III. L'ECRITURE ASSISTEE PAR ORDINATEUR

Un des problèmes principaux en informatique et concerne la différence existant entre le langage dont l'homme se sert pour s'exprimer [LEVY 1987] et les langages formels qui régissent les ordinateurs.

Le compositeur utilise normalement le langage naturel, oral et écrit, imprégné souvent d'analogies et de métaphores. La langue qu'il utilise pour s'exprimer possède une grammaire, un lexique et des règles de syntaxe. Cependant, cette langue possède une grande part d'ambiguïté qui permet toutes les formes de relations incertaines dont se nourrit souvent l'imaginaire du compositeur [BOULEZ&GREUSSAY 1988]. Par contre, l'ordinateur appartient à un monde purement syntaxique régit par des règles strictes de transformation et de calcul. Les instructions doivent être sans ambiguïté car la machine ne sait pas interpréter d'ordre en fonction de situations particulières ou de contextes généraux, sauf si cela a été prévu dans la programmation [BARBAUD 1968, 142]. Avec la EAO, la formalisation est devenue un outil nécessaire pour communiquer avec la machine.

Il y a un écart important entre les démarches formalistes de la première moitié du XX^{e} siècle et le besoin formel imposé par l'ordinateur en EAO. La grande nouveauté dans cette deuxième moitié du XX^{e} siècle, est que l'utilisation de l'ordinateur en musique ne consiste pas en la formalisation de la théorie, mais en la formalisation de la pratique, domaine traditionnel de l'expérience personnelle.

L'assistance informatique à l'écriture peut être une discipline qui demande une introspection de la part du compositeur puisqu'elle permet de tracer les limites entre créativité/intuition et mécanisme/métier. Actuellement les compositeurs héritent, pour ainsi dire, de quelques unes des pratiques de la EAO, mais ils se rendent compte que la formalisation (la modélisation) n'est plus seulement un support intellectuel à la musique et à sa connaissance, elle devient un outil indispensable.

L'utilisation de l'ordinateur et de la programmation[1] vont sans doute changer la manière de penser des compositeurs, l'ordinateur étant à la fois un outil de calcul et un outil de pensée. La facilité qu'il offre pour établir des liens entre tous les niveaux d'une composition donne la possibilité d'utiliser des concepts analytiques et théoriques en les exprimant sous la forme d'algorithmes. L'ordinateur incite le compositeur à penser la structure musicale différemment, il stimule une réflexion sur le processus de la composition, et invite à un nouveau type de relation entre le créateur et le matériau.

Il est difficile de penser la EAO sans un environnement informatique approprié. Il faut tenir compte en premier lieu de la diversité des manières de penser la composition musicale. Il existe autant de manières de composer que de compositeurs, et une aide à la composition et plus précisément une aide à l'écriture sera toujours limitée par un cadre esthétique et formel. Penser un environnement informatique pour la EAO sera en quelque sorte définir des règles d'organisation et de manipulation d'objets musicaux sous une représentation informatique.

III.1. Interfaces et représentations

Le logiciel est, pour ainsi dire, le moteur « intellectuel » d'un ordinateur. Les logiciels ne sont pas des outils neutres, ils proposent une manière d'agir, une «instrumentalité », et par conséquent une manière de penser.

Par exemple, les séquenceurs sont des logiciels permettant d'enregistrer des événements MIDI ou audionumériques pour construire des séquences. Dans un certain sens ces logiciels permettent le même type d'opérations que l'on trouve sur les logiciels de traitement de texte. Il est facile de faire des corrections, des coupures et des déplacements d'événements ou de groupes d'événements, d'imprimer en notation musicale (pour le MIDI) et d'écouter le résultat. Ce type de logiciel conduit à une construction par transformations. Pour un compositeur qui travaille avec des outils MIDI (séquenceurs et synthétiseurs), ou audionumériques (comme Protools), l'écriture et l'audition ne sont plus des phases disjointes. L'ordinateur instaure un paradigme de génération, de traitement et d'édition dans lequel matériau et processus ne font plus qu'un.

---

[1] »Programmer un ordinateur revient tout simplement, plus ou moins, à communiquer avec lui dans un langage «intelligible» pour la machine comme pour le programmateur. » [PAPERT 981, 16].

Pour un compositeur, le choix d'un environnement informatique est complexe car les logiciels de CAO ou d'informatique musicale offrent une multitude de représentations et de syntaxes, les logiciels tels que Vision, Performer, Logic, Cubase, MAX, Patchwork, Symbolic Composer, Common Music, CARLA, l'UPIC, CSound, etc., en sont des exemples. Le fait que la syntaxe varie beaucoup d'un environnement à l'autre conduit à des manières différentes d'aborder les problèmes. Plusieurs approches de la représentation sont possibles.

La représentation textuelle, représente les données musicales sous forme de langages informatiques textuels. La représentation du processus musical est dissociée de la représentation musicale symbolique. Par exemple, dans l'environnement « Common-Lisp-Music Notation » (CMN) de Bill Schottstaedt, la programmation d'une séquence se fait par un langage textuel (Common-LISP).

Une représentation comme celle proposée par CMN est une représentation formelle avec beaucoup de ressources mais elle est très éloignée de la culture musicale. Les utilisateurs sont davantage attirés par des interfaces plus conviviales et plus proches de leur univers graphique.

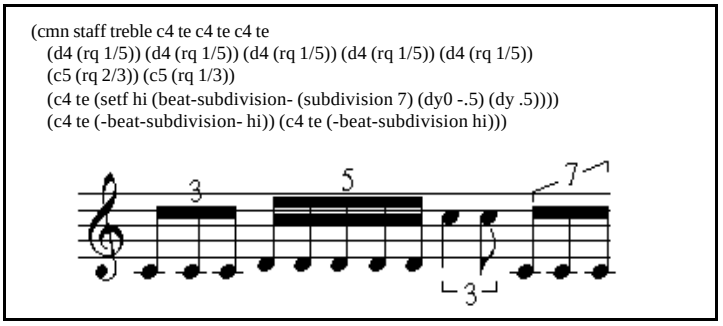

Figure 26 : Langage Common-LISP et représentation de la séquence.

Actuellement, la représentation textuelle tend à être délaissée au profit de l'approche symbolique et de l'approche sonore.

Le type de représentation correspondant à l'approche symbolique essaie d'utiliser la notation musicale traditionnelle. Ce type de notation est généralement choisi par des compositeurs ayant une prédilection pour l'écriture instrumentale. Un exemple d'environnement de ce genre est Patchwork[2].

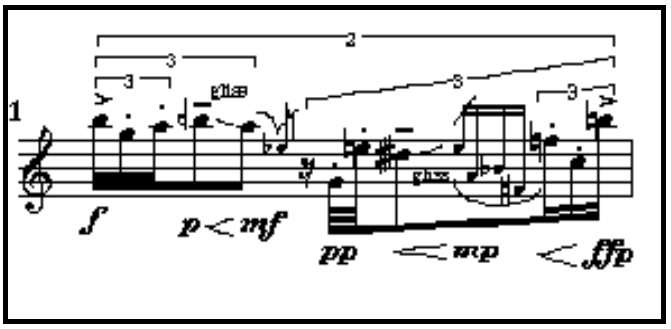

Figure 27 : Représentation dans Patchwork.

La représentation sonore propose des notations qui restituent les caractéristiques sensibles des sons perçus par le compositeur ou des représentations graphiques des paramètres analytiques du son. Le développement d'outils de cette nature a principalement commencé en France avec le développement de l'UPIC (L'Unité Polyagogique Informatique du C.E.M.A.Mu[3]) au milieu des années soixante.

[2] Rappelons-nous que les compositeurs engagés dans le développement de cet environnement à ses débuts (Esquisse) étaient Magnus Lindberg, Kaija Saariaho Marc André Dalbavie et Jean Baptiste Barrière, tous ayant une éducation musicale liée à l'écriture musicale dans le sens traditionnel du terme.

[3] C.E.M.A.Mu, Centre d'Études de Mathématiques et Automatique Musicale.

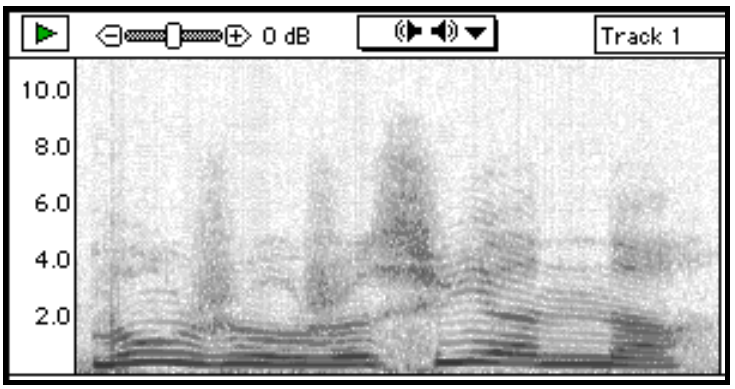

Figure 28 : Spectrogramme dans le logiciel SoundEdit [4]

Avec la vulgarisation de l'outil informatique il a été nécessaire de rendre les machines plus accessibles ; ceci a conduit à la création d'interfaces proposant des métaphores de situations plus familières à l'utilisateur : des « pages » graphiques à l'écran peuvant être « feuilletées », des dossiers à remplir avec des documents, un espace pour les poser, le bureau, et même une corbeille pour détruire les documents et dossiers inutiles. C'est l'actuelle représentation de l'espace de travail proposée par le système opérationnel d'ordinateurs tels le Macintosh[5] ou les ordinateurs PC avec un système Windows[6].

Avant l'apparition des interfaces graphiques, l'interaction homme/machine était peu évidente : claviers rudimentaires, cartes perforées et attentes interminables. Une description des conditions de travail de Hiller avec l'ordinateur ILLIAC I dans les années 50, est relatée par James Bohn[7].

Les interfaces graphiques ont été élaborées dans les années 60 au Stanford Research Institut au cours de recherches sur les partitions d'écran annonciatrices des fenêtres et sur l'utilisation de la souris. Ces recherches se sont poursuivies dans les années 70 au Palo Alto Research Center de Xerox. Elles ont été diffusées en 1983 et en 1984 par les sorties respectives de Lisa et du Macintosh d'APPLE. Depuis, les interfaces de type WIMP[8] se sont multipliées : Windows, X-Window, NextStep...

Quelques exemples d'interfaces :

```
     (pw : :defunp dec-to-bin ((dec fix/float) &optional (bin list
( :value '(0)) )) list
      »decimal à binaire »
     (cond
           ((eq dec 1) (setf (nth (- (length bin) 1) bin) 1)
                 bin)
           ....
```

Figure 29 : Langage Common-LISP

```
     /* minimum.c -- output the minimum of a group of numbers ------
- */
     #include  »ext.h »
     #define MAXSIZE 32
```

[4] Macromedia.
[5] APPLE Computer, inc.
[6] Microsoft.
[7] [BOHN 1996]. Je remercie spécialement James Bohn pour la gentillesse qu'il a montrée à mon égard en me cédant des documents originaux.
[8] WIMP : windows, Icons, Menus and Pointer : fenêtres, icônes, menus et pointeurs.

```
typedef struct minimum
{
struct object m_ob
Atom m_args[MAXSIZE]
...
} Minimum
void *minimum_bang()
void *minimum_float()
void *class
main(f)
fptr *f
{     setup(&class,    minimum_new,0L,    sizeof(Minimum),    0L,
A_GIMME, 0)
addbang(minimum_bang)  ... }
```

Figure 30 : Langage « C »

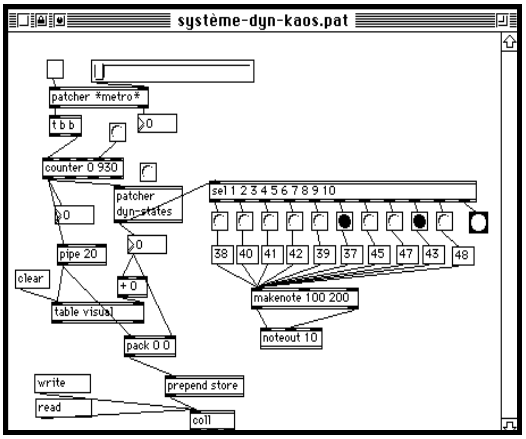

Figure 31 : *Langage graphique de l'environnement MAX*

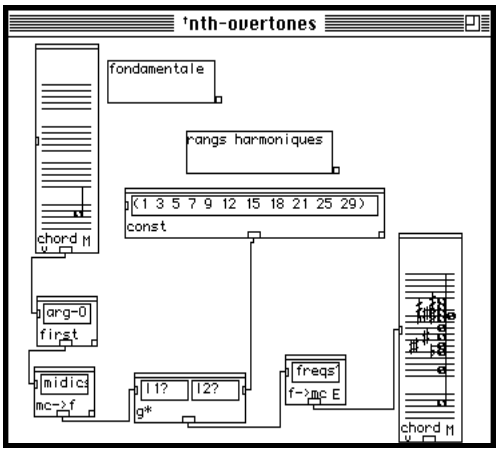

Figure 32 : Langage graphique de l'environnement Patchwork

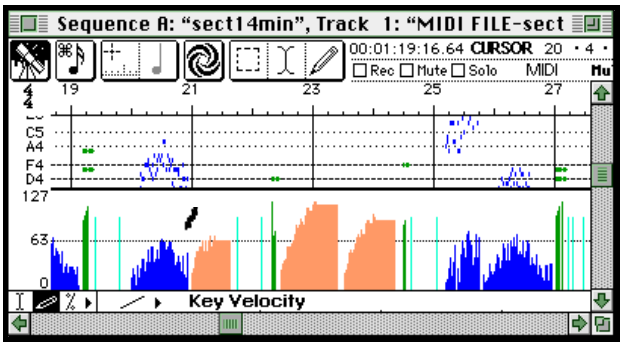

Figure 33 : *Fenêtre graphique du logiciel Vision*

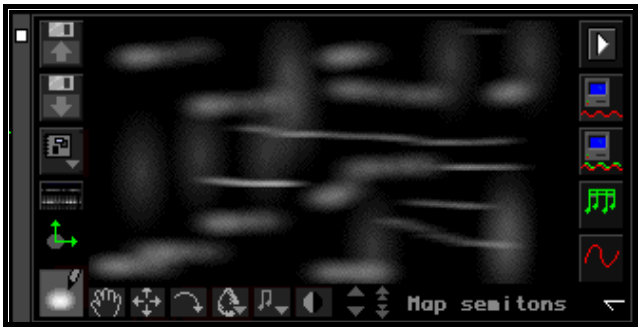

Figure 34 : *Fenêtre graphique du logiciel Metasynth d'Éric Wenger*

Indépendamment des divers niveaux que peuvent présenter les interfaces, l'informatique a modifié nos relations avec le réel. Les interfaces rendent plus conviviale notre relation avec le monde informatique mais elles induisent un décalage cognitif qui déplace l'action. L'utilisateur n'agit plus sur les objets mais sur leurs représentations[9]. Par exemple, la visualisation et la manipulation d'un objet sonore, fichier numérique stocké sur un disque, passe par un logiciel qui nous montre une représentation[10] de cet objet, qui peut être une forme d'onde ou un spectre *amplitude* **x** *fréquence*.

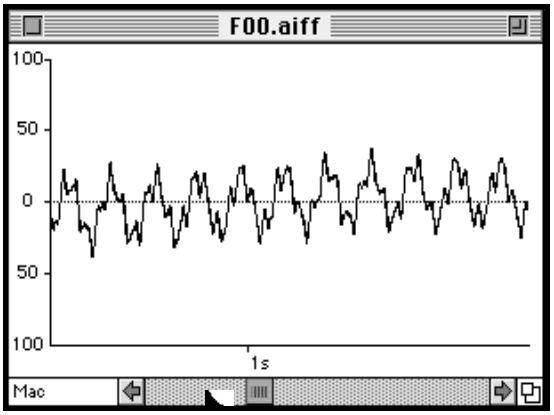

Figure 35 : Représentation de la forme d'onde

[9] [BOULEZ 1981b] et [DUFOURT 1981].
[10] oute interface est, implicitement, un modèle de représentation.

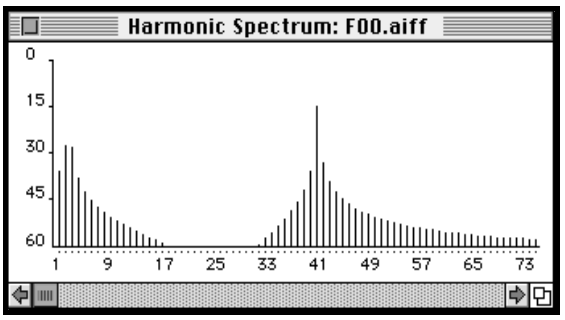

Figure 36 : Représentation du spectre amplitude x rang d'harmoniques

La commande et le contrôle des processus dépendent d'une combinaison de codes, de chiffres et de signes sur l'écran. Les opérations ne se font pas sur la matière sonore, mais à partir de combinaisons d'opérations graphiques qui vont du « copier- coller » au dessin de la forme d'onde.

L'ordinateur est considéré comme une sorte de médiateur numérique, c'est pour cela que beaucoup de chercheurs et de musiciens s'intéressent aux interfaces gestuelles.

Une partie du métier de compositeur réside dans une aptitude à transcrire des représentations musicales en un mode de notation, soit pour les transmettre à des instrumentistes, soit pour les transférer dans un univers symbolique pour les manipuler[11], Une représentation est toujours liée à une pensée musicale, c'est pourquoi, le choix de la représentation est fondamental pour le travail du compositeur. En ce qui concerne le problème des relations entre la représentation graphique d'un événement musical et ses caractéristiques sensibles, il semble qu'une relation biunivoque simple entre ces deux espaces soit très difficile.

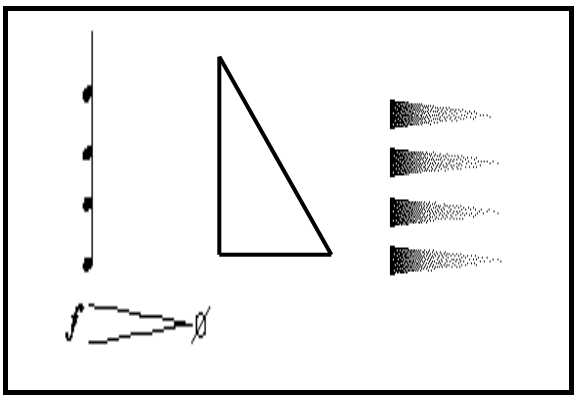

Figure 37 : Trois représentations d'un même résultat sonore.

Si ces trois représentations étaient soumises à un ensemble de compositeurs, ceux-ci seraient partagés en choisissant la représentation qui s'adapterait le mieux à leur propre imaginaire. Cela, malgré la constatation que l'objet qui est représenté par ces trois images est sensiblement le même.

Une représentation idéale serait une représentation définissable par l'utilisateur en fonction de ses besoins. La dissociation entre l'objet musical et la représentation graphique est, d'un certain point de vue, une solution à divers problèmes et en particulier à celui de la création de langages personnels. Un type d'environnement avec des représentations flexibles permettrait à chaque compositeur d'associer son système de notation à ses processus. Chacun pourrait ainsi construire, plus qu'une syntaxe, une sémantique personnelle au niveau graphique. Un tel principe permettrait de travailler sur l'ordinateur à plusieurs niveaux, de la macroforme à la microforme et vice-versa. De surcroît, ce type de démarche permettrait l'utilisation d'un même environnement informatique par plusieurs compositeurs, chacun pouvant attribuer une sémantique particulière à des processus qui pourraient être communs. Dans l'environnement Patchwork, il existe des processus de manipulation compositionnelle, des invariants, au niveau de la manipulation des objets sonores, auxquels chaque compositeur attribue une signification particulière selon les contraintes de l'ordre du contexte musical.

[11] A propos du problème des relations entre la représentation graphique d'un événement musical et ses caractéristiques sensibles [MALT 1995b] et [MALT 1996c].

