vous désirez étendre la taille des grilles d'accords sur lesquelles générer l'accompagnement automatique ainsi que les possibilités rythmiques des arpèges ; le choix de taille maximum de grille est ici de huit mesures et la décomposition maximum du temps est la double croche, pour une pulsation à la noire, ce qui donne un nombre maximum de notes de 8 fois seize, soit 128 notes : (par défaut, la métrique considérée ici est de quatre unités de temps). Chaque note est codée sur trois octets qui correspondent respectivement à la durée, hauteur et vélocité de la note ; dans un choix d'accompagnement monotone ne comptant que des doubles croches, le nombre maximum d'octets à considérer dans la chaîne à jouer serait donc de 3 fois 128, soit 384 ;

Les variables **cycle|**, **style|**, **nbrerep&**, **tempo&**, **pause$** contiennent les codages des indicateurs de mode cycle ON ou OFF, choix de style d'accompagnement, nombre de répétitions de la grille si le cycle est sur la position ON, ainsi que la valeur du tempo à la précision du 1/200e de seconde, (moins le temps de traitement de l'instruction en Basic GFA) ainsi que le codage de la chaîne **pause$** correspondant à un soupir, (note de durée égale au temps de hauteur nulle et de vélocité égale à zéro).

Les tableaux

Les tableaux **sortie$( )** et **cycle$( )** contiennent les options de choix de sortie audio, MIDI ou Générateur Sonore et de positionnement de cycle ON ou OFF.

Le tableau **tonique$( )** contient les noms des notes toniques des accords à sélectionner ; choix est fait de la notation américaine et des dièses mais si cela froisse votre esprit, il suffit de rajouter une option de notation et de choix d'altération qui vous fera opter pour la lecture d'un autre tableau ; vous pouvez utiliser un tableau à deux dimensions dont le premier indice correspond par exemple à la valeur de votre drapeau de test de choix de notation et le second jouera le même rôle que l'indice courant dans le cas actuel.

Le tableau **type$( )** contient les types principaux d'accords de trois et quatre sons ; les choix d'appellation et de codage ne sont présentés ici que pour l'état fondamental de l'accord, mais si vous désirez pouvoir choisir le ren-



versement, reportez-vous au programme d'Analyse d'Accords qui vous présente tous les renversements des accords de trois à sept sons ; ce tableau contient aussi le silence. Le tableau **interv$( )** code les intervalles successifs des accords répertoriés dans le tableau **type$( )**.

Le tableau **style$( )** contient quatre styles d'accompagnement proposés mais n'est évidemment pas exhaustif ; vous aurez intérêt, en en créant d'autres, à grouper les découpages rythmiques en petits schémas qui peuvent s'enchaîner les uns aux autres dans n'importe quel ordre, ce qui vous permettra de personnaliser votre accompagnement au moyen de l'ordinateur.

Les variables d'affichage

La variable temps | correspond aux nombres de temps de la mesure et pourra servir à afficher en gras une barre de mesure, et calculer le nombre total de temps de la grille, donc de cases à remplir de symboles, (nombre de temps multiplié par nombre de mesures).

Les variables **xdep|**, **ydep|**, **grillarg|**, **grilhaut|**, **curs|**, **cursx|** et **ecart|** servent à positionner les coordonnées de départ et la taille d'affichage de la boîte correspondant au premier temps de la grille. Les trois dernières variables servent respectivement à positionner le curseur sous la boîte qui correspond au temps d'affichage, sans en effacer le trait inférieur, et positionner un écart dans le cas d'une grille de huit mesures puisque chaque ligne ne peut présenter qu'un maximum de quatre mesures pour des raisons de clarté d'affichage.

Les variables **bl$** et **curs$** représentent l'espace et le curseur et permettent d'accélérer l'affichage du curseur lorsqu'il se déplace sur une position de grille.

La variable **eff$** sert à effacer la partie d'écran où on désire afficher ou charger une nouvelle grille d'accords.

La procédure **new1** initialise un choix de taille de grille.

L'affichage de la grille correspondant au choix précédent et le positionnement du curseur sur le premier temps s'effectuent ensuite au moyen de la procédure grille.

Les instructions qui suivent réalisent l'affichage initial sur le choix d'accord de CM, (DO majeur) MIDI OUT sur le canal 1, mode cycle ON avec deux répétitions, style d'accompagnement folk et tempo à 120 à la noire. Les autres affichages positionnent simplement les options classiques charger, sauver et new à cliquer avec la souris pour activer. Le running statut en NOTE ON est activé sur le canal 1.

Traitement général

Test du clavier Atari

Par l'instruction BIOS(1, 2), on teste si on a frappé sur une touche du clavier. Les seules touches qui nous intéressent sont les touches curseur haut (code 200), bas (code 208), gauche (code 203) et droit (code 205) dont les fonctions sont les suivantes :

— curseur haut : écrit le nom d'accord sélectionné dans la case temps correspondant à la position du curseur ;

— curseur bas : efface la case temps correspondant à la position du curseur ;

— curseur droit : déplace le curseur d'une position vers la droite ;

— curseur gauche : déplace le curseur d'une position vers la gauche.

La variable **point|** contient le temps qui correspond à la position du curseur, ceci depuis le début de la grille ; sur une grille de quatre mesures, cette variable doit donc être comprise entre 0 et 15. De façon générale, la valeur maximum permise est égale à **tempstot| −1**. Les variables **cursxanc&** et **cursyanc&** sont recalculées sur une base **MOD 16** pour la première (4 mesures de 4 temps au maximum par ligne) et **DIV 16** pour la seconde (déplacement vertical d'une valeur égale à **ecart|** lorsqu'on écrit une nouvelle ligne) ; notez que si vous désirez introduire une métrique à trois temps et respecter la même présentation, il faudra procéder à une rectification par **MOD 12** et **DIV 12** ; de façon plus générale ce serait **MOD (4*temps|)** et **DIV(4*temps|)** qu'il faudrait utiliser en effectuant tout de même un test pour ne pas descendre au-dessous de l'écran.

Dans le cas où l'option curseur haut a été sélectionnée, l'accord ne sera écrit que si le silence n'a pas été choisi dans le type de l'accord ; ce dernier a en effet priorité sur un type quelconque d'accord. L'élément numéro **point|** du tableau **accomp$( )** est rempli avec le codage de l'accord sélectionné en note fondamentale puis intervalles le définissant, et la variable **acc$** est initialisée avec la dénomination de l'accord et affichée sur la position courante du curseur. Le silence est codé par le caractère A dans le tableau **accomp$( )**. Il sera aisé par la suite de savoir si une case contient un silence, (chaîne de longueur 1), rien, (chaîne vide), ou un accord, (chaîne de longueur supérieure à 1).

Test de lecture de la souris

La lecture de la souris incrémente sur clic gauche et décrémente de façon classique sur clic droit les valeurs de tous les paramètres de jeu et appelle sur un clic à gauche les fonctions sauver, charger et new. L'affichage du tempo réel se fait à l'aide de l'instruction **12000 DIV tempo&**, le principe de durée de chaque événement sonore étant du type boucle d'attente jusqu'à ce que la valeur **tempo&** ou un de ses diviseurs soit atteinte ; tempo&=200 correspond à un temps d'attente de une seconde, donc à un tempo de soixante à la noire. Dans l'exemple proposé, les valeurs des durées des événements musicaux sont codées sur un octet et ne peuvent dépasser la valeur 255, mais vous pouvez générer des durées quelconques en codant cette valeur sur deux octets. La valeur minimum de la variable tempo& est ici de 40 pour éventuellement générer des durées de double-croches, (durées 10), sans que le Basic GFA devienne trop imprécis à un tempo de 300 à la noire, ce qui est plus rapide que les tempi conventionnels.

Les procédures

Le tableau **accomp$( )** contient les codages des symboles d'accords ou de silence à jouer dans le style sélectionné. Notez que si une case ne contient aucun symbole, elle sera supposée représenter le dernier symbole précé-

demment écrit, un peu à la manière du running statut dans le système MIDI. Tant qu'on n'écrit pas de nouveau symbole, c'est le symbole précédent qui est validé.

Par exemple l'écriture C+,,,, est équivalente à C+, C+, C+, C+. Le programme présente deux traitements distincts du fait que le jeu en accords s'effectue en accords plaqués dans le cas du MIDI et en accords arpégés de schéma fixe dans le cas du générateur sonore ; le traitement sera fait dans ce cas en temps réel alors que dans les autres choix de styles, il est souhaitable de pouvoir analyser si un accord dure un, deux, trois ou quatre temps afin de générer des arpèges qui ne soient pas simplement la répétition d'un arpège sur un temps. Pour l'instant, ces autres arpèges correspondent à des procédures qui existent dans leur fonction mais qui seront remplacées dans une version ultérieure.

Procédure jouer

C'est simplement un branchement de traitement sur la procédure accords ou la procédure ana qui va analyser les durées des accords à jouer pour les trois autres choix de style.

Procédure accords

La variable **ring$** contient l'état courant du symbole à analyser. Elle est initialisée en chaîne vide. Le tableau **accomp$( )** est analysé élément par élément du début à la fin. La variable logique **sil!** est vraie si l'état courant du symbole est un silence, codé par le caractère « A ». Dans le cas où le symbole analysé du tableau est la chaîne vide, on se reporte à l'état courant contenu dans ring$. C'est le cas traité dans la seconde partie du traitement conditionnel, intérieur au second ELSE.

Les procédures **accompacc** et silence généreront en temps réel l'événement sonore qui est codé dans **ring$**.

La boucle terminale sert à générer éventuellement les répétitions dans le mode cycle. Si vous voulez accélérer le traitement, testez l'instruction **DEC nbre&** avant de remplacer éventuellement nbre& = nbre& – 1, car suivant la version de Basic GFA que vous avez, vous risquez d'avoir des surprises.

Procédure ana

Cette procédure reprend les grandes lignes de la procédure accords mais analyse en plus la durée des accords jusqu'à un maximum de 4. On retrouve donc les variables **ring$** et **sil!** qui fonctionnent comme précédemment ; la variable **longacc|** contient en permanence la durée de l'accord courant dont on veut générer l'arpège. Cette variable a une valeur maximum de 4 et est remise à zéro dès qu'elle atteint cette valeur ; **jouefolk|** est le pointeur qui compte les accords analysés ; lorsqu'il atteindra la valeur **tempstot|**, on pourra faire jouer la suite d'arpèges générée. Le pointeur **pointjoue&** permet de préciser la position où on doit écrire la donnée musicale que l'on va générer ; chaque fois qu'on insère une donnée, cette position sera incrémentée de trois puisque le codage se fait sur trois octets. N'oubliez pas d'incrémenter de 4 cette valeur si vous désirez introduire des durées supérieures à la

noire. Ces variables sont initialisées en début de procédure puis on se branchera sur la procédure de génération d'arpèges **joueana** dès que **longacc|** =4, ou que le nouvel élément analysé du tableau **accomp$( )** est différent de l'état courant stocké dans **ring$**, ou qu'on analyse le dernier élément du tableau ; on se branche de façon équivalente sur la procédure **anasilence** lorsqu'on doit générer du silence, avec la différence essentielle de ne pas considérer la longueur d'un silence pour générer des arpèges différents si arpèges de silence il y avait !

La variable **notdep|** contient le code de la fondamentale de l'accord analysé puisque les notes suivantes s'en déduisent automatiquement avec le type de l'accord. Le code 60 correspond à DO et à la première note de la troisième octave du générateur sonore.

Procédure accompacc

C'est la procédure de sortie lorsque l'on a sélectionné les accords comme style d'accompagnement. Cette procédure effectue deux traitements différents suivant l'option de sortie choisie :

— option MIDI :

on envoie les NOTE ON de toutes les notes de l'accord courant contenu dans **ring$** avec une vélocité égale à 90 puis on attend le temps tempo&, par la procédure silence avant d'envoyer les NOTE OFF correspondants.

— option générateur sonore :

On ne peut jouer les accords plaqués à cause de la limitation de la polyphonie à trois notes ; on choisit donc d'arpéger de façon régulière les notes de l'accord. Un accord de quatre sons, **(long| = 4)**, sera arpégé de façon montante et un accord de trois sons redoublera sa tierce en fin d'arpège. La décomposition se fait en doubles croches dont la durée est **tempo& DIV 4**.

Notez que dans le cas du générateur sonore on est obligé d'arrêter le son avant la fin d'une note pour réentendre éventuellement cette même note si elle est jouée immédiatement après. Le temps à déduire est obligatoirement supérieur à 4/200 de seconde soit 1/50 de seconde puisque c'est la fréquence de lecture de base du générateur sonore ; ici la valeur d'attente de 5/200 de seconde alors que l'arrêt du son par SOUND 1, 0, 0, 0 ne dure que 4/200 de seconde prend en compte le temps des instructions qui nuirait à la régularité du déroulement en pas à pas.

Procédure joueana

Cette procédure est un simple branchement sur les différentes générations d'arpèges suivant la longueur des accords analysés. Pour chaque accord analysé, on incrémente le pointeur d'analyse de la valeur égale à la longueur de l'accord. Lorsque l'analyse est terminée, **(jouefolk| = tempstot|)**, on réalise la sortie sonore par la procédure **play**.

Procédure anasilence

Cette procédure introduit dans la chaîne à jouer le codage d'un silence.



Procédures ana1, ana2, ana3, ana4

Ce sont les procédures de branchement sur les générations d'arpèges suivant la durée de l'accord et le style sélectionné.

Procédure motif (mottemps|)

C'est la procédure qui insère dans la chaîne **joue$** une hauteur de tierce, quinte ou septième de l'accord codé dans **ring$** par un choix équiprobable. La durée de la note insérée correspond au paramètre **mottemps|**. Sa vélocité est égale à 80.

Procédure motdep(depart&,tr&)

C'est la procédure qui insère dans la chaîne **joue$** une note de durée **depart&**, de hauteur **notdep| + tr&** et de vélocité 80.

Procédure play

C'est la procédure qui décode et envoie vers le MIDI OUT ou le générateur sonore les octets de la chaîne joue$ en les lisant trois par trois en se rappelant elle-même nbrerep& de fois si le mode cycle est positionné sur ON.

Procédures jouefolk1, jouedisco1,jouelatin1

Ces procédures présentent un certain nombre d'arpèges dont le choix se fait de façon aléatoire par la fonction RANDOM afin d'éviter trop de monotonie. **Tempo&** correspond à une noire, sa division par deux correspond à une croche et par quatre à une double croche.

Procédures jouefolk2, jouedisco2,jouelatin2
Procédures jouefolk3, jouedisco3,jouelatin3
Procédures jouefolk4, jouedisco4,jouelatin4

Ces neuf procédures doivent générer des arpèges correspondant à des durées d'accords de deux, trois et quatre temps. Pour l'instant elles ne font qu'appeler deux trois et quatre fois les procédures de rang 1, mais seront implémentées ultérieurement.

Procédure sauver

C'est la procédure qui sauvegarde sur disquette la grille d'accords ; en début de chaîne sont codés le nombre d'octets de la chaîne et le nombre de temps de la grille. Avant de coder un accord, on code sur le premier octet disponible la longueur de chaque accord pour pouvoir récupérer ensuite tous les éléments de l'accord au cours de la procédure de chargement.

Procédure charger

C'est la procédure de chargement et affichage d'une chaîne sur disquette. Après avoir lus longueur de chaîne et taille de la grille, il suffit ensuite de décoder la chaîne accord par accord en se référant au premier octet qui donne le nombre d'éléments de l'accord qui suit ; on efface ensuite la grille précédente et on dessine la grille actuelle par la procédure grille puisqu'on a récupéré la valeur de **tempstot|** dans la chaîne chargée. Il suffit ensuite

de retrouver les notations des accords codés pas à pas et d'en faire l'affichage en déplaçant le curseur comme au début du programme.

Procédure grille

C'est la procédure qui affiche des boîtes pour les **tempstot**| temps de la grille ; cet affichage se fait à raison de quatre mesures maximum par ligne, et avec une génération d'un écart en cas de deuxième ligne ; tous les **temps** | **temps** une ligne verticale d'un pixel supplémentaire est tracée afin de repérer le début de chaque mesure. A la fin de ce traitement, le curseur est positionné en début de première mesure.

Procédures new et new1

Elles initialisent chacune une nouvelle grille, mais **new1** le fait en début de programme et n'a pas besoin d'effacer éventuellement une grille déjà existante.



```
transfert$=STRING$(1000,254)
joue$=SPACE$(500)
t%=TIMER
cycle|=0
style|=1
nbrerep&=2
tempo&=100
pause$=CHR$(tempo&)+CHR$(0)+CHR$(0)
DEFTEXT 1,0,0,13
DIM tonique$(11),type$(7),accomp$(31)
DIM interv$(6),style$(3),cycle$(1)
DIM sortie$(1)
sortie$(0)="midi"
sortie$(1)="puce"
FOR i|=0 TO 11
  READ tonique$(i|)
NEXT i|
FOR i|=0 TO 7
  READ type$(i|)
NEXT i|
FOR i|=0 TO 3
  READ style$(i|)
NEXT i|
cycle$(0)=" on"
cycle$(1)="off"
DATA C,C#,D,D#,E,F,F#,G,G#,A,A#,B
DATA M,m,dm,+,s4,7,m7,sil
DATA accords,folk,disco,latin
interv$(0)=CHR$(4)+CHR$(7)
interv$(1)=CHR$(3)+CHR$(7)
interv$(2)=CHR$(3)+CHR$(6)
interv$(3)=CHR$(4)+CHR$(8)
interv$(4)=CHR$(5)+CHR$(7)
interv$(5)=CHR$(4)+CHR$(7)+CHR$(10)
interv$(6)=CHR$(3)+CHR$(7)+CHR$(10)
temps|=4
xdep|=5
ydep|=100
grillarg|=38
grilhaut|=50
curs|=20
cursx|=5
ecart|=40
bl$=CHR$(32)
curs$=CHR$(45)
GET 0,100,630,300,eff$
@new1
```

(suite)

```
acc$=tonique$(ton|)+type$(typ|)
TEXT 5,10,tonique$(ton|)
TEXT 30,10,type$(typ|)
TEXT 60,10,"jouer"
TEXT 110,10,"out"
TEXT 110,30,sortie$(sortie|)
TEXT 150,30,1
TEXT 160,10,"sauver"
TEXT 220,10,"new"
TEXT 260,10,"charger"
TEXT 360,10,style$(style|)
TEXT 460,10,"cycle"
TEXT 460,30,cycle$(cycle|)
TEXT 520,10,"rep"
TEXT 520,30,nbrerep&
TEXT 560,10,"tempo"
TEXT 560,30,12000 DIV tempo&
OUT 3,144
ON BREAK GOSUB fin
DO
  MOUSE x&,y&,k|
  IF BIOS(1,2)
    clavier|=INP(2)
    IF clavier|=205
      TEXT cursxanc&,cursyanc&,bl$
      INC point|
      point|=MIN(point|,tempstot|-1)
      cursxanc&=xdep|+cursx|+(point| MOD 16)*grillarg|
      yanc&=ydep|+curs|+ecart|*(point| DIV 16)
      cursyanc&=yanc&+grilhaut|*(1+point| DIV 16)
      TEXT cursxanc&,cursyanc&,curs$
    ENDIF
    IF clavier|=203
      TEXT cursxanc&,cursyanc&,bl$
      point|=MAX(point|,1)
      DEC point|
      cursxanc&=xdep|+cursx|+(point| MOD 16)*grillarg|
      yanc&=ydep|+curs|+ecart|*(point| DIV 16)
      cursyanc&=yanc&+grilhaut|*(1+point| DIV 16)
      TEXT cursxanc&,cursyanc&,curs$
    ENDIF
```



(suite)

```
    IF clavier|=200
      IF typ|<7
        acc$=tonique$(ton|)+type$(typ|)
        accomp$(point|)=CHR$(ton|+60)+interv$(typ|)
      ELSE
        acc$=type$(typ|)
        accomp$(point|)="A"
      ENDIF
      TEXT  cursxanc&,cursyanc&-2*curs|,"      "
      TEXT cursxanc&,cursyanc&-2*curs|,acc$
    ENDIF
    IF clavier|=208
      accomp$(point|)=""
      TEXT  cursxanc&,cursyanc&-2*curs|,"      "
    ENDIF
  ENDIF
  IF k|=1
    IF x&<18 AND y&<20
      INC ton|
      ton|=MIN(ton|,11)
      TEXT 5,10,"  "
      TEXT 5,10,tonique$(ton|)
      PAUSE 5
    ENDIF
    IF x&>22 AND x&<50 AND y&<20
      INC typ|
      typ|=MIN(typ|,7)
      TEXT 30,10,"      "
      TEXT 30,10,type$(typ|)
      PAUSE 5
    ENDIF
    IF x&>70 AND x&<100 AND y&<10
      @jouer
    ENDIF
    IF x&>110 AND x&<130 AND y&>10 AND y&<30
      INC sortie|
      sortie|=MIN(sortie|,1)
      TEXT 110,30,"      "
      TEXT 110,30,sortie$(sortie|)
      OUT 3,143+midi|
      PAUSE 5
    ENDIF
```

(suite)

```
IF x&>150 AND x&<170 AND y&>10 AND y&<30
   INC midi|
   midi|=MIN(midi|,16)
   TEXT 150,30,"  "
   TEXT 150,30,midi|
   OUT 3,143+midi|
   PAUSE 5
ENDIF
IF x&>160 AND x&<210 AND y&<10
   @sauver
ENDIF
IF x&>220 AND x&<240 AND y&<10
   @new
ENDIF
IF x&>260 AND x&<320 AND y&<10
   @charger
ENDIF
IF x&>360 AND x&<420 AND y&<10
   INC style|
   style|=MIN(style|,3)
   TEXT 360,10,"              "
   TEXT 360,10,style$(style|)
   PAUSE 5
ENDIF
IF x&>460 AND x&<480 AND y&>20 AND y&<30
   INC cycle|
   cycle|=MIN(cycle|,1)
   TEXT 460,30,"     "
   TEXT 460,30,cycle$(cycle|)
   PAUSE 5
ENDIF
IF x&>520 AND x&<540 AND y&>20 AND y&<30
   INC nbrerep&
   TEXT 520,30,"     "
   TEXT 520,30,nbrerep&
   PAUSE 5
ENDIF
```



(suite)

```
    IF x&>560 AND x&<580 AND y&>20 AND y&<30
      DEC tempo&
      tempo&=MAX(tempo&,40)
      pause$=CHR$(tempo&)+CHR$(0)+CHR$(0)
      TEXT 560,30,"      "
      TEXT 560,30,12000 DIV tempo&
      PAUSE 2
    ENDIF
  ENDIF
  IF k|=2
    IF x&<18 AND y&<20
      ton|=MAX(ton|,1)
      DEC ton|
      TEXT 5,10,"  "
      TEXT 5,10,tonique$(ton|)
      PAUSE 5
    ENDIF
    IF x&>22 AND x&<50 AND y&<20
      typ|=MAX(typ|,1)
      DEC typ|
      TEXT 30,10,"      "
      TEXT 30,10,type$(typ|)
      PAUSE 5
    ENDIF
    IF x&>110 AND x&<130 AND y&>10 AND y&<30
      sortie|=MAX(sortie|,1)
      DEC sortie|
      TEXT 110,30,"      "
      TEXT 110,30,sortie$(sortie|)
      OUT 3,143+midi|
      PAUSE 5
    ENDIF
    IF x&>150 AND x&<170 AND y&>10 AND y&<30
      midi|=MAX(midi|,2)
      DEC midi|
      TEXT 150,30,"  "
      TEXT 150,30,midi|
      OUT 3,143+midi|
      PAUSE 5
    ENDIF
```

(suite)

```
      IF x&>360 AND x&<420 AND y&<10
        style|=MAX(style|,1)
        DEC style|
        TEXT 360,10,"              "
        TEXT 360,10,style$(style|)
        PAUSE 5
      ENDIF
      IF x&>460 AND x&<480 AND y&<30 AND y&>20
        cycle|=MAX(cycle|,1)
        DEC cycle|
        TEXT 460,30,"     "
        TEXT 460,30,cycle$(cycle|)
        PAUSE 5
      ENDIF
      IF x&>520 AND x&<540 AND y&>20 AND y&<30
        DEC nbrerep&
        nbrerep&=MAX(nbrerep&,1)
        TEXT 520,30,"     "
        TEXT 520,30,nbrerep&
        PAUSE 5
      ENDIF
      IF x&>560 AND x&<580 AND y&>20 AND y&<30
        INC tempo&
        tempo&=MIN(tempo&,254)
        pause$=CHR$(tempo&)+CHR$(0)+CHR$(0)
        TEXT 560,30,"     "
        TEXT 560,30,12000 DIV tempo&
        PAUSE 2
      ENDIF
    ENDIF
  LOOP
  PROCEDURE jouer
    nbre&=nbrerep&
    SELECT style|
    CASE 0
      @accords
    CASE 1 TO 3
      @ana
    ENDSELECT
  RETURN
```



(suite)

```
PROCEDURE accords
  ring$=""
  FOR il=0 TO tempstotl-1
    IF LEN(accomp$(il))>1
      sil!=FALSE
      ring$=accomp$(il)
      notdepl=ASC(accomp$(il))
      @accompacc
    ELSE
      IF accomp$(il)="A"
        @silence
        sil!=TRUE
      ELSE
        IF sil!=FALSE
          @accompacc
        ELSE
          @silence
        ENDIF
      ENDIF
    ENDIF
  NEXT il
  IF cyclel=0
    WHILE nbre&>1
      nbre&=nbre&-1
      @accords
    WEND
  ENDIF
RETURN
PROCEDURE ana
  jouefolk!=0
  ring$=""
  pointjoue&=1
  longaccl=0
  sil!=TRUE
```

(suite)

```
    FOR il=0 TO tempstotl-1
      IF LEN(accomp$(il))>1
        sil!=FALSE
        IF ring$<>accomp$(il)
          IF ring$<>""
            @joueana
          ENDIF
          ring$=accomp$(il)
          notdepl=ASC(accomp$(il))
          longaccl=1
          IF il=tempstotl-1
            @joueana
          ENDIF
        ELSE
          INC longaccl
          IF longaccl=4 OR il=tempstotl-1
            @joueana
            longaccl=0
          ENDIF
        ENDIF
      ELSE
        IF accomp$(il)="A"
          IF ring$<>""
            @joueana
            longaccl=0
            ring$=""
          ENDIF
          @anasilence
          sil!=TRUE
        ELSE
          IF sil!=FALSE
            INC longaccl
            IF longaccl=4 OR il=tempstotl-1
              @joueana
              longaccl=0
            ENDIF
          ELSE
            @anasilence
          ENDIF
        ENDIF
      ENDIF
    NEXT il
  RETURN
```



(suite)

```
PROCEDURE accompacc
  IF sortie|=0
    OUT 3,notdep|,90
    FOR j|=2 TO LEN(ring$)
      OUT 3,ASC(MID$(ring$,j|,1))+notdep|,90
    NEXT j|
    @silence
    OUT 3,notdep|,0
    FOR j|=2 TO LEN(ring$)
      OUT 3,ASC(MID$(ring$,j|,1))+notdep|,0
    NEXT j|
  ELSE
    long|=LEN(ring$)
    depart|=notdep|-59
    SOUND 1,15,depart|,3
    t%=TIMER
    REPEAT
    UNTIL TIMER-t%>=(tempo& DIV 4)-5
    SOUND 1,0,0,0
    t%=TIMER
    REPEAT
    UNTIL TIMER-t%>=4
    FOR j|=2 TO 3
      SOUND 1,15,depart|+ASC(MID$(ring$,j|,1)),3
      t%=TIMER
      REPEAT
      UNTIL TIMER-t%>=(tempo& DIV 4)-5
      SOUND 1,0,0,0
      t%=TIMER
      REPEAT
      UNTIL TIMER-t%>=4
    NEXT j|
    IF long|=4
      SOUND 1,15,depart|+ASC(MID$(ring$,4,1)),3
      t%=TIMER
      REPEAT
      UNTIL TIMER-t%>=(tempo& DIV 4)-5
      SOUND 1,0,0,0
      t%=TIMER
      REPEAT
      UNTIL TIMER-t%>=4
```

(suite)

```
      ELSE
        SOUND 1,15,depart|+ASC(MID$(ring$,2,1)),3
        t%=TIMER
        REPEAT
        UNTIL TIMER-t%>=(tempo& DIV 4)-5
        SOUND 1,0,0,0
        t%=TIMER
        REPEAT
        UNTIL TIMER-t%>=4
      ENDIF
    ENDIF
  RETURN
  PROCEDURE silence
    t%=TIMER
    REPEAT
    UNTIL TIMER-t%>=tempo&
  RETURN
  PROCEDURE fin
    OUT 3,123,0
    SOUND 1,0,0,0
    EDIT
  RETURN
  PROCEDURE joueana
    SELECT longacc|
    CASE 1
      @ana1
    CASE 2
      @ana2
    CASE 3
      @ana3
    CASE 4
      @ana4
    ENDSELECT
    ADD jouefolk|,longacc|
    IF jouefolk|=tempstot|
      @play
    ENDIF
  RETURN
  PROCEDURE anasilence
    MID$(joue$,pointjoue&,3)=pause$
    ADD pointjoue&,3
    INC jouefolk|
    IF jouefolk|=tempstot|
      @play
    ENDIF
  RETURN
```



(suite)

```
PROCEDURE ana1
  SELECT style|
  CASE 1
    @jouefolk1
  CASE 2
    @jouedisco1
  CASE 3
    @jouelatin1
  ENDSELECT
RETURN
PROCEDURE ana2
  SELECT style|
  CASE 1
    @jouefolk2
  CASE 2
    @jouedisco2
  CASE 3
    @jouelatin2
  ENDSELECT
RETURN
PROCEDURE ana3
  SELECT style|
  CASE 1
    @jouefolk3
  CASE 2
    @jouedisco3
  CASE 3
    @jouelatin3
  ENDSELECT
RETURN
PROCEDURE ana4
  SELECT style|
  CASE 1
    @jouefolk4
  CASE 2
    @jouedisco4
  CASE 3
    @jouelatin4
  ENDSELECT
RETURN
PROCEDURE motif(mottempsl)
  hautaleal=ASC(MID$(ring$,2+RANDOM(LEN(ring$)-1),1))
  hauteur$=CHR$(notdepl+hautaleal)
  motif$=CHR$(mottempsl)+hauteur$+CHR$(80)
  MID$(joue$,pointjoue&,3)=motif$
  ADD pointjoue&,3
RETURN
```

(suite)

```
PROCEDURE motdep(depart&,tr&)
  motif$=CHR$(depart&)+CHR$(notdep|+tr&)+CHR$(80)
  MID$(joue$,pointjoue&,3)=motif$
  ADD pointjoue&,3
RETURN
PROCEDURE play
  IF sortie|=0
    FOR ii&=1 TO pointjoue&-1 STEP 3
      OUT 3,ASC(MID$(joue$,ii&+1,1)),ASC(MID$(joue$,ii&+2,1))
      arret|=ASC(MID$(joue$,ii&,1))
      t%=TIMER
      REPEAT
      UNTIL TIMER-t%>=arret|
      OUT 3,ASC(MID$(joue$,ii&+1,1)),0
    NEXT ii&
  ELSE
    FOR ii&=1 TO pointjoue&-1 STEP 3
      SOUND 1,15,ASC(MID$(joue$,ii&+1,1))-59,3
      arret|=ASC(MID$(joue$,ii&,1))
      t%=TIMER
      REPEAT
      UNTIL TIMER-t%>=arret|*5
      t%=TIMER
      SOUND 1,0,0,0
      REPEAT
      UNTIL TIMER-t%>=4
    NEXT ii&
  ENDIF
  IF cycle|=0
    WHILE nbre&>1
      nbre&=nbre&-1
      @play
    WEND
  ENDIF
RETURN
```

(suite)

```
PROCEDURE jouefolk1
  choix1|=RANDOM(6)
  SELECT choix1|
  CASE 0
    @motdep(tempo& DIV 2,0)
    @motif(tempo& DIV 4)
    @motif(tempo& DIV 4)
  CASE 1
    @motdep(tempo& DIV 4,0)
    @motif(tempo& DIV 4)
    @motif(tempo& DIV 2)
  CASE 2
    @motdep(tempo& DIV 4,0)
    @motif(tempo& DIV 4)
    @motif(tempo& DIV 4)
    @motif(tempo& DIV 4)
  CASE 3
    @motdep(tempo& DIV 4,0)
    @motdep(tempo& DIV 4,2)
    @motif(tempo& DIV 4)
    @motif(tempo& DIV 4)
  CASE 4
    @motdep(tempo& DIV 4,0)
    @motdep(tempo& DIV 4,9)
    @motif(tempo& DIV 4)
    @motif(tempo& DIV 4)
  CASE 5
    @motdep(tempo& DIV 4,0)
    @motif(tempo& DIV 4)
    @motif(tempo& DIV 4)
    @motdep(tempo& DIV 4,12)
  ENDSELECT
RETURN
PROCEDURE jouefolk2
  @jouefolk1
  @jouefolk1
RETURN
PROCEDURE jouefolk3
  @jouefolk1
  @jouefolk1
  @jouefolk1
RETURN
```

(suite)

```
PROCEDURE jouefolk4
  @jouefolk1
  @jouefolk1
  @jouefolk1
  @jouefolk1
RETURN
PROCEDURE jouedisco1
  SELECT RANDOM(5)
  CASE 0
    @motdep(tempo& DIV 2,0)
    @motdep(tempo& DIV 2,12)
  CASE 1
    @motdep(tempo& DIV 2,0)
    @motdep(tempo& DIV 4,12)
    @motdep(tempo& DIV 4,12)
  CASE 2
    @motdep(tempo& DIV 2,0)
    @motdep(tempo& DIV 4,12)
    @motif(tempo& DIV 4)
  CASE 3
    @motdep(tempo& DIV 4,0)
    @motdep(tempo& DIV 4,0)
    @motdep(tempo& DIV 4,12)
    @motdep(tempo& DIV 4,12)
  CASE 4
    @motdep(tempo&,0)
  ENDSELECT
RETURN
PROCEDURE jouedisco2
  @jouedisco1
  @jouedisco1
RETURN
PROCEDURE jouedisco3
  @jouedisco1
  @jouedisco1
  @jouedisco1
RETURN
PROCEDURE jouedisco4
  @jouedisco1
  @jouedisco1
  @jouedisco1
  @jouedisco1
RETURN
```



(suite)

```
PROCEDURE jouelatin1
  SELECT RANDOM(4)
  CASE 0
    @motdep(3*tempo& DIV 4,0)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  CASE 1
    @motdep(3*tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 4,0)
  CASE 2
    @motdep(tempo& DIV 4,0)
    @motif(tempo& DIV 2)
    @motif(tempo& DIV 4)
  CASE 3
    @motdep(tempo& DIV 4,0)
    @motdep(tempo& DIV 2,0)
    @motdep(tempo& DIV 4,12)
  ENDSELECT
RETURN
PROCEDURE jouelatin2
  @jouelatin1
  @jouelatin1
RETURN
PROCEDURE jouelatin3
  @jouelatin1
  @jouelatin1
  @jouelatin1
RETURN
PROCEDURE jouelatin4
  @jouelatin1
  @jouelatin1
  @jouelatin1
  @jouelatin1
RETURN
PROCEDURE sauver
  range|=3
  FOR i|=0 TO tempstot|-1
    MID$(transfert$,range|,1)=CHR$(LEN(accomp$(i|)))
    INC range|
    MID$(transfert$,range|,LEN(accomp$(i|)))=accomp$(i|)
    ADD range|,LEN(accomp$(i|))
  NEXT i|
  MID$(transfert$,1,1)=CHR$(range|-1)
  MID$(transfert$,2,1)=CHR$(tempstot|)
  FILESELECT "A:\*.GRI","",nom$
  IF nom$="" OR RIGHT$(nom$)="\"
    GOTO f1
```

(suite)

```
    ENDIF
    IF INSTR(nom$,".")<>0
      nom$=LEFT$(nom$,INSTR(nom$,"."))+"GRI"
    ELSE
      nom$=nom$+".GRI"
    ENDIF
    BSAVE nom$,VARPTR(transfert$),range|-1
    f1:
  RETURN
  PROCEDURE charger
    FILESELECT "A:\*.GRI","",nom$
    IF nom$="" OR RIGHT$(nom$)="\"
      GOTO f2
    ENDIF
    BLOAD nom$,VARPTR(transfert$)
    tempstot|=ASC(MID$(transfert$,2,1))
    range|=ASC(transfert$)
    IF range|>2
      posi|=4
      FOR i|=0 TO tempstot|-1
        indi|=ASC(MID$(transfert$,posi|-1,1))
        accomp$(i|)=MID$(transfert$,posi|,indi|)
        ADD posi|,indi|+1
      NEXT i|
    ENDIF
    PUT 0,100,eff$
    @grille
    FOR i|=1 TO tempstot|
      xx&=xdep|+cursx|+((i|-1) MOD 16)*grillarg|
      yy&=ydep|+grilhaut|*(1+(i|-1) DIV 16)+curs|+ecart|*((i|-1) DIV 16)
      IF LEN(accomp$(i|-1))>1
        IF LEN(accomp$(i|-1))>2
          FOR j|=0 TO 6
            IF interv$(j|)=MID$(accomp$(i|-1),2,LEN(accomp$(i|-1))-1)
              acc$=tonique$(ASC(accomp$(i|-1))-60)+type$(j|)
            ENDIF
          NEXT j|
        ELSE
          acc$=type$(7)
        ENDIF
        TEXT xx&,yy&-2*curs|,acc$
      ELSE
        TEXT xx&,yy&-2*curs|,"    "
      ENDIF
    NEXT i|
    f2:
  RETURN
```

(suite)

```
PROCEDURE grille
  FOR i|=1 TO tempstot|
    xx&=xdep|+((i|-1) MOD 16)*grillarg|
    yy&=ydep|+((i|-1) DIV 16)*grilhaut|+ecart|*((i|-1) DIV 16)
    BOX xx&,yy&,xx&+grillarg|,yy&+grilhaut|
    IF i| MOD temps|=1
      LINE xx&+1,yy&,xx&+1,yy&+grilhaut|
    ENDIF
  NEXT i|
  point|=0
  cursxanc&=xdep|+cursx|
  cursyanc&=ydep|+grilhaut|+curs|
  TEXT cursxanc&,cursyanc&,curs$
RETURN
PROCEDURE new
  ALERT 2,"voulez-vous|effacer|la grille actuelle|",1,"oui|non",r|
  IF r|=1
    ALERT 2,"taille de |grille",3,"2|4|8",n|
    tempstot|=temps|*2^n|
    PUT 0,100,eff$
    @grille
  ENDIF
RETURN
PROCEDURE new1
  ALERT 2,"taille de |grille",3,"2|4|8",n|
  tempstot|=temps|*2^n|
  @grille
RETURN
```

```
D  m7  jouer out   sauver new  charger     folk        cycle  rep  tempo
            midi 1                                      on     2    120
```

CM	DM	D#M	EM	A7	Dm7	GM	F7	A#M	sil	Bm7	AM	G#M		F#M	FM

Em		Dm7	C#m	Cm		Ddm		Edm	Fdm	F+		G+		As4	As4

COMPLÉMENT DE LA GÉNÉRATION D'ACCOMPAGNEMENT AUTOMATIQUE

Dans le listing précédent, il suffit de remplacer les procédures jouefolk2, jouefolk3, jouefolk4, jouedisco2, jouedisco3, jouedisco4, jouelatin2, jouelatin3, jouelatin4 par les nouvelles procédures de même nom. Les motifs générés sont interchangeables dans un même style et sur une même durée d'accord. Les choix sont présentés de façon équiprobable ici, mais vous pouvez regrouper plusieurs sélections de valeurs entières sous un même CASE si vous voulez privilégier un arpège. Les tensions introduites sur les accords se limitent volontairement à la seconde en note de passage pour ne pas générer d'accident désagréable à l'oreille. Suivant le côté tonal ou modal de vos styles vous pourrez aussi introduire la sixte majeure sur les accords majeurs et la sixte majeure ou mineure sur les accords mineurs suivant que votre accord aura une fonction modale ou tonale. La quarte se joue sans problème sur les accords mineurs ; pour les autres accords, reportez-vous aux tensions inclues dans les définitions des accords dans le programme d'analyse d'accords ou aux gammes utilisées dans la dictée mélodique. Pensez qu'une bonne tension peut durer sur un temps fort de l'accompagnement, (1 et 3), alors qu'une note de passage a pour caractéristique de glisser sur les temps faibles, (2 et 4), et dernières notes de chaque temps.



```
' ----------ACCOMPAGNEMENT AUTOMATIQUE-----
' ----------
PROCEDURE jouefolk2
  SELECT RANDOM(8)
  CASE 0
    @jouefolk1
    @jouefolk1
  CASE 1
    @motdep(tempo& DIV 2,0)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 4,0)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
    @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
  CASE 2
    @motdep(tempo& DIV 2,0)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 4,0)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
  CASE 3
    @motdep(tempo& DIV 2,0)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
    @motdep(tempo& DIV 2,0)
  CASE 4
    @motdep(tempo& DIV 2,0)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 2,12)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
  CASE 5
    @motdep(tempo& DIV 2,0)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 4,12)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
```

(suite)

```
    CASE 6
      @motdep(tempo& DIV 2,0)
      @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
      @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
      @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
      @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
      @motdep(tempo& DIV 2,12)
    CASE 7
      @motdep(tempo& DIV 2,0)
      @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
      @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
      @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
      @motdep(tempo& DIV 4,12)
      @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
      @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
    RETURN
    PROCEDURE jouefolk3
      SELECT RANDOM(10)
      CASE 0
        @jouefolk2
        @jouefolk1
      CASE 1
        @motdep(tempo& DIV 2,0)
        @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
        @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
        @motdep(tempo& DIV 4,0)
        @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
        @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
        @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
        @motdep(tempo& DIV 4,0)
        @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
        @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
      CASE 2
        @motdep(tempo& DIV 2,0)
        @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
        @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
        @motdep(tempo& DIV 4,0)
        @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
        @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
        @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
        @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
        @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
        @motdep(tempo& DIV 4,0)
```

(suite)

```
CASE 3
  @motdep(tempo& DIV 2,0)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
  @motif(tempo& DIV 4)
CASE 4
  @motdep(tempo& DIV 2,0)
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,12)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motif(tempo& DIV 2)
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motif(tempo& DIV 4)
CASE 5
  @motdep(tempo& DIV 2,0)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
  @motdep(tempo& DIV 4,12)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motif(tempo& DIV 2)
CASE 6
  @motdep(tempo& DIV 2,0)
  @motif(tempo& DIV 4)
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 2,12)
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
  @motif(tempo& DIV 4)
```

(suite)

```
CASE 7
  @motdep(tempo& DIV 2,0)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 4,12)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
  @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
  @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
CASE 8
  @motdep(tempo& DIV 2,0)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 4,12)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motif(tempo& DIV 2)
  @motdep(tempo& DIV 4,12)
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
CASE 9
  @motdep(tempo& DIV 2,0)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 2,12)
  @motif(tempo& DIV 4)
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
  @motdep(tempo& DIV 4,2)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
RETURN
```

(suite)

```
PROCEDURE jouefolk4
  SELECT RANDOM(10)
  CASE 0
    @jouefolk2
    @jouefolk2
  CASE 1
    @jouefolk3
    @jouefolk1
  CASE 2
    @motdep(tempo& DIV 2,0)
    @motif(tempo& DIV 4)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 4,12)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motif(tempo& DIV 2)
    @motif(tempo& DIV 4)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motif(tempo& DIV 4)
    @motdep(tempo& DIV 4,12)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motif(tempo& DIV 2)
  CASE 3
    @motdep(tempo& DIV 2,0)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
    @motif(tempo& DIV 4)
    @motdep(tempo& DIV 4,12)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motif(tempo& DIV 2)
    @motdep(tempo& DIV 4,12)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motif(tempo& DIV 4)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
    @motdep(tempo& DIV 4,12)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
    @motdep(tempo& DIV 4,2)
```

(suite)

```
CASE 4
   @motdep(tempo& DIV 2,0)
   @motif(tempo& DIV 4)
   @motif(tempo& DIV 4)
   @motdep(tempo& DIV 2,12)
   @motif(tempo& DIV 4)
   @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
   @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
   @motif(tempo& DIV 4)
   @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
   @motif(tempo& DIV 4)
   @motdep(tempo& DIV 2,0)
   @motdep(tempo& DIV 4,2)
   @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
CASE 5
   @motdep(tempo& DIV 2,0)
   @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
   @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
   @motdep(tempo& DIV 4,0)
   @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
   @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
   @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
   @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
   @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
   @motdep(tempo& DIV 4,0)
   @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
   @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
   @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
   @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
CASE 6
   @motdep(tempo& DIV 2,0)
   @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
   @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
   @motdep(tempo& DIV 2,12)
   @motif(tempo& DIV 4)
   @motif(tempo& DIV 4)
   @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
   @motdep(tempo& DIV 4,2)
   @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
   @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
   @motdep(tempo& DIV 4,2)
   @motdep(tempo& DIV 4,0)
   @motdep(tempo& DIV 4,2)
   @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
```

(suite)

```
CASE 7
  @motdep(tempo& DIV 4,2)
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 4,2)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
  @motdep(tempo& DIV 2,0)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 2,12)
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motif(tempo& DIV 4)
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
CASE 8
  @motdep(tempo& DIV 2,12)
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motif(tempo& DIV 4)
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
  @motdep(tempo& DIV 4,0)
CASE 9
  @motif(tempo& DIV 2)
  @motdep(tempo& DIV 4,12)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
  @motdep(tempo& DIV 4,12)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motif(tempo& DIV 2)
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 4,12)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
RETURN
```

(suite)

```
PROCEDURE jouedisco2
  SELECT RANDOM(9)
  CASE 0
    @motdep(tempo&,0)
    @motdep(tempo& DIV 4,0)
    @motdep(tempo& DIV 4,0)
    @motdep(tempo& DIV 4,12)
    @motdep(tempo& DIV 4,12)
  CASE 1
    @motdep(tempo& DIV 2,0)
    @motdep(tempo& DIV 4,12)
    @motdep(tempo& DIV 4,12)
    @motdep(tempo& DIV 4,0)
    @motdep(tempo& DIV 4,0)
    @motdep(tempo& DIV 2,12)
  CASE 2
    @motdep(tempo& DIV 2,0)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 4,12)
    @motdep(tempo& DIV 4,0)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 2,12)
  CASE 3
    @motdep(tempo&,0)
    @motdep(tempo& DIV 4,0)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 4,12)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  CASE 4
    @motdep(tempo& DIV 2,0)
    @motdep(tempo& DIV 2,0)
    @motdep(tempo& DIV 2,12)
    @motdep(tempo& DIV 2,12)
  CASE 5
    @motdep(tempo& DIV 2,0)
    @motdep(tempo& DIV 2,0)
    @motdep(tempo& DIV 2,12)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 4,12)
```

(suite)

```
    CASE 6
      @motdep(tempo& DIV 2,0)
      @motdep(tempo& DIV 2,0)
      @motdep(tempo& DIV 4,12)
      @motdep(tempo& DIV 4,12)
      @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
      @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
    CASE 7
      @motdep(tempo& DIV 4,0)
      @motdep(tempo& DIV 4,0)
      @motdep(tempo& DIV 4,0)
      @motdep(tempo& DIV 4,0)
      @motdep(tempo& DIV 4,0)
      @motdep(tempo& DIV 4,0)
      @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
      @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    CASE 8
      @motdep(tempo& DIV 4,0)
      @motdep(tempo& DIV 4,0)
      @motif(tempo& DIV 4)
      @motif(tempo& DIV 4)
      @motdep(tempo& DIV 4,0)
      @motdep(tempo& DIV 4,0)
      @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
      @motif(tempo& DIV 4)
    RETURN
    PROCEDURE jouedisco3
      SELECT RANDOM(10)
      CASE 0
        @jouedisco2
        @jouedisco1
      CASE 1
        @motdep(tempo&,0)
        @motdep(tempo& DIV 4,0)
        @motdep(tempo& DIV 4,0)
        @motdep(tempo& DIV 4,12)
        @motdep(tempo& DIV 4,12)
        @motdep(tempo& DIV 2,0)
        @motdep(tempo& DIV 4,12)
        @motdep(tempo& DIV 4,12)
```

(suite)

```
CASE 2
  @motdep(tempo&,0)
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,12)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 2,0)
  @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
CASE 3
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,12)
  @motdep(tempo& DIV 4,12)
  @motdep(tempo& DIV 4,12)
  @motdep(tempo& DIV 4,12)
CASE 4
  @motdep(tempo& DIV 2,0)
  @motdep(tempo& DIV 2,0)
  @motdep(tempo& DIV 2,12)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,12)
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 2,12)
CASE 5
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 4,0)
  @motif(tempo& DIV 4)
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 2,0)
  @motdep(tempo& DIV 2,12)
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
  @motif(tempo& DIV 4)
CASE 6
  @motdep(tempo&,0)
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 4,12)
  @motdep(tempo& DIV 4,12)
  @motdep(tempo&,2)
  @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
```



(suite)

```
    CASE 7
      @motdep(tempo& DIV 2,0)
      @motdep(tempo& DIV 2,0)
      @motdep(tempo& DIV 2,12)
      @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
      @motdep(tempo& DIV 4,12)
      @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
      @motdep(tempo& DIV 2,2)
    CASE 8
      @motdep(tempo& DIV 2,0)
      @motdep(tempo& DIV 2,0)
      @motdep(tempo& DIV 2,12)
      @motdep(tempo& DIV 2,12)
      @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
      @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
      @motdep(tempo& DIV 4,2)
      @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    CASE 9
      @motdep(tempo& DIV 2,0)
      @motdep(tempo& DIV 2,0)
      @motdep(tempo& DIV 2,12)
      @motdep(tempo& DIV 2,12)
      @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
      @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
    RETURN
    PROCEDURE jouedisco4
      SELECT RANDOM(9)
      CASE 0
        @jouedisco3
        @jouedisco1
      CASE 1
        @jouedisco2
        @jouedisco2
      CASE 2
        @motdep(tempo& DIV 2,0)
        @motdep(tempo& DIV 2,0)
        @motdep(tempo& DIV 2,12)
        @motdep(tempo& DIV 2,12)
        @motdep(tempo& DIV 2,0)
        @motdep(tempo& DIV 2,2)
        @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
        @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
```

(suite)

```
CASE 3
   @motdep(tempo& DIV 4,0)
   @motdep(tempo& DIV 4,0)
   @motif(tempo& DIV 4)
   @motif(tempo& DIV 4)
   @motdep(tempo& DIV 2,0)
   @motdep(tempo& DIV 2,12)
   @motdep(tempo& DIV 4,0)
   @motdep(tempo& DIV 4,0)
   @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
   @motif(tempo& DIV 4)
   @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
   @motdep(tempo& DIV 2,2)
CASE 4
   @motdep(tempo& DIV 4,0)
   @motdep(tempo& DIV 4,0)
   @motdep(tempo& DIV 4,0)
   @motdep(tempo& DIV 4,0)
   @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
   @motdep(tempo& DIV 2,2)
   @motdep(tempo& DIV 4,0)
   @motdep(tempo& DIV 4,0)
   @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
   @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
   @motdep(tempo& DIV 4,12)
   @motdep(tempo& DIV 4,12)
   @motdep(tempo& DIV 4,12)
   @motdep(tempo& DIV 4,12)
CASE 5
   @motdep(tempo&,0)
   @motdep(tempo& DIV 4,0)
   @motdep(tempo& DIV 4,0)
   @motdep(tempo& DIV 4,12)
   @motdep(tempo& DIV 4,12)
   @motdep(tempo&,2)
   @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
   @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
   @motif(tempo& DIV 4)
   @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
```

(suite)

```
CASE 6
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 4,0)
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,12)
  @motdep(tempo& DIV 4,12)
  @motdep(tempo& DIV 4,12)
  @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
  @motdep(tempo& DIV 2,2)
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
CASE 7
  @motdep(tempo& DIV 2,0)
  @motdep(tempo& DIV 2,0)
  @motdep(tempo& DIV 2,12)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,12)
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,12)
  @motdep(tempo& DIV 4,12)
  @motdep(tempo&,0)
CASE 8
  @motdep(tempo&,0)
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 4,12)
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 4,12)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,12)
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 2,12)
  @motdep(tempo& DIV 4,12)
  @motdep(tempo& DIV 4,12)
RETURN
```

(suite)

```
PROCEDURE jouelatin2
  SELECT RANDOM(9)
  CASE 0
    @motdep(3*tempo& DIV 4,0)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(3*tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 4,0)
  CASE 1
    @motdep(3*tempo& DIV 4,0)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 4,12)
    @motdep(3*tempo& DIV 4,ASC(MID$(ring$,3,1)))
  CASE 2
    @motdep(3*tempo& DIV 4,0)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
    @motdep(3*tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
  CASE 3
    @motdep(3*tempo& DIV 4,0)
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 4,0)
    @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  CASE 4
    @motdep(tempo& DIV 4,0)
    @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 4,0)
    @motdep(tempo& DIV 4,0)
    @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
  CASE 5
    @motdep(tempo& DIV 4,0)
    @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 4,0)
    @motdep(tempo& DIV 4,12)
    @motif(tempo& DIV 2)
  CASE 6
    @motdep(tempo& DIV 4,0)
    @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
    @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
    @motdep(tempo& DIV 4,0)
    @motdep(tempo& DIV 4,12)
    @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
```

(suite)

```
    CASE 7
      @motdep(tempo& DIV 4,0)
      @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
      @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
      @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
      @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
      @motdep(tempo& DIV 2,0)
    CASE 8
      @motdep(3*tempo& DIV 4,0)
      @motdep(tempo& DIV 4,12)
      @motif(3*tempo& DIV 4)
      @motif(tempo& DIV 4)
    RETURN
    PROCEDURE jouelatin3
      SELECT RANDOM(10)
      CASE 0
        @jouelatin2
        @jouelatin1
      CASE 1
        @motdep(tempo& DIV 4,0)
        @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
        @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
        @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
        @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
        @motdep(tempo& DIV 2,0)
        @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
        @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
        @motdep(tempo& DIV 2,0)
      CASE 2
        @motdep(tempo& DIV 4,0)
        @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
        @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
        @motdep(tempo& DIV 4,0)
        @motdep(tempo& DIV 4,12)
        @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
        @motdep(tempo& DIV 2,0)
        @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
      CASE 3
        @motdep(tempo& DIV 4,0)
        @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
        @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
        @motdep(tempo& DIV 4,0)
        @motdep(tempo& DIV 4,0)
        @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
        @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
        @motdep(tempo& DIV 2,2)
        @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
```

(suite)

```
CASE 4
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 4,12)
  @motif(tempo& DIV 2)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motif(tempo& DIV 2)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
CASE 5
  @motdep(3*tempo& DIV 4,0)
  @motdep(tempo& DIV 4,12)
  @motif(3*tempo& DIV 4)
  @motif(tempo& DIV 4)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motif(tempo& DIV 2)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
CASE 6
  @motdep(3*tempo& DIV 4,0)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,2)
  @motdep(tempo& DIV 4,2)
  @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
CASE 7
  @motdep(3*tempo& DIV 4,0)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(3*tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,0)
  @motif(tempo& DIV 4)
  @motif(tempo& DIV 2)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
CASE 8
  @motdep(3*tempo& DIV 4,0)
  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
  @motdep(3*tempo& DIV 4,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
  @motdep(tempo& DIV 4,0)
  @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
  @motdep(tempo& DIV 4,0)
```



(suite)

```
        CASE 9
          @motdep(3*tempo& DIV 4,0)
          @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
          @motdep(tempo& DIV 4,12)
          @motdep(3*tempo& DIV 4,ASC(MID$(ring$,3,1)))
          @motdep(tempo& DIV 4,0)
          @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
          @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
        RETURN
        PROCEDURE jouelatin4
          SELECT RANDOM(10)
          CASE 0
            @jouelatin3
            @jouelatin1
          CASE 1
            @jouelatin2
            @jouelatin2
          CASE 2
            @motdep(tempo& DIV 4,0)
            @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
            @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
            @motdep(tempo& DIV 4,0)
            @motdep(tempo& DIV 4,12)
            @motif(tempo& DIV 2)
            @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
            @motif(tempo& DIV 2)
            @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
            @motdep(tempo& DIV 4,2)
            @motdep(tempo& DIV 4,2)
            @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
          CASE 3
            @motdep(3*tempo& DIV 4,0)
            @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
            @motdep(tempo& DIV 4,0)
            @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
            @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
            @motdep(tempo& DIV 4,2)
            @motdep(tempo& DIV 4,2)
            @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
            @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
            @motdep(tempo& DIV 4,2)
            @motdep(tempo& DIV 2,0)
```

(suite)

```
CASE 4
   @motdep(tempo& DIV 4,0)
   @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
   @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
   @motdep(tempo& DIV 4,0)
   @motdep(tempo& DIV 4,12)
   @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
   @motdep(3*tempo& DIV 4,0)
   @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
   @motdep(3*tempo& DIV 4,2)
   @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
CASE 5
   @motdep(tempo& DIV 4,2)
   @motdep(tempo& DIV 4,2)
   @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
   @motdep(tempo& DIV 4,0)
   @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
   @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
   @motdep(tempo& DIV 4,0)
   @motdep(tempo& DIV 4,0)
   @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
   @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
   @motdep(tempo& DIV 2,2)
   @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
CASE 6
   @motdep(tempo& DIV 4,0)
   @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
   @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
   @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
   @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
   @motdep(tempo& DIV 2,0)
   @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
   @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
   @motdep(tempo& DIV 2,0)
   @motdep(tempo& DIV 4,2)
   @motdep(tempo& DIV 2,ASC(MID$(ring$,3,1)))
   @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
```

(suite)

```
                CASE 7
                  @motdep(3*tempo& DIV 4,0)
                  @motdep(tempo& DIV 4,12)
                  @motdep(3*tempo& DIV 4,12)
                  @motdep(tempo& DIV 4,0)
                  @motif(3*tempo& DIV 4)
                  @motif(tempo& DIV 4)
                  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
                  @motif(tempo& DIV 2)
                  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
                CASE 8
                  @motdep(3*tempo& DIV 4,0)
                  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
                  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
                  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
                  @motdep(tempo& DIV 2,0)
                  @motdep(tempo& DIV 4,12)
                  @motdep(3*tempo& DIV 4,ASC(MID$(ring$,3,1)))
                  @motdep(tempo& DIV 4,0)
                  @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
                  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
                CASE 9
                  @motdep(3*tempo& DIV 4,0)
                  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
                  @motdep(3*tempo& DIV 4,ASC(MID$(ring$,3,1)))
                  @motdep(tempo& DIV 4,0)
                  @motif(tempo& DIV 4)
                  @motdep(tempo& DIV 4,0)
                  @motdep(tempo& DIV 2,ASC(MID$(ring$,2,1)))
                  @motdep(tempo& DIV 4,ASC(MID$(ring$,3,1)))
                  @motif(tempo& DIV 2)
                  @motdep(tempo& DIV 4,ASC(MID$(ring$,2,1)))
                RETURN
' ------Fin du complément N° 8 sur l'accompagnement automatique------
```

6/1.6

Jeux musicaux en GFA

INTRODUCTION

Sous l'appellation jeux musicaux, nous vous proposons une série de quatre programmes en Basic-GFA.

Les jeux reposent sur des devinettes d'intervalles et d'accords ainsi que des quizz sur les mêmes sujets. Les exemples peuvent être implémentés sur le générateur sonore aussi bien que sur le Midi, en prenant un schéma simple d'équivalence de codes claviers et Midi, comme cela a été fait dans les exemples musicaux traitant des dictées musicales et autres analyses d'accords ou d'intervalles. Les exemples d'accords ou d'intervalles choisis sont volontairement simples et font référence aux combinaisons les plus courantes. Mais même avec ces combinaisons de base, vous pourrez avoir quelques surprises. Avant d'entrer dans le vif du sujet des programmes présentés, il est bon de faire quelques rappels ou compléments d'ordre musical.

On appelle «intervalle» un écart entre deux hauteurs de notes; les intervalles utilisés ici sont mesurés en nombre de demi-tons et font référence aux degrés des gammes majeure et mineure. Les intervalles peuvent être ascendants (la seconde note est plus aiguë que la première) ou descendants dans le cas contraire. Nous traiterons les deux cas, bien que l'oreille soit bien plus habituée à reconnaître les intervalles ascendant que les descendants. L'ambitus maximum sera l'octave, mais vous pouvez facilement étendre les exemples sur deux octaves en agrandissant les tableaux. Théoriquement les intervalles sont répertoriés lorsque deux notes sont différentes, mais nous avons inclus l'unisson dans nos programmes sur les intervalles, que les puristes veuillent bien nous en excuser! Les intervalles traités entrent dans la composition de base des accords traditionnels de trois et quatre sons.

En voici un tableau récapitulatif:

Intervalle	Nombre de demi-tons
Unisson	0
Tierce min	3
Tierce maj	4
Quarte juste	5
Quinte juste	7
Septième min	10
Octave juste	12

Ces intervalles permettent de repérer les degrés correspondants des gammes dont les appellations correspondent à la terminologie des intervalles. Ces degrés se repèrent à partir de la note de base de la gamme ou «tonique». Dans les jeux, cette tonique sera choisie au hasard sur l'un des douze demi-tons constituant une octave au choix et permettra de tester votre oreille relative. Les intervalles les plus communs seront testés dans le niveau 1 du jeu, et tous les intervalles le seront dans le niveau deux. Vous pouvez intervertir à loisir quelques choix dans les niveaux et rajouter d'autres niveaux ou intervalles, puisque la logique est la même. N'oubliez pas de bien redimensionner les tableaux correspondants.

Dans le cas où le jeu s'effectue sur le générateur sonore, il vous faudra simuler un clavier de piano sur votre Atari.

Les accords sélectionnés font partie des accords de base construits sur les degrés de la gamme majeure. Rappelons que ces accords se construisent par superpositions de tierces successives (en sautant une note sur deux dans la gamme).

Prenons l'exemple de la gamme de Do, et l'ordre des notes sur une octave et demi:

Do Ré Mi Fa Sol La Si Do Ré Mi Fa

1 2 3 4 5 6 7 8 9 10 11

Au dessous des notes sont notés les degrés de la gamme en commençant par le degré 1 ou tonique.

L'accord du premier degré est constitué par les degrés 1, 3, 5 et ainsi de suite jusqu'au septième degré puisque les gammes heptatoniques fonctionnent en modulo 8. Ces accords à trois sons font partie de la panoplie de base de notre jeu sur les accords.



Nous trouvons les dénominations et suites d'intervalles sur les degrés de la gamme majeure:

Degré	Degrés de l'accord	Suite d'intervalles	Nom de l'accord
1	1,3,5	4,3	Majeur
2	2,4,6	3,4	mineur
3	3,5,7	3,4	mineur
4	4,6,8	4,3	Majeur
5	5,7,9	4,3	Majeur
6	6,8,10	3,4	mineur
7	7,9,11	3,3	mineur quinte dim

Les accords à quatre sons (donc trois intervalles) utilisés sont l'accord de septième de dominante (formé sur le cinquième degré ou dominante ou quinte) et l'accord formé sur le septième degré ou accord mineur septième quinte diminuée. Les suites respectives d'intervalles et degrés sont :

Degré	Degrés de l'accord	Suite d'intervalles	Nom de l'accord
5	5,7,9,11	4,3,3	septième
7	7,9,11,13	3,3,4	min7 qte dim

Deux autres accords ne faisant pas directement référence à la gamme majeure, mais communément usités, sont employés dans les jeux sur les accords. Ce sont : l' accord Sus 4, ou accord suspendu dans lequel la quarte remplace la tierce (Majeure ou mineure), et l'accord de septième diminuée emprunté à la gamme mineure harmonique (suite de demi-tons : 0, 2, 3, 5, 7, 8, 11, 12) et construit par succession de tierces mineures sur le septième degré de cette gamme.

Degré	Degrés de l'accord	Suite d'intervalles	Nom de l'accord
1	1,4,5	5,2	sus 4
7	7,9,11,13	3,3,3	septième dim

Notez que la structure de cet accord à quatre sons est la même que celle de l'accord de mineur 7 quinte dim, mais le degré 6, ou 13 à l'octave supérieure est plus bas d'un demi-ton que dans la gamme majeure.

Principe des jeux

Deux niveaux sont proposés pour chacun des quatre jeux. Pour choisir le niveau, il vous suffit de cliquer dans la case correspondante. Dans le niveau 1 seules sont affichées ou sélectionnables les solutions

correspondant à ce niveau. Dans les jeux de devinettes vous devez cliquer dans la case qui vous semble correspondre à l'intervalle ou l'accord entendu. Vous avez droit à un certain nombre de réponses fausses avant que la bonne solution vous soit proposée. Il faut évidement que le nombre de solutions possibles soit inférieur au nombre de tentatives pour que le jeu ait un sens. Avant de sélectionner une réponse, vous avez tout le loisir de réécouter l'exemple proposé. Dans le niveau 2, le nombre de mauvaises réponses permises est abaissé et les exemples sont joués plus vite. Du fait de la polyphonie du générateur sonore limitée à trois, les notes des accords seront égrenées et non plaquées. Dans les jeux des quizz, il vous faudra jouer l'accord demandé. Le programme se permet quand même d'analyser une bonne réponse de votre part si vous jouez un renversement d'accord au lieu de sa position fondamentale! Sympa, non?

A ce sujet, il n'est fait aucunement mention d'un renversement proposé dans les jeux sur les accords, mais si vous désirez introduire cette difficulté supplémentaire, vous pouvez vous inspirer grandement du codage et de l'analyse des renversements et positions d'accords (voir 6/2.4.4).

DEVINETTE D'INTERVALLE

Le programme en général

Le programme se présente en trois parties:

— l'initialisation;

— la boucle principale;

— les procédures.

Dans l'initialisation, les affichages et les tableaux de variables sont initialisés.

Les tableaux de niveaux 1 et 2 sont lus avec les suites d'intervalles choisies (**nivdevint1|()**, **oppdevint1|()**). Le tableau **intervalle|()** contient toutes les valeurs d'intervalles possibles sur une octave. Les titres d' options sont inscrits et les sélections par défaut sont faites sur les variables **asc&=1** (intervalle ascendant), **trouvint!** et **gen!** sur **FALSE** puisque le jeu n'a pas été généré et que, par conséquent, aucun intervalle n'a été trouvé.

Le programme en détail

Le corps de la boucle est une suite de test de clics de souris afin de se brancher soit sur des transformations de paramètres de choix de jeu, soit sur des traitements spécifiques du jeu.

Lorsque le niveau change, la variable logique **gen!** reprend la valeur fausse puisque le jeu se réinitialise; même remarque lorsque l'intervalle passe d' ascendant à descendant, et réciproquement. La procédure **rejouer** ne pourra s'effectuer que si le jeu a été généré.



Les procédures principales

genereint

Cette procédure génère et joue un intervalle respectant les critères de choix sélectionnés (**asc&** et **niveau**). Les deux notes sont choisies au hasard à partir de la tonique choisie, elle aussi, au hasard. Il n'y a pas mémorisation du tirage d'intervalle, ce qui peut produire deux tirages identiques à la suite. Une variable **souvenir** peut facilement pallier à ce défaut, après le premier tirage en incluant le tirage aléatoire jusqu'à ce que **r|** soit différent de **souvenir|**. Le sens montant ou descendant de la déduction se fait au moment du jeu par **asc&** et la valeur de **r|**.

C'est cette dernière valeur qui mémorise l'intervalle proposé.

analyseint

Cette procédure n'analyse l'intervalle sélectionné que si le jeu a démarré (test de **gen!**). Il suffit de comparer **i|** et **r|** pour savoir si le choix est bon ou non, et le répertorier dans la variable **trouvint!**.

Au bout de **4-nivdevint|** mauvaises réponses, on donne la bonne réponse par l'appel de la procédure **ilfallaitint**.

rejouerint()

Si le jeu a débuté (**gen!** = TRUE), les deux notes sont jouées successivement, la position de la seconde par rapport à la première étant corrigée par la valeur de **mode&**, et le tempo du jeu étant modifié par le niveau.

ilfallaitint

L'intervalle sélectionné est mémorisé par l'indice **i|** et l'intervalle à trouver par l'indice **r|**. La procédure **rejouer** est appelée successivement dans ces deux cas.

```
' ------ LISTING DE DEVINE INTERVALLE -------
DIM interv$(12),intervalle|(12),nivdevint1|(6),oppdevint1|(5)
FOR i|=0 TO 12
  READ interv$(i|)
NEXT i|
DATA Un,2m,2M,3m,3M,4J
DATA 5-,5J,5+,6M,7m
DATA 7M,8J
FOR i|=0 TO 6
  READ nivdevint1|(i|)
NEXT i|
DATA 0,3,4,5,7,10,12
FOR i|=0 TO 5
  READ oppdevint1|(i|)
NEXT i|
DATA 1,2,6,8,9,11
FOR i|=0 TO 12
  intervalle|(i|)=i|
NEXT i|
TEXT 200,20,"DEVINNETTE D'INTERVALLES"
aff
trouvint!=FALSE
gen!=FALSE
asc&=1
```

(suite)

```
nivdevint1|=1
TEXT 280,50,"NIVEAU"
TEXT 280,70,nivdevint1|
TEXT 350,50,"JOUER"
TEXT 420,50,"ECOUTER"
TEXT 420,70,"MIDI 1"
TEXT 500,50,"QUITTER"
' TEXT 570,50,"TEMPO"
' TEXT 570,70,"60"
tempo|=50
TEXT 610,50,"INT"
TEXT 610,70,"ASC"
DO
  MOUSE x&,y&,k|
  IF k|=1
    FOR i|=0 TO 12
      IF x&>20*i| AND x&<20*i|+15 AND y&<50 AND y&>40
        @analyseint
      ENDIF
    NEXT i|
    IF x&>350 AND x&<380 AND y&>40 AND y&<50
      @genererint
    ENDIF
    IF x&>280 AND x&<300 AND y&>60 AND y&<70
      IF nivdevint1|=1
        nivdevint1|=2
        TEXT 280,70," "
        TEXT 280,70,nivdevint1|
        gen!=FALSE
        aff2
      ENDIF
    ENDIF
    IF x&>420 AND x&<450 AND y&>40 AND y&<50
      @rejouerint(r|,asc&,tempo|)
    ENDIF
    IF x&>500 AND x&<530 AND y&>40 AND y&<50
      @quitter
    ENDIF
    IF x&>615 AND x&<630 AND y&>60 AND y&<70
      IF asc&=-1
        gen!=FALSE
        TEXT 610,70,"  "
        TEXT 610,70,"ASC"
      ENDIF
      asc&=1
    ENDIF
  ENDIF
  IF k|=2
```



(suite)

```
    IF x&>615 AND x&<630 AND y&>60 AND y&<70
      IF asc&=1
        gen!=FALSE
        TEXT 610,70,"  "
        TEXT 610,70,"DESC"
      ENDIF
      asc&=-1
    ENDIF
    IF x&>280 AND x&<300 AND y&>60 AND y&<70
      IF nivdevint1|=2
        nivdevint1|=1
        TEXT 280,70," "
        TEXT 280,70,nivdevint1|
        gen!=FALSE
        aff1
      ENDIF
    ENDIF
  ENDIF
LOOP
PROCEDURE analyseint
  IF gen!=TRUE
    trouvint!=FALSE
    IF i|=r|
      trouvint!=TRUE
    ENDIF
    TEXT 200,300,"                "
    IF trouvint!=TRUE
      TEXT 200,300,"c'est tout bon"
    ELSE
      IF test1<4-nivdevint1|
        TEXT 200,300,"ce n'est pas tout bon"
        INC test|
        @ilfallaitint
      ELSE
        TEXT 200,300,"IL FALLAIT CLIQUER"
        TEXT 400,300,interv$(r|)
      ENDIF
    ENDIF
  ELSE
    TEXT 200,300," CLIQUEZ D'ABORD SUR JOUER"
  ENDIF
RETURN
PROCEDURE genererint
  test|=0
  TEXT 200,300,"                                    "
  TEXT 200,200,"CLIQUEZ DANS LA CASE DE L'INTERV ENTENDU"
  gen!=TRUE
  notdep|=60-(asc&=-1)*12+RANDOM(13)
```

(suite)

```
  PAUSE 50
  SELECT nivdevint1|
  CASE 2
    r|=RANDOM(13)
    tempo|=40
  CASE 1
    r|=nivdevint1|(RANDOM(7))
    tempo|=50
  ENDSELECT
  @rejouerint(r|,asc&,tempo|)
RETURN
PROCEDURE rejouerint(rer|,mode&,temps|)
  IF gen!=TRUE
    OUT 3,144,notdep|,80
    PAUSE temps|
    OUT 3,notdep|,0
    OUT 3,notdep|+mode&*intervalle|(rer|),80
    PAUSE temps|
    OUT 3,144,notdep|+mode&*intervalle|(rer|),0
  ELSE
    TEXT 200,300,"                                "
    TEXT 200,300," CLIQUEZ D'ABORD SUR JOUER"
  ENDIF
RETURN
PROCEDURE quitter
  EDIT
RETURN
PROCEDURE ilfallaitint
  ' PAUSE 100
  TEXT 200,300,"                                "
  TEXT 200,300,"ECOUTEZ L'INTERV QUE VOUS AVEZ SELECTIONNE"
  PAUSE 100
  rejouerint(i|,asc&,tempo|-15)
  TEXT 200,300,"                                                "
  TEXT 200,300,"ECOUTEZ L'INTERV QUE VOUS DEVEZ TROUVER"
  PAUSE 100
  @rejouerint(r|,asc&,tempo|)
  TEXT 200,300,"                                        "
RETURN
PROCEDURE aff
  DEFFILL 1,0,0
  FOR i|=0 TO 12
    BOX 20*i|,35,20*i|+18,55
    TEXT 20*i|,50,interv$(i|)
  NEXT i|
  aff1
RETURN
PROCEDURE aff1
```



(suite)

```
  DEFFILL 1,2,2
  FOR i|=0 TO 5
    PBOX 20*oppdevint1|(i|)+1,35,20*oppdevint1|(i|)+17,55
    TEXT 20*(oppdevint1|(i|)),50,interv$(oppdevint1|(i|))
  NEXT i|
RETURN
PROCEDURE aff2
  DEFFILL 1,0,0
  FOR i|=0 TO 5
    PBOX 20*oppdevint1|(i|)+1,35,20*oppdevint1|(i|)+17,55
    TEXT 20*(oppdevint1|(i|)),50,interv$(oppdevint1|(i|))
  NEXT i|
RETURN
```

JEU DE DEVINETTE D'ACCORD

Le programme en général

Le programme commence par une initialisation des tableaux d'accords et leurs renversements. Bien que ce tableau se nomme **RENV$()**, il ne contient que les positions fondamentales des accords. Les positions des degrés des accords sont codées à partir de la note de base ou fondamentale de l'accord. Trois positions d'accords de trois notes sont aussi codées. En plus des accords répertoriés précédemment, se trouve l'accord de septième mineure qui est formé sur le second degré de la gamme majeure, de tierce en tierce, soit les degrés 2,4,6,8 (ou les notes Ré Fa La Do). Les intervalles en demi-tons notés à partir de la note de base sont: 3,7,10. Les codages dans le programme se font par l'intermédiaire du tableau chaîne de caractères **renv$()** à partir du code ASCII de **0(48)**.

La boucle principale se trouve dans le **DO LOOP** et teste simplement les clics de souris, suivant la même logique que le jeu sur les intervalles.

Les procédures sont définies en fin de programme et peuvent facilement être mêlées aux procédures du jeu précédent par un merge, si vous désirez réaliser un menu pour sélectionner l'un ou l'autre des deux jeux.

Le programme en détail

La particularité de l' initialisation est de séparer en deux parties les niveaux et les références des codages des accords. Les indices 0,1,5,6 correspondent aux accords Maj, Min, 7 et min7 et au niveau 1 du jeu, ce jugement étant corrélé à l'utilisation prépondérante de ces accords dans la musique la plus répandue. Il est aisé de modifier ce choix en prenant garde à ce que les deux ensembles de niveaux soient disjoints. Le jeu commencera ensuite en positionnant les indicateurs logiques du jeu(**gen!**) et de sa réponse(**touvacc!**) sur la valeur FALSE et le niveau du jeu sur **1** par défaut.

Le menu succint de présentation ainsi que le tempo de départ sont ensuite affichés. La boucle principale effectue principalement l'appel des procédures de génération du jeu (**generer**), d'analyse de la réponse (**analyseaccord**) et de reprise de l'audition de l'exemple (**rejouer**); elle repositionne en outre les indicateurs de niveaux en réinitialisant les variables logiques précitées.

Les procédures principales

generer

Cette procédure initialise le jeu côté ordinateur et instaure un tirage au hasard sur un degré quelconque permis; le compteur de réponses pour le tirage qui va être effectué est remis à 0. Seules quatre valeurs d'indices sont autorisées pour le degré 1 et le tempo instauré pour arpéger les notes de l'accord est plus lent que pour le niveau 2. La note de départ est choisie aléatoirement sur l'octave supérieure au Do3. Les notes correspondantes sont ensuite jouées par la procédure **rejouer()**.

rejouer

Si le jeu ordinateur est initialisé, les codes ASCII du codage de l'accord sont décryptés dans la variable **renv$()**, en corrigeant ce code par la soustraction du code de **0(48)**. Le tempo est inversement proportionnel au temps de pause, et il n'est pas utile pour un jeu simple de vouloir un tempo aussi précis que sur un séquenceur.

analyseaccord

La réponse de l'utilisateur est initialisée sur FALSE par défaut et son nombre de tentatives incrémenté. Si la case sur laquelle il a cliqué est bonne, cela correspond à l'indice choisi puisque les accords sont présentés dans le même ordre que leur affichage. Lorsque la réponse est mauvaise, le compteur de mauvaises réponses est incrémenté et comparé à la valeur maximum autorisée pour le niveau correspondant. Dans le cas où cette limite est dépassée, la solution est affichée. Par une extrême bonté de votre Atari et par l'intermédiaire de la procédure **il-fallait**, vous avez tout de même l'opportunité de réentendre l'exemple original et votre choix afin de les comparer. Dans cet exemple, la sortie audio se fait par l'intermédiaire du Midi, pour varier un peu les plaisirs, mais vous aurez très prochainement une merveilleuse procédure unique qui vous permettra de sortir indifféremment sur le Midi ou le générateur sonore avec les mêmes codes de hauteur et vélocité.

```
' LISTING DE DEVINE ACCORD
'
DIM accord$(7),renv$(7),nivdevacc1|(3),opnivacc1|(3)
FOR i|=0 TO 3
  READ nivdevacc1|(i|)
NEXT i|
DATA 0,1,5,6
FOR i|=0 TO 3
  READ opnivacc1|(i|)
NEXT i|
DATA 2,3,4,7
RESTORE acc
FOR i|=0 TO 7
```



(suite)

```
  READ accord$(i|)
  EXIT IF accord$(i|)="FACC"
NEXT i|
acc:
DATA MAJ,MIN
DATA AUG,SU4,DIM,7,m7,7dm,FACC
RESTORE renv
FOR i|=0 TO 4
  READ renv$(i|)
  EXIT IF renv$(i|)="FRENV"
NEXT i|
renv:
DATA 47,37,48,57,36,FRENV
renv$(5)=CHR$(52)+CHR$(55)+CHR$(58)
renv$(6)=CHR$(51)+CHR$(55)+CHR$(58)
renv$(7)=CHR$(51)+CHR$(54)+CHR$(57)
' ------------------------------------
TEXT 200,20,"DEVINETTE D'ACCORDS "
aff
trouvacc!=FALSE
gen!=FALSE
nivdevacc1|=1
TEXT 280,50,"NIVEAU"
TEXT 300,70,nivdevacc1|
TEXT 350,50,"JOUER"
TEXT 420,50,"ECOUTER"
TEXT 420,70,"MIDI 1"
TEXT 500,50,"QUITTER"
tempo|=50
DO
  MOUSE x&,y&,k|
  IF k|=1
    FOR i|=0 TO 7
      IF x&>30*i| AND x&<30*i|+20 AND y&<50 AND y&>40
        @analyseaccord
      ENDIF
    NEXT i|
    IF x&>300 AND x&<350 AND y&>60 AND y&<70
      IF nivdevacc1|=1
        nivdevacc1|=2
        TEXT 300,70,nivdevacc1|
        genere!=FALSE
        aff2
      ENDIF
    ENDIF
    IF x&>350 AND x&<400 AND y&>40 AND y&<50
      @generer
    ENDIF
```

(suite)

```
    IF x&>420 AND x&<480 AND y&>40 AND y&<50
      @rejouer(r|)
    ENDIF
    IF x&>s00 AND x&<550 AND y&>40 AND y&<50
      @quitter
    ENDIF
  ENDIF
  IF x&>300 AND x&<350 AND y&>60 AND y&<70 AND k|=2
    IF nivdevacc1|=2
      nivdevacc1|=1
      TEXT 300,70,nivdevacc1|
      genere!=FALSE
      aff1
    ENDIF
  ENDIF
LOOP
PROCEDURE analyseaccord
  IF genere!=TRUE
    INC test|
    trouvacc!=FALSE
    IF i|=r|
      trouvacc!=TRUE
    ENDIF
    TEXT 200,300,"                "
    IF trouvacc!=TRUE
      TEXT 200,300,"c'est tout bon"
    ELSE
      INC testjeuacc|
      IF testjeuacc|<4-nivdevacc1|
        TEXT 200,300,"ce n'est pas tout bon"
        @ilfallait
      ELSE
        TEXT 200,300,"LA REPONSE ETAIT"
        TEXT 400,300,accord$(r|)
      ENDIF
    ENDIF
  ELSE
    TEXT 200,300," CLIQUEZ D'ABORD SUR JOUER"
  ENDIF
RETURN
PROCEDURE generer
  testjeuacc|=0
  TEXT 200,200,"CLIQUEZ DANS LA CASE DE l'ACCORD ENTENDU"
  TEXT 200,300,"                "
  genere!=TRUE
  SELECT nivdevacc1|
  CASE 2
    r|=RANDOM(8)
```



(suite)

```
    tempo|=40
  CASE 1
    r|=nivdevacc1|(RANDOM(4))
    tempo|=50
  ENDSELECT
  notdep|=60+RANDOM(13)
  PAUSE 50
  @rejouer(r|,tempo|)
RETURN
PROCEDURE rejouer(rer|,temps|)
  IF genere!=TRUE
    OUT 3,144,notdep|,80
    PAUSE temps|
    OUT 3,notdep|,0
    FOR t|=1 TO LEN(renv$(rer|))
      OUT 3,notdep|+ASC(MID$(renv$(rer|),t|,1))-48,80
      PAUSE temps|
      OUT 3,144,notdep|+ASC(MID$(renv$(rer|),t|,1))-48,0
    NEXT t|
  ELSE
    TEXT 200,300,"                          "
    TEXT 200,300," CLIQUEZ D'ABORD SUR JOUER"
  ENDIF
RETURN
PROCEDURE quitter
  EDIT
RETURN
PROCEDURE ilfallait
  PAUSE 100
  TEXT 200,300,"                                    "
  TEXT 200,300,"ECOUTEZ L'ACCORD QUE VOUS AVEZ SELECTIONNE"
  PAUSE 100
  rejouer(i|,tempo|-15)
  TEXT 200,300,"                                        "
  TEXT 200,300,"ECOUTEZ L'ACCORD QUE VOUS DEVEZ TROUVER"
  PAUSE 100
  @rejouer(r|,tempo|)
  TEXT 200,300,"                                    "
RETURN
PROCEDURE aff
  DEFFILL 1,0,0
  FOR i|=0 TO 7
    BOX 30*i|,35,30*(i|+1),55
    TEXT 30*i|+2,50,accord$(i|)
  NEXT i|
  aff1
RETURN
PROCEDURE aff1
```

(suite)

```
  DEFFILL 1,2,2
  FOR i|=0 TO 3
    PBOX 30*opnivacc1|(i|)+1,35,30*(opnivacc1|(i|)+1),55
    TEXT 30*opnivacc1|(i|)+2,50,accord$(opnivacc1|(i|))
  NEXT i|
RETURN
PROCEDURE aff2
  DEFFILL 1,0,0
  FOR i|=0 TO 3
    PBOX 30*opnivacc1|(i|)+1,35,30*(opnivacc1|(i|)+1),55
    TEXT 30*opnivacc1|(i|)+2,50,accord$(opnivacc1|(i|))
  NEXT i|
RETURN
```

QUIZZ INTERVALLES

Le programme en général

Ce programme propose un choix d'intervalles que l'on demande à l'utilisateur de jouer. La réponse est analysée et traitée en conséquence. Comme dans les programmes précédents, l'initialisation effectue le remplissage des tableaux et variables, et la correspondance des niveaux. Toutes les valeurs possibles d'intervalles sur une octave sont rentrées, de 0 à 12 demi-tons avec l'appellation correspondante d'intervalles.

Les valeurs d'indices sélectionnées pour le niveau 1 sont 0,3,4,5,7,12 qui correspondent aux intervalles théoriquement les plus faciles à reconnaître (unisson,tierce mineure et majeure, quarte et quinte justes, septième de dominante et octave juste). Le jeu est positionné par défaut sur le niveau **1** et la variable **geneint!** sur la valeur **FALSE** tant que le jeu n'aura pas commencé.

Le programme en détail

Le corps du programme procède comme dans les exemples précédents à l'intérieur du DO LOOP.

Les procédures spécifiques à ce programme sont **geneint** et **enreint** qui vont créer et jouer un intervalle pour la première et enregistrer et analyser la réponse pour la seconde.

geneint

Cette procédure initialise le jeu côté ordinateur par la variable logique **geneint!**, puis choisit un codage d'intervalle au hasard en correspondance avec le niveau. Ce codage est mémorisé dans la variable **questinterv$**. Il appartient maintenant à l'utilisateur de jouer l'intervalle demandé. Si on reclique sur la case **JOUER**, un nouvel intervalle est demandé, mais il suffirait de chaîner la procédure **enreint** à la présente pour se voir interdire ce choix et être obligé de répondre à une question épineuse!



enreint

C'est la procédure principale du programme. Après s'être assuré que le jeu est bien lancé, le buffer Midi est remis à zéro et la variable **harmo$** destinée à recevoir la chaîne résultat est initialisée en chaîne vide. Le programme fait fi du *running status* éventuel du codage puis termine l'enregistrement des données de **NoteOn** par un retour chariot ou deux notes jouées. Dans le cas où deux notes ont été enregistrées, on a bien affaire à un intervalle, et on peut se brancher sur la procédure du même nom qui va tester le résultat. Si une seule note a été jouée ou si plus de deux notes ont été jouées (il faudrait pour ce dernier supprimer le OR **LEN(harmo$)=2**) en fin d'enregistrement, on passe aux messages d'erreurs adéquats.

intervalle

Cette procédure teste la valeur absolue du nombre de demi-tons séparant les deux notes enregistrées. Les traitements des intervalles ascendants et descendants ne sont pas distincts ici. Il suffit ensuite de regarder si l'intervalle enregistré se retrouve dans **interv$()** et correspond à **interv$**. Si la valeur absolue de l'intervalle est supérieure à 12, le message intervalle trop grand apparaît, puisque seuls sont traités les intervalles sur une octave.

```
' LISTING DE JEU INTERVALLES
DIM interv$(12),intervalle|(12),nivjeuint|(5)
FOR i|=0 TO 12
  READ interv$(i|)
NEXT i|
FOR i|=0 TO 5
  READ nivjeuint|(i|)
NEXT i|
DATA unisson,sec_min,sec_MAJ
DATA tce_min,tce_MAJ,quarte-juste
DATA quinte_dim,quinte_juste
DATA quinte_aug,sixte,sept_min
DATA sept_MAJ,octave
DATA 0,3,4,5,7,12
nivjeuint|=1
TEXT 300,20,"JEU INTERVALLES"
TEXT 10,50,"NIVEAU"
TEXT 10,70,nivjeuint|
TEXT 100,50,"JOUER"
TEXT 200,50,"ENREGISTRER"
TEXT 300,50,"QUITTER"
TEXT 400,50,"SON"
TEXT 400,70,"MIDI 1"
TEXT 300,200,"VOTRE INT"
TEXT 150,300,"JOUEZ DEUX NOTES SUR UNE OCTAVE"
TEXT 200,310,"L'INTERV DE"
geneint!=FALSE
DO
  MOUSE x&,y&,k|
```

(suite)

```
  IF k|=1
    IF x&>10 AND x&<50 AND y&>60 AND y&<70
      IF nivjeuint|=1
        geneint!=FALSE
        nivjeuint|=2
        TEXT 10,70," "
        TEXT 10,70,nivjeuint|
        TEXT 200,350,"              "
      ENDIF
    ENDIF
    IF x&>100 AND x&<180 AND y&>40 AND y&<50
      @geneint
    ENDIF
    IF x&>200 AND x&<280 AND y&>40 AND y&<50
      @enreint
    ENDIF
    IF x&>300 AND x&<380 AND y&>40 AND y&<50
      @quitter
    ENDIF
  ENDIF
  IF k|=2
    IF x&>10 AND x&<50 AND y&>60 AND y&<70
      IF nivjeuint|=2
        geneint!=FALSE
        nivjeuint|=1
        TEXT 10,70,""
        TEXT 200,350,"                "
        TEXT 10,70,nivjeuint|
      ENDIF
    ENDIF
  ENDIF
LOOP
PROCEDURE enreint
  IF geneint!=TRUE
    clav|=0
    TEXT 400,200,"                "
    TEXT 400,200,"C'EST A VOUS DE JOUER"
    harmo$=""
    SLPOKE XBIOS(14,2)+6,0
    SLPOKE XBIOS(14,1)+6,0
    REPEAT
      IF BIOS(1,3)
        a|=INP(3)
        IF a|<128
          IF note!=TRUE
            b|=inp(3)
            ' si note on, on rajoute dans la chaine
            IF commande|>143 AND b|>0
```



(suite)

```
            harmo$=harmo$+CHR$(a|)
          ENDIF
        ENDIF
      ELSE
        IF a|>127 AND a|<160
          note ! =TRUE
          commande|=a|
        ELSE
          IF a|<240
            note!=FALSE
          ENDIF
        ENDIF
      ENDIF
    ENDIF
    IF BIOS(1,2)
      clav|=lnp(2)
    ENDIF
  UNTIL clav|=13 OR LEN(harmo$)=2
  IF LEN(harmo$)=2
    @intervalle
  ELSE
    IF LEN(harmo$)<2
      TEXT 400,200,"           "
      TEXT 400,200," PAS ASSEZ DE NOTES "
    ELSE
      TEXT 400,200,"           "
      TEXT 400,200," TROP DE NOTES "
    ENDIF
  ENDIF
ELSE
  TEXT 400,200,"                  "
  TEXT 400,200," CLIQUEZ D'ABORD SUR JOUER"
ENDIF
RETURN
PROCEDURE intervalle
  test!=FALSE
  interv|=ABS(ASC(harmo$)-ASC(RIGHT$(harmo$)))
  IF interv|<13
    interv$=interv$(interv|)
    IF interv$=questinterv$
      TEXT 400,200,"                  "
      TEXT 400,200,"SUPER!"
    ELSE
      FOR i|=0 TO 12
        IF interv$=interv$(i|)
          test!=TRUE
          EXIT IF test!=TRUE
        ENDIF
```

(suite)

```
      NEXT i|
      TEXT 400,200,"                "
      TEXT 400,200,interv$(i|)
    ENDIF
  ELSE
    TEXT 400,200,"                  "
    TEXT 400,200," INTERV TROP GRAND"
  ENDIF
RETURN
PROCEDURE geneint
  geneint!=TRUE
  TEXT 400,200,"                          "
  SELECT nivjeuint|
  CASE i|
    i|=RANDOM(6)
    questinterv$=interv$(nivjeuint|(i|))
  CASE 2
    i|=RANDOM(13)
    questinterv$=interv$(i|)
  ENDSELECT
  TEXT 200,350,"              "
  TEXT 200,350,questinterv$
RETURN
PROCEDURE quitter
  EDIT
RETURN
```

QUIZZ ACCORDS

Le programme en général

Ce programme correspond à l'extension du programme sur les intervalles. Suivant que les accords générés et attendus sont à trois ou quatre notes, il faudra analyser deux ou trois intervalles en même temps. Le début du programme sépare les codages des accords et leurs renversements suivant le nombre de notes demandées. Les renversements ne sont rentrés qu'en vue de l'analyse et non au cours de la génération des exemples. Les tableaux de niveaux sont aussi initialisés en fonction du nombre de degrés des accords(3 et 4). A l'opposé des intervalles, et du jeu de devinette sur les accords, les codages sont entrés par succession d'intervalles de tierces, à vous de choisir le codage qui vous convient le mieux : soit à partir de la fondamentale ou note la plus grave de l'accord, soit à partir du degré précédent. Les tableaux **renv$()** codent respectivement les suites d'intervalles et les tableaux **accords$()** codent les appellations des accords.



Le programme en détail

Comme dans le programme sur le Quizz d'intervalles, le corps du traitement se fait dans le DO LOOP et au moyen de deux procédures: geneacc et enreacc qui proposent un accord, et qui l'enregistrent. Les changements de niveaux et la régénération d'accords se font de la même façon et avec la même bienveillance que pour les intervalles.

Les procédures principales

geneacc

Cette procédure initialise le jeu (**geneacc!** = TRUE). La main est passée au choix aléatoire de l'accord à trouver, en correspondance avec le niveau et le nombre de notes de l'accord, puisque certains accords de trois notes sont du niveau 2 et certains accords de quatre notes sont du niveau 1. Ce choix s'effectue par l'intermédiaire de la procédure **choix()**. L'accord à trouver est codé dans les variables **questrenv$** et **questacc$**.

enreacc

Cette procédure se charge d'enregistrer l'accord joué par l'utilisateur. Si le jeu est commencé, le filtrage du *running status* est le même que pour les intervalles. La variable **harmo$** code aussi les hauteurs jouées jusqu'à activation de la touche **return** ou nombre de notes jouées inférieures au nombre de notes de l'accord plus un. Si le nombre de notes jouées est compris entre 3 et 4, on peut analyser l'accord éventuel par la procédure renversement, sinon des messages d'erreurs sont affichés suivant le nombre trop grand ou trop petit de notes jouées.

range

C'est la première étape de la procédure renversement puisque les notes n'ont pas été forcément jouées dans l'ordre ascendant. Il faudrait supprimer les doublons et les octaves pour ne garder dans **renv$** que les notes à signification harmonique éventuelle(de codes de hauteur différents à modulo 12). Comme nous sommes très exigeants, nous demandons à l'utilisateur de jouer exactement l'accord demandé sur la position requise, ce qui demande aussi moins d'efforts de programmation.

renversement

La variable **test!** est positionnée sur TRUE dès que l'on trouve une égalité entre **renv$** et un élément de **renv$()**. Si aucun codage de suite d'intervalles ne correspond (avec la correction par rapport au code ASCII de 0, 48), le message accord non répertorié est affiché. Pour être complètement rigoureux, il faudrait analyser complètement l'accord enregistré pour en donner peut être une appellation et un renversement possibles, la notion de non répertorié évoquant peut-être trop facilement l'idée de non existence de l'accord joué.

```
' LISTING DE JEU ACCORD
'
DIM accord3$(12),renv3$(12)
DIM accord4$(8),renv4$(8),niv3|(4),niv4|(2)
FOR i|=0 TO 4
  READ niv3|(i|)
NEXT i|
DATA 0,3,6,7,10
FOR i|=0 TO 2
```

(suite)

```
  READ niv4|(i|)
NEXT i|
DATA 0,4,8
RESTORE acc3
FOR i|=0 TO 12
  READ accord3$(i|)
  EXIT IF accord3$(i|)="FACC3"
NEXT i|
acc3:
DATA MAJ,MAJ_1erRenv,MAJ_2èmeRenv
DATA MIN,Min_1erRenv,MIN_2èmeRenv
DATA AUG,SUS4 SUS4_1erRenv,SUS4_2èmeRenv
DATA DIM,DIM_1erRenv,DIM_2èmeRenv,FACC3
RESTORE renv3
FOR i|=0 TO 12
  READ renv3$(i|)
  EXIT IF renv3$(i|)="FRENV3"
NEXT i|
renv3:
DATA 43,35,54,34,45,53,44,52,25,55
DATA 33,36,63,FRENV3
RESTORE acc4
FOR i|=0 TO 8
  READ accord4$(i|)
  EXIT IF accord4$(i|)="FACC4"
NEXT i|
acc4:
DATA 7,7_1erR,7_2èmeR,7_3èmeR
DATA m7,m7_1erR,m7_2èmeR,m7_3èmeR,7dim,FACC4
RESTORE renv4
FOR i|=0 TO 8
  READ renv4$(i|)
  EXIT IF renv4$(i|)="FRENV4"
NEXT i|
renv4:
DATA 433,332,324,243
DATA 343,432,323,234,333,FRENV4
nivjeuacc|=1
TEXT 300,20,"ACCORDS "
TEXT 10,50,"NIVEAU"
TEXT 10,70,nivjeuacc|
TEXT 100,50,"JOUER"
TEXT 200,50,"ENREGISTRER"
TEXT 300,50,"QUITTER"
TEXT 400,50,"SON"
TEXT 400,70,"MIDI 1"
TEXT 300,200,"VOTRE ACC"
TEXT 200,300,"JOUEZ UN ACCORD"
```



(suite)

```
TEXT 220,320,"NOTES"
TEXT 180,320,"DE"
geneacc ! =FALSE
DO
  MOUSE x&,y&,k|
  IF k|=1
    IF x&>10 AND x&<50 AND y&>60 AND y&<70
      IF nivjeuacc|=1
        geneacc!=FALSE
        TEXT 200,350,"           "
        nivjeuacc|=2
        TEXT 10,70," "
        TEXT 10,70,nivjeuacc|
      ENDIF
    ENDIF
    IF x&>100 AND x&<180 AND y&>40 AND y&<50
      @geneacc
    ENDIF
    IF x&>200 AND x&<280 AND y&>40 AND y&<s0
      @enreacc
    ENDIF
    IF x&>300 AND x&<380 AND y&>40 AND y&<50
      @quitter
    ENDIF
  ENDIF
  IF k|=2
    IF x&>10 AND x&<50 AND y&>60 AND y&<70
      IF nivjeuacc|=2
        geneacc ! =FALSE
        nivjeuacc|=1
        TEXT 200,350,"           "
        TEXT 10,70," "
        TEXT 10,70,nivjeuacc|
      ENDIF
    ENDIF
  ENDIF
LOOP
PROCEDURE enreacc
  IF geneacc!=TRUE
    clavl=0
    TEXT 400,200,"               "
    TEXT 400,200,"C'EST A VOUS DE JOUER"
    harmo$=""
    SLPOKE XBIOS(14,2)+6,0
    SLPOKE XBIOS(14,1)+6,0
    REPEAT
      IF BIOS(1,3)
        a|=INP(3)
```

(suite)

```
        IF a|<128
          IF note!=TRUE
            b|=INP(3)
            ' si note on, on rajoute dans la chaine
            IF commande|>143 AND b|>0
              harmo$=harmo$+CHR$(a|)
            ENDIF
          ENDIF
        ELSE
          IF a|>127 AND a|<160
            note ! =TRUE
            commande|=a|
          ELSE
            IF a|<240
              note!=FALSE
            ENDIF
          ENDIF
        ENDIF
      ENDIF
      IF BIOS(1,2)
        clav|=INP(2)
      ENDIF
    UNTIL clav|=13 OR LEN(harmo$)=LEN(questrenv$)+1
    IF LEN(harmo$)>2 AND LEN(harmo$)<5
      @renversement
    ELSE
      IF LEN(harmo$)<3
        TEXT 400,200,"                "
        TEXT 400,200," PAS ASSEZ DE NOTES "
      ELSE
        TEXT 400,200,"                "
        TEXT 400,200," TROP DE NOTES "
      ENDIF
    ENDIF
  ELSE
    TEXT 400,200,"                          "
    TEXT 400,200," CLIQUEZ D'ABORD SUR JOUER"
  ENDIF
RETURN
PROCEDURE range
  REPEAT
    k!=TRUE
    FOR i|=1 TO LEN(harmo$)
      FOR j|=1 TO i|-1
        IF MID$(harmo$,j|,1)>MID$(harmo$,i|,1)
          k!=FALSE
          stock$=MID$(harmo$,j|,1 )
          MID$(harmo$,j|,1)=MID$(harmo$,i|,1)
```



(suite)

```
          MID$(harmo$,i|,1)=stock$
        ENDIF
      NEXT j|
    NEXT i|
  UNTIL k!=TRUE
RETURN
PROCEDURE renversement
  @range
  renv$=""
  FOR i|=2 TO LEN(harmo$)
    j|=ASC(MID$(harmo$,i|))-ASC(MID$(harmo$,i|-1))
    renv$=renv$+CHR$(j|+48)
  NEXT i|
  ' IF renv$=questrenv$
  TEXT 400,200,""
  TEXT 400,200,"super! a changer"
  ' ELSE
  test!=FALSE
  SELECT LEN(harmo$)
  CASE 3
    FOR i|=0 TO 12
      IF renv$=renv3$(i|)
        TEXT 400,200,"                    "
        TEXT 400,200,accord3$(i|)
        test!=TRUE
      ENDIF
      EXIT IF test!=TRUE
    NEXT i|
  CASE 4
    FOR i|=0 TO 8
      IF renv$=renv4$(i|)
        TEXT 400,200,"                      "
        TEXT 400,200,accord4$(i|)
        test!=TRUE
      ENDIF
      EXIT IF test!=TRUE
    NEXT i|
  ENDSELECT
  IF test!=FALSE
    TEXT 400,200,"                  "
    TEXT 400,200," NON REPERT "
  ENDIF
  ' ENDIF
RETURN
PROCEDURE geneacc
  geneacc!=TRUE
  TEXT 400,200,"                          "
  SELECT nivjeuacc|
```

(suite)

```
  CASE 1
    @choixacc(2,2)
  CASE 2
    @choixacc(5,3)
  ENDSELECT
  TEXT 200,320," "
  TEXT 200,320,LEN(questrenv$)+1
  TEXT 200,350,"              "
  TEXT 200,350,questacc$
RETURN
PROCEDURE blanc
  TEXT 400,200,"                  "
  TEXT 200,300,"              "
  TEXT 200,350,"              "
RETURN
PROCEDURE quitter
  EDIT
RETURN
PROCEDURE choixacc(ert1|,ert2|)
  SELECT RANDOM(2)
  CASE 0
    i|=RANDOM(ert1|)
    questacc$=accord3$(niv31(i|))
    questrenv$=renv3$(niv31(i|))
  CASE 1
    i|=RANDOM(ert2|)
    questacc$=accord4$(niv41(i|))
    questrenv$=renv4$(niv41(i|))
  ENDSELECT
RETURN
```

6/2

L'environnement MIDI

6/2.1

Généralités

Il convient pour décrire complètement l'environnement MIDI de définir d'une part l'interface MIDI et d'autre part la norme MIDI. La première étant construite dans le respect de la deuxième.

L'interface MIDI (*Musical Instrument Digital Interface*), qui équipe l'Atari ST, a été conçue pour interconnecter des instruments de musique tels que ordinateurs, synthétiseurs, boîtes à rythmes, etc.

Cette interface est constituée d'un émetteur et d'un récepteur fonctionnant en régime asynchrone de la même manière que l'interface série RS232 qui équipe l'Atari ST. Par contre, la vitesse de transmission est fixée à 31.25 Kbaud et les données sont composées de 10 bits (1 bit de start, 8 bits de données et 1 bit de stop). Cela correspond donc à 320 micro-secondes pour un octet transmis.

La norme MIDI utilise, pour transférer des informations, des messages composés de plusieurs octets. D'une manière générale, un message est composé d'un premier octet de status suivi par un ou deux octets de données. Les deux exceptions à ce format sont les messages exclusifs et les messages temps réel.

6/2.2

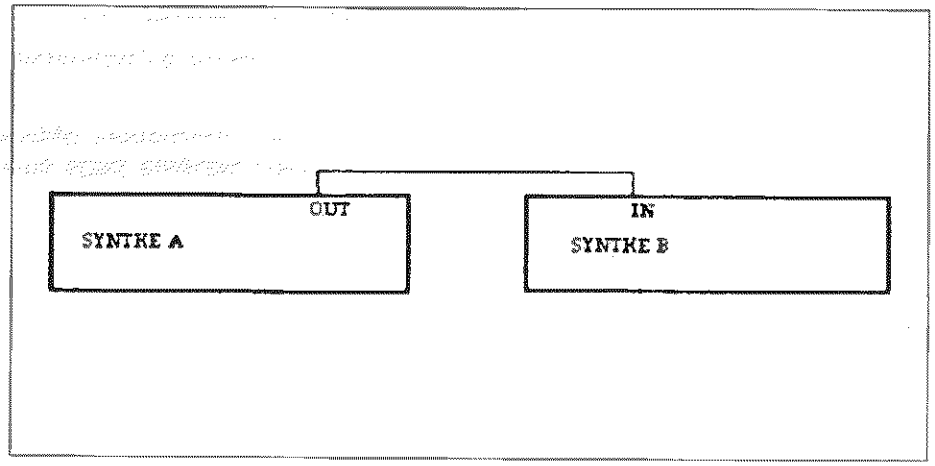
Le câblage MIDI

Avant de décrire les codes MIDI les plus usités, des notions de câblage sont nécessaires. Le câblage MIDI s'effectue à l'aide de trois prises, IN, OUT et THRU. IN et OUT sont facilement compréhensibles, OUT émet les informations de l'instrument MIDI, alors que IN reçoit des informations en provenance d'un autre appareil. Quant à la prise THRU, son rôle est de renvoyer toutes les informations reçues par IN. En aucun cas THRU n'engendre ni ne filtre les codes MIDI. THRU est entièrement dépendant de IN, c'est une dérivation passive de IN.

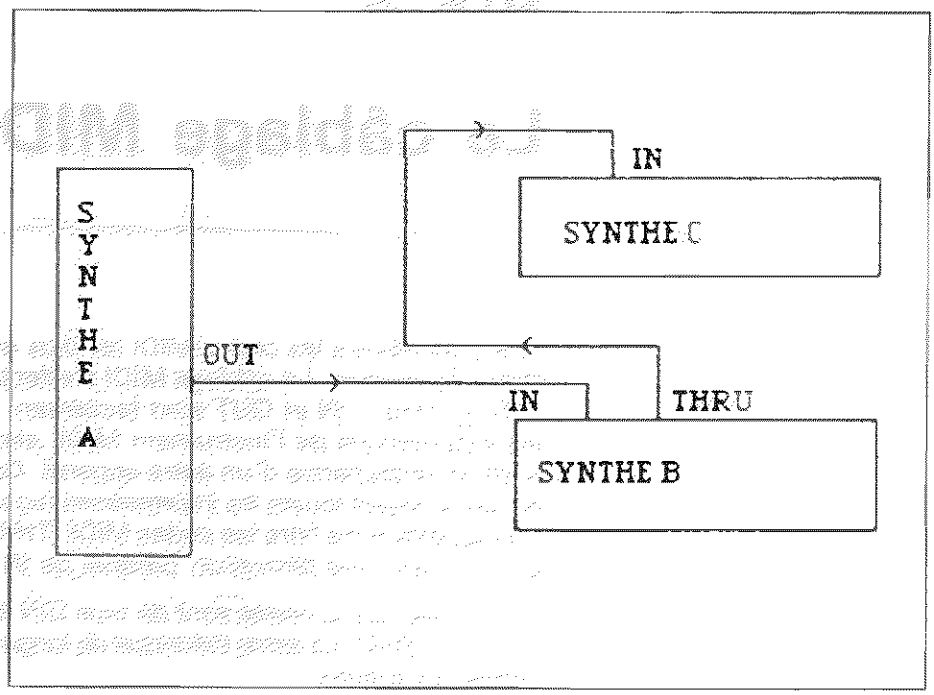
Les connecteurs utilisés sont de type DIN à 5 broches, identiques à ceux utilisés en HI-FI. La limite théorique de longueur de câble à ne pas dépasser est de 15 mètres.

Voici quelques exemples de connexions MIDI :

— Le A pilote le B



— Le A pilote le B et le C

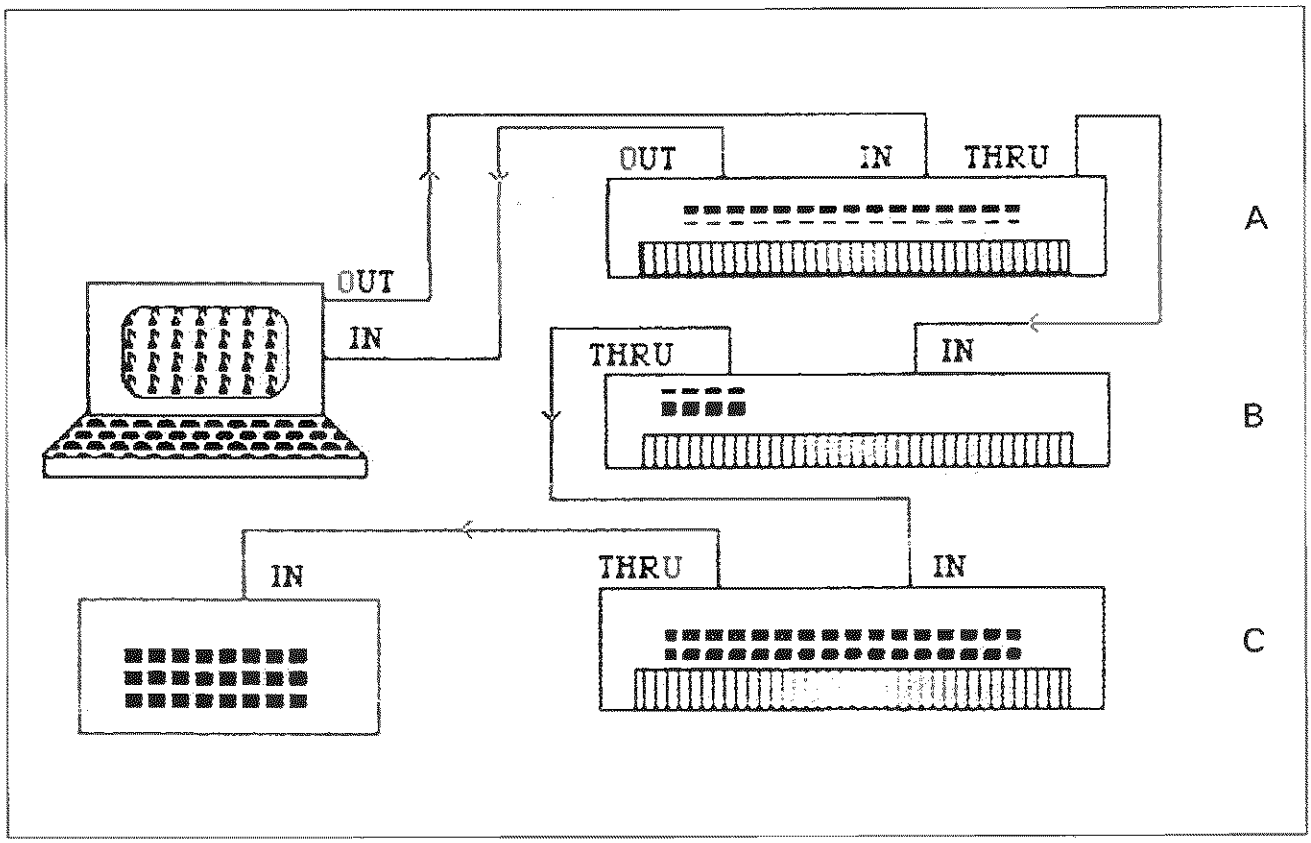


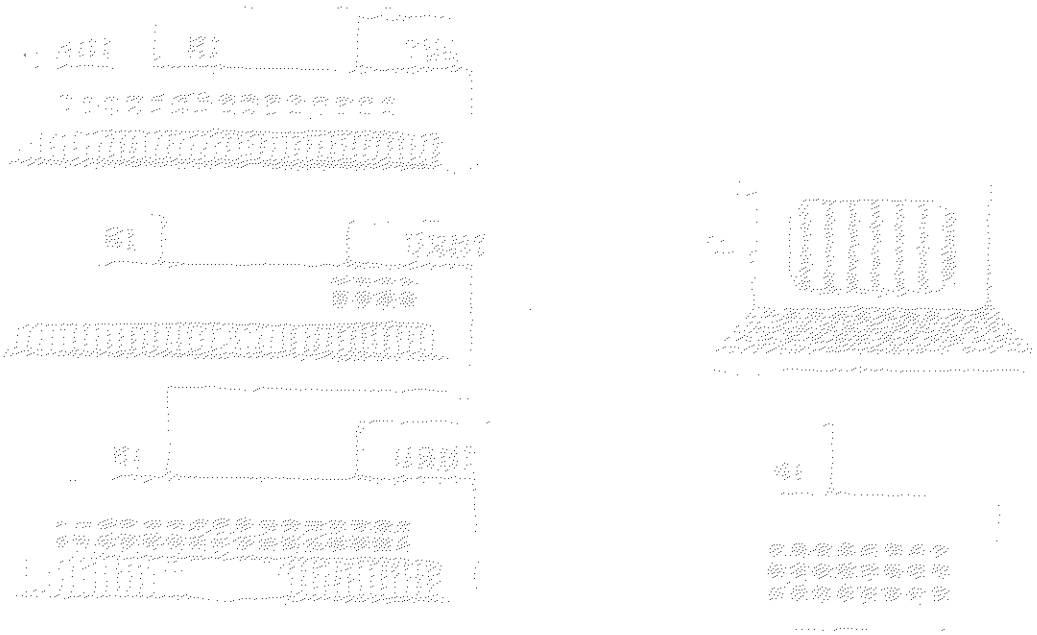
— Connexion entre 4 synthétiseurs et un ordinateur.

Deux circuits distincts apparaissent :

Le A envoie les informations à l'ordinateur, phase d'enregistrement des données.

L'ordinateur envoie les informations MIDI au A et au B, phase de restitution des données (voir schéma page suivante).


OUT
IN
THRU
A
OUT
IN
THRU
IN
B
IN
THRU
IN
C



6/2.3

Les codes MIDI

Le système MIDI permet de gérer plusieurs appareils simultanément, que ce soient des synthétiseurs, des boîtes à rythmes, des séquenceurs, en fait tout système pourvu d'une interface MIDI. La sélection des informations s'effectue par les 16 canaux MIDI. Si des informations sont émises sur le canal 5, seuls les appareils en réception sur ce canal interpréteront les informations. De ce fait, le système MIDI peut gérer jusqu'à 16 sources sonores indépendantes. Plusieurs appareils peuvent aussi être réglés sur le même canal MIDI de réception, et, dans ce cas, ils joueront tous les mêmes événements simultanément.

Les codes utilisés par le système MIDI sont répartis en 5 groupes :

— message canal
— message modes
— message temps réel
— message système commun
— message système exclusif.

Tout message MIDI est au moins fait d'une commande, qui peut être suivie de une ou deux données suivant le type de message. Seuls les messages de système exclusif, qui sont la zone libre allouée aux constructeurs, échappent à cette règle, le nombre de données envoyées étant lié au type de l'appareil et à la fonction recherchée.

La commande la plus simple pour aborder le système MIDI appartient au message canal. Cette commande déclenche une note, puis l'extinction de cette note. Pour exécuter cette opération, nous avons besoin de deux codes commande distincts :

1) *Déclenchement de la note*

Pour déclencher une note, plusieurs informations doivent parvenir au synthétiseur.

• La première information est le code NOTE ON (déclencher une note) qui est une commande. La valeur du code de cette commande varie de 144 à 159. Ce code inclut le numéro de canal MIDI (de 0 à 15, soit 16 canaux).

Important :

Les exemples suivants utilisent la numérotation de 1 à 16 alors que le MIDI les comprend de 0 à 15.

144 → NOTE ON canal 1
145 → NOTE ON canal 2
146 → NOTE ON canal 3
159 → NOTE ON canal 16

Le codage est plus compréhensible en binaire. Les quatre bits de poids fort correspondent à la commande alors que les 4 bits de poids faible indiquent le numéro de canal MIDI.

1001 0000 NOTE ON canal 1
1001 0001 NOTE ON canal 2
1001 1111 NOTE ON canal 16

• La deuxième information est une donnée, c'est le numéro de la note à exécuter. Le codage des notes en MIDI s'échelonne de 0 à 127. La référence est le DO 3 qui se trouve au milieu du clavier et qui porte le code 60. La numérotation des notes s'effectue par demi-ton.

DO — 60
DO# — 61
RE — 62
RE# — 63
MI — 64

Le système MIDI ne sait gérer que les demi-tons et ne fait aucune différence entre un DO dièse et un RE bémol.

• La troisième information indispensable est la dynamique de la note, c'est la force (en fait c'est souvent la vitesse qui est analysée) avec laquelle vous avez enfoncé la touche. Cette donnée est codée de 0 à 127. En cas d'utilisation d'un appareil non dynamique, cette valeur est envoyée à 64 par défaut.

Pour déclencher une note il faut donc transmettre :

NOTE ON	144 à 159	commande
N° de la note	0 à 127	donnée
Dynamique	0 à 127	donnée

2) *Désactiver la note*

Pour désactiver une note, il nous faut également trois codes.

• La commande NOTE OFF. Le code de cette commande s'échelonne de 128 à 143 de la même manière que pour le NOTE ON.

128 → NOTE OFF canal 1
143 → NOTE OFF canal 16

• La deuxième information est une donnée, c'est le numéro de la note codée de 0 à 127.

• La troisième information est la vitesse de relâchement de la touche. Certains appareils sont capables d'interpréter des effets différents suivant le relâchement. Cette donnée est codée de 0 à 127, dans la majeure partie des cas ce code n'est pas utilisé et prend la valeur 64 par défaut.



Pour désactiver une note il faut donc transmettre :

NOTE OFF	128 à 143	commande
N° de la note	0 à 127	donnée
Dynamique	0 à 127	donnée

Les codes commandes sont toujours supérieurs à 127, alors que les données sont toujours inférieures à 128.

Pour la programmation en GFA, l'instruction OUT 3,n permet de sortir une valeur d'octet sur le port MIDI de l'Atari. Si l'on désire envoyer une note DO sur le canal 3, puis désactiver cette note, il faut programmer ainsi :

```
OUT 3, 147      note on canal 3
OUT 3, 60       do
OUT 3, 127      dynamique maximum
PAUSE 50        attente avant suite
OUT 3, 131      note off canal 3
OUT 3, 60       do
OUT 3, 64       pas de dynamique
```

Dans le cas d'un accord dont chacune des notes est jouée par un synthétiseur différent, il faut programmer ceci :

Accord de base DO-MI-SOL

```
OUT 3, 144      note on canal 1
OUT 3, 60       DO
OUT 3, 64       dynamique
OUT 3, 145      note on canal 2
OUT 3, 64       MI
OUT 3, 64       dynamique
OUT 3, 146      note on canal 3
OUT 3, 67       SOL
OUT 3, 64       dynamique
PAUSE 50
OUT 3, 128      note off canal 1
OUT 3, 60       DO
OUT 3, 64       dynamique
OUT 3, 129      note off canal 2
OUT 3, 64       MI
OUT 3, 64       dynamique
OUT 3, 130      note off canal 3
OUT 3, 67       SOL
OUT 3, 64       dynamique
```

A chaque NOTE ON envoyé par l'interface MIDI d'un synthétiseur, correspond un NOTE OFF sur le même canal. Le système MIDI étant très gourmand en information, il faut très vite trouver des astuces pour ne pas surcharger la mémoire de l'ordinateur.

```
'
'   -------- Jouer une suite de notes ----------------
'   --------      MIDI et ATARI     ------------------
'
' variable à changer en cas de nouveaux
' remplissage du tableau
r%=25
'
' remplissage du tableau avec les datas
'
DIM note_duree(r%)
FOR a%=0 TO r%
  READ note_duree(a%)
NEXT a%
'
' note MIDI suivie de sa durée
'
DATA 60,24,61,12,62,12,63,24,64,24,65,24,66,24
DATA 67,24,68,12,69,12,70,24,71,24,72,96
'
' traitement du tableau à jouer
'
FOR a%=0 TO r% STEP 2
  OUT 3,144                ! NOTE ON canal 1
  OUT 3,note_duree(a%)     ! n° de note
  OUT 3,RANDOM(127)        ! dynamique alléatoire
  '
  SOUND 1,15,note_duree(a%)-60,3
  '
  PAUSE note_duree(a%+1)   ! durée de la note
  OUT 3,128                ! NOTE OFF canal 1
  OUT 3,note_duree(a%)     ! n° de note
  OUT 3,64                 ! pas de dynamique
  '
  SOUND 1,0,note_duree(a%)-60,3 ! volume à zero
  '
NEXT a%
'
' boucle de sortie sur la barre d'espace
'
DO
  EXIT IF INKEY$=" "
LOOP
```



Nous allons programmer une gamme par demi-ton qui sera interprétée à la fois par le MIDI et par le générateur sonore de l'Atari. Il vous est très facile de changer la hauteur des notes ainsi que leur durée pour programmer vos propres mélodies. La variable r% indique le nombre de données à exécuter.

A la fonction OUT 3,n correspond la fonction d'entrée INP(3). Cette fonction attend un événement sur le port MIDI. Le problème que l'on rencontre le plus souvent en utilisant les premières fois cette fonction est le blocage pur et simple de l'Atari. L'ordinateur n'est pas buggé, il est tout simplement en attente, et seule la réception d'un code MIDI peut rendre la main à l'utilisateur. Pour pallier ce problème, la fonction INP(3) est associée à la fonction INP?(3). INP?(3) teste si une donnée MIDI est disponible en lecture dans le buffer MIDI, mais ne lit pas cette donnée. Si la condition est remplie, INP(3) peut alors lire et traiter cette donnée.

Le programme suivant affiche la note reçue par l'interface MIDI après un NOTE ON sur le canal 1.

```
'
'
'  --- affichage de la note MIDI ---
'
'
DIM note$(11)
FOR a%=0 TO 11
  READ note$(a%)
NEXT a%
'
' nom des notes
' respecter les espaces
'
DATA DO  ,DO# ,RE  ,RE# ,MI  ,FA
DATA FA# ,SOL ,SOL#,LA  ,LA#,SI
'
' traitement des entrées
'
DO
  IF INP?(3)          ! si une donnée MIDI?
    IF INP(3)=144     ! note on canal 1 ?
      '
      '  alors chercher le nom dans note$
      '  avec entrée MIDI suivante
      '  modulo 12
      '
      PRINT AT(38,14);note$(MOD(INP(3),12))
    ENDIF
  ENDIF
  ' sortie par la barre d'espace
  EXIT IF INKEY$=" "
LOOP
```

I - Les codes message canal

Il existe six fonctions différentes dans les codes message canal. Tous les codes de commande incluent obligatoirement le canal MIDI. Les codes s'échelonnent de 128 à 239.

NOTE OFF 128-143

N° de la note 0-127
Dynamique 0-127

Ce code a été décrit précédemment.

NOTE ON 144-159

N° de la note 0-127
Dynamique 0-127

Ce code a été décrit précédemment.

Le code NOTE ON est utilisé par certains synthétiseurs à la place du NOTE OFF. La valeur de la dynamique est alors égale à zéro.

AFTER TOUCH POLY 160-175

N° de la note 0-127
Valeur de l'effet 0-127

L'AFTER TOUCH est souvent traduit par seconde pression sur le clavier. Après l'enfoncement d'une touche il est possible sur certains claviers d'obtenir un effet par simple pression supplémentaire. Le taux de l'effet est en fonction de la pression exercée sur le clavier. L'effet dépend entièrement de la programmation du synthétiseur.

Dans le cas d'un AFTER TOUCH POLY, chacune des notes peut avoir un taux d'effet différent. Les effets les plus couramment utilisés sont la modulation d'amplitude ou la modulation de hauteur.

CONTROL CHANGE 176-191

N° du contrôleur 0-127
Valeur 0-127

Les contrôleurs sont les codes MIDI les plus intéressants à manipuler par le musicien. Les codes NOTE ON ou AFTER TOUCH sont plus faciles à jouer en direct qu'à manipuler code par code, alors que le travail sur les contrôleur peut s'effectuer pas à pas. La manipulation des contrôleurs s'apparente au mixage sonore. Les contrôleurs agissent par exemple sur le volume, le panoramique, la modulation, le sustain. Actuellement seuls 20 codes contrôleurs sont utilisés.

Liste des contrôleurs utilisables :

01 Modulation Wheel
Molette de modulation se trouvant sur tous les synthétiseurs.

02 Breath controller
Contrôleur de souffle. Interface permettant d'actionner des effets à l'aide du souffle, très utilisé pour les vents.

04 Foot controller
Contrôle aux pieds de fonctions propres à chaque synthétiseur.

05 Portamento time
Temps de portamento, temps de liaison entre deux notes.

06 Data entry
Curseur permettant le réglage des paramètres du synthétiseur.

07 Main volume
Volume se situant après le volume du synthétiseur. Si le volume général du synthétiseur est à la moitié, le volume MIDI ne pourra dépasser cette valeur.

08 Balance
La balance existe sur les synthétiseurs où il est possible d'affecter deux sons sur le clavier. La balance représente l'équilibre de volume entre les deux sonorités. L'équilibre entre les deux sons est donné par la valeur 64. Pour favoriser la gauche, la valeur se situe entre 0 et 63, pour favoriser la droite, la valeur se situe entre 65 et 127.

10 Pan
C'est le panoramique, soit la répartition du son dans l'espace entre la droite et la gauche. Réglage identique à la balance. A utiliser uniquement sur les synthétiseurs stéréo.

11 Expression
L'expression permet de jouer un peu plus fortement une partie, c'est une accentuation. Ce paramètre est en relation avec le contrôleur de volume. Si le contrôleur de volume est à 127 (maximum), aucun effet de l'expression n'est possible. Si le contrôleur de volume est à 80, une plage de réglage entre 81 et 127 est disponible.

64 Sustain
Pédale de sustain, c'est le maintien de la note. Lorsque la valeur est à 0, le sustain est nul. Lorsque la valeur est différente de 0, le sustain est activé.

65 Portamento
Pédale de portamento, c'est la mise en/hors fonction du portamento, ne pas confondre avec le taux de portamento, contrôleur n° 05. Les valeurs sont identiques à celles utilisées par le sustain.

66 Sustenuto
Maintien de la première note jouée après l'action du code. Cet effet peut être utilisé pour maintenir un son de basse continu. Les valeurs sont identiques à celles utilisées par le sustain.

67 Soft pedal
Diminution du son par rapport à son volume. Les valeurs sont identiques à celles utilisées par le sustain.

69 Hold 2
Ce contrôleur est un deuxième sustain. Certains synthétiseurs possèdent deux sustains, soit pour deux sonorités différentes, soit pour déclencher le sustain à un autre point du son. Les valeurs sont identiques à celles utilisées par le sustain.

92 Tremolo depth
Réglage du tremolo, variation de l'amplitude du son. La valeur est de 0 à 127.

93 Chorus depth
Réglage de l'effet de chorus de 0 à 127.

94 Celeste depth
Réglage du detune (accord), de 0 à 127.

95 Phaser depth
Réglage de l'effet de phasing, de 0 à 127.

96 Data incrément
C'est le bouton de réglage de paramètres + ou ON.

97 Data décrement
C'est le bouton de réglage de paramètres – ou OFF.

Ce programme vous permet de tester l'effet des différents contrôleurs sur vos synthétiseurs. Attention, les synthétiseurs n'interprètent pas tous les contrôleurs.



```
Print At(21,5);"******* LES CONTROLEURS MIDI *******"
Y%=300
Defline 1,1,1,1
Deffill 0,0
Line 100,Y%,150,Y%
Line 250,Y%,300,Y%
Line 400,Y%,450,Y%
Print At(5,20);"1"
Print At(25,20);"0"
Print At(45,20);"0"
Yi%=300
Yo%=300
Yu%=300
Can%=1
Val%=0
Con%=0
Dim Nom$(127)
For A%=0 To 127
  Read Nom$(A%)
Next A%
Nom_data:
'
'   nom des controleurs
'   respecter le nombre de données
'
Data ,modulation,breath ,,foot control
Data porta time ,data entry,volume
Data balance,,panoramique,expression
Data ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
Data ,,,,,,,,,,,,sustain,portamento
Data sostenuto,soft pedal,,hold 2
Data ,,,,,,,,,,,,,,,,,,,,,,,
Data tremolo,chorus,celeste,phasing,+1,-1,,
Data ,,,,,,,,,,,,,,,,,,,,,,,
Data local control,all notes off,omni off
Data omni on,mono on,poly on
'
'    programme principale
'
Do
  Mouse X%,Y%,K%
  If K%=1
    Rem ------------------CANAL MIDI-----------
    If X%<155 And X%>95
      Y%=Min(300,Y%)
```

(suite)

```
      Y%=Max(173,Y%)
      Yx%=Yi%
      @Dessin(100,150,5,(300-Y%) Div 8+1)
      Yi%=Yx%
    Endif
    Rem ----------------CONTROLEUR----------
    If X%<310 And X%>245
      Y%=Min(300,Y%)
      Y%=Max(173,Y%)
      Yx%=Yo%
      Con%=300-Y%
      @Dessin(250,300,25,300-Y%)
      Yo%=Yx%
      If Nom$(Con%)=""
        Nom$(Con%)="vide          "
      Endif
      Print At(32,22);Nom$(Con%);"          "
    Endif
    Rem ----------------VALEUR----------------
    If X%<460 And X%>405
      Y%=Min(300,Y%)
      Y%=Max(173,Y%)
      Yx%=Yu%
      @Dessin(400,450,45,300-Y%)
      Yu%=Yx%
    Endif
  Endif
  Exit If K%=3 Or Inkey$=" "
Loop
Rem ---------------------DESSIN-----------------
Procedure Dessin(Z%,E%,P%,Q%)
  If Y%<>Yx%
    Pbox Z%,Yx%-5,E%,Yx%+5
  Endif
  Line Z%,Y%,E%,Y%
  Yx%=Y%
  Print At(P%,20);Q%;"   "
  Out (3),175+Can%
  Out (3),Con%
  Out (3),Val%
Return
```

PROGRAMME CHANGE 192-207

N° de programme 0-127

Le code PROGRAMME CHANGE permet de commander les changements de sons sur les synthétiseurs. De manière générale il suffit d'envoyer le numéro du son désiré moins 1, le son 1 du synthétiseur correspondant souvent au numéro 0 pour le MIDI. Si votre synthétiseur est équipé d'une cartouche de sons supplémentaire, pour accéder à ces sons il faut additionner le nombre de sons internes au n° de son de la cartouche (– 1). Dans le cas où votre synthétiseur possède 32 sons internes et que vous désirez obtenir le son n° 15 de votre cartouche, il suffit d'envoyer au synthétiseur 32 + 15(– 1) comme valeur de programme.

AFTER TOUCH MONO 208-223

Valeur 0-127

L'AFTER TOUCH MONO est comparable à l'AFTER TOUCH POLY, la différence réside dans le fait qu'un seul taux d'effet est interprété par le synthétiseur. Toutes les notes sont affectées sans distinction du taux d'enfoncement. C'est la dernière valeur qui est prioritaire.

PITCH BEND 224-239

1re donnée 0-127
2e donnée 0-127

Le code PITCH BEND permet de désaccorder ponctuellement la totalité du clavier à l'aide d'une molette. Le taux de l'effet est fonction de la programmation du son. Le PITCH BEND utilise deux données qu'il est assez fastidieux de programmer soi-même.

II - Les codes message mode

Les modes MIDI sont très importants, ces codes servent à configurer le synthétiseur pour l'émission ou la réception des données MIDI. Ces codes peuvent être envoyés directement à l'aide des contrôleurs, mais en général la configuration s'effectue sur l'appareil.

MODE OMNI

OMNI OFF Contrôleur n° 124 valeur 0
OMNI ON Contrôleur n° 125 valeur 0

En mode OMNI ON, le synthétiseur reçoit les informations MIDI sans aucune sélection de canal. Le synthétiseur interprète tous les codes.

En mode OMNI OFF, le synthétiseur filtre les données MIDI et n'interprète que les informations correspondant à son canal de réception. C'est le mode le plus couramment utilisé.

MODE MONO Contrôleur n° 126 *valeur 0*

Le mode MONO permet, à l'émission ou à la réception, d'attribuer un canal MIDI différent à chacune des notes. Ce mode est très gourmand en canaux MIDI. Les guitares MIDI utilisent ce mode pour disposer d'un PITCH BEND différent sur chaque corde.

Si la valeur du contrôleur est différente de 0, cette valeur représente le nombre de canaux MIDI mis à la disposition du mode MONO (1 à 15). Le canal de base est celui indiqué par le synthétiseur.

Exemple :

valeur = 5 canal de base = 4

Les cinq premières notes sont jouées sur les canaux 4, 5, 6, 7, 8, au-delà de cette limite le cycle des canaux MIDI recommence sur le canal 4. Dans le cas de 8 notes jouées, les canaux émis sont 4, 5, 6, 7, 8, 4, 5, 6. Le synthétiseur organise lui-même la gestion des canaux MIDI.

MODE POLY *Contrôleur n° 127* *valeur 0*

Le mode POLY permet au synthétiseur d'émettre ou de recevoir sur un seul canal MIDI. C'est la position la plus utilisée.

LOCAL CONTROL *Contrôleur n° 122* *valeur 0 = ON*
< > 0 = OFF

Le LOCAL CONTROL sépare le clavier de l'unité de son. Cette fonction est très utile car elle permet de jouer sur le clavier maître sans entendre son unité de sons, tout en pilotant une autre source sonore. Dans le cas d'une configuration MIDI avec un séquenceur, cette fonction est indispensable.

ALL NOTE OFF Contrôleur n° 123 *valeur 0*

La fonction ALL NOTE OFF permet de désactiver toutes les notes d'un canal MIDI. Certains synthétiseurs ont besoin de ce code en fin de phrase.

III - Les codes message temps réel

Les codes message temps réel servent à gérer la synchronisation de différents appareils, par exemple un séquenceur et une boîte à rythmes. Ces codes peuvent être envoyés à n'importe quel moment.

TIMING CLOCK 248

Ce code est émis par un séquenceur ou une boîte à rythmes à raison de 24 codes par noire. C'est l'horloge MIDI. La fréquence d'émission de ce code détermine la vitesse d'exécution d'un morceau musical. L'horloge MIDI ne suffit pas au démarrage d'une séquence.

START 250

Ce code est émis au début de chaque morceau, il active le départ de la synchronisation (code 248). Le code START indique toujours la première mesure du morceau.

CONTINU 251

Ce code permet de repartir en milieu de séquence lorsque celle-ci est arrêtée à l'aide du code stop.

STOP 252

Arrêt d'une séquence et de l'horloge MIDI.

ACTIVE SENSING 254

Le code ACTIVE SENSING est optionnel. Certains synthétiseurs émettent ce code pour signaler leur branchement. Ce code est émis toutes les 300 ms (maximum). Aucune réelle application ne peut être envisagée par ce code, le synthétiseur émetteur ne pouvant être identifié par ce seul code.

SYSTEME RESET 255

Quand ce code est reçu par un synthétiseur, celui-ci se met dans l'état où il se trouve à la mise sous tension. C'est une initialisation du système.

IV - Les codes message système commun

SONG POSITION POINTER 242

1re donnée 0-127 LSB
2e donnée 0-127 MSB

Cette commande associée aux codes REAL TIME permet la synchronisation totale de deux séquenceurs. Les séquences peuvent être déclenchées à n'importe quel emplacement. Le SONG POSITION POINTER donne le nombre de code horloge divisé par 6, entre le début du morceau et le nouveau point de démarrage. La résolution maximale est la double croche.

Si l'on veut commencer le morceau à la cinquième mesure dans le cas de mesure à 4 temps, le SONG POSITION POINTER a la forme suivante :

(5 mesures × 4 temps × 24 horloge/temps) / 6
242 80 0

Après l'émission de ce code, un code CONTINU est émis.

TUNE REQUEST 246

Cette commande est une demande d'accord du synthétiseur à la hauteur prédéfinie en usine par le constructeur.

EOX 247

Ce code signale la fin d'un message de système exclusif.

V - Les codes message système exclusif

Le système exclusif est la partie non standard du MIDI. Seuls les deux premiers codes et la fin d'un message exclusif sont définis par la norme MIDI.

SYSTEME EXCLUSIF	240
ID (identification constructeur)	0-127
	xx
Données définies par le constructeur	0-127
	xx
EOX (fin de message exclusif)	247

Un message exclusif a toujours la même forme d'en-tête. Le code 240 indique qu'un message exclusif est envoyé, tous les appareils sont à l'écoute. L'ID permet d'effectuer un tri parmi les différents appareils. A chaque constructeur d'appareils MIDI correspond un code. Ce code permet d'identifier la marque de l'appareil. Par exemple YAMAHA possède le code 67, ROLAND le 65, KORG le 66. Ces codes sont attribués par le MMA (*MIDI Manufacture Association*) et le JMSC (*Japan MIDI Standards Committee*).

A partir de l'identification constructeur, chacune des marques a son propre protocole de communication à ses appareils. Aucun standard de système exclusif n'existe pour les synthétiseurs, seuls les échantillonneurs peuvent communiquer entre eux à l'aide du MIDI SAMPLE DUMP. Le code EOX est indispensable, il indique la fin du message exclusif.

Le système exclusif permet d'accéder à toutes les fonctions d'un synthétiseur ou autre appareil MIDI qui ne sont pas accessibles par les codes usuels. C'est par le système exclusif qu'il est possible de sauvegarder les sons d'un synthétiseur, d'accéder à la programmation des timbres, de modifier en temps réel des paramètres de réglage d'un son. Le système exclusif est le plus intéressant à travailler pour le programmeur.

Ce programme vous permet d'afficher tous les codes contenus dans le buffer MIDI. L'analyse des codes vous permettra de mieux comprendre comment est régi le système MIDI.

Pour plus de facilité d'interprétation des codes MIDI, la fonction PRINT CHR$(27) ; CHR$(13) ordonne un retour chariot dès qu'une commande est reçue.

```
'
'
'    -------- ESPION DU MIDI ---------
'
'
DO
  IF INP?(3)                  ! si entrée
    a%=INP(3)                 ! a%=code MIDI
    IF a%>127                 ! si commande ?
      PRINT CHR$(27);CHR$(13)!retour chariot
      PRINT a%';              ! écrire n° commande
    ELSE
      PRINT a%';              ! écrire donnée
    ENDIF
  ENDIF
  EXIT IF INKEY$=" "          ! sortie
LOOP
```

6/2.4

Programmation MIDI

6/2.4.1

Programme de détection du running statut

Le système de communication MIDI doit suivre des règles très précises sous peine de non fonctionnement. Nous savons que chaque information, pour être complète, doit être constituée d'une commande, code supérieur à 127, suivie de 0 à 2 données, code inférieur à 128, exception faite du message système exclusif qui peut posséder x données.

Les commandes sont répétées généralement à chaque opération, même si elles sont de nature identique.

Exemple :

OUT 3, 144,60,10,144,64,10

La commande **NOTE ON 144** est répétée avant chaque code hauteur de note et vélocité. La répétition de ce code entraîne une surcharge du MIDI, surtout si les codes sont de type molette de modulation ou **pitch bend**.

La vitesse de transmission du MIDI étant de 32,500 KB, la quantité d'information est limitée à 3 200 octets/seconde, soit une moyenne de 1 000 commandes/seconde. En cas d'utilisation excessive des molettes sur plusieurs canaux, le MIDI arrive vite à saturation, l'action sur une molette pouvant facilement compter 60 commandes (180 octets). Pour éviter l'engorgement du système, la norme MIDI comprend un protocole de communication permettant d'omettre le code commande si celui-ci est identique au précédent. Ce système se nomme le RUNNING STATUT.

Dans notre exemple, les codes précédemment envoyés deviennent sous running statut :

OUT 3, 144,60,10,64,10

La commande 144 n'est pas répétée.

L'économie faite est d'un octet, ce qui est peu, mais notre exemple est court.

Exemple plus long :

Sans running statut :

```
OUT 3, 144,60,10,144,64,1,144,67,10,144,72,10
```

Avec running statut :

```
OUT 3, 144,60,10,64,10,67,10,72,10
```

Pour 4 instructions, le running statut permet une économie de 3 octets.

Pour une action molette de 180 octets en mode normal, le running statut permet de compresser l'action sur 121 codes, soit ~ 1/3 de temps gagné pendant la transmission, ce qui est très appréciable. Tous les synthétiseurs sont capables d'interpréter le running statut, mais tous ne sont pas capables de l'émettre. La fiche d'implémentation MIDI, toujours fournie par le constructeur, vous donne le système de fonctionnement de votre matériel.

L'utilisation du running statut permet aussi un autre codage des instructions **NOTE OFF**. En fait la commande **NOTE OFF** est remplacée par **NOTE ON** avec une valeur nulle pour la vélocité, ce qui supprime encore un octet dans la transmission d'informations.

Exemple :

```
Out 3, 144,60,120,128,60,64
```

Ce groupe d'informations contient 2 commandes pour la même note.

```
Out 3, 144,60,120,60,0.
```

Une seule commande suffit pour activer et supprimer la note. Ce système implique que l'émetteur de codes MIDI ne dispose pas de vélocité de relâchement.

Tous les séquenceurs et autres logiciels tiennent compte de ce mode de communication, il va de soi que si vous créez un logiciel MIDI, il faut, sous peine de fonctionnement incertain, inclure la détection et la gestion du running statut.

PROGRAMME DE DÉTECTION DU RUNNING STATUT

Ce programme permet de tester s'il y a running statut ou non. Le test s'effectue sur le nombre de données envoyées (variable **nb__data&**), si cette variable est supérieure à 2 après un code de commande, alors il y a running statut.

La variable **nb__data&** est remise à 0 après chaque code de commande. Le running statut est aussi activé en cas d'envoi d'information de système exclusif.

La procédure **affiche** permet uniquement la visualisation des codes, elle peut être supprimée.

La simulation sur le clavier de l'Atari est possible en remplaçant les instructions **INP(3)** par **INP(2)**. Les codes commandes qui sont obligatoirement supérieurs à 127 sont simulés par les touches **UNDO, HELP**, etc.

```
--------------------run 1

.
.              Running statut
.
.               détection
.
.
DO
  IF INP?(3)
    a| =INP(3)
    IF a| <128
      INC nb_data&
      IF nb_data&>2
        PRINT AT(10,10);"running en cour"
      ENDIF
    ELSE
      nb_data&=0
    ENDIF
    affiche
    PRINT AT(10,10);"                    "
  ENDIF
LOOP
PROCEDURE affiche
  e%=MOD(e%,20)
  PRINT AT((e%*4)+1,20);a| '
  INC e%
RETURN
```

PROGRAMMES DE MIDI THRU

Ces deux programmes analysent les données MIDI reçues par l'Atari et renvoient les données après d'éventuelles transformations.

Ces programmes peuvent vous servir à analyser et filtrer les données reçues et ainsi facilement reconstituer des codes compatibles avec vos programmes.

Ce premier programme reconstitue les codes reçus sous running statut en codes complets de type commande + données. La variable **nb__data&** totalise le nombre de données inférieures à 128, si cette variable est supérieure à 1 (0 et 1 soit deux données), la commande contenue dans la variable **code__commande** est réactivée et **nb__data&** est remis à 0.

Entrée normale et running statut.

Supprime le running statut en sortie.

```
'----------------------------run 2

'
'            Running statut
'
'             MIDI THRU 1
'
'     supprime le running statut
'
DO
  IF INP?(3)
    a| =INP(3)
    IF a| <128
      IF nb_data&>1
        OUT (3),code_commande|
        nb_data&=0
      ENDIF
      INC nb_data&
    ELSE
      code_commande| =a|
      nb_data&=0
    ENDIF
    OUT 3,a|
  ENDIF
LOOP
```

Ce second programme reconstitue les codes sous le format running statut.

Entrée normale et running statut.

Ajoute le running statut si besoin est.

Ce programme teste si le code de la commande est identique au code précédemment reçu. S'il y a égalité, le code n'est pas transmis. Si le code commande est différent, il est alors mémorisé dans la variable **code_commande** pour attendre le prochain test.

```
---------------------- run 3


'
'            Running statut
'
'             MIDI THRU 2
'
'     ajoute le running statut
'
DO
  IF INP?(3)
    a| =INP(3)
    IF a| <128
      OUT 3,a|
    ELSE
      IF code_commande| <>a|
        OUT 3,a|
        code_commande| =a|
      ENDIF
    ENDIF
  ENDIF
LOOP
```

6/2.4.2

Arpégiateur harmonique sur le générateur sonore commandé depuis un clavier MIDI

GÉNÉRALITÉS

Le but du programme est de pouvoir déclencher sur le générateur sonore, à partir d'un clavier MIDI, une suite de notes, jouées soit simultanément, soit sous forme de deux arpèges ; le premier arpège se fait sur les notes d'un accord à trois sons ; le second arpège égrène itérativement une suite de notes déduites les unes des autres par ajout d'un intervalle constant.

Les fonctions à réaliser sont les suivantes :

— Lecture du clavier MIDI.

— Lecture du clavier de l'ordinateur.

— Lecture de la souris.

— Traitement des données MIDI pour envoi sur le générateur.

LE PROGRAMME EN DÉTAILS

La lecture du clavier MIDI doit réaliser deux fonctions essentielles :

a) Lire les informations provenant d'un clavier MIDI quel que soit le mode d'émission des données.

b) Réaliser un filtrage des données MIDI pour ne garder que les hauteurs de notes en vue de leur transformation sur la puce son.

La lecture du clavier de l'ordinateur permettra de faire varier les valeurs des deux intervalles successifs formant l'accord à trois sons.

La lecture de la souris permettra de choisir entre les trois modes de transformation des données MIDI : Accord, Arpège sur le premier intervalle et arpège sur l'accord à trois sons.

Le traitement de la hauteur des notes transmises par le MIDI se fait dans les trois procédures **joue**, **joue1**, **joue2**. La note envoyée par le MIDI fait office de fondamentale de l'accord égrené ou plaqué.

INITIALISATION

La partie initialisation du programme positionne les valeurs initiales des deux intervalles h1 | et h2 | sur **4** et 3, c'est-à-dire l'accord majeur, ces intervalles étant notés en demi-tons. Après avoir fait apparaître un message de réglage du volume sonore au maximum, le buffer MIDI de l'Atari est remis à zéro. Par défaut, la variable logique **acc!** est positionnée sur la valeur vraie, ce qui initialise le traitement des données sur la position accord.

LECTURE DE LA SOURIS

Une lecture simple de la souris permet, quel que soit le mode de résolution de votre moniteur, de positionner les variables logiques **acc!** et **arp!** dont les valeurs vraies correspondent à jouer l'accord et l'arpège de l'accord. Notez que pour **mousek = 3**, (les deux touches enfoncées), il est nécessaire d'attendre le relâchement complet des deux touches avant de passer à la suite du programme ; dans le cas contraire, vu l'impossibilité de relâcher les deux touches exactement en même temps, on repasserait par l'une des deux options touche gauche ou droite enfoncée.

LECTURE DU CLAVIER DE L'ORDINATEUR

La touche **espace** permet de faire apparaître une boîte d'alerte en vue de sélectionner deux nouvelles valeurs d'intervalles ; ces valeurs doivent être comprises entre zéro, (unisson) et 9, ce qui permet de réaliser tous les accords classiques à trois sons ainsi que bon nombre de configurations intéressantes. Notez que dans le cas où la somme des deux intervalles dépasse 12, vous pourrez obtenir un effet d'arpège brisé (montant et descendant), si vous avez sélectionné l'option arpège sur accord. La lecture des nouvelles valeurs d'intervalles se fait à partir du pavé numérique du clavier de l'ordinateur et le contrôle des valeurs entrées s'effectue au moyen de la procédure **test**. Cette procédure utilise la propriété de récursivité qu'offre le Basic GFA, c'est-à-dire de pouvoir s'appeler elle-même ; on restera dans cette procédure jusqu'à ce qu'une valeur de code ASCII comprise entre 48 et 57, (chiffres 0 et 9), soit disponible.

LECTURE ET FILTRAGE DES DONNÉES MIDI

Si un octet est disponible dans le buffer MIDI, il faut arriver à déterminer si cet octet est intéressant ou non pour le traitement futur. Dans tous les cas, il faut lire et tester si cet octet correspond à une commande (valeur > 127), ou à une donnée ; dans le cas d'une donnée, nous devons filtrer tout ce qui ne correspond pas à une donnée de note, que ce soit **Note on** ou un **Note off** ; c'est le rôle de l'indicateur logique **note!** ; lorsqu'il est positionné sur **TRUE**, c'est que le dernier code commande enregistré était soit un code de **Note on**, soit un code de **Note off** ; nous devons donc lire les deux octets disponibles dans le buffer qui correspondent à la hauteur et à la vélocité

de la note. Le positionnement de l'indicateur logique **note !** permet de lire les données de notes, que le synthétiseur utilise ou non le running statut en émission ; les hauteur et vélocité de notes sont stockées dans les variables octet **a|** et **b|**. La variable **b|** n'est pas utilisée dans ce programme (autrement que pour déceler un ***Note off***), mais il est facile de transformer la vélocité d'attaque d'une note en pas de volume sur le générateur sonore ; si on choisit une transformation linéaire, il faut transformer 128 valeurs, (de 0 à 127), en 16 valeurs potentielles, (de 0 à 15), soit huit fois moins ; l'opération (**b| DIV 8**) réalise cette opération ; si vous désirez avoir un effet sonore dépendant de la vélocité, remplacez par la formule précédente les valeurs 15 ou 13 mises par défaut dans les instructions **SOUND**.

TRAITEMENT DES DONNEES MIDI VERS LE GENERATEUR SONORE

La ventilation du traitement sonore se fait suivant la valeur des indicateurs logiques **acc!** et **arp!** vers les trois procédures **joue**, **joue1** et **joue2**.

Joue Cette procédure joue en temps réel sur le générateur, l'accord construit sur la fondamentale dont le code hauteur MIDI est contenu dans **a|** ; le premier test à effectuer est de savoir si la commande correspond bien à un **Note on** avant de la faire jouer ; dans le cas contraire, pour arrêter le son, on envoie un son de volume 0 sur les trois canaux de la puce. Pour envoyer l'accord sur le générateur, il faut d'abord convertir le code MIDI en octave et code note de 1 à 12 pour l'utiliser dans l'instruction **SOUND** ; la variable **c|** contient l'octave, puisque diviseur par 12 du code MIDI, et le reste de cette même division correspond au numéro de degré dans la gamme. Il faut corriger cette valeur de 1 car les codes MIDI de DO sont des multiples exacts de 12 et le code de DO est de 1 pour **SOUND**. La décrémentation de 1 en ce qui concerne l'octave permet de pouvoir mieux utiliser l'étendue sonore de générateur sonore par rapport à un clavier MIDI standard. Pensez à vérifier que votre clavier envoie bien un code de hauteur égale à 60 pour le DO3, car certains constructeurs ne respectent pas le standard MIDI. Ne soyez pas non plus trop surpris de trouver un désaccord entre votre synthétiseur et le générateur car les fréquences de circuit ne correspondent pas exactement à celles d'un clavier bien tempéré. Les hauteurs des deux notes suivantes de l'accord se déduisent en ajoutant respectivement ***h1|*** et ***h2|*** à **a|** ; il faut prendre soin de faire une correction d'octave dans le cas où cette valeur dépasse 12.

Joue1 Après avoir effectué les mêmes corrections que précédemment pour récupérer octave et degré de la note dans la gamme, cette procédure génère une suite de 15 notes de hauteurs distantes de la valeur ***h1|*** ; le temps d'attente entre deux notes est décroissant avec une valeur minimale de 3 pour des raisons musicales, mais il est intéressant de noter que le Basic GFA ignore purement et simplement l'instruction **PAUSE x** lorsque **x** est négatif ou nul.

Joue2 Cette procédure reprend le calcul initial de la procédure **joue** et propose l'arpège de l'accord ascendant de façon itérative sur quatre octaves. Les arpèges sont joués de plus en plus vite de la note grave à la note aiguë, à l'intérieur de chaque itération. Si vous désirez obtenir un effet continu d'accélération en gardant 15 comme valeur de départ de temps de pause, il faut utiliser les formules **15 – 4*i |**, **13 – 4*i |**, **11 – 4*i |** ; si vous désirez conserver un temps de pause minimal de 3, les fonctions **MAX(15 – 4*i |,3)**, **MAX(13 – 4*i |,3)** et **MAX(11 – 4*i |,3)** rempliront cet office.

```
' ------------INITIALISATION----------
' ------------------------------------
' ------------------------------------
text$="ENTREZ DEUX VALEURS¦ NUMERIQUES COMPRISES¦ ENTRE 0 ET 9"
text1$="  POSITIONNEZ LE VOMUME¦   DE VOTRE MONITEUR¦     A SON MAXIMUM "
ALERT 1,text1$,1,"OK",err
h1¦=4
h2¦=3
PRINT AT(10,10);"premier intervalle= ";h1¦
PRINT AT(10,11);"second intervalle= ";h2¦
SLPOKE XBIOS(14,2)+6,0
PRINT AT(5,5);"accord"
acc!=TRUE
DO
  ' ----------
  ' ----------
  ' LECTURE DE LA SOURIS
  ' ----------
  ' ----------
  IF MOUSEK=1
    PRINT AT(5,5);"accord"
    acc!=TRUE
  ENDIF
  IF MOUSEK=2
    PRINT AT(5,5);"arpege"
    arp!=FALSE
    acc!=FALSE
  ENDIF
  IF MOUSEK=3
    PRINT AT(5,5);"arpacc"
    arp!=TRUE
    acc!=FALSE
    REPEAT
    UNTIL MOUSEK=0
  ENDIF
  ' ----------------
  ' ----------------
  ' LECTURE DU CLAVIER ORDINATEUR
  ' ----------------
  ' ----------------
  IF INP?(2)
    car!=INP(2)
    IF car¦=32
      ALERT 1,text$,1,"OK",err
      @test
      h1¦=car¦-48
      PRINT AT(10,10);"premier intervalle= ";h1¦
      @test
      h2¦=car¦-48
      PRINT AT(10,11);"second intervalle= ";h2¦
    ENDIF
  ENDIF
  ' ----------------------
```

(suite)

```
'
' ----------------------
' ---LECTURE ET TRAITEMENT DES DONNEES MIDI----
' ----------------------------
IF INP?(3)
  '  PRINT "aze"
  a|=INP(3)
  IF a|<128
     SOUND 1,15,a|,c|
     PAUSE MAX(15-3*i|,3)
  NEXT i|
 ELSE
  SOUND 1,0
 ENDIF
RETURN
PROCEDURE joue2
 IF commande|>143 AND b|>0
  c|=a| DIV 12-1
  a|=1+a| MOD 12
  a1|=a|+h1|
  IF a1|>12
    SUB a1|,12
    c1|=c|+1
  ELSE
    c1|=c|
  ENDIF
  a2|=a1|+h2|
  IF a2|>12
    SUB a2|,12
    c2|=c1|+1
  ELSE
    c2|=c1|
  ENDIF
  FOR i|=0 TO 3
    SOUND 1,15,a|,c|+i|
    PAUSE 15-2*i|
    SOUND 1,15,a1|,c1|+i|
    PAUSE 13-2*i|
    SOUND 1,15,a2|,c2|+i|
    PAUSE 11-2*i|
  NEXT i|
  SOUND 1,0
 ELSE
  SOUND 1,0
 ENDIF
RETURN
PROCEDURE test
 REPEAT
 UNTIL INP?(2)
 car|=INP(2)
 IF car|<48 OR car|>57
   @test
 ENDIF
RETURN
```

(suite)

```
        IF note!=TRUE
          b!=INP(3)
          ' a! est la hauteur de la note
          '  b! est la velocit d'attaque ou de relachement
          IF acc!=TRUE
            @joue
          ELSE
            IF arp!=TRUE
              @joue2
            ELSE
              @joue1
            ENDIF
          ENDIF
        ENDIF
      ELSE
        IF a!>127 AND a!<160
          note!=TRUE
          commande!=a!
        ELSE
          note!=FALSE
        ENDIF
      ENDIF
    ENDIF
  LOOP
  PROCEDURE joue
    IF commande!>143 AND b!>0
      c!=a! DIV 12-1
      a!=1+a! MOD 12
      SOUND 1,13,a!,c!
      ADD a!,h1!
      IF a!>12
        SUB a!,12
        INC c!
      ENDIF
      SOUND 2,13,a!,c!
      ADD a!,h2!
      IF a!>12
        SUB a!,12
        INC c!
      ENDIF
      SOUND 3,13,a!,c!
      WAVE 7,1,0,20
    ELSE
      SOUND 1,0
      SOUND 2,0
      SOUND 3,0
    ENDIF
  RETURN
  PROCEDURE joue1
    IF commande!>143 AND b!>0
      c!=a! DIV 12-1
      a!=1+a! MOD 12
      SOUND 1,15,a!,c!
      PAUSE 15
      FOR i!=1 TO 15
        ADD a!,h1!
        IF a!>12
          SUB a!,12
          INC c!
        ENDIF
```

6/2.4.3
Boîte d'effets MIDI

GÉNÉRALITÉS

Le but de ce programme est de pouvoir enregistrer des informations de notes provenant d'un clavier MIDI quelconque, de les traiter et de les renvoyer sur un élément de synthèse, toujours par le moyen du système MIDI. La version qui est présentée ici est une première approche de boîte d'effets qui s'étendra par la suite au moyen de fonctions supplémentaires sous forme de procédures facilement intégrables dans le programme. Ce dernier reprend quelques notions et principes de traitement utilisés dans l'arpégiateur harmonique et la programmation d'un écho sur le générateur sonore.

La boîte d'effets doit réaliser les fonctions de répétition de notes en proposant une série de variations sur la vélocité, la hauteur et le délai des notes répétées ; le nombre de répétitions est ici limité à 12 pour qu'une version non compilée du programme marche sans problème, vu les énormes emplacements mémoire qu'il faudrait gérer, mais ce programme pourrait très bien générer un grand nombre de répétitions sur plusieurs minutes ; pensez toutefois que ce nombre sera limité par la polyphonie de votre synthétiseur, surtout si vous choisissez des sons avec du sustain et un long release.

Par mesure de sécurité, une procédure de Reset est prévue afin de ne pas devoir redémarrer à zéro si l'interface MIDI de votre synthétiseur rend l'âme !

Les fonctions essentielles à assurer sont les suivantes :

— Lecture du clavier MIDI

— Lecture de la souris

— Lecture du clavier de l'ordinateur

— Mémorisation des paramètres de transformation des notes

— Stockage des données MIDI filtrées dans un buffer

— Traitement des données stockées pour envoi sur le MIDI OUT.

LE PROGRAMME EN DÉTAIL

Lecture du clavier MIDI

Comme dans l'exemple de l'arpégiateur harmonique, on ne doit considérer que les informations de notes pour les traiter ultérieurement. Mais ici on doit lire aussi les notes off qui donneront la durée de la note à répéter, ainsi que les vélocités d'attaque pour pouvoir intervenir sur le timbre du son. Pensez à vérifier que votre synthétiseur est bien dynamique, et que le son choisi

est bien sensible à la vélocité sinon la partie du traitement concernant la vélocité ne produira aucun effet. Comme la majorité des synthétiseurs ne gèrent pas la vélocité de relâchement, les NOTE OFF sont systématiquement interprétées comme des NOTE ON avec vélocité à zéro afin d'utiliser le running statut.

Lecture de la souris

La lecture de la souris permettra de fixer le nombre de répétitions, de choisir les modes de transformation de la hauteur, de la vélocité et du délai au cours de ces répétitions. L'incrémentation des paramètres se fera par cliquage sur la touche gauche de la souris et la décrémentation par cliquage sur la touche droite. Ces traitements peuvent être accélérés en restant dans les boucles correspondantes jusqu'à ce que MOUSEK=0, ou en adjoignant un pas supplémentaire si MOUSEK=3, (en cliquant l'autre touche de la souris) ; n'oubliez pas dans ce cas de remplacer les aiguillages d'entrée IF kl = 1 par IF MOUSEK=1 ; autre précaution à prendre, ne pas jouer et s'assurer qu'aucune note ne doive sortir lorsque vous modifiez un paramètre sous peine de générer un décalage. Pour remédier à ce problème, il faudrait envisager une priorité de gestion du MIDI IN et du MIDI OUT sur le traitement de la souris.

Lecture du clavier de l'ordinateur

Les fonctions de sortie et de Reset sont commandées par le clavier de l'ordinateur afin de ne pas surcharger l'écran et laisser la place à une présentation ultérieure plus complète.

Mémorisation des paramètres de transformation

Dès qu'un des paramètres de reproduction est modifié, les nouvelles valeurs de correction des données de vélocité, hauteur et délai sont stockées dans les tableaux correspondants, ceci afin de n'avoir qu'un minimum de traitements à faire en temps réel au moment de l'envoi sur le MIDI OUT. Par défaut, les nouvelles valeurs de correction de données sont stockées pour le nombre maximal de répétitions, ceci même si vous désirez n'en effectuer que deux ou trois ; cela vous permet aussi de changer d'avis en cours de jeu, même si dans ce cas il faudrait envisager une gestion de niveaux de priorité comparable à ce qui se passe dans une gestion d'interruptions.

Stockage des données filtrées dans un buffer

Le buffer de stockage fonctionne de façon circulaire en utilisant les trois pointeurs p_in&( ), nb_out&( ), p_out&( ) qui sont respectivement le pointeur d'entrée, le pointeur de nombre de données à jouer et le pointeur d'adresse à jouer, ceci pour chacune des répétitions. Contrairement à la gestion du buffer MIDI ou clavier de l'Atari, le stockage ne s'effectue que si le buffer n'est pas plein, avec priorité aux données en place sur les données arrivant.

Traitement et envoi des données sur le MIDI OUT

La procédure joue n'est activée que si le nombre de répétitions et le délai sont différents de zéro. Chaque valeur d'indice des tableaux correspond à un numéro de répétition, ce qui permet de pouvoir jouer et corriger n'importe quel niveau de répétition sans faire référence à son ordre d'entrée ; les hauteurs de note sont corrigées par la variable **valmel&** et les vélocités par la variable **valvel&**, exception faite pour cette dernière des vélocités égales à zéro puisque le running statut NOTE ON (sur canal 1) est utilisé dans cet exemple.

INITIALISATION

Les tableaux **melo$( )**, **velo$( )** et **délai$( )** contiennent les options de transformation des paramètres de hauteur, vélocité et délai pour chaque répétition de note. Vous pouvez rajouter toutes les options qui vous paraissent intéressantes à envisager ; les tableaux **valdel&( )**, **valmel&( )** et **valvel&( )** vont contenir systématiquement les données à ajouter aux valeurs des paramètres de départ de la note jouée. Notez que le choix de tableaux de mots entiers se justifie sans problème pour le délai (valeurs pouvant être supérieures à 255), et pour des raisons de signe en ce qui concerne les valeurs de vélocité et de hauteur, puisque le décrément se fera par addition de valeur négative. Par ailleurs, il est intéressant de noter que les traitements de variables mot ou octet se font à la même vitesse en Basic GFA. Les variables **al** et **bl** contiendront respectivement la hauteur et la vélocité de la note jouée avant le stockage dans le buffer dont la taille est de 480 octets.

Les variables **notel** et **velol** contiendront les hauteurs et vélocités des notes à envoyer sur le MIDI OUT ; c'est pour des raisons de rapidité d'affectation d'adresse qu'elles sont initialisées ici et non au cours du traitement en temps réel.

Le running statut est initialisé en canal 1, (pensez à régler votre synthétiseur), puis le buffer MIDI de l'Atari est vidé et les tableaux **valdel&( )**, **valmel&( )** et **valvel&( )** sont remplis en correspondance avec les options et les valeurs choisies au départ pour le délai, la hauteur et la vélocité, dans les procédures **tdel**, **tmel** et **tvel**.

LECTURE DE LA SOURIS

La lecture de la souris sert à changer les options de nombres de répétitions, de choix et de valeurs de transformation de vélocité, hauteur et délai.

- **Procédure repet**

Cette procédure fait varier et affiche les valeurs du nombre de répétitions comprises entre 0 et 12. Cette procédure est la seule du groupe qui suit à ne pas appeler une des procédures de remplissage des tableaux **valdel&( )**, **valmel&( )** et **valvel&( )**.

Procédures de choix d'options

- **Procédure veloci**

Cette procédure fixe un des choix de transformation de la vélocité d'entrée :

— égale à l'entrée,

— constante et égale à la valeur contenue dans **valvell**,

— incrément constant, **(valvell** additionné d'un pas, **(pasvell)** à chaque répétition,

— décrément d'une valeur constante diminuée d'un pas à chaque répétition.

- **Procédure interv**

Cette procédure fixe les choix d'option de transformation de la hauteur de la note entrée :

— égale à l'entrée (écho traditionnel),

— addition d'une même constante pour chaque répétition,

— addition d'une constante et d'un pas d'incrément pour chaque répétition,

— soustraction d'une constante et d'un pas de décrément pour chaque répétition.

- **Procédure delay**

Cette procédure fixe les choix de différents types de délai d'échos :

— délai constant entre chaque note,

— délai croissant linéaire à partir de la première répétition,

— délai décroissant linéaire à partir de la première répétition.

Procédures de réglages de valeurs

- **Procédure valveloci**

Cette procédure permet de choisir la valeur de base de vélocité qui sera attribuée soit comme constante, soit comme premier incrément ou comme premier décrément de vélocité d'écho suivant l'option précédemment sélectionnée. Cette vélocité correspond à celle d'un NOTE ON et devra être donc comprise entre 1 et 127.

- **Procédure pasveloci**

Cette procédure règle de la même façon le pas d'incrément ou de décrément de vélocité.

- **Procédures valinterv et pasmelodie**

Ces procédures fonctionnent exactement comme les précédentes, mais se réfèrent à la hauteur des notes.

- **Procédure valdelay**

Un délai maximal est positionné par défaut sur la valeur 2000, soit dix secondes, mais libre à vous de n'imposer aucune borne supérieure.



• **Procédure pasdelay**

Aucune borne supérieure n'est imposée à la variable **pasdel&** qui servira de pas d'incrément ou de décrément au délai.

MÉMORISATION DES PARAMÈTRES

• **Procédures tdel, tdel0, tdel1, tdel2**

La procédure tdel est systématiquement appelée dès qu'on modifie une option de délai, (**delay**) la valeur de délai, (**valdelay**) ou du pas de délai (**pasdelay**).

La première de ces procédures est un aiguillage sur les traitements spécifiques de chaque option :

tdel0 Elle positionne la valeur du délai de façon linéaire dans le tableau **valdel&( )**.

tdel1 et tdel2 Elles positionnent le délai avec incrément, (décrément) linéaire dans le tableau **valdel&( )**. On peut aussi utiliser une formule directe de sommation de série pour remplir ce même tableau, (somme des ihl premiers entiers). Il faut simplement penser à limiter à zéro la valeur du délai dans le cas du décrément (Procédure **tdel2**).

• **Procédures tmel, tmel0, tmel1, tmel2, tmel3**

Ces procédures fonctionnent suivant le même principe que les précédentes : l'aiguillage tmel est appelé systématiquement dès qu'on touche à un des paramètres de hauteur : interv, valinterv, pasmelodie. Le tableau **valmel&( )** est ensuite actualisé.

tmel0 La correction de hauteur est nulle puisqu'elle correspond à l'option reproduire la hauteur jouée en entrée.

tmel1 La correction de hauteur est constante et égale à la valeur affichée pour toutes les répétitions.

tmel2 L'incrément est constant et vaut **pasmell** pour chaque répétition. La seule précaution à prendre est de majorer à 127 la valeur entrée dans le tableau, puisque l'arpège obtenu est ascendant.

tmel3 La valeur à entrer dans le tableau est exactement l'opposée de la précédente ; il faut cette fois minorer à **−127** cette valeur.

• **Procédures tvel, tvel0, tvel1, tvel2, tvel3**

Les procédures de modification d'options, (**veloci**), ou de valeurs de vélocité, (**valveloci** et **pasveloci**), appellent systématiquement la procédure **tvel** qui va aiguiller le traitement afin de remplir le tableau **valvel&( )** :

tvel0 Option vélocité d'entrée, donc modification nulle.

tvel1 Option vélocité constante, donc valeur constante dans tout le tableau.

tvel2 et tvel3 Options incrément et décrément linéaires ; chaque valeur est déduite de la précédente par incrément, (décrément), à partir de la première valeur **valvell** comptée positive dans le premier cas et négative dans le second. Comme précédemment, il faut majorer à 127 et minorer à – 127 dans chacun des cas.

LECTURE DU CLAVIER ORDINATEUR

La lecture du clavier ne sert qu'à appeler par la touche RETURN la fonction Reset qui réalise un ALL NOTE OFF, une initialisation des tableaux de pointeurs, du running statut NOTE ON et du buffer MIDI Atari ;

La procédure **fin** est appelée par le BREAK, libère la mémoire du buffer et génère un ALL NOTE OFF.

LECTURE DU CLAVIER MIDI

Le filtrage des données MIDI permet de n'enregistrer que les informations de note et de vélocité : reportez-vous au programme de l'arpégiateur harmonique pour la logique du filtrage et la génération du running statut. La variable **commandel** contient le dernier code commande reçu ce qui peut aussi vous permettre, le cas échéant, de positionner votre clavier en local off et envoyer l'instruction : **out 3,a1,b1** en fonction MIDI **thru** filtré, avant de stocker les octets dans le buffer. Si vous désirez changer de canal MIDI, ou si votre synthé prend les vélocités de relâchement, vous pouvez aussi utiliser l'instruction :

out 3,commandel,a1,b1

Dans le cas traité, il suffit de tester la valeur de **commandel** par rapport à 144 pour savoir si les deux octets se rapportent à un NOTE OFF et mettre l'octet de vélocité, (b1), à zéro. Le running statut permet ainsi de jouer avec plus de rapidité les données en sortie. Notez bien que le running statut n'est pas interrompu par des codes commande non suivi de valeurs de paramètres. Si votre synthétiseur génère le code 254 (Active sensing) ou bien l'horloge Midi (248). S'il possède un séquenceur, il faudra filtrer ces informations par un test supplémentaire du type :

```
IF a1 < 240
  note ! = FALSE
ENDIF
```

dans la dernière partie du traitement (ELSE) pour ne pas risquer d'ignorer des notes jouées trop proches les unes des autres.

STOCKAGE DES DONNÉES MIDI

Le principe du stockage est le même que celui utilisé dans la programmation d'un écho sur le générateur sonore. L'enregistrement des données ne s'effectue que si le buffer a de la place disponible, c'est-à-dire si le nombre

de données à jouer ne dépasse pas 80 dans le cas présent. Les données ne seront stockées que si la boîte d'écho avec délai est utilisée, donc si **repl** et **valdel&** ne sont pas nuls ; (notez que le test >0 est plus rapide que < >0). Si les conditions précédentes sont réalisées, les données sont enregistrées dans le buffer sur six octets, avec dans l'ordre hauteur (un octet), vélocité (un octet), et Timer sur quatre octets. Ensuite, les pointeurs relatifs aux entrées et nombre d'octets à jouer sont incrémentés de six pour les adresses et de un pour les événements.

ENVOI SUR LE MIDI OUT

La procédure joue tient ce rôle. Cette procédure est inclue dans la boucle générale et se trouve appelée de façon systématique. Pour éviter tout décalage dans le jeu, il faudrait théoriquement réaliser un va et vient permanent à même niveau de priorité entre la lecture du MIDI IN et l'envoi sur le MIDI OUT, et donner priorité à la première note sur toutes les répétitions suivantes dans le traitement, le stockage et l'envoi de l'information ; dans la majorité des cas, vous vous rendrez compte que même en version non compilée, les décalages sont minimes et souvent plus imputables à un blocage au niveau de la polyphonie. Pour un niveau donné de répétitions, on ne rentre dans le traitement de la procédure que si un événement est susceptible d'être joué ; le test d'entrée se fait par rapport au délai de niveau **ijl** et la correction de hauteur et de vélocité à ce même niveau. La correction de vélocité n'est effectuée que si la valeur d'origine n'est pas nulle. Si l'option choisie pour le traitement de la vélocité est la constante, cette valeur est mise dans **velol**. Après envoi sur le MIDI OUT, le pointeur d'adresse de sortie est incrémenté et celui du nombre d'événements à jouer est décrémenté de la même façon qu'au moment du stockage.

PROGRAMME BOITE D'EFFETS MIDI

```
DIM p_in&(12),nb_out&(12),p_out&(12)
DIM melo$(3),velo$(3),delai$(2)
DIM valdel&(12),valvel&(12),valmel&(12)
FOR il=0 TO 12
  p_in&(il)=0
  nb_out&(il)=0
  p_out&(il)=0
NEXT il
FOR il=0 TO 3
  READ melo$(il)
  velo$(il)=melo$(il)
NEXT il
DATA entr
suite:
DATA cste,incr,décr
RESTORE suite
FOR il=0 TO 2
  READ delai$(il)
NEXT il
tail&=480
x%=MALLOC(tail&)
al=0
bl=0
vell=3
repl=12
mell=3
dell=1
pasdel&=2
pasvell=5
pasmell=5
valdel&=10
valmell=3
valvell=8
notel=0
velol=0
colonnel=5
lignel=10
PRINT AT(colonnel,lignel);"REP","VEL","INT","DEL"
PRINT AT(colonnel,lignel+1),velo$(vell),melo$(mell),delai$(dell)
PRINT AT(colonnel,lignel+3),"PASVEL","PASMEL","PASDEL"
PRINT AT(colonnel+16,lignel+2);USING "###",valvell
PRINT AT(colonnel+32,lignel+2);USING "###",valmell
PRINT AT(colonnel+48,lignel+2);USING "###.##",valdel&/200
PRINT AT(colonnel+16,lignel+4);USING "###",pasvell
PRINT AT(colonnel+32,lignel+4);USING "###",pasmell
PRINT AT(colonnel+48,lignel+4);USING "###.##",pasdel&/200
PRINT AT(colonnel,lignel+1);USING "###",repl
posy&=16*(lignel)
posx&=8*(colonnel-1)
spxl=40
OUT 3,144
SLPOKE XBIOS(14,2)+6,0
commandel=144
ON BREAK GOSUB fin
@tmel
@tvel
```

(suite)

```
@tdel
DO
  ' lecture de la souris
  MOUSE x&,y&,k|
  IF k|
    IF y&<posy&+64 AND y&>posy&
      IF y&<posy&+16
        FOR i|=0 TO 3
          IF x&>128*i|+posx& AND x&<128*i|+posx&+spx|
            ON i|+1 GOSUB repet,veloci,interv,delay
          ENDIF
        NEXT i|
      ELSE
        IF y&<posy&+32
          FOR ii|=1 TO 3
            IF x&>128*ii|+posx& AND x&<128*ii|+posx&+spx|
              ON ii| GOSUB valveloci,valinterv,valdelay
            ENDIF
          NEXT ii|
        ELSE
          FOR ii|=1 TO 3
            IF x&>128*ii|+posx& AND x&<128*ii|+posx&+spx|
              ON ii| GOSUB pasveloci,pasmelodie,pasdelay
            ENDIF
          NEXT ii|
        ENDIF
      ENDIF
    ENDIF
  ENDIF
  ' lecture du clavier ATARI
  IF BIOS(1,2)
    az|=INP(2)
    IF az|=13
      @reset
    ENDIF
  ENDIF
  ' sortie MIDI OUT
  @joue
  ' LEcture clavier MIDI
  IF BIOS(1,3)
    a|=INP(3)
    IF a|<128
      IF note!=TRUE
        b|=INP(3)
        IF commande|<144
          b|=0
        ENDIF
        IF nb_out&(0)*6<tail& AND rep|>0 AND valdel&>0
          POKE x%+p_in&(0),a|
          POKE x%+p_in&(0)+1,b|
          LPOKE x%+p_in&(0)+2,TIMER
          FOR j|=0 TO rep|-1
            ADD p_in&(j|),6
            INC nb_out&(j|)
            p_in&(j|)=MOD(p_in&(j|),tail&)
          NEXT j|
        ENDIF
      ENDIF
```

(suite)

```
      ELSE
        IF a|>127 AND a|<160
          note!=TRUE
          commande!=a|
        ELSE
          note!=FALSE
        ENDIF
      ENDIF
    ENDIF
  LOOP
  PROCEDURE repet
    IF k|=1
      INC rep|
      rep|=MIN(rep|,12)
      PAUSE 5
    ELSE
      rep|=MAX(1,rep|)
      DEC rep|
      PAUSE 5
    ENDIF
    PRINT AT(colonne|,ligne|+1);USING "###",rep|
  RETURN
  PROCEDURE veloci
    IF k|=1
      INC vel|
      vel|=MIN(vel|,3)
      PAUSE 5
    ELSE
      vel|=MAX(1,vel|)
      DEC vel|
      PAUSE 5
    ENDIF
    PRINT AT(colonne|+16*i|,ligne|+1);velo$(vel|)
    @tvel
  RETURN
  PROCEDURE interv
    IF k|=1
      INC mel|
      mel|=MIN(mel|,3)
      PAUSE 5
    ELSE
      mel|=MAX(1,mel|)
      DEC mel|
      PAUSE 5
    ENDIF
    PRINT AT(colonne|+16*i|,ligne|+1);melo$(mel|)
    @tmel
  RETURN
  PROCEDURE delay
    IF k|=1
      INC del|
      del|=MIN(del|,2)
      PAUSE 5
    ELSE
      del|=MAX(1,del|)
      DEC del|
      PAUSE 5
    ENDIF
```

(suite)

```
    PRINT AT(colonnel+16*il,lignel+1);delaiS(del)
    @tdel
  RETURN
  PROCEDURE valveloci
    IF kl=1
      INC valvell
      valvell=MIN(valvell,127)
      PAUSE 5
    ELSE
      valvell=MAX(1,valvell)
      DEC valvell
      PAUSE 5
    ENDIF
    PRINT AT(colonnel+16*il,lignel+2);USING "###",valvell
    @tvel
  RETURN
  PROCEDURE valinterv
    IF kl=1
      INC valmell
      valmell=MIN(valmell,127)
      PAUSE 5
    ELSE
      valmell=MAX(1,valmell)
      DEC valmell
      PAUSE 5
    ENDIF
    PRINT AT(colonnel+16*il,lignel+2);USING "###",valmell
    @tmel
  RETURN
  PROCEDURE valdelay
    IF kl=1
      INC valdel&
      valdel&=MIN(valdel&,2000)
      PAUSE 3
    ELSE
      valdel&=MAX(1,valdel&)
      DEC valdel&
      PAUSE 3
    ENDIF
    PRINT AT(colonnel+16*il,lignel+2);USING "###.##",valdel&/200
    @tdel
  RETURN
  PROCEDURE pasveloci
    IF kl=1
      pasvell=MIN(pasvell,126)
      INC pasvell
      PAUSE 5
    ELSE
      pasvell=MAX(1,pasvell)
      DEC pasvell
      PAUSE 5
    ENDIF
    PRINT AT(colonnel+16*il,lignel+4);USING "###",pasvell
    @tvel
  RETURN
  PROCEDURE pasmelodie
    IF kl=1
      pasmell=MIN(pasmell,126)
```

(suite)

```
      INC pasmel!
      PAUSE 5
    ELSE
      pasmel!=MAX(1,pasmel!)
      DEC pasmel!
      PAUSE 5
    ENDIF
    PRINT AT(colonne!+16*ii!,ligne!+4);USING "###",pasmel!
    @tme!
  RETURN
  PROCEDURE pasdelay
    IF k!=1
      INC pasdel&
      PAUSE 5
    ELSE
      pasdel&=MAX(1,pasdel&)
      DEC pasdel&
      PAUSE 5
    ENDIF
    PRINT AT(colonne!+16*ii!,ligne!+4);USING "####.##",pasdel&/200
    @tde!
  RETURN
  PROCEDURE tde!
    ON de!+1 GOSUB tde!0,tde!1,tde!2
  RETURN
  PROCEDURE tme!
    ON me!+1 GOSUB tme!0,tme!1,tme!2,tme!3
  RETURN
  PROCEDURE tve!
    ON ve!+1 GOSUB tve!0,tve!1,tve!2,tve!3
  RETURN
  PROCEDURE tde!0
    FOR ih!=0 TO 12
      valde!&(ih!)=(ih!+1)*valde!&
    NEXT ih!
  RETURN
  PROCEDURE tde!1
    valde!&(0)=valde!&
    FOR ih!=1 TO 12
      valde!&(ih!)=valde!&(ih!-1)+valde!&+ih!*pasde!&
    NEXT ih!
  RETURN
  PROCEDURE tde!2
    valde!&(0)=valde!&
    FOR ih!=1 TO 12
      valde!&(ih!)=MAX(valde!&(ih!-1)+valde!&-ih!*pasde!&,0)
    NEXT ih!
  RETURN
  PROCEDURE tme!0
    FOR ih!=0 TO 12
      valme!&(ih!)=0
    NEXT ih!
  RETURN
  PROCEDURE tme!1
    FOR ih!=0 TO 12
      valme!&(ih!)=valme!!
    NEXT ih!
  RETURN
```

(suite)

```
PROCEDURE tmel2
  FOR ihl=0 TO 12
    valmel&(ihl)=MIN(valmell+ihl*pasmell,127)
  NEXT ihl
RETURN
PROCEDURE tmel3
  FOR ihl=0 TO 12
    valmel&(ihl)=MAX(-valmell-ihl*pasmell,-127)
  NEXT ihl
RETURN
PROCEDURE tvel0
  FOR ihl=0 TO 12
    valvel&(ihl)=0
  NEXT ihl
RETURN
PROCEDURE tvel1
  FOR ihl=0 TO 12
    valvel&(ihl)=valvell
  NEXT ihl
RETURN
PROCEDURE tvel2
  valvel&(0)=valvell
  FOR ihl=1 TO 12
    valvel&(ihl)=MIN(valvel&(ihl-1)+pasvell,127)
  NEXT ihl
RETURN
PROCEDURE tvel3
  valvel&(0)=-valvell
  FOR ihl=1 TO 12
    valvel&(ihl)=MAX(valvel&(ihl-1)-pasvell,-127)
  NEXT ihl
RETURN
PROCEDURE joue
  IF repl>0 AND valdel&>0
    FOR ijl=0 TO repl-1
      IF nb_out&(ijl)>0
        IF LPEEK(x%+p_out&(ijl)+2)+valdel&(ijl)<=TIMER
          notel=PEEK(x%+p_out&(ijl))
          notel=MIN(127,notel&+valmel&(ijl))
          notel=MAX(0,notel&)
          velol=PEEK(x%+p_out&(ijl)+1)
          IF velol>0
            velol=MIN(127,velol+valvel&(ijl))
            velol=MAX(1,velol)
            IF vell=1
              velol=valvell
            ENDIF
          ENDIF
          OUT 3,notel,velol
          ADD p_out&(ijl),6
          p_out&(ijl)=MOD(p_out&(ijl),tail&)
          DEC nb_out&(ijl)
        ENDIF
      ENDIF
    NEXT ijl
  ENDIF
RETURN
PROCEDURE reset
```

(suite)

```
    OUT 3,176,123,0,144
    FOR il=0 TO 12
      p_in&(il)=0
      nb_out&(il)=0
      p_out&(il)=0
    NEXT il
    SLPOKE XBIOS(14,2)+6,0
RETURN
PROCEDURE fin
    ~MFREE(x%)
    OUT 3,176,123,0
    END
RETURN
```

6/2.4.4
Analyse d'accords

GÉNÉRALITÉS

Ce programme se propose de faire une analyse en temps réel d'une position d'accord jouée à partir d'un clavier MIDI. En même temps que cet essai d'analyse, le programme affiche le nom des notes jouées, les intervalles qui séparent les notes entre elles ainsi que l'intervalle séparant chaque note de la note la plus basse, ou fondamentale de l'accord. La notation des accords ainsi que leur chiffrage sont faits en dehors de tout contexte harmonique ou d'arrangement, c'est-à-dire que ce programme, dans cette première version ne pourra analyser que les renversements traditionnels d'accords. Il restera à lui adjoindre ultérieurement les procédures traitant des omissions et des drops de degrés. Ces opérations ne changent ni la fonction harmonique ni l'appellation d'un accord mais sont des façons de le faire sonner dans un contexte non isolé.

Les fonctions essentielles que doit réaliser ce programme sont les suivantes :

— lecture en permanence de l'interface MIDI afin de considérer toutes les informations de NOTE ON ou NOTE OFF qui parviennent ;

— analyse et rangement de la hauteur de la nouvelle note par rapport aux notes précédentes si NOTE ON ;

— analyse et suppression de la nouvelle hauteur si NOTE OFF ;

— rangement des notes par hauteurs croissantes ;

— suppression des notes doubles qui n'ont aucune fonction harmonique ;

— réajustement de l'octave de la fondamentale sur les hauteurs suivantes pour tout ramener dans deux octaves ;

— affichage du renversement d'accord trouvé ou d'un autre message dans le cas contraire.

LE PROGRAMME EN DÉTAIL

Initialisation

Les tableaux **intervalle$( )** et **note$( )** sont utilisés pour réaliser l'affichage des noms d'intervalles et de notes jouées. Si le tableau **intervalle$( )** limite ses appellations sur la double octave, c'est que tout accord peut se comprimer sur cette étendue si l'on résume chaque note à sa fonction harmonique.

Les tableaux **accord$( )** vont être utilisés pour afficher la dénomination des accords alors que les tableaux **renv$( )** vont contenir les renversements des

accords sous forme de successions d'intervalles codés en chaînes de caractères ; le chiffre final sert à préciser le nombre de notes constituant l'accord. Les tableaux d'accords à trois et quatre sons sont rentrés sous forme directe par lecture de DATA alors qu'un artifice de calcul est utilisé pour ne rentrer en DATA que la position fondamentale de l'accord à cinq sons et en déduire les quatre renversements.

Cette astuce sera aussi utilisée dans le complément ultérieur qui permettra au programme d'analyser les accords à six et sept sons et leurs renversements. Mais revoyons rapidement la définition d'un accord et de ses renversements. Un accord à trois sons est caractérisé par la suite de deux intervalles qui séparent ces trois notes. Effectuer un renversement, c'est prendre la note la plus grave de l'accord ou fondamentale et la transposer à l'octave supérieur afin qu'elle devienne la note la plus aiguë de l'accord. Dans le cas d'un accord à trois sons (accord de quinte) ou d'un accord à quatre sons (accord de septième), l'intervalle manquant est la distance séparant la note aiguë de la fondamentale à l'octave supérieure.

Premier exemple :

Prenons l'accord majeur parfait constitué des intervalles de tierce majeure et de tierce mineure. L'intervalle manquant est la quarte, puisque tierce majeure + tierce mineure + quarte = octave

(4+3+5)demi-tons=12demi-tons

Les renversements successifs s'écrivent donc, à partir de l'état fondamental et en notant l'intervalle manquant entre parenthèses :

4 3 (5),3 5 (4),5 4 (3)

Deuxième exemple :

Prenons l'accord de septième majeure quinte diminuée constitué des intervalles de tierce majeure, seconde majeure et quarte ; on vérifie facilement que la distance d'un demi-ton entre la septième majeure et la tonique octavée est bien égale à 12(octave)–4(tierce majeure)–2(seconde majeure)–5(quarte) l'accord à l'état fondamental s'écrit : 4 2 5 (1) et les renversements se déduisent par permutations circulaires des intervalles en notant l'intervalle manquant entre parenthèses, soit :

2 5 1 (4), 5 1 4 (2), 1 4 2 (5)

Notez à ce propos que l'accord de septième de dominante quinte bémol qui s'écrit : 4 2 4 (2) n'aura qu'un renversement hors d'un contexte harmonique et que l'on confondra les appellations de Majeur sixième et de mineur septième, puisque :

Majeur sixième : 4 3 2 (3)

Mineur septième : 3 4 3 (2)

on voit que dès la première permutation, l'accord de septième mineure (1er renversement) est égal au majeur sixième à l'état fondamental.

C'est cette logique qui est appliquée dans l'accord à cinq sons. La seule différence est que la transposition doit se faire deux octaves plus haut pour la



fondamentale puisque la neuvième se trouve au-dessus de la première octave de la tonique.

Exemple :

Accord de neuvième majeure :

état fondamental:tierce maj,tierce min,tierce min,tierce maj

soit la succession chiffrée :4 3 3 4

L'intervalle manquant est le complément à deux octaves, soit 24 demi-tons, de la somme des intervalles précédents :24 – (4 + 3 + 3 + 4) = 10 l'accord complet se note maintenant :

4 3 3 4 (10)

Les renversements se déduisent par permutations circulaires :

3 3 4 10 (4)

3 4 10 4 (3)

4 10 4 3 (3)

10 4 3 3 (4)

C'est cette logique qui est programmée pour déduire tous les renversements des accords à cinq sons à partir des états fondamentaux entrés dans **renv5f$( )**. Le tableau **tot( )** sert à stocker l'intervalle manquant, sans oublier de réajuster le code ascii des intervalles par la valeur 48, code ascii de zéro. Ensuite le caractère correspondant est rajouté à droite de la chaîne fondamentale.

LECTURE DE L'INTERFACE MIDI

La boucle de lecture de l'interface MIDI reprend le filtrage classique des données de notes en considérant les cas de COMMANDE NOTE ON et NOTE OFF. En cas de NOTE ON, on effectue simplement une opération de concaténation dans la chaîne harmo$ et dans le cas de NOTE OFF on recherche la note à enlever dans la chaîne. Le drapeau logique harmo! sert à sortir de la boucle itérative de test dès qu'on a trouvé la note à supprimer. Après avoir effacé l'écran, le traitement s'oriente dans les deux cas vers les procédures de rangement des notes vers les hauteurs croissantes (procédure **range**) et d'affichage des notes et intervalles de l'accord (procédure **affiche**). Dans le cas où l'accord est constitué de plus de trois notes on va essayer de supprimer les notes inutiles à l'analyse (procédure **supprime**), puis explorer les tableaux de renversements et afficher le résultat (procédure **renversement**). Comme dans ce programme vous avez toutes les chances de plaquer des accords pour les faire analyser, le test de filtrage de codes commande supérieurs 5240 est obligatoire pour ne commettre aucune omission de note.

Procédure range

Cette procédure utilise un algorithme de tri par degrés conjoints qui permet d'être assez performant pour que l'analyse en temps réel permette un affichage rapide. Pour l'instant, avec la version actuelle du Basic GFA, il est préférable de passer par une variable chaîne de caractères de transfert quand on veut échanger deux éléments de la chaîne **harmo$**.

Procédure affiche

La procédure affiche toujours au moins la fondamentale et les intervalles dès que l'accord a au moins deux notes.

Procédure supprime

Pour l'instant, cette procédure supprime les redoublements d'octaves de notes et ne garde que la note inférieure, ce qui explique le test modulo 12. De plus, lorsque la fondamentale est à un intervalle de plus d'une octave de la note immédiatement supérieure, elle est transposée à l'octave supérieure pour retrouver une fonction harmonique normale ; ceci est effectué dans la chaîne **harmo1$** qui ne sert qu'à l'analyse.

Procédure renversement

Cette procédure n'est effective que lorsque le contenu de la chaîne **harmo1$** peut correspondre à un accord éventuellement répertorié, donc de trois sons au moins. Le corps de la procédure est un aiguillage de recherche dans les tableaux de renversements d'accords, suivant le nombre de notes de l'accord. Notez que la chaîne **renv$**, qui code les intervalles de l'accord, est remise systématiquement à zéro dès que l'on passe dans l'affichage. De la même façon que précédemment, le drapeau logique test ! permet de sortir de la boucle de recherche dès que le test d'égalité est positif. Si cette variable booléenne est toujours sur la valeur FALSE en fin d'aiguillage, c'est que l'accord joué n'est pas répertorié sur cette position avec le nombre voulu de notes.

```
DIM intervalle$(24),note$(127),accord3$(127),renv3$(127)
DIM accord4$(127),renv4$(127),accord5$(127)
DIM renv5f$(17),renv5$(127),tot(17)
DEFTEXT 1,1,0,6
FOR i=0 TO 24
  READ intervalle$(i)
NEXT i
DATA unisson,sec_min,sec_MAJ,tce_min,tce_MAJ,quarte
DATA quinte_dim,quinte,quinte_aug,sixte,sept_min
DATA sept_MAJ,octave,neuv_min,neuv_MAJ,neuv_aug,dixième
DATA onzième,onzième_aug,douzième,treize_min,treize_MAJ
DATA quatorze_min,quatorze_MAJ,dbleoctave
FOR i=0 TO 127
  READ note$(i)
NEXT i
DATA .....
DATA C0,C#0,D0,D#0,E0,F0,F#0,G0,G#0,A0,A#0,B0
DATA C1,C#1,D1,D#1,E1,F1,F#1,G1,G#1,A1,A#1,B1
DATA C2,C#2,D2,D#2,E2,F2,F#2,G2,G#2,A2,A#2,B2
DATA C3,C#3,D3,D#3,E3,F3,F#3,G3,G#3,A3,A#3,B3
DATA C4,C#4,D4,D#4,E4,F4,F#4,G4,G#4,A4,A#4,B4
DATA C5,C#5,D5,D#5,E5,F5,F#5,G5,G#5,A5,A#5,B5
DATA .....
RESTORE acc3
FOR i=0 TO 127
  READ accord3$(i)
  EXIT IF accord3$(i)="FACC3"
NEXT i
acc3:
DATA MAJ,MAJ1erRenv,MAJ2èmeRenv,MIN,Min1erRenv,MIN2èmeRenv
DATA M5+,SUS4,SUS41erRenv,SUS42èmeRenv,DIM,DIM1erRenv,DIM2èmeRenv
DATA M5-,M5-1erRenv,M5-2èmeRenv,FACC3
RESTORE renv3
FOR i=0 TO 127
  READ renv3$(i)
  EXIT IF renv3$(i)="FRENV3"
NEXT i
renv3:
DATA 43,35,54,34,45,53,44,52,25,55
DATA 33,36,63,42,26,64,FRENV3
RESTORE acc4
FOR i=0 TO 127
  READ accord4$(i)
  EXIT IF accord4$(i)="FACC4"
NEXT i
acc4:
DATA 6,6_1erR,6_2èmeR,6_3èmeR
DATA 7M,7M_1erR,7M_2èmeR,7M_3èmeR
DATA 7,7_1erR,7_2èmeR,7_3èmeR
DATA m6,m6_1erR,m6_2èmeR,m6_3èmeR
DATA m7M,m7M_1erR,m7M_2èmeR,m7M_3èmeR
DATA 7dim
DATA 7sus4,7sus4_1erR,7sus4_2èmeR,7sus4_3èmeR
DATA 5#7,5#7_1erR,5#7_2èmeR,5#7_3èmeR
DATA 5#7M,5#7M_1erR,5#7M_2èmeR,5#7M_3èmeR
DATA 5b7,5b7_1erR
DATA 5b7M,5b7M_1erR,5b7M_2èmeR,5b7M_3èmeR,FACC4
RESTORE renv4
FOR i=0 TO 127
  READ renv4$(i)
  EXIT IF renv4$(i)="FRENV4"
NEXT i
renv4:
DATA 432,323,234,343
DATA 434,341,414,143
DATA 433,332,324,243
DATA 342,423,233,334
DATA 344,441,413,134
```

(suite)

```
DATA 333
DATA 523,232,325,252
DATA 442,422,224,244
DATA 443,431,314,144
DATA 424,242
DATA 425,251,514,142,FRENV4
RESTORE acc5
FOR il=0 TO 127
  READ accord5$(il)
  EXIT IF accord5$(il)="FACC5"
NEXT il
acc5:
DATA 69,69_1erR,69_2èmeR,69_3èmeR,69_4èmeR
DATA 7M9,7M9_1erR,7M9_2èmeR,7M9_3èmeR,7M9_4èmeR
DATA 79,79_1erR,79_2èmeR,79_3èmeR,79_4èmeR
DATA 79b,79b_1erR,79b_2èmeR,79b_3èmeR,79b_4èmeR
DATA 79#,79#_1erR,79#_2èmeR,79#_3èmeR,79#_4èmeR
DATA m69,m69_1erR,m69_2èmeR,m69_3èmeR,m69_4èmeR
DATA m79,m79_1erR,m79_2èmeR,m79_3èmeR,m79_4èmeR
DATA m7M9,m7M9_1erR,m7M9_2èmeR,m7M9_3èmeR,m7M9_4èmeR
DATA 79sus4,79sus4_1erR,79sus4_2èmeR,79sus4_3èmeR,79sus4_4èmeR
DATA 5#79,5#79_1erR,5#79_2èmeR,5#79_3èmeR,5#79_4èmeR
DATA 5#79b,5#79b_1erR,5#79b_2èmeR,5#79b_3èmeR,5#79b_4èmeR
DATA 5#79#,5#79#_1erR,5#79#_2èmeR,5#79#_3èmeR,5#79#_4èmeR
DATA 5#7M9,5#7M9_1erR,5#7M9_2èmeR,5#7M9_3èmeR,5#7M9_4èmeR
DATA 5b79,5b79_1erR,5b79_2èmeR,5b79_3èmeR,5b79_4èmeR
DATA 5b79b,5b79b_1erR,5b79b_2èmeR,5b79b_3èmeR,5b79b_4èmeR
DATA 5b79#,5b79#_1erR,5b79#_2èmeR,5b79#_3èmeR,5b79#_4èmeR
DATA 5b7M9,5b7M9_1erR,5b7M9_2èmeR,5b7M9_3èmeR,5b7M9_4èmeR,FACC5
RESTORE renv5
FOR il=0 TO 16
  READ renv5f$(il)
  EXIT IF renv5f$(il)="FRENV5"
NEXT il
renv5:
DATA 4325,4343,4334,4333,4335,3425,3434
DATA 3443,5234,4424,4423,4425,4433,4244
DATA 4243,4245,4253,FRENV5
FOR il=0 TO 16
  FOR jl=1 TO 4
    ADD tot(il),ASC(MID$(renv5f$(il),jl))-48
  NEXT jl
  renv5f$(il)=renv5f$(il)+CHR$(72-tot(il))
NEXT il
FOR il=0 TO 16
  FOR jl=0 TO 4
    renv5$(5*il+jl)=LEFT$(renv5f$(il),4)
    renv5f$(il)=RIGHT$(renv5f$(il),4)+LEFT$(renv5f$(il),1)
  NEXT jl
NEXT il
SLPOKE XBIOS(14,2)+6,0
DO
  IF BIOS(1,3)
    al=INP(3)
    IF al<128
      IF note!=TRUE
        bl=INP(3)
        ' si note on, on rajoute dans la chaine
        IF commande!>143 AND bl>0
          harmo$=harmo$+CHR$(al)
        ELSE
          ' si note off on enlève de la chaine
          harmo!=FALSE
          FOR il=1 TO LEN(harmo$)
            IF MID$(harmo$,il,1)=CHR$(al)
              harmo$=LEFT$(harmo$,il-1)+RIGHT$(harmo$,LEN(harmo$)-il)
              harmo!=TRUE
            ENDIF
```

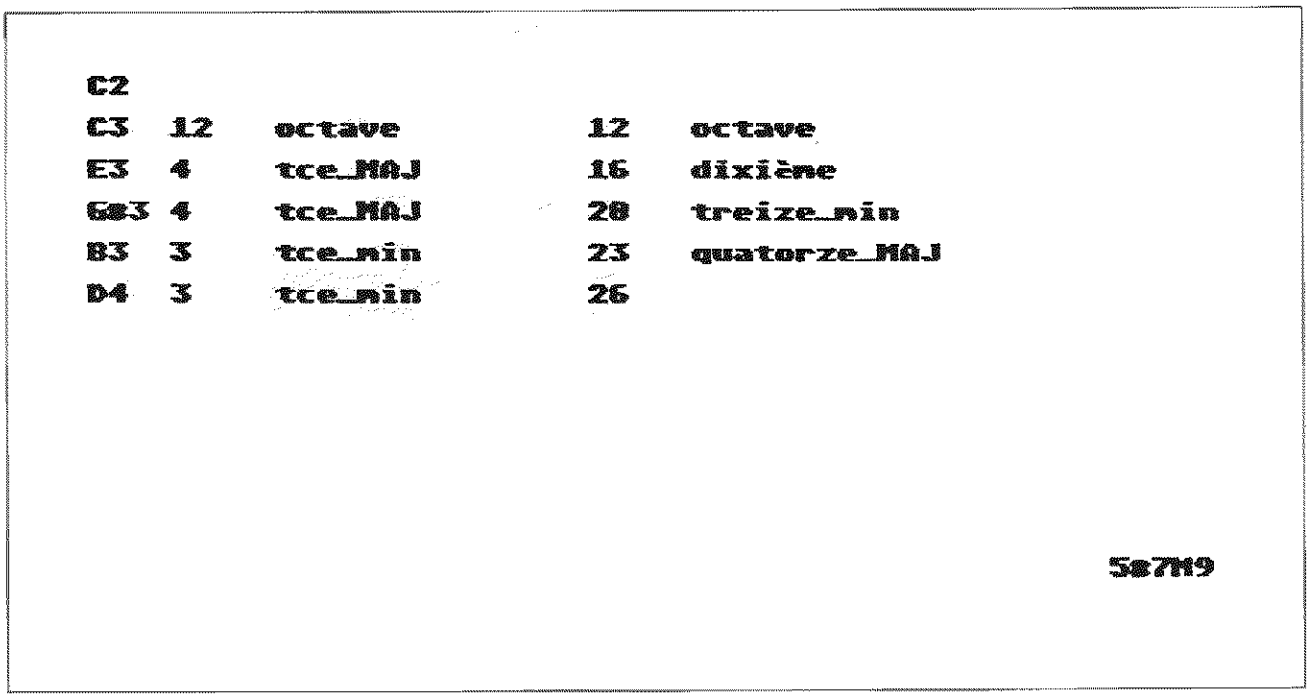


(suite)

```
                    EXIT IF harmo!=TRUE
                  NEXT i!
                  CLS
                ENDIF
                @range
                @affiche
                IF LEN(harmo$)>2
                  @supprime
                  @renversement
                ENDIF
              ENDIF
            ELSE
              IF a!>127 AND a!<160
                note!=TRUE
                commande!=a!
              ELSE
                IF a!<240
                  note!=FALSE
                ENDIF
              ENDIF
            ENDIF
          ENDIF
        LOOP
        PROCEDURE range
          REPEAT
            k!=TRUE
            FOR i!=1 TO LEN(harmo$)
              FOR j!=1 TO i!-1
                IF MID$(harmo$,j!,1)>MID$(harmo$,i!,1)
                  k!=FALSE
                  stock$=MID$(harmo$,j!,1)
                  MID$(harmo$,j!,1)=MID$(harmo$,i!,1)
                  MID$(harmo$,i!,1)=stock$
                ENDIF
              NEXT j!
            NEXT i!
          UNTIL k!=TRUE
        RETURN
        PROCEDURE affiche
          renv$=""
          FOR i!=1 TO LEN(harmo$)
            TEXT 8,16*i!,"      "
            TEXT 8,16*i!,note$(ASC(MID$(harmo$,i!)))
          NEXT i!
          IF LEN(harmo$)>1
            FOR i!=2 TO LEN(harmo$)
              j!=ASC(MID$(harmo$,i!))-ASC(MID$(harmo$,i!-1))
              k!=ASC(MID$(harmo$,i!))-ASC(MID$(harmo$,1))
              TEXT 40,16*i!,j!
              TEXT 80,16*i!,"                    "
              IF j!<25
                TEXT 80,16*i!,intervalle$(j!)
              ENDIF
              TEXT 200,16*i!,k!
              TEXT 240,16*i!,"                    "
              IF k!<25
                TEXT 240,16*i!,intervalle$(k!)
              ENDIF
            NEXT i!
          ENDIF
        RETURN
        PROCEDURE supprime
          harmo1$=harmo$
          REPEAT
            sup!=FALSE
            FOR i!=1 TO LEN(harmo1$)-1
              FOR j!=i!+1 TO LEN(harmo1$)
                IF (ASC(MID$(harmo1$,j!,1))-ASC(MID$(harmo1$,i!,1))) MOD 12=0
```

(suite)

```
                harmo2$=LEFT$(harmo1$,j-1)+RIGHT$(harmo1$,LEN(harmo1$)-j)
                sup!=TRUE
                harmo1$=harmo2$
              ENDIF
              EXIT IF sup!=TRUE
            NEXT j
            EXIT IF sup!=TRUE
          NEXT i
        UNTIL sup!=FALSE
        WHILE ASC(MID$(harmo1$,2,1))-ASC(harmo1$)>12
          MID$(harmo1$,1,1)=CHR$(ASC(harmo1$)+12)
        WEND
      RETURN
      PROCEDURE renversement
        IF LEN(harmo1$)>2
          FOR i=2 TO LEN(harmo1$)
            j=ASC(MID$(harmo1$,i))-ASC(MID$(harmo1$,i-1))
            renv$=renv$+CHR$(j+48)
          NEXT i
          test!=FALSE
          SELECT LEN(harmo1$)
          CASE 3
            FOR i=0 TO 127
              IF renv$=renv3$(i)
                TEXT 400,200,"                "
                TEXT 400,200,accord3$(i)
                test!=TRUE
              ENDIF
              EXIT IF test!=TRUE
            NEXT i
          CASE 4
            FOR i=0 TO 127
              IF renv$=renv4$(i)
                TEXT 400,200,"                "
                TEXT 400,200,accord4$(i)
                test!=TRUE
              ENDIF
              EXIT IF test!=TRUE
            NEXT i
          CASE 5
            FOR i=0 TO 127
              IF renv$=renv5$(i)
                TEXT 400,200,"                "
                TEXT 400,200,accord5$(i)
                test!=TRUE
              ENDIF
              EXIT IF test!=TRUE
            NEXT i
          CASE 6 TO 7
            TEXT 400,200,"                "
            TEXT 400,200,"pas dispo"
            test!=TRUE
          DEFAULT
          ENDSELECT
          IF test!=FALSE
            TEXT 400,200,"                "
            TEXT 400,200," INEXISTANT"
          ENDIF
        ENDIF
      RETURN
```



COMPLÉMENT D'ANALYSE D'ACCORDS

Ce programme introduit les accords de six et sept sons, leurs renversements et propose des méthodes plus évoluées de remplissage des tableaux ainsi qu'un schéma plus fin d'analyse harmonique.

La première insertion dans le programme précédent se fait au niveau des déclarations de tableaux.

Le tableau **suff$( )** contient les suffixes représentant les numéros des renversements à ajouter à la dénomination fondamentale de l'accord. Le remplissage des noms d'accords se fait par pas de 5 pour les accords de onzième qui ont cinq renversements et par pas de six pour les accords de treizième qui possèdent six renversements.

La génération du codage des renversements d'accords à six et sept sons se fait comme dans le cas des accords à cinq sons car on peut réduire l'ambitus mélodique à deux octaves lorsque l'on considère la fonction harmonique d'une note.

Vous pouvez remarquer que les accords de six sons et sept sons sont moins nombreux que ceux de dimension moindre ; des problèmes peuvent d'ailleurs se présenter en cours d'analyse puisque une suite d'intervalles peut représenter plusieurs accords ou renversements d'accords de toniques différentes ; vous avez le choix de présenter toutes les solutions, d'essayer de faire une analyse harmonique par rapport aux accords précédents et suivants ou ne présenter qu'une solution suivant un critère que vous aurez choisi.

C'est cette solution qui est proposée ici avec deux choix différents pour les accords de six et sept notes ; on sort des boucles de test dès qu'on a trouvé la même suite d'intervalles dans le tableau **renv$( )** correspondant à la longueur de l'accord réduit par l'instruction **EXIT IF TEST!=TRUE.**

Dans le cas des accords à six sons c'est le premier trouvé dans l'ordre de rangement du tableau **renv6$( )** qui prévaut.

Dans le cas des accords à sept sons, on teste en priorité l'état fondamental de l'accord, puis les renversements par ordre croissant ; cela veut dire que certains renversements introduits n'apparaîtront jamais ; mais libre à vous de changer en cours d'analyse de critère de choix ; il vous suffit d'écrire les différents algorithmes de recherche et de les choisir par l'affichage du critère de choix au moyen d'une case indicée, à incrémenter et décrémenter par la souris. Pour fixer les idées, prenons l'accord de DO treizième.

Le deuxième renversement de cet accord est constitué des notes :
sol sib ré fa la do mi.

Cette suite de notes correspond aussi à l'accord de sol m 13 à l'état fondamental.

Le critère de choix proposé ici analysera la suite d'intervalles 343434 comme état fondamental du m13 et non comme second renversement de l'accord de 13.

Procédure oct

Cette procédure se charge de réduire toutes les notes non doublées dans un intervalle de deux octaves afin de faire une analyse harmonique plus complète que précédemment. Chaque note de la chaîne de traitement harmonique **harmo1$** est testée par rapport à l'élément gauche de la chaîne, (note la plus grave et donc fondamentale de l'accord). Toute différence de code ASCII supérieure à 24 est décrémentée de 12 et l'élément est replacé dans la chaîne. La réduction à une octave supplémentaire dépend du nombre de notes de l'accord. Les deux procédures **plus4sons** et **moins5sons** se chargent de considérer les différentes possibilités par rapport aux tensions possibles des notes de l'accord.

Pour un accord de quatre sons au plus, toutes les notes doivent être ramenées dans un intervalle d'une octave puisque nous n'envisageons a priori que des accords complets, donc de septième au maximum.

La considération des accords incomplets se fera plus tard en introduisant un traitement supplémentaire avant de déclarer un accord non répertorié.

Pour un accord de cinq sons différents et plus, on ramènera dans la première octave les notes de base de l'accord, tierce dans le cas où une quinte juste existe déjà, ainsi que quinte ou septième à l'octave, pour essayer de reconstituer l'accord de base de septième sur lequel on peut bâtir une structure comprenant des tensions.



```
' SUITE DES DÉCLARATIONS DE TABLEAUX ET RENVERSEMENTS

DIM renv6f$(17),renv6$(127),accord6$(127)
DIM renv7f$(17),renv7$(127),accord7$(127)
DIM suff$(6)
FOR il=1 TO 6
  READ suff$(il)
NEXT il
DATA _1erR,_2èmeR,_3èmeR,_4èmeR,_5èmeR,_6èmeR

' ------------------------
' SUITE DES REMPLISSAGES DES TABLEAUX DE RENVERSEMENTS
' ------------------------
ARRAYFILL tot(),0
RESTORE acc6
FOR il=0 TO 13
  READ accord6$(6*il)
  EXIT IF accord6$(il)="FACC6"
  FOR jl=1 TO 5
    accord6$(6*il+jl)=accord6$(6*il)+suff$(jl)
  NEXT jl
NEXT il
acc6:
DATA 6911#,7M911#,7911,7911#,79b11
DATA 79b11#,79#11#,m6911,m7911,m7M911
DATA 5#7911,5#79b11,5#79#11,5b7911
DATA FACC6
RESTORE renv6
FOR il=0 TO 13
  READ renv6f$(il)
  EXIT IF renv6f$(il)="FRENV6"
NEXT il
renv6:
DATA 43254,43434,43343,43344,43334
DATA 43335,43353,34253,34343,34433
DATA 44243,44234,44252,42443
DATA FRENV6
FOR il=0 TO 13
  FOR jl=1 TO 5
    ADD tot(il),ASC(MID$(renv6f$(il),jl))-48
  NEXT jl
  renv6f$(il)=renv6f$(il)+CHR$(72-tot(il))
NEXT il
FOR il=0 TO 13
  FOR jl=0 TO 5
    renv6$(6*il+jl)=LEFT$(renv6f$(il),5)
    renv6f$(il)=RIGHT$(renv6f$(il),5)+LEFT$(renv6f$(il),1)
  NEXT jl
NEXT il
```



(suite)

```
FOR i|=0 TO 83
  IF renv$=renv6$(i|)
    TEXT 400,200,"                    "
    TEXT 400,200,accord6$(i|)
    test!=TRUE
  ENDIF
  EXIT IF test!=TRUE
NEXT i|
' ici on teste d'abord si on trouve l'accord
' en fondamental puis 1er Renv, 2ème Renv.....
CASE 7
FOR i|=0 TO 6
  FOR j|=0 TO 7
    IF renv$=renv7$(i|+7*j|)
      TEXT 400,200,"                    "
      TEXT 400,200,accord7$(i|+7*j|)
      test!=TRUE
    ENDIF
    EXIT IF test!=TRUE
  NEXT j|
  EXIT IF test!=TRUE
NEXT i|
ENDSELECT
IF test!=FALSE
TEXT 400,200,"                    "
TEXT 400,200," NON REPERT "
ENDIF
' ------------------------------------------------
ENDIF
' ------------------------------------------------
PROCEDURE oct
REPEAT
logic!=TRUE
FOR aa|=2 TO LEN(harmo1$)
  john|=ASC(MID$(harmo1$,aa|,1))-ASC(harmo1$)
  IF john|>24
    MID$(harmo1$,aa|,1)=CHR$(ASC(MID$(harmo1$,aa|,1))-12)
  ENDIF
  IF LEN(harmo1$)>4 AND john|>12
    @plus4sons
  ENDIF
  IF LEN(harmo1$)<5 AND john|>12
    @moins5sons
  ENDIF
NEXT aa|
@range1
UNTIL logic!=TRUE
RETURN
```

(suite)

```
PROCEDURE range1
REPEAT
k!=TRUE
FOR i|=1 TO LEN(harmo1$)
  FOR j|=1 TO i|-1
    IF MID$(harmo1$,j|,1)>MID$(harmo1$,i|,1)
      k!=FALSE
      stock$=MID$(harmo1$,j|,1)
      MID$(harmo1$,j|,1)=MID$(harmo1$,i|,1)
      MID$(harmo1$,i|,1)=stock$
    ENDIF
  NEXT j|
NEXT i|
UNTIL k!=TRUE
RETURN
PROCEDURE plus4sons
quinte!=FALSE
FOR i|=2 TO LEN(harmo1$)
j|=ASC(MID$(harmo1$,i|))-ASC(harmo1$)
IF j|=7
  quinte!=TRUE
ENDIF
NEXT i|
IF john|>12 AND ((john|=16 AND quinte!=TRUE) OR john|=19 OR john|>21)
MID$(harmo1$,aa|,1)=CHR$(ASC(MID$(harmo1$,aa|,1))-12)
logic!=FALSE
ENDIF
RETURN
PROCEDURE moins5sons
MID$(harmo1$,aa|,1)=CHR$(ASC(MID$(harmo1$,aa|,1))-12)
logic!=FALSE
RETURN
```

6/2.4.4.1
Analyse des positions incomplètes d'accords

GÉNÉRALITÉS

Le programme précédent vous a permis d'analyser en temps réel tous les accords répertoriés de trois à sept sons ainsi que leurs renversements. Les modifications apportées par les trois programmes qui suivent vont permettre de considérer aussi les positions incomplètes des accords. Avant d'entrer plus avant dans le vif du sujet, quelques remarques fonctionnelles s'imposent.

1) La définition d'un accord se fait à partir de trois notes différentes plaquées, à un modulo 12 près. Un accord incomplet de trois sons ne sera pas analysé puisque le fait de supprimer une note le ramène à un intervalle (deux sons), traitement qui fera l'objet d'un autre programme.

2) Les accords réduits et incomplets susceptibles d'être analysés auront leur nombre de notes qui varie de trois (incomplet de 4 notes) à six (incomplet de 7 notes).

3) Les accords incomplets peuvent être définis de plusieurs façons différentes : soit à partir d'une position réelle dont on supprime une note en conservant la position, soit à partir d'une position réelle en réduisant les nouveaux intervalles selon la méthode d'analyse vue précédemment. C'est un panachage de ces deux façons de procéder qui est présenté dans les programmes qui suivent.

4) Les accords incomplets analysés ne seront incomplets que d'un degré, par exemple on ne cherchera pas à retrouver un accord de cinq sons comme un renversement d'accord de 13e sans quinte ni tierce.

5) Lorsqu'on enlève à un accord sa tonique, généralement il se retrouve être l'accord dont la nouvelle tonique est la tierce de l'ancien accord, puisque les accords se construisent généralement par superposition de tierces :

L'accord de Mi mineur septième se compose des notes **Mi Sol Si Ré**.

Il est composé de deux accords de quinte qui se superposent :

— L'accord de Mi mineur : **Mi Sol Si**.

— L'accord de Sol Majeur : **Sol Si Ré**.

En enlevant la tonique de Mi mineur septième, on conserve la structure supérieure de l'accord, à savoir l'accord de Sol majeur qu'il n'est évidemment pas utile de considérer comme position incomplète d'un autre accord !

Les exceptions à cette règle apparaissent lorsque les accords ont des tierces diminuées successives dans leur structure, ou des intervalles de seconde, ce qui est le cas des accords altérés.

6) Il est inutile d'enlever à un accord sa tension supérieure puisque un accord de onzième redevient un accord de neuvième etc.

7) Les seules positions essentielles qui restent par défaut correspondent donc aux accords sans tierce, sans quinte et sans septième si on cherche à analyser toutes positions incomplètes de façon exhaustive.

Dans cette première approche, les datas seront rentrés de façon manuelle. Ultérieurement, une logique de création automatique de toutes les positions avec constitution d'un dictionnaire proposant l'élimination ou la sélection prioritaire des positions doubles.

LE PROGRAMME EN DÉTAIL

Le complément se présente sous la forme de trois morceaux de programmes :

— Les datas des positions incomplètes avec le remplissage des tableaux d'appellations et de codages des intervalles.

incacc4$** est le tableau des appellations de positions incomplètes d'accords de quatre sons, donc correspondant à des structures de trois sons (ou deux intervalles) codés dans le tableau **renvinc4$ etc.

```
DIM renvinc4$(100),incacc4$(100)
DIM renvinc5$(100),incacc5$(100)
DIM renvinc6$(100),incacc6$(100)
DIM renvinc7$(100),incacc7$(100)
RESTORE inc4
FOR il=0 TO 100
  READ incacc4$(il)
  EXIT IF incacc4$(il)="FINCACC4"
NEXT il
inc4:
DATA 6_sstce,6_1erR_sstce,6_3èmeR_sstce
DATA 6_sston,6_2èmeR_sston,6_3èmeR_sston
DATA 7M_sstce,7M_1erR_sstce,7M_3èmeR_sstce
DATA 7M_ss5te,7M_1erR_ss5te,7M_2èmeR_ss5te
DATA 7_sstce,7_1erR_sstce,7_3èmeR_sstce
DATA 7_ss5te,7_1erR_ss5te,7_2èmeR_ss5te
DATA m7M_ss5te,m7M_1erR_ss5te,m7M_2èmeR_ss5te
DATA 5#7_sstce,5#7_1erR_sstce,5#7_3èmeR_sstce
DATA 5#7M_sstce,5#7M_1erR_sstce,5#7M_3èmeR_sstce
DATA 5b7_sstce,5b7_1erR_sstce,5b7_3èmeR_sstce
DATA 5b7_sston,5b7_2èmeR_sston,5b7_3èmeR_sston
DATA 5b7M_sstce,5b7M_1erR_sstce,5b7M_3èmeR_sstce
DATA 5b7M_sston,5b7M_2èmeR_sston,5b7M_3èmeR_sston
DATA FINCACC4
RESTORE rinc4
FOR il=0 TO 100
  READ renvinc4$(il)
  EXIT IF renvinc4$(il)="FRENVINC4"
```

(suite)

```
NEXT il
rinc4:
DATA 72,23,37
DATA 32,27,73
DATA 74,41,17
DATA 47,71,14
DATA 73,32,27
DATA 46,62,24
DATA 38,81,13
DATA 82,22,28
DATA 83,31,18
DATA 64,42,26
DATA 24,46,62
DATA 65,51,16
DATA 25,55,52,FRENVINC4
RESTORE inc5
FOR il=0 TO 100
  READ incacc5$(il)
  EXIT IF incacc5$(il)="FINCACC5"
NEXT il
inc5:
DATA 69_sston,69_2èmeR_sston,69_3èmeR_sston,69_4èmeR_sston
DATA 69_sstce,69_1erR_sstce,69_3èmeR_sstce,69_4èmeR_sstce
DATA 69_ss5te,69_1erR_ss5te,69_2èmeR_ss5te,69_4èmeR_ss5te
DATA 9_ss7ème,9_1erR_ss7ème,9_2èmeR_ss7ème,9_3èmeR_ss7ème
DATA 7M9_sstce,7M9_1erR_sstce,7M9_3èmeR_sstce,7M9_4èmeR_sston
DATA 7M9_ss5te,7M9_1erR_ss5te,7M9_2èmeR_ss5te,7M9_4èmeR_ss5te
DATA 79_sstce,79_1erR_sstce,79_3èmeR_sstce,79_4èmeR_sstce
DATA 79_ss5te,79_1erR_ss5te,79_2èmeR_ss5te,79_4èmeR_ss5te
DATA 79b_sstce,79b_1erR_sstce,79b_3èmeR_sstce,79b_4èmeR_sstce
DATA 79b_ss5te,79b_1erR_ss5te,79b_2èmeR_ss5te,79b_4èmeR_ss5te
DATA 79b_ss7ème,79b_1erR_ss7ème,79b_2èmeR_ss7ème,79b_3èmeR_ss7ème
DATA 79#_ss5te,79#_1erR_ss5te,79#_2èmeR_ss5te,79#_4èmeR_ss5te
DATA 79#_ss7ème,79#_1erR_ss7ème,79#_2èmeR_ss7ème,79#_3èmeR_ss7ème
DATA m69_ss5te,m69_1erR_ss5te,m69_2èmeR_ss5te,m69_4èmeR_ss5te
DATA m79_ss5te,m79_1erR_ss5te,m79_2èmeR_ss5te,m79_4èmeR_ss5te
DATA m79_ss7ème,m79_1erR_ss7ème,m79_2èmeR_ss7ème,m79_3èmeR_ss7ème
DATA m7M9_ss5te,m7M9_1erR_ss5te,m7M9_2èmeR_ss5te,m7M9_4èmeR_ss5te
DATA 7s4_ss5te,7s4_1erR_ss5te,7s4_2èmeR_ss5te,7s4_4èmeR_ss5te
DATA 5#79_sstce,5#79_1erR_sstce,5#79_3èmeR_sstce,5#79_4èmeR_sstce
DATA 5#79b_sstce,5#79b_1erR_sstce,5#79b_3èmeR_sstce,5#79b_4èmeR_sstce
DATA 5#79#_sstce,5#79#_1erR_sstce,5#79#_3èmeR_sstce,5#79#_4èmeR_sstce
DATA 5#7M9_sstce,5#7M9_1erR_sstce,5#7M9_3èmeR_sstce,5#7M9_4èmeR_sstce
DATA 5b79_sstce,5b79_1erR_sstce,5b79_3èmeR_sstce,5b79_4èmeR_sstce
DATA 5b79b_sstce,5b79b_1erR_sstce,5b79b_3èmeR_sstce,5b79b_4èmeR_sstce
DATA 5b79#_sstce,5b79#_1erR_sstce,5b79#_3èmeR_sstce,5b79#_4èmeR_sstce
DATA 5b7M9_sstce,5b7M9_1erR_sstce,5b7M9_3èmeR_sstce,5b7M9_4èmeR_sstce
```

(suite)

```
DATA FINCACC5
RESTORE rinc5
FOR il=0 TO 100
  READ renvinc5$(il)
  EXIT IF renvinc5$(il)="FRENVINC5"
NEXT il
rinc5:
DATA 325,252,523,232
DATA 725,232,325,523
DATA 455,532,528,289
DATA 437,352,522,223
DATA 743,412,125,541
DATA 473,712,122,271
DATA 734,322,225,532
DATA 464,622,222,262
DATA 733,321,216,163
DATA 463,621,213,136
DATA 436,351,513,133
DATA 465,623,231,316
DATA 438,353,531,313
DATA 365,632,321,216
DATA 374,722,221,217
DATA 347,452,521,214
DATA 384,812,121,218
DATA 734,322,225,253
DATA 824,222,226,262
DATA 823,221,217,172
DATA 825,223,235,352
DATA 833,312,126,263
DATA 644,422,224,244
DATA 643,421,215,154
DATA 645,423,233,344
DATA 653,512,124,245
DATA FRENVINC5
RESTORE inc6
FOR il=0 TO 127
  READ incacc6$(il)
  EXIT IF incacc6$(il)="FINCACC6"
NEXT il
inc6:
DATA FINCACC6
RESTORE rinc6
FOR il=0 TO 100
  READ renvinc6$(il)
  EXIT IF renvinc6$(il)="FRENVINC6"
NEXT il
rinc6:
DATA FRENVINC6
```



(suite)

```
RESTORE inc7
FOR il=0 TO 100
  READ incacc7$(il)
  EXIT IF incacc7$(il)="FINCACC7"
NEXT il
inc7:
DATA FINCACC7
RESTORE rinc7
FOR il=0 TO 100
  READ renvinc7$(il)
  EXIT IF renvinc7$(il)="FRENVINC7"
NEXT il
rinc7:
DATA FRENVINC7
```

— Une modification de la logique de traitement dans la boucle DO LOOP principale. Il faut d'abord analyser une position susceptible d'être incomplète avant de réduire toutes ses notes sur une octave par la procédure OCT : Un accord de neuvième (composé de cinq sons) serait alors considéré comme un accord potentiel de septième et ses notes regroupées sur une octave ce qui empêcherait de retrouver cette position incomplète renversée.

Par exemple, l'accord de Do 9e en position fondamentale se compose des notes : **Do Mi Sol Sib Ré**.

L'accord sans quinte en position fondamentale se compose des notes : **Do Mi Sib Ré**.

Si on ne modifiait rien au programme d'analyse d'accords, cet accord se réduirait à : **Do Ré Mi Sib**.

Cette réduction permet de retrouver l'accord au sens large mais ignore le renversement en ramenant les notes en degrés conjoints.

```
IF LEN(harmo$)>2
  @supprime
  @renversement
  @incomplet
  IF test!=FALSE
    ' il faut enlever OCT dans la procédure supprime
    @oct
    @renversement
    @incomplet
    IF test!=FALSE
      TEXT 400,200,"                    "
      TEXT 400,200," NON REPERT "
    ENDIF
  ENDIF
ENDIF
```

— Une suite d'instructions analysant les positions incomplètes et leurs renversements avec affichage de la solution trouvée. Pour l'instant, le programme analyse les positions incomplètes d'accords de moins de six sons (jusqu'aux accords de neuvième).

```
PROCEDURE incomplet
  SELECT LEN(harmo1$)
  CASE 3
    FOR il=0 TO 100
      IF renv$=renvinc4$(il)
        TEXT  400,200,"                "
        TEXT 400,200,incacc4$(il)
        test!=TRUE
      ENDIF
      EXIT IF test!=TRUE
    NEXT il
  CASE 4
    FOR il=0 TO 100
      IF renv$=renvinc5$(il)
        TEXT  400,200,"                "
        TEXT 400,200,incacc5$(il)
        test!=TRUE
      ENDIF
      EXIT IF test!=TRUE
    NEXT il
  CASE 5
    FOR il=0 TO 100
      IF renv$=renvinc6$(il)
        TEXT  400,200,"                "
        TEXT 400,200,incacc6$(il)
        test!=TRUE
      ENDIF
      EXIT IF test!=TRUE
    NEXT il
    '-------------------------------------------
  CASE 6
    FOR il=0 TO 100
      IF renv$=renvinc7$(il)
        TEXT  400,200,"                "
        TEXT 400,200,incacc7$(il)
        test!=TRUE
      ENDIF
      EXIT IF test!=TRUE
    NEXT il
  ENDSELECT
RETURN
```

6/2.4.5

Dictée musicale

GÉNÉRALITÉS

Le programme qui suit se propose de vous faire réaliser, via le système MIDI, un schéma de dictée musicale. Pour cette première approche, vous pourrez choisir le tempo de votre dictée, la tonalité, la gamme sur laquelle vous voudrez tester votre oreille et le nombre de notes de votre dictée ; certains paramètres ont été ici volontairement maximisés, mais vous pouvez les transformer à votre gré. Dans un premier temps, cette dictée ne sera que mélodique et monophonique, mais des procédures supplémentaires permettront facilement de lui adjoindre plus tard les lectures rythmiques et polyphoniques, (bien que moins facilement pour cette dernière). Dans la version actuelle du programme, les dictées sont proposées sur une étendue d'une octave à partir de la note tonique choisie, mais vous pouvez augmenter ou diminuer facilement cet ambitus mélodique maximal.

Une analyse des notes que vous jouerez permettra de comparer votre solution à la dictée initiale, de présenter et rejouer les notes originales dans le cas où vous auriez commis quelques erreurs. Tous les réglages des paramètres ainsi que le choix des fonctions de dictée de jeu et d'enregistrement se font à l'aide de la souris, exception faite de la fin d'enregistrement qui s'effectue par la frappe d'une touche quelconque du clavier.

LE PROGRAMME EN DÉTAIL

Initialisation

La première fonction que réalise le programme est de rentrer les schémas de gammes à utiliser dans les dictées qui seront proposées ; les tableaux correspondants portent les mêmes appellations que les gammes. Les intervalles successifs de ces gammes ne dépassant pas l'octave, (12 demi-tons), ce sont bien évidemment des tableaux d'octets. Notez qu'il serait facile de construire à partir de ces gammes tous les modes bâtis sur chacun des degrés, en faisant une permutation circulaire à chaque étape.

Exemple :

Si l'on choisit le mode pentatonique Majeur qui est proposé ici, dont la suite d'intervalles comptés à partir de la note tonale sont :

0,2,4,7,9

on obtient tous les modes dérivés et leurs intervalles en faisant un décalage à gauche, récupérant l'octave, et en soustrayant le premier intervalle aux suivants :

(2 – 2),(4 – 2),(7 – 2),(9 – 2),(12 – 2)

soit :

0,2,5,7,10

puis en opérant de même :

0,3,5,8,10

0,2,5,7,9

0,3,5,7,10

Vérifiez que lorsque vous ferez la dernière permutation, vous retombez bien sur la gamme de départ :

0,2,4,7,9

Dans le programme qui suit, chaque note de la gamme a la même probabilité d'être choisie, ce qui rend cette exploration des modes inutile dans ce contexte ; mais si vous décidez de compliquer un peu le modèle proposé, il est facile de faire une distribution de probabilités différentes afin que les notes tonales importantes sortent plus souvent et privilégient ainsi le côté modal de votre dictée. Pour chaque exemple de gamme à sept degrés, vous aurez ainsi 6 choix supplémentaires de gammes.

Les tableaux **notetot$** et **note$** sont utilisés respectivement pour réaliser l'affichage de la solution et de la tonalité choisie.

Le tableau **option$** contient à la fois les noms des paramètres de la dictée, tonalité, gamme, tempo, nombre de notes et les trois fonctions essentielles qui sont :

— réaliser une dictée dans la tonalité, sur la gamme et avec le nombre de notes choisies ;

— jouer cette dictée au tempo choisi ;

— enregistrer les hauteurs de notes envoyées par un clavier MIDI, les comparer à l'original et afficher le résultat de la comparaison.

Par défaut, les paramètres de départ sont C3, code 60 de hauteur de note MIDI pour la tonalité, nombre de notes égal à huit, tempo à 60 car pause de une seconde, et gamme Majeure. Pensez à vérifier que votre synthé envoie bien le code 60 pour C3 (théoriquement le DO du milieu de votre clavier), car cette norme MIDI n'est malheureusement pas respectée par certains constructeurs ; si c'est le cas et que le code 60 corresponde à C2, positionnez notdepl à 72.

Avant de rentrer dans la boucle principale, le running statut NOTE ON est initialisé et le drapeau logique dictee! est positionné sur la valeur FALSE pour interdire d'enregistrer ou jouer avant d'avoir choisi des notes sur une gamme et dans une tonalité.

Boucle principale

La boucle principale est un test de cliquage et de position de souris pour aiguiller le traitement soit sur une modification de paramètres, soit sur une fonction spécifique de la dictée.

Modification des paramètres

L'incrément des valeurs se fait par la touche gauche et le décrément par la touche droite. Le schéma est classique et ne présente aucune particularité si ce n'est le fonctionnement inversé du tempo ; la précision de ce dernier n'étant pas essentielle dans cette version, il est fixé à l'aide de l'instruction PAUSE tempo! qui sépare deux notes successives ; ce temps de pause est donc inversement proportionnel au tempo. Les notes jouées seront considérées comme ayant toutes des valeurs de noires et la pause 50, soit une seconde, comme un tempo de 60 à la noire.

Vous pouvez aussi augmenter le nombre maximal de notes de la dictée, ici 20, mais pensez aussi à augmenter la taille des tableaux **tempsl(( ), jouel( ),enregistrel( )**.

Appel des fonctions

Fonction dictee :

Cette fonction remplit les tableaux **temps(il)** et **jouel(il)**, respectivement avec les durées et les hauteurs de notes correspondant aux gammes et tonalités choisies. Toutes les notes ont la même durée égale à la variable tempo!, mais la mise en place de ce tableau de durée permettra de proposer ultérieurement une lecture rythmique en plus de la lecture mélodique. La première action de ce traitement est d'effacer les résultats éventuels d'une dictée précédente et de positionner le drapeau dictee! sur la valeur TRUE. La sélection de la gamme choisie se fait ensuite au moyen de la variable **gamml**, positionnée dans les paramètres précédents.

Fonction joue

Cette procédure ne fonctionne que si une dictée a été choisie précédemment. Dans le cas contraire, l'invitation de cliquer sur DICTEE est affichée.

Afin de préciser la tonalité choisie, vous avez ici l'option d'entendre la note tonale, mais vous pouvez aussi choisir le LA (code 69), si vous regrettez les dictées plus traditionnelles !

Ensuite, il ne vous reste plus qu'à écouter le nombre de notes correspondant au nombre de notes affichées. Notez ici qu'il vous est tout à fait possible d'avoir choisi une dictée de 10 notes, trouver que c'est trop, et n'en écouter que 6 ; vous ferez votre dictée petit à petit, en augmentant le nombre de notes jusqu'à sa valeur de départ et cela ne modifiera jamais le tableau jouel( ) puisque seule la procédure dictee le fait.

On peut sortir de la fonction avant le déroulement complet de la dictée si on clique sur les deux touches de la souris.

Fonction **enregistre** :

L'entrée de cette procédure est conditionnée de la même façon que dans la fonction joue.

Après le message annonçant le début de l'enregistrement, on se met en attente d'une entrée MIDI pour ne laisser passer que les NOTE ON. La logique de ce filtre est la même que celle de la boîte d'effets. Mais ici on n'enregistre l'octet correspondant à la hauteur que si la vélocité, ici **bl** est différente de zéro ; la procédure **stocke** écrit ces octets dans le tableau **enregistrel( )** à concurrence de vingt données. On peut aussi sortir de cette procédure d'enregistrement soit en cliquant sur les deux touches de la souris et recommencer à zéro en cas d'erreur, soit en frappant une touche quelconque du clavier pour terminer l'enregistrement. Notez que si MOUSEK=3, on rappelle la procédure enregistre alors que l'on est toujours dans la procédure enregistre. C'est une possibilité intéressante qu'offre le Basic GFA à une procédure de s'appeler elle-même (récursivité).

La dernière partie du traitement compare les tableaux **joue!( )** et **enregistrel( )**. La première comparaison se fait sur le nombre de notes enregistrées et présente un message d'erreur dans le cas où ce nombre est différent de **nbrenotl**. La variable octet **bonnotl** permet de comptabiliser dans tous les cas le nombre de notes qui correspondent à la même position dans chacun des tableaux.

La procédure **solution** affiche en notation américaine et joue les notes de la dictée au tempo affiché. Si vous voulez pouvoir afficher plus de 20 notes, il faudra aménager quelque peu cette procédure, soit en réduisant l'espacement entre les notes, soit en modifiant le DEFTEXT à ce moment-là, soit en utilisant une ou plusieurs lignes supplémentaires à l'intérieur de la boucle au moyen d'une ordonnée d'affichage fonction de **ml DIV 15** si vous désirez afficher 15 notes par ligne.

La procédure **efface** remet à blanc tous les emplacements de messages et solutions.

La procédure **message** affiche un message d'erreur lorsque le nombre de notes enregistrées **(il)** ne correspond pas au nombre de notes demandées **(nbrenotl)**



```
quest$="voulez-vous écouterlla note tonalel avant de commencer?"
DEFTEXT 1,1,0.6
DIM majeurl(7),mineurharml(7),mineurmell(7)
DIM bluesl(9),pental(5),diminuel(8),tonl(6),dodecal(12)
DIM gammes$(7),options$(6),tempsl(20),jouel(20),enregistrel(20)
DIM note$(11)
DIM notetot$(127)
FOR il=0 TO 127
   READ notetot$(il)
   EXIT IF notetot$(il)="FNOTE"
NEXT il
DATA .....
DATA C0,C#0,D0,D#0,E0,F0,F#0,G0,G#0,A0,A#0,B0
DATA C1,C#1,D1,D#1,E1,F1,F#1,G1,G#1,A1,A#1,B1
DATA C2,C#2,D2,D#2,E2,F2,F#2,G2,G#2,A2,A#2,B2
DATA C3,C#3,D3,D#3,E3,F3,F#3,G3,G#3,A3,A#3,B3
DATA C4,C#4,D4,D#4,E4,F4,F#4,G4,G#4,A4,A#4,B4
DATA C5,C#5,D5,D#5,E5,F5,F#5,G5,G#5,A5,A#5,B5,FNOTE
FOR il=0 TO 11
   READ note$(il)
NEXT il
notes:
DATA C3,C#3,D3,D#3,E3,F3,F#3,G3,G#3,A3,A#3,B3
FOR il=0 TO 7
   READ majeurl(il)
NEXT il
majeur:
DATA 0,2,4,5,7,9,11,12
FOR il=0 TO 7
   READ mineurharml(il)
NEXT il
mineurharm:
DATA 0,2,3,5,7,8,11,12
FOR il=0 TO 7
   READ mineurmell(il)
NEXT il
mineurmel:
DATA 0,2,3,5,7,9,11,12
FOR il=0 TO 9
   READ bluesl(il)
NEXT il
blues:
DATA 0,2,3,4,5,6,7,9,10,12
FOR il=0 TO 5
   READ pental(il)
NEXT il
penta:
DATA 0,2,4,7,9,12
FOR il=0 TO 8
   READ diminuel(il)
NEXT il
diminue:
DATA 0,2,3,5,6,8,9,11,12
FOR il=0 TO 6
   READ tonl(il)
NEXT il
tons:
DATA 0,2,4,6,8,10,12
FOR il=0 TO 12
   READ dodecal(il)
NEXT il
chromatic:
DATA 0,1,2,3,4,5,6,7,8,9,10,11,12
FOR il=0 TO 6
   READ options$(il)
NEXT il
options:
DATA Tonalité,Gamme,Tempo,nbrenotes,Dictée,Jouer,Enregistrer
```

(suite)

```
FOR il=0 TO 6
  TEXT 80*il+20,20,options$(il)
NEXT il
FOR il=0 TO 7
  READ gammes$(il)
NEXT il
gammes:
DATA Majeur,Minharm,Minmel,Blues,penta,dimin,tons,chroma
notdepl=60
nbrenotl=8
tempol=50
gaml=0
tonalitel=0
TEXT 20,40,note$(tonalitel)
TEXT 100,40,gammes$(gaml)
TEXT 180,40,INT(60*(50/tempol))
TEXT 260,40,nbrenotl
OUT 3,144
dictee!=FALSE
DO
  MOUSE x&,y&,k
  IF k=1
    IF x&>20 AND x&<40 AND y&>30 AND y&<40
      INC tonalitel
      tonalitel=MIN(tonalitel,11)
      TEXT 20,40,"  "
      TEXT 20,40,note$(tonalitel)
      PAUSE 5
    ENDIF
    IF x&>100 AND x&<140 AND y&>30 AND y&<40
      INC gaml
      gaml=MIN(gaml,7)
      TEXT 100,40,"        "
      TEXT 100,40,gammes$(gaml)
      PAUSE 5
    ENDIF
    IF x&>180 AND x&<200 AND y&>30 AND y&<40
      tempol=MAX(tempol,2)
      DEC tempol
      TEXT 180,40,"    "
      TEXT 180,40,INT(60*(50/tempol))
      PAUSE 3
    ENDIF
    IF x&>260 AND x&<280 AND y&>30 AND y&<40
      INC nbrenotl
      nbrenotl=MIN(nbrenotl,20)
      TEXT 260,40,"  "
      TEXT 260,40,nbrenotl
      PAUSE 5
    ENDIF
    IF x&>340 AND x&<380 AND y&>10 AND y&<20
      @dictee
    ENDIF
    IF x&>420 AND x&<460 AND y&>10 AND y&<20
      @joue
    ENDIF
    IF x&>500 AND x&<540 AND y&>10 AND y&<20
      @enregistre
    ENDIF
  ENDIF
  IF k=2
    IF x&>20 AND x&<40 AND y&>30 AND y&<40
      tonalitel=MAX(tonalitel,1)
      DEC tonalitel
      TEXT 20,40,"  "
      TEXT 20,40,note$(tonalitel)
      PAUSE 5
    ENDIF
```

(suite)

```
        IF x&>100 AND x&<140 AND y&>30 AND y&<40
          gaml=MAX(gaml,1)
          DEC gaml
          TEXT 100,40," "
          TEXT 100,40,gammes$(gaml)
          PAUSE 5
        ENDIF
        IF x&>180 AND x&<200 AND y&>30 AND y&<40
          INC tempo
          tempo=MIN(tempo,254)
          TEXT 180,40," "
          TEXT 180,40,INT(60*(50/tempo))
          PAUSE 3
        ENDIF
        IF x&>260 AND x&<280 AND y&>30 AND y&<40
          nbrenotl=MAX(nbrenotl,2)
          DEC nbrenotl
          TEXT 260,40," "
          TEXT 260,40,nbrenotl
          PAUSE 5
        ENDIF
      ENDIF
      IF kl=3
        @reset
      ENDIF
    LOOP
    PROCEDURE reset
      OUT 3,176,123,0,144
    RETURN
    PROCEDURE dictee
      @efface
      dicteel=TRUE
      SELECT gaml
      CASE 0
        FOR il=1 TO nbrenotl
          jouel(il)=notdepl+tonalitel+majeurl(RANDOM(8))
          tempsl(il)=tempo
        NEXT il
        TEXT 200,200,"c'est bon, cliquez sur jouer"
      CASE 1
        FOR il=1 TO nbrenotl
          jouel(il)=notdepl+tonalitel+mineurharml(RANDOM(8))
          tempsl(il)=tempo
        NEXT il
        TEXT 200,200,"c'est fini, cliquez sur jouer"
      CASE 2
        FOR il=1 TO nbrenotl
          jouel(il)=notdepl+tonalitel+mineurmell(RANDOM(8))
          tempsl(il)=tempo
        NEXT il
        TEXT 200,200,"c'est fini, cliquez sur jouer"
      CASE 3
        FOR il=1 TO nbrenotl
          jouel(il)=notdepl+tonalitel+bluesl(RANDOM(10))
          tempsl(il)=tempo
        NEXT il
        TEXT 200,200,"c'est fini, cliquez sur jouer"
      CASE 4
        FOR il=1 TO nbrenotl
          jouel(il)=notdepl+tonalitel+pental(RANDOM(6))
          tempsl(il)=tempo
        NEXT il
        TEXT 200,200,"c'est fini, cliquez sur jouer"
      CASE 5
        FOR il=1 TO nbrenotl
          jouel(il)=notdepl+tonalitel+diminuel(RANDOM(9))
          tempsl(il)=tempo
        NEXT il
```

(suite)

```
      TEXT 200,200,"c'est fini, cliquez sur jouer"
    CASE 6
      FOR i|=1 TO nbrenot|
        joue|(i|)=notdep|+tonalite|+ton|(RANDOM(7))
        temps|(i|)=tempo|
      NEXT i|
      TEXT 200,200,"c'est fini, cliquez sur jouer"
    CASE 7
      FOR i|=1 TO nbrenot|
        joue|(i|)=notdep|+tonalite|+dodeca|(RANDOM(13))
        temps|(i|)=tempo|
      NEXT i|
      TEXT 200,200,"c'est fini, cliquez sur jouer"
    ENDSELECT
  RETURN
  PROCEDURE joue
    IF dictee!=TRUE
      ALERT 2,quest$,1,"oui|non",rep|
      IF rep|=1
        OUT 3,notdep|+tonalite|,80
        PAUSE 100
        OUT 3,notdep|+tonalite|,0
      ENDIF
      TEXT 200,200,"                                                    "
      @efface
      FOR i|=1 TO nbrenot|
        OUT 3,joue|(i|),64
        PAUSE temps|(i|)
        ' PAUSE tempo|
        OUT 3,joue|(i|),0
        EXIT IF MOUSEK=3
      NEXT i|
      @reset
    ELSE
      TEXT 200,200,"CLIQUEZ D'ABORD SUR DICTÉE!"
    ENDIF
  RETURN
  PROCEDURE enregistre
    IF dictee!=TRUE
      TEXT 200,200,"C'EST A VOUS DE JOUER MAINTENANT"
      @efface
      i|=0
      SLPOKE XBIOS(14,2)+6,0
      DO
        IF BIOS(1,3)
          a|=INP(3)
          IF a|<128
            IF note!=TRUE
              b|=INP(3)
              ' a| est la hauteur de la note
              ' b| est la velocité d'attaque
              IF b|
                @stocke
              ENDIF
            ENDIF
          ELSE
            IF a|>143 AND a|<160
              note!=TRUE
              commande|=a|
            ELSE
              note!=FALSE
            ENDIF
          ENDIF
        ENDIF
        EXIT IF BIOS(1,2)
        IF MOUSEK=3
          @reset
          TEXT 200,200,"                                                "
```

(suite)

```
            TEXT 200,200,"REPRENEZ TOUT A ZÉRO"
            PAUSE 50
            @enregistre
          ENDIF
        LOOP
        ~INP(2)
        TEXT 200,200,"                                    "
        @efface
        IF il>nbrenotl
          FOR jl=1 TO nbrenotl
            IF joue(jl)=enregistre(jl)
              INC bonnotl
            ENDIF
          NEXT jl
          TEXT 200,200,"TROP DE NOTES"
          @message
          @solution
        ENDIF
        IF il<nbrenotl
          FOR jl=1 TO il
            IF joue(jl)=enregistre(jl)
              INC bonnotl
            ENDIF
          NEXT jl
          TEXT 200,200,"PAS ASSEZ DE NOTES"
          @message
          @solution
        ENDIF
        IF il=nbrenotl
          FOR jl=1 TO nbrenotl
            IF joue(jl)=enregistre(jl)
              INC bonnotl
            ENDIF
          NEXT jl
          IF bonnotl=nbrenotl
            TEXT 200,200,"SUPER"
          ELSE
            TEXT 200,280,"VOUS  AVEZ "
            TEXT 310,280,bonnotl
            TEXT 330,280," DE JUSTES"
            @solution
          ENDIF
        ENDIF
      ELSE
        TEXT 200,200,"CLIQUEZ D'ABORD SUR DICTÉE!!"
      ENDIF
    RETURN
    PROCEDURE stocke
      IF il<20
        INC il
        enregistre(il)=al
      ENDIF
    RETURN
    PROCEDURE solution
      TEXT 200,340,"VOICI LA SOLUTION"
      FOR ml=1 TO nbrenotl
        OUT 3,joue(ml),64
        TEXT 30*ml,380,notetot$(joue(ml))
        PAUSE tempol
        OUT 3,joue(ml),0
      NEXT ml
    RETURN
    PROCEDURE efface
      TEXT 200,340,"                                  "
      TEXT 200,210,"                                  "
      TEXT 200,220,"                                  "
      TEXT 200,280,"                                  "
      FOR ml=1 TO 20
```

(suite)

```
        TEXT 30*ml,380,"  "
      NEXT ml
    RETURN
    PROCEDURE message
      TEXT 200,210,"VOUS EN AVEZ JOUÉ "
      TEXT 360,210,il
      TEXT 200,220,"IL EN FALLAIT "
      TEXT 360,220,nbrenotl
      TEXT 200,280,"VOUS EN AVEZ "
      TEXT 310,280,bonnotl
      TEXT 330,280," DE JUSTES"
    RETURN
```

```
Tonalité   Gamme      Tempo      nbrenotes Dictée     Jouer      Enregistrer
C3         Blues      60         20
```

```
                    PAS ASSEZ DE NOTES
                    VOUS EN AVEZ JOUÉ    18
                    IL EN FALLAIT        20


                    VOUS EN AVEZ  14  DE JUSTES


                    VOICI LA SOLUTION

A3  E3  E3  D#3 C4  A#3 F3  C3  A#3 E3  G3  A3  D3  C4  C3  F3  D3  G3  D#3 A3
```

6/2.4.6

Principe et construction d'un éditeur de sons

INTRODUCTION

La difficulté dans l'élaboration d'un éditeur de sons vient souvent de la construction de chaque fonction que doit assurer l'éditeur.

C'est le but des programmes qui suivent.

Définissons d'abord le rôle de l'éditeur de sons :

— visualiser les paramètres de synthèse sonore relatifs à un synthétiseur particulier ;

— effectuer les transferts et les sauvegardes des paramètres de synthèse via le système MIDI ou les disquettes ;

— proposer des systèmes simples et variés de construction et comparaison de sons.

Lorsque les paramètres de synthèse ou de fonction sont trop nombreux, il est indispensable de les regrouper en pages différentes, traitement qui n'est pas proposé dans ce programme mais qu'il est facile de réaliser par des SGET et SPUT, et des drapeaux logiques qui mémorisent la page affichée.

Trois programmes de traitement vous sont proposés :

— I. Editeur des paramètres sonores ;

— II. Enveloppe à cinq points avec affichage des paramètres ;

— III. Transfert des données exclusives avec gestion du buffer MIDI.

Dans les exemples qui suivent, un schéma général est adopté pour le principe des messages exclusifs, reportez-vous à la notice de votre synthétiseur pour replacer (et ajouter ou supprimer) les bons codes aux bonnes positions, exception faite des débuts de message exclusif (240 ou F0 en hexadécimal) et fin de message exclusif (247 ou F7 en hexadécimal). Pour la bonne marche visuelle du traitement, une simulation des transferts via le système MIDI est effectuée. Dans le cas de l'enveloppe, il suffit de modifier les paramètres de positionnement de la boîte cadre de l'enveloppe et des facteurs multiplicatifs de l'ambitus des valeurs de paramètres. Prenez au moins 1 comme facteur multiplicatif de la correspondance valeur vers pixel (et de préférence un rapport entier !) pour saisir avec le plus d'exactitude possible les points d'enveloppe par la souris.

I. Editeur des paramètres sonores

BUT DU PROGRAMME

Le but du premier programme de la série est de présenter les paramètres de construction d'un son (ici 51 par défaut), tester les connections MIDI avec le synthétiseur branché, modifier les paramètres du son et les envoyer sur le synthétiseur, effectuer les sauvegardes sur disquette et réaliser les fonctions classiques d'un menu déroulant.

LES FONCTIONS GÉNÉRALES DU PROGRAMME

Le programme se présente en quatre parties principales :

— Initialisation de l'affichage et tests de câblages.

— Définition du menu déroulant.

— Boucle principale de traitement.

— Procédures individuelles de traitement.

• Initialisation de l'affichage et tests de câblage

L'écran doit présenter les boîtes de noms de paramètres et leurs valeurs. Les définitions des paramètres sont arbitraires ici mais elles correspondent à la majorité des abréviations utilisées dans tous les synthétiseurs : OSC pour oscillateur, EG pour générateur d'enveloppe, VCF pour filtre, LEV pour level etc.

• Définition du menu déroulant

Le menu déroulant présente les fonctions classiques relatives au traitement des sons et banques de sons. Ici le nombre des paramètres et des sons dans les banques seront arbitrairement fixés mais l'essentiel est de comprendre le principe.

• Boucle principale de traitement

C'est dans cette boucle que l'on teste la sélection du menu, la sélection du son original ou du son sur lequel on travaille, ainsi que la modification éventuelle des paramètres du son.

• Procédures individuelles de traitement

Les procédures effectuent l'affichage, le branchement des options du menu, la réception et l'envoi des paramètres du son via le système MIDI.

LE PROGRAMME EN DÉTAIL

• Initialisation et affichage

Les variables I& et I1& servent à fixer les largeurs des colonnes de présentation des paramètres du son. La colonne de gauche contient l'appellation du paramètre, les trois suivantes contiennent respectivement la valeur maxi-



male possible pour le paramètre, la valeur actuelle et la valeur de départ du paramètre. Le tableau **nom$( )** contient le nom des paramètres, le tableau **max|( )** contient les valeurs maximales des parametres, le tableau **val|( )** la valeur actuelle du paramètre, et le tableau **original|( )** la valeur de départ du paramètre. Après avoir dessiné les lignes et les boîtes de présentation, le tableau **m.option$( )** de définition du menu est rempli par les données suivant l'étiquette **m.datas**.

Le test de réglage du canal MIDI de réception peut être utile si votre synthé possède un paramètre de réglage du canal de réception pour message exclusif. Ici le test est court-circuité en absence de synthétiseur. Il faut introduire un test en aller retour avec le synthé le cas échéant en positionnant la valeur de la réponse sur la variable **rep1|** en lieu et place de la boîte d'alerte.

La procédure **id__request** est présente sur presque tous les synthétiseurs. C'est une demande d'identification du modèle de synthétiseur. La réponse doit être exacte puisqu'un éditeur ne fonctionne qu'avec un synthétiseur particulier. C'est le rôle de la variable chaîne de caractères **test$** du début du programme. Si votre synthé ne possède pas ce type de message exclusif, il faut supprimer cet appel de procédure. La procédure **recoit__son** lit les paramètres du son qui est contenu dans la mémoire tampon du synthétiseur, en charge les tableaux **val|( )** et **original|( )** effectue l'affichage des valeurs puis sélectionne le son de travail par la procédure **scratch**.

- Le menu déroulant

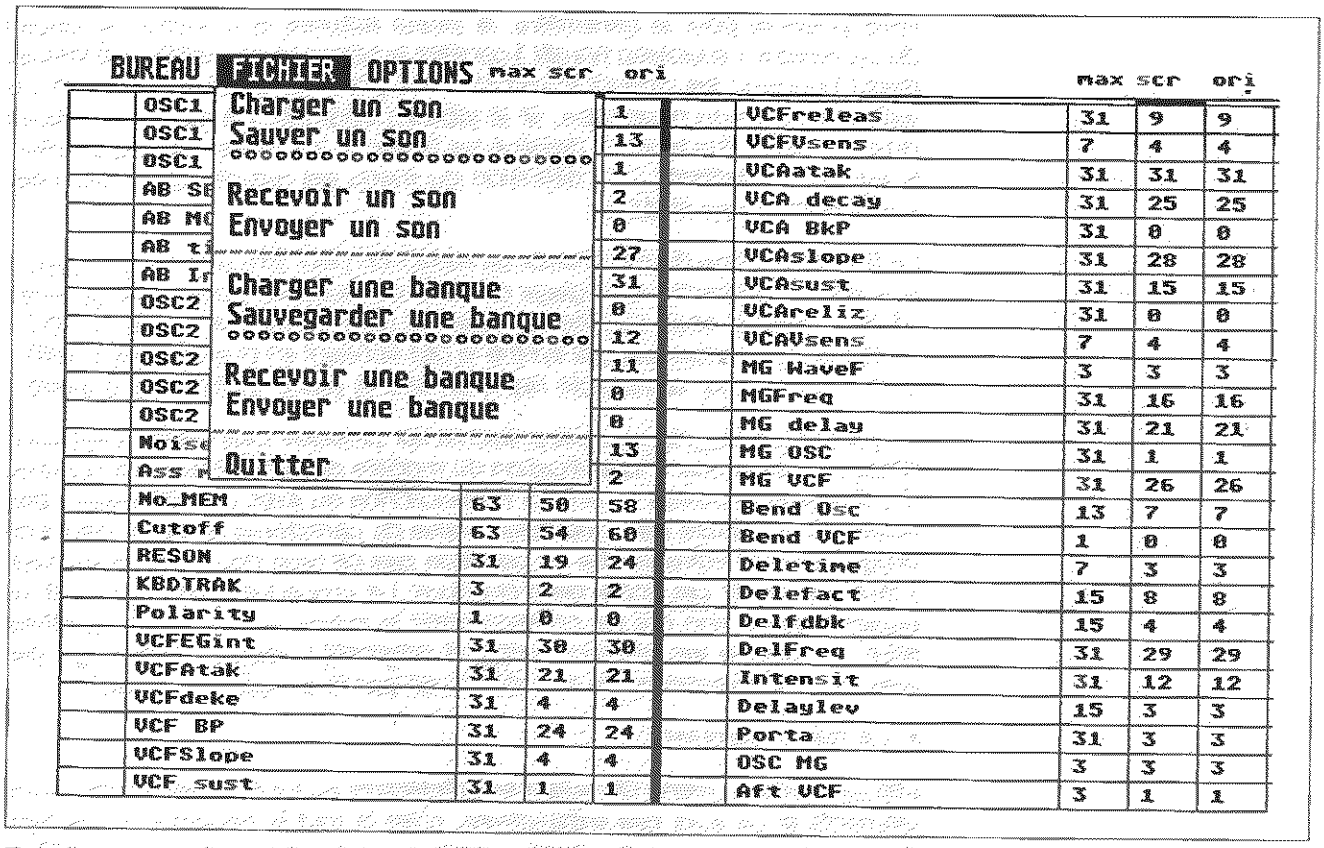
Les fonctions du menu déroulant se préciseront au fur et à mesure des compléments et ne sont pas exhaustives, c'est là tout le bénéfice de ce type de présentation qui pourra aussi examiner vos suggestions le cas échéant.

- Boucle principale de traitement

Elle est inclue dans la boucle DO LOOP.

La première instruction teste la sélection éventuelle d'une option du menu. On ne passe ensuite dans les différents traitements que si l'on a cliqué sur la souris. Le premier test de position de click de souris détermine si les rectangles de sélection du son scratch ou de l'original ont été sélectionnés. Les procédures scratch ou original sélectionnent graphiquement leur rectangle et désélectionnent l'autre rectangle. Les procédures **envoiscratch** ou **envoioriginal** envoient les paramètres du son sélectionné dans le buffer du synthétiseur. La partie principale de la boucle teste un click éventuel de la souris sur l'une des cases paramètres du son **scratch**.

Le click gauche incrémente les valeurs et le click droit décrémente les valeurs. Cette opération ne peut se faire que si le son **scratch** est sélectionné, ce qui correspond à positionner la variable logique **original !** sur la valeur FALSE. Le test de la position de click, l'incrémentation de la valeur du paramètre correspondant (bornée supérieurement dans **max|( )**, il suffit de regarder dans le mode d'emploi dans la zone réservée aux messages exclusifs pour trouver la bonne valeur), ainsi que la décrémentation (bornée inférieurement par la valeur zéro) se font de façon classique. La seule originalité est d'introduire les fonctions **MODULO 25** et **DIV 25** pour séparer en une seule boucle les deux colonnes et les 25 lignes par colonne.



Exemple de menu déroulant.

- Les procédures

Procédure ***requestdata*** *:*

Après avoir remis le buffer de l'Atari à zéro, le message de demande d'envoi du son contenu dans le buffer du synthétiseur est envoyé. La PAUSE 5 est nécessaire pour permettre à la majorité des synthétiseurs d'avoir le temps de réagir. Dans le programme, on suppose que le message exclusif envoyé par le synthétiseur comporte un Header de 7 octets, valeur qui dépend de chaque synthétiseur. Il suffit de remplir ensuite les tableaux **original( )** et **val( )** par les 51 valeurs des paramètres du son. La simulation de cette réception MIDI se fait par génération aléatoire d'une valeur comprise entre 0 et la valeur maximale permise pour chaque paramètre.

Procédure ***choix*** *:*

C'est la procédure de branchement sur les options du menu par des tests sur les chaînes de caractères des énonces d'options. Pensez à respecter les espaces et les majuscules éventuelles.

Procédure ***id_request*** *:*

Après avoir remis le buffer MIDI de l'Atari à zéro, on envoie le message de demande d'identification au synthétiseur et on compare la réponse à ce



qu'elle devrait être, en bloquant la poursuite du programme jusqu'à ce que le test soit positif.

*Procédures **scratch** et **original** :*

Elles sélectionnent les rectangles situés respectivement sous les appellations **scratch** et **original** en positionnant la variable logique **original!** sur la valeur **TRUE** dans le cas de sélection de la boîte **scratch**, et sur la valeur FALSE si on sélectionne la case **original**.

*Procédures **envoioriginal** et **envoiscratch** :*

Après envoi du Header relatif à l'envoi de l'ensemble des paramètres du son (5 octets ici) on envoie tous les paramètres du son contenus dans le tableau **original|( )** ou le tableau **val|( )**. Le message exclusif se termine par le code 247.

*Procédure **parametre** :*

Après envoi du Header relatif au message de changement de valeur de paramètre (ici 5 octets), on envoie l'adresse du paramètre (ici simplement son offset dans la liste des paramètres), puis sa valeur suivi du code 247 de fin de message exclusif.

*Procédure **charge__son** :*

Cette procédure charge un fichier son avec suffixe **SND** et décode caractère par caractère la chaîne chargée dans la variable **transfert$**. La longueur de la chaîne n'est pas codée dans le fichier ; on suppose donc que le suffixe **SND** n'est appliqué qu'aux paramètres de sons provenant du synthétiseur étudié. Attention aux contrefaçons ! pour plus de sécurité, il faudrait coder les 51 valeurs de paramètres en leur adjoignant une en-tête de fichier pour ne prendre que les fichiers corrects parmi tous ceux qui possèdent un suffixe **SND**. Les valeurs décodées sont chargées dans les tableaux **val|( )** et **original|( )** puis affichées et envoyées sur le synthétiseur.

*Procédure **sauve__son** :*

Le tableau **val|( )** est codé en une chaîne de 51 caractères puis sauvegardé avec le suffixe **SND** sans en-tête de fichier.

*Procédures **envoie__son** et **recoit__son** :*

Ces procédures remplissent et lisent le buffer du synthétiseur. La procédure **recoit__son** réalise en plus l'affichage des paramètres reçus.

*Procédure **compare** :*

C'est la fonction compare classique des synthétiseurs. Elle trouve quel est de l'**original** ou du **scratch**, le son sélectionné. Elle sélectionne l'autre son par l'intermédiaire d'une première boîte d'alerte avant de revenir au premier son par l'intermédiaire d'une seconde boîte d'alerte.

Procédure ***voix_init :***

Tous les synthétiseurs ne possèdent pas cette fonction de *voice init*. La simulation de l'initialisation se fait ici par une affectation arbitraire de valeurs de paramètres égales à la moitié entière de la valeur maximale permise. Sur les synthétiseurs qui possèdent cette fonction, il suffit d'envoyer le message exclusif correspondant. Pour tous les autres synthétiseurs, il est intéressant de pouvoir définir quelles sont les différentes voix initiales que l'on peut définir, suivant les familles de timbres spécifiques que l'on cherche à approcher. Il sera intéressant de définir des voix initiales pour des sons cuivrés, nappés, percussifs ou autres, et ceci à votre convenance. Il suffit de créer des tableaux de valeurs initiales de paramètres attachées à chaque famille de timbres et de réaliser le transfert du tableau choisi vers les tableaux **val|( )** et **original|( )**. Mais ceci sera l'objet d'un futur complément.

```
l&=318
ll&=30
test$=CHR$(240)+CHR$(66)+CHR$(48)+CHR$(3)+CHR$(247)
compare1$="TAPERSURUNE|     TOUCHE | POUR JOUER | L'ANCIEN "
compare2$="TAPERSURUNE|     TOUCHE | POUR JOUER | LE NOUVEAU"
transfert$=SPACE$(51)
DIM nom$(50),val|(50),max|(50),original|(50)
DATA OSC1 OCT,3,OSC1 WF,15,OSC1 LEV,31,AB SEL
DATA 3,AB MODE,1,AB time,31,AB Int,31
DATA OSC2 OCT,3,OSC2 WF,15,OSC2 LEV,31,OSC2 int
DATA 5,OSC2 det,7,Noise LI,31,Ass mode,3,No_MEM,63
DATA Cutoff,63,RESON,31,KBDTRAK,3,Polarity
DATA 1,VCFEGint,31,VCFAtak,31,VCFdeke,31
DATA VCF BP,31,VCFSlope,31,VCF sust,31
DATA VCFreleas,31,VCFVsens,7,VCAatak,31
DATA VCA decay,31,VCA BkP,31,VCAslope,31
DATA VCAsust,31,VCAreliz,31,VCAVsens,7
DATA MG WaveF,3,MGFreq,31,MG delay,31
DATA MG OSC,31,MG VCF,31,Bend Osc,13
DATA Bend VCF,1,Deletime,7,Delefact,15
DATA Delfdbk,15,DelFreq,31,Intensit,31
DATA Delaylev,15,Porta,31,OSC MG
DATA 3,Aft VCF,3,Aft VCA,3
FOR il=0 TO 50
  READ nom$(il),max|(il)
NEXT il
f=21
g=396
DEFLINE 1,2,3
BOX 2,f,637,g
DEFLINE 1,3,0,0
LINE 317,f,317,g
```



(suite)

```
DEFLINE 1,2,3
LINE 37,f,37,g
LINE 212,f,212,g
LINE 248,f,248,g
LINE 284,f,284,g
LINE 354,f,354,g
LINE 529,f,529,g
LINE 565,f,565,g
LINE 601,f,601,g
h=3
i=639
DEFTEXT 1,0,1,6
DEFLINE 1,1,3
FOR johnl=0 TO 24
  LINE 3,21+15*johnl,639,21+15*johnl
  TEXT 42,31+15*johnl,nom$(johnl)
  TEXT 360,31+15*johnl,nom$(johnl+25)
  TEXT 218,31+15*johnl,maxl(johnl)
  TEXT 536,31+15*johnl,maxl(johnl+25)
NEXT johnl
PBOX 248,18,284,21
PBOX 248+317,18,284+317,21
DIM m.option$(52)
RESTORE m.datas
FOR il=0 TO 50
  READ m.option$(il)
  EXIT IF m.option$(il)="~~~~~"
NEXT il
m.option$(il)=""
m.datas:
DATA BUREAU
DATAInformation
DATA, , , , , ,""
DATA FICHIER
DATACharger un son
DATASauver un son
DATA**********
DATARecevoir un son
DATAEnvoyer un son
DATA ----------
DATACharger une banque
DATASauvegarder une banque
DATA**********
DATARecevoir une banque
DATAEnvoyer une banque
DATA ----------, Quitter,""
DATA OPTIONS
```

(suite)

```
DATA  Masque de copie
DATA ~~~~~~~~~~~~~~~~~~
DATA  Masque aléatoire
DATA ~~~~~~~~~~~~~~~~~~
DATA  Comparer
DATA ~~~~~~~~~~~~~~~~~~
DATA  Voix initiale
DATA ~~~~~~~~~~~~~~~~~~
DATAInscrire le son
DATAInscrire la banque,""
DATA ~~~~
MENU m.option$()
ON MENU GOSUB choix
TEXT 225,10,"max"
TEXT 255,10,"scr"
TEXT 295,10,"ori"
TEXT 535,10,"max"
TEXT 565,10,"scr"
TEXT 605,10,"ori"
parametre$="SYNTHE  sur  Canal  1|   REGLER LE | PARAMETRE N°? | "
REPEAT
  ALERT 2,parametre$,1," OUI | NON ",rep1|
  rep1|=1
UNTIL rep1|=1
@id_request
@recoit_son
DO
  ON MENU
  MOUSE x&,y&,k|
  IF k|
    IF x&>247 AND x&<286 AND y&<21
      IF original!=TRUE
        @scratch
        @envoiscratch
      ENDIF
    ENDIF
    IF x&>279 AND x&<318 AND y&<21
      IF original!=FALSE
        @original
        @envoioriginal
      ENDIF
    ENDIF
    IF original!=FALSE
      FOR john|=0 TO 49
        IF x&>254+1&*(john| DIV 25) AND x&<254+11&+1&*(john| DIV 25)
          IF y&<31+15*(john| MOD 25) AND y&>21+15*(john| MOD 25)
            IF k|=1
```



(suite)

```
                REPEAT
                  INC val(johnl)
                  val(johnl)=MIN(val(johnl),maxl(johnl))
                  TEXT 254+l&*(johnl DIV 25),31+15*(johnl MOD 25),"   "
                  TEXT 254+l&*(johnl DIV 25),31+15*(johnl MOD 25),val(johnl)
                  @parametre(johnl)
                  PAUSE 5
                UNTIL MOUSEK=0
              ENDIF
              IF k|=2
                REPEAT
                  val(johnl)=MAX(val(johnl),1)
                  DEC val(johnl)
                  TEXT 254+l&*(johnl DIV 25),31+15*(johnl MOD 25),"   "
                  TEXT 254+l&*(johnl DIV 25),31+15*(johnl MOD 25),val(johnl)
                  @parametre(johnl)
                  PAUSE 5
                UNTIL MOUSEK=0
              ENDIF
            ENDIF
          ENDIF
        NEXT johnl
      ENDIF
    ENDIF
  LOOP
  PROCEDURE requestdata
    SLPOKE XBIOS(14,2)+6,0
    OUT 3,240,66,48,3,16,247
    PAUSE 5
    buffer$=INPMID$
    FOR il=0 TO 50
      val(il)=ASC(MID$(buffer$,il+6))
      original(il)=val(il)
    NEXT il
    ' ---------simulation de l'envoi de paramètres---------
    ' ------------par le synthétiseur-------------
    FOR il=0 TO 50
      val(il)=RANDOM(maxl(il)+1)
      original(il)=val(il)
    NEXT il
  RETURN
  PROCEDURE choix
    MENU OFF
    ml=MENU(0)
    IF m.option$(ml)=" Information"
      @information
    ENDIF
```

(suite)

```
    IF m.option$(ml)=" Charger un son"
      @charge_son
    ENDIF
    IF m.option$(ml)=" Sauver un son"
      @sauve_son
    ENDIF
    IF m.option$(ml)=" Recevoir un son"
      @recoit_son
    ENDIF
    IF m.option$(ml)=" Envoyer un son"
      @envoie_son
    ENDIF
    IF m.option$(ml)=" Charger une banque"
      @charge_banque
    ENDIF
    IF m.option$(ml)=" Sauver une banque"
      @sauve_banque
    ENDIF
    IF m.option$(ml)=" Recevoir une banque"
      @recoit_banque
    ENDIF
      IF m.option$(ml)=" Envoyer une banque"
        @envoie_banque
      ENDIF
      IF m.option$(ml)=" Masque de copie"
        @mask_cop
      ENDIF
      IF m.option$(ml)=" Masque aléatoire"
        @mask_alea
      ENDIF
      IF m.option$(ml)=" Comparer"
        @compare
      ENDIF
      IF m.option$(ml)=" Voix initiale"
        @voix_init
      ENDIF
      IF m.option$(ml)=" Inscrire le son"
        @inscrit_son
      ENDIF
      IF m.option$(ml)=" Inscrire la banque"
        @inscrit_banque
      ENDIF
      IF m.option$(ml)=" Quitter"
        @quitter
      ENDIF
    RETURN
    PROCEDURE id_request
```



(suite)

```
      SLPOKE XBIOS(14,2)+6,0
      OUT 3,240,66,64,247
      PAUSE 1
      buffer$=INPMID$
      IF buffer$<>test$
        ALERT 1,"   VERIFIER | LES CONNECTIONS!!",1,"continue",rep2|
        ' Rappeler la meme procédure tant que le test n'est pas bon
        '  @id_request
      ENDIF
    RETURN
    PROCEDURE scratch
      DEFFILL 1
      PBOX 248,19,286,20
      PBOX 565,19,603,20
      original!=FALSE
      DEFFILL 0
      PBOX 285,19,317,20
      PBOX 602,19,634,20
    RETURN
    PROCEDURE original
      DEFFILL 0
      PBOX 248,19,286,20
      PBOX 565,19,603,20
      original!=TRUE
      DEFFILL 1
      PBOX 285,19,317,20
      PBOX 602,19,634,20
    RETURN
    PROCEDURE envoioriginal
      OUT 3,240,66,48,3,64
      FOR il=0 TO 50
        OUT 3,original(il)
      NEXT il
      OUT 3,247
    RETURN
    PROCEDURE envoiscratch
      OUT 3,240,66,48,3,64
      FOR il=0 TO 50
        OUT 3,val(il)
      NEXT il
      OUT 3,247
    RETURN
    PROCEDURE parametre(kooll)
      OUT 3,240,66,48,3,65,kooll,val(kooll),247
    RETURN
    PROCEDURE information
      phrase1$="  ce logiciel n'est pas encore terminé"
```

(suite)

```
    phrase1$=phrase1$+"| mais la fin approche"
    ALERT 1,phrase1$,2,"bravolcourage",rep%
  RETURN
  PROCEDURE quitter
    EDIT
  RETURN
  PROCEDURE charge_son
    FILESELECT "A:\*.SND","",nom$
    IF nom$="" OR RIGHT$(nom$)="\"
      GOTO ff2
    ENDIF
    BLOAD nom$,VARPTR(transfert$)
    FOR il=0 TO 50
      val(il)=ASC(MID$(transfert$,il+1,1))
      original(il)=val(il)
    NEXT il
    @aff_ori
    @aff_val
    @envoioriginal
    @envoiscratch
    @scratch
    ff2:
  RETURN
  PROCEDURE sauve_son
    FILESELECT "A:\*.SND","",nom$
    IF nom$="" OR RIGHT$(nom$)="\"
      GOTO ff1
    ENDIF
    IF INSTR(nom$,".")<>0
      nom$=LEFT$(nom$,INSTR(nom$,"."))+"SND"
    ELSE
      nom$=nom$+".SND"
    ENDIF
    FOR il=0 TO 50
      MID$(transfert$,il+1,1)=CHR$(val(il))
    NEXT il
    BSAVE nom$,VARPTR(transfert$),51
    ff1:
  RETURN
  PROCEDURE recoit_son
    @requestdata
    @aff_val
    @aff_ori
    @scratch
  RETURN
  PROCEDURE envoie_son
    @envoiscratch
```

(suite)

```
RETURN
PROCEDURE charge_banque
RETURN
PROCEDURE sauve_banque
RETURN
PROCEDURE recoit_banque
RETURN
PROCEDURE envoie_banque
RETURN
PROCEDURE mask_cop
RETURN
PROCEDURE mask_alea
RETURN
PROCEDURE compare
  IF original!=TRUE
    ALERT 1,compare1$,1,"OK",rep2|
    @scratch
    @envoiscratch
    ALERT 1,compare2$,1,"OK",rep2|
    @original
    @envoioriginal
  ELSE
    ALERT 1,compare1$,1,"OK",rep2|
    @original
    @envoioriginal
    ALERT 1,compare2$,1,"OK",rep2|
    @scratch
    @envoiscratch
  ENDIF
RETURN
PROCEDURE voix_init
  ' Simulation d'une voix init
  FOR johnl=0 TO 49
    original|(johnl)=max|(johnl) DIV 2
    val|(johnl)=original|(johnl)
  NEXT johnl
  @aff_val
  @aff_ori
RETURN
PROCEDURE inscrit_son
RETURN
PROCEDURE inscrit_banque
RETURN
PROCEDURE aff_ori
  FOR johnl=0 TO 49
    TEXT 290+l&*(johnl DIV 25),31+15*(johnl MOD 25),original|(johnl)
  NEXT johnl
```

(suite)

```
    RETURN
    PROCEDURE aff_val
      FOR johnl=0 TO 49
        TEXT 254+1&*(johnl DIV 25),31+15*(johnl MOD 25),val(johnl)
      NEXT johnl
    RETURN
```



II. ADSR à cinq points avec affichage des paramètres

BUT DU PROGRAMME

Réaliser un schéma simple d'enveloppe à cinq points AD S1 S2 R avec correspondance de points écran et de valeurs de paramètres.

Réciproquement, la décrémentation ou l'incrémentation des valeurs de paramètres redéfinit le tracé de l'enveloppe. Le générateur d'enveloppe peut ensuite s'appliquer indifféremment à un amplificateur, un oscillateur ou un filtre.

LES FONCTIONS GÉNÉRALES DU PROGRAMME

— Initialiser l'affichage et les tableaux de départ.

— Boucle de test de prise de points d'enveloppe ou de transformation des valeurs des paramètres.

— Procédures de traitements particuliers.

LE PROGRAMME EN DÉTAIL

Initialisation de l'affichage et des tableaux :

C'est le rôle de la procédure **init**. Le tableau contenant les coordonnées des cinq points actifs de l'ADSR est séparé en **x&( )** pour les abcisses et **y&( )** pour les ordonnées en vue d'utilisation par les instructions *POLYLINE* et *POLYMARK*. Les tableaux **xpar&( )** et **ypar&( )** contiennent les valeurs des paramètres de synthèse relatifs au synthétiseur traité. Le tableau **xpar&( )** correspond aux temps ou aux vitesses suivant les marques de synthétiseurs, ici ce sont des temps. Le tableau **ypar&( )** retranscrit les niveaux. Pour passer des pixels aux valeurs des paramètres, il faut réaliser une proportion. Le rapport est de 3 pour les x et de 9 pour les y. Ce choix est arbitraire et dépend du nombre d'enveloppes dont vous voulez disposer à l'écran. Ici, l'ambitus des valeurs va de 0 à 31. Les variations corrigées par le coefficient multiplicateur sont donc de 93 pour les temps **(xmax&)** et de 279 pour les niveaux **(ymax&)**.

x0& et **y0&** sont les positions de départ de la boîte et, **l&** et **h&** la longueur et la hauteur de la boîte. Ce sont les valeurs de ces paramètres qui vous permettront de modifier la taille et l'emplacement de votre enveloppe. La procédure **aff(1)** affiche l'enveloppe sur les points correspondant à l'initialisation des tableaux **x&( )** et **y&( )**. La procédure **para( )** calcule et affiche les valeurs corrigées des paramètres.

BOUCLE DE TRAITEMENT

C'est la suite d'instructions comprises dans le DO LOOP. Lecture de la souris avec sortie si click simultané sur les deux touches. Le traitement se décompose en deux :

— Prise des points de l'enveloppe.

— Modification des valeurs des paramètres.

Prise des points de l'enveloppe :

(IF kl = 1). Le traitement général de la prise de points d'enveloppe se décompose en trois suivant qu'on clique sur un point avant le point de *sustain*, sur le point de *sustain*, ou sur le point de *release*. La notion de point est étendue au motif de marquage de point introduit par POLYMARK ici de taille 8 pixels. En déplaçant un des points avant le *sustain*, il faut répercuter ces variations graphiques sur les points situés à droite sans modifier les paramètres de ces points : c'est le rôle de la procédure avsus.

Si l'on clique sur le dernier point de *sustain*, on ne peut en modifier que le niveau et le répercuter sur le premier point de *sustain* : c'est le rôle de la procédure *sus*.

Si l'on clique sur le point de *release*, cela n'affecte en rien les autres points de l'enveloppe. Seul le temps du *release* est changé puisque son niveau reste constamment à zéro : c'est le rôle de la procédure rel.

Modification des valeurs des paramètres

(IF kl = 1 OR kl = 2). La même distinction se fait au niveau des trois traitements envisagés suivant qu'on clique sur un paramètre d'un point avant le dernier point de *sustain* (procédure paravsus), sur le dernier point de *sustain* (procédure parasus) ou le point de *release* (pararel).

LES PROCÉDURES DE TRAITEMENT

Procédure avsus :

Après avoir effacé la position actuelle de l'enveloppe et du marquage des points (aff(0)), on établit les nouvelles positions du point sélectionné par la souris. La logique est la suivante :

Les nouvelles coordonnées suivent la souris. Les x du nouveau point sont bornés à droite et à gauche par les x du point suivant et précédent corrigés par le marquage (8).

Les y sont limités par les dimensions du cadre corrigées par dl. Le déplacement en y ne modifie en rien les y des autres points de l'enveloppe. Le déplacement en x doit être répercuté sur les points suivants, c'est le rôle de la

variable **di&** qui est additionnée aux points situés à droite du point sélectionné par la souris, dans la boucle :

FOR jl=il+1 TO 3

Il suffit ensuite de réafficher la nouvelle position de l'enveloppe **(aff(1))**, de retranscrire les nouvelles valeurs des paramètres et d'appeler **para(4)** si on a touché au premier point de *sustain*.

*Procédure **sus** :*

On appelle d'abord la procédure **aff(0)** qui efface la position actuelle de la ligne et le marquage des points. La nouvelle position en **y** suit la souris mais est bornée par les dimensions du cadre diminué de la valeur de **di**. La nouvelle valeur du *sustain* est ensuite affectée au premier point de *sustain*. La procédure **aff(1)** affiche la nouvelle enveloppe et marque les nouveaux points. Les nouvelles valeurs des paramètres sont calculées et affichées par les procédures **para(il)**.

*Procédure **re1** :*

Le principe est le même, mais on ne modifie que la position en x du release. La nouvelle position suit la souris mais est bornée à droite par le point de *sustain* augmenté de l'ambitus maximal **(xmax&)**. Elle est bornée à gauche par l'abcisse du point de *sustain*. Ces valeurs sont corrigées par la dimension du marquage de point(8). La nouvelle valeur du *release* est affichée et calculée par la procédure **re1**.

*Procédure **para( ) :***

Cette procédure fait la conversion des valeurs pixels, **x&(il)** et **y&(il)**, en valeurs de paramètres, **xpar&(il)** et **ypar&(il)**, par un simple rapport d'échelle de trois pour les x et de neuf pour les y. Les nouvelles valeurs des **xpar&( )** et **ypar&( )** sont affichées.

*Procédure **parasus** :*

jl=1 indique qu'on a cliqué sur la case des niveaux, seul paramètre que l'on peut modifier sur le second point de *sustain*. On incrémente à gauche et décrémente à droite en bornant les valeurs par 31 et 0. Il suffit ensuite de réaliser l'affichage comme dans les procédures précédentes : effacer l'ancienne enveloppe par **aff(0)**, calculer les nouvelles coordonnées pixels **y&( )** et en multipliant les différences des **ypar&** par 9 et réafficher la nouvelle position d'enveloppe par **aff(1)**.

Procédure ***pararel*** *:*

jl = 0 indique qu'on a cliqué dans la case des temps, seul paramètre qu'on ait le droit de modifier sur le point de *release*. Les bornes sont les mêmes que précédemment et le calcul de la nouvelle abcisse suit une logique équivalente. On multiplie la différence des temps par 3 en corrigeant le résultat par la largeur du marquage de point (8) pour interdire toute superposition de points à l'écran.

Procédure ***paravsus*** *:*

jl = 1 indique qu'on a sélectionné les niveaux et la logique de traitement est identique à celle de la procédure **parasus**.

jl = 0 indique qu'on a sélectionné les temps, l'incrémentation ou la décrémentation des paramètres **xpar&(il)** se fait de façon classique en bornant les valeurs par **0** et **31**. L'appel de la procédure **aff(0)** efface la position courante de l'enveloppe. La nouvelle différence correspondante entre les **x&(il)** et **x&(il – 1)** est calculée en multipliant le **xpar&(il)** par 3, puis en lui rajoutant la dimension du marquage (8). Il reste ensuite à répercuter cette nouvelle valeur des temps sur les points suivants de l'enveloppe (bouche **FOR NEXT**) et afficher l'enveloppe avec ses nouveaux points.



PROGRAMME

Exécuter le programme suivant :

```
@init
DO
  MOUSE x&,y&,k|
  EXIT IF k|=3
  IF k|=1
    FOR i|=1 TO 3
      IF x&<x&(i|)+3 AND x&>x&(i|)-3 AND y&>y&(i|)-3 AND y&<y&(i|)+3
        @avsus
      ENDIF
    NEXT i|
    IF x&<x&(4)+3 AND x&>x&(4)-3 AND y&>y&(4)-3 AND y&<y&(4)+3
      @sus
    ENDIF
    IF x&<x&(5)+3 AND x&>x&(5)-3 AND y&>y&(5)-3 AND y&<y&(5)+3
      @rel
    ENDIF
  ENDIF
  IF k|=1 OR k|=2
    FOR i|=1 TO 5
      FOR j|=0 TO 1
        IF x&>90*i| AND x&<90*i|+20 AND y&>335+20*j| AND y&<350+20*j|
          IF i|=4
            @parasus
          ELSE
            IF i|=5
              @pararel
            ELSE
              @paravsus
            ENDIF
          ENDIF
        ENDIF
      NEXT j|
    NEXT i|
  ENDIF
LOOP
PROCEDURE init
  DIM x&(5),y&(5),xpar&(5),ypar&(5)
  xmax&=93
  ymax&=279
  x0&=20
  y0&=20
  l&=500
  h&=300
  hh&=60
```

(suite)

```
    BOX x0&,y0&,x0&+l&,y0&+h&
    BOX x0&,y0&+h&,x0&+l&,y0&+h&+hh&
    dl=6
    y1&=y0&+h&
    x&(0)=x0&+dl
    y&(0)=y1&-dl
    x&(1)=x&(0)+xmax&-5
    y&(1)=150
    x&(2)=x&(1)+xmax&-15
    y&(2)=50
    x&(3)=x&(2)+xmax&-10
    y&(3)=200
    x&(4)=400
    y&(4)=y&(3)
    x&(5)=x&(4)+xmax&
    y&(5)=y&(0)
    @aff(1)
    FOR il=1 TO 5
      @para(il)
    NEXT il
  RETURN
  PROCEDURE aff(cl)
    COLOR cl
    DEFMARK cl,4
    POLYLINE 6,x&(),y&()
    POLYMARK 6,x&(),y&()
  RETURN
  PROCEDURE avsus
    REPEAT
      @aff(0)
      xi&=MOUSEX
      xi&=MAX(x&(il-1)+8,xi&)
      xi&=MIN(xi&,x&(il-1)+xmax&+8)
      y&(il)=MAX(MOUSEY,y1&-ymax&-dl)
      y&(il)=MIN(y&(il),y1&-dl)
      di&=xi&-x&(il)
      FOR jl=il+1 TO 3
        ADD x&(jl),di&
      NEXT jl
      x&(il)=xi&
      y&(4)=y&(3)
      @aff(1)
      @para(il)
      IF il=3
        @para(4)
      ENDIF
    UNTIL MOUSEK=0
  RETURN
```



(suite)

```
PROCEDURE sus
  REPEAT
    @aff(0)
    y&(4)=MAX(MOUSEY,y1&-ymax&-dl)
    y&(4)=MIN(y&(4),y1&-dl)
    y&(3)=y&(4)
    @aff(1)
    @para(3)
    @para(4)
  UNTIL MOUSEK=0
RETURN
PROCEDURE rel
  REPEAT
    @aff(0)
    x&(5)=MAX(MOUSEX,x&(4)+8)
    x&(5)=MIN(x&(5),x&(4)+xmax&+8)
    @aff(1)
    @para(5)
  UNTIL MOUSEK=0
RETURN
PROCEDURE para(il)
  IF il<>4
    xpar&(il)=(x&(il)-x&(il-1)-8) DIV 3
  ENDIF
  ypar&(il)=(y&(0)-y&(il)) DIV 9
  TEXT 90*il,350,"    "
  TEXT 90*il,370,"    "
  TEXT 90*il,350,xpar&(il)
  TEXT 90*il,370,ypar&(il)
RETURN
PROCEDURE parasus
  IF jl=1
    IF kl=1
      INC ypar&(4)
      ypar&(4)=MIN(ypar&(4),31)
    ELSE
      DEC ypar&(4)
      ypar&(4)=MAX(ypar&(4),0)
    ENDIF
    @aff(0)
    y&(4)=-9*ypar&(4)+y&(0)
    y&(3)=y&(4)
    @para(4)
    @para(3)
    @aff(1)
  ENDIF
RETURN
```

(suite)

```
PROCEDURE pararel
  IF jl=0
    IF kl=1
      INC xpar&(5)
      xpar&(5)=MIN(xpar&(5),31)
    ELSE
      DEC xpar&(5)
      xpar&(5)=MAX(xpar&(5),0)
    ENDIF
    @aff(0)
    x&(5)=x&(4)+8+3*xpar&(5)
    @para(5)
    @aff(1)
  ENDIF
RETURN
PROCEDURE paravsus
  IF jl=1
    IF kl=1
      INC ypar&(il)
      ypar&(il)=MIN(ypar&(il),31)
    ELSE
      DEC ypar&(il)
      ypar&(il)=MAX(ypar&(il),0)
    ENDIF
    @aff(0)
    y&(il)=-9*ypar&(il)+y&(0)
    @para(il)
    IF il=3
      y&(4)=y&(3)
      @para(4)
    ENDIF
    @aff(1)
  ELSE
    IF kl=1
      INC xpar&(il)
      xpar&(il)=MIN(xpar&(il),31)
    ELSE
      DEC xpar&(il)
      xpar&(il)=MAX(xpar&(il),0)
    ENDIF
    @aff(0)
    x&(il)=x&(il-1)+8+3*xpar&(il)
    @para(il)
    FOR jjl=il+1 TO 3
      x&(jjl)=x&(jjl-1)+8+3*xpar&(jjl)
    NEXT jjl
    @aff(1)
  ENDIF
RETURN
```



III. Transfert des données exclusives

But du programme

Permettre à partir d'un synthétiseur le transfert de données quelconques en système exclusif (dump de son, ou de banque, ou dump total) sa sauvegarde sur disquette, ou le chargement à partir d'une disquette puis le transfert des données chargées vers le synthétiseur.

Les fonctions générales

— Initialisation des adresses du buffer MIDI.

— Boucle de traitement avec test de sauvegarde ou de chargement des données.

— Procédures de traitement.

Le programme en détail

Initialisation

C'est le rôle de la procédure **init**. On y définit une variable chaîne de caractères **(dump$)** de taille 32000 octets destinée à prendre la place du buffer MIDI de l'Atari qui lui ne contient que 128 octets. Le buffer MIDI de l'Atari fonctionne comme un tampon circulaire. L'index de tête **(tetebuf%)** indique la prochaine position d'écriture et l'index de queue **(quebuf%)** donne la position de lecture. Si le buffer n'est pas plein, moins de 128 octets, il suffit de mettre 0 dans l'index de tête pour vider le buffer MIDI ; pour des raisons de prudence il est préférable de mettre la même valeur (ici 0) dans ces deux index pour vider le buffer MIDI. Les buffers du clavier et de la RS 232 fonctionnent d'ailleurs de la même façon, la seule différence résidant dans le numéro de sélection du périphérique, 1 pour le clavier et 0 pour la RS 232.

L'adresse du buffer est stockée dans **adrbuf%**, la taille du buffer est stockée dans **longbuf%** et les deux pointeurs d'écriture et de lecture sont stockés dans les deux variables précitées plus haut.

Boucle de traitement

Ce programme tourne en boucle et vous présente en permanence les options de :

— sauver des datas sur disquette après avoir effectué un dump par le système MIDI ;

— charger des datas à partir d'un fichier disquette et les envoyer sur le synthétiseur ;

— sortir du programme.

Sauver les datas après un dump

Après affichage de l'opération en cours, l'adresse du buffer MIDI est transférée dans celle de la variable **dump$** et les tailles des tampons sont échangées. Le buffer MIDI est ensuite vidé. Le programme se met en attente de lecture des informations MIDI. Dès que vous démarrez le dump manuel du synthétiseur (ou un transfert midi quelconque), la variable **dump$** enregistre les données automatiquement à concurrence de sa taille. Le programme ne prévoit pas de solution si votre transfert dépasse 32000 octets, mais le problème ne se pose pas dans les cas usuels.

On sort de la boucle d'enregistrement dès que la fin du message exclusif est signifiée par la valeur **eox|** ou dès qu'on appuie sur une touche du clavier.

Dès que le transfert MIDI est terminé, on repositionne le buffer MIDI de l'Atari à son ancienne adresse puis on le remet à zéro. Les instructions qui suivent traitent la chaîne **dump$** afin d'en isoler le message exclusif. Les octets situés à gauche du code de début de message exclusif (240) sont supprimés et la chaîne est lue jusqu'au message de fin de message exclusif (eox|) par l'intermédiaire de **dump%** qui compte le nombre d'octets jusqu'au code 247. La variable **exclusif$** contient le message exclusif complet code 240 et 247 compris. Si le message MIDI n'était pas du type exclusif, ou s'il a été interrompu par une touche du clavier, le transfert est annulé après détection d'une longueur de message égale à zéro.

L'impression de cette variable sur l'écran ne l'est que pour mémoire, car si vous dépassez les 500 octets il vous sera difficile de tout visualiser. La variable **exclusif$** est ensuite stockée sur disquette de façon classique.

Charger les datas d'une disquette

Le fichier est ouvert de façon classique puis décrypté et codé dans la variable **dump$**. Après le message de positionnement du synthétiseur en acceptation de chargement de message exclusif, les octets du message exclusif sont envoyés sur le port **MIDI OUT**.

LES PROCÉDURES

Procédure ***bougebuffer*** :

Stockage de l'adresse et de la taille de la variable **dump$** dans les adresses et tailles du buffer MIDI de l'Atari données par la fonction **XBIOS(14,2)**.

Procédure ***remetbuffer*** :

Restitution des adresse et taille du buffer MIDI. L'échange des valeurs de ces paramètres s'effectue en sens inverse du précédent.



Procédure videbuffer :

Remet à zéro les pointeurs de lecture et écriture du buffer.

```
' ------------------ D U M P     M I D I ----------
@init
exclu$="DUMP DES DONNEES EXCLUSIVES| "
' ---------------------------------------------
'
DO
  ALERT 1,exclu$,3,"CHARGER|SAUVER|SORTIR",rep&
  ' ---------------------------------------------
  IF rep&=3
    EDIT
  ENDIF
  ' ---------------------------------------------
  IF rep&=2
    CLS
    PRINT "SAUVEGARDE DES DONNEES DANS :";
    @select
    PRINT fichier$
    @bougebuffer
    @videbuffer
    PRINT
    PRINT"TRANSMETTEZ LES DONNEES EXCLUSIVES PAR UN DUMP MANUEL"
    PRINT"POUR ARRETER LE TRANSFERT EN COURS APPUYER SUR UNE TOUCHE"
    PRINT "                    DU CLAVIER ATARI"
    midi|=0
    REPEAT
      IF BIOS(1,3)
        midi|=INP(3)
      ENDIF
      EXIT IF midi|=eox|
    UNTIL BIOS(1,2)
    ~INP(2)
    @remetbuffer
    @videbuffer
    dump%=INSTR(dump$,CHR$(eox|))
    IF dump%=0
      ALERT 3,"  ANNULATION|ERREUR OU BUFFER SATURE",1,"FIN",re&
      EDIT
    ENDIF
    exclusif$=LEFT$(dump$,dump%)
    WHILE LEFT$(exclusif$)<>CHR$(240)
      exclusif$=MID$(exclusif$,2)
    WEND
    PRINT exclusif$
    IF fichier$<>""
      BSAVE fichier$,VARPTR(exclusif$),dump%
    ENDIF
```

(suite)

```
  ENDIF
  ' ------------------------------------
  IF rep&=1
    CLS
    PRINT "CHARGER LES DONNEES DE:";
    @select
    PRINT fichier$
    OPEN "I",#1,fichier$
    OPEN "",#2,"MID:"
    lgfich%=LOF(#1)
      BGET #1,VARPTR(dump$),lgfich%
      BPUT #2,VARPTR(dump$),lgfich%
      CLOSE
      mess$="  POSITIONNEZ| LE SYNTHE EN "
      mess$=mess$+"| RECEPTION MIDI| MESSAGE EXCLUSIF"
      ALERT 3,mess$,1,"FIN",re&
      FOR i|=1 TO lgfich%
        OUT 3,ASC(MID$(dump$,i|,1))
      NEXT i|
      CLS
      ALERT 3,"FIN DU TRANSFERT",1,"SUPER!",rep&
    ENDIF
  LOOP
  ' ------------PROCEDURES----------
  PROCEDURE init
    eox|=247
    HIDEM
    dump$=STRING$(32000,CHR$(254))
    pointbuf%=XBIOS(14,2)
    adrbuffer%=LPEEK(pointbuf%)
    longbuffer%=DPEEK(pointbuf%+4)
    tetebuffer%=LPEEK(pointbuf%+6)
    queubuffer%=LPEEK(pointbuf%+8)
  RETURN
  PROCEDURE bougebuffer
    SLPOKE pointbuf%,VARPTR(dump$)
    SDPOKE pointbuf%+4,32000
  RETURN
  PROCEDURE remetbuffer
    SDPOKE pointbuf%+4,longbuffer%
    SLPOKE pointbuf%,adrbuffer%
  RETURN
  PROCEDURE videbuffer
    SLPOKE pointbuf%+6,0
    SLPOKE pointbuf%+8,0
  RETURN
```



(suite)

```
PROCEDURE select
  encore:
  FILESELECT "A:\*.SND","",fichier$
  IF fichier$="A:\"
    GOTO encore
  ENDIF
  IF fichier$=""
    END
  ENDIF
RETURN
```

6/2.4.7

Dictée rythmique

Le rythme fait partie des composantes indispensables à la connaissance musicale. Ce programme, écrit en GFA, fera de vous un champion toutes catégories de la dictée rythmique et de la gestion de tableau à deux dimensions.

DESCRIPTIF GÉNÉRAL DU PROGRAMME

— Génération aléatoire d'une dictée rythmmique.
— Choix parmi 22 formules rythmiques.
— Ecoute de la dictée à volonté.
— Changement du tempo à tout moment.
— Affichage du résultat.
— Métronome (4 tops avant le départ).
— Fonctionne avec la puce son interne.
— Nécessite un écran monochrome.

Les figures rythmiques utilisées sont les plus courantes, si toutefois il s'avère que la dictée est trop dure, il vous est toujours possible de brider le logiciel dans le nombre de figures que comporte la dictée ainsi que le nombre de motifs utilisés.

PROGRAMMATION

Les tableaux

Les motifs rythmiques sont décrits de deux manières différentes, le tableau **rythme(0,0)** pour l'affichage et le tableau **jouer(0,0)** pour la génération sonore.

Les sprites utilisés dans ce programme sont identiques au programme « Créer un éditeur de son » (tableau **note$( )** se trouvant au chapitre 6/1.3.8. Il vous suffit donc de recopier l'intégralité de la procédure **charge_sprites** de ce logiciel précédent. Attention, l'emplacement de la procédure est juste indiqué dans le programme « dictée rythmique ».

• Tableau pour l'affichage

Le tableau rythme possède deux dimensions. La première valeur indique le numéro de la figure alors que la deuxième valeur décrit la construction de la figure.

Descriptif de la deuxième dimension :

1re valeur : Nombre de symboles note.
2e valeur : Décalage en x de l'affichage du premier symbole.
3e valeur : N° du premier symbole dans le tableau note$( ).
4e valeur : Décalage en x de l'affichage du deuxième symbole.
5e valeur : N° du deuxième symbole dans le tableau note$( ).
''
'' Deux données par symbole supplémentaire.
''
6e valeur : N° de ligne dans le symbole.
7e valeur : Décalage en x du début de la ligne.
8e valeur : Décalage en y du début de la ligne.
9e valeur : Décalage en x de la fin de la ligne.
10e valeur : Décalage en y de la fin de la ligne.
''
'' Quatre données par ligne supplémentaire.
''
11e valeur : 1 si triolet, sinon 0

A noter que deux croches ne s'affichent pas avec deux symboles croche, mais avec deux noires reliées par une ligne, de même quatre doubles croches ne s'affichent pas avec quatre symboles double croche mais avec quatre noires reliées par deux lignes.

- Tableau pour l'écoute

Le tableau **jouer(0,0)** possède deux dimensions. La première valeur indique le numéro de la figure alors que la deuxième valeur décrit la durée rythmique de la figure.

Descriptif de la deuxième dimension :

1re valeur : Nombre de symboles.
2e valeur : 1 si note à jouer, 0 si silence.
3e valeur : Valeur de la note en temps;
4 = ronde, 2 = blanche, 1 = noire, 0.5 = croche,
0.25 = double croche, 0.33 = croche en triolet.
4e valeur : 1 si note à jouer, 0 si silence.
5e valeur : Valeur de la note en temps.
''
'' Deux données par symbole supplémentaire.
''

Les procédures

*Procédure **init_variables** :*

Cette procédure initialise toutes les variables en début de programme. C'est dans cette procédure que les tableaux sont remplis grâce à l'instruction **READ DATA**. En début de procédure se trouvent l'initialisation de la puce son et la suppression de l'action du clavier sur la puce son.

Toutes les valeurs d'affichage sont initialisées et une dictée est générée par défaut à l'aide de la fonction **RANDOM(22)**. Ensuite c'est l'initialisation des tableaux **rythme(0,0)** et **jouer(0,0)**.

*Procédure **Fond** :*

Cette procédure crée la présentation générale du programme.

*Procédure **affiche_rythme(a,px,py)** :*

Cette procédure affiche chaque figure rythmique. **a** représente le numéro de la figure dans le tableau rythme (première dimension), **px** est la position en x de l'affichage, **py** est la position en y de l'affichage.

base_des est l'indice de déplacement dans la deuxième dimension du tableau rythme. **Base_des** est initialisé à 1 en début de procédure car l'indice 0 indique le nombre de symboles à afficher. **Base_des = 0** est utilisé pour la première boucle **FOR NEXT**. A chaque cycle, **Base_des** est incrémenté de 2. La deuxième boucle gère les lignes à afficher pour la liaison graphique des notes.

Le test **if rythme(a,base_des) = 1** détermine s'il faut afficher un trois au-dessus du symbole rythmique.

*Procédure **tempo** :*

Cette procédure gère la variable tempo de 40 à 240 tops par minute. C'est la valeur métronomique de la dictée.

*Procédure **nouveau** :*

Cette procédure génère une nouvelle dictée rythmique. Le nombre de figures est décrit par **a%**, alors que le choix de la figure est décrit par **RANDOM(22)**, sélection parmi 22 possibilités. Si vous désirez limiter le logiciel, ce sont les deux valeurs qu'il faut manipuler. La fonction **PBOX** efface l'ancienne dictée de l'écran, les variables **ppx,ppy,nb_ry**, pour l'affichage de votre réponse, sont initialisées. A chaque demande de nouvelle dictée, la fonction **ecouter** est appelée.

*Procédure **ecouter** :*

Cette procédure permet d'écouter la dictée rythmique. La variable **unit_temps%** est calculée à chaque lancement de procédure car il est possible de changer à tout moment le tempo. Suit l'initialisation de la puce **son**.

La première boucle génère les 4 tops avant le démarrage de la dictée. Ensuite c'est le tableau **jouer(0,0)** qui prend le relais dans la deuxième

boucle. La variable **b%** décrit le numéro de la figure dans le tableau alors que **bas_j** est la deuxième dimension et sert de descripteur de la figure.

Le Beep est réactivé lorsque la figure rythmique est supérieure à un temps. La variable **voix** est utilisée pour la polyphonie entre le canal un de la puce **son** qui joue la dictée et le canal gauche qui est le Beep.

La variable **buffer** contient le nombre de **s/200 – 4**. Le **– 4** sert à arrêter la note avant sa fin théorique, car si deux notes se suivent, la deuxième attaque ne serait pas perceptible. Deux noires seraient interprétées comme une blanche. La valeur retranchée **(– 4)** est ajoutée en fin de traitement par la fonction **PAUSE 1** qui équivaut à **4 s/200**.

*Procédure **resultat** :*

Cette procédure joue la dictée rythmique générée par la procédure **Nouveau**. Elle passe les paramètres du tableau **dictee(0)** à la procédure **affiche_rythme(0,0)**.

*Procédure **quitter** :*

Fin du programme.

*Procédure **beep** :*

Génère un Beep très court, c'est le métronome.

La gestion du programme est assurée dans la boucle **DO/LOOP** ; tous les accès s'effectuent à la souris.

Le programme

```
!
!       DICTEE RYTHMIQUE
!
!
!       ECRAN MONOCHROME
!
!
!


init_variable
charge_sprites
fond
DO
  IF GEMDOS(11)
    IF INP(2)=8 AND nb_ry>0
      DEC nb_ry
      SUB ppx,65
      DEFFILL 0,0
      PBOX ppx,95,ppx+65,200
```

(suite)

```
        DEFFILL 1,0
      ENDIF
    ENDIF
    IF MOUSEK
      MOUSE x,y,k
      IF y>365 AND y<386
        sel=INT(x/127)+1
        ON sel GOSUB tempo,nouveau,ecouter,resultat,quitter
      ENDIF
      init_num=0
      num_ry=50
      FOR a=5 TO 620 STEP 57
        IF x>a AND y>280 AND x<a+57 AND y<320
          num_ry=init_num
        ENDIF
        IF x>a AND y>320 AND x<a+57 AND y<360
          num_ry=init_num+11
        ENDIF
        INC init_num
      NEXT a
      IF num_ry<>50 AND nb_ry<8
        affiche_rythme(num_ry,ppx,ppy)
        ADD ppx,65
        INC nb_ry
        REPEAT
        UNTIL MOUSEK=0
      ENDIF
    ENDIF
  LOOP
  PROCEDURE init_variable
    SPOKE &H484,14          ! supprine le beep clavier
    SOUND 1,0,2,2           ! sonorité de la dictée
    SPOKE &HFF8800,9        !
    SPOKE &HFF8802,15       ! initialisation
    SPOKE &HFF8800,8        !
    SPOKE &HFF8802,15       ! de la puce
    SPOKE &HFF8800,7        !
    SPOKE &HFF8802,255      ! son
    voix=255                ! variable gérant la polyphonie
    tempo=60                ! tempo par défaut
    DIM dictee(7)           ! tableau recevant la dictée
    FOR a%=0 TO 7
      dictee(a%)=RANDOM(22)
    NEXT a%
    ppx=30
    ppy=150
    nb_ry=0
```

(suite)

```
DIM rythme(21,20)
FOR a%=0 TO 4
  READ rythme(0,a%)  !ronde
NEXT a%
DATA 1,14,0,0,0
FOR a%=0 TO 4
  READ rythme(1,a%)  !pause
NEXT a%
DATA 1,10,5,0,0
FOR a%=0 TO 4
  READ rythme(2,a%)  !blanche
NEXT a%
DATA 1,10,1,0,0
FOR a%=0 TO 4
  READ rythme(3,a%)  !demi pause
NEXT a%
DATA 1,10,6,0,0
FOR a%=0 TO 4
  READ rythme(4,a%)  !noire
NEXT a%
DATA 1,10,2,0,0
FOR a%=0 TO 4
  READ rythme(5,a%)  !soupir
NEXT a%
DATA 1,10,7,0,0
FOR a%=0 TO 10
  READ rythme(6,a%)  !deux croches
NEXT a%
DATA 2,7,2,20,2,1,16,0,36,0,0
FOR a%=0 TO 6
  READ rythme(7,a%)  !1/2soupir     croche
NEXT a%
DATA 2,7,8,20,3,0,0
FOR a%=0 TO 18
  READ rythme(8,a%)  !4 double croches
NEXT a%
DATA 4,0,2,12,2,12,2,12,2,2,2,9,0,45,0,9,4,45,4,0
FOR a%=0 TO 18
  READ rythme(9,a%)  !1/4 soupir 3 double croches
NEXT a%
DATA 4,0,9,12,2,12,2,12,2,2,2,21,0,45,0,21,4,45,4,0
FOR a%=0 TO 16
  READ rythme(10,a%)  !2 double croches 1 croche
NEXT a%
DATA 3,5,2,12,2,12,2,2,14,0,38,0,14,4,26,4,0
FOR a%=0 TO 16
  READ rythme(11,a%)  !1 croche 2 double croches
```

(suite)

```
NEXT a%
DATA 3,0,2,24,2,12,2,2,9,0,45,0,34,4,45,4,0
FOR a%=0 TO 20
  READ rythme(12,a%)  !double croche  croche       double croche
NEXT a%
DATA 3,0,2,12,2,24,2,3,9,0,45,0,9,4,14,4,40,4,45,4,0
FOR a%=0 TO 16
  READ rythme(13,a%)  !croche 1/4 soupir double croche
NEXT a%
DATA 3,0,2,20,9,16,2,2,9,0,45,0,40,4,45,4,0
FOR a%=0 TO 16
  READ rythme(14,a%)  !1/2 soupir 2 double croches
NEXT a%
DATA 3,0,8,24,2,12,2,2,33,0,45,0,33,4,45,4,0
FOR a%=0 TO 16
  READ rythme(15,a%)  !2 double croches 1/2 soupir
NEXT a%
DATA 3,5,2,12,2,12,8,2,14,0,26,0,14,4,26,4,0
FOR a%=0 TO 16
  READ rythme(16,a%)  !1/4 soupir croche  double croche
NEXT a%
DATA 3,5,9,10,2,21,2,2,24,0,45,0,40,4,45,4,0
FOR a%=0 TO 12
  READ rythme(17,a%)  !3 croches en triolet
NEXT a%
DATA 3,5,2,16,2,16,2,1,14,0,46,0,1
FOR a%=0 TO 6
  READ rythme(18,a%)  !noire croche en triolet
NEXT a%
DATA 2,5,2,28,3,0,1
FOR a%=0 TO 6
  READ rythme(19,a%)  !croche noire en triolet
NEXT a%
DATA 2,5,3,14,2,0,1
FOR a%=0 TO 12
  READ rythme(20,a%)  !1/2 soupir 2 croches en triolet
NEXT a%
DATA 3,5,8,14,2,14,2,1,28,0,42,0,1
FOR a%=0 TO 6
  READ rythme(21,a%)  !soupir croche en triolet
NEXT a%
DATA 2,5,7,28,3,0,1
DIM jouer(21,8)
FOR a%=0 TO 2
  READ jouer(0,a%)   !ronde
NEXT a%
DATA 1,1,4
```

(suite)

```
FOR a%=0 TO 2
  READ jouer(1,a%)   !pause
NEXT a%
DATA 1,0,4
FOR a%=0 TO 2
  READ jouer(2,a%)   !blanche
NEXT a%
DATA 1,1,2
FOR a%=0 TO 2
  READ jouer(3,a%)   !1/2 pause
NEXT a%
DATA 1,0,2
FOR a%=0 TO 2
  READ jouer(4,a%)   !noire
NEXT a%
DATA 1,1,1
FOR a%=0 TO 2
  READ jouer(5,a%)   !soupir
NEXT a%
DATA 1,0,1
FOR a%=0 TO 4
  READ jouer(6,a%)   !2 croches
NEXT a%
DATA 2,1,0.5,1,0.5
FOR a%=0 TO 4
  READ jouer(7,a%)   !1/2 soupir croche
NEXT a%
DATA 2,0,0.5,1,0.5
FOR a%=0 TO 8
  READ jouer(8,a%)   !4 double croches
NEXT a%
DATA 4,1,0.25,1,0.25,1,0.25,1,0.25
FOR a%=0 TO 8
  READ jouer(9,a%)   !1/4 soupir 3 double croches
NEXT a%
DATA 4,0,0.25,1,0.25,1,0.25,1,0.25
FOR a%=0 TO 6
  READ jouer(10,a%)  !2 double croches 1 croche
NEXT a%
DATA 3,1,0.25,1,0.25,1,0.5
FOR a%=0 TO 6
  READ jouer(11,a%)  !1 croche 2 double croches
NEXT a%
DATA 3,1,0.5,1,0.25,1,0.25
FOR a%=0 TO 6
  READ jouer(12,a%)  !1 double 1 croche 1 double croche
NEXT a%
```

(suite)

```
    DATA 3,1,0.25,1,0.5,1,0.25
    FOR a%=0 TO 6
      READ jouer(13,a%)   !1 croche 1/4 soupir 1 double croche
    NEXT a%
    DATA 3,1,0.5,0,0.25,1,0.25
    FOR a%=0 TO 6
      READ jouer(14,a%)   !1/2 soupir 2 double croches
    NEXT a%
    DATA 3,0,0.5,1,0.25,1,0.25
    FOR a%=0 TO 6
      READ jouer(15,a%)   !2 double croches 1/2 soupir
    NEXT a%
    DATA 3,1,0.25,1,0.25,0,0.5
    FOR a%=0 TO 6
      READ jouer(16,a%)   !1/4 soupir 1 croche 1 double croche
    NEXT a%
    DATA 3,0,0.25,1,0.5,1,0.25
    FOR a%=0 TO 6
      READ jouer(17,a%)   !triolet de croches
    NEXT a%
    DATA 3,1,0.34,1,0.33,1,0.33
    FOR a%=0 TO 4
      READ jouer(18,a%)   !1 noire 1 croche en triolet
    NEXT a%
    DATA 2,1,0.67,1,0.33
    FOR a%=0 TO 4
      READ jouer(19,a%)   !1 croche 1 noire en triolet
    NEXT a%
    DATA 2,1,0.34,1,0.66
    FOR a%=0 TO 6
      READ jouer(20,a%)   !1/2 soupir 2 croches en triolet
    NEXT a%
    DATA 3,0,0.34,1,0.33,1,0.33
    FOR a%=0 TO 4
      READ jouer(21,a%)   !soupir 1 croche en triolet
    NEXT a%
    DATA 2,0,0.67,1,0.33
  RETURN
  >>>>    PROCEDURE charge_sprites  <<<<
  >>>>    RETURN                    <<<<
  PROCEDURE fond
    FOR a=2 TO 540 STEP 127
      BOX a,365,a+127,386
    NEXT a
    PRINT AT(24,2);"******  DICTEE RYTHMIQUE  ******"
    PRINT AT(5,24);"TEMPO  ";USING "###",tempo
    PRINT AT(21,24),"NOUVEAU"
    PRINT AT(37,24);"ECOUTER"
```

(suite)

```
    PRINT AT(52,24);"RESULTAT"
    PRINT AT(69,24);"QUITTER"
    FOR a=5 TO 620 STEP 57
      BOX a,280,a+57,320
      BOX a,320,a+57,360
    NEXT a
    py=293
    px=10
    FOR a=0 TO 10
      affiche_rythme(a,px,py)
      ADD px,57
    NEXT a
    py=333
    px=10
    FOR a=11 TO 21
      affiche_rythme(a,px,py)
      ADD px,57
    NEXT a
  RETURN
  PROCEDURE affiche_rythme(a,px,py)
    base_des=1
    basex=px
    FOR nb=1 TO rythme(a,0)
      ADD basex,rythme(a,base_des)
      INC base_des
      PUT basex,py,la_note$(rythme(a,base_des)),7
      INC base_des
    NEXT nb
    DEFLINE 1,3
    FOR nb=1 TO rythme(a,base_des)
      INC base_des
      LINE px+rythme(a,base_des),py+rythme(a,base_des+1),
                          px+rythme(a,base_des+2),py+rythme(a,base_des+3)
      ADD base_des,3
    NEXT nb
    DEFLINE 1,1
    INC base_des
    DEFTEXT 1,0,0,6
    IF rythme(a,base_des)=1
      TEXT px+25,py-4,"3"
    ENDIF
  RETURN
  PROCEDURE tempo
    IF MOUSEK=1
      DEC tempo
      tempo=MAX(tempo,40)
    ELSE
```

(suite)

```
      INC tempo
      tempo=MIN(tempo,240)
    ENDIF
    PRINT AT(12,24);USING "###",tempo
  RETURN
  PROCEDURE nouveau
    FOR a%=0 TO 7
      dictee(a%)=RANDOM(22)
    NEXT a%
    DEFFILL 0,0
    PBOX 1,90,640,200
    DEFFILL 1,0
    ppx=30
    ppy=150
    nb_ry=0
    ecouter
  RETURN
  PROCEDURE ecouter
    unit_temps%=(60/tempo)*200
    SPOKE &HFF8800,7
    SPOKE &HFF8802,255
    SPOKE &HFF8800,8
    SPOKE &HFF8802,15
    FOR a=1 TO 4
      beep
      t%=TIMER
      REPEAT
      UNTIL TIMER-t%>unit_temps%
    NEXT a
  FOR a%=0 TO 7
    b%=dictee(a%)
    beep
    bas_j=0
    FOR l%=1 TO jouer(b%,0)
      r%=TIMER
      t%=TIMER
      INC bas_j
      IF jouer(b%,bas_j)=1
        SPOKE &HFF8800,7
        SPOKE &HFF8802,254
        voix=254
      ELSE
        SPOKE &HFF8800,7
        SPOKE &HFF8802,255
        voix=255
      ENDIF
      INC bas_j
```

(suite)

```
        buffer=(jouer(b%,bas_j)*unit_temps%)-4
        REPEAT
          IF TIMER-t%>unit_temps%
            beep
            t%=TIMER
          ENDIF
        UNTIL TIMER-r%>buffer
        IF voix=254
          SPOKE &HFF8800,7
          SPOKE &HFF8802,255
        ENDIF
        PAUSE 1
      NEXT l%
    NEXT a%
    SPOKE &HFF8800,7
    SPOKE &HFF8802,255
    voix=255
  RETURN
  PROCEDURE resultat
    px=30
    py=100
    FOR a%=0 TO 7
      affiche_rythme(dictee(a%),px,py)
      ADD px,65
    NEXT a%
    ecouter
  RETURN
  PROCEDURE quitter
    END
  RETURN
  PROCEDURE beep
    SPOKE &HFF8800,7
    SPOKE &HFF8802,voix AND 253
    SPOKE &HFF8802,voix
  RETURN
```

6/3

Plus loin

6/3.1

Les possibilités sonores de l'Atari STE

Dans la version STE, l'Atari propose une puce son supplémentaire qui permet la gestion et la lecture d'échantillons sonores. Ce soundchip est relié au DMA (Direct Memory Access) et n'utilise pas le 68000. La taille de l'échantillon est uniquement limité par la taille mémoire de l'Atari, des réglages tel que les graves, les aigus et le volume sont prévus. Une option de bouclage de l'échantillon est aussi disponible.

Deux sorties audio au format cinch ont été ajoutées sur l'Atari pour le raccordement à un ampli ou une chaîne Hi-Fi, ce qui n'empêche pas l'utilisation du haut-parleur interne du moniteur.

L'échantillonnage

L'échantillonnage est une technique d'enregistrement qui au lieu d'utiliser la bande magnétique pour le stockage, utilise la RAM d'un ordinateur. Une conversion de la donnée audio en un signal numérique est donc obligatoire. C'est un convertisseur analogique digital qui se charge de cette fonction, le signal analogique est analysé toutes les n fois, puis converti en données

numériques. Chaque mesure réalisée est un échantillon. La qualité de l'échantillon dépend de deux critères, le nombre d'échantillons pris par seconde qui se nomme la fréquence d'échantillonnage, et le mode de convertion du convertisseur, 8, 12 ou 16 bits. Par exemple le compact disque a une fréquence d'échantillonnage de 44.1 KHZ, soit 44 100 échantillons par seconde, et le codage s'effectue sur 16 bits, 0 à 65535.

L'Atari STE sans atteindre cette résolution offre néanmoins une bonne qualité de reproduction. L'Atari STE ne possède que le système de lecture d'échantillons, il est donc dédié à la reproduction de morceaux pré-enregistrés.

UTILISATION DU SOUNDCHIP

Le mode mono et stéréo

Le soundchip possède deux modes de travail, mono et stéréo. Suivant le mode utilisé, le codage est différent, mais attention le soundchip ne travaille qu'en mots (2 octets).

- Mode monophonique

L'échantillon est codé en octets lus l'un à la suite de l'autre, mais l'adresse doit toujours être paire pour le travail en mots.

- Mode stéréophonique

Les échantillons sont codés différemment sur un mot. L'octet de poids fort du mot représente la valeur pour le canal gauche, l'octet de poids faible représente la valeur pour le canal droit.

Codage de l'échantillon

Le convertisseur de l'Atari STE est un convertisseur 8 bits, ce qui lui donne la possibilité de gérer 256 états différents.

Les valeurs de l'échantillon vont de – 128 à + 127, le 0 représente le silence, ce qui diffère du codage habituellement utilisé par les échantillonneurs, la valeur 128 représentant le silence pour un convertisseur 8 bits.

Pour utiliser des échantillons venant d'un autre système, une convertion sera généralement nécessaire.

Les registres de contrôle du soundchip

Le soundchip utilise 13 registres de $FF8900 à FF8924.

- Registre de contrôle du DMA : lecture/écriture

$FF8900 ....nn
nn
00 : pas de son DMA
01 : utilisation du DMA, une lecture
11 : utilisation du DMA, lecture en boucle



• **Registre d'adresse de départ du son** : lecture/écriture

$FF8902	00nnnnnn	haut
$FF8904	nnnnnnnn	med
$FF8906	nnnnnnn0	bas

L'adresse de début de l'échantillon est inscrite à ces adresses,

l'adresse de l'échantillon doit être paire.

• **Registre du compteur d'adresse** : lecture

$FF8908	00nnnnnn	haut
$FF890A	nnnnnnnn	med
$FF890C	nnnnnnn0	bas

Ces adresses permettent de suivre l'évolution de la lecture DMA en indiquant l'adresse actuellement en lecture. L'écriture à ces registres est sans effet.

• **Registre de fin de son** : lecture/écriture

$FF890E	00nnnnnn	haut
$FF8910	nnnnnnnn	med
$FF8912	nnnnnnn0	bas

L'adresse de fin de l'échantillon est inscrite à ces adresses,

l'adresse doit être paire.

• **Registre de mode de lecture de l'échantillon** : lecture/écriture

$FF8920 00000000x00000nn

x
0: lecture stéréophonique
1: lecture monophonique

nn fréquence de lecture
00: 6258 hz (hertz)
01: 12517 hz
10: 25033 hz
11: 50066 hz

L'Atari STE dispose de 4 fréquences de lecture de l'échantillon, plus la fréquence est élevée et plus le son est haut.

• **Registre de contrôle de MICROWIRE** : lecture/écriture

$FF8922	nnnnnnnn nnnnnnnn	donnée
$FF8924	nnnnnnnn nnnnnnnn	masque

Le MICROWIRE gère la communication entre le soundchip DMA, l'ancienne puce son et le LMC 1992. Le LMC permet d'intervenir sur le son du soundchip, il agit sur le volume, les graves et les aigus de l'échantillon. Pour envoyer un message au LMC par le MICROWIRE il faut initialiser le registre de masque. Plusieurs possibilités s'offrent à nous pour la constitution du masque, nous ne retiendrons que la plus simple. Le registre de masque est toujours égal à $7FF. Pour la petite histoire, le masque indique le nombre (toujours 11) ainsi que l'emplacement des bits utilisés dans le registre de données.

Si le masque est à $7FF les données sont organisées comme suit :

	0	7	F	F
Masque	0000	0111	1111	1111
Données	0000	0ppx	xxnn	nnn

pp adresse du LMC toujours égal à 10
xx numéro de commande
nn valeur de la commande

Les commandes xx

011 Volume général
Valeur de la commande de 0 à 40 0 = volume maximum

101 Volume canal gauche
Valeur de la commande de 0 à 20 0 = volume maximum

100 Volume canal droit
Valeur de la commande de 0 à 20 0 = volume maximum

010 Aigu
Valeur de la commande de 0 à 10 6 = sans effet

001 Grave
Valeur de la commande de 0 à 10 6 = sans effet

000 Mixage
4 valeurs possibles
00 volume −12dB
01 mixage avec l'ancienne puce son
10 ne se mélange pas avec l'ancienne puce son
11 réservé

Exemple de communication :

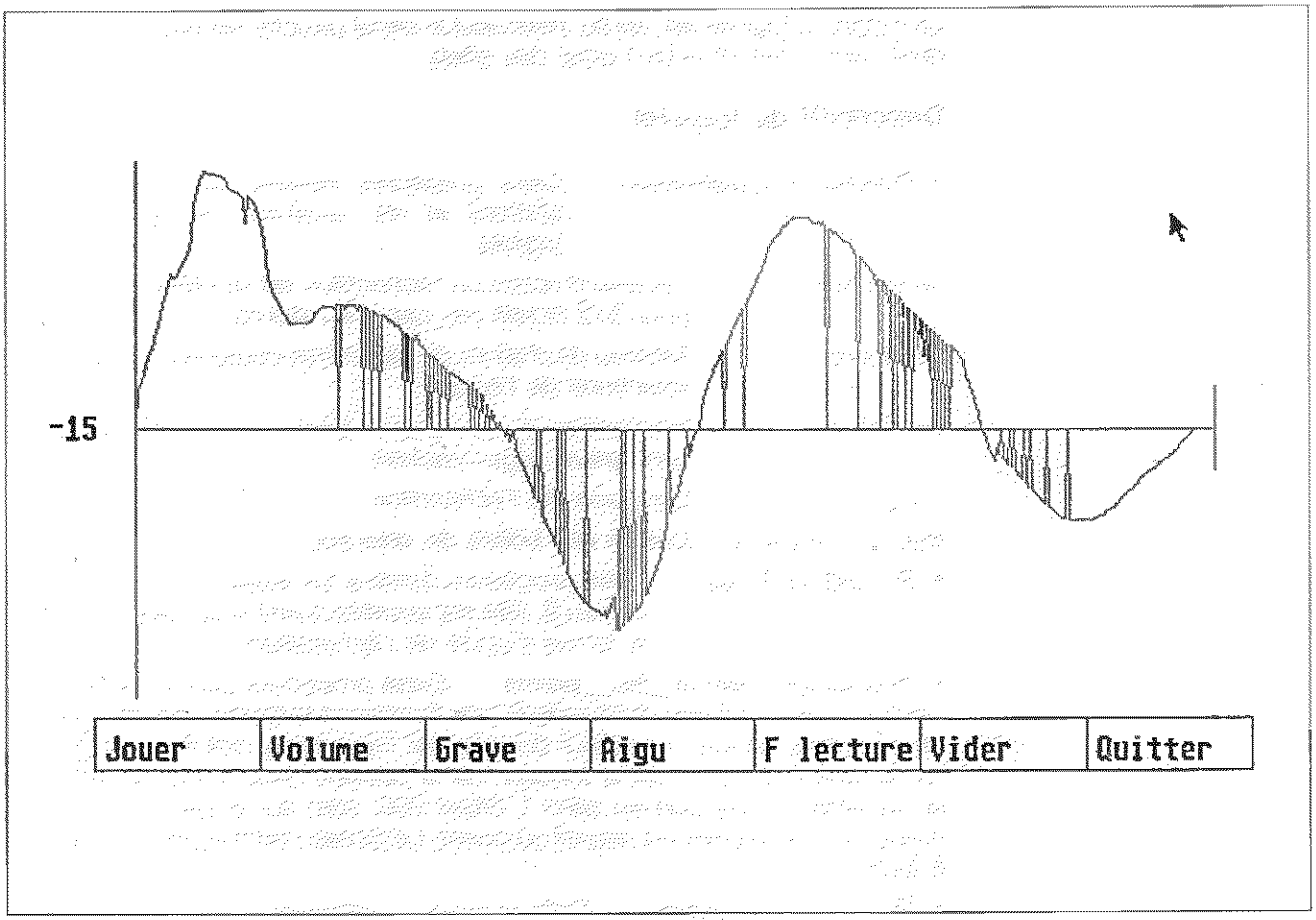
Vous désirez supprimer des aigus

```
DPOKE $FF8924,$7FF          !Masque
DPOKE $FF8922,%100100011    !Donnée
```

10	010	0011
adresse LMC	commande aigu	valeur des aigus

PROGRAMME DE DESSIN D'ÉCHANTILLONS POUR LE STE

Ce programme permet de dessiner une forme d'onde occupant au maximum 512 octets. Cette forme d'onde peut être raccourci à l'aide du click droit de la souris en déplaçant le pointeur du registre de fin d'échantillon. Le traitement de la forme d'onde peut s'effectuer en temps réel.



Utilisation du logiciel

Jouer : Joue l'échantillon en cours.

Volume : Cliquer à gauche pour baisser et à droite pour augmenter le volume de sortie.

Grave : Cliquer à gauche pour baisser et à droite pour augmenter les graves.

Aigu : Cliquer à gauche pour baisser et à droite pour augmenter les aigus.

F. Lecture: Cliquer à gauche pour baisser et à droite pour augmenter la fréquence de lecture de l'échantillon.

Vider : Remplir la chaîne avec du vide.

Quitter : Sortie du logiciel.

Pour modifier la longueur de l'échantillon, cliquer à droite dans la partie réservée au dessin, le pointeur de fin suivra la souris.

Le mode du logiciel est stéréo, mais seul le canal gauche est utilisé, le canal droit sera utilisé plus tard pour des effets.

Descriptif du logiciel

- **Procédure initialisation** Cette procédure initialise les tableaux, les chaînes et les variables utilisés dans le logiciel.

ondel(1026) : Tableau d'octets ou l'échantillon est stocké 1026 octets pour 512 octets par canal en stéréo.

option$(6) : Tableau de chaînes de caractères contenant le nom des sélections de l'écran.

Les DPOKE : Initialisation du soundchip, stéréo, soundchip et puce son, grave, aigu, volume.

Long_echan% : Longueur de l'échantillon.

Bouche FOR/NEXT: Dessin des boîtes de sélection.

- **Procédure Fond** Cette procédure dessine les limites de dessin et le niveau 0. Elle est appelée avant chaque dessin de la forme d'onde de l'échantillon.

- **Procédure dessine_les_points** Cette procédure permet de dessiner des points à l'écran et de remplir simultanément le tableau **ondel( )** avec de nouvelles valeurs. La valeur donnée par l'axe des y doit être convertie car la valeur du point est la fonction de la position de la souris et non de la symétrie par rapport au zéro. L'offset 200, zéro sur le dessin, permet d'apprécier si le point est négatif ou positif. Le tableau est rempli de la valeur à jouer.

- **Procédure redessine** Cette procédure redessine la forme d'onde de l'échantillon après effacement de l'ancien échantillon et appel de la procédure **fond**.

Un calcul inverse à la procédure **dessine_les_points** dessine l'emplacement des plots. Les variables **xx&** et **yy&** sont uniquement utilisées pour la fonction DRAW TO, elles mémorisent les avant derniers points dessinés.

- **Procédure inverse(scase)** Fonction graphique qui inverse le fond de la boîte de sélection.

- **Procédure longueur_échantillon**

Cette procédure gère le pointeur de fin de l'échantillon, c'est la variable **long_echan%** qui est modifiée.

- **Procédure vidage** Remplie le tableau ondel( ) avec des 0.

- **Procédure Jouer_onde** Cette procédure a deux fonctions, soit l'échantillon n'est pas activé, alors la procédure **jouer** est appelée, si l'échantillon jouait alors il est arrêté à l'aide du registre de contrôle du DMA.

- **Procédure jouer** Cette procédure envoie tous les pointeurs du tableau aux différents registres et active le registre de contrôle du DMA.

- **Procédures fréquence, grave, aigu et volume**

Ces procédures significatives par leur nom se passe de commentaires par leurs simplicités.



```
'
'    dessin d'une forme d'onde
'
'              ATARI STE
'
'
'
'              version
'            monochrome
'
initialisation
fond
DO
  MOUSE x,y,k
  IF k=1 AND x<562 AND x>50 AND y>72 AND y<328
    dessine_les_points
    redessine
  ENDIF
  IF k=2 AND x<562 AND x>60 AND y>72 AND y<328
    longueur_echantillon
  ENDIF
  IF k<>0 AND y<365 AND y>340
    choix=INT((x-30)/80)
    SELECT choix
    CASE 0
      jouer_onde
    CASE 1
      volume
    CASE 2
      grave
    CASE 3
      aigu
    CASE 4
      frequence
    CASE 5
      vidage
    CASE 6
      END
    ENDSELECT
  ENDIF
LOOP
PROCEDURE initialisation
  DIM ondel(1026)
  ARRAYFILL ondel(),0
  DIM option$(6)
  FOR a%=0 TO 6
    READ option$(a%)
  NEXT a%
  DATA Jouer,Volume,Grave,Aigu,F Lecture,Vider,Quitter
  SDPOKE &HFF8920,1                    ! stéréo
  SDPOKE &HFF8924,&X1111111111
  SDPOKE &HFF8922,&X1                  ! mélange sound ship et puce son
  SDPOKE &HFF8924,&X1111111111
  SDPOKE &HFF8922,&X10001000110 ! grave
  SDPOKE &HFF8924,&X1111111111
  SDPOKE &HFF8922,&X10010000110 ! aigu
  SDPOKE &HFF8924,&X1111111111
```

(suite)

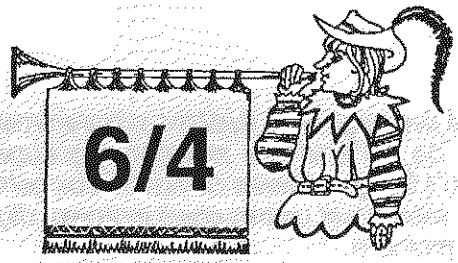
```
    SDPOKE &HFF8922,&X10011101000 ! volume
    vol=43
    long_echan%=1024
    FOR a%=30 TO 520 STEP 80
      BOX a%,340,a%+80,365
      TEXT a%+5,360,option$((a%-30)/80)
    NEXT a%
  RETURN
  PROCEDURE fond
    LINE 50,200,570,200
    LINE 50,70,50,330
    LINE 60+(long_echan%/2),180,60+long_echan%/2,220
  RETURN
  PROCEDURE dessine_les_points
    REPEAT
      PLOT x,y
      le_point=200-y
      PRINT AT(1,13);le_point"  "
      IF le_point>=0
        ondel((x-50)*2)=le_point
      ELSE
        ondel((x-50)*2)=(256 XOR le_point) AND 255
      ENDIF
      MOUSE x,y,k
    UNTIL k=0 OR x>562 OR x<50 OR y<73 OR y>327
  RETURN
  PROCEDURE redessine
    DEFFILL 0,0
    PBOX 50,70,562,330
    DEFFILL 1,0
    fond
    xx&=50
    yy&=200
    FOR a&=0 TO 512
      IF ondel(a&*2)>127
        le_point&=((ondel(a&*2) AND 127) XOR 0)-128
      ELSE
        le_point&=ondel(a&*2)
      ENDIF
      DRAW xx&,yy& TO a&+50,200-le_point&
      xx&=a&+50
      yy&=200-le_point&
    NEXT a&
  RETURN
  PROCEDURE inverse(scase)
    DEFFILL 1,1
    GRAPHMODE 3
    PBOX 30+(scase*80),340,110+(scase*80),365
    GRAPHMODE 1
    DEFFILL 1,0
  RETURN
  PROCEDURE longueur_echantillon
    s%=VARPTR(ondel(0))
    REPEAT
      MOUSE x,y,k
      a%=60+lon_echan%
```

(suite)

```
      GRAPHMODE 3
      LINE 60+long_echan%/2,180,60+long_echan%/2,220
      long_echan%=(x-60)*2
      GRAPHMODE 1
      LINE 60+long_echan%/2,180,60+long_echan%/2,220
      SDPOKE &HFF890E,((s%+long_echan%) AND &HFF0000)/&H10000
      SDPOKE &HFF8910,((s%+long_echan%) AND &HFF00)/&H100
      SDPOKE &HFF8912,(s%+long_echan%) AND &HFF
    UNTIL k=0 OR x<60 OR x>562
    redessine
  RETURN
  PROCEDURE vidage
    inverse(choix)
    ARRAYFILL onde%(),0
    redessine
    inverse(choix)
  RETURN
  PROCEDURE jouer_onde
    inverse(choix)
    IF en_route!=0
      jouer
      en_route!=1
    ELSE
      PRINT AT(4,4);"          "
      en_route!=0
      SDPOKE &HFF8900,0
    ENDIF
    REPEAT
    UNTIL MOUSEK=0
  RETURN
  PROCEDURE jouer
    s%=VARPTR(onde%(0))
    SDPOKE &HFF8902,(s% AND &HFF0000)/&H10000
    SDPOKE &HFF8904,(s% AND &HFF00)/&H100
    SDPOKE &HFF8906,s% AND &HFF
    SDPOKE &HFF890E,((s%+long_echan%) AND &HFF0000)/&H10000
    SDPOKE &HFF8910,((s%+long_echan%) AND &HFF00)/&H100
    SDPOKE &HFF8912,(s%+long_echan%) AND &HFF
    SDPOKE &HFF8900,3
  RETURN
  PROCEDURE frequence
    IF k=1
      DEC fr
      fr=MAX(fr,0)
    ELSE
      INC fr
      fr=MIN(fr,3)
    ENDIF
    SDPOKE &HFF8920,fr
    PAUSE 10
  RETURN
  PROCEDURE grave
    IF k=1
      DEC grave
      grave=MAX(grave,0)
    ELSE
```

(suite)

```
      INC grave
      grave=MIN(grave,10)
    ENDIF
    SDPOKE &HFF8924,&X11111111111
    SDPOKE &HFF8922,&X10001000000+grave
    PAUSE 10
  RETURN
  PROCEDURE aigu
    IF k=1
      DEC aigu
      aigu=MAX(aigu,0)
    ELSE
      INC aigu
      aigu=MIN(aigu,10)
    ENDIF
    SDPOKE &HFF8924,&X11111111111
    SDPOKE &HFF8922,&X10010000000+aigu
    PAUSE 10
  RETURN
  PROCEDURE volume
    IF k=1
      DEC vol
      vol=MAX(vol,0)
    ELSE
      INC vol
      vol=MIN(vol,47)
    ENDIF
    SDPOKE &HFF8924,&X11111111111
    SDPOKE &HFF8922,&X10011000000+vol
    PAUSE 3
  RETURN
```



6/4 Les grands logiciels musicaux

6/4.1 FM Melody Maker STE Melody Maker

UN SYNTHÉTISEUR DANS UNE CARTOUCHE

Les modestes performances de l'Atari STF dans le domaine de la restitution sonore ont incité la société Richard Watts à réaliser une cartouche contenant un générateur de sons polyphonique basé sur le principe de la synthèse à modulation de fréquence que l'on trouve sur certains synthétiseurs professionnels.

Une fois connecté au port cartouche de l'Atari et à un amplificateur extérieur par l'intermédiaire des deux fiches CINCH qui l'équipent, ce petit boîtier distribué par Upgrade permet de produire de la musique avec une excellente qualité de restitution sans avoir à investir dans un système complexe nécessitant clavier, expandeur, logiciel séquenceur...

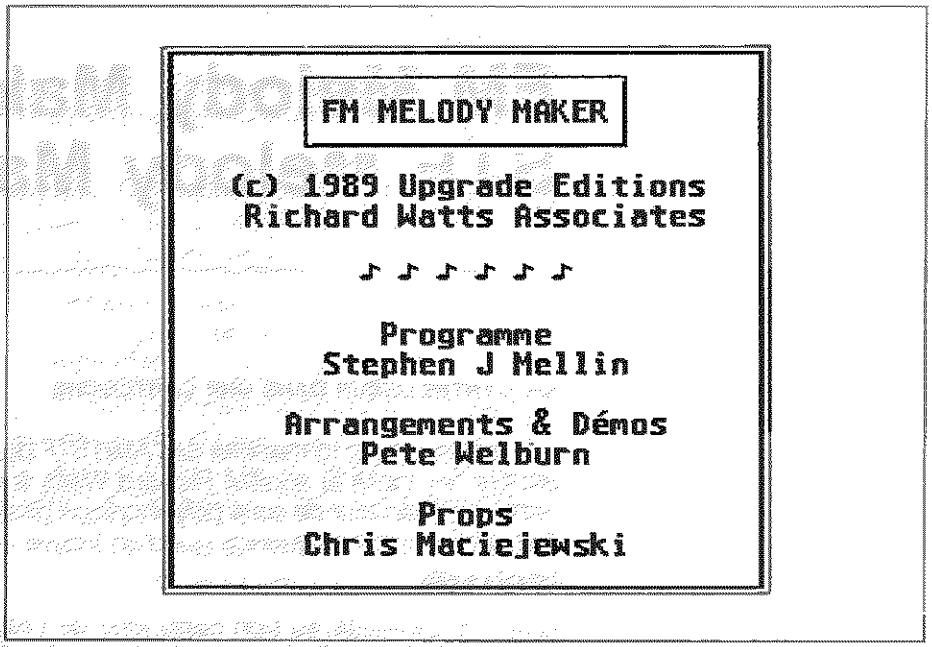
Pour le développeur, la programmation s'effectue en déposant successivement, par l'intermédiaire de routines en assembleur, les différents paramètres de sélection des notes et des instruments à l'intérieur des registres de la cartouche, ce qui peut sembler une gageure puisque le port cartouche de l'Atari ne peut être que lu et est normalement inaccessible en écriture (voir Annexes).

Le principal avantage d'un système programmable tient au fait que le code permettant de générer une musique n'occupe que quelques octets en mémoire qu'il faut par ailleurs comparer aux centaines de kilo-octets nécessaires pour mémoriser un très court morceau digitalisé restitué avec un réalisme bien moindre.

On pourrait ainsi imaginer dans un avenir proche des logiciels sur STF, compatibles FM Melody Maker, sonorisés en continu avec une qualité équivalente à celle que l'on connaît sur les récentes consoles de jeux 16 bits. Upgrade incite les développeurs à travailler dans ce sens en leur fournissant gratuitement sur demande le kit de développement constitué d'une notice en anglais et d'une disquette de routines en assembleur avec leurs sources ainsi qu'un exemple de programme en C.

AVEC OU SANS CARTOUCHE ?

Afin de rendre sa cartouche immédiatement utilisable par le plus grand nombre et d'en assurer la promotion, Richard Watts la fournit avec un logiciel de création musicale, appelé également FM Melody Maker.



Ce programme est tellement réussi qu'il l'a même repris fin 90, à quelques détails près pour l'Atari STE sous le nom de STE Melody Maker. Cette dernière version se passe de cartouche puisque le STE dispose lui d'un circuit spécialisé dans la production du son ainsi que d'une sortie audio pour chaîne HI-FI. La version STF de FM Melody Maker et sa cartouche fonctionnent par ailleurs parfaitement sur un STE, alors que STE Melody Maker est actuellement un des rares programmes conçus spécialement pour tirer parti des ressources de cette version de l'Atari à laquelle il est exclusivement destiné. Dans les deux cas ces logiciels se révèlent être des outils de composition musicale particulièrement rapides et efficaces puisque l'utilisateur n'a besoin

que de transcrire la mélodie et les accords : le programme se charge de réaliser au mieux l'arrangement mélodique et rythmique avec plusieurs voix de cuivres, de bois, de cordes, ..., une basse, une batterie, ceci en fonction de l'accompagnement choisi parmi les 16 styles disponibles (Pop, Disco, Rock, Afro...). Le résultat final est tout à fait convainquant et même impressionnant de clarté et de réalisme comme le montrent les exemples de compositions fournis sur la disquette du logiciel.

La retranscription d'une partition et même la réalisation de petites compositions se trouvent ainsi mises à la portée d'un débutant qui, connaissant quelques règles d'harmonisation, n'a pas besoin d'être un instrumentiste virtuose pour « jouer » de son Atari.

Un outil complet d'initiation musicale

Les deux logiciels d'Upgrade ne se limitent pas à la composition mélodique et possèdent d'autres atouts qui en font pour le débutant d'intéressants outils d'initiation aux techniques de programmation de boîtes à rythme ou d'utilisation d'instruments MIDI. On peut ainsi :

— envoyer une mélodie qu'on a composée vers un instrument ou un expandeur MIDI externe ;

— si on possède un clavier MIDI, transformer l'Atari en expandeur et profiter de tous les sons synthétisés de la cartouche (uniquement avec FM Melody maker) ;

— disposer d'un enregistreur MIDI une piste pour s'initier au principe de fonctionnement des séquenceurs ; la séquence mémorisée ne peut malheureusement pas être retravaillée depuis l'éditeur de musique ;

— composer des morceaux rythmiques élaborés à l'aide d'une boîte à rythmes programmable, sans avoir malheureusement la possibilité de les utiliser avec une composition mélodique créée par ailleurs.

Les deux restrictions annoncées pour ces derniers points sont d'ailleurs les principaux reproches que l'on peut formuler : les différents modules (compositeurs, enregistreur et boîte à rythmes) sont indépendants les uns des autres et n'offrent pas de possibilité de transfert des données de l'un à l'autre.

Caractéristiques techniques

La cartouche de FM Melody Maker peut générer 9 voix simultanées ou 6 voix plus 5 percussions. Chaque voix est elle-même choisie parmi 15 sons prédéfinis. On dispose également d'une voix entièrement programmable (en assembleur), comme sur un véritable synthétiseur : c'est ainsi que le logiciel de composition musicale propose en réalité 78 instruments différents pour la mélodie alors qu'il se limite aux 15 instruments pré-définis pour l'accompagnement. Les possibilités de combinaisons d'instruments pour un arrangement sont donc considérables.

Le logiciel STE Melody Maker est forcément limité en nombre de voix disponibles puisque tout doit être chargé dans la mémoire de l'ordinateur. On y trouve tout de même 12 instruments différents dont certains, à notre avis,

sont même plus réalistes que ceux de la cartouche (saxo), et 11 sons de percussions, soit 6 de plus que dans FM Melody Maker. Les voix simultanées sont cependant réduites à 5 et l'utilisateur ne peut pas intervenir sur l'accompagnement.

Le fonctionnement des deux logiciels étant très semblable, nous les décrirons simultanément en signalant au fur et à mesure les détails particuliers propres à chacun.

Branchements

Pour utiliser FM Melody Maker, il faut obligatoirement relier la cartouche qui comporte deux prises CINCH à une entrée ligne (AUX, TUNER, CD, etc.) d'un amplificateur extérieur.

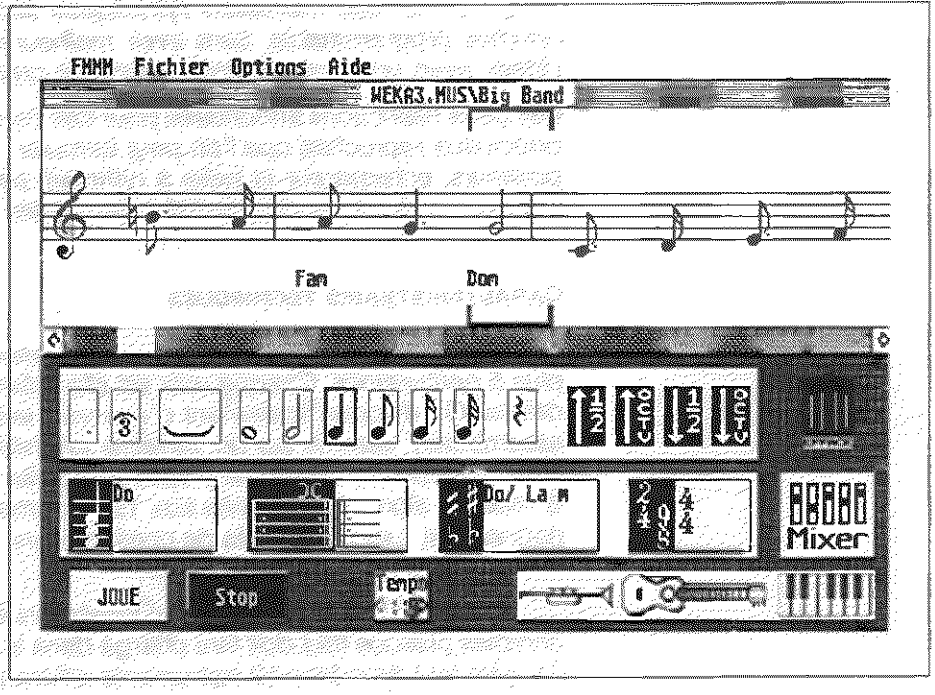
Sans que cela soit obligatoire, on a intérêt à faire de même en reliant la sortie audio de l'Atari STE lorsqu'on utilise STE Melody Maker car le haut parleur du moniteur est généralement de piètre qualité et ne permet pas de bénéficier de l'effet stéréo.

Si on dispose d'un clavier ou d'un orgue MIDI on peut le connecter aux prises MIDI de l'Atari à l'aide de deux cordons DIN 5 broches : MIDI IN de l'Atari dans MIDI OUT du clavier et réciproquement.

Le lancement du logiciel s'effectue par un clic sur MAKFRNC.PRG (FMMM) ou sur STEMELMK.PRG (STEMM). Attention, avec ce dernier sur 520 STE la totalité de la mémoire est nécessaire et il est préférable de ne charger aucun accessoire de bureau.

Le compositeur de musique

L'écran qui est proposé au chargement du logiciel permet de saisir immédiatement une musique depuis une partition ou de réaliser une composition



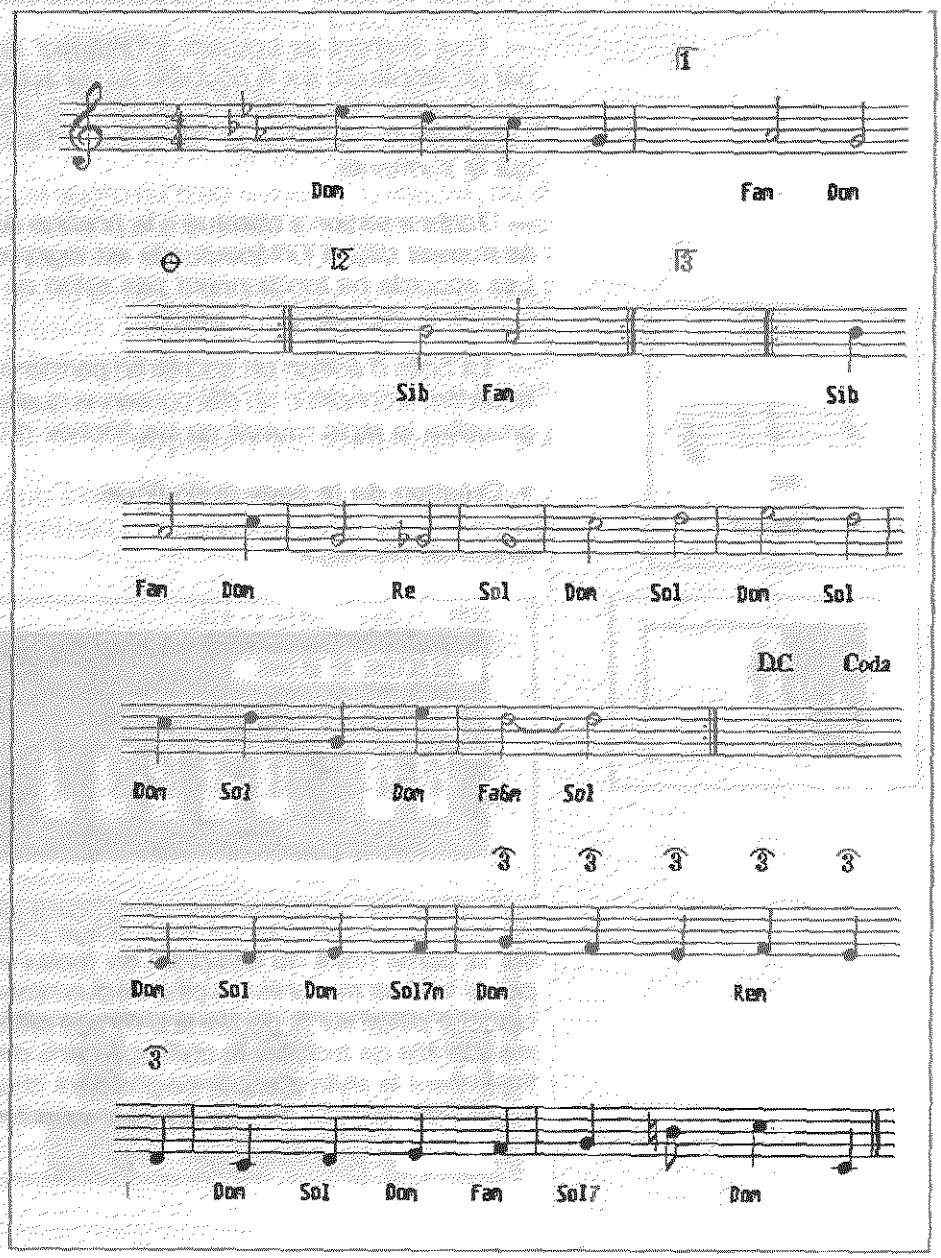
originale. Cet écran est divisé en deux parties de tailles égales : une boîte à outils surmontée d'une fenêtre de saisie sur laquelle est tracée une portée en clé de sol.

Avec FM Melody Maker, une musique est constituée :

— d'une ligne mélodique qu'il faut créer note à note sur la portée affichée à l'écran ;

— d'un accompagnement mélodique qu'on ajoute sous forme d'accords sous la portée ;

— et d'un accompagnement rythmique.



• **Effacement d'une musique**

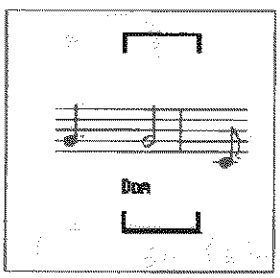
Le menu **OPTIONS/EFFACE NOTES** permet de disposer d'une page de travail vierge, prête à recevoir une nouvelle composition. Nous vous proposons, comme support de travail, cette composition originale d'une quinzaine de mesures qui reprend la plupart des notions qui seront abordées au cours des explications : chiffrage, armure, durées et hauteurs de notes, triolets, altérations, accords, signes de répétition, etc.

• **La boîte d'édition**

L'éditeur de saisie encadre la portée de deux larges crochets horizontaux :

— Tout élément se trouvant à l'intérieur des crochets peut être supprimé par un simple clic sur la poubelle ou par appuis sur **[DELETE]** : si deux objets sont pointés simultanément par l'éditeur de saisie (une note et un accord), l'effacement s'effectue en deux fois et l'accord est supprimé avant la note qui le surmonte.

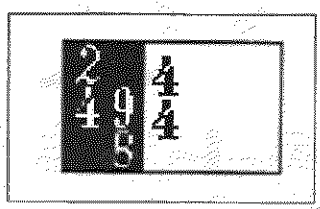
— Toute insertion s'effectue à la position de l'éditeur de saisie, si une note se trouvait déjà à l'intérieur, elle est logiquement repoussée vers la droite. Les accords ne fonctionnent pas selon ce principe et tout nouvel accord écrase le précédent s'il existe.

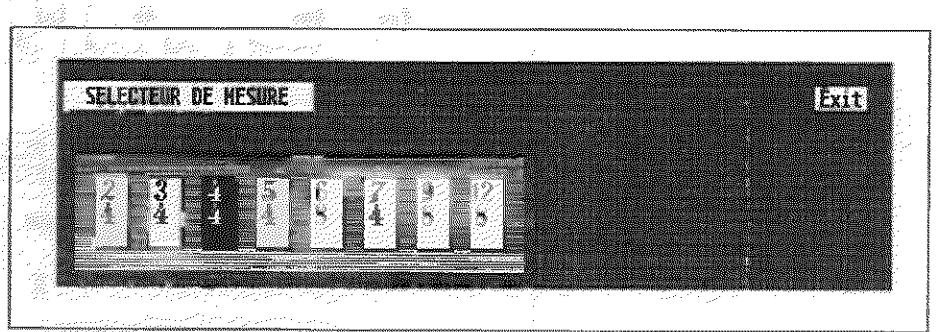


— La boîte d'édition est ancrée en permanence au centre de l'écran et c'est la portée, à condition qu'elle contienne quelque chose, qui se déplace quand on utilise le tapis roulant ou les flèches droite et gauche du clavier.

• **Création de la ligne mélodique**

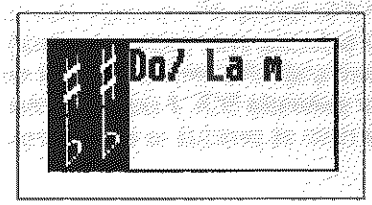
Choix du chiffrage

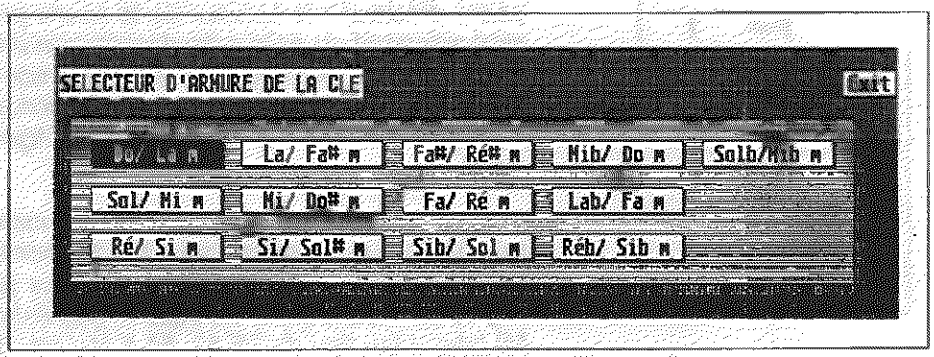




Au chargement du logiciel, le chiffrage proposé par défaut est **4/4**. Un clic sur la partie noire du sélecteur de mesures offre une possibilité de choix parmi les huit mesures les plus fréquemment utilisées ; un clic sur la partie blanche place sur la portée le chiffrage affiché dans la boîte. Il est possible, mais inutile de modifier le chiffrage dans une même partition puisque c'est finalement le style d'accompagnement choisi qui impose son propre chiffrage (voir « Choix de l'accompagnement »).

Choix de l'armure



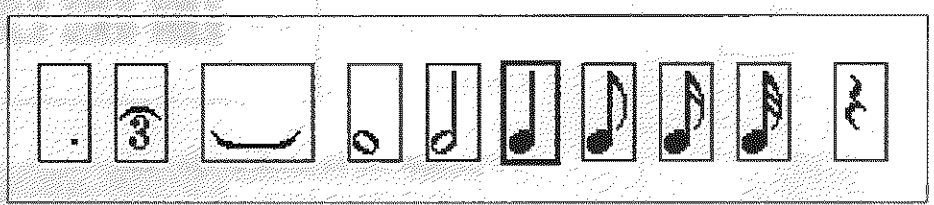


Le sélecteur d'armure de la clé fonctionne sur le même principe et propose treize armures différentes avec leurs équivalents majeur/mineur.

Dans notre exemple, l'armure choisie est MI bémol (majeur) qui place les trois bémols à la clé.

Choix de la valeur de note

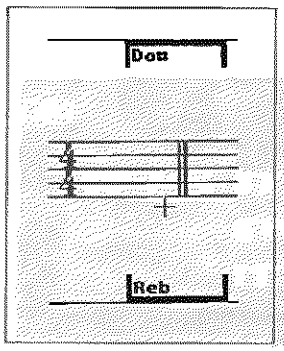
Le sélecteur de notes affiche les valeurs de notes de la ronde à la triple croche.



Un clic sur une valeur de note fait passer son cadre en gras ; cette valeur peut être modifiée par l'activation du point qui allonge sa durée de la moitié ou par celle du triolet qui diminue sa valeur d'un tiers (trois croches « triolées » sont jouées dans la durée de deux croches) : l'activation avec la souris doit se faire très rapidement car ces cases fonctionnent sur le principe d'une bascule qui passe alternativement d'un état à l'autre, ce qui ne manque pas de se produire si on maintient trop longtemps le clic.

Attention : contrairement à la notation habituelle qui fait qu'une seule note surmontée du symbole de triolet est jouée en réalité comme trois notes dans la durée de deux, il faut ici placer soi-même les trois notes du triolet successivement en les surmontant chacune du symbole 3.

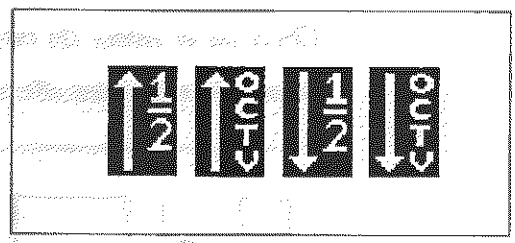
La liaison provoque un effet de glissement entre plusieurs notes de niveaux différents ou fait entendre sans interruption plusieurs notes de même hauteur (son tenu) ce qui permet éventuellement de changer d'accord comme dans la quatrième ligne de notre partition.



Choix de la hauteur de note

Sa valeur étant activée, on peut poser la note sur la portée à l'intérieur du cadre d'édition : le curseur de la souris prend la forme d'une croix (écran monochrome) ou de la note dont la valeur est sélectionnée (écran couleur). Au gré des déplacements de la souris, son nom s'affiche simultanément en toutes lettres au niveau du crochet d'édition, par exemple SOL # ou LA bémol. Lorsque ce dernier est correct, un clic fige la note et produit le son correspondant.

Il est regrettable qu'on ne puisse pas choisir plus précisément le type d'altération puisque c'est l'ordinateur qui l'impose : ainsi, et bien que le son produit soit le même, on obtient toujours sur la portée un DO # même là où les règles de théorie de la musique imposeraient un RE bémol ; de même pour MI bémol (au lieu de RE #), FA # (au lieu de sol bémol), SOL # (au lieu de LA bémol), SI bémol (au lieu de LA #). Il est vrai que le codage MIDI qui est choisi ici ne s'embarrasse pas de ces détails. Deux octaves peuvent être affichées directement sur la portée (de DO4 à DO6 dans le système MIDI) ; les autres notes peuvent être placées sur la portée puis amenées à la bonne hauteur grâce aux icônes de translation d'octave vers le haut ou vers le bas : la mélodie peut donc s'étendre sur quatre octaves maximum de DO3 à DO7 puisqu'on ne peut travailler qu'en clé de Sol...

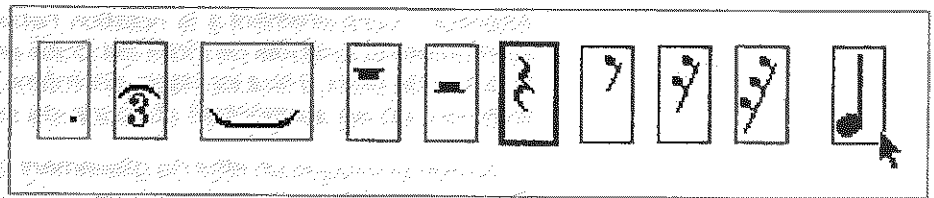


Correction de la hauteur de note

La poubelle permet de supprimer la note incorrecte qui se trouve dans la boîte d'édition, il est cependant possible de corriger sa hauteur en la faisant glisser demi-ton par demi-ton vers le bas ou vers le haut.

Choix des silences

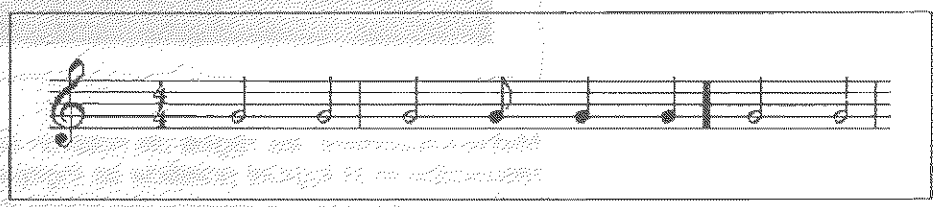
Lorsqu'on active la case de bascule qui représente un silence, on obtient à la place des valeurs de notes les différentes valeurs correspondantes de silences, de la pause au triple soupir.



Ils peuvent être sélectionnés et placés sur la portée de la même façon que les notes. On retrouve les valeurs de notes dans le menu en cliquant à nouveau sur la case de bascule.

Barres de mesure

Le logiciel trace automatiquement les barres de mesure au fur et à mesure que les notes sont placées. Il peut arriver que le compte ne soit pas bon : si, dans une mesure à quatre temps, on place successivement une blanche, une croche, une noire (soit trois temps et demi), puis à nouveau une noire (un temps), la barre de mesure apparaît alors sous la forme d'une triple barre verticale, indiquant que la mesure précédente est incorrecte.

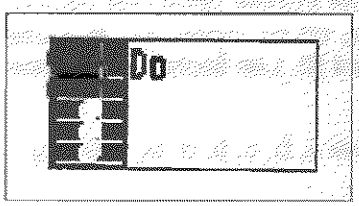


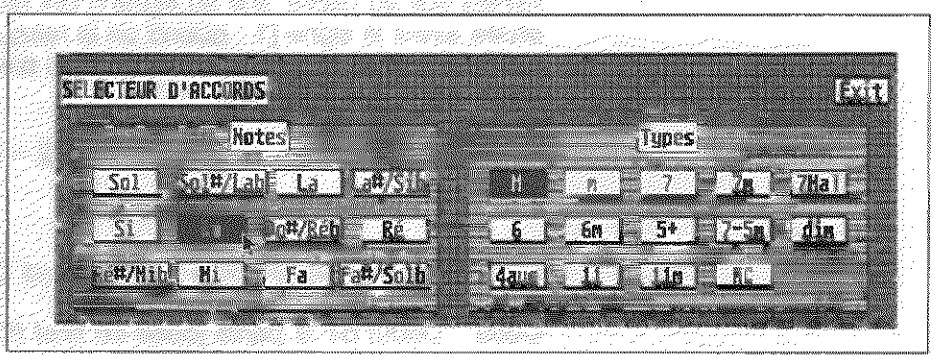
Une mesure incorrecte.

Cela n'empêche pas FM Melody maker de jouer l'air et de compenser l'erreur pour recadrer son accompagnement au début de la mesure suivante.

- **Positionnement des accords d'accompagnement**

On place l'éditeur de saisie sous la note ou le silence à partir duquel l'accord se fera entendre. Un clic sur la partie noire de la case de sélection des accords présente la boîte de sélection qui propose dans sa partie droite la fondamentale de l'accord et dans sa partie gauche le type d'accord.

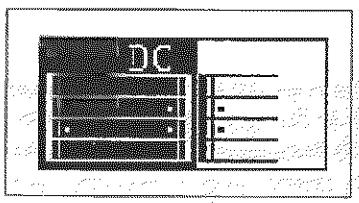


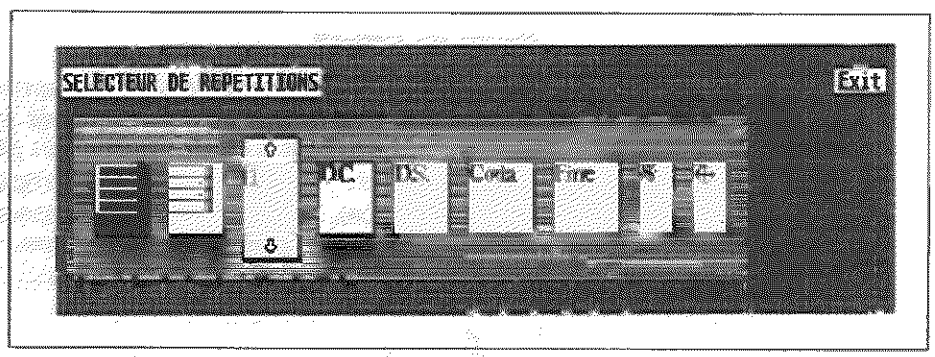


Un clic dans la partie blanche de la case de sélection place l'accord sous la portée. En cas d'erreur, on peut facilement supprimer un accord en l'amenant (ainsi que la note ou le silence qui le surmonte) dans l'éditeur de saisie puis en cliquant sur la poubelle — la note, elle, n'est pas supprimée. Dans la boîte de sélection, **NC** annule l'effet du dernier accord. Sur un écran couleur, faute de place sur la portée, les accords apparaissent sous leur notation anglo-saxonne c'est-à-dire **A** pour LA, **B** pour **SI**, **C** pour **DO**, **D** pour **RE**, **E** pour **MI**, **F** pour FA, **G** pour SOL. Ainsi **Solm7** apparaît sous la forme **G7m**.

- **Programmation des répétitions**

La case de sélection des répétitions propose, lorsqu'on clique sur sa partie noire, les différents symboles en usage dans les partitions.





Malheureusement, les règles de notation classiques ne sont pas toujours respectées et le logiciel possède sa logique propre qui apparaît dans les quelques exemples ci-dessous (les lettres minuscules a, b, c ... figurent les mesures à répéter) :

[: Début de la partie à répéter.

:] Fin de la partie à répéter.

Exemple : **a [: b :] c** joue successivement les parties a, b, b et c. Le b placé entre les signes de répétition est joué deux fois puis on passe à la suite (respect de la notation classique).

[1 Après avoir joué le bloc délimité par ce signe et par le signe **..]**, on revient soit au début, soit au signe **[:** qui doit se trouver dans une mesure située avant le signe **[1** ; quand on la retrouve, on ne répète pas la partie délimitée par **[1** et **:]** et on saute au bloc suivant **[2**. Les flèches de la case d'activation font accéder à 2, 3, 4 etc.

Exemple : **a [: b [1 c :] [2 d :] [3 e :] [4 f g** ... joue a, b, c, b, d, b, e, b, **f**, g...

D.S. Da Segno : retour au signe **$**.

$ Segno : signe utilisé par le renvoi **Da Segno**.

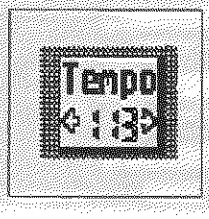
Coda Début de la partie jouée en final (Coda = la queue).

@ A la **Coda**, renvoie à la partie finale mais uniquement après une répétition.

Exemple : **a $ b @ c D.S. CODA d** joue a, b, c, b, d. Lorsqu'on joue a, b, c les signes **$** et **@** ne sont pas pris en compte puisqu'il n'y a pas encore eu de répétition ; après c, **D.S** (Da Segno) renvoie au signe **$**, on joue alors à nouveau b, le signe **@** est cette fois-ci pris en compte puisqu'on est dans une répétition et on saute à la **CODA** pour terminer par d.

Fine Fin de la **CODA**, si celle-ci, du fait des répétitions, se trouve à l'intérieur du morceau.

D.C. Da Capo : Retour au début du morceau (Caput = la tête).



Exemple : a @ b CODA *c* FINE *d* D.C. joue a, b, c, d, a, c. Lorsqu'on joue pour la première fois, il n'y a pas encore de répétition et a, b, c, d s'enchaînent : après d, le signe D.C. (Da Capo) renvoie au début du morceau, on rejoue a puis le signe @ renvoie à la CODA qui se termine à FINE, on finit donc par c.

- **Choix du tempo**

Les deux flèches de la case d'ajustage permettent de le régler précisément dans une fourchette allant de 40 à 240 temps par minute.

- **Jouer l'air**

La case **JOUE** ou un appui sur **[ESPACE]** lance l'exécution du morceau qui peut être interrompu par la case **STOP** ou par appui sur **[RETURN]**. Au cours de l'exécution, la partition défile à l'écran. Il n'est malheureusement pas possible de faire démarrer l'exécution à une mesure autre que la mesure 1 : aussi, lors d'une composition, à chaque nouvel ajout ou correction, il est nécessaire de réécouter tout le début du morceau.

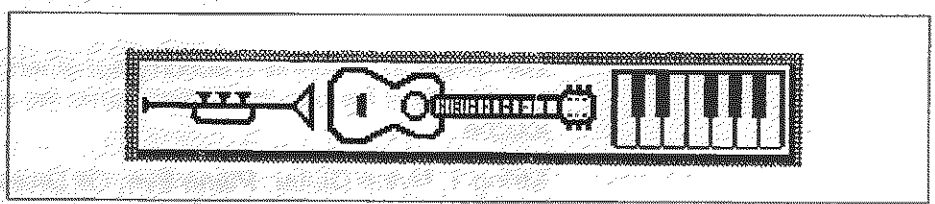
Pour réaliser les opérations qui vont suivre et qui peuvent s'effectuer en temps réel, il est préférable que la musique soit en cours de déroulement (au moyen d'une boucle sans fin par exemple), ce qui facilite les réglages.

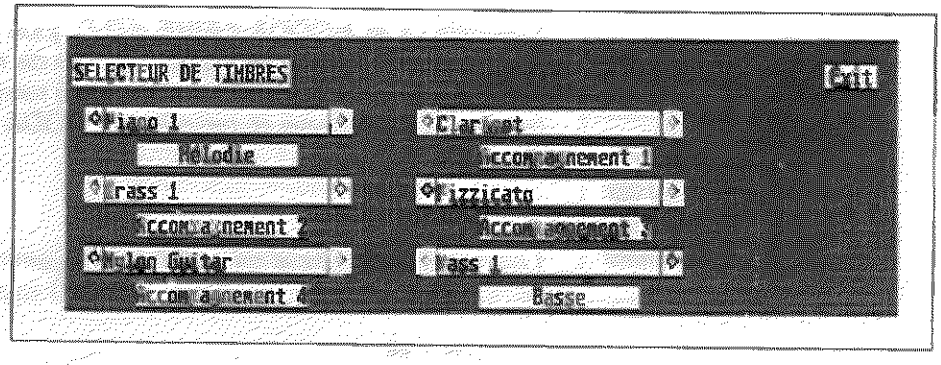
- **Choix des instruments**

Sur STE Melody Maker, le choix est limité : il ne concerne que le son de la mélodie qui doit être choisi en cliquant sur les flèches droite ou gauche de la boîte de sélection en bas de l'écran à droite. On dispose des douze sons suivants :

Grand Piano, Elec. Piano, Marimba, Xylophone, Harp, Elec. Organ, Choir, Clarinet, Oboe, Trumpet, Saxophone, Synth. Bass.

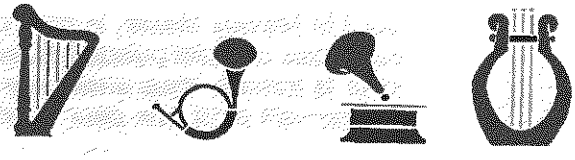
Sur FM Melody Maker, la case de sélection fait accéder à un menu beaucoup plus complet puisqu'il autorise non seulement le choix de l'instrument pour la mélodie mais aussi pour chacun des quatre instruments d'accompagnement ainsi que pour la voix de basse.





Les sons disponibles sont également plus nombreux du fait de l'absence de limitation mémoire due à la cartouche additionnelle. On dispose ici de 78 sons différents pour la mélodie (15 sons de base plus 53 sons programmés) :

Cimes, Waves, Bell, Way Out, Meaty Job, Alien, Synth Harpsi, Synth Lead, Synth Bass 2, Synth Bass 1, Synth 3, Synth 2, Tubular Bells, Steel Drums 2, Koto 2, Koto 1, Glock 2, Glock 1, Vibes 2, Sitar, Jaw Harp 2, Jaw Harp 1, Harp, Lute, Slap bass, Rasp bass, Walking bass, Elec Bass 2, Bass 2, Distort Guitar, Blues Guitar, Guitar, Accordion, Toy Piano, Clavichord, Harpsichord 3, Harpsichord 2, Harpshicord 1, Reed Organ, Organ, Elec Piano 3, Elec Piano 2, Elec Piano 1, Piano 2, Piano 1, Harmonica, Saxophone, Jazz Flute, Bassoon, Oboe, Flute, Piccolo, Bagpipes, Cello, Violin, Synth Strings, Strings 2, Tuba, Trombone, Trumpet, Bell Brass, Wah bass, Brass 2, Banjo, Electric Bass 1, Bass 1, Vibes 1, Pizzicato, Synth 1, French Horn, Blues Harp, Brass 1, Marimba, Clarinet, Oil Drum, Piano, Syn, Nylon Guitar, String 1.

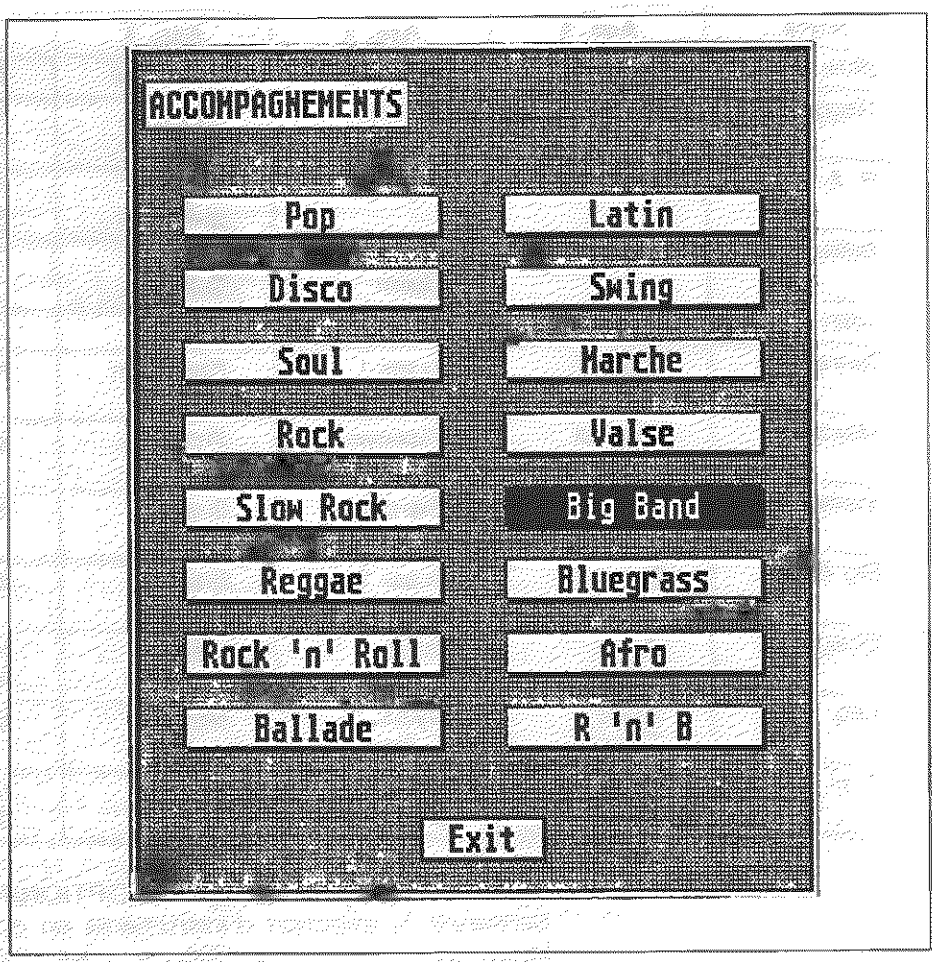


Les quatre instruments d'accompagnement et la basse sont choisis indépendamment les uns des autres parmi les quinze sons de base de la cartouche :

String 1, Nylon Guitar, Piano Syn, Oil Drum, Clarinet, Marimba, Brass 1, Blues Harp, French Horn, Synth 1, Pizzicato, Vibes 1, Bass 1, Electric Bass 1, Banjo.

- Choix de l'accompagnement

Le menu OPTIONS/ACCOMPAGNEMENT propose seize styles différents qui dans tous les cas imposent leur rythmique au morceau et génèrent automatiquement un accompagnement en fonction des accords placés.



Le logiciel s'arrange toujours pour que l'accompagnement « colle » à la mélodie quitte à ignorer les barres de mesure. Il est néanmoins plus satisfaisant pour l'esprit d'adapter dès le départ le chiffrage à l'accompagnement souhaité : mise à part la valse qui demande du 3/4, tous les autres styles se satisfont du 4/4 ou même du 2/4 pour Soul, Rock et Swing. A chaque style, correspond un choix d'instruments différents pour l'accompagnement ; voici ceux proposés au départ et que l'on peut modifier sur FM Melody Maker uniquement :

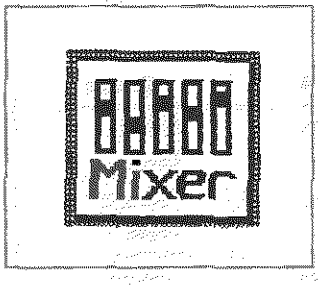
Style	Acc. 1	Acc. 2	Acc. 3	Acc. 4	Basse
Pop	Nylon Guitar	Nylon Guitar	Nylon Guitar	Pizzicato	Electric Bass 1
Disco	Vibes 1	Marimba	Marimba	Pizzicato	Electric Bass 1
Soul	Pizzicato	Pizzicato	Piano Syn	Piano Syn	Electric Bass 1
Rock	Brass 1	Piano Syn	Synth 1	Synth 1	Bass 1

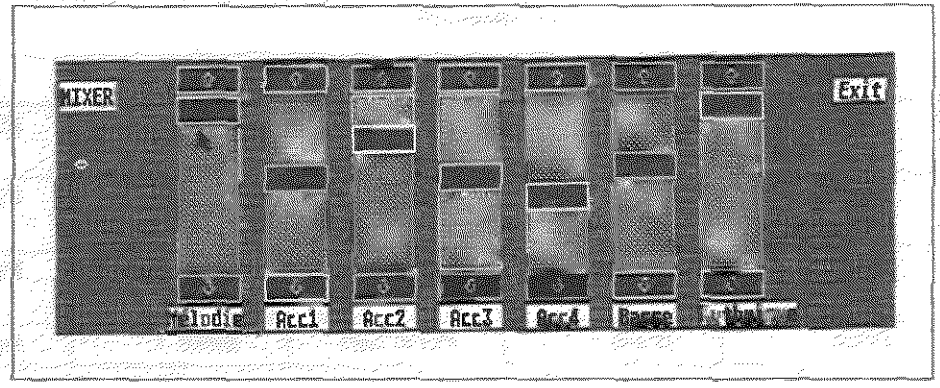
Style	Acc. 1	Acc. 2	Acc. 3	Acc. 4	Basse
Slow R.	Vibes 1	Piano Syn	Piano Syn	Piano Syn	Bass 1
Reggae	Pizzicato	Pizzicato	Nylon Guitar	Nylon Guitar	Bass 1
R. & R.	Brass 1	Brass 1	Nylon guitar	Nylon guitar	Bass 1
Ballade	Vibes 1	Nylon Guitar	Nylon Guitar	Oil Drum	Electric Bass 1
Latin	Pizzicato	Pizzicato	Oil Drum	Oil Drum	Synth 1
Swing	Brass 1	Brass 1	Piano Syn	Piano Syn	Electric Bass 1
Marche	Clarinet	Pizzicato	Pizzicato	Brass 1	Electric Bass 1
Valse	Piano Syn	Piano Syn	Brass 1	Brass 1	Electric Bass 1
Big Band	Brass 1	Brass 1	Brass 1	Brass 1	Electric Bass 1
Bluegr.	Banjo	Banjo	Nylon Guitar	Nylon Guitar	Synth 1
Afro	Nylon Guitar	Nylon Guitar	Marimba	Marimba	Electric Bass 1
R 'n' B	Nylon Guitar	Nylon Guitar	Nylon Guitar	Nylon Guitar	Electric Bass 1

Avec FM Melody Maker, le menu OPTION/PAR DEFAUT OFF permet de conserver la sélection d'instruments en cours pour l'accompagnement lorsqu'on choisit un style différent ; c'est utile lorsqu'on a réalisé sa propre sélection et que l'on veut tester différents styles en conservant celle-ci ; sans cela la sélection d'instruments associée au nouveau style remplacerait la sélection actuelle.

- Contrôle des voix

Le MIXER règle indépendamment et précisément chacune des voix afin d'obtenir un équilibre sonore agréable.





STE Melody Maker ne donne accès à ce niveau qu'à deux canaux d'accompagnement au lieu de quatre. On peut profiter de ce menu pour mettre à zéro certains canaux gênants d'accompagnement ou de rythmique et ne conserver par exemple que la mélodie.

Attaque d'un expandeur

Le morceau créé avec FM Melody Maker ou STE Melody Maker peut être joué par un expandeur ou tout instrument électronique équipé de prises MIDI, en utilisant donc les sons propres à ce dernier.

Il faut d'abord activer les sorties MIDI depuis le menu **OPTIONS/SORTIES MIDI** puis mettre en correspondance les voix et les canaux MIDI.

Les numéros des hauteurs de notes lorsqu'ils apparaissent dans les menus correspondent aux codes MIDI, c'est-à-dire **60** pour **DO** (octave 4), **61** pour **DO #**, **62** pour **RE**, etc. (voir Annexes).

Il faut évidemment savoir auparavant si l'instrument utilisé, un synthétiseur par exemple, fonctionne :

— en mode **OMNI** : tout ce qui arrive de l'ordinateur est joué, quel que soit le numéro du canal choisi. La programmation est des plus simples puisqu'on peut choisir n'importe quel canal en sortie, mais on ne dispose alors que d'un seul son d'instrument pour la mélodie et les voix d'accompagnement ce qui risque de produire un effet quelque peu confus.

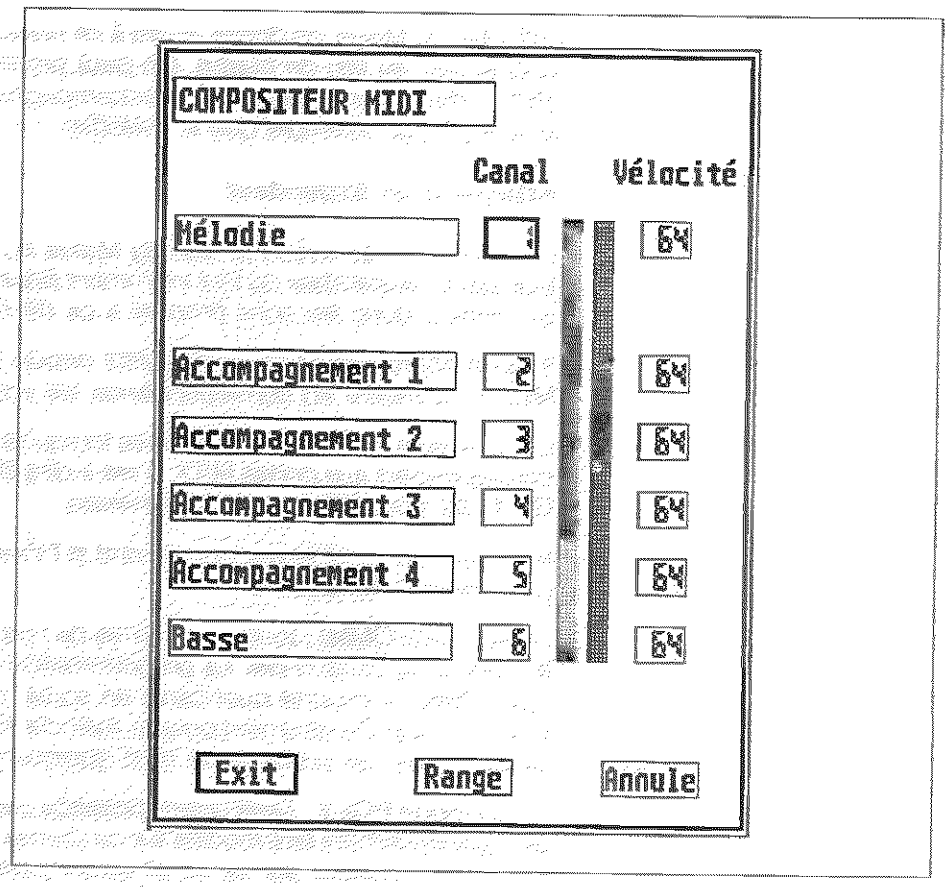
— en mode **POLY** : l'instrument possède son propre numéro de canal, qui peut généralement être modifié sur ce dernier, et seules les informations arrivant de l'ordinateur sur ce canal seront jouées. Sur **FMMM** on peut brancher de la sorte six instruments différents (quatre sur **STEMM**) qui joueront chacun leur partie (voir le branchement de la prise **MIDI THRU** en Partie 6 Chapitre 2.2). Il faut ici connaître ou régler en conséquence le numéro de canal de chaque instrument afin de pouvoir l'associer à la partie instrumentale correspondante de **FMMM** ou de **STEMM**.

— en mode **MONO** ou **MULTI** : à ce stade très élaboré, l'instrument simule une formation orchestrale complète puisqu'il possède plusieurs voix associées chacune à un canal MIDI, une seule voix par canal en mode **MONO** et même un ensemble polyphonique par canal en mode **MULTI**.

Le menu **OPTION/COMPOSITEUR MD** donne accès à une boîte de réglage qui permet d'affecter chacune des parties instrumentales à un des seize canaux MIDI correspondants.

Pour modifier le numéro d'un canal, il faut d'abord l'activer en cliquant dessus puis le modifier en cliquant sur l'une des deux grandes barres verticales d'incrémentation ou de décrémentation.

Le réglage de la vélocité s'effectue de la même manière, à condition que l'instrument utilisé y soit sensible : la vélocité correspond par exemple à l'intensité avec laquelle la touche d'un clavier est frappée ; l'intensité sonore et le timbre de la note s'en trouvent évidemment modifiés. En faisant varier la vélocité de certaines parties d'accompagnement on peut modifier l'équilibre général sans recourir au **MIXER**. Avant de quitter cet écran par **EXIT**,



il faut cliquer sur RANGE pour que les valeurs choisies soient effectivement affectées aux canaux MIDI. ANNULE rétablirait les valeurs originelles.

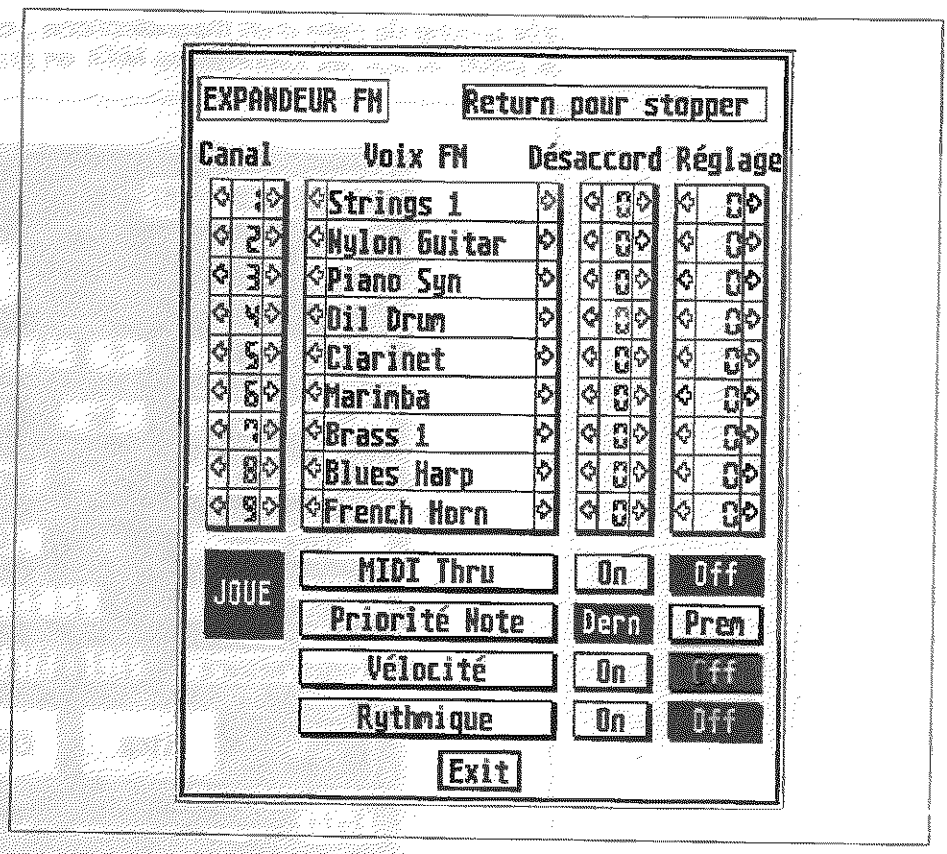
FM Melody Maker utilisé comme expandeur

Attention, l'expandeur n'existe pas dans STE Melody Maker. Avec cette option, il suffit de disposer d'un instrument équipé d'une prise MIDI OUT (un simple orgue de salon par exemple) pour pouvoir jouer directement au clavier en bénéficiant de tous les sons de FM Melody Maker.

Le menu OPTION/EXPANDEUR FM ouvre une boîte de réglage qui met en correspondance un maximum de 9 canaux MIDI avec 9 voix FM. Un même canal peut se voir attribuer plusieurs voix : par exemple trois canaux seront programmés avec trois voix chacun. Le premier canal programmé (quel que soit son numéro) permet d'accéder aux 78 sons des mélodies, les huit autres canaux sont limités aux 15 sons d'accompagnement.

Chaque voix peut être transposée (REGLAGE) de plus ou moins deux octaves maximum ou être désaccordée (DESACCORD).

MIDI THRU renvoie en sortie vers un autre expandeur les données reçues en entrée sur l'Atari.



Priorité note : l'expandeur fonctionne comme un synthétiseur polyphonique mais est limité à la réception de 9 notes simultanément (soit une note par canal si on a programmé 9 canaux, 3 notes si on n'en a programmé que 3, etc.). Si à un moment donné on demande plus de 9 notes, cette option donne la priorité soit aux premières touches enfoncées et ignore les suivantes, soit aux dernières qui annulent alors l'effet des premières.

Vélocité peut être mis à **ON** si le clavier est sensible à la force avec laquelle les touches sont enfoncées.

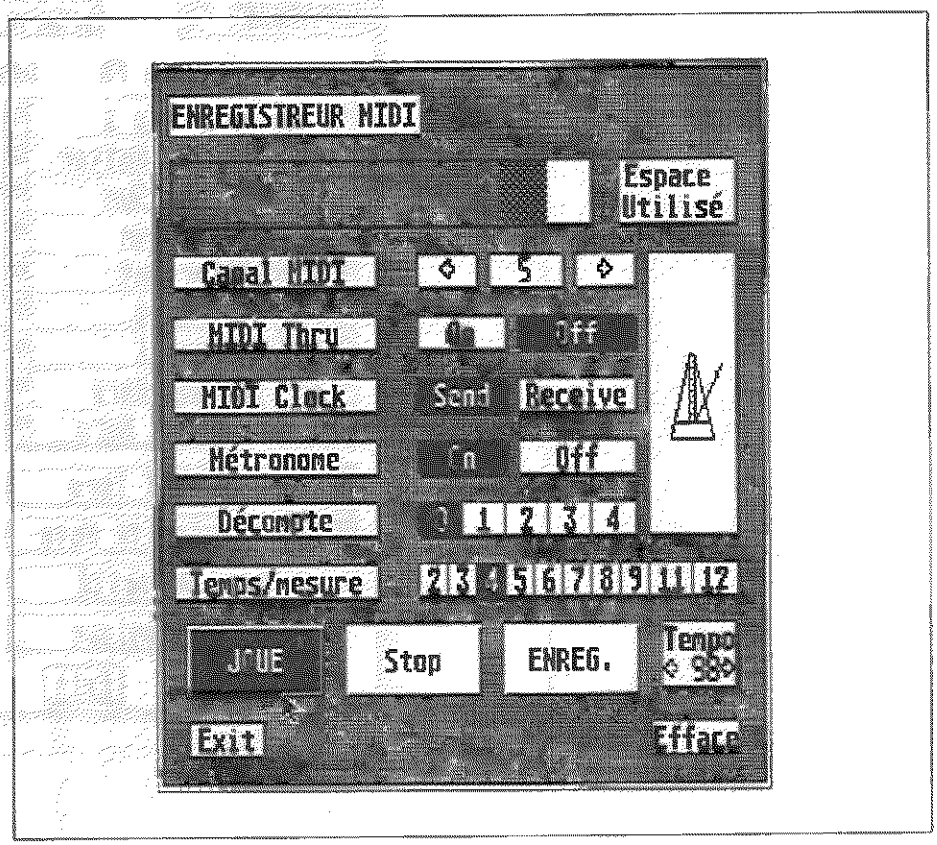
Rythmique permet d'utiliser en plus des 78 sons de base les cinq voix (Pied, Caisse, Hihat, Cymbale et Toms) de la sortie **OPTION/RYTHME MIDI**. Cela réduit par conséquent le nombre de canaux accessibles aux voix FM de l'Atari.

Attention : contrairement à ce qu'on pourrait penser, **RYTHMIQUE** ne permet pas de réutiliser un rythme qu'on aurait programmé depuis la boîte à rythme de FM Melody Maker.

Enregistreur MIDI

Ce module aux performances modestes (il ne peut travailler que sur un canal), montre bien le fonctionnement d'un enregistreur MIDI dont l'utilisation est

très proche de celle d'un magnétophone ; ici, cependant on mémorise à la place du son les informations MIDI en provenance du clavier.



Il faut d'abord sélectionner le canal MIDI sur lequel l'instrument envoie ses informations sur l'entrée **MIDI IN** de l'Atari.

Si le son de l'instrument est généré non par le clavier mais par un expandeur branché sur la prise **OUT** de l'ordinateur, il faut mettre à **ON** l'option **MIDI THRU** afin de pouvoir entendre ce que l'on joue.

MIDI CLOCK génère le tempo soit par l'horloge interne (**Send**) de l'Atari, soit par l'horloge de l'instrument (**Receive**).

Le **DECOMPTE** permet de ne commencer l'enregistrement qu'après une, deux, trois ou quatre mesures à vide, ou immédiatement après l'activation de **ENREG.** (les mesures à vide sont de toute façon enregistrées !).

TEMPS/MESURE demande sur quel temps de la mesure le métronome doit produire un son plus fort.

ENREG. débute l'enregistrement : tout ce qui est reçu sur l'entrée MIDI est stocké en mémoire. Une barre croissante indique quelle est la place occupée en mémoire. Cette barre croît évidemment plus vite si des informations de vélocité ou des modulations sont envoyées par le clavier car l'information est alors beaucoup plus abondante pour chaque note entrée.

JOUE retourne à l'instrument branché sur la prise MIDI OUT de l'Atari les informations telles qu'elles ont été reçues au moment de l'enregistrement.

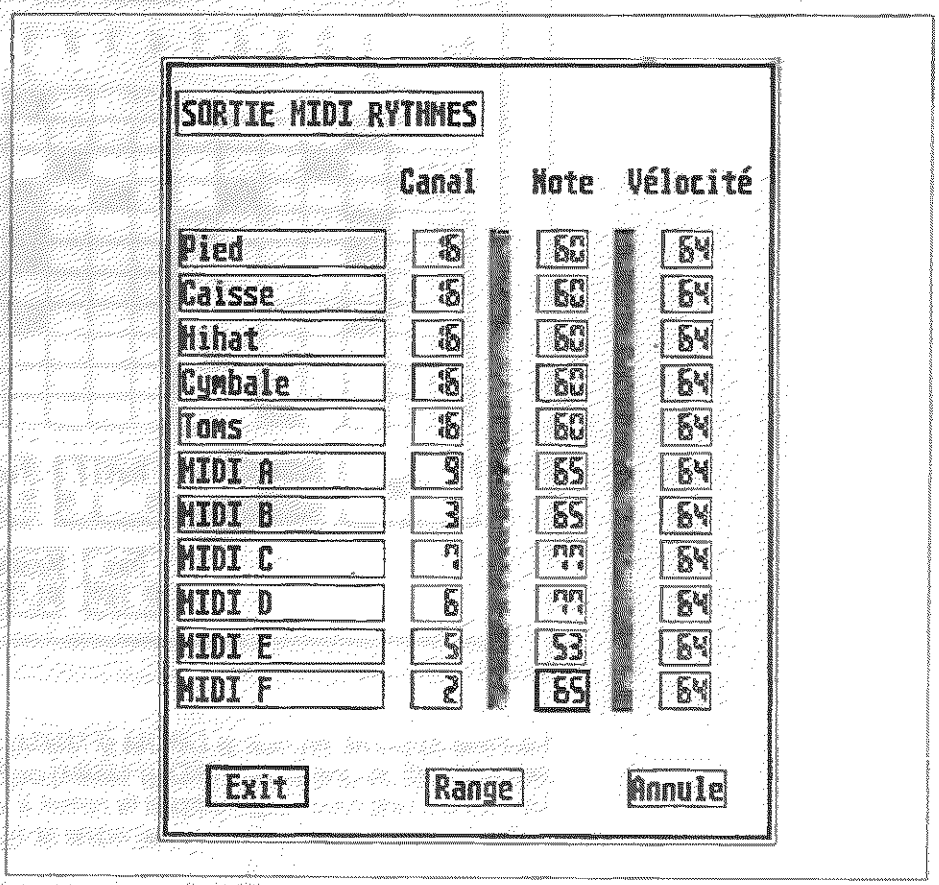
TEMPO modifie la vitesse de lecture de l'enregistrement réalisé.

La boîte à rythme

Les possibilités de la boîte à rythme de FM Melody Maker diffèrent sensiblement de celles de STE Melody Maker bien que le principe de fonctionnement reste le même.

Avec FM Melody Maker, on ne dispose que de cinq percussions (Pied, Caisse, Hihat, Cymbale, Toms) qui peuvent éventuellement être redirigées vers l'extérieur si on sélectionne MIDI à la place de DRUMS : dans ce cas, aux cinq canaux de percussions s'ajoutent six autres canaux MIDI.

Pour utiliser la sortie MIDI, il faut mettre OPTIONS/SORTIE MIDI sur ON et ajuster les paramètres de la boîte de saisie : OPTION/RYTHMES MD.



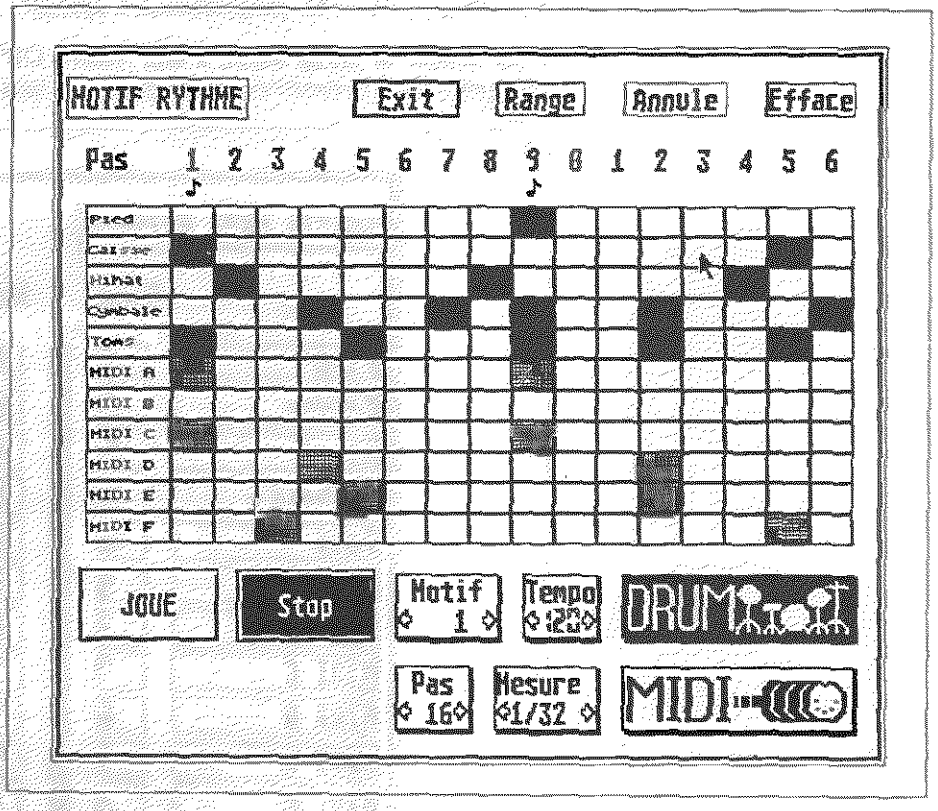
Dans STE Melody Maker, les sorties MIDI ont été abandonnées et on dispose d'un ensemble plus riche de 11 percussions (Bass, Snare, Closed Hihat, Open Hihat, High Tom, Mid Tom, Low Tom, Hand Clap, Rim Click, Cowbell, Claves). On bénéficie de surcroît d'un spectaculaire effet stéréo. Mal-

heureusement, contrairement à FM Melody Maker, on ne peut jouer que de deux instruments en même temps, un sur le canal gauche, un sur le canal droit.

Pour créer un morceau rythmique, il faut d'abord définir un à un les différents motifs le composant, puis les assembler dans l'ordre souhaité. On peut créer jusqu'à 20 motifs différents puis les réunir pour constituer un morceau de 40 motifs maximum.

- *Créer un motif*

Depuis le menu **OPTIONS/MOTIF RYTHME**, il suffit d'activer la case **JOUE** puis de cliquer sur les cases de la grille aux temps où l'instrument doit être joué.



La case apparaît en noir si **DRUM** (FMMM) ou **LEFT** (STEMM) est activé, elle apparaît en gris si c'est **MIDI** (FMMM) ou **RIGHT** (STEMM) qui est choisi. Un clic sur une case déjà remplie la remet à vide. Avec **STEMM**, rappelons qu'on ne peut avoir qu'une case noire et qu'une case grise par colonne.

Un motif peut être constitué de un à seize pas de programme. Si on sélectionne un nombre de pas inférieur à 16, il est bien sûr inutile de remplir les dernières cases qui seront ignorées.

Le début de la mesure se trouve repéré à l'écran par une note qui surmonte la grille de travail ; elle peut varier suivant les trois valeurs suivantes :

— 1/8, deux pas par mesure et huit mesures dans la grille ;

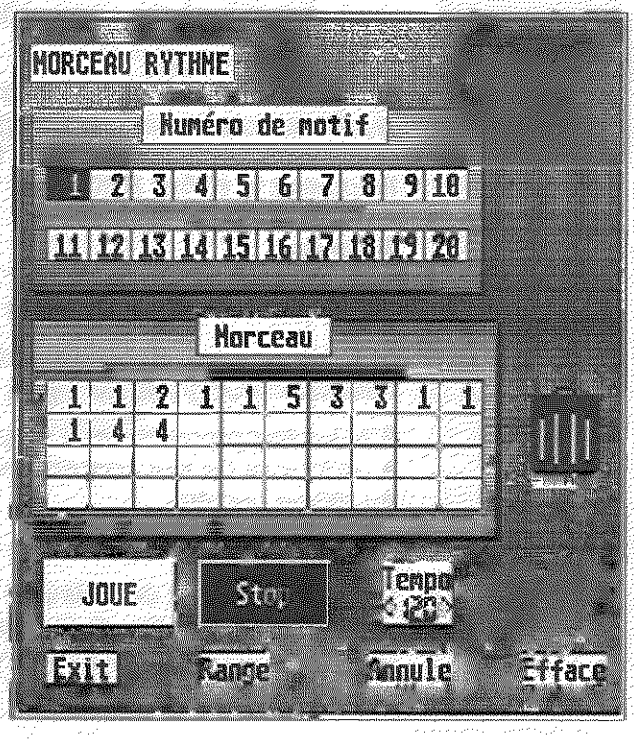
— 1/16, quatre pas par mesure et quatre mesures ;

— 1/32, huit pas par mesure et deux mesures.

Lorsque le premier motif paraît satisfaisant, il faut le mettre en mémoire avec **RANGE** puis incrémenter le numéro en cliquant sur la flèche droite de la case **MOTIF**. Le motif 2 et les suivants peuvent alors être programmés selon la même procédure. **EFFACE** remet à blanc toutes les cases du tableau alors qu'**ANNULE** supprime toutes les modifications effectuées depuis le dernier rangement.

• **Lier les motifs**

Depuis le menu **OPTIONS/MORCEAU RYTHME**, on place dans le tableau **MORCEAU** les numéros des motifs dans l'ordre où on souhaite les entendre. Pour cela, on clique d'abord sur le numéro du motif puis sur la case du tableau **MORCEAU**. On ne doit pas laisser de case vide dans le tableau **MORCEAU** lorsqu'on place les motifs. On peut par contre insérer un motif en cliquant sur une case déjà remplie.



On peut effacer un motif mal placé à l'aide de la corbeille : on clique une fois sur celle-ci afin d'inverser sa couleur, puis on clique sur tous les motifs à supprimer ; un nouveau clic sur la corbeille annule cette fonction.

EFFACE supprime tous les motifs déjà placés alors qu'**ANNULE** affiche

le dernier motif rangé en supprimant toutes les modifications effectuées.

Si on souhaite sauvegarder le morceau sur disquette, il faut d'abord utiliser **RANGE** pour le stocker en mémoire avant de sortir de cette boîte.

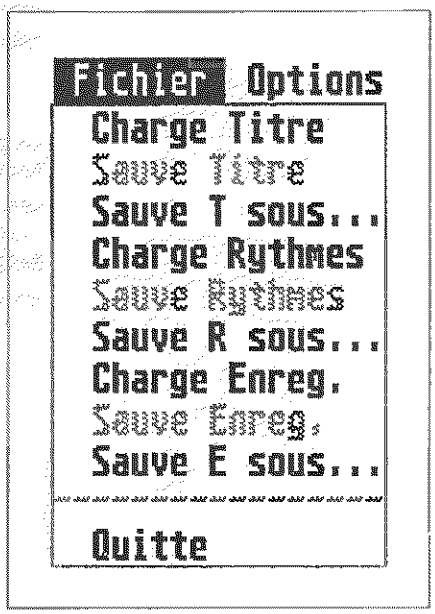
Archivage

Le menu **FICHIER** propose les différentes options de sauvegarde et de chargement des fichiers.

TITRE concerne les airs de musique entrés sur la partition. Ils portent le suffixe **MUS**. Les fichiers FM Melody Maker et STE Melody Maker ont le même format, ce qui permet les échanges.

RYTHMES archive les rythmes programmés dans la grille. Ils portent le suffixe **DRM**. Les fichiers générés pour la cartouche ne sont pas compatibles avec STE Melody Maker et inversement du fait des importantes différences concernant ces modules.

ENREG., sur FM Melody Maker uniquement, conserve les enregistrements réalisés sur l'enregistreur MIDI. Ils portent le suffixe **REC**.



Annexes

- **Support technique Upgrade**

Les possesseurs dûment répertoriés de FM Melody Maker disposent de l'assistance Upgrade au 16 (1) 43 44 78 88.

S'ils veulent programmer la cartouche pour intégrer le code de commande à un logiciel de leur conception, ils peuvent demander le kit développeur au même numéro.

- **Codage des notes MIDI**

Ce tableau permet de mettre en relation les notes avec les codes MIDI utilisés par certains écrans du logiciel. C'est à l'aide de ce codage que les notes sont représentées en mémoire et dans le fichier d'archivage.

Code		Note	Octave
48	30H	DO	3
49	31H	DO#/REb	3
50	32H	RE	3
51	33H	RE#/MIb	3
...			
60	3CH	DO	4
...			
72	48H	DO	5
...			
84	54H	DO	6
...			
96	60H	DO	7

- **Format des fichiers MUS**

La connaissance du format des fichiers contenant les mélodies et les accords permet à l'utilisateur de les réutiliser en les appelant à l'intérieur d'un programme de sa conception.

Les 95 premiers octets contiennent des données système destinées au logiciel.

A partir de l'octet 96 (60H) on trouve les données, par groupe de deux (même si le deuxième octet est parfois sans signification), concernant le chiffrage, la valeur des notes et leur durée, les accords et les répétitions.

Le décodage peut se faire manuellement ou par n'importe quel logiciel sachant que :

Code	Premier octet	Deuxième octet
00H	Silence	Longueur
0CH	à 68H code MIDI de la note	Longueur
80H	Accord	Code de l'accord
81H	Début de répétition	Sans signification
82H	Fin de répétition	Sans signification
83H	Répétition numérotée	Numéro de la répétition
84H	Da Capo	Sans signification
85H	Dal Segno	Sans signification
86H	CODA	Sans signification
87H	FINE	Sans signification
88H	Segno	Sans signification
89H	à la CODA	Sans signification
90H	Armure	Numéro de l'armure
91H	Chiffrage	Numéro du chiffrage

Les longueurs de notes (ou de silences correspondants) sont codées ainsi :

00H	Ronde
01H	Blanche
02H	Noire
03H	Croche
04H	Demi-croche
05H	Triple-croche

Au code de longueur on ajoute :

06H	Pour pointer la note
0CH	Pour un triolet

Si le bit 6 de l'octet est mis à un, la note est liée. Un **dump** du fichier de la partition donnée en exemple au début de cette partie nous permet de constater que 266 octets suffisent pour la coder intégralement (dont les 95 premiers sont pris par l'en-tête).

```
00000000:  00 36 00 55 09 09 09 09-09 09 09 09 09 09 3C  .6.U..........<
00000010:  3D 3E 3F 40 41 42 43 44-45 46 40 40 40 40 40  =>?@ABCDEF@@@@@
00000020:  40 40 40 40 40 00 00 00-00 00 00 00 00 40 01 02  @@@@@........@..
00000030:  03 04 05 40 40 40 40 40-00 71 00 02 00 00 00 0F  ...@@@@@.q......
00000040:  00 00 00 00 00 00 00 0E-00 0F 00 0F 00 05 00 02  ................
00000050:  00 02 00 02 00 00 00 0E-00 10 00 10 00 00 00 00  ................
00000060:  90 09 91 02 80 25 4B 02-4A 02 48 02 43 02 83 01  .....%K.J.H.C...
00000070:  80 2A 44 01 80 25 43 01-89 00 82 00 83 02 80 13  .*D..%C.........
00000080:  46 01 80 2A 44 01 82 00-83 03 81 00 80 13 46 02  F..*D.........F.
00000090:  80 2A 44 01 80 25 48 02-43 01 80 17 42 01 80 10  .*D..%H.C...B...
000000A0:  43 00 80 25 48 01 80 10-4A 01 80 25 4B 01 80 10  C..%H...J..%K...
000000B0:  4A 01 80 25 48 02 80 10-4A 02 43 02 80 25 4B 02  J..%H...J.C..%K.
000000C0:  80 7A 4A 41 80 10 4A 01-82 00 84 00 86 00 80 25  .zJA..J........%
000000D0:  3C 02 80 10 3E 02 80 25-3F 02 80 40 41 02 80 25  <...>..%?..@A..%
000000E0:  43 0E 41 0E 3F 0E 80 27-41 0E 3F 0E 3E 0E 80 25  C.A.?..'A.?.>..%
000000F0:  3C 02 80 10 3E 92 80 25-3F 02 80 2A 41 02 80 30  <...>..%?..*A..0
00000100:  43 02 47 03 80 25 48 00-3C 02 FF 00              C.G..%H.<...
```

Voici, en se référant aux tableaux ci-dessus, ce que donne l'analyse de la première mesure à partir de 60H :

90 l'armure
09 dont le code correspond à MIb
91 le chiffrage
02 dont le code correspond à 4/4
80 le premier accord
25 dont le code correspond à DOm
4B la note de rang 75 (4BH) dans le codage MIDI (MI de l'octave 5)
02 qui est une noire
4A la seconde note de rang 74 (RE)
02 qui est aussi une noire
48 troisième note de rang 72 (DO)
02 noire
43 quatrième note de rang 67 (SOL)
02 noire

La deuxième mesure commence en 6EH :

```
83      répétition
01          de numéro 1
80      nouvel accord
2A          correspondant à FAm
44      note de rang 68 (LA)
01          blanche
80      nouvel accord
25          correspondant à DOm
43      note de rang 67 (SOL)
01          blanche
89      à la CODA
00          octet sans signification
82      fin de répétition
00          octet sans signification
```

Observons aussi C2H et C3H (mesure contenant la liaison) :

```
4A      note de rang 74 (RE)
41          blanche liée : 01H (blanche) + 40H (bit 6 à 1)
```

Ainsi que E0H et E1H (début de la mesure contenant les triolets) :

```
43      note de rang 67 (SOL)
0E          noire « triolée » : 02H (noire) + 0CH (triolet)
```

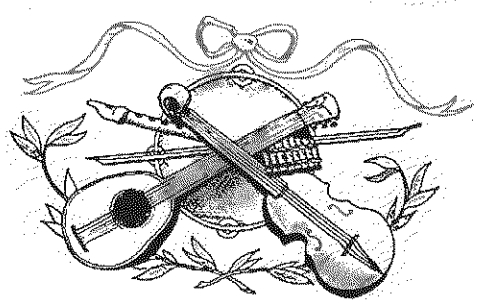
Et enfin 106H et 107H (dernière mesure contenant une noire pointée) :

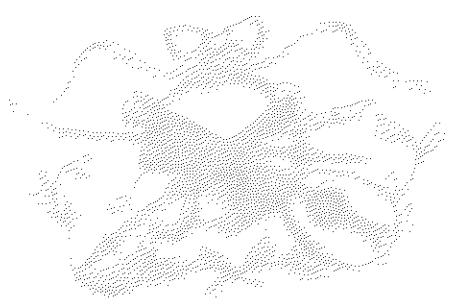
```
48      note de rang 72 (DO)
08          noire pointée : 02H (noire) + 06H (point)
```

- **Programmation en sortie du port cartouche**

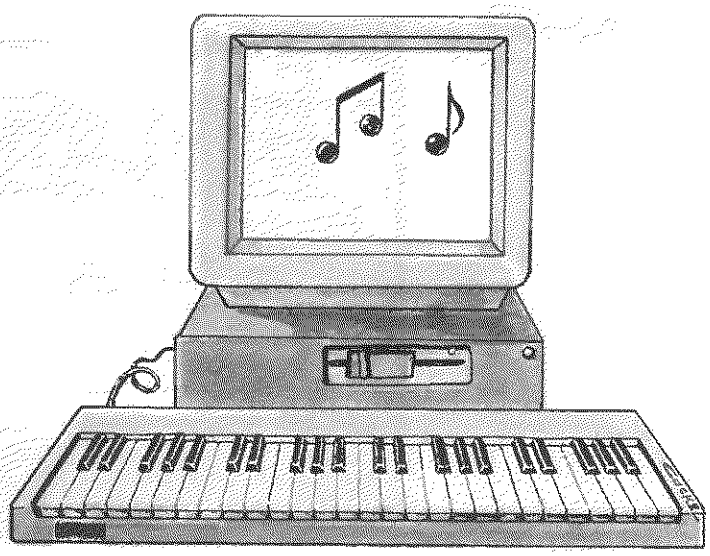
Il peut sembler étonnant que l'on puisse envoyer des informations en sortie sur le port cartouche de l'Atari alors que ce dernier est conçu pour être uniquement utilisé en lecture. Le principe est simple et est basé sur l'alternance adresse/donnée : les adresses des registres de la cartouche s'étendent de **$FB0000** à **$FB0038** ; l'opération se déroule en deux temps : on sélectionne d'abord le registre dans lequel on veut écrire, simplement en le lisant. On envoie dans un deuxième temps la valeur à déposer dans le registre en lisant l'adresse **$FB0000** à laquelle on a ajouté, par convention, deux fois la valeur à déposer ; cela a pour objet de mettre à 0 et à 1 certains des bits sur le bus d'adresse qui est lu alors comme une donnée par la cartouche.

La mise en pratique est un peu plus complexe puisqu'il faut respecter un certain délai exprimé en cycles d'horloge entre l'activation de chaque adresse et de chaque donnée, ce qui impose le recours à l'assembleur.





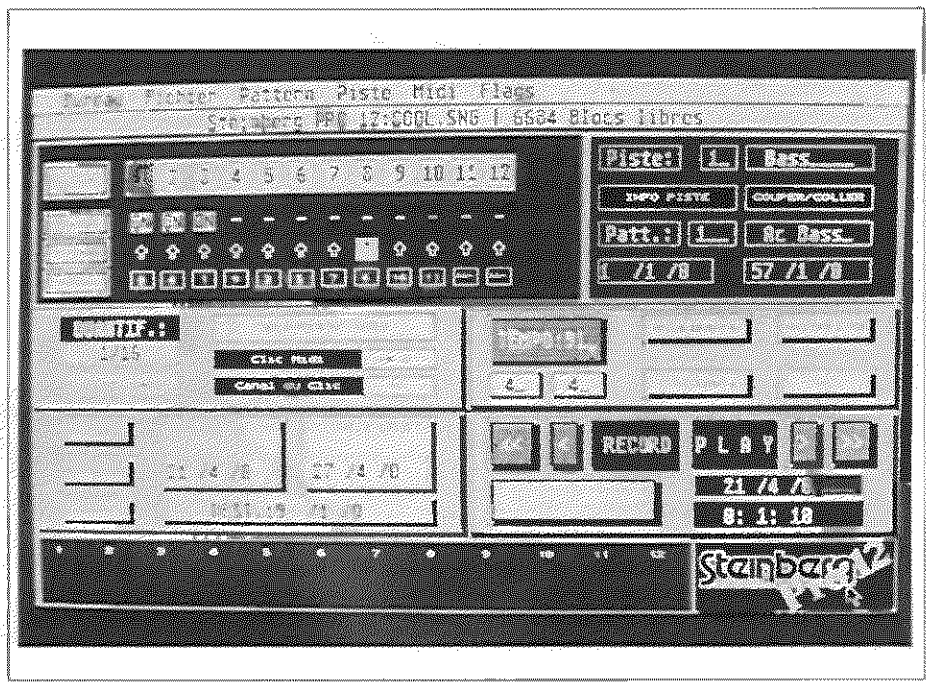
6/4.2



PRO12 de Steinberg

PRO12 est un logiciel séquenceur et enregistreur MIDI dont le fonctionnement s'apparente à celui d'un magnétophone à bande multipistes.

Contrairement à FM Melody Maker qui inclut une cartouche de sons synthétisés, le logiciel et l'ordinateur utilisés seuls ne produisent aucune note audible s'ils ne sont reliés à un ou plusieurs générateurs sonores externes tels que synthétiseurs, expandeur, boîte à rythme, piano numérique, orgue de salon ou tout autre instrument électronique équipé de prises MIDI.



PRO24, LA POINTURE SUPÉRIEURE

PRO12 est un dérivé du logiciel professionnel PRO24. Il ne dispose plus que de douze pistes, d'où son nom, et il abandonne un certain nombre de fonctions dont les principales sont le **Grid-Edit** (éditions des notes en mode MIDI sur une grille), le **Drum Edit** (grille d'édition des percussions, mais on peut tout de même piloter une boîte à rythme), le **Logical EDIT** (opérations logiques et mathématiques sur la musique), la télécommande (accès aux fonctions du logiciel depuis un instrument MIDI), le multi-recording (enregistrement de plusieurs pistes simultanément) et le mix-down (mélange de plusieurs pistes sur une seule). Rassurez-vous, même débarrassé de tout cela PRO12 se révèle suffisamment complexe d'emploi si on souhaite le pousser aux limites de ses possibilités, d'autant plus que sa notice, qui privilégie une approche par l'exemple, omet d'approfondir certaines fonctions.

Tout enregistrement réalisé sur PRO12 peut être récupéré par PRO24 car le format d'enregistrement et les fonctions de base des deux logiciels sont strictement semblables : l'utilisateur de PRO12 peut utiliser immédiatement les mêmes commandes dans PRO24 sur lequel il retrouvera à quelques infimes détails près la même ergonomie.

PRO12 ET LA MIDI

L'intérêt de stocker des données MIDI plutôt que des sons sur plusieurs pistes comme sur un magnétophone est multiple :

– faible encombrement des données stockées : le code de chaque note plus quelques paramètres occupe considérablement moins de place qu'un son digitalisé par exemple ;
– souplesse des modifications qui sont permises par l'éditeur de partition ; chacune des douze pistes peut être corrigée à la note près, et on peut même entrer les notes une à une en mode pas à pas si on n'est pas un instrumentiste virtuose ;
– précision du repérage grâce à un compteur gradué en mesure/temps/fraction de temps ce qui facilite le montage par copie et collage ;
– souplesse de reproduction : une fois que les codes sont enregistrés, on peut faire des essais de restitution sur des instruments tout à fait différents (mettre des cuivres à la place des cordes...), modifier le tempo, transposer, ajouter de la vélocité et en faisant varier les arrangements à l'infini trouver celui qui est le plus agréable.

CONNEXIONS

Au moment de la connexion, tous les appareils doivent se trouver de préférence hors tension. Les câbles utilisés sont terminés aux deux extrémités par des prises DIN mâles à 3 ou 5 broches standards (cordons audio utilisés en HI-FI).

On relie :

– la prise MIDI In du synthé à la prise MIDI Out de l'Atari ;
– la prise MIDI Out du synthé à la prise MIDI In de l'Atari.

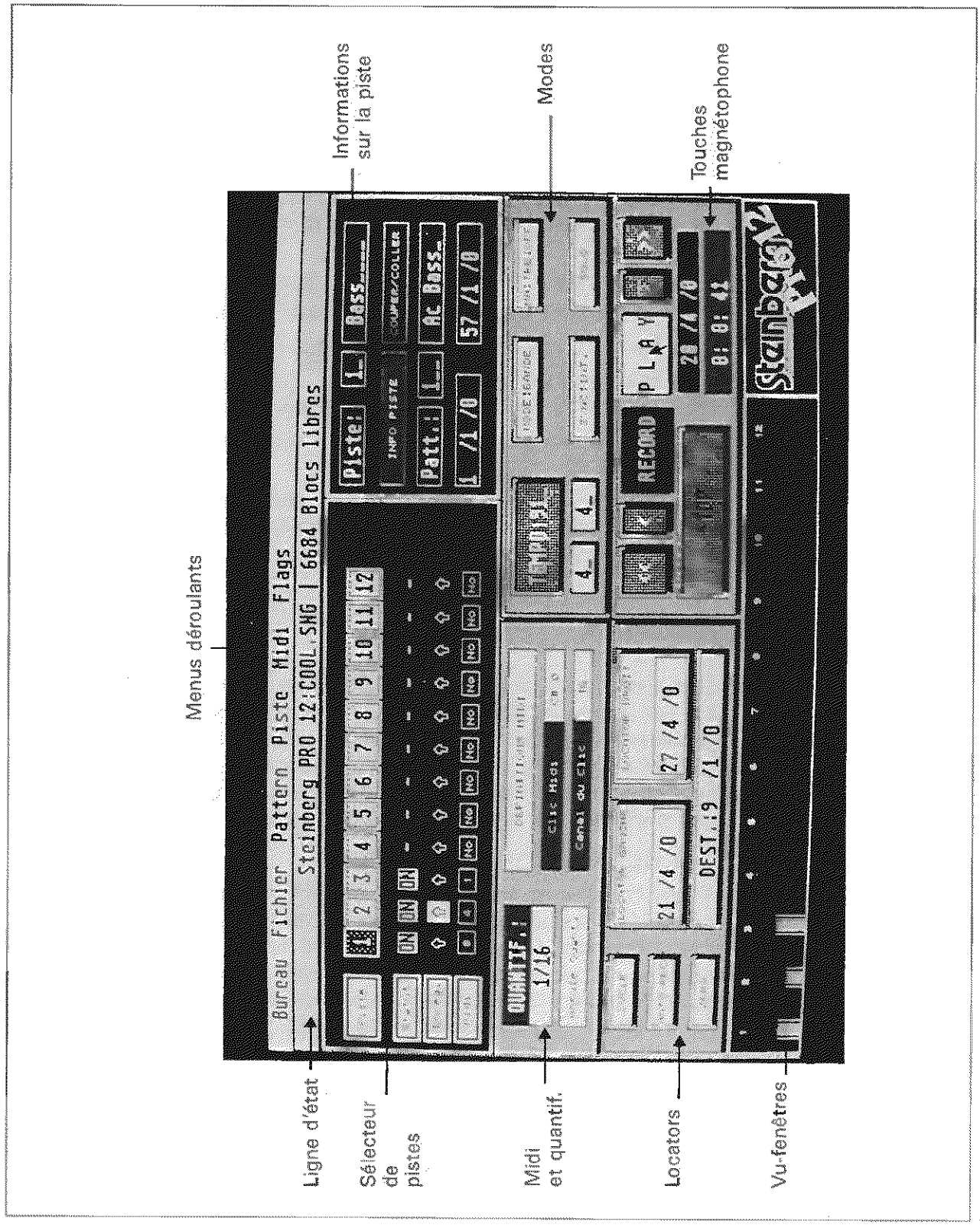
Si l'on dispose de plusieurs synthés, expandeurs ou boîtes à rythme, on utilise les prises MIDI Thru qui permettent de mettre plusieurs instruments en série (MIDI Out de l'Atari vers MIDI In du premier instrument, MIDI Thru du premier instrument vers MIDI In du suivant, et ainsi de suite) ; ou mieux encore, on utilise un boîtier de répartition MIDI en étoile qui évite tout retard de transmission dû aux déphasages qui s'accumulent au fil des circuits. L'instrument qui sert à l'entrée des données (clavier MIDI) doit bien sûr être relié à la prise MIDI In de l'Atari.

LES RÉGLAGES

La plupart des valeurs numériques affichées à l'écran peuvent être modifiées à l'aide de la souris. Si on clique sur une valeur, on l'augmente en utilisant le bouton droit de la souris et on la diminue en utilisant le bouton gauche. On peut aussi cliquer une fois sur la case pour la sélectionner puis modifier les valeurs à l'aide des touches [+] et [-] éventuellement combinées à la touche [ALTERNATE] qui en amplifie l'effet.

L'ÉCRAN PRINCIPAL

L'écran principal du séquenceur est divisé en huit fenêtres qui facilitent le repérage des boutons lors de leurs manipulations (voir écran 1).


Informations sur la piste
Modes
Touches magnétophone
Menus déroulants
Ligne d'état
Sélecteur de pistes
Midi et quantif.
Locators
Vu-fenêtres
Bureau Fichier Pattern Piste Midi Flags
Steinberg PRO 12:COOL SHG | 6684 Blocs libres
Piste:
Patt.:
Bass
Ac Bass
QUANTIF.:
1/16
RECORD
Steinberg

Ecran 1.

Sous la ligne des menus déroulants, on trouve une ligne d'état qui rappelle le nom de la musique courante ainsi que le nombre de blocs de 256 octets disponibles en mémoire.

- *La fenêtre du sélecteur* de pistes se compose de quatre lignes et de douze numéros de pistes.
- *La fenêtre d'information* sur la piste permet de nommer la piste et les patterns (morceaux de partition) qu'elle contient, de modifier les paramètres de restitution (case INFO PISTE), d'effectuer des duplications.
- *La fenêtre MIDI* permet d'activer les filtres MIDI (case DEFINITION MIDI) et d'ajuster les paramètres de quantification.
- *La fenêtre mode* définit le tempo, la synchro, etc.
- *La fenêtre des locators* marque une zone de la bande magnétique virtuelle sur laquelle s'effectueront des opérations de portée limitée.
- *La fenêtre des touches magnéto* reproduit les fonctions habituelles de cet appareil.
- *La fenêtre des vu-mètres* permet de vérifier sur quel canal transitent les informations MIDI et que la communication s'effectue correctement avec l'instrument.

CANAUX MIDI

Chacune des douze pistes de PRO12 peut envoyer ses informations sur l'un des seize canaux MIDI. Si l'on souhaite que les informations parviennent bien à l'instrument auquel elles sont destinées, il faut d'abord connaître le canal MIDI reconnu par ce dernier ou, si celui-ci le permet, choisir un numéro de canal qui lui sera propre ; cette initialisation s'effectue sur l'appareil lui-même à l'aide de sa notice.

Trois cas généraux peuvent se présenter :

1/ Les instruments anciens ou qui ne bénéficient pas d'une implémentation MIDI complète sont capables de recevoir sans réglage les données provenant de n'importe quel canal. En contrepartie il est préférable de ne connecter qu'un seul appareil de ce type à l'ordinateur puisqu'il traite indifféremment toutes les données, y compris celles qui ne lui sont pas destinées (mode **OMNI**).

2/ Si l'on possède plusieurs synthétiseurs mono-timbraux qui ne peuvent restituer qu'un seul type de son à la fois, on affecte à chacun un numéro de canal différent de façon à pouvoir obtenir simultanément des sons d'instruments différents (mode **POLY**). Certaines boîtes à rythme ne permettent pas de changer le numéro des canaux MIDI qui leur sont attribués ; si on ne connaît pas ces numéros, on peut les

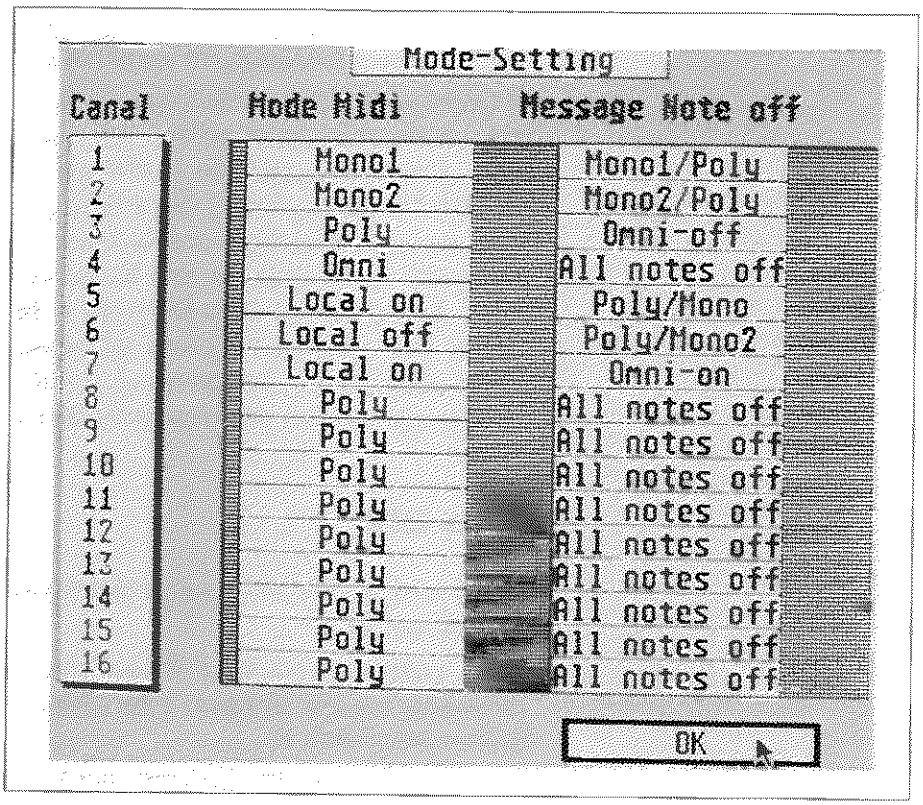
retrouver en faisant jouer une musique sur la piste 1 de PRO12 tout en modifiant progressivement le numéro de canal affecté à cette piste.

3/ Dans le cas des synthétiseurs ou des expandeurs multi-timbraux chaque machine peut se voir attribuer autant de canaux MIDI différents qu'il est possible de définir de voix simultanées (mode **MONO** ou **MULTI**). Les numéros de canaux étant connus et réglés sur les claviers ou les expandeurs connectés à l'ordinateur, il reste à les mettre en relation avec chacune des douze voix de PRO12. Après le chargement du logiciel, la ligne MIDI de la fenêtre de sélection des pistes affiche une série de numéros de canaux au hasard parfois remplacés par le code No. Un clic-droit sur le numéro de canal MIDI incrémente celui-ci, alors qu'un clic-gauche le décrémente.

Modes MIDI

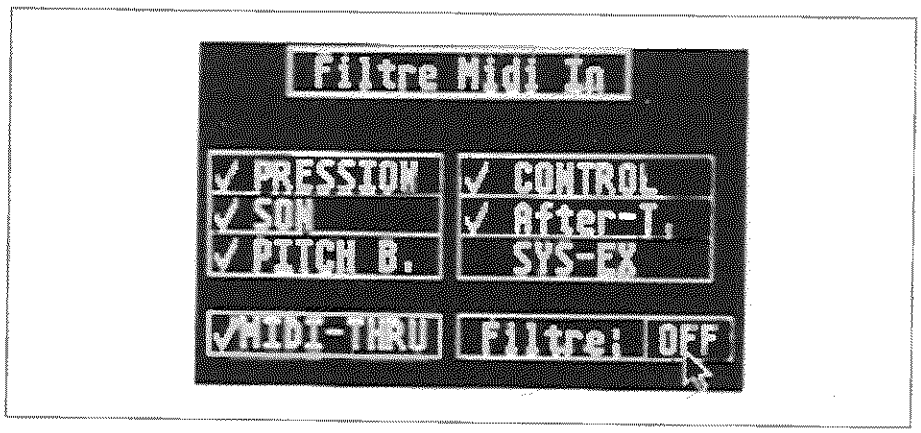
Cette fenêtre de configuration accessible par **MIDI/MODES** MIDI permet de définir :

– les codes à envoyer à chacun des instruments connectés sur les différents canaux de façon à les initialiser dans leur mode principal (par défaut mode **POLY**) ;
– le code qui permet d'arrêter une note si le déroulement de la bande est interrompu (par défaut *all notes off*, si une note reste bloquée après appui sur **STOP**, il faut essayer les autres codes).



Filtres MIDI

Dans la fenêtre MIDI, un clic sur la case DEFINITIONS MIDI fait apparaître les filtres disponibles en entrée sur l'Atari. Ces filtres laissent passer les codes MIDI correspondants s'ils sont précédés d'une coche ; dans le cas contraire les codes ne sont pas pris en compte.





PRESSION	Filtre la pression polyphonique applicable pour chaque touche (vélocité).
SON	Filtre les changements de sons (Program change).
PITCH BEND	Filtre les commandes de pitch bend.
CONTROL	Filtre les contrôleurs de tous types, pédales, molettes de modulation...
AFTER TOUCH	Filtre la pression monophonique applicable à l'ensemble des touches (volume).
SYS-EX *(System Exclusive)*	Filtre les codes de commande particuliers à certains synthés et qui ne sont pas prévus dans la norme MIDI (modification des paramètres d'un son par exemple). Si on coche ce filtre, il est préférable d'enregistrer ces codes seuls sur une piste séparée qui pourra être désactivée ou effacée si on change d'instrument par la suite.
MIDI-THRU	Doit être activé si on utilise en entrée un instrument qui ne produit pas de sons directs (accordéon MIDI, guitare...) de façon à pouvoir l'entendre sur un expandeur ou un synthé placé en sortie lorsqu'on joue. Avec un synthé, le **MIDI-THRU** de l'Atari risque de renvoyer les codes qui sortent de la prise OUT de l'instrument vers la prise IN de ce dernier ce qui fait entendre avec un léger décalage à la fois le son direct et l'effet des codes qui ont transité par l'ordinateur (le filtre LOCAL du synthé peut également résoudre ce type de problème).
FILTRE	Permet de sélectionner un canal MIDI parmi les seize existant, et qui ne renverra pas d'information sur la **MIDI-THRU** de l'Atari même si cette case est cochée (pour placer un synthé qui ne possède pas de local on/off par exemple).

AUTRES PARAMETRES MIDI

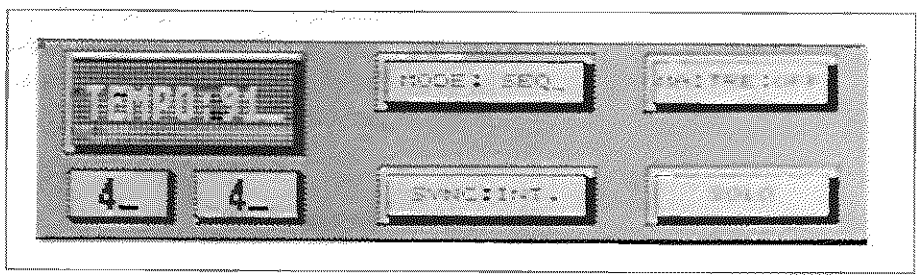
RESET APRES STOP	Cette option du menu **FLAGS** remet à zéro tous les contrôleurs ainsi que le pitch bend à chaque interruption par **STOP**.
RUNNING STATUS	Cette option du menu **FLAGS** compresse les données MIDI pour accélérer les échanges (les octets d'état ne sont pas répétés à chaque note s'ils ne sont pas modifiés). En cas de problème avec certains (vieux !) synthés, on peut essayer de désactiver ce drapeau.
SYNC	Ce bouton (touche **[M]**) peut prendre la valeur **INT** (synchro interne de PRO12) ou **MIDI** (synchro externe). Dans ce cas, le tempo, le départ et l'arrêt des morceaux dépendent de l'instrument connecté.
Le carnet de notes	Accessible depuis le menu **MIDI/CARNET DE NOTES** propose six lignes destinées à conserver un éventuel commentaire au moment de la sauvegarde.

Pour effectuer des manipulations sur les commandes qui vont suivre, on peut charger la musique COOL.SNG fournie sur la disquette PRO12 depuis le menu **FICHIER/CHARGER SONG**. Celle-ci occupe trois pistes (basse, accompagnement, mélodie). Si on ne possède qu'un seul synthé réglé sur un unique canal, on initialisera chacune des pistes sur ce même canal MIDI dans la fenêtre du sélecteur de pistes et on choisira un son qui permet de bien différencier les notes jouées sur chacune des pistes, par exemple un son de piano.

Métronome

Si on active le menu **FLAGS/BIP** on entend le battement d'un métronome sur le haut-parleur du moniteur de l'Atari en mode **PLAY**. En mode **RECORD**, le métronome du moniteur est automatiquement activé.

On peut doubler (ou remplacer en mode **PLAY**) le métronome du moniteur par le menu **MIDI/CLIC MIDI** qui fait entendre le battement sur un synthé extérieur : dans la fenêtre MIDI, la case **CANAL DU CLIC** permet de définir le canal sur lequel est branché l'appareil qui doit générer le clic et la case **CLIC MIDI** détermine la note qui doit être jouée pour ponctuer les temps (pour être audible, elle ne doit pas être trop grave : par exemple c2).



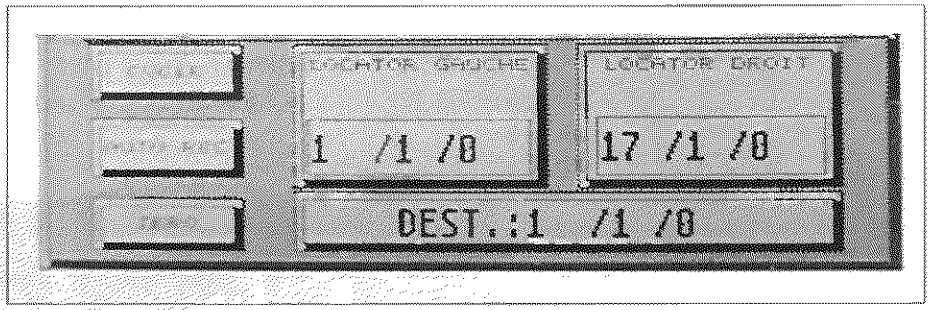
Tempo

Il s'agit d'une modification globale qui concerne la reproduction de la totalité d'une musique. Il suffit de faire un clic-droit ou un clic-gauche sur le mot **TEMPO** de la fenêtre MODE pour accélérer ou ralentir la musique sans que la fréquence des sons ne soit pour autant modifiée. On peut aussi enregistrer plusieurs modifications du tempo à l'intérieur d'un même morceau en utilisant la piste maître comme nous le verrons par la suite.



LES LOCATORS

Les Locators sont un élément essentiel au fonctionnement de PRO12 : les valeurs qu'ils contiennent sont pratiquement toujours prises en compte quelles que soient les commandes utilisées.



Ce sont deux points qui délimitent une partie de la bande ; on les nomme locator gauche et locator droit en se référant au sens de défilement d'une bande magnétique. Leur valeur apparaît sous la forme mesure/temps/fraction de temps, ce dernier paramètre avec une précision d'un 96e de temps.

Ils permettent de délimiter une zone d'enregistrement ou une pattern, de créer une boucle répétitive en lecture, de délimiter une portion de bande à couper ou à insérer...

On peut modifier les valeurs qui se trouvent dans chacun des locators de quatre façons différentes :

1/ Par un clic-droit ou un clic-gauche de la souris sur la case de chaque locator.

2/ Grâce aux touches [L] *(Left)* ou [R] *(Right)* qui font apparaître une ligne de saisie de la nouvelle valeur à entrer au clavier. Dans le cas du locator droit, on peut se contenter de taper le signe [+] suivi du nombre de mesures que l'on veut délimiter et le logiciel effectue lui-même le calcul du point d'arrivée.

3/ Par recopie de la valeur de n'importe quelle case de l'écran qui contient une indication sous la forme mesure/temps/fraction de temps ; on fait par exemple un cliqué-maintenu sur le compteur de défilement et on le tire sur le locator gauche ou droit qui prend alors cette valeur.

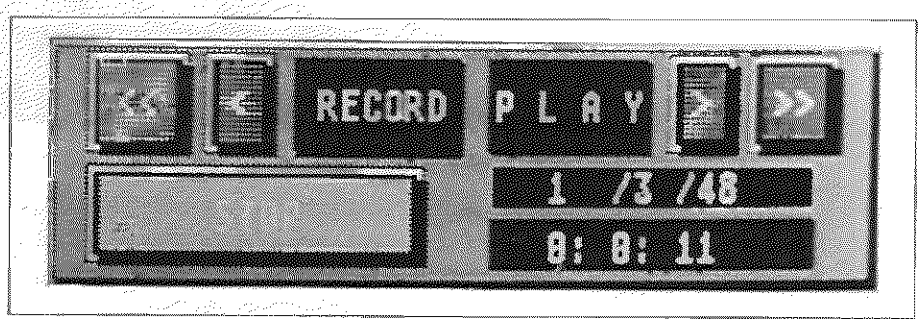
4/ Au vol en mode PLAY ; il suffit d'appuyer sur [SHIFT][L] ou [SHIFT][R] pour affecter le locator correspondant de la valeur du compteur de défilement.

Les valeurs des deux locators peuvent être mémorisées si on appuie sur la touche [SHIFT] en même temps qu'une des dix touches de

fonction. Par la suite, un nouvel appui sur la touche de fonction seule, repositionnera les locators à la valeur mémorisée.

LES TOUCHES MAGNÉTO

Le fonctionnement en mode bande utilise les commandes d'un magnétophone classique, aussi n'est-on pas surpris d'en retrouver le pavé de boutons de défilement : PLAY, RECORD...



On dispose cependant de deux types de touches de rembobinage et d'avance :

- rapide, mesure par mesure << et >> ;
- précise, à la croche près < et >.

Le compteur de défilement est double ; il affiche dans sa fenêtre supérieure le numéro de la mesure et du temps courant, la fenêtre inférieure chronomètre à la seconde près la durée d'exécution. Lorsqu'on arrête la bande par STOP, le compteur affiche toujours une fraction de temps égale à 0 ou 48. Si on souhaite une valeur plus précise, il faut demander un arrêt automatique à l'aide du locator droit (en enregistrement seulement).

Le bouton PLAY est doublé au clavier par la touche [RETURN]. Le morceau commence à être joué depuis la position du compteur et se déroule jusqu'à ce qu'on appuie sur la touche [STOP] qui est doublée par la barre d'espace. Le mode PLAY n'est pas concerné par la position des locators.

Le mode RECORD sera détaillé dans la partie traitant de l'enregistrement.

RESET DU COMPTEUR

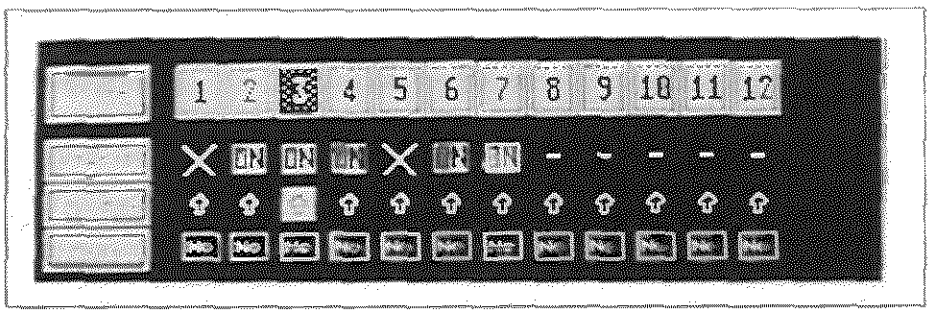
Le bouton ZERO rembobine toute la bande et remet les compteurs de défilement et de temps à zéro. Il est doublé au clavier par la

touche [0]. On peut aussi ramener directement le compteur de défilement à la valeur contenue dans l'un des deux locators, en cliquant sur la zone blanche qui se trouve immédiatement sous le nom LOCATOR DROIT ou LOCATOR GAUCHE ou en appuyant sur les touches [(] ou [)] du clavier.

Cycle

Le bouton CYCLE permet de répéter indéfiniment une série de mesures délimitées par les locators gauche et droit. Ce bouton est doublé au clavier par la touche [C] et fonctionne comme une bascule. En mode CYCLE, la musique est déjà jouée une fois de la mesure 1 à la mesure placée dans le locator droit, à la suite de quoi elle boucle du locator droit au locator gauche tant que le bouton reste activé.

Le sélecteur de pistes



Le sélecteur de piste sert à choisir :

- la piste dont on veut modifier, copier, effacer le contenu ;
- les pistes que l'on souhaite entendre (y compris pendant que l'on enregistre) ;
- la piste sur laquelle on veut enregistrer ;
- le numéro du canal MIDI affecté à la piste.

Aussi le sélecteur est-il constitué de quatre lignes :

1/ Le numéro de la piste

On peut cliquer dessus pour sélectionner celle-ci ou appuyer sur un des chiffres de 1 à 9, [SHIFT][0], [SHIFT][1] et [SHIFT][2] pour les pistes 10 ,11 et 12. Sélectionner une piste a pour effet de modifier la fenêtre d'information sur la piste située à droite.

2/ Le statut de la piste

Il peut être à - (piste vide), ON (piste active : un clic), X (piste momentanément désactivée : un clic) ou S (mode séquenceur : un double-clic, le bouton MODE étant placé sur SEQ au lieu de BANDE).

Il est intéressant de pouvoir désactiver momentanément une piste en cours d'écoute ou d'enregistrement afin de mieux entendre certaines parties qui seraient sinon noyées parmi toutes les voix (voir aussi le bouton SOLO).

3/ La flèche d'enregistrement

On ne peut la positionner que sous une seule piste à la fois : c'est sur cette piste que s'effectue l'enregistrement lorsqu'on appuie sur RECORD. On peut la déplacer à l'aide des touches fléchées droite et gauche du clavier.

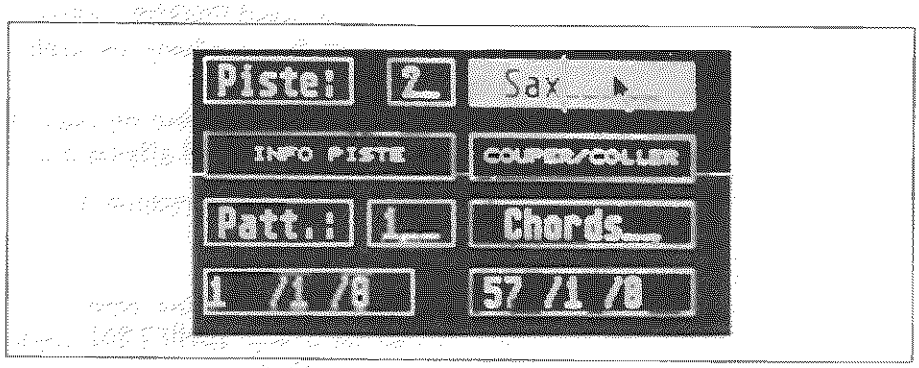
4/ Le numéro du canal MIDI affecté à chaque piste

On peut le faire varier de 1 à 16, No signifie qu'il s'agit du canal qui a été utilisé à l'enregistrement (le changement de voix et le réglage du volume monophonique ne doivent alors pas être utiles).

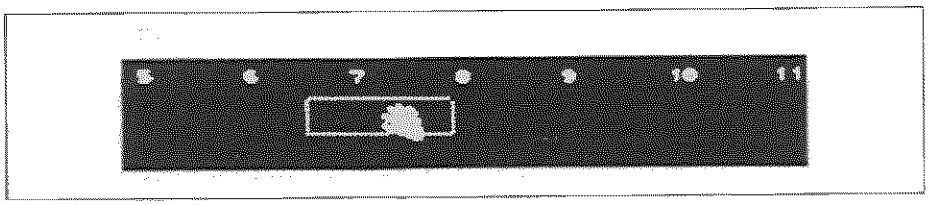
Voici quelques règles de fonctionnement et manipulations utiles sur les pistes :

• Pour sélectionner une piste, on clique sur son numéro dans le sélecteur de piste. Cette piste devient la piste courante. Attention, la piste courante (dont le numéro est en inversion vidéo) n'est pas forcément la même que la piste active pour l'enregistrement (sous laquelle se trouve une flèche). Toutes les opérations d'édition, de copie, de suppression concernent la piste courante, donc prudence !

• Pour nommer la piste courante, on clique sur son nom dans la fenêtre d'information sur les pistes, on appuie sur [ESC] ou [BACK SPACE] pour vider la case puis on tape le nouveau nom qu'on valide par [RETURN].



• Pour effacer définitivement la piste courante, on fait un cliqué-maintenu sur le nom PISTE de la fenêtre d'information sur la piste, afin que le pointeur prenne la forme d'une main, puis on la tire sur la zone des vu-mètres qui fait office de poubelle. Après confirmation de l'effacement, la ligne STATUT affiche le signe moins sous le numéro de la piste indiquant que celle-ci est maintenant vide.



• Pour copier la piste courante sur une autre, on la prend de la même façon que pour un effacement, mais on la tire sur le numéro de la piste sur laquelle on veut la dupliquer. Tout recouvrement est signalé, mais il est trop tard pour changer d'avis.

Mode Solo

Le bouton **SOLO** de la fenêtre **MODE** n'autorise l'écoute que d'une seule piste à la fois ; si on clique sur une des croix de la ligne **STATUS** elle passe à ON et la piste précédemment active se retrouve à son tour automatiquement neutralisée.

Modification du son

Après avoir sélectionné la piste et la pattern désirée, on clique sur **INFO PISTE** (dans la fenêtre d'information sur la piste) ce qui ouvre une fenêtre fugitive proposant divers réglages ; cette fenêtre est fugitive, car elle disparaît dès que la souris en sort.



Les valeurs à OFF ou à zéro signifient qu'aucune modification n'est apportée aux réglages d'origine de l'expandeur ou du synthé.

Les modifications concerneront soit toutes les patterns de la piste active soit une seule pattern selon le message que l'on affiche en cliquant sur la première case de la fenêtre. Comme les paramètres de reproduction sont passés une seule fois à l'expandeur avant que les

notes ne soient envoyées (program change), il est nécessaire de revenir au début de la pattern pour que les valeurs soient effectivement actives et que le son apparaisse modifié lorsqu'on appuie sur **PLAY**. Cependant pour certaines options en mode **PLAY**, telle que le choix de la voix ou du volume monophonique, si on souhaite juger de l'effet produit au moment où on modifie les valeurs à l'aide de la souris, on les transmet immédiatement à l'instrument en maintenant appuyée la touche **[CONTROL]** du clavier au moment où l'on clique (**[CONTROL]** force l'envoi immédiat du program change au synthé).

Si on n'utilise qu'un seul appareil mono-timbral fonctionnant sur plusieurs pistes mais sur un unique canal, il est recommandé de régler les paramètres **Voix** et **Volume** toujours sur la même piste (par exemple la 1) puisqu'il s'agit alors de paramètres globaux (le synthé ne peut reproduire qu'une seule voix à la fois et cette voix n'a qu'une seule intensité). Les autres paramètres sont eux indépendants pour chacune des pistes.

VOIX

Envoie un numéro de program change qui définit le numéro du nouveau son affecté au canal (piano, saxo, xylophone...) à condition que le numéro du canal MIDI de la piste ne soit pas à No. On décale vers le haut le numéro de la voix courante de la valeur souhaitée. On ne dépasse cependant jamais la valeur 128 qui correspond au nombre maximal de voix adressables en MIDI sur un synthé ; au-delà de 128, on obtient un numéro de voix MODULO 129. Ainsi, on parvient à décaler le numéro de la voix vers le bas en lui ajoutant 129 moins la valeur du décalage. Ex. : le numéro de la voix courante est 42. Pour passer à la voix 40, on ajoute 127 (soit 129 - 2). On obtient alors le résultat de l'opération (42 + 127) MODULO 129 c'est-à-dire 40. Il est préférable de laisser ce paramètre à OFF si on utilise une boîte à rythme.

VOLUME

Modifie en valeur absolue l'intensité sonore du canal sélectionné dans une fourchette allant de 1 à 127, à condition que le canal MIDI de la piste ne soit pas à No.

VELOCITE

Définit le volume polyphonique, ce qui en fonction de l'instrument sélectionné affecte soit l'intensité du son, soit son timbre. Si la case **DECAL** à droite de **VELOCITE** est active, la valeur de vélocité est ajoutée ou retranchée à la valeur initialement enregistrée pour chacune des notes jouées : une valeur de - 128 produit un son plus doux, plus feutré alors que + 128 donne de la brillance, augmente les harmoniques de la note jouée. En cliquant sur **DECAL**, on fait passer la case à **FIXE**. La valeur de vélocité initialement enregistrée avec chaque note est alors ignorée et c'est la valeur sélectionnnée dans la fenêtre qui est appliquée à toutes les notes de la piste qui sont alors jouées avec une égale intensité.

DELAI

Anticipe ou retarde l'attaque des notes de la piste ; on peut ainsi créer des effets de gros son ou d'écho. Les valeurs maximales sont de plus ou moins 96, soit l'équivalent d'une noire.

TRANSPOSE

Change la hauteur de la piste courante par pas d'un demi-ton dans la limite de plus ou moins 48 demi-tons : une valeur de plus ou moins 12 transpose la piste à l'octave supérieure ou inférieure.

ENREGISTREMENT

Attention : la durée et la position de l'enregistrement sur une piste dépend du réglage des locators.

Après avoir positionné ceux-ci et amené la flèche de la ligne **ENREG** sous la piste sur laquelle on souhaite enregistrer, on clique sur **RECORD** (touche [*] du clavier) dans la fenêtre des touches magnéto. Le bouton **AUTO-REC** passe en inversion vidéo, PRO12 active le métronome et fait entendre deux mesures à vides sur le moniteur de l'Atari ; c'est seulement après ces deux mesures que les boutons **RECORD** et **PLAY** passent en inversion vidéo et que le compteur commence à défiler. Tout ce qui est joué sur l'instrument est stocké en mémoire. L'enregistrement se termine lorsqu'on clique sur la touche **STOP** (touche [ESPACE] du clavier) ou lorsque le compteur arrive à la mesure indiquée dans le locator droit. La fenêtre d'information sur la piste affiche sous PATT, le numéro des mesures de début et de fin de ce qui a été enregistré : on vient de créer une pattern dont le numéro est indiqué à droite de PATT.

On peut ensuite enregistrer une autre partie sur une deuxième piste tout en écoutant ce qui se trouve sur la première.

Attention : chaque fois qu'on enregistre sur une piste qui n'est pas vierge, on efface ce qui s'y trouve, comme sur un magnétophone normal.

PUNCH IN MANUEL

Si on souhaite se référer au contexte en écoutant d'abord ce qu'on vient d'enregistrer avant de continuer, on peut activer l'enregistrement à n'importe quel moment pendant la lecture de la bande (le mode **CYCLE** étant désactivé) ; en mode **PLAY** il suffit d'appuyer sur le bouton **RECORD** ou la touche [*] au moment voulu pour que l'enregistrement s'effectue « dans la foulée ».

Punch in automatique

On peut également programmer un enregistrement automatique en cours de lecture. On entend ce qui a déjà été enregistré, puis au moment programmé dans le locator gauche l'enregistreur se met automatiquement en route et s'arrête à la mesure placée dans le locator droit. Pour cela il faut déjà régler les locators à la valeur souhaitée, cliquer sur le bouton **AUTO-REC** (touche **[A]** du clavier) puis lancer une lecture normale par **PLAY** depuis le début de la bande ou quelques mesures avant le premier point.

Enregistrement en mode cycle

Il est souvent intéressant de boucler sur un passage difficile de façon à pouvoir le saisir plusieurs fois de suite et ne s'arrêter que lorsque la « prise » est bonne. Avant de lancer l'enregistrement, on positionne les locators sur la partie à répéter et on active le bouton **CYCLE** (touche **[C]**). On dispose depuis le menu **FLAGS/CYCLES** de trois types de cycles :

– **NORMAL** : à chaque nouvelle boucle on efface tout ce qui avait été précédemment enregistré (comme lorsqu'on utilise une boucle sans fin sur un magnéto : il est préférable d'avoir de bons réflexes pour s'arrêter à temps !) ;

– **MIX** : on n'efface jamais rien et tout ce qu'on joue s'accumule sur la piste (mixage : on supprime l'effet de la tête d'effacement du magnéto) ;

– **AUTO** : on n'efface que si on joue une nouvelle note au cycle suivant et seulement à partir de cette note (**punch-in** automatique sur une boucle).

Les patterns

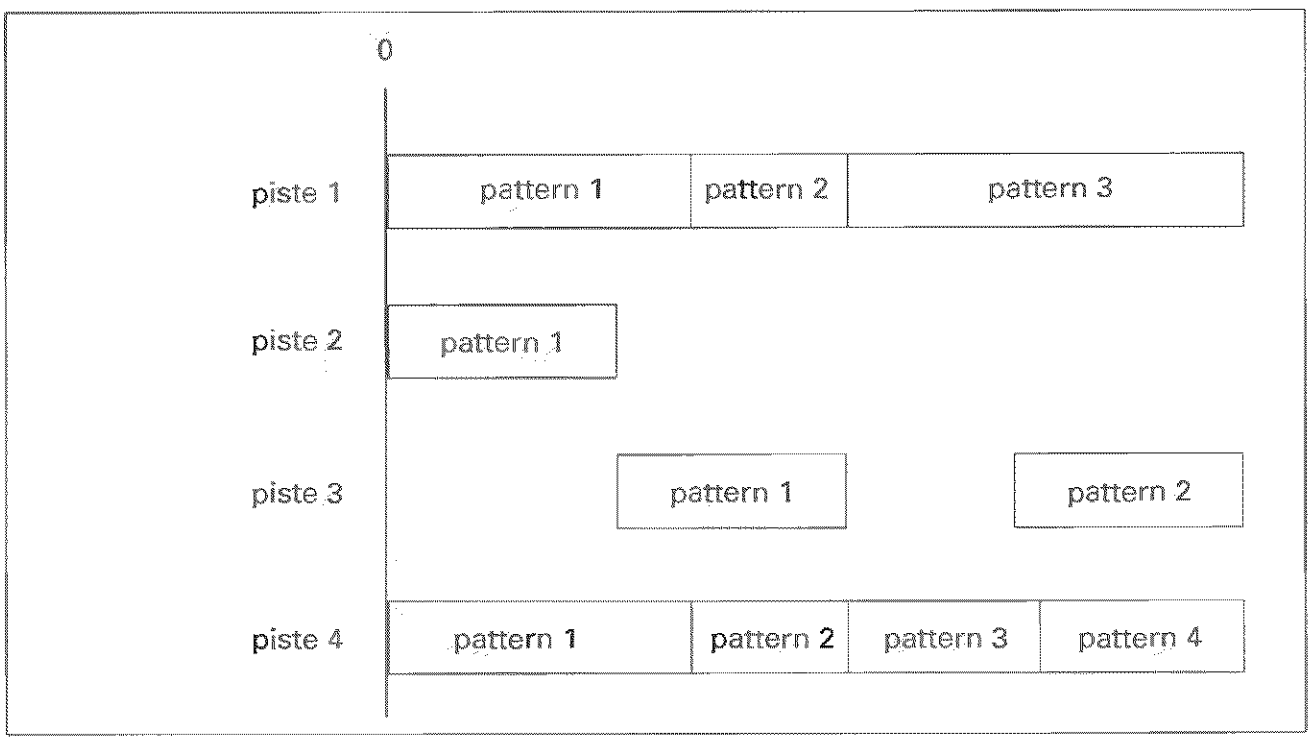
La pattern est tout à fait comparable en musique à ce que peut être un sous-programme en informatique : c'est un moyen de diviser un ensemble complexe, à savoir une musique (une *song* dans la terminologie PRO12), en éléments plus petits et plus simples qui pourront être travaillés et mis au point indépendamment les uns des autres avant d'être copiés, déplacés puis réunis pour former l'œuvre finale.

Nous avons vu une première possibilité de décomposition qui consiste à enregistrer sur des pistes séparées les différentes voix d'un morceau. Toutes ces voix sont par la suite jouées simultanément.

Les patterns réalisent, elles, une décomposition chronologique à l'intérieur d'une même piste, par exemple intro, couplet, refrain...

Une pattern commence et se termine à une position définie par les locators mais tout à fait indépendante de celle des autres patterns. Elle porte un numéro d'ordre sur la piste sur laquelle elle se trouve et on peut même lui donner un nom qui facilite son identification, surtout si on la copie sur une autre piste où elle a toutes les chances de porter alors un numéro différent : toutes les premières patterns de toutes les pistes portent le numéro 1, mais la pattern 1 de la piste 7 n'est sûrement pas la même que la pattern 1 de la piste 8.

Les patterns peuvent être séparées par des vides de longueurs quelconque et leurs tailles sont très différentes, une ou deux mesures pour une partie de base qui est répétée tout au long du morceau, cinq ou six pour un refrain, plusieurs dizaines pour une partie mélodique. On les duplique sur une même piste ou d'une piste à l'autre.



Création de pattern

Pour créer une pattern, on dispose de deux possibilités :

– on utilise le mode enregistrement comme nous l'avons vu précédemment : tout ce qu'on enregistre se trouve nécessairement à l'intérieur d'une pattern ;

– on crée une pattern vide (ou plutôt remplie de silences) depuis le menu PATTERN/CREER PATTERN puis on la complète à l'aide de la souris ou note à note à l'aide d'un instrument MIDI depuis le menu PATTERN/PARTITION. Cette méthode est surtout intéressante pour de très courtes patterns destinées à être recopiées (parties de percussions par exemple). Un chapitre spécial est dédié au fonctionnement de l'éditeur de partition.

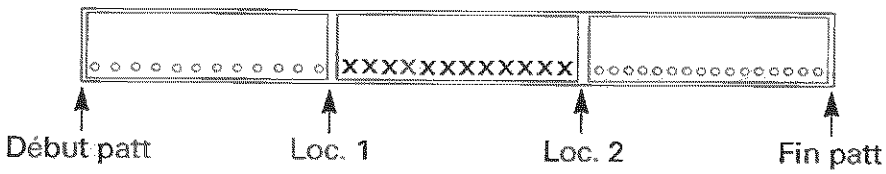
REGLES ET MANIPULATIONS CONCERNANT DES PATTERNS

Locators

Tout ce qu'on enregistre l'est à l'intérieur d'une pattern délimitée par les locators qu'il faut penser à initialiser aux valeurs correctes avant d'appuyer sur le bouton RECORD ; l'enregistrement s'arrête automatiquement à la position du locator droit mais on peut cependant l'interrompre prématurément par le bouton STOP.

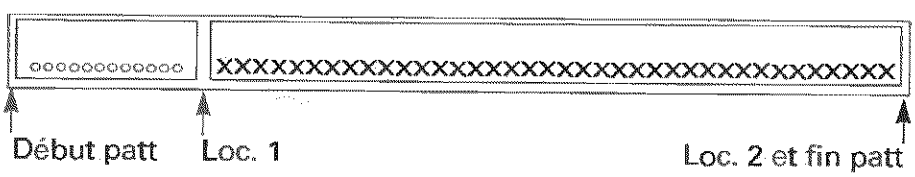
Correction

Si on enregistre à l'intérieur d'une pattern existante on ne modifie que la partie située entre les nouveaux locators ; c'est utile pour corriger quelques mesures sans refaire toute la pattern. La pattern ne change alors ni de longueur, ni de position.



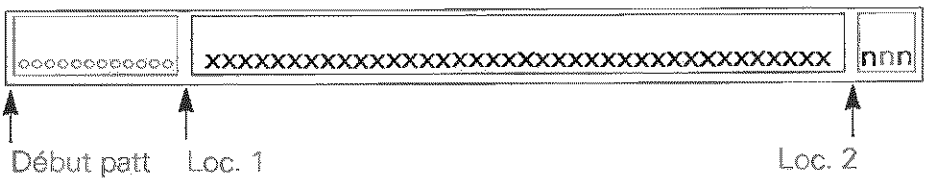
Allongement

Si on commence l'enregistrement à l'intérieur d'une pattern alors qu'on a placé le locator 2 après la fin de la pattern, celle-ci est rallongée jusqu'à ce nouveau point d'arrêt.



Fusion

Si en dépassant de cette façon la longueur de la pattern on vient chevaucher la pattern qui la suit immédiatement, les deux patterns sont réunies en une seule et prennent le numéro et le nom de la première.



Renumérotation

Si on insère d'une façon quelconque une pattern devant une autre, leurs numéros d'ordre sur la piste sont automatiquement réactualisés (d'où l'utilité de nommer les patterns plutôt que de se fier à des numéros changeants).

Sélection

On passe d'une pattern à l'autre sur une même piste en cliquant sur le numéro situé à droite du mot PATT dans la fenêtre d'information sur les pistes ou en utilisant la combinaison de touche **[SHIFT] [Flèche droite]** ou **[SHIFT] [Flèche gauche]**. La pattern dont le numéro est affiché est la pattern courante pour certaines opérations. Son nom et sa position apparaissent dans la case située tout à fait à droite de la ligne PATT.



Nom

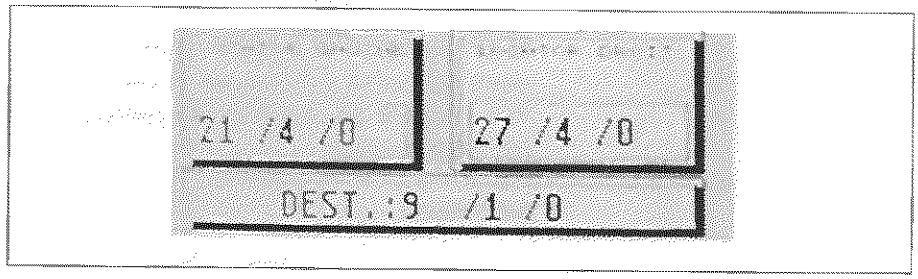
On donne un nom ou on change le nom de la pattern courante en cliquant sur le nom affiché à droite de son numéro dans la fenêtre d'information sur les pistes. **[ESC]** ou **[BACKSPACE]** effacent l'ancien nom et il ne reste plus qu'à frapper le nouveau et à le valider par **[RETURN]**.

Effacement

On efface la pattern courante en faisant un cliqué-maintenu sur son nom et en la tirant sur la zone des vu-mètres. Une boîte de confirmation de l'effacement apparaît alors et on peut changer d'avis.

Copie

Avant de copier une pattern on doit définir la position de destination dans la case **DEST** située sous les locators. On clique alors sur le mot **PATT** comme pour un effacement mais on le glisse sur le numéro de la piste où doit s'effectuer la copie (ce peut être la même piste mais à un endroit différent).



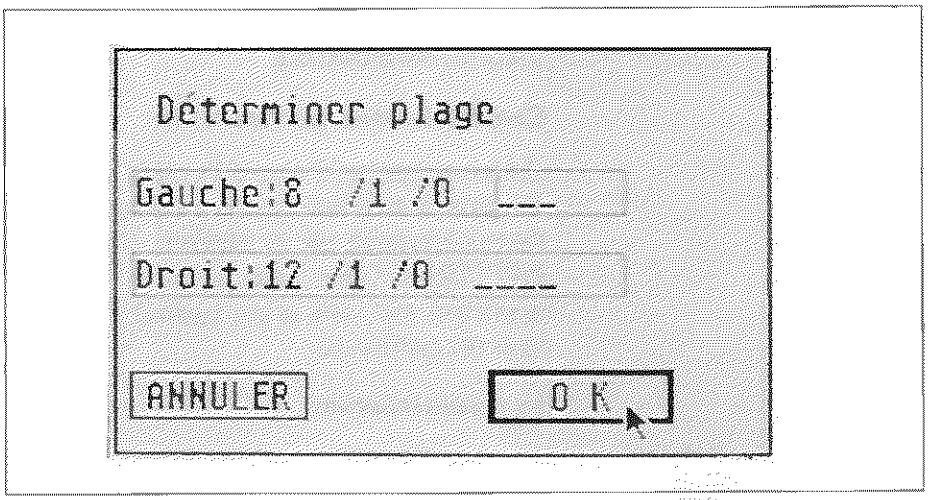
En cas de recouvrement d'une pattern existante, une boîte d'alerte apparaît. Attention, c'est la petite main qui tient la case que l'on doit placer exactement sur le numéro de la piste. La pattern ainsi dupliquée porte le même nom que l'original mais est précédée du signe moins (les patterns doivent toutes avoir des noms différents quelle que soit la piste sur laquelle elles se trouvent). Si on appuie sur la touche **[SHIFT]** pendant l'opération de copie, la pattern ainsi dupliquée conserve tous ses liens avec l'originale : si on quantifie ou corrige cette dernière depuis l'écran d'édition, les modifications sont répercutées automatiquement sur le double. Le double se reconnaît au fait que son nom est précédé du signe **&** à la place du signe moins.

Déplacement

On déplace une pattern de la même façon qu'on la copie mais en maintenant la touche **[CONTROL]** enfoncée pendant l'opération.

Copie d'une partie

On extrait préalablement les mesures à copier en double-cliquant sur **COUPER/COLLER** dans la fenêtre des informations sur la piste. Une boîte de saisie apparaît dans laquelle on entre les données souhaitées.



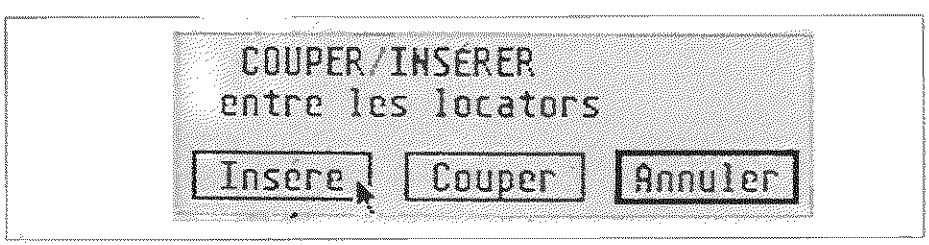
La copie s'effectue ensuite de la façon habituelle mais c'est la case COUPER/COLLER qu'on tire sur le numéro de la piste au lieu de la case PATT.

Modification de longueur

On positionne les locators de façon à ce qu'ils encadrent la partie à conserver sur la pattern courante. On choisit le menu PATTERN/COUPER LA PATTERN.

Insertion

Pour insérer une pattern sur une piste sans écraser celles qui s'y trouvent déjà, il faut positionner les locators de façon à ce qu'ils encadrent la zone à définir puis on appelle le menu PISTE/DECOUPE INSERTION GLOBALE qui affiche une boîte de confirmation. Celle-ci permet soit d'insérer une portion de bande vierge, soit de supprimer le morceau qui se trouve entre les locators. La partie qu'on ajoute ou qu'on retranche modifie toutes les pistes comme sur un magnéto normal, sauf si certaines pistes sont désactivées (statut X), auquel cas cette modification ne les concerne pas.



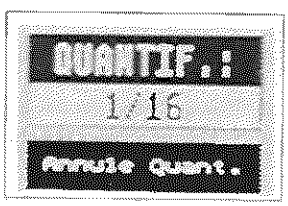
Toutes les patterns qui se trouvaient sur les pistes activées (statut ON) sont repoussées et leur locators réactualisés. On peut alors créer

une nouvelle pattern dans la zone ainsi insérée. Utile pour rajouter une intro qu'on a oubliée.

Création d'une pattern vide

On sélectionne d'abord la piste sur laquelle on veut créer la pattern. On positionne les locators de façon à ce qu'ils définissent la taille et l'emplacement de la pattern. On active le menu PATTERN/CREER PATTERN. Créer une pattern vide (elle est en réalité remplie de silences) est utile si on souhaite enregistrer note à note depuis l'éditeur de partitions.

La quantification

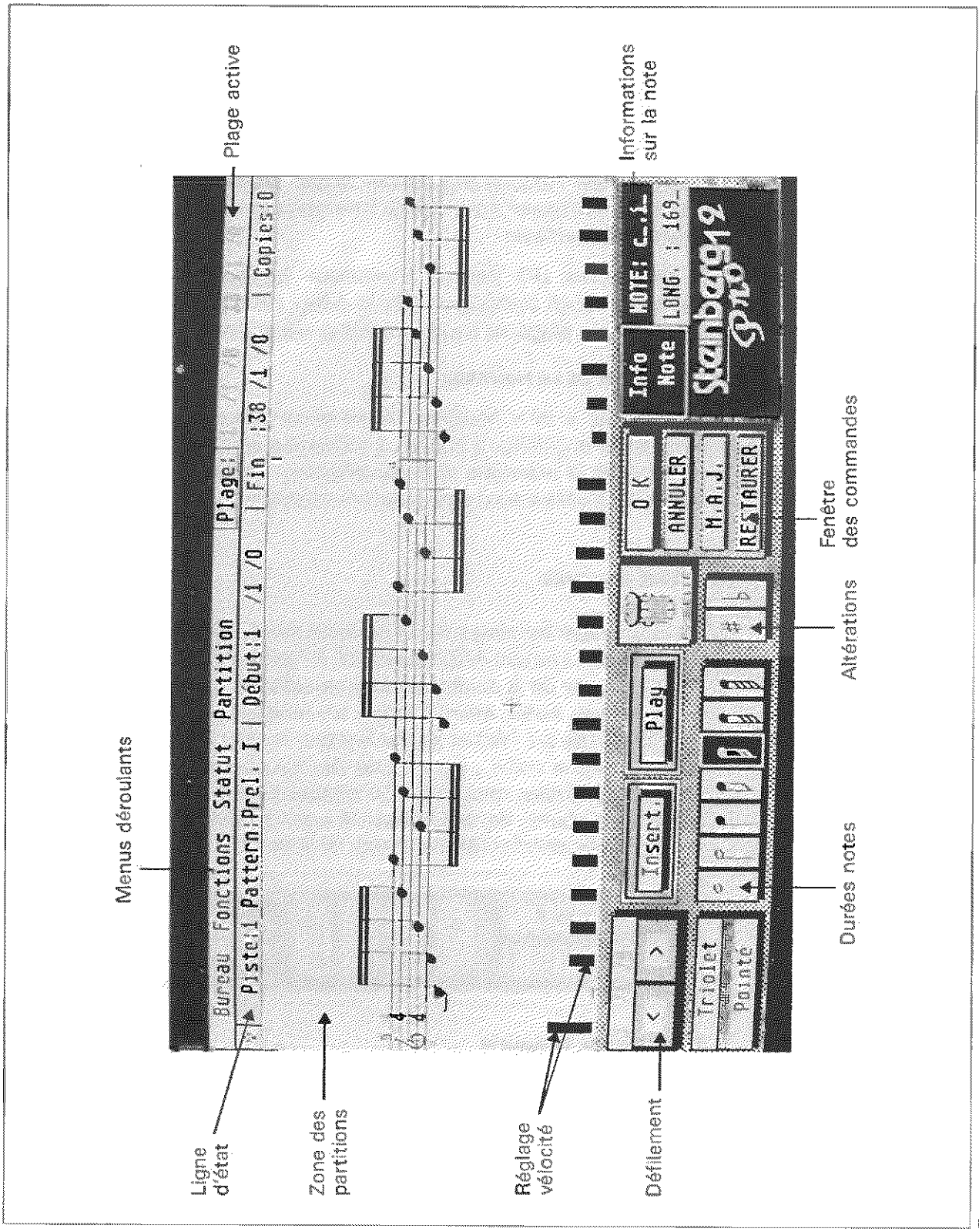


La quantification permet d'aligner automatiquement les notes sur certains temps ou fraction de temps qu'on a définis, ainsi les notes tombent « juste » dans le rythme sans qu'on soit nécessairement un virtuose.

Pour quantifier une pattern, il suffit de la sélectionner puis de choisir une valeur de quantification (de 1/4 à 1/32T de temps – T pour triolet) dans la fenêtre MIDI et dans la case située sous le mot QUANTIF qui contient OFF au départ. On clique alors sur le mot QUANTIF ce qui a pour objet de passer la case ANNULE QUANT. en inversion vidéo. On dispose en effet là d'un buffer qui sauvegarde la pattern originale et qui permet de la récupérer si la quantification se révèle décevante. Attention, ce buffer ne contient qu'une pattern à la fois et toute nouvelle quantification écrase la pattern précédemment sauvée.

L'éditeur de partitions

L'éditeur permet d'accéder aux notes de la pattern courante. Il peut s'agir d'une pattern saisie depuis un instrument et qui contient déjà des notes ou d'une pattern créée depuis le menu PATTERN/CREER PATTERN et qui ne contient que des silences. Le menu PATTERN/PARTITION (touche [D] donne alors accès à un nouvel écran (voir écran 2).


Plage active
Informations sur la note
Menus déroulants
Fenêtre des commandes
Altérations
Durées notes
Ligne d'état
Zone des partitions
Réglage vélocité
Défilement
Bureau Fonctions Statut Partition
Plage:
Piste:1 Pattern:Prel. I | Début:1 /1 /0 | Fin :38 /1 /0 | Copies:0
Info Note
NOTE: c_i
LONG. : 169
O K
ANNULER
M.A.J.
RESTAURER
Play
Insert.
Triolet
Pointé

Ecran 2.

Le bouton OK (touche [RETURN]) ramène à l'écran principal de PRO12 en validant les modifications effectuées sur la pattern.

ANNULER (touche [UNDO]) ramène à l'écran principal en abandonnant les modifications effectuées sur la pattern. Il en est de même de la case de fermeture située à gauche de la ligne d'état.

RESTAURER (Touche [R]) annule toutes les modifications de la session et on retrouve l'écran tel qu'il se trouvait lorsqu'on est entré dans l'éditeur de partition.

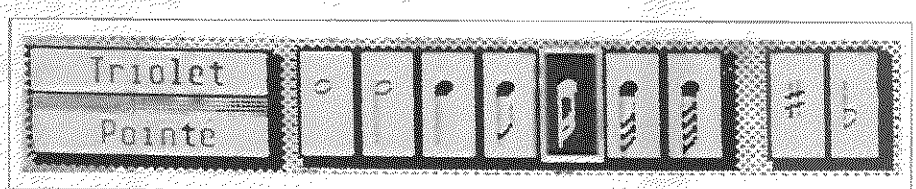
PLAY (touche [P]) permet d'entendre la partition affichée qui se répète de façon cyclique depuis le début de la pattern ou éventuellement dans la plage de travail si celle-ci est active.

DÉFILEMENT DE LA PARTITION

Les flèches < et > (touches flèche-droite, flèche-gauche du clavier) font défiler la partition à l'écran. L'indicateur situé au-dessus indique le numéro de la première mesure affichée. On peut observer que les barres de mesure sont également numérotées.

ENTRÉE DE NOTES

On peut placer les notes sur la partition soit à l'aide de la souris, soit à l'aide de l'instrument MIDI connecté à l'entrée de l'Atari, sans avoir à se préoccuper de la durée pendant laquelle on appuie sur les touches de ce dernier. Aussi, avant d'entrer les notes, il faut d'abord sélectionner leur durée sur l'écran de l'ordinateur en cliquant sur les symboles ronde, blanche noire... ou à l'aide des touches [1] à [7], indiquer si elles sont pointées (touche [.]) ou triolées (touche [T]). Si on les écrit à l'aide de la souris, on doit activer le type d'altération éventuellement désiré, dièse (touche [#]) ou bémol (touche [B]).



Entrée avec la souris

On coche la touche INSERT (touche [INSERT] du clavier) puis on place le pointeur de la souris sur la partition très exactement sous le silence devant lequel la note doit être placée. Le nom de la note s'affiche en haut de l'écran à gauche suivant le codage anglo-saxon. Un clic dessine la note à l'écran. Si on positionne le pointeur sur une note

existante on réalise un accord. Pour placer un silence, on place le curseur entre les notes à séparer et on appuie sur la touche [0] du clavier sans cliquer : c'est un silence de la valeur de la note sélectionnée qui est affiché. On quitte le mode INSERT en le sélectionnant à nouveau ou par un clic-droit.

Entrée pas à pas avec un clavier MIDI

On fait un clic-droit sur la touche INSERT qui affiche alors le mot STEP précédé d'une coche (ou touches [CONTROL][INSERT]). La partition s'affiche à partir de la dernière note de la pattern (donc au début si elle ne contient que des silences), on entre les notes ou les accords pas à pas (lentement de préférence) d'une main grâce au clavier de l'instrument, alors que l'autre main sélectionne les durées sur l'écran de l'Atari soit à l'aide de la souris, soit à l'aide des touches du clavier de l'ordinateur. La touche [BACKSPACE] supprime la dernière note ou le dernier accord entré. La touche [0] insère un silence de la durée globale d'une mesure, les barres de séparation se mettent un peu n'importe où mais tout rentre dans l'ordre lorsqu'on sort du mode STEP et on obtient éventuellement des notes liées d'une mesure à l'autre.

Correction de notes

La correction d'une note de la partition permet d'agir sur trois paramètres : la hauteur de la note, sa durée et sa vélocité.

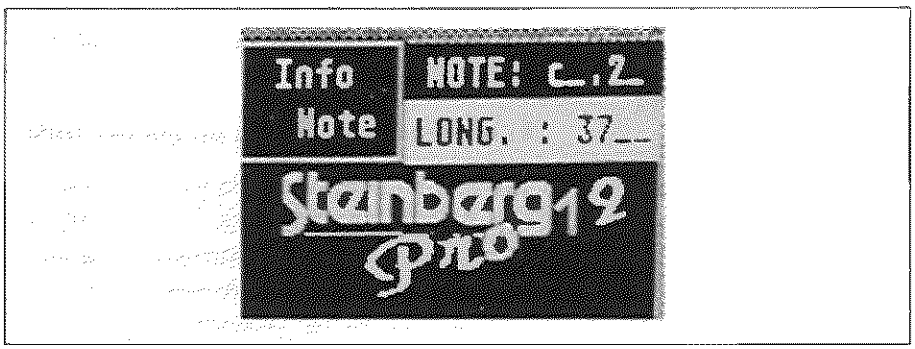
Sélection de la note

On sélectionne la note en cliquant une fois dessus ; elle apparaît en inversion vidéo à l'intérieur d'un petit rectangle noir et elle est jouée. On peut passer d'une note à l'autre en utilisant les flèches haute et basse du clavier de l'ordinateur. La touche [Clr Home] amène le rectangle de sélection alternativement sur la dernière ou la première note de la pattern.

Modification de la note

Par un clic-droit ou un clic-gauche sur la case NOTE de la fenêtre INFO NOTE on modifie sa hauteur par pas d'un demi-ton : le rectangle de sélection se déplace alors à l'écran et on entend simultanément la nouvelle hauteur. La durée est modifiée de la même façon par la case LONG de la même fenêtre (la croche a une durée de 48, la noire de 96, etc.). Ces deux paramètres doivent être figés par le bouton M.A.J. (mise à jour) ou touche [U] *(Update)*. Une autre solution qui n'utilise pas la souris consiste, lorsqu'on a sélectionné une note, à appuyer sur la touche [F2] qui sélectionne l'option Note de INFO NOTE puis à

modifier sa hauteur par les touches [+] et [-]. Le même procédé s'applique à la longueur qu'on choisit par la touche [F3].

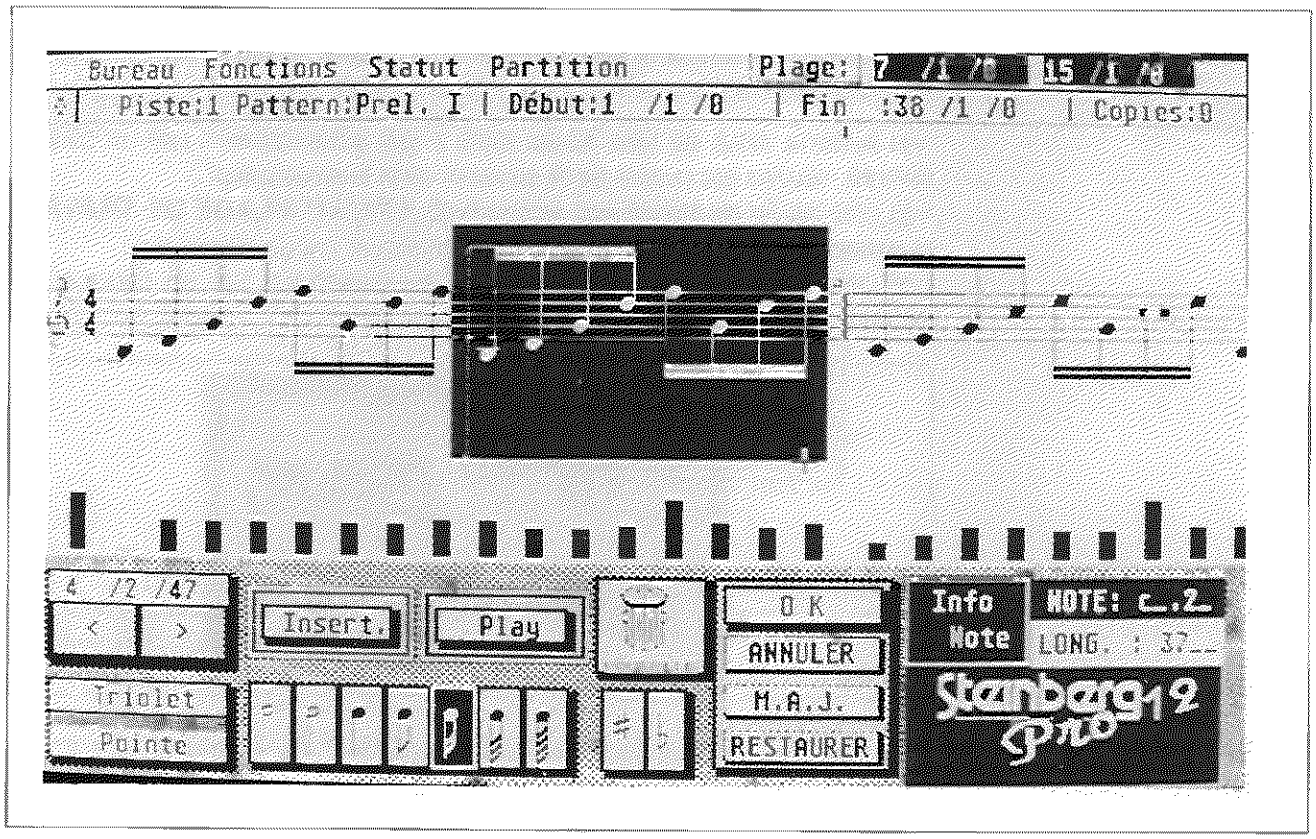


Modification de la vélocité

Les rectangles noirs situés sous chaque note symbolisent la vélocité avec laquelle cette dernière est produite. Lorsque la note est sélectionnée on peut accroître ou diminuer cette vélocité en faisant un cliqué-maintenu sur le rectangle, puis en le tirant vers le haut ou vers le bas. On peut ainsi donner à chaque note d'un accord une vélocité différente en les sélectionnant les unes après les autres.

Effacement de notes

La POUBELLE efface une note ou une série de notes. On sélectionne d'abord la note ou la série de notes en l'enfermant dans un rectangle de sélection obtenu par un cliqué-tiré ; on fait ensuite un cliqué-maintenu sur le rectangle qu'on tire sur la poubelle.



LES MENUS DE L'ÉCRAN PARTITION

Fonctions
Quantific. globale
Quantifier Note on/off
Remplir
Transposer
Déterminer plage de travail

REMPLIR

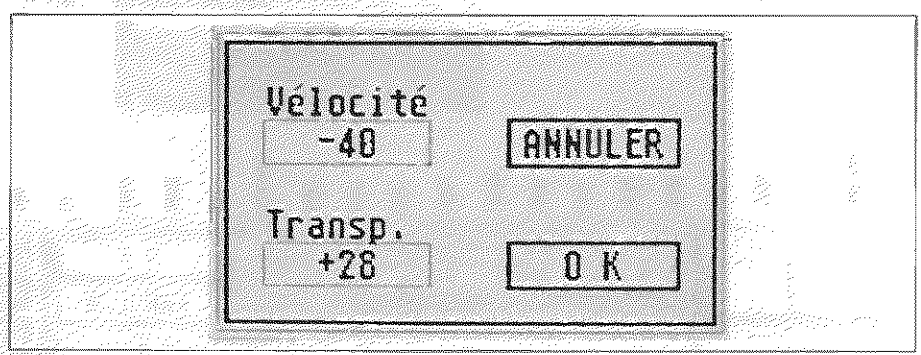
Remplit toute la pattern ou seulement la plage sélectionnée (voir **STATUT/PLAGE DE TRAVAIL** ci-dessous) avec la note MI dont la durée est définie par le sélecteur de durées en bas d'écran ; c'est très pratique pour créer rapidement une ligne rythmique qui ne nécessite ensuite que quelques modifications. La hauteur de cette note peut ensuite être ajustée par l'option **TRANSPOSER**.

TRANSPOSER

Modifie soit la hauteur des notes par demi-ton, soit leur vélocité qui apparaît sous forme d'une barre noire dans la partie inférieure de la

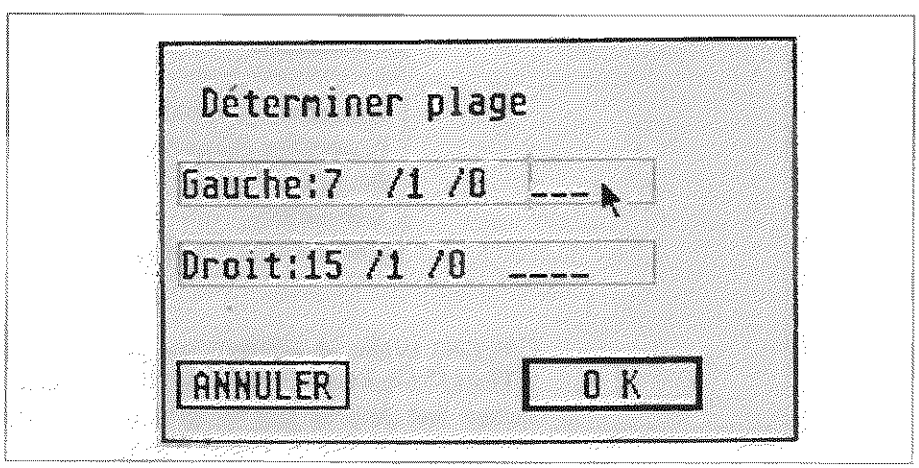
zone des portées ; la variation s'effectue positivement ou négativement relativement à la valeur de la note. Cette fonction agit différemment suivant qu'une sélection a été effectuée ou non :

– si aucune note ou une seule note a été sélectionnée par un cliqué simple, toutes les notes de la pattern sont transposées ;
– si une plage de travail est active (voir STATUT/PLAGE DE TRAVAIL ci-dessous), seules les notes de cette zone sont concernées ;
– si une ou plusieurs notes ont été sélectionnées par un cliqué-tiré, seules ces notes sont concernées.



DETERMINER PLAGE DE TRAVAIL

Fait apparaître deux locators qui permettent de localiser une zone sur laquelle les paramètres de remplissage et de transposition seront actifs ; la saisie s'effectue par [ESC] suivi de l'entrée d'une nouvelle valeur, les flèches haute et basse passent d'un locator à l'autre.



Une autre solution consiste à saisir les notes qui doivent faire partie de la zone de travail en les sélectionnant par un cliqué-tiré puis en activant ce même menu qui n'affiche alors plus les locators.



C'est le menu STATUT/PLAGE DE TRAVAIL qui rendra par la suite cette zone effectivement active.

Statut
Bip Plage de travail Toutes les pistes

BIP

Fait entendre le métronome dans le haut-parleur du moniteur de l'Atari lorsque le bouton **PLAY** est activé (cette option n'est jamais précédée d'une coche indiquant qu'elle est active).

PLAGE DE TRAVAIL

N'active que les mesures sélectionnées depuis le menu FONCTION/PLAGE DE TRAVAIL. Si on coche cette option, les valeurs de délimitation de la plage de travail apparaissent en inversion vidéo dans l'angle supérieur droit de l'écran.



En mode **PLAY**, la musique boucle alors entre ces deux valeurs. Les fonctions de quantification et de remplissage ne concernent également plus que cette zone. Les touches [(] et [)] permettent d'afficher la partition à partir du début ou après la fin de la plage de travail.

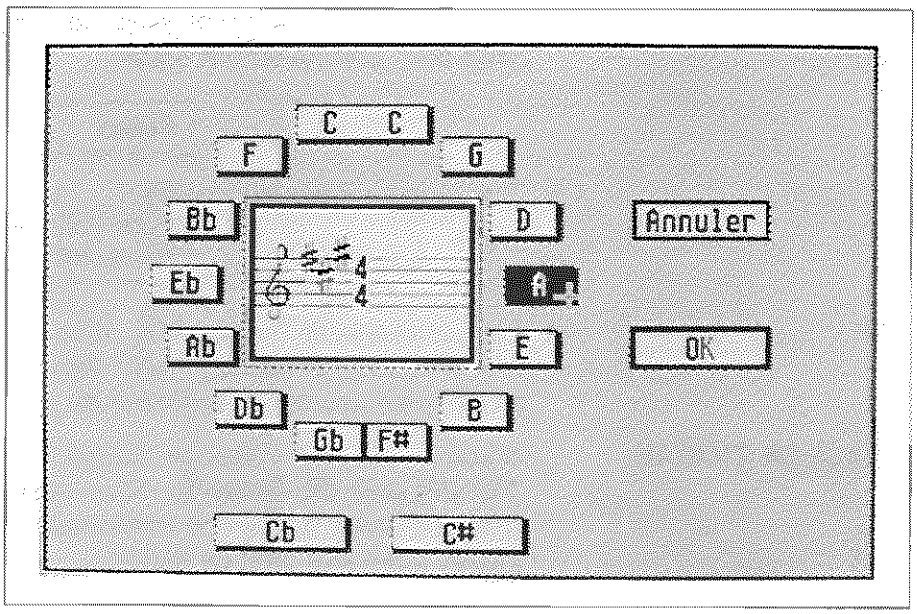
TOUTES LES PISTES

Permet d'entendre en mode **PLAY** ce qui se trouve au même moment sur les autres pistes.

Partition
Changer armature Changer clef Deux portées Quantif. Pause : 1/16 Quantif. Note : 1/16 Pas de recouvrement

CHANGER ARMATURE

Propose le choix de l'armure du morceau. Toute note ajoutée par la suite se conformera à l'armure courante ; par contre, les notes déjà placées sur la portée avant le changement d'armure conservent leur hauteur d'origine.

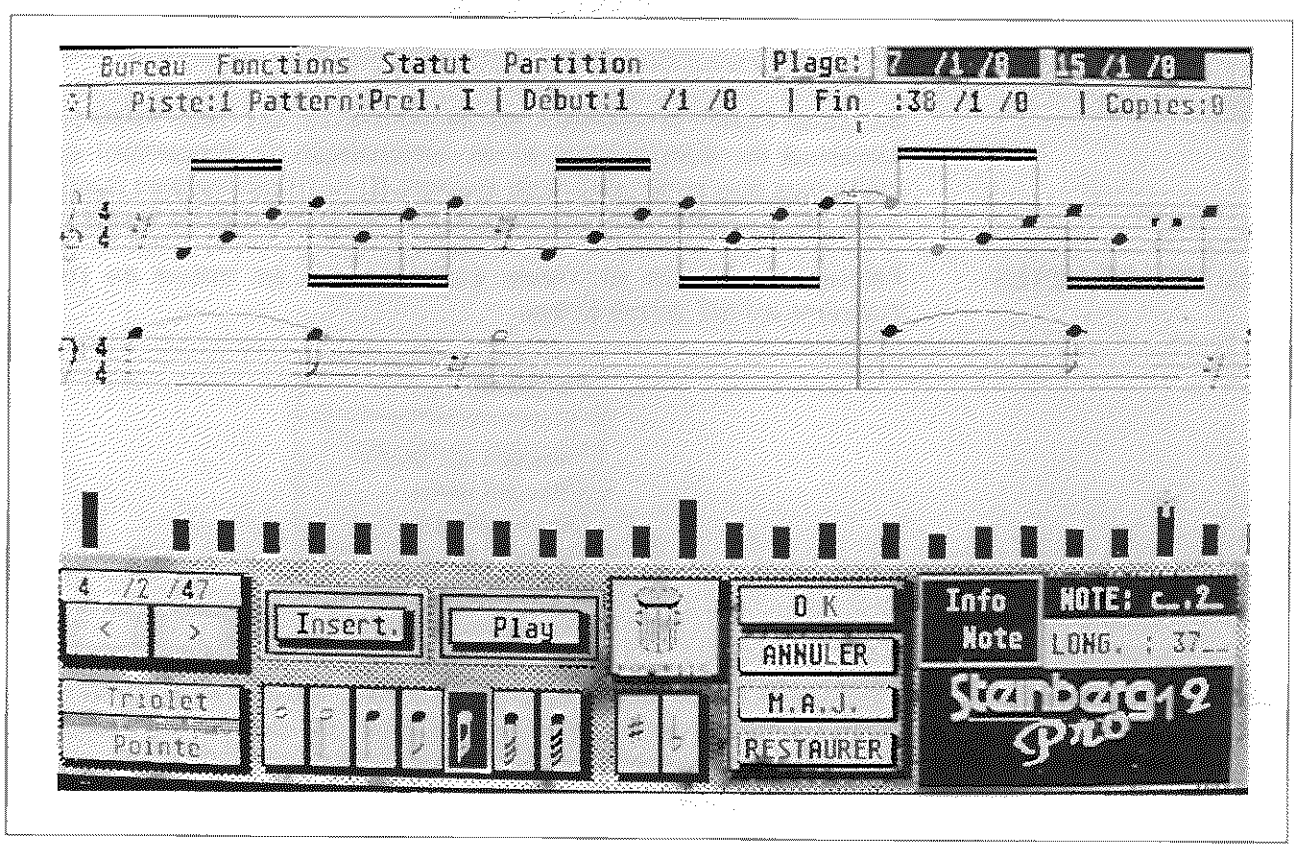


CHANGER CLEF

Fait passer de la clé de SOL à la clé de FA et inversement.

DEUX PORTEES

Affiche simultanément une portée en clé de sol commençant au DO 3 et une portée en clé de fa.



Si on insère des notes alors que les deux portées sont affichées, celle qui n'est pas utilisée passe en inversion vidéo.

QUANTIF. PAUSE
QUANTIF. NOTE

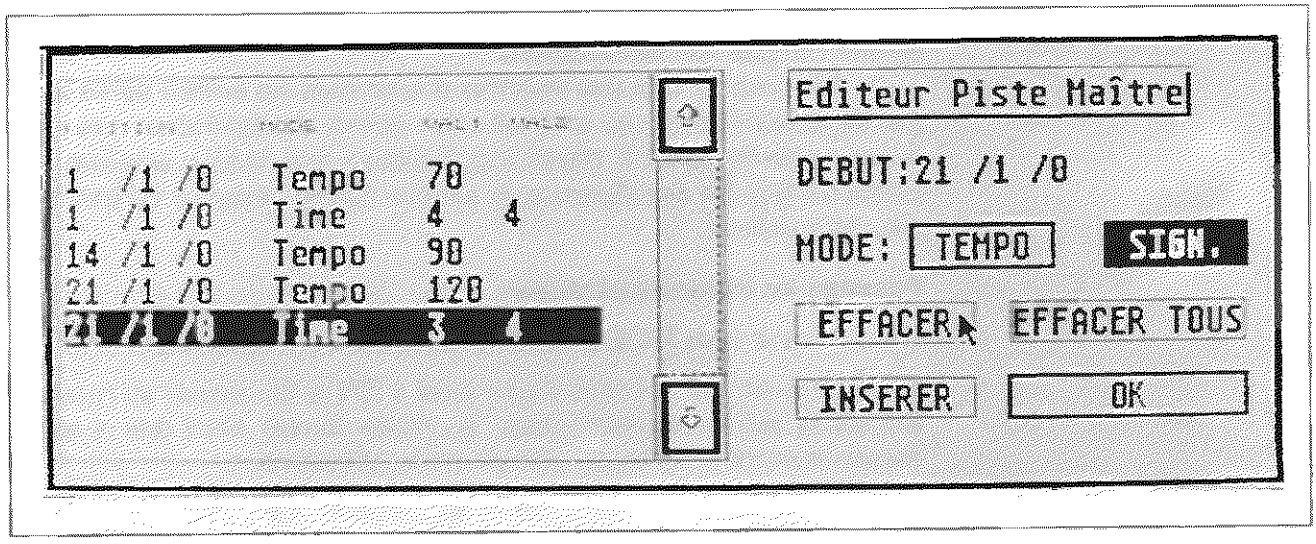
Affiche le morceau avec la résolution demandée (mais ne change pas la quantification des notes déjà saisies). Pour modifier la valeur de quantification, il faut déjà sélectionner une note dans le sélecteur de durées (par exemple une croche) puis activer cette option. Il faut ajuster cette valeur à la plus courte durée de note que l'on souhaite afficher. Le choix d'une mauvaise valeur de quantification produit des affichages étranges, et génère notamment des accords qu'on n'a pas créés ; il suffit généralement de modifier cette valeur pour retrouver une partition correcte sans rien modifier d'autre. La résolution de quantification ne peut être inférieure à la durée d'une noire.

PAS DE RECOUVREMENT

Evite de faire apparaître sous forme de notes liées les petits retards de lâcher lorsque la saisie s'est faite à l'aide d'un instrument.

LA PISTE MAÎTRE

Cette treizième piste n'est reliée à aucun instrument ; elle n'enregistre que les changements de signature rythmique et les tempi au cours d'un morceau. On accède à son éditeur par le menu PISTE/PISTE MAITRE de l'écran principal.



La liste des changements apparaît dans une fenêtre à quatre colonnes :

– POSITION du changement ;
– MODE : il s'agit du type de changement qui ne peut être que TEMPO ou TIME (signature) ;

– VAL 1 et VAL 2 contiennent le nombre de pulsations ou les valeurs de la signature. On peut modifier ces nombres par un clic-droit ou un clic-gauche directement dessus.

Insérer un nouvel événement

On clique sur la valeur de mesure/temps/fraction de temps placée à droite de DEBUT puis on entre la position du nouvel événement. On définit le type d'événement en cliquant sur la case TEMPO ou la case SIGN à droite de MODE. On confirme par la case INSERER.

Effacer

La case EFFACER supprime de la liste l'événement courant (en inversion vidéo). Le premier événement de chaque type ne peut bien sûr pas être supprimé.

La case EFFACER TOUS supprime tous les événements du type défini à droite de la case MODE à condition que leur position soit différente de 1/1/0.

Le mode séquenceur

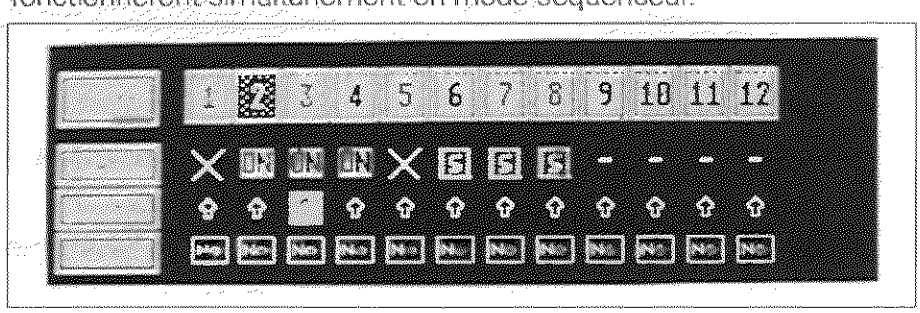
Ce nouveau mode utilise le principe de fonctionnement des séquenceurs traditionnels qui présente quelques contraintes :

– il faut réaliser les patterns en mode bande puisque le séquenceur ne permet pas l'enregistrement ;
– on doit créer la liste des patterns à enchaîner en évitant de se tromper dans leurs noms qui sont les seuls points de repères ;
– les patterns doivent toutes commencer à la même position que celles de la piste de référence et avoir la même longueur ou être inférieures à celle-ci.

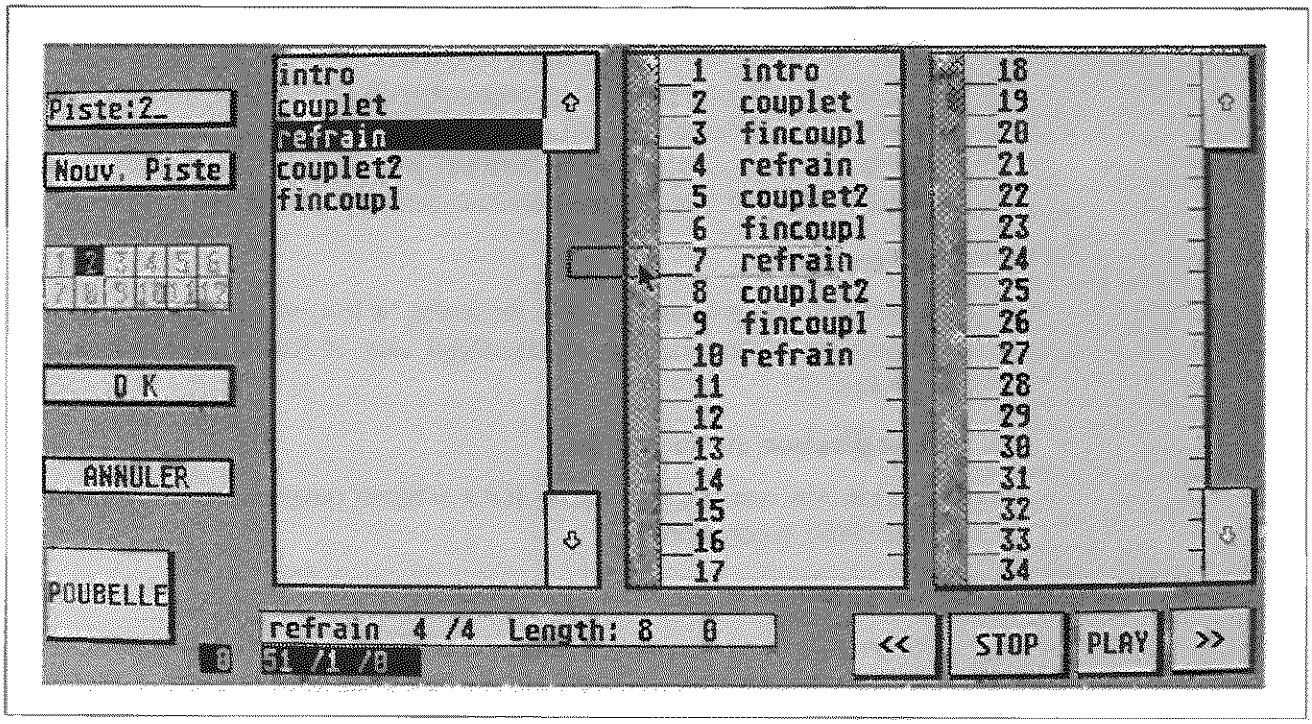
Passage en mode séquenceur

Un clic sur la case MODE la fait passer de BANDE à SEQ.

Un double-clic sur le statut d'une piste à ON la fait passer au mode S. Certaines pistes peuvent rester en mode bande alors que d'autres fonctionneront simultanément en mode séquenceur.



Le menu PISTE/ARRANGER LA SONG ouvre une fenêtre qui contient trois listes à l'intérieur desquelles l'arrangement va être créé.



PISTE suivi d'un numéro indique quelle est la piste de référence : ce sont les patterns de cette piste qui serviront de points de repère. On peut entrer une autre valeur sans la valider par [RETURN] ; la validation se fait en cliquant sur la case NOUV. PISTE. Après confirmation toutes les patterns contenues sur la piste apparaissent dans la première colonne ainsi que dans la colonne suivante puisqu'elles se trouvent déjà sur la piste de référence.

Pistes pouvant être liées

Dans le tableau des numéros de pistes situé sous la case NOUV PISTE, les nombres en grisé signifient qu'il s'agit d'une piste en mode bande ou vide alors que les nombres en noir présentent les pistes en mode séquenceur qui sont (peut être) utilisables.

Pour cela, il faut en effet que les patterns qu'elles contiennent commencent à la même mesure que celles de la piste de référence et ne soient pas trop longues. On le vérifie en cliquant successivement sur chacune des patterns de la première colonne : tous les numéros qui passent à chaque fois en inversion vidéo satisfont cette condition.

Création de la séquence

On place dans les colonnes 2 et 3 les patterns dans l'ordre où on veut les écouter : il suffit de les cliquer-tirer et de les placer à la suite de la

liste qui affiche 34 pas à l'écran, mais qui s'étend sur 64 pas si on utilise les ascenseurs.

Insertion de patterns

On peut poser une pattern sur une autre qu'elle écrase, à moins que le pointeur de la souris ne soit à ce moment là situé sur la bande grisée de la colonne auquel cas la pattern est insérée avant celle qu'elle devait recouvrir : toutes les patterns sont alors décalées.

Effacement

L'effacement s'effectue *via* la poubelle, le trou ainsi créé est automatiquement rebouché par un décalage vers le haut des patterns suivantes.

Localisation sur la portée

Un clic sur une pattern placée fait s'afficher dans une fenêtre sa position dans la partition.

Compteur

Si on joue l'arrangement à l'aide du bouton disponible dans la fenêtre, le compteur en bas d'écran affiche le numéro de la pattern en cours ainsi que les habituelles indications sur la mesure. On peut activer le déroulement directement à partir de la pattern que l'on souhaite écouter en la faisant glisser sur le bouton PLAY ce qui permet de se dispenser d'utiliser les touches de rembobinage.

Sortie et validation

La fenêtre étant globalement confirmée par OK, on peut rejouer l'arrangement depuis l'écran principal. ANNULER perd l'arrangement.

Le menu fichiier

Il permet les opérations classiques de sauvegarde et de chargement sur les musiques et sur les patterns.

Fichier
Charger pattern Sauver pattern
Charger song Sauver song
Nouveau Quitter



Charger patterns

Si le fichier contient plusieurs patterns, elles sont chargées à partir de la position du locator gauche à raison d'une par piste à partir de la piste sélectionnée pour l'enregistrement et sur les pistes suivantes.

Sauver pattern

La sauvegarde s'effectue depuis le menu de façon globale et assez curieuse :

– les patterns de toutes les pistes à ON sont sauvegardées automatiquement les unes après les autres ;
– elles doivent être comprises entièrement entre les locators ;
– elles ne doivent pas être séparées par un blanc, auquel cas la sauvegarde s'arrête après la pattern suivie par le blanc ;
– les patterns contigües sont regroupées en une seule pattern et ne portent plus qu'un seul nom.

Les fichiers de patterns comportent le suffixe **PAT** sur le disque.

Charger song

Charge un morceau avec tous ses réglages et toutes ses patterns. Au moment de son lancement, PRO12 charge automatiquement le fichier **DEF.SNG** dans lequel on a intérêt à stocker les paramètres par défaut des instruments que l'on utilise (canal, initialisations et filtres MIDI,...) ainsi que les éléments d'usage courant (position des locators dans les touches de fonction, diverses patterns de rythmique ou de basse,...).

Sauver song

Le morceau est sauvegardé avec toutes ses patterns, ses définitions et filtres MIDI, sa piste maître, le carnet de notes,... de sorte qu'au chargement on retrouve la configuration de travail dont on disposait lors de la sauvegarde.

Les songs sont sauvegardées avec le suffixe **SNG** sur le disque.

Nouveau

Ce menu efface tout ce qui se trouve sur les pistes mais ne réinitialise pas les définitions globales qui conservent leurs valeurs.

QUELQUES PROBLEMES COURANTS

1/ Plusieurs voix ont été programmées mais le synthétiseur joue toutes les pistes avec un seul son d'instrument.

Seul un synthé multi-timbral est capable de reproduire plusieurs sons d'instruments différents simultanément ; encore faut-il que les voix soient correctement affectées aux différents canaux MIDI au moment du partage du clavier.

2/ Certaines notes disparaissent lors de la reproduction. Cela peut arriver si on utilise plusieurs instruments en série, il faut dans ce cas se procurer un répartiteur MIDI. Si on utilise un seul instrument piloté par plusieurs pistes sur le même canal, il faut vérifier dans sa notice le nombre de notes qu'il est capable de jouer simultanément : si c'est quatre notes, il est normal que la cinquième et celles qui arrivent au-delà ne soient pas prises en compte.

3/ Les modifications de vélocité ne sont pas perceptibles. Il faut que le synthé soit capable de gérer ce paramètre ; on s'en aperçoit au fait que l'intensité sonore de la note produite varie en fonction de la force avec laquelle la touche du clavier est frappée. En outre, n'y a t-il pas quelque part un filtre qui bloque ce contrôleur ?

4/ Les modifications de volume ne sont pas perceptibles. Le contrôleur est-il implémenté sur le synthé (MIDI Volume ou contrôleur 7) ? Si c'est le cas, il faut vérifier que le canal MIDI ne soit pas à No sur l'écran de l'ordinateur et que le filtre correspondant ne soit pas à OFF.

5/ Que choisir, le mode bande ou le mode séquenceur ?

Le mode bande est incontournable pour réaliser un enregistrement de pattern. Quant à la reproduction, il est plus simple d'organiser les patterns depuis le séquenceur que depuis le mode bande où on travaille par copies sans visualiser d'un seul coup d'œil la structure globale du morceau. Le travail par copies de mode bande est cependant plus souple car on peut modifier par la suite les éléments dupliqués grâce à l'éditeur de partition : le séquenceur, lui, enchaîne toujours les mêmes éléments.

6/ Comment trouver les notes correspondant aux instruments d'une boîte à rythme ?

Si on connaît le canal permettant d'activer la boîte, il suffit d'entrer dans l'éditeur de partition, de faire varier une note depuis INFO NOTE (en clé de fa) et de noter le nom de la note chaque fois qu'un son de percussion différent se fait entendre ; par exemple C0 pour la grosse caisse, D0 pour la caisse claire, etc.

7/ Comment créer rapidement une ligne de percussions à partir d'une unique pattern ?

La duplication est assez rapide si on suit la méthode suivante : on sélectionne la pattern dans la fenêtre des informations sur la piste, on

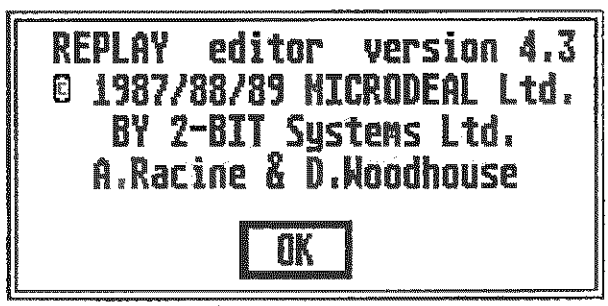


tire son numéro de fin sur la case DEST (sous les locators), on tire le nom PATT sur le numéro de la piste concernée. On sélectionne la nouvelle pattern qu'on vient de copier en cliquant sur son numéro à côté du mot PATT puis ont tire son numéro de fin sur la case DEST, etc.

6/4.3

ST Replay 4, logiciel d'échantillonnage sonore

Au même titre que DEGAS qui constitue la référence en matière de dessin point par point, du moins en ce qui concerne les formats des fichiers images, ST Replay 4 se doit de figurer dans la boîte à outils logicielle de l'atariste qui souhaite pouvoir manipuler le son avec la même facilité et la même précision que l'image dont il est l'essentiel complément.



Survol

Un échantillonneur en temps réel

Ce logiciel Microdeal distribué en France par Guillemot International permet de numériser sur une voix une séquence sonore depuis n'importe quelle source susceptible d'être connectée sur un amplificateur : magnétophone à cassette, lecteur de CD, microphone par l'intermédiaire d'un préamplificateur...

La qualité de restitution est, en fréquence, proche de celle obtenue depuis un disque compact à condition que les meilleures options

d'échantillonnage soient choisies ; le rapport signal bruit ne dépasse cependant pas 48 db. Le travail en temps réel permet de pouvoir contrôler en permanence la qualité de l'échantillonnage car le son numérisé est également disponible en sortie au fur et à mesure de la prise de son.

Un mini-studio de montage numérique

La séquence mémorisée peut être travaillée sur l'écran avec une précision chirurgicale (au centième de seconde près) qui relègue au rang d'antiquités bande magnétique, ciseaux et adhésifs chers aux amateurs de montages sonores.

Un outil de création

Les possibilités MIDI sont pour une grande part utilisables même si on ne connecte pas d'instrument à l'ordinateur. Elles autorisent des effets sonores spectaculaires : modification de fréquence qui rend une voix méconnaissable mais compréhensible (pitch), mémorisation et rappel rapide de jingles (affectation des séquences sonores à des touches), mémorisation et déformation de séquences provenant des sources les plus diverses (chansons, bulletins d'information, discours des hommes politiques...), créations de sons nouveaux destinés aux claviers MIDI, l'ordinateur étant utilisé en expandeur. Le résultat est ensuite archivé sur disquette ou enregistré analogiquement sur une bande magnétique.

Le logiciel est fourni avec une cartouche à brancher sur le port d'extension de l'Atari. Cette cartouche numérise les sons qu'on envoie à l'ordinateur ; on a également la possibilité d'y brancher un amplificateur en sortie afin d'obtenir une meilleure restitution sonore. La cartouche n'est d'ailleurs pas indispensable si on se contente d'écouter sur le moniteur ou de modifier des séquences déjà enregistrées sur disquette.

Un outil pour le développeur

La disquette contient à titre d'exemple des routines dans plusieurs version de l'interpréteur Basic (Basic Atari, GFA, STOS et HISOFT qui a servi à écrire une grande partie du logiciel ST Replay). Ainsi l'utilisateur peut intégrer les numérisations qu'il a réalisées à ses propres programmes ; ceux-ci pourront ensuite être utilisés sans que la cartouche ne soit connectée à l'ordinateur ce qui autorise leur distribution. Nous verrons d'ailleurs qu'il est souvent très facile de modifier les séquences sonores de certains des programmes du commerce car le format de fichier de ST Replay 4 est couramment utilisé par les concepteurs de logiciels.



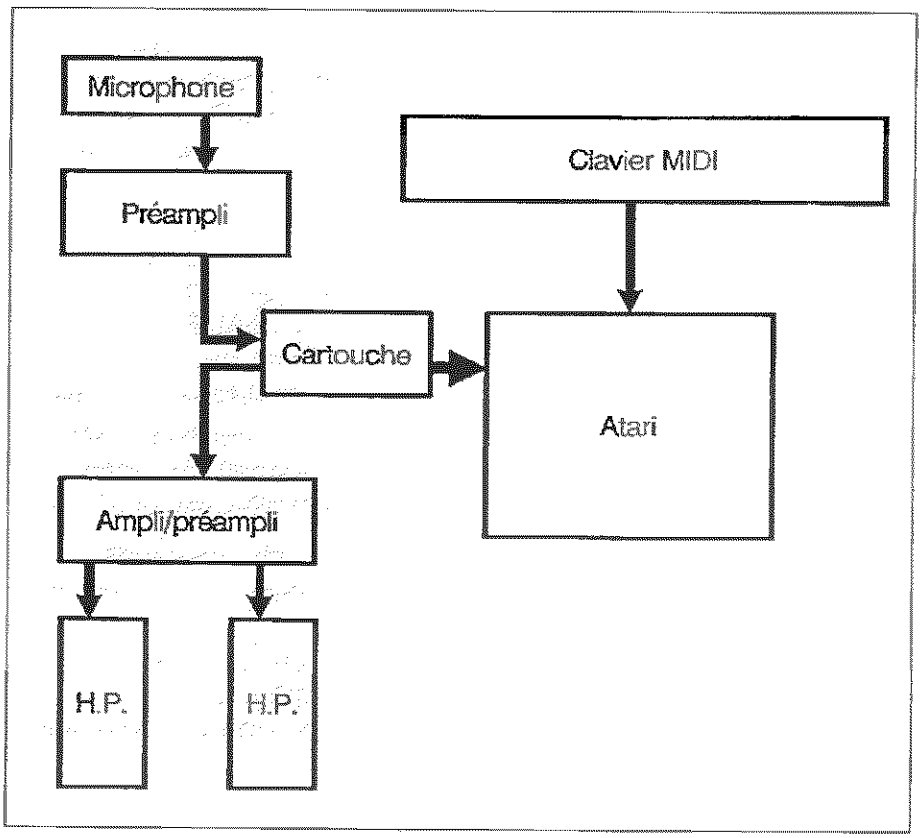
Les deux disquettes fournies contiennent de nombreux programmes destinés à montrer les multiples possiblités de la cartouche. On y trouve en plus de ST Replay 4 les fichiers suivants :

– la version ST Replay 3 qui avec ses 15 kilo-octets d'occupation mémoire et son interface utilisateur simplifiée laisse un maximum de place pour les échantillons sonores ;
– le programme EFFECTS qui produit des effets de réverbération et d'écho en temps réel à une fréquence d'échantillonnage malheureusement trop faible ;
– un répertoire Basic qui renferme les routines pour diverses versions de l'interpréteur Basic et qui permet l'utilisation des fichiers numérisés depuis les programmes utilisateur ;
– un fichier source EXAMPLE.S qui décrit une partie de la réalisation de ST Replay en Basic HISOFT ;
– le programme DRUMBEAT qui utilise des échantillons de batterie à 20 kHz fournis sur la disquette. L'ordinateur est transformé en un mini séquenceur, le musicien conservant la possibilité d'y intégrer ses propres sons numérisés depuis ST Replay.

La réalisation de tels programmes demande une parfaite connaissance de l'assembleur et du matériel (programmation en entrée du port de sortie de l'Atari : voir à ce propos le principe succinctement décrit dans FM Melody Maker), incluant la maîtrise de programmation du 68901 qui est utilisé comme source d'interruptions (précision indispensable des temporisations dans le domaine sonore).

EN DÉTAILS

Connexion de la cartouche

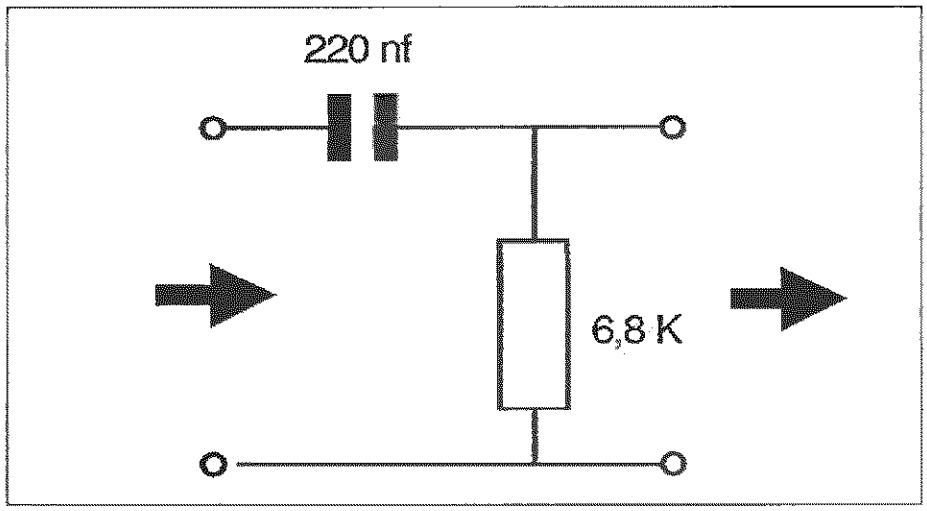


Quelques connections possibles avec ST Replay.

Le clavier MIDI qui figure sur le schéma n'est pas indispensable.

On peut se procurer au besoin un câble en Y qui connecté à l'entrée de la cartouche permet de récupérer simultanément les données des deux canaux d'une source stéréo. Un câble de même type branché en sortie permet d'exciter les deux canaux d'un amplificateur.

Si on travaille à l'aide d'un micro, un filtre passe-haut très simple constitué d'une résistance et d'un condensateur peut efficacement limiter les fréquences parasites à très basse fréquence, normalement inaudibles, mais qui viennent perturber le convertisseur. Voici un schéma très simple de filtre dont la fréquence de coupure se situe au environ de 100 Hz.



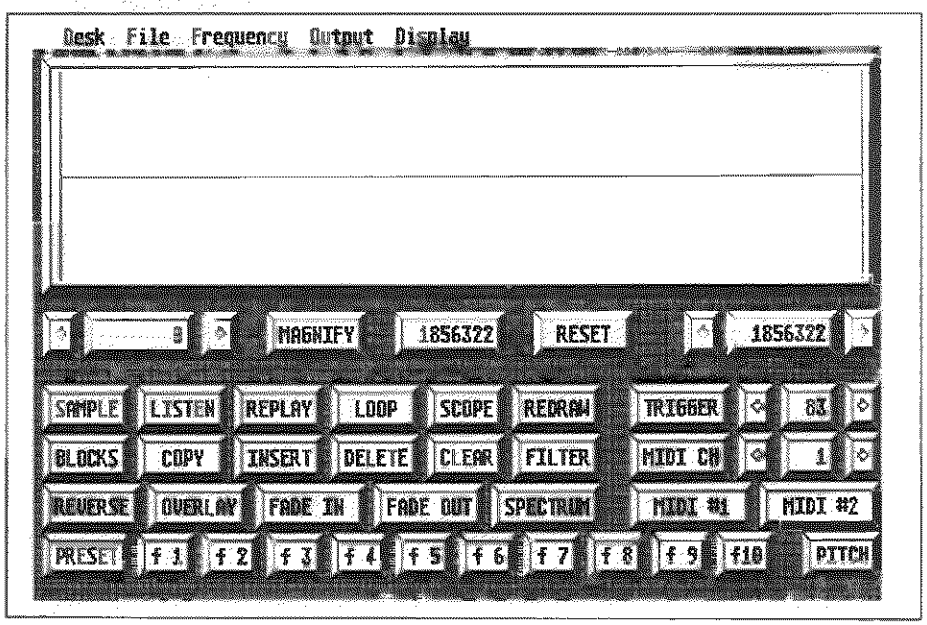
Un filtre passe-haut à placer à l'entrée de la cartouche.

L'écran de travail

Le logiciel fonctionne sur un écran monochrome ou sur un écran couleur en moyenne résolution.

L'écran est divisé en trois parties :

– une ligne de menus déroulants ;
– une fenêtre de visualisation du son qui ne contient à la mise en route qu'une ligne horizontale en son centre figurant une mémoire vierge délimitée par deux barres verticales ;
– un panneau de commandes dont la première ligne sert à définir la zone de travail qui est encadrée par les deux barres décrites ci-dessus.



Choix de la sortie

Output
Monitor HI-FI

Le son doit-il être reproduit par le moniteur de l'Atari (Monitor) ou par un amplificateur branché sur la sortie OUT de la cartouche (HI-FI) ?

Le choix du mode de reproduction définit la gamme des fréquences d'échantillonnage disponibles dans le menu FREQUENCY. Celle-ci s'étend par multiples de 5 et de 7,5 kHz jusqu'à 30 kHz pour la sortie sur moniteur et jusqu'à 50 kHz pour la sortie HI-FI.

Plus cette fréquence est élevée, meilleure est la qualité bien qu'il soit difficile de percevoir une nette différence au-delà de 30 kHz. En contrepartie, la durée de la digitalisation est réduite d'autant.

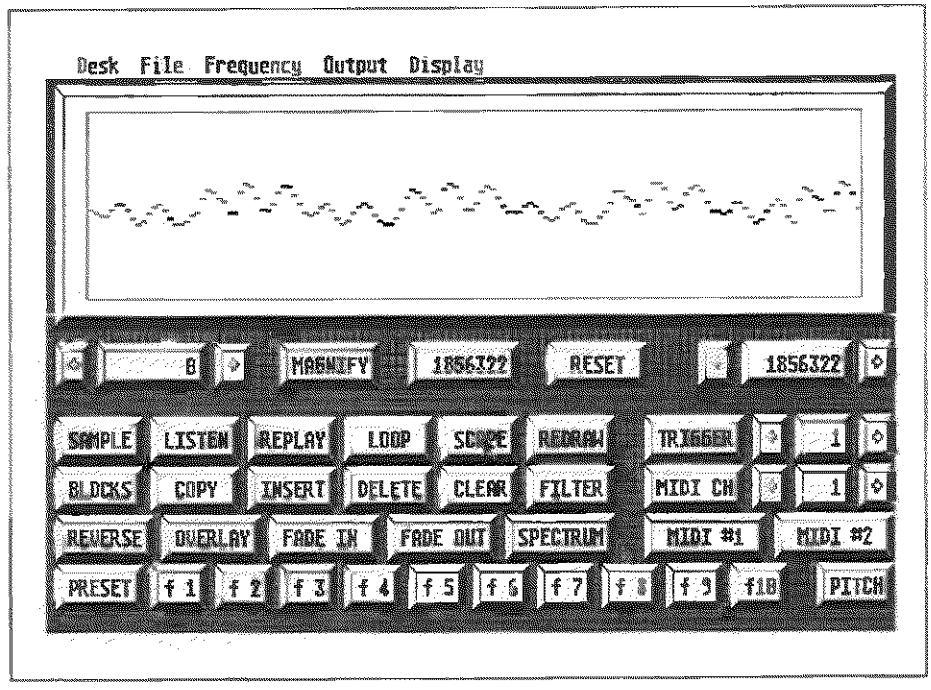
A une fréquence d'échantillonnage de 20 kHz qui commence à donner une reproduction tout à fait correcte de la musique, on peut compter sur 5 s d'enregistrement par portion de 100 Kilo-octets de mémoire disponible (le filtre passe-bas de la cartouche possède une fréquence de coupure situé à 10 kHz et devient efficace à partir d'un échantillonnage à 20 kHz en supprimant l'effet d'aliasage qui produit la distorsion du son et l'effet métallique accompagnant les voix). Ainsi sur un méga ST2 qui après chargement du programme dispose de 1 887 kocets (environ 1,9 Méga-octets sur les 2 d'origine) on peut enregistrer un peu plus d'une minute trente de son (5 x 1 887/100). Si la fréquence d'échantillonnage double, cette durée est bien sûr diminuée de moitié (au pire 40 secondes à 50 kHz sur un méga ST2 ce qui fait tout de même moins de 10 secondes sur un 520 ST).

Le tableau ci-dessous présente quelques chronométrages réalisés avec une mémoire disponible de 1 887 822 octets.

5 kHz	6 mn 25 s
7,5 kHz	4 mn 07 s
10 kHz	3 mn 17 s
15 kHz	2 mn 07 s
20 kHz	1 mn 36 s
30 kHz	1 mn 02 s
40 kHz	46 s
50 kHz	39 s

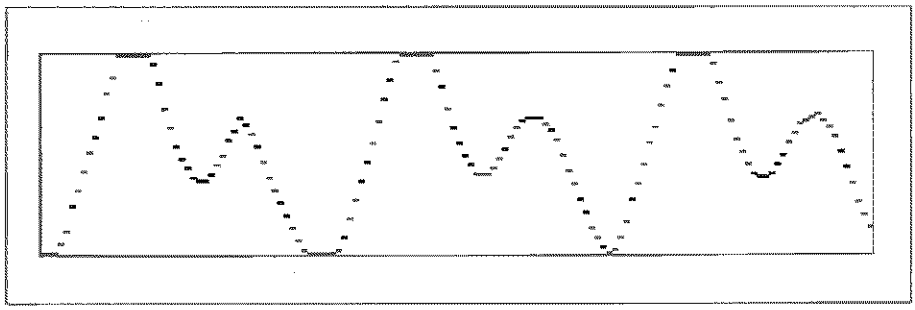
Réglages préalables à l'enregistrement

Après avoir connecté en entrée une source sonore délivrant un signal de moins de 2,5 V (préamplificateur, sortie de magnétocassette ou de lecteur CD...) on peut activer l'option SCOPE afin d'ajuster le niveau du signal injecté, ceci afin éviter toute saturation. SCOPE transforme la zone de visualisation du son en un écran d'oscilloscope dont le faisceau est modulé en temps réel par le signal présent à l'entrée de la cartouche. L'affichage peut être gelé à tout moment par appui sur la touche ESC.



En mode SCOPE, aspect d'un SOL sur une clarinette en Sib.

Bien qu'il ne soit pas audible en sortie, on peut ainsi vérifier que le signal est réellement présent en entrée et d'un niveau suffisant. Si on dispose d'un potentiomètre de réglage sur le préampli, on doit éviter que la courbe ne touche les limites supérieure et inférieure de la zone de visualisation (saturation).



Légère saturation à l'entrée.

On sort du mode SCOPE par la touche ESC puis par appui sur le bouton REDRAW.

Le bouton SPECTRUM fait apparaître un analyseur de spectre dans la zone de visualisation du son. Le spectre sonore analysé s'étend de 100 Hz à un peu plus de 5 kHz, ce qui correspond bien à la totalité des fréquences couvertes par la voix humaine (dont la limite supérieure est plus proche de 3 kHz que de 5). Si on utilise un micro, on voit l'intérêt d'utiliser un pare-vent (mouchoir par exemple) et un pied support qui diminuent tous deux considérablement les effets parasites dans les fréquences les plus graves (la différence est très visible lorsqu'on essaie de siffler dans le micro avec et sans pare-vent).

On sort de SPECTRUM par ESC.

En plus de leur aspect spectaculaire, les options SCOPE et SPECTRUM présentent un intérêt pédagogique évident car elles permettent de multiples expérimentations sur le son : visualiser la hauteur des notes de différents instruments, montrer leur richesse en harmoniques, etc.

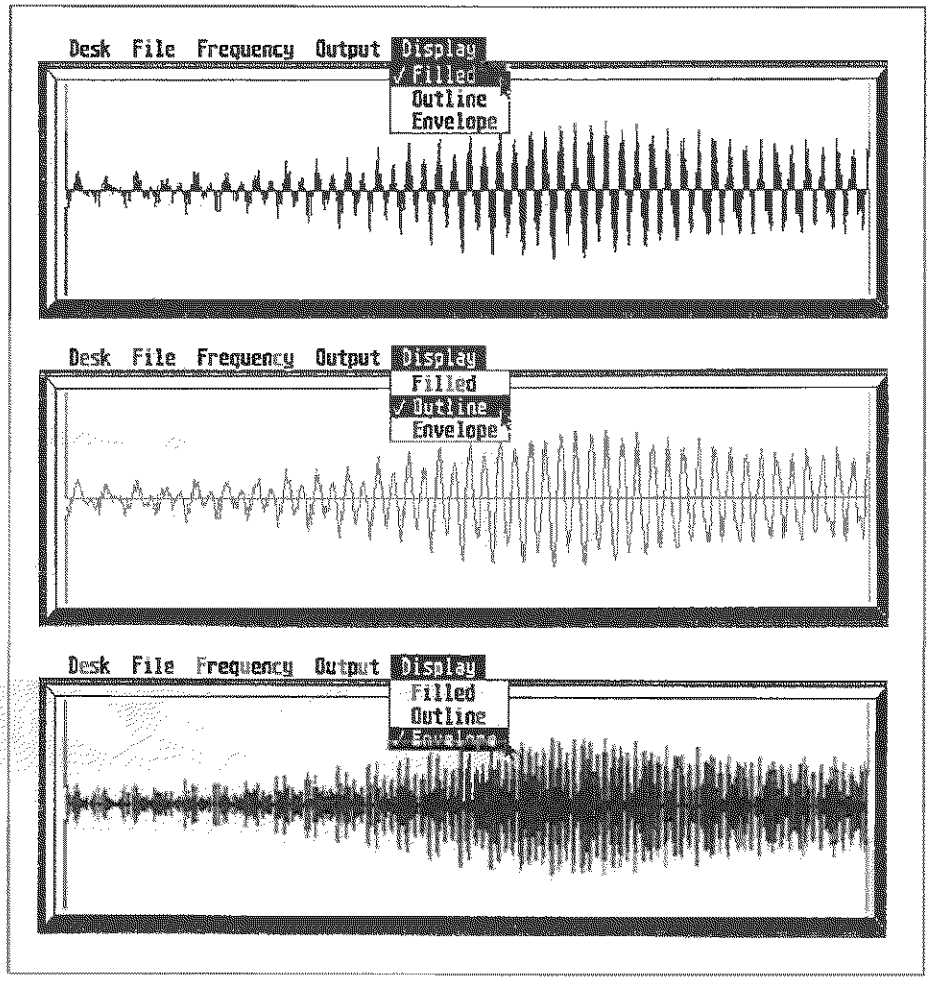
Le bouton LISTEN (ou touche L) lorsqu'il est activé reproduit en sortie le signal présent sur l'entrée avec la qualité propre à la fréquence d'échantillonnage adoptée : on constate qu'à partir de 15 kHz, le filtre passe-bas du logiciel commence à agir ; les fréquences parasites dues à l'échantillonnage sont alors suffisamment élevées en fréquence et atténuées en intensité pour devenir inaudibles : la parole perd cette sonorité métallique gênante aux fréquences d'échantillonnage plus basses.

Mémorisation d'une séquence sonore

On clique SAMPLE (ou touche S) qui passe en inversion vidéo. Le son présent en entrée est reproduit simultanément en sortie comme avec LISTEN (avec un micro attention au larsen). On peut également lancer la mémorisation depuis LISTEN en appuyant sur la touche S (SAMPLE).

L'enregistrement se poursuit jusqu'à ce que la mémoire soit pleine ou que l'utilisateur appuie sur ESC. C'est seulement à ce moment que la courbe sonore apparaît à l'écran. Cette visualisation peut se faire sous trois modes accessibles depuis le menu DISPLAY.

DISPLAY
Filled Outline Enveloppe



Trois aspects d'un même enregistrement.

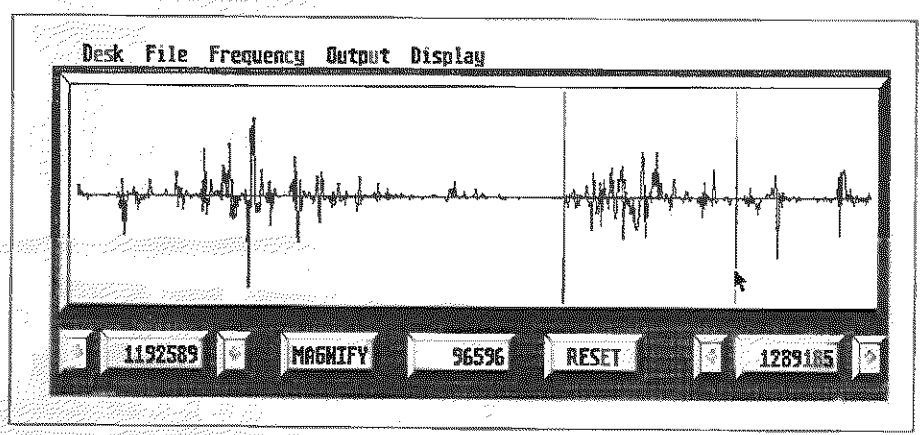
Le TRIGGER déclenche automatiquement l'enregistrement lorsqu'un seuil sonore prédéfini est atteint ; c'est pratique pour éviter d'enregistrer le silence qui précède une chanson sur une cassette ou un disque. La plage de réglage, très sensible, s'étend de 1 à 127. Un seuil de 11 déclenche l'enregistrement pour un murmure dans le micro. Une fois commencé, l'enregistrement se poursuit alors jusqu'à ce que la zone de travail soit remplie, même si aucun signal sonore n'est présent en entrée.

Pour utiliser le TRIGGER, il faut préalablement définir son seuil de déclenchement, activer le bouton TRIGGER qui passe en inversion vidéo puis activer la touche SAMPLE. Le pointeur de la souris se transforme en symbole d'attente (abeille), jusqu'à ce qu'un signal suffisant excite l'entrée ; la touche SAMPLE passe alors en vidéo inverse indiquant que l'enregistrement est commencé.

Reproduction du son mémorisé

La partie que l'on vient de mémoriser se trouve délimitée par les deux barres verticales. Si on a rempli la totalité de la mémoire, ces deux barres se trouvent aux deux extrémités de la zone de visualisation du son.

Elles peuvent être déplacées soit directement par un cliqué-tiré à l'aide de la souris, soit à l'aide des boutons immédiatement situés sous la fenêtre de visualisation.



On peut isoler un mot grâce aux barres de délimitation.

Les deux flèches qui encadrent le compteur de droite permettent de modifier la position de la barre droite : il en est de même pour la barre gauche à l'aide des deux boutons qui encadrent le compteur gauche ; la pression simultanée des deux boutons de la souris accroît la vitesse de déplacement. Au départ le compteur droit indique 0 alors que le compteur gauche affiche la taille mémoire disponible. Le compteur

central indique toujours la taille de l'échantillon situé entre les deux barres (au départ, il est donc égal à la taille mémoire disponible).

Le bouton **MAGNIFY** étend la taille de l'échantillon délimité par les barres de façon à ce qu'il occupe toute la fenêtre de visualisation. Cette possibilité est impressionnante car on peut agrandir ainsi considérablement un passage pour le retravailler avec une précision maximale comme nous le verrons dans la partie traitant des modifications de l'échantillon. La limite d'agrandissement est cependant limitée à 600 octets pour toute la largeur de l'écran (ce qui représente environ 3/100e de secondes à 20 kHz).

Le bouton **RESET** replace les deux barres aux extrémités de la fenêtre si l'option **MAGNIFY** n'a pas été sélectionnée auparavant. Sinon, **RESET** restitue au premier appui la vue de la totalité de la zone mémoire disponible (effet de zoom arrière) et au deuxième appui replace les barres aux extrémités de la fenêtre.

La touche **REPLAY** (ou touche **R**) relit l'échantillon situé entre les barres. Si on appuie sur la barre d'espace avant que l'échantillon ne soit entièrement reproduit, on le fait repartir depuis le début, ce qui permet de créer ces fameux effets de bégaiement chers aux rappeurs.

La touche **LOOP** permet de boucler sur l'échantillon en mode **REPLAY** et ce tant qu'on n'appuie pas sur la touche **ESC**.

Manipulations sur le son en mémoire

Ces manipulations n'affectent que la partie de l'échantillon située entre les barres délimitatrices. On a donc la possibilité d'appliquer des traitements variés aux différentes parties d'un même échantillon. Il suffit d'encadrer à chaque fois par les barres la zone concernée.

SAMPLE réalise un enregistrement comme nous l'avons vu, précédemment, mais en le plaçant dans la zone délimitée : l'enregistrement s'arrête automatiquement lorsque le pointeur mémoire arrive sur la deuxième barre délimitatrice.

CLEAR remet à zéro la partie située entre les barres. Pour tout remettre à zéro, il faut donc ramener les barres aux extrémités de la fenêtre avant de choisir **CLEAR**. La remise à zéro peut prendre plusieurs secondes si la taille de la zone à réinitialiser est importante car chaque octet est effectivement mis à zéro en mémoire.

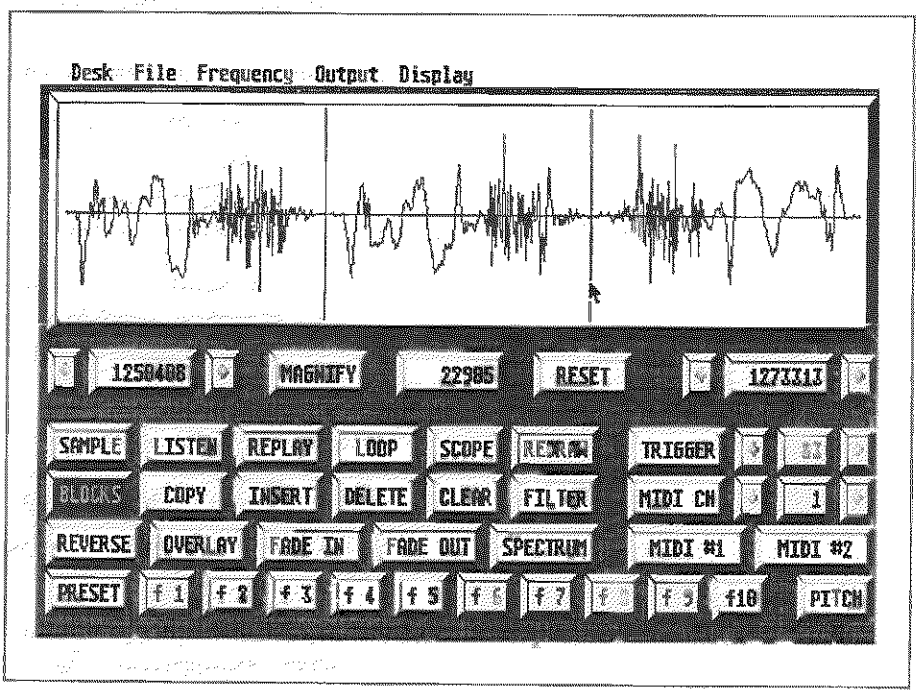
DELETE supprime la zone encadrée : c'est comme si on enlevait un morceau de bande magnétique. Toute la partie située à droite de l'échantillon supprimé est donc décalée vers la gauche et un espace libre, de la taille de l'échantillon supprimé, apparaît en haut de mémoire ce qui demande un certain temps.

INSERT duplique la zone située entre les barres à la droite de cette dernière et sans écraser ce qui suit. La partie située à droite de l'échantillon étant repoussée en haut de mémoire, la partie excédentaire en fin de mémoire est perdue. Après avoir dupliqué une partie d'échantillon de cette manière, on peut l'effacer avec **CLEAR** pour enregistrer sur la zone libre ainsi insérée.

COPY doit être utilisé avec **BLOCK** qui mémorise d'abord la position des barres délimitatrices. Lorsque **BLOCK** est sélectionné (inversion vidéo), les barres restent à distance constante lorsqu'on les déplace. On peut donc les amener au point où on souhaite dupliquer le bloc, ce qui est réalisé par **COPY**. La partie ainsi dupliquée écrase la partie recouverte.

OVERLAY fonctionne de la même façon que **COPY** mais sans effacer la partie recouverte. Les deux parties se mélangent, leurs niveaux sonores respectifs étant diminués de moitié. Il est de la sorte facile de créer un effet d'écho en recopiant tout en déplaçant légèrement une zone sur une autre.

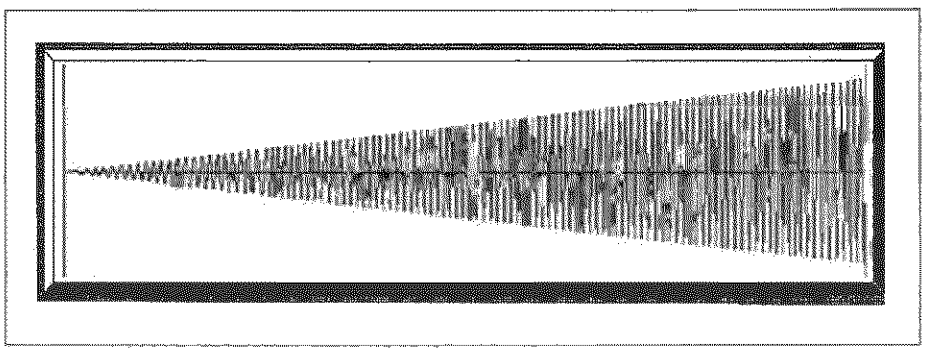
REVERSE inverse la partie de l'échantillon délimité, comme si on écoutait une bande à l'envers. On peut effectuer **REVERSE** plusieurs fois de suite sans que l'échantillon ne subisse la moindre altération. **REVERSE** ne prend pas plus de temps à s'effectuer que **CLEAR**.



Un même mot dupliqué deux fois puis inversé.

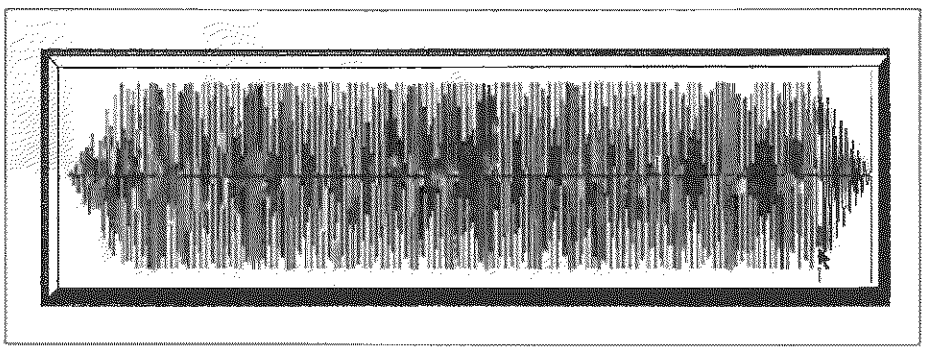
FILTER modifie l'échantillon en mémoire de façon définitive en atténuant les fréquences supérieures à la moitié de la fréquence d'échantillonnage (filtre passe-bas logiciel). Si l'enregistrement a été échantillonné à 5 kHz, les fréquences supérieures à 2,5 kHz sont supprimées : l'effet métallique provoqué par les harmoniques élevées de la fréquence d'échantillonnage est donc fortement atténué, il en est de même des fréquences aiguës de l'enregistrement. Ce filtrage n'est utile que pour les basses fréquences d'échantillonnage car le logiciel active automatiquement un filtre passe-bas pour les fréquences élevées.

FADE IN et FADE OUT modifient l'échantillon en mémoire de façon définitive, le premier en créant un effet d'augmentation du volume qui part de zéro, le second en provoquant une atténuation progressive de celui-ci. La rapidité de croissance ou d'atténuation progressive dépend de la largeur de l'échantillon sélectionné. Si on applique FADE IN sur la totalité de la mémoire, le son va croître régulièrement du délimiteur gauche au délimiteur droit et l'effet obtenu risque d'être décevant (bruit de fond important par rapport au signal).



FADE IN appliqué à la totalité d'un enregistrement (ici sur un larsen).

Il est préférable de n'appliquer l'atténuation que sur quelques secondes au début ou à la fin de l'échantillon.



FADE IN et FADE OUT de quelques secondes en début et fin d'enregistrement.

Touches de fonction

On peut attribuer une séquence sonore particulière à chacune des touche de fonction. Pour cela on sélectionne déjà la zone à mémoriser, on active **PRESET** puis la touche de fonction que l'on veut affecter. On donne ensuite un nom de huit caractères maximum à la séquence.

Si on souhaite retrouver et afficher une séquence déjà définie, il suffit d'appuyer sur la touche de fonction correspondante.

Attention :

Lors de l'affectation d'une touche de fonction, seule la position des barres délimitatrices est mémorisée ; si on insère ou supprime des éléments situés avant les blocs, leur contenu s'en trouvent donc décalés.

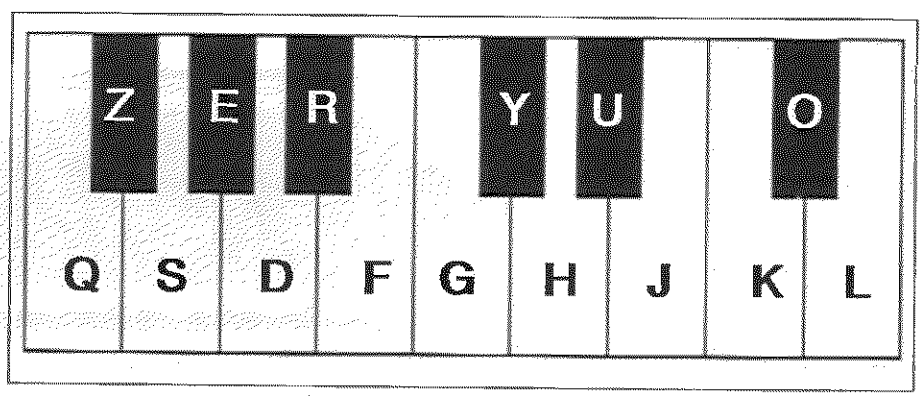
Utiliser le menu MIDI sans instrument

Une partie des options proposées par les menus MIDI sont accessibles même si on ne dispose pas d'un instrument MIDI connecté à l'ordinateur : certaines touches du clavier de l'Atari remplacent les touches de l'instrument.

Pour utiliser cette possibilité il faut avoir préalablement affecté les touches de fonction à l'aide de séquences créées ou chargées depuis la disquette.

On active **MIDI #2**. La fenêtre de visualisation de son est remplacée par un tableau indiquant quelles sont les séquences enregistrées et quelles sont les touches de fonction permettant leur rappel.

Si on appuie alors sur la touche **G** de l'ordinateur, la séquence est reproduite dans la tonalité dans laquelle elle a été enregistrée. Si on appuie sur **F**, elle est jouée décalée d'un ton vers le bas alors que **H** la joue décalée d'un ton vers le haut. En effet les touches du clavier ont alors la signification suivante :



Correspondance des touches sur l'Atari si on ne dispose pas d'un clavier MIDI.

Il faut retenir que la touche **G** correspond à la tonalité normale de l'échantillon (qui part donc de DO) et les décalages s'effectuent par demi-tons relativement à cette dernière.

On peut rapidement changer de séquence en appuyant sur une autre touche de fonction.

La commande **PITCH** offre les mêmes possiblités de décalage du son mais en temps réel depuis l'entrée de la cartouche. Chanter au micro avec le **PITCH** activé tout en s'écoutant simultanément au casque deux tons plus haut garantit le fou-rire à tous les spectateurs !

Extension MIDI

Les possibilités avec un instrument MIDI

Si on possède un clavier MIDI il est facile de le connecter à l'Atari car un seul câble suffit de la prise **MIDI OUT** de l'instrument vers la prise **MIDI IN** de l'Atari. On ajuste ensuite le canal MIDI de l'ordinateur grâce aux flèches situées à droite du bouton **MIDI CH** de façon à ce qu'il corresponde à celui de l'instrument (canal MIDI de 1 à 16). L'ordinateur est alors utilisé en expandeur.

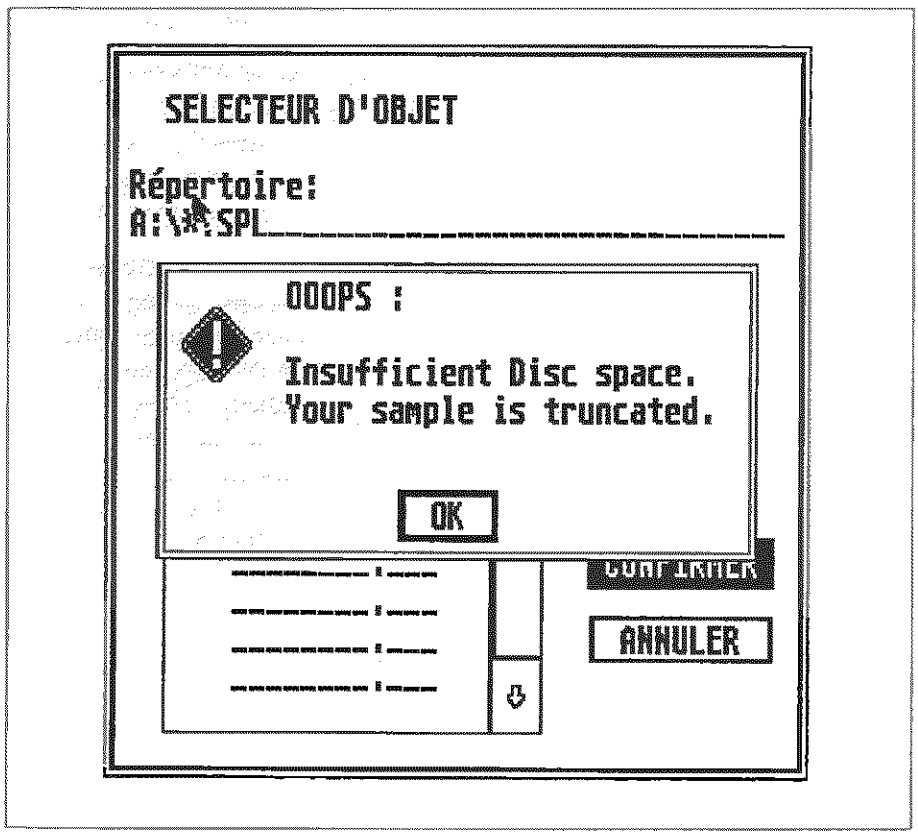
Les options **MIDI #2** et **PITCH** offrent les mêmes possibilités depuis le clavier MIDI que celles vues ci-dessus depuis le clavier de l'ordinateur si ce n'est que l'étendue de variation de fréquence est beaucoup plus étendue sur l'instrument que sur l'Atari dont les touches du clavier n'agissaient que sur un peu plus d'une octave.

L'option **MIDI #1** permet d'affecter à dix touches du clavier les dix séquences préalablement enregistrées dans les touches de fonction. Pour cela il faut activer le bouton **MIDI CH** qui passe en inversion vidéo inverse ; on termine en appuyant sur la touche du clavier de l'instrument que l'on souhaite affecter de cette séquence sonore (par exemple DO pour un aboiement de chien), les deux boutons repassent alors en vidéo normale indiquant que l'affectation est bien réalisée. On répète l'opération pour chacune des autres touches de fonction. Ceci fait, on peut activer **MIDI #1** qui fait apparaître la liste des affectations réalisées : nom de la touche de fonction, nom de la séquence, numéro MIDI de la touche de l'instrument. Un appui sur la touche de fonction ou la touche de l'instrument fait entendre la séquence correspondante.

Le menu fichier

Files
Load Sample Save Sample
Load Set Save Set
Info
Quit

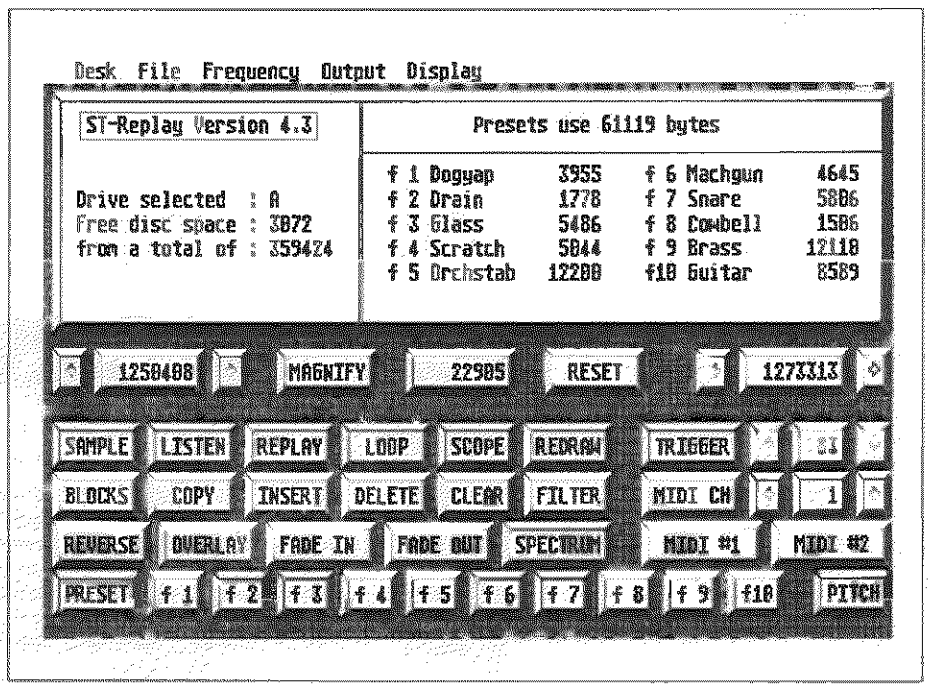
SAVE SAMPLE sauve sur la disquette la zone située entre les barres délimitatrices. On doit de préférence donner au fichier le suffixe SPL, ce que le logiciel ne fait pas automatiquement. Si on ne dispose pas d'un disque dur, il est illusoire de tenter de sauver un échantillon de plus de 720 kilo-octets sur une disquette ; la partie excédentaire est tout simplement ignorée lorsque la disquette est pleine.



LOAD SAMPLE charge un échantillon depuis la disquette et le place entre les barres délimitatrices préalablement positionnées. Si l'échantillon est trop long pour tenir entre les barres, il est tronqué.

LOAD SET et SAVE SET gèrent l'archivage des échantillons affectés aux touches de fonction. Le fichier contient en plus de l'échantillon, l'ensemble des paramètres le concernant : nom, taille, note MIDI... Lors du chargement, les échantillons ne sont pas forcément attribués aux mêmes touches de fonction : c'est le cas lorsque certaines touches n'étaient pas affectées au moment de la sauvegarde ou contenaient le mot EMPTY.

INFO donne la place disponible sur le disque ainsi que la liste des échantillons attribués aux touches de fonction.



Modifier le son d'un programme du commerce

Nous avons réalisé un essai sur ROBOCOP II. Au lancement de ce jeu un message de quelques secondes annonce les intentions du héros *« Serve the public trust, protect the innocents... »*.

Après avoir chargé ST Replay 4, nous introduisons la disquette ROBOCOP dans le lecteur. Pour trouver le fichier qui contient l'échantillon, nous transformons le filtre de recherche *.SPL en *.*.

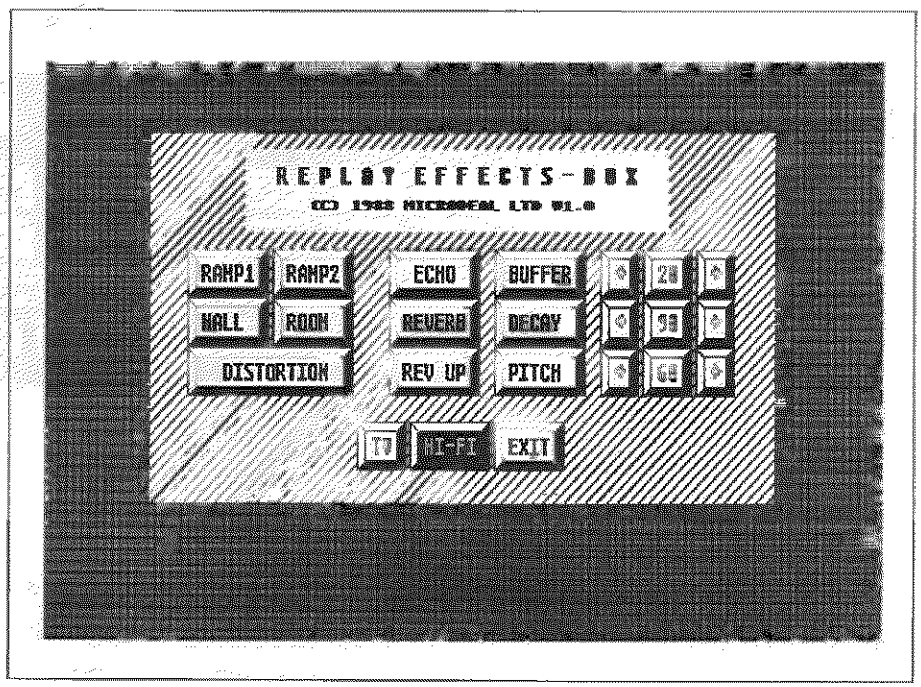
Parmi la liste des fichiers qui apparaît alors, le titre DIRECTIV.SAM (SAM pour SAMPLE) semble convenir. Ce fichier de 72 kilo-octets est mis en mémoire. Un essai de lecture par REPLAY à 20 kHz donne un son trop aigü. A 15 kHz c'est parfait (la longueur du fichier, 70 kilo-octets correspond bien à la durée d'enregistrement à 15 kHz, un peu moins de 5 secondes, ce qui explique d'ailleurs que ROBOCOP ne soit pas plus prolixe au cours de ce jeu car une seule disquette ne suffirait pas pour tout stocker).

Nous enregistrons notre propre message entre les deux barres délimitatrices. Une sauvegarde sous le même nom sur la disquette ROBOCOP et voici une version personnalisée qui ne manquera pas d'étonner les habitués du jeu.

Effets spéciaux

Le programme EFFECTS

Ce programme très simple fonctionne en temps réel et se compose d'un unique écran de travail.



EFFECTS, un exemple de programme qui pourrait mieux tirer parti de la cartouche.

TV ou HI-FI offrent le choix du dispositif de reproduction du son.



RAMP1 ou **RAMP2** créent un effet de vibrato ou plutôt de fading assez lent.

HALL et **ROOM** restituent la sonorité propre à ces deux salles en ajoutant de l'écho à la séquence captée au microphone.

DISTORSION accentue cet effet dans un logiciel qui n'en a déjà que trop à cause de la faible fréquence d'échantillonnage !

ECHO et **REVERB** diffèrent très peu si ce n'est dans l'importance du retard ajustable par **DECAY. BUFER** mémorise la durée avant extinction de l'effet.

REV UP est assez spectaculaire, : il s'agit d'un écho qui, tant qu'il se poursuit, augmente régulièrement en fréquence.