III. LE PROGRAMME PATCHWORK ET SON UTILISATION POUR LE CONTROLE DE LA SYNTHESE SONORE

Dans ce chapitre, nous allons décrire le programme PatchWork (PW), environnement d'aide à la composition — ou de CAO, Composition Assistée par Ordinateur — et indiquer les possibilités qu'il offre pour le contrôle de la synthèse sonore.

III.1. Les langages informatiques pour la composition

III.1.1. Le langage Lisp

Pour qu'une machine effectue des opérations déterminées, il faut lui donner des instructions précises. Le système permettant de transmettre des informations à une machine et d'interpréter les résultats est constitué par un ensemble de caractères, de symboles et de règles permettant de les assembler. C'est ce que l'on appelle un langage.

Les langages informatiques peuvent être classés en fonction de différents critères. On distingue les **langages textuels** des **langages graphiques**. Les langages textuels utilisent un support linéaire, le texte, comme moyen de programmation alors que les langages graphiques permettent la programmation sur deux dimensions ou plus.

Une autre méthode de classification est couramment utilisée. Les langages « impératifs » sont des langages avec lesquels le programmeur écrit des instructions élémentaires pour décrire une tâche. Chaque instruction correspond à une action spécifique. Le langage ***Assembleur*** fait partie de cette catégorie. Les langages « procéduraux » sont un sous-ensemble des langages « impératifs ». Ils permettent de contrôler la complexité d'un programme par un regroupement des opérations en procédures. Ces procédures peuvent être appelées plusieurs fois à l'intérieur d'un même programme. Le ***FORTRAN***, le ***C*** et le ***Pascal*** sont des langages « procéduraux ». Dans les langages « fonctionnels », les actions sont divisées en sous-unités que l'on appelle des fonctions. Ces fonctions sont indépendantes de l'état global du système. C'est pour cette raison que ces langages permettent de comprendre et de construire plus facilement des programmes complexes. Le langage ***Lisp*** appartient à cette catégorie. Les langages « déclaratifs » ou « descriptifs » sont destinés à la résolution de problèmes complexes. Il ne s'agit plus de programmer pas à pas un algorithme mais plutôt de décrire l'état du système auquel on souhaite parvenir. ***Prolog*** est un langage de ce type. Enfin, dans les langages « orientés-objets », les concepts de données et de procédures sont remplacés par ceux d'objets et de méthodes. Un objet est un ensemble de données et la description de sa manipulation. Une méthode est la spécification d'une des manipulations de l'objet. Selon l'objet auquel elle se rapporte, une méthode peut effectuer des opérations différentes. ***CLOS***, **C++** et ***Smalltalk*** sont des langages « orientés-objets ».

Lisp est l'un des principaux langages de l'Intelligence Artificielle (IA). Il connaît un regain d'utilisation lié aux travaux réalisés sur les ordinateurs de la génération actuelle. La programmation symbolique, caractéristique de ce langage, est aujourd'hui au centre des préoccupations de nombreux chercheurs engagés dans des projets internationaux. Les recherches en IA et musique se rapportent à la nature de la connaissance et du savoir musical. Plusieurs questions servent de points de départ : quelles sont les connaissances musicales et quelle est

l'intelligence musicale ; le savoir musical est-il une compétence humaine autonome ou une collection de compétences indépendantes assemblées par l'histoire ; la compétence musicale est-elle différente des compétences linguistiques ou d'autres types de compétences ; la compétence musicale dépend-elle des périodes historiques et des cultures ou est-elle universelle ; comment les relations s'établissent-elles entre le savoir musical et l'action musicale ?

> « Il y a des différences entre le fait de décrire des connaissances musicales et celui de les formaliser, par exemple pour l'utilisation en informatique.
>
> Ce qu'on entend généralement par « science informatique » correspond en général à quelque chose de très technique. Or dans certains domaines, comme la sémantique de programmation, le formalisme est souvent prématuré, inutilement limité en conception, et inadéquat pour exprimer certains aspects de l'informatique.
>
> Par exemple, la logique mathématique joue un rôle déterminant dans les formalisations contemporaines mais est pour l'instant très faible pour exprimer la connaissance heuristique nécessaire pour développer en Intelligence Artificielle. La formalisation est une sorte de description d'un état basée sur certaines interprétations spécifiques. […]
>
> Les méthodes logiques sont basées sur l'utilisation de règles rigides d'inférence pour réaliser des déductions. Cela fonctionne bien dans les domaines formalistes, ces mondes artificiels qui restent imaginaires. Pour couvrir des domaines inconnus et incertains de la réalité, nous devons baser nos actions sur l'expérience et cela nous demande de raisonner par analogies. Il est donc nécessaire de pouvoir corréler deux expériences similaires et de comprendre quelles différences peuvent apparaître comme significatives. » Minsky dans [Balaban, Ebcioglu, Laske - 1992]

Issu des travaux de John Mac Carthy vers 1958, Lisp fut développé au MIT. Il a été graduellement amélioré, pour aboutir en 1962 à la version *Lisp 1.5* qui a longtemps servi de référence et de modèle théorique. En 1978, la version *Scheme* de Guy Steele et Gerald Sussman a introduit des idées nouvelles sur la liaison lexicale et l'intégration de l'environnement de programmation en Lisp. Ceci a influencé les efforts de rationalisation des divers dialectes Lisp et est à l'origine de *Common Lisp* paru en 1984. Common Lisp dispose d'une extension permettant la programmation orientée objet, il s'agit du système « CLOS » (Common Lisp Object System).

Lisp fait partie des langages de programmation fonctionnels. Il se présente à l'utilisateur comme un « interprète ». Il lit ce qui est tapé au clavier, en interprète le sens, réalise une opération et retourne directement un résultat. Les outils de mise au point et d'édition, la souris et le « multi-fenêtrage », donnent à la programmation interactive tout son sens.

De façon assez unanime, les informaticiens se sont tournés vers le langage Lisp pour développer des outils destinés aux compositeurs (Formes[1], Common Music[2], Common Lisp Music[3], Modalys[4], PatchWork, OpenMusic[5], Holophon[6]). Plusieurs raisons en font un outil de travail apprécié pour l'aide à la composition : il a une syntaxe uniforme ; les données et les fonctions sont représentées de la même façon ; la gestion de la mémoire est automatique ; il est flexible ; il permet différents styles de programmation et offre plus de 700 fonctions prédéfinies. Lisp est un langage facile à apprendre par des débutant en informatique.

[1] Premier programme de l'Ircam pour l'aide à la composition (1984) écrit par Xavier Rodet et Pierre Cointe en langage Le-Lisp [Rodet-Cointe – 1985].
[2] Programme écrit par Heinrich Taube (1991) en Common Lisp [Taube-1989].
[3] Common Lisp Music, écrit par Bill Schottstaedt en 1991 en Common Lisp à l'université de Stanford est un programme spécialisé dans la synthèse sonore. Il fonctionne sur ordinateur NeXt, Macintosh, SGI, Sun et les machines Intel sous NeXTStep [Roads-1996, p. 790].
[4] Modalys, anciennement Mosaic, programme Ircam pour la synthèse sonore par modèles physiques, écrit par Jean-Marie Adrien (1991) puis porté sur Macintosh par Adrien Lefevre. Ce programme est écrit en Scheme, dialecte du langage Lisp [Adrien – 1990].
[5] OpenMusic est le successeur de PatchWork [Agon-1998].
[6] Holophon est un éditeur algorithmique et graphique pour le contrôle de la spatialisation de sources sonores indépendantes ou de tout autre paramètre musical [Pottier-2000].

Depuis peu, le langage Java s'impose comme alternative au langage Lisp. A mi chemin entre les langages Lisp/CLOS et C++, il offre l'avantage d'une parfaite portabilité sur les différentes plates-formes et est spécialisé dans le travail à distance par le réseau internet.

Parmi les environnements écrits en Java, signalons les programmes « jmax », « Elody » ou « MIDISpace ». « jmax », de l'Ircam [Dechelle—1999], est une nouvelle version de Max, totalement réécrite et destinée à la synthèse et au traitement de signal, « Elody », est l'environnement de composition musicale du GRAME [Orlarey, Fober, Letz-1999] et « MIDISpace » est un programme pour le contrôle interactif par contraintes de la spatialisation du son [Pachet, Delerue-1998].

III.1.2. La programmation « objet »

La programmation informatique fonctionne avec des éléments qui permettent de constituer des programmes. On peut combiner des procédures et des données primitives pour construire de nouvelles procédures. On peut également utiliser des **abstractions** pour construire des structures emboîtées permettant d'affronter la complexité de grands systèmes. Ces outils ne suffisent pas à élaborer des programmes. La construction effective des programmes demande des principes d'organisation. En particulier, il est nécessaire de mettre en oeuvre des stratégies pour aider à structurer de grands systèmes, afin de les rendre « modulaires ». Cela consiste à les diviser « naturellement » en éléments cohérents pouvant être développés et mis à jour séparément.

Par exemple, une de ces stratégies est particulièrement adaptée à la construction de programmes modélisant des systèmes physiques. Elle consiste à calquer la structure des programmes sur celle du système à modéliser. Un « objet informatique » est associé à chaque objet du système et une opération symbolique correspond à chaque action. Avec une telle méthode, le modèle peut être étendu à de nouveaux objets ou à de nouvelles actions sans que la stratégie choisie soit remise en cause, si ce n'est par l'addition de nouvelles représentations symboliques de ces objets ou de ces actions. Ainsi, un ajout ou une mise au point ne fait intervenir que sur une partie limitée du système.

La stratégie fondée sur les « objets » considère un grand système comme un ensemble d'objets distincts dont le comportement est à même d'évoluer dans le temps. L'approche par objets est une approche qui a un grand potentiel pour l'organisation des modèles informatiques. Ces modèles sont modulaires. Chaque objet informatique possède ses propres variables d'état locales décrivant son état interne. Ces variables doivent pouvoir évoluer. Si le temps est simulé par la séquence des opérations effectuées par la machine, le comportement des objets doit évoluer au fil des opérations.

III.2. PatchWork, langage pour la composition assistée par ordinateur

PatchWork, est un langage de programmation graphique écrit à partir des langages Common Lisp et CLOS (Common Lisp Object System) pour ordinateurs Macintosh. PatchWork est spécialisé dans la production et l'analyse de matériaux musicaux. C'est un langage unique dans ce domaine car il combine plusieurs avantages : c'est un langage basé sur la programmation orienté-objet et sur la programmation graphique qui dispose d'éditeurs sophistiqués, permettant la notation musicale. Il permet la construction d'objets musicaux complexes et il est facilement extensible.

La richesse de PatchWork tient, en outre, au fait qu'il intègre des années de recherches et d'expériences dans le domaine de la « Composition Assistée par Ordinateur ».

III.2.1. Évolution des outils pour la CAO

La première utilisation de l'informatique pour la composition est attribuée à Lejaren Hiller pour la pièce « Illiac Suite for String Quartet » programmée en 1956 sur l'ordinateur Illiac à l'université de l'Illinois. Les données sont générées en grand nombre par des techniques aléatoires utilisant l'algorithme de Monte-Carlo puis soumises à l'ensemble des règles du contrepoint [Fatus—1989].

En France, la première pièce réalisée par ordinateur est « Factorielle 7 » (1960) de Pierre Barbaud [Barbaud—1993]. Il utilise la théorie des ensembles, celle des processus aléatoires et la théorie des automates finis et des matrices stochastiques.

Michel Philippot et André Riotte ont amené l'école française à une place importante dans le développement de la composition assistée par ordinateur [Fatus—1989].

Dans les années 50, Iannis Xenakis a proposé pour la distribution des paramètres musicaux l'utilisation des lois de probabilités, comme alternative aux démarches sérielles. Depuis, il a écrit de nombreux programmes pour l'écriture de ses pièces [Xenakis—1981], allant jusqu'à programmer par des lois stochastiques l'onde sonore elle-même [Serra—1993]. Par ailleurs, il a réalisé un système de composition — pouvant produire à la fois des notes et des sons — à partir de techniques uniquement graphiques : l'Upic.

Depuis les premières recherches en CAO, de nombreux outils ont été mis au point : composition fonctionnelle, composition par contraintes, composition automatique [Loy-1989].

Devant le développement des outils portant la dénomination Composition Assistée par Ordinateur, il faut établir une distinction entre les outils de composition automatique et ceux pour l'aide à la composition. Les outils de composition automatique intègrent les éléments du langage musical qu'ils permettent seulement de mettre en pratique alors que les outils pour l'aide à la composition doivent servir à mettre au point ces éléments.

Dès le début des années 80, l'Ircam a commencé à travailler sur la représentation et la manipulation des structures et des connaissances dans le domaine de la composition et de la synthèse sonore. Le synthétiseur « Chant » a été associé à un outil permettant au compositeur de gérer l'appel aux bases de données et de contrôler dans le temps, de façon automatique en utilisant des règles, l'évolution des paramètres de synthèse. C'est dans ce but que, dès le début des années 80, « Formes » a été mis au point à l'Ircam.

« Formes » est un système interactif, écrit en langage Lisp (VLisp) par Pierre Cointe et Xavier Rodet en 1984 et qui est destiné à la composition musicale et à la synthèse sonore [Rodet, Cointe—1985]. L'architecture de « Formes » consiste en une hiérarchie d'objets, ou programmes, organisée dans le temps. Ces objets sont structurés en une arborescence dont les couches supérieures correspondent aux règles les plus abstraites de la synthèse[7]. Pour produire la synthèse, l'arborescence est évaluée à chaque instant et les différentes règles conduisent à la production d'un ensemble de paramètres qui peuvent être envoyés au synthétiseur Chant.

Le but d'un programme comme Formes est d'une part d'autoriser différents degrés de complexité sans arriver à saturation et d'autre part de permettre l'ajout de nouveaux éléments à un modèle développé préalablement. Seule la

[7] Dans Formes, les couches inférieures sont plutôt destinées à des objets définissant les valeurs des paramètres comme la fréquence fondamentale ou les fréquences des formants. Au-dessus on peut trouver des objets décrivant des modes de jeu comme le vibrato ou l'articulation, ensuite viennent des objets décrivant, par exemple, le style de la voix. Les niveaux supérieurs contiennent les règles relevant de la composition.

programmation par objet offre une discipline de programmation qui peut permettre cela.

Parmi les bibliothèques existant au sein de l'environnement « Formes », « Esquisse » est un programme spécialisé dans l'aide à la composition conçu par Pierre-François Baisnée et Jacques Duthen. Ce programme regroupe de nombreuses fonctions utilisées par des compositeurs ayant travaillé à l'Ircam (Magnus Lindberg, Marc-André Dalbavie, Jean-Baptiste Barrière, Kaija Saariaho). Ces fonctions sont classées en fonction du type de connaissances musicales qu'elles représentent : fonctions orientées intervalles, fonctions de traitement harmonique, interpolations.

> « Esquisse a donné de la force à la démarche fonctionnelle et l'on a très vite vu que ce paradigme pouvait être appliqué de manière homogène au domaine de la synthèse et de la CAO. » [Agon -1998, p. 12]

En 1984, Apple France a encouragé l'Ircam à développer des programmes pour les ordinateurs Macintosh. Le Macintosh apparaissait alors comme un ordinateur prometteur mais ne possédait pas d'outil de développement en langage de haut niveau, ni d'interface MIDI. En décembre 1984, une version du langage Le_Lisp est apparue sur Macintosh. En janvier 1985, Camillo Rueda entrepris le portage du programme Formes sur Macintosh puis quand en 1985 un premier prototype d'interface MIDI fut disponible, il mit au point une interface entre Le_Lisp et les instruments MIDI, ouvrant la voie vers des contrôles en temps réel.

En 1986, plusieurs personnes se sont jointes à l'équipe de recherche de l'Ircam : Pierre Lavoie de Act Informatique, Yann Orlarey du studio GRAME de Lyon qui avait développé le programme MIDI-LOGO pour Macintosh et Lee Boynton qui développa avec Jacques Duthen une interface graphique de Formes pour Macintosh, PREFORM [Boynton, Lavoie, Orlarey, Rueda, Wessel-1986]. Lee Boyton développa également un « scheduler » permettant de produire des processus fonctionnant en temps réel en parallèle.

A la même période, plusieurs compositeurs, dont Magnus Lindberg, ont commencé à développer des outils de représentation musicale en Lisp pour la notation traditionnelle de la musique.

PatchWork a été conçu et réalisé en Finlande par Mikael Laurson avec Common Lisp sur Macintosh en 1990. Les développements ultérieurs ont été réalisés à l'Ircam par Jacques Duthen et Camillo Rueda [Duthen, Laurson—1990] puis par Gérard Assayag [Assayag, Rueda-1993] et Carlos Agon. Ils ont dans un premier temps intégré à PatchWork l'essentiel des outils de CAO de l'Ircam puis ont mis au point de nouvelles fonctionnalités et de nouvelles bibliothèques.

Le paradigme de PatchWork est basé sur la programmation fonctionnelle réalisée à travers la construction de patchs[8] graphiques. L'utilisateur peut également avoir accès à l'environnement de programmation, de type texte, de Common Lisp et CLOS.

En 1990, une interface pour le contrôle du synthétiseur Chant par PatchWork a été écrite par Francisco Iovino [Iovino, Laurson, Pottier—1994].

PatchWork comporte un ensemble d'outils destinés à aider le compositeur dans son travail « précompositionnel ». Il permet de programmer des algorithmes, ou « patchs », destinés à créer et à manipuler le matériau musical représenté par des notes, des rythmes, des enveloppes ou des structures formelles. Il permet également de contrôler différents programmes de synthèse sonore. Il peut s'agir

[8] Le terme « patch », provenant de la langue anglaise, restera employé dans la suite du texte pour désigner un assemblage de modules graphiques. Il n'y a pas de traduction satisfaisante, c'est pourquoi ce terme technique est couramment employé en français.

de synthétiseurs MIDI ou de logiciels de synthèse comme Csound et Chant. PatchWork peut communiquer avec de nombreux programmes - d'analyse, de traitement, de synthèse, de séquences, d'édition de partition – et sous différentes formes - fichiers texte, MIDI, binaire, etc.

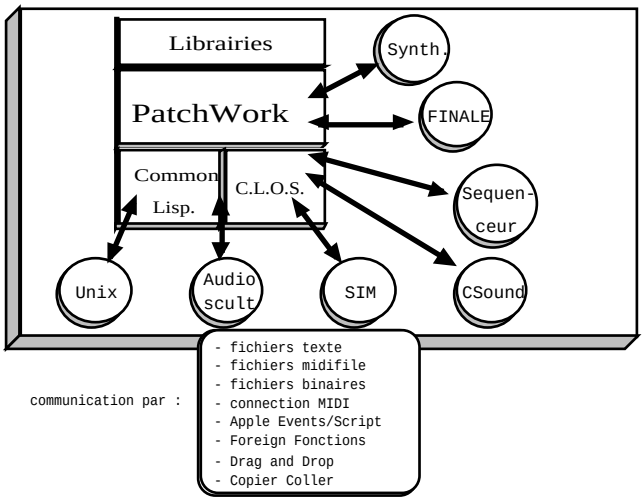


figure III–1 : diagramme représentant l'environnement PatchWork et ses relations possibles avec les autres outils du studio informatique.

III.2.2. Les outils graphiques de PatchWork

Le programme PatchWork offre au langage Lisp une interface graphique.

La programmation dans PatchWork s'effectue de façon graphique à l'intérieur de plusieurs types de fenêtres qui ont chacune un comportement différent.

La fenêtre principale sert à créer et à éditer des « patchs ». Pour cela, l'utilisateur dispose de bibliothèques de modules prédéfinis incluant des objets musicaux comme les notes, les accords, les séquences ou les rythmes. La programmation s'effectue en disposant les modules sur la fenêtre et en les connectant entre eux.

Pour connaître le résultat produit par un patch, il faut l'évaluer selon un processus de type « demand-driven ». Cela signifie que le module pour lequel on va demander l'évaluation va « interroger » ses différentes entrées pour connaître les valeurs des données dont il a besoin pour effectuer une opération. Ce processus entraîne une interrogation du patch effectuée du bas vers le haut.

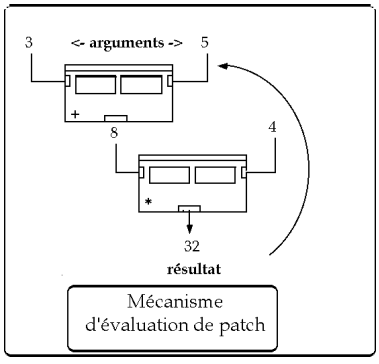


figure III–2 : mécanismes d'évaluation d'un patch avec le programme PatchWork.

L'éditeur de fonctions par segments « BPF » permet de créer ou d'éditer point par point des fonctions graphiques linaires par segments. Ces fonctions peuvent être stockées dans des bibliothèques pour être rappelées ultérieurement.

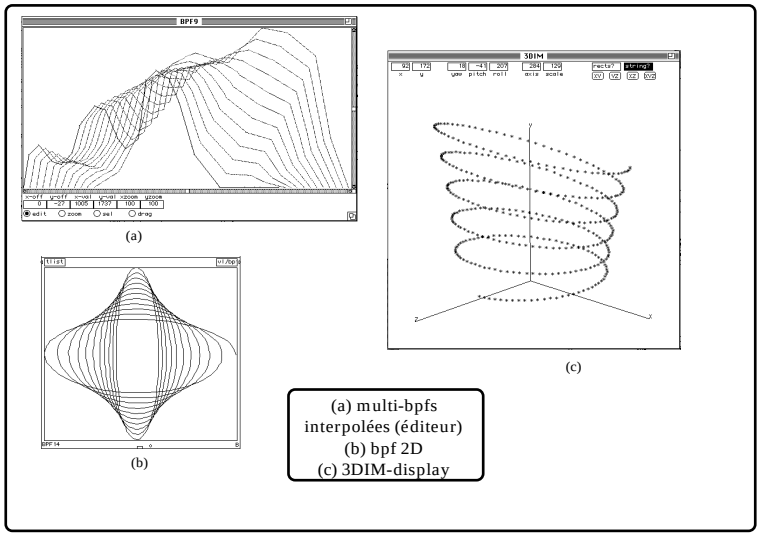


figure III–3 : représentations graphiques réalisées avec l'éditeur de BPF (a) et (b) ou avec le module 3dim-display (c).

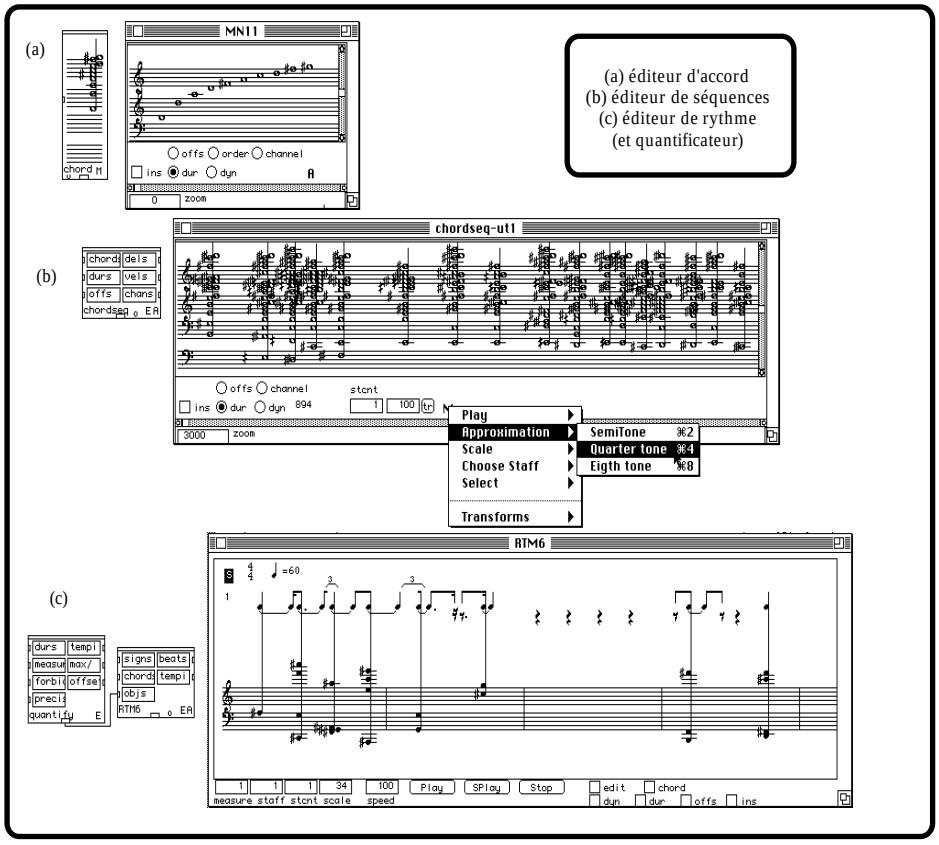


figure III–4 : visualisation des notes dans les trois principaux éditeurs de notation musicale de PatchWork.

Plusieurs éditeurs de notation musicale sont disponibles dans PatchWork et sont représentés dans la figure III–4.

L'éditeur d'accords (a), donné par le module spécifique « chord », permet de créer et d'éditer les différents paramètres d'un accord.

L'éditeur de notation musicale « chordseq » (b) contient les informations décrivant des séquences d'accords. Il