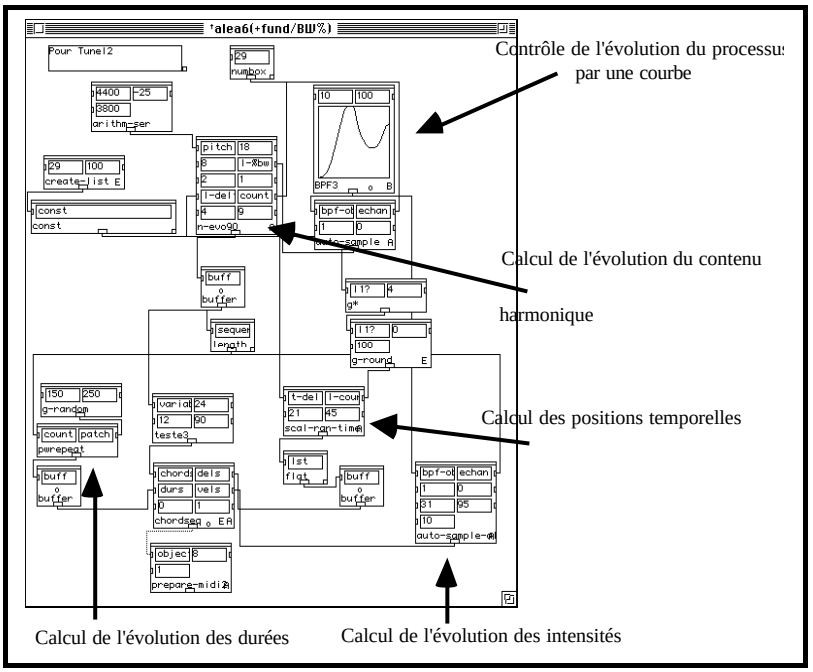

Figure 38 : Exemple d'un patch *Patchwork pour générer et contrôler l'évolution d'une masse d'événements [MALT 1995b], via MIDI ou par synthèse.*

Le dessin de la figure ci-dessous est beaucoup plus parlant que la configuration de boîtes du *patch* de la figure précédente.

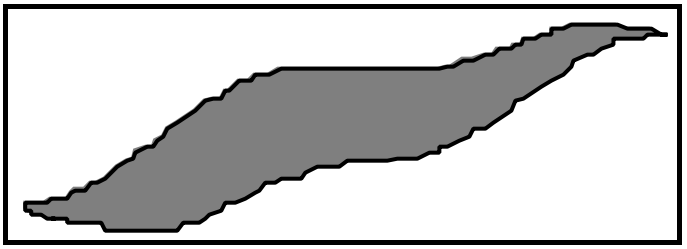

Figure 39 : Evolution d'un paramètre musical.

L'environnement Patchwork offre des facilités au niveau du calcul et de la concaténation des divers éléments du processus. Le point faible concerne la représentation du concept musical de l'évolution d'une masse d'événements. Un environnement offrant au compositeur un même degré de liberté en ce qui concerne la notation permettrait sans doute de franchir le pas, d'un point de vue informatique, entre les représentations abstraites du compositeur, souvent notées sur « l'esquisse » et la réalisation finale de l'oeuvre.

Un environnement bien conçu devrait permettre à un premier niveau de tracer des esquisses et à un deuxième niveau d'associer des processus à chacun des objets tracés. Cet environnement pourrait être utilisé comme un « cahier d'esquisses » pour enregistrer des idées en utilisant un langage graphique. Dans la plupart des logiciels destinés à la CAO il est possible de construire de nombreux processus de contrôle mais ils échouent tous lorsqu'il s'agit de représenter l'imaginaire du compositeur.

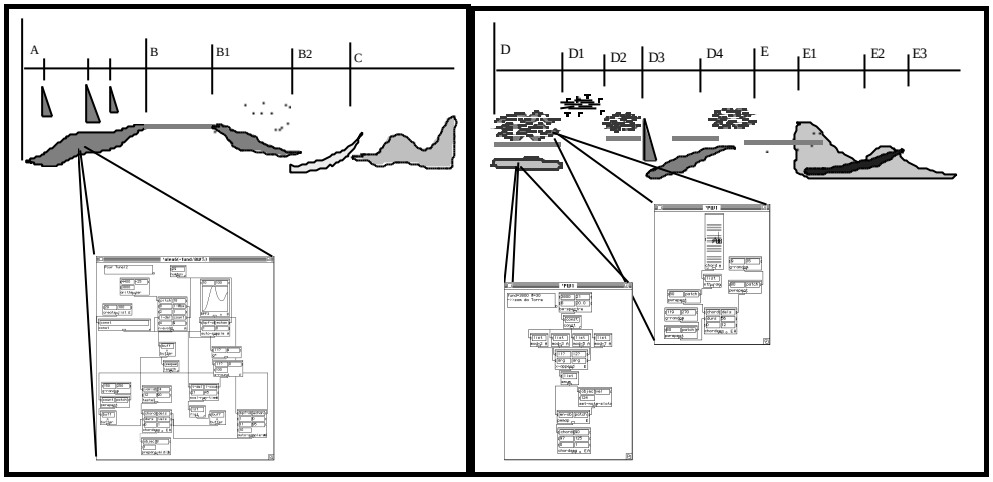

Figure ***Erreur! Argument de commutateur inconnu.*** *: Esquisses de bande de Mikhail MALT pour la séquence vidéo « Descente au Paradis » de Sabine PORADA [MALT 1995b].*

Souvent un *patch* repris quelque temps après sont élaboration, perd son sens pour le compositeur. Il ne reste plus de trace graphique de la signification musicale du processus, il ne reste qu'une syntaxe abstraite qui n'a plus rien à avoir avec les concepts d'origine. La programmation visuelle dans des environnements comme Patchwork a rendu plus intuitive la tâche de programmation, mais elle n'a pas réussi à constituer une sémantique visuelle fondée sur la configuration.

Le logiciel OpenMusic permet l'élaboration de projets musicaux à partir de la phase d'esquisse. Le compositeur peut, dans un premier temps, définir les articulations de son oeuvre pour ensuite leur attribuer un sens musical. Le compositeur peut également définir sa propre représentation picturale des événements musicaux. La figure ci-dessous montre un éditeur d'OpenMusic, avec une succession de triangles représentant des accords, des objets en dent de scie représentant des accords répétés, une trille avec crescendo représéntée par un cone, etc.. Dans cet exemple, les symboles ont été choisis par l'utilisateur et la signification musicale exacte définie ultérieurement dans un *patch*.

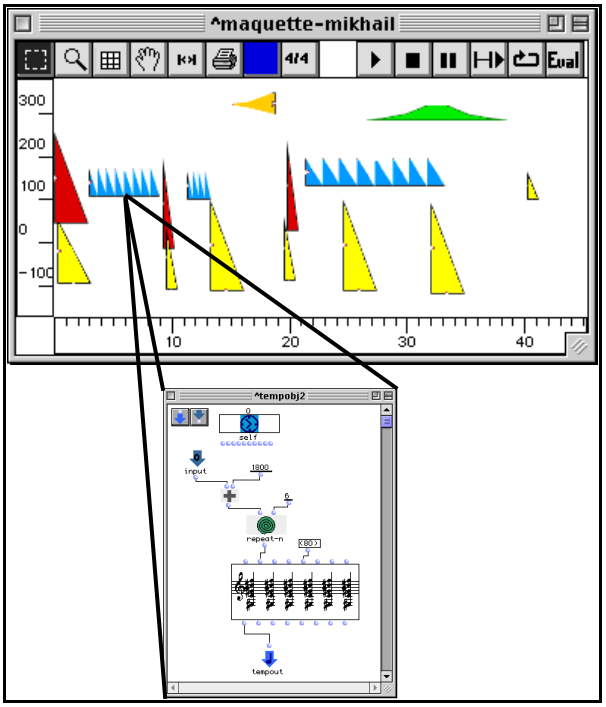

Figure 40 : Exemple d'un éditeur d'OpenMusic.

III.2. Les outils logiciels

Il existe deux types de systèmes opposés mais avec beaucoup de nuances : les systèmes descriptifs et les systèmes intelligents.

Les systèmes descriptifs sont des systèmes qui permettent une représentation primaire des données, avec un minimum d'opérations de base sur la représentation. Par exemple, les séquenceurs MIDI, les traitements de texte, l'environnement d'aide à la composition Patchwork, l'environnement MAX, le logiciel CSound, les langages de programmation en général, etc...

Les systèmes intelligents sont des systèmes qui sont capables de corriger les données au niveau de leur grammaire ou de leur sens. Ces systèmes utilisent les concepts extraits de l'intelligence artificielle. Par exemple, les systèmes experts, les systèmes à base de contraintes, le logiciel « Compose » de Charles Aimes, la librairie « Situation » [RUEDA&BONNET 1993b] dans l'environnement Patchwork, le programme CARLA qui intègre la notion d'apprentissage [COURTOT 1992a], etc..

Le choix entre la spécialisation et la flexibilité du système pose un problème : plus le système est spécialisé et performant (système intelligent) moins il est flexible, et, moins il est spécialisé (système descriptif) plus il est flexible.

Par ailleurs, il faut connaître la destination de l'application : une utilisation en temps réel ou en temps différé influence le choix des structures de données et du langage utilisé. Un exemple est donné par le binôme MAX-Patchwork, deux environnements pouvant être considérés comme des langages graphiques et visuels de programmation. MAX[12], destiné à l'origine au temps réel et au contrôle de l'ancienne carte 4X, devait être écrit dans un langage compilé et rapide, le langage « C ». Dans le cas de Patchwork, la priorité a été donnée à la représentation de connaissances et à la manipulation symbolique de données, aussi, le langage Lisp était plus adapté.

III.2.1. Logiciels de notation

Les Logiciels de notation sont des environnements dédiés à la représentation et au codage de la notation musicale. Score de Leland Smith, Finale [13], Notewriter[14], Encore [15], Common Music Notation[16], en font partie.

Ces environnements sont des systèmes descriptifs qui ont pour but principal la représentation symbolique d'événements sonores pour la gravure et l'édition de partitions. Il existe un grand éventail de logiciels, allant des langages textuels pour la description de partitions (Common Music Notation, par exemple) à des environnements avec des interfaces graphiques sophistiquées, et des fonctionnalités ajoutées (Encore et Finale) en passant par des logiciels purement graphiques (Notewriter).

```
(cmn (size 24)
 (system brace
 (staff treble (meter 6 8)
 (c4 e. tenuto) (d4 s) (ef4 e sf)
 (c4 e) (d4 s) (en4 s) (fs4 e (fingering 3)))
 (staff treble (meter 3 4)
 (c5 e. marcato) (d5 s bartok-pizzicato) (ef5 e)
 (c5 e staccato tenuto) (d5 s down-bow)
 (en5 s) (fs5 e)))
 (system bracket
 (staff bar bass (meter 6 16)
 (c4 e. wedge) (d4 s staccato)
 (ef4 e left-hand-pizzicato)
 (c4 e tenuto accent rfz) (d4 s mordent)
 (en4 s pp) (fs4 e fermata))))
```

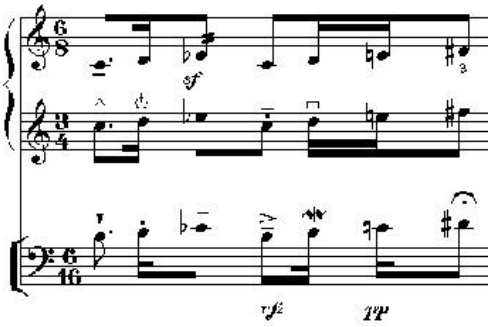

Figure 41 **:** *Exemple de description textuelle d'une partition en Common Music Notation*

[12] Opcode&Ircam.
[13] Coda Music Technology.
[14] Keith A. Hamel.
[15] Passport Designs, Inc.
[16] Bill Schottstaedt / Stanford University.

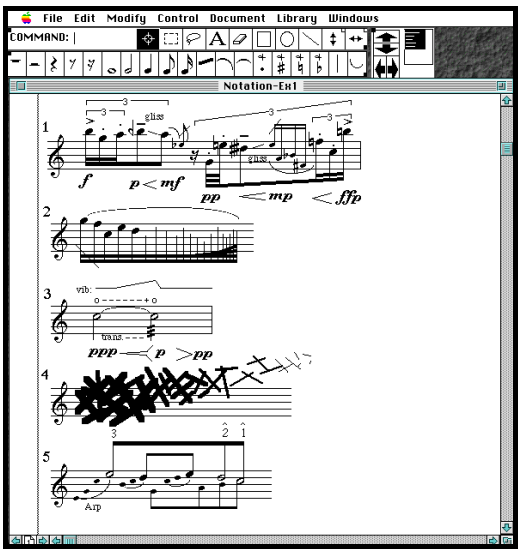

Figure 42 **:** *Exemple de partition graphique dans l'environnement Notewriter*

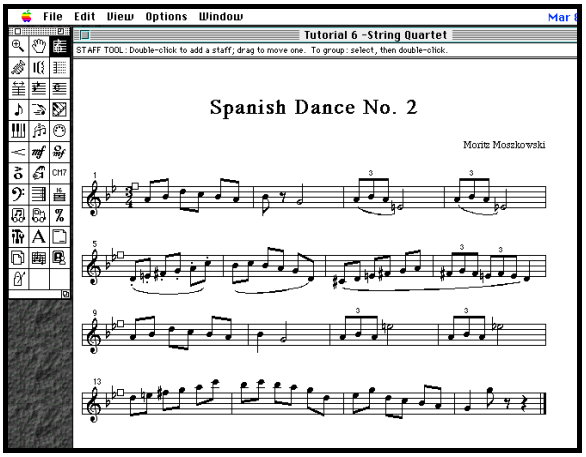

Figure 43 **:** *Exemple de l'environnement Finale*

III.2.2. L'audionumérique

L'audionumérique concerne les langages et les environnements dédiés à la synthèse, au traitement et à la manipulation de fichiers sonores numérisés. Les environnements de cette catégorie se subdivisent en logiciels de synthèse, de traitement de fichiers sonores, de mixage et de montage de fichiers de sons numérisés.

Parmi ces logiciels et environnements se trouvent Alchemy, Turbosynth, Csound, SuperCollider, la Station Ircam, l'environnement Syter du GRM et le GRM-tools, AudioSculpt, Modalis, Soundhack, Lemur, SoundEdit, Hyperprism, Phonogramme, le logiciel DECK et Protools[17] pour le « mixage » de fichiers de sons numérisés. Ces environnements proposent des représentations formalisées du fait sonore : des représentations numériques de paramètres physiques (Csound, SuperCollider, etc.), des représentations de formes d'onde et des sonagrammes de temps versus fréquences. Chaque environnement propose une manière de représenter, de penser et de manipuler des processus et du son.

Les logiciels Csound et SuperCollider sont des langages textuels appliqués à la synthèse et au traitement du son.

[17] Alchemy : Passport Designs, Inc. ; Turbosynth : Digidesign, Inc. ; Csound : Barry Vercoe, MIT ; SuperCollider : James McCartney ; Syter et GRM-tools : Groupe de recherches Musicales ; AudioSculpt et Modalis : Ircam ; Soundhack : Tom Erbe ; Lemur Bill Walker et Kelly Fritz, University of Illinois ; SoundEdit : Macromedia ; Hyperprism : TM de Arboretum Systems, Inc. ; Phonogramme : Vincent LESBROS au Groupe Art et Informatique de Vincennes à St Denis, Université Paris 8 ; DECK : Macromedia ; Protools : Digidesign, Inc.

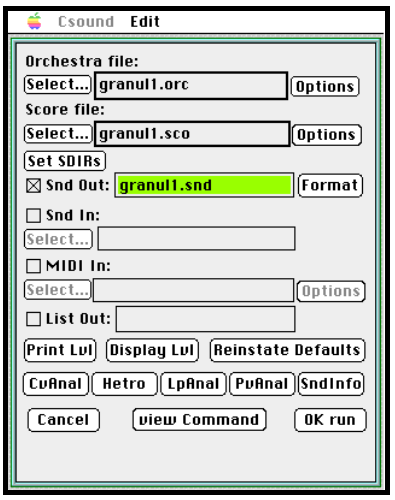

Figure 44 **:** *Interface de l'environnement Csound*

Audiosculpt, Alchemy, Soundesigner, Lemur, SoundEdit permettent des représentations du son sous forme de sonagrammes ou de diagrammes de formes d'onde issus de concepts analytiques du traitement du signal. D'autres logiciels favorisent une représentation du processus, soit, par enchaînement de modules comme Turbosynth, soit, par la captation du geste de l'utilisateur par la souris de l'ordinateur tels que GRMTools ou Hyperprism.

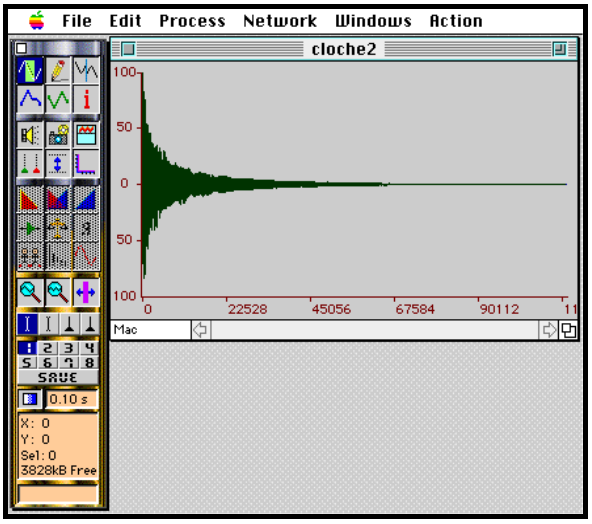

Figure 45 **:** *Interface de l'environnement Alchemy*

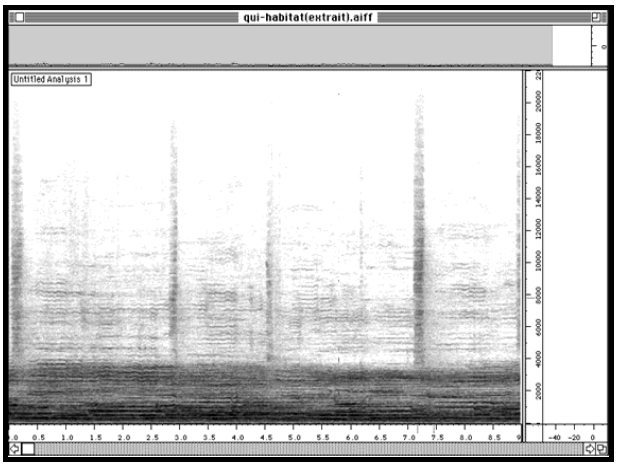

Figure 46 **:** *Sonagramme dans l'environnement AudioSculpt*

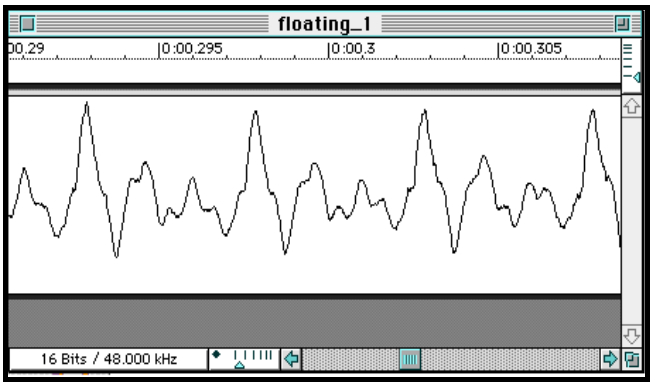

Figure 47 **:** *Représentation du son en forme d'onde dans le logiciel SoundEdit*

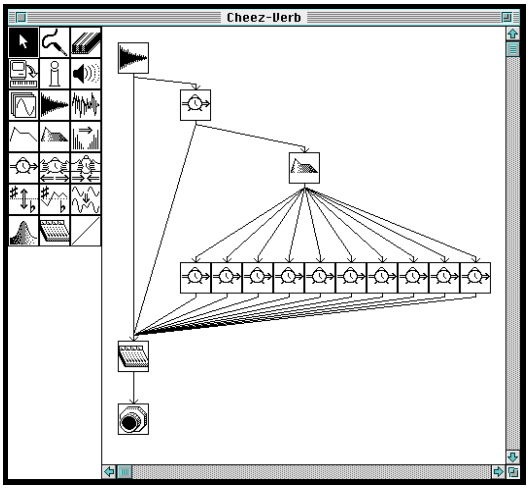

Figure 48 **:** *Représentation d'un processus de traitement du son dans le logiciel Turbosynth*

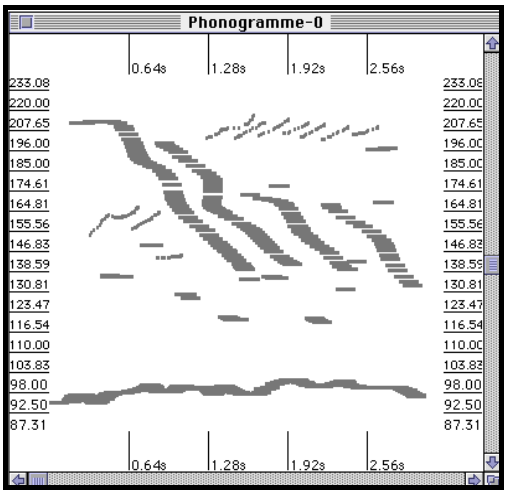

Figure 49 **:** *Fenêtre principale du logiciel Phonogramme*

Des logiciels tels que Phonogramme ou Metasynth[18], récupèrent le concept d'interface graphique liés à l'UPIC[19] permettant de « dessiner » des évolutions temporelles de fréquences avec une correspondance directe entre l'espace temps versus fréquence et l'espace graphique en deux dimensions.

Il existe également des environnements représentant des extraits de sons numérisés comme des objets à combiner, tels que Protools ou DECK.

[18] Eric Wenger.
[19] [XENAKIS 1976, 200-202].

III.2.3. Le temps réel

Parmi les langages et les environnements destinés au contrôle d'événements en temps réel[20],se trouvent les logiciels MAX[21], SuperCollider, l'environnement FTS[22], la station MARS[23], Jmax[24] et MSP[25].

Ces environnements se caractérisent par un intervalle de temps très court entre le moment d'entrée du signal et le moment de sortie après traitement. Ils sont plutôt utilisés en situation de concert ou dans des installations.

III.2.4. Le calcul de structures musicales

Les langages et environnements destinés au calcul d'événements et de structures musicales, tels que Symbolic Composer[26], BP2[27], Cæcilia[28], Common Music[29], HMSL, Aleatoric composer, Elody[30], Patchwork et OpenMusic n'ont pas pour tâche principale la production de son mais la détermination de coordonnées de paramètres musicaux dans un espace.

III.2.5. Des points de contact

Dans la pratique ces divisions ne sont pas étanches, plusieurs environnements permettant d'accéder à beaucoup de fonctionnalités. Par exemple, l'environnement Finale, bien qu'utilisé principalement comme logiciel pour la gravure de partitions de musique, peut aussi envoyer et recevoir des informations MIDI tel que le font les séquenceurs. D'autres environnements, comme ceux destinés aux calculs d'événements et de structures musicales, intègrent des fonctionnalités d'édition et de communication MIDI, de manière à permettre une plus grande interactivité de la part de l'utilisateur avec le matériau généré.

MAX est un exemple d'environnement multiforme. Conçu pour contrôler de la station 4X, il est très vite utilisé comme un logiciel de contrôle pour des dispositifs MIDI, par exemple, dans « Allégories » de Tristan Murail. D'autres compositeurs l'utilisent comme plate-forme graphique de programmation pour le calcul de structures musicales. Ainsi, ce logiciel qui était un logiciel destiné au contrôle de dispositifs externes en temps réel est aussi utilisé pour l'aide à l'écriture.

En ce qui concerne les logiciels ayant une destination plus commerciale comme, par exemple, les logiciels de notation et les séquenceurs, des contraintes de marché fondées principalement sur la concurrence finissent par imposer une accumulation de fonctionnalités.

Contrairement aux développements des années 70 où la tendace était aux « workstations », environnements informatiques à tout faire, actuellemement, le développement de logiciels semble aller vers la production d'outils légers et spécialisés qui permettent une communication aisée entre eux.

III.3. Les langages de programmation

Chaque langage est fondé sur un paradigme de programmation. Papert définit un paradigme de programmation comme étant le « cadre structurant qui est sous-jacent à l'activité de programmer » [PAPERT 1991, 8], il ajoute, par ailleurs, que le choix de ce paradigme peux changer remarquablement la manière de penser la tâche de programmation.

Norvig [NORVIG 1992, 434] définit cinq styles primaires desquels il découle quatre paradigmes de programmation : procédural, fonctionnel, déclaratif et orienté-objet.

---

20 Avec l'évolution de l'informatique, la vitesse de calcul a grandi à un rythme pratiquement exponentiel, pour cette raison certains chercheurs proposent une autre classification en ce qui concerne le temps. Par exemple, pour des environnements à temps différé, la distinction est faite aussi entre les environnements à temps lent et les environnements à temps rapide.
21 Opcode&Ircam.
22 Ircam.
23 IRIS. [ANDRENNACI&al 1997].
24 Ircam.
25 MSP est une librairie pour l'environnement MAX qui permet le traitement de signal en temps réel.
26 Tonality Systems.
27 Bernard Bel.
28 Jean Piche.
29 De Heinrich Taube et Tobias Kunze.
30 GRAME.

Le paradigme procédural à partir duquel un programme est construit comme une série d'instructions, chacune effectuant une action, par exemple dans les langages Assembler, FORTRAN, C et PASCAL.

Le paradigme fonctionnel dont part l'hypothèse que toute tâche peut être subdivisée en sous-tâches, appelées fonctions, par exemple dans le langage Lisp.

Le paradigme déclaratif qui conduit à exprimer un problème sous la forme des propriétés de sa solution. L'utilisateur n'exprime pas des algorithmes sous forme impérative (pas-à-pas), mais décrit une base de données et un ensemble d'assertions exprimant des propriétés de la solution, ou d'un ensemble de solutions. Une classe importante de ce paradigme est l'ensemble connu comme langages logiques, parmis lesquels se trouve le langage PROLOG. Il y a aussi dans ce paradigme toute une classe de modèles informatiques pour la résolution de problèmes par contraintes [LAURSON 1996, 3].

Le paradigme orienté-objet propose un autre point de vue sur l'activité de programmation. Au lieu de penser un programme comme étant un ensemble d'actions qui manipulent des objets, il pense à un ensemble d'objets manipulés par des actions [NORVIG 1992, 435], chaque objet étant pensé aussi comme un acteur. La connaissance n'est plus centralisée dans des bases de données, mais elle est répartie entre les différents acteurs du programme (objets) [LIEBERMAN 1982, 9].

Programmer et utiliser un environnement ou un langage fondé sur l'un des paradigmes cités, implique de représenter des problèmes ou le même problème, selon différents modèles de pensée. BARANAUSKAS [BARANAUSKAS 1993, 48] cite le psycholinguiste Benjamin Lee Worth en rapportant qu'un paradigme de programmation peut être envisagé comme une structure sous-jacente du langage qui influence la manière dont nous envisageons la construction et l'utilisation d'environnements informatiques, en insistant sur le fait que ces modèles sous-jacents sont plus importants que les langages en eux-mêmes.

L'exemple de trois environnements, CARLA [COURTOT 1992], Patchwork et Csound, dérivant de trois paradigmes de programmation différents - logique, fonctionnel et procédural – permet de se rendre compte, de l'influence du paradigme de programmation sur la manière d'utiliser un environnement et de la nécessité de bien choisir le langage pour pouvoir mettre en adéquation les concepts et leurs représentations. Cet aspect peut être bien saisi si on fait une comparaison avec les instruments de musique traditionnels dont les caractéristiques physiques induisent toujours certains modes de jeux particuliers.

III.4. Les outils d'aide à l'écriture

> « Ainsi, l'avantage de la CAO n'est pas seulement le fait de pouvoir automatiser le processus de composition, mais aussi le fait qu'elle nous aide à formuler nos problèmes musicaux d'une manière précise et efficace. » [LAURSON 1996, 14] [31]

L'histoire des environnements de CAO, et plus précisément de l'aide à l'écriture, se confond avec l'histoire des environnements de codage de la musique. Les environnements d'aide à l'écriture permettent au compositeur la génération, le traitement et l'édition de données qui peuvent, par la suite, être exploitées pour la composition instrumentale, pour la synthèse et le traitement sonore ou pour le contrôle de dispositifs. Chacun de ces environnements suppose une formalisation, une représentation de la musique ou des objets musicaux qu'il traite. L'écriture finale est, soit, notée traditionnellement, soit, réalisée par un mixage numérique dans des logiciels tels que Protools ou Deck. Chacun de ces environnements peut ainsi être utilisé dans une phase du processus de composition.

III.4.1. Quelques systèmes d'aide à l'écriture

Par rapport aux logiciels existant, le nombre d'environnements d'aide à l'écriture est minime. En plus des environnements cités il existe beaucoup de petits outils, souvent écrits en langage « C » ou en Common-LISP, pour des applications très ponctuelles ou pour des projets personnels. Il existe très peu d'environnements proposant des solutions plus générales pour la représentation et la manipulation de processus musicaux.

En 1993, une liste d'outils destinés à la composition algorithmique, éditée par Leonidas Hepis, est apparue sur le réseau internet. Cette liste donnait un aperçu des logiciels, ou environnements, d'aide à l'écriture

[31] « Thus, the advantage of CAC is not only in the automation of the compositional process, but also in the fact that it helps us to formulate musical problems in a precise and efficient way. » CAC veut dire Computer assisted composition, soit composition assistée par ordinateur.

disponibles[32]. Très peu de ces environnements ont survécu, soit, à cause des évolutions des plates-formes informatiques, soit en raison de leur manque de généralité et de leur spécialisation[33].

	Mac	IBM	Amiga	Atari	NeXT/Unix
Aleatoric Composer		X			
AlgoRhythms			X		
BIAB	X	X		X	
Bol Processor BP2 *	X				
Common Music *	X	?			X
COMP2 *					
Drummer		X			
f1 *					
ForthMox				X	
HMSL	X		X		
Hyperlisp	X				
Hyperupic					X
Interactor	X				
Jammer		X			
Keynote *					
(continued)	Mac	IBM	Amiga	Atari	NeXT/Unix
M or M/PC	X	X	X	X	
Markov_SMUS			X		
Max	X				
mf2t *					
MidiLyre			X		
MODE	X	X			X
Music Mouse	X		X	X	
MusicBox		X			
Pip		X			
Powerchords		X			
Ravel		X			
RGS			X		
Sound Globs		X			
(continued)	Mac	IBM	Amiga	Atari	NeXT/Unix

Figure 50: liste éditée par Leonidas Hepis.

« Compose » de Charles Aimes, écrit en langage « C », est un exemple d'environnement personnel. Sa programmation textuelle est facilitée par des fenêtres graphiques, mais il reste assez limité dans le cadre d'une composition, étant plutôt destiné à calculer des petits extraits, ou à générer de la musique algorithmique.

[32] Encore de nos jours, dans les pays anglo-saxons le concept d'aide à la composition ou notamment d'aide à l'écriture est très méconnu, le terme de composition algorithmique englobant ainsi tous les aspects de l'utilisation de l'ordinateur dans ce que nous avons appelé d'aide à l'écriture, en allant du simple calcul d'un matériau pré-compositionnel au calcul d'une pièce entière.

[33] Par exemple : MusicKit en 1997, développé au CCRMA, Stanford University ; Nyquist développé par Roger Dannenberg à la CMU School of Computer Science ; Silence développé par Michael Gogins en 1997 ; KeyKit : développé par Tim Thompson en 1996, AT&T Corp.

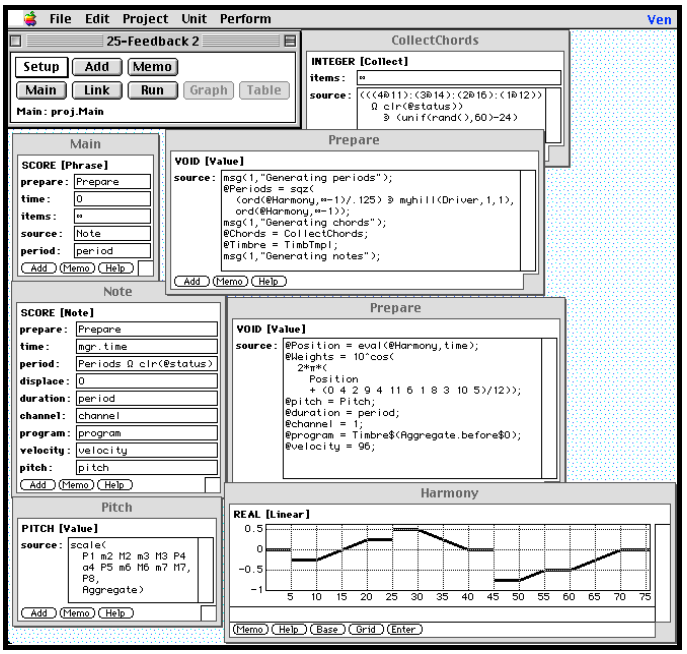

Figure 51 **:** *Ensemble de fenêtres et menus proposés par le logiciel « Compose »*

Dans cet environnement, tout est pensé en termes de programmation textuelle, même l'édition des courbes (ou fonctions) par segments se fait par l'édition individuelle de chaque portion. La figure ci-dessous montre la nécessité d'assigner textuellement les coordonnées pour le début et la fin de chaque segment, sans solution graphique.

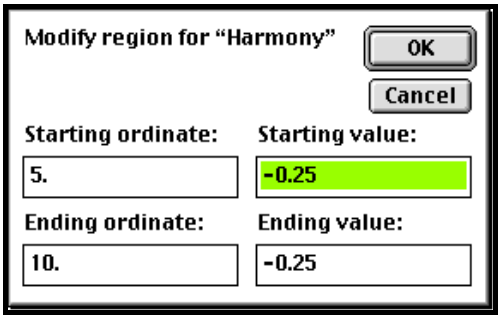

Figure 52 **:** *Détail de l'éditeur de fonctions par segments*

Actuellement, les environnements les plus utilisés et qui proposent des solutions originales à un niveau informatique (principalement en ce qui concerne l'interface) ou à un niveau musical sont principalement HMSL, Symbolic Composer, BP2, MAX, Common Music, Patchwork et OpenMusic.

III.4.2. HMSL

HMSL, « the Hierarchical Music Specification Language », est un langage de programmation destiné à la composition et à la performance musicale. Basé sur le langage Forth, il a été développé en 1980 par Phil Burk, Larry Polansky et David Rosenboom au « Mills College Center for Contemporary Music » et est actuellement distribué par le réseau internet. Ce langage fonctionne principalement sur des plates-formes informatiques du type Macintosh et Amiga.

HMSL est essentiellement un environnement de programmation textuel mais permettant l'échange d'informations par le protocole MIDI en temps réel. Ce langage est structuré sur un modèle informatique de programmation objet permettant la gestion de données complexes. L'intérêt de cet environnement est d'ajouter des fonctionnalités au langage Forth comme le « Score Entry System » (SES) qui propose une représentation musicale fondée sur la notation alphabétique anglo-saxonne.

HMSL manque d'une interface graphique standard mais contient des librairies de classes qui en permettent la construction. Il est possible de construire sa propre fenêtre avec des boutons et des potentiomètres coulissants (*sliders*), pour permettre une interactivité en temps réel. Toutefois, l'utilisation de HMSL demande une certaine compétence dans la programmation textuelle, ce qui le rend très peu convivial et demande beaucoup d'efforts pour la construction algorithmes et d'interfaces de bas niveau.

Dans la figure ci-dessous, nous pouvons voir comment il est possible de combiner des opérations caractéristiques du langage Forth, telles que DO LOOP et IF avec des commandes SES. Cette « partition » peut être jouée directement sur un dispositif MIDI.

```
\ Define a simple melody as a motif.
: MOTIF1  ( -- , this is a comment )
      C3 D G E4 A3 G
;
\ Play that motif as quarter notes then sixteenths
PLAYNOW   1/4 motif1  1/16 motif1

\ Play random notes in one octave range
\ in parallel with an accelerando
: DOIT
   par{
      16 0  \ set up loop
      DO  12 choose \ random over 1 octave
         60 + \ above middle C
         note \ play it
      LOOP
   }par{
      1/2 \ half notes
      \ accelerate using floating point
      \ by a factor of 1.2 every 3 half notes
      1.2  3 2 faccel{
         D E F#
         _pp A B \ softly
         _ff C E \ loud
      }faccel \ stop accelerating
      1/6 C D E C D E  \ triplets
   }par
;
```

Figure 53 : *Exemple de programmation dans l'environnement HMSL*

III.4.3. Common-Music

Common Music [TAUBE 1991] est un environnement d'aide à l'écriture orienté-objet. Son but premier est la génération de données de contrôle pour d'autres environnements ou pour des dispositifs extérieurs à l'ordinateur, via MIDI. Cet environnement permet la génération de données dans une grande variété de protocoles comme Lisp Music, Music Kit, C Mix, C Music, M4C, RT, Mix et Common Music Notation.

Common Music, qui est écrit dans le langage Common Lisp, dispose de deux interfaces principales, « Stella » et « Capella ». La première est une interface textuelle courante dans le monde de la programmation Lisp. La deuxième, [KUNZE &TAUBE 1995], fonctionnant actuellement sur des plates-formes Macintosh, propose quelques solutions graphiques telles que des menus et des éditeurs de fonctions par segments.

Heinrich Taube, le concepteur, souhaitait que cet environnement soit utilisable sur diverses plate formes informatiques[34]. L'interface textuelle, bien que peu commode du point de vue l'utilisateur lui a semblé être la solution plus commode puisqu'elle ne dépendait pas des boîte à outils interne des sytèmes pour l'affichage des

[34] Il l'est déjà sur des plate-formes Macintosh et PC et il existe aussi des versions sous UNIX, Sun et NextStep.

parties graphiques. Une des raisons qui lui a fait préférer l'interface textuelle est que le texte est un format qui permet facilement les échanges entre les diverses applications et entre les diverses plateformes informatiques.

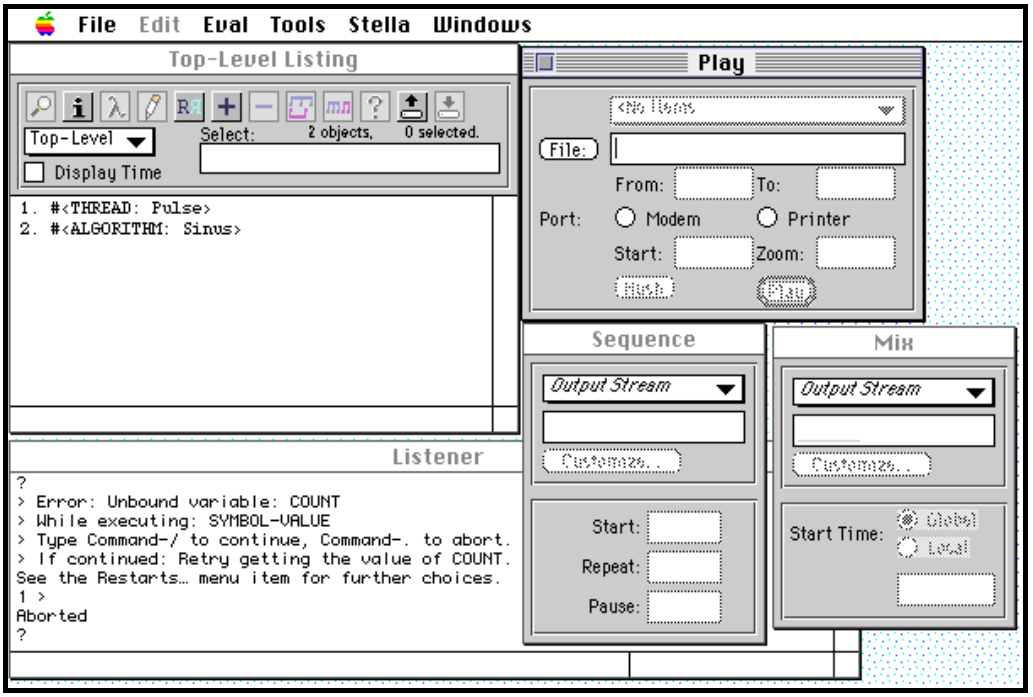

Figure 54 : *Common Music sur la plate forme « Macintosh » avec l'interface « Capella ».*

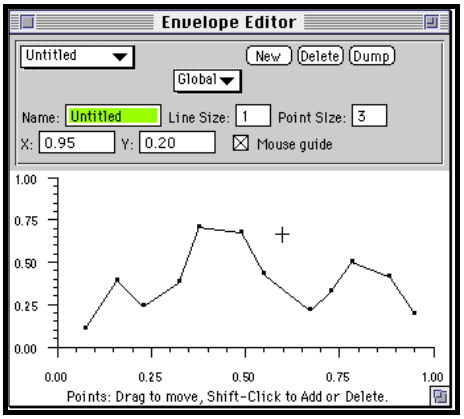

Figure 55 : *Editeur de fonctions par segments de l'interface « Capella »*

Une autre caractéristique de Common Music est d'avoir une structure d'objets définie à l'avance, facilement extensible et fondée sur une classe générique appelée « object ». La figure ci-dessous montre un schéma de l'auteur [TAUBE 1997] qui représente les relations entre les principales classes de cet environnement.

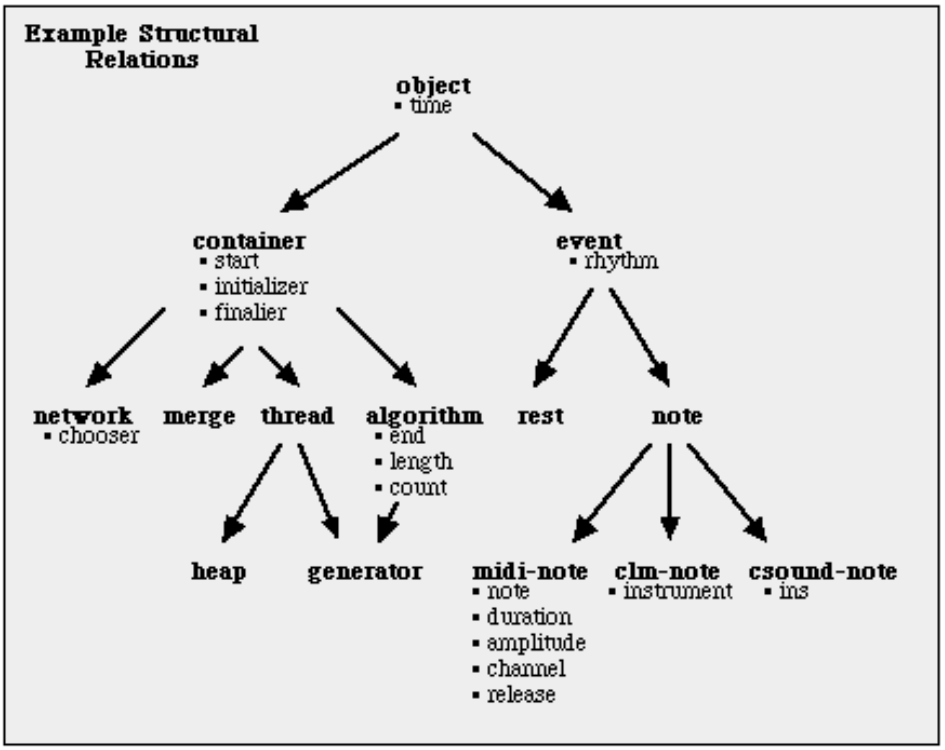

Figure 56 **:** *Système de classes de Common Music*

```
(defun sierpinksi (level beg dur pitch)
  (algorithm nil midi-note (length 3 start beg channel level)
    (setf rhythm dur)
    (setf duration dur)
    (setf note (item (intervals 0 7 2 from pitch)))
    (when (> level 1)
      (sprout (sierpinksi (1- level) time note (/ rhythm length))))))
```

Figure 57 **:** *Exemple de définition de fonction sous l'environnement Common Music*

Malgré le soin apporté aux les fonctionnalités (opérations et commandes), Common Music pose un problème d'utilisation pour des personnes non habituées à la programmation. En effet, la définition de nouvelles opérations doit se faire au niveau de la programmation textuelle. L'interface graphique disponible sert principalement à gérer les opérations d'entrée et de sortie et à afficher graphiquement des fonctions par segments. La représentation des événements musicaux ne se fait que par des descriptions textuelles symboliques dérivées de la notation anglo-saxonne ou numériques.

```
(algorithm chords midi-note (amplitude .75)
  (setf note
    (item (notes (chord (notes c6 d ef f g af bf
                        in heap for 4))
                 (chord (notes c5 d ef f g af bf
                        in heap for 4))
                 (chord (notes c4 d ef f g af bf
                        in heap for 4))
                 r)
          :kill 6))
  (setf rhythm
    (item (rhythms 32 (rhythms e q. h for 1) e e)))
  (setf duration rhythm))
```

Figure 58 **:** *Exemple de description symbolique des hauteurs en Common-Music*

III.4.4. Symbolic Composer

Symbolic Composer, développé par une équipe de Tonality Systems dirigée par Peter Stone, ressemble à une version commerciale de Common Music. Il s'agit d'un environnement fondé sur la programmation textuelle qui ne permet l'affichage de données que sous forme géométrique (graphes de points) ou textuelle (alphanumérique).

Dans cet environnement les opérations disponibles sont accessibles par le biais de menus. Pourtant, cette fonctionnalité ne dispense pas l'utilisateur de la programmation textuelle et ne résout pas le problème de l'échange d'informations entre les divers opérateurs. Cependant, « Symbolic Composer » se montre extensible par la possibilité donnée à l'utilisateur de définir ses propres opérations et ses bibliothèques de fonctions.

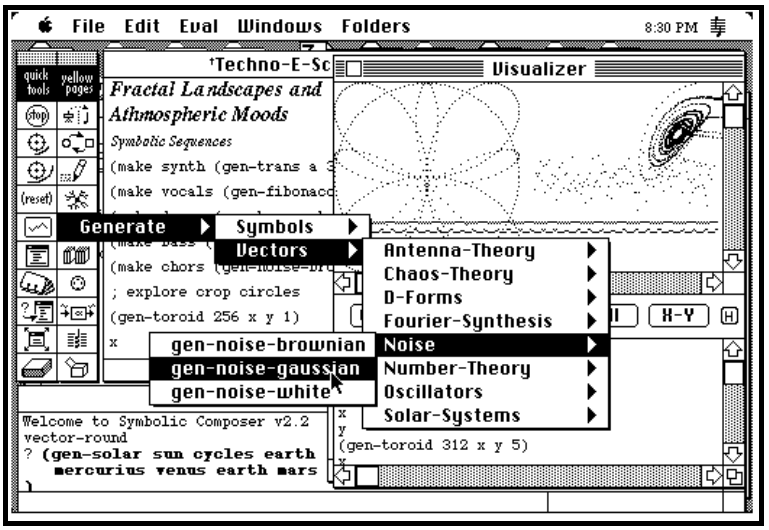

Figure 59 **:** *Ensemble de fenêtres et de menus de l'environnement « Symbolic Composer »*

Malgré tout le soin apporté à l'interface, les solutions proposées pour la représentation musicale sont très formelles et décalées de la réalité plus immédiate du monde musical. Par exemple la représentation rythmique proposée dans le logiciel est fondée sur une liste de fractions : la suite « croche, double-croche, double-croche » sera représentée par la liste « 1/8, 1/16, 1/16 ».

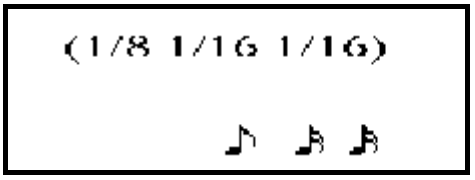

Figure 60 **:** *Représentation d'une cellule rythmique par une liste de fractions*

La définition de séquences plus complexes passe aussi par une programmation textuelle. L'expression : (def-length violin '(1/8 1/16 1/16 1/16 1/8 1/16)) produira la séquence suivante

Ce genre de représentation rythmique ne s'adapte pas aisément à des représentations internes de bas niveau alors que l'utilisateur est obligé de les manipuler directement. Toutes les solutions données dans les documentations annexes [MORGAN 1997] poussent l'utilisateur à l'utilisation de représentations alphanumériques.

III.4.5. MAX

L'environnement MAX est un langage graphique pour la manipulation de flux de données à partir d'un ensemble appelé *patch*.

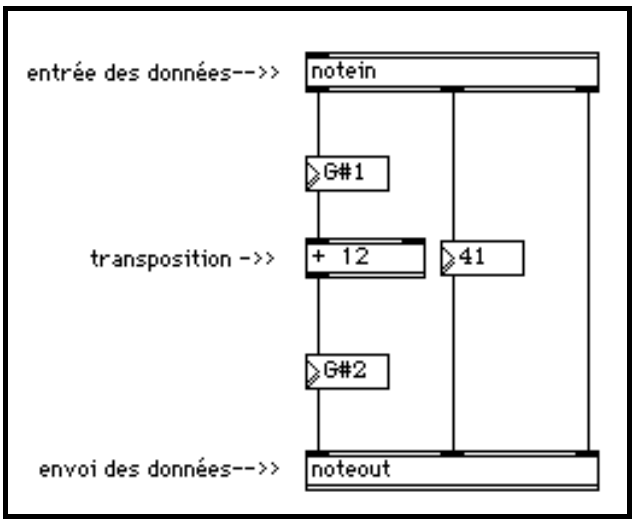

Figure 61 **:** *Exemple d'un* patch *MAX*

Au début des années 80, Miller Puckette, qui travaillait au MIT (Massachusset Institut of Technology) avec Max Mathews a défini les concepts de ce qui deviendra le logiciel MAX[35] [DUDAS 1997]. Ce premier environnement s'appelait « The Patcher » [PUCKETTE 1988].

L'environnement MAX, tel que nous le connaissons aujourd'hui, procède d'une longue évolution qui a commencé entre 1987 et 1988 quand Miller Puckette a commencé à travailler à l'Ircam [MANOURY 1988]. Le premier projet était destiné à la création d'une interface graphique pour contrôler le processeur numérique de signal en temps réel, la 4X [DI GIUGNO&KOTT 1981]. Parallèlement, le compositeur Philippe Manoury composait sa pièce « Pluton » pour piano et 4X et essayait de tirer le plus grand parti des possibilités proposées par ce nouvel environnement de contrôle. Ce travail d'étroite collaboration entre le chercheur et le compositeur a eu une grande influence sur la conception et le design final de cet environnement.

> « Tandis que je commençais les premières expériences avec un système de boîtes et de lignes qui est devenu MAX, Philippe Manoury posait les bases d'une pièce pour piano et synthétiseur... Pendant que je développais MAX, Philippe commençait à l'utiliser dans sa pièce. [...] C'est ainsi que le développement de MAX suivait en grande partie les besoins de ce qui est maintenant appelé Pluton, pour piano et 4X. » [MANOURY 1988, Préface]

Ensuite, Miller Puckette a continué à développer MAX sur la plate-forme informatique Macintosh. MAX sera par la suite distribué par Opcode Systems[36] et son développement continué par David Zicarelli.

Avec l'arrivée de la Station de travail Ircam [LINDERMANN&al 1991], Miller Puckette [PUCKETTE 1991] commence à porter MAX pour l'environnement informatique NeXT. La finalité étant d'obtenir une interface unique pour décrire les flux de contrôle et de signal, puisque historiquement MAX était prévu pour du contrôle MIDI (la 4X ne pouvant communiquer avec un ordinateur Macintosh que par une connexion série MIDI).

De nos jours MAX compte un grand nombre d'objets externes, des modules programmés en langage « C », permettant la manipulation de toutes sortes de données et le contrôle de périphériques connectés à l'ordinateur. Il intègre une bibliothèque audio, MSP, par David Zicarelli qui permet le traitement de signal audio en temps réel.

MAX a actuellement une position assez ambiguë en ce qui concerne sa fonction. En dépit de sa vocation d'outil destinée au contrôle de processus en temps réel lors de concerts et de performances, cet environnement est beaucoup utilisé pour la composition algorithmique et pour la génération de matériau musical ou de contrôle. Par exemple, le compositeur Pierre Jodlovsky l'a utilisé pour la génération de fichiers de contrôle de synthèse dans l'environnement Csound pour la composition de sa pièce « Dialog/No Dialog » pour Flûtes et électronique créée en 1997 à l'Ircam dans le cadre de l'atelier Cursus de composition et d'Informatique musicale. Plusieurs autres compositeurs l'utilisent aussi comme outil pour le temps différé dans le calcul de structures musicales

[35] Le nom »MAX » est une dédicace à Max Mathews, qui en plus d'être un des pionniers de l'informatique musicale est aussi l'auteur de Music V, un des premiers environnements logiciels pour la synthèse sonore.
[36] Situé en Californie, USA

diverses. Par exemple, Marc Battier dans «Mixt-Média », 1991, pour clarinette et bande, et «Toca » pour marimba et bande, 1993 ; Karleinz Essel dans « Amazing maze », 1998 ; Richard Dudas dans «Számítógép Zenéje 3,0 », 1991, etc.

De nombreux compositeurs ont trouvé en MAX une solution au problème général du calcul de structures musicales. Ce logiciel, conçu comme une interface de contrôle, est utilisé pour des tâches telles que l'aide à l'écriture.

Max a été le premier environnement grand public proposant un langage graphique de programmation appliquée à la musique et rendant possible une interactivité par la possibilité d'entendre une grande partie du matériau généré par des dispositifs MIDI, ou de voir des processus se déroulant à l'écran en temps réel, et une grande flexibilité d'utilisation. Ce type d'environnement graphique de programmation, permet une approche plus intuitive de la programmation par la matérialisation sur l'écran d'opérations abstraites en tant qu'objets graphiques connectés entre eux. Cependant, il manque une représentation symbolique de la notation musicale, ce qui rend difficile la manipulation du matériau généré à des fins de composition.

D'un point de vue informatique la gestion des types en MAX est encore assez primitive. Les types informatiques qu'il gère se bornent aux types standards, soit des nombres, des caractères, des chaînes de caractères et des listes. Il manque la possibilité de pouvoir manipuler des structures informatiques de plus haut niveau comme les objets informatiques. L'imaginaire du compositeur se trouve limité lors de la gestion d'éléments très simples. Etant un logiciel dépendant du temps, la synchronisation d'un grand nombre de données se trouve limitée à un niveau de complexité assez bas. Toutefois, ceci n'empêche pas que bon nombre d'utilisateurs l'utilise avec beaucoup d'adresse pour arriver à leurs fins.

III.4.6. Elody

L'environnement « ELODY » a été mis au point au GRAME. Il est basé sur le concept d'environnement pour l'écriture musicale [37], et des rapports entre le calcul et les manipulations musicales.

> « Les programmes définis dans Elody sont des expressions musicales généralisées qui sont obtenues en rendant variable des parties d'une expression existante. Ces abstractions peuvent être appliquées sur d'autres objets pour produire de nouveaux résultats ou composées pour créer de nouveaux programmes. Ainsi il n'y a pas de réelles distinctions entre les programmes et les objets musicaux. Les programmes sont seulement des objets musicaux dont certaines parties sont variables. Ils peuvent être joués, visualisés comme les autres objets musicaux». [ORLAREY&all 1998]

Le principe de manipulation d'Elody est innovant, il n'y a plus de programmation textuelle, ni de boîtes à connecter entre elles. Le principe général d'Elody consiste à construire des objets musicaux en combinant des objets simples (notes ou silence) ou des programmes. L'interface est basée sur le système « glisser/déposer » et sur les représentations visuelles des différentes fonctionnalités. Chaque action de l'utilisateur donne un résultat sonore et graphique immédiat [ORLAREY&all 1998].

[37] Voir [ORLAREY&all 1997a], [ORLAREY&all 1997b] et [ORLAREY&all 1998].

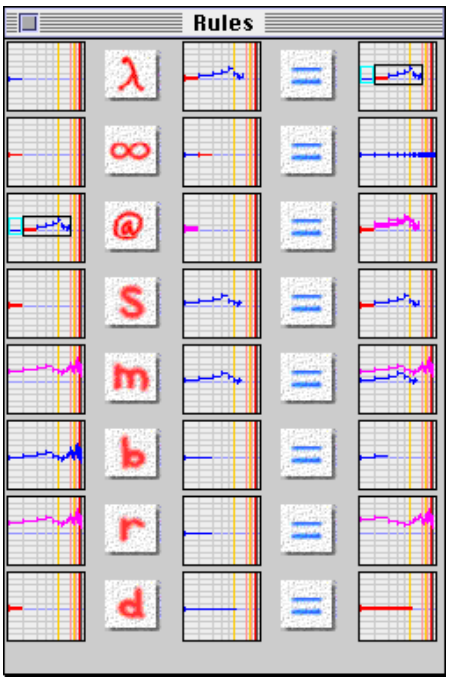

Figure 62 **:** *La fenêtre du « constructeur de règles » d'Elody*

III.4.7. Patchwork et OpenMusic

C'est sans doute avec Patchwork (PW) [LAURSON 1996] que l'aide à l'écriture a fait un saut qualitatif important en ce qui concerne les solutions graphiques. Le point fort de Patchwork est de proposer une représentation pour les processus musicaux et pour la notation musicale. L'environnement graphique est ouvert et flexible. L'utilisateur peut le personnaliser par le développement de fonctions et de librairies personnalisées.

Avec OpenMusic (OM) la représentation musicale des événements a progressé. En plus d'un ensemble de représentations unifiées (les représentations de hauteurs, rythmes, séquences MIDI et de fichiers de son numérisés), chaque compositeur peut « masquer » cette représentation par des notations personnelles. OpenMusic propose également des solutions graphiques à la représentation de processus par le biais d'un « éditeur de maquettes ». Il est désormais possible de représenter l'articulation d'un morceau musical ainsi que certaines relations logiques entre les diverses articulations.

III.5. Conclusion

Les environnements d'aide à l'écriture sont conçus comme des langages informatiques ou comme des extensions de ceux-ci. La tendance actuelle est aussi à la création de langages spécialisés[38]. Certains environnements tels que Symbolic Composer, HMSL et Common Music, sont plutôt des langages informatiques avec des fonctionnalités ajoutées, ils ne changent pas l'interface, ni les moyens d'accès. Les propositions d'interfaces graphiques se bornent à de simples visualisations ou à des éditeurs de fonctions par segments ou à des possibilités de choix d'opérations par des menus. Plusieurs de ces environnements sont très bien réalisés et très cohérents d'un point de vue informatique. Il leur manque toutefois une interface graphique afin d'accélérer les processus d'apprentissage et d'utilisation de l'environnement.

Une réflexion plus générale est nécessaire en ce qui concerne l'interface de contrôle. La volonté de chercher de nouvelles possibilités pour la notation est montrée dans la documentation de l'environnement Symbolic composer mais celles-ci vont à l'encontre de la pratique musicale instrumentale. Cela peut poser un problème au compositeur habitué à communiquer avec une notation qui englobe une bonne partie de la notation musicale symbolique classique. Il existe un décalage entre le monde de la musique et une partie du monde de l'informatique musicale qui ignore la réalité de la pratique instrumentale.

A part Patchwork, OpenMusic et Elody, aucun logiciel offre des solutions graphiques plus générales pour la représentation musicale ou pour la représentation des processus. La représentation se borne à des descriptions

[38] Certains environnements d'aide à l'écriture très particuliers, tels que le BP2 de Bernard Bel [BEL 1997], proposent en effet un langage fondé sur un modèle formel de grammaires génératives.

textuelles ou à des représentations graphiques cartésiennes, les abscisses représentant le temps et les ordonnées les hauteurs. Bien que la représentation proposée par Patchwork et OpenMusic soit fondée sur la représentation symbolique classique, elle permet d'avoir un repère plus général.

Les outils d'aide à la composition doivent permettre au compositeur de réutiliser les étapes précédentes de son travail, de définir des structures et des hiérarchies dans lesquelles le matériau de bas niveau peut être travaillé, ou de définir des formes et des structures permettant de produire le matériau musical. La capacité de choisir le niveau d'accès aux objets musicaux (par exemple une note, un temps, une mesure, une section, une partie de la pièce) permet de travailler sur le matériau musical ainsi que sur la forme musicale de façon continue. On attend en général d'un environnement de composition assistée par ordinateur qu'il soit capable de communiquer avec d'autres programmes pour l'édition, l'exécution en concert, la notation musicale ou la synthèse sonore. Enfin, un environnement de composition assistée par ordinateur doit être extensible. Il doit fournir des outils permettant aussi bien de créer et de tester des structures d'objets, que ceux destinés à créer et à personnaliser de nouveaux concepts et de nouveaux formalismes. En bref, ce qui différencie un environnement de composition assistée par ordinateur des autres programmes musicaux tels que les séquenceurs ou les programmes de synthèse, est le fait qu'il devrait être un outil capable de gérer et d'interagir avec la connaissance musicale à tous les niveaux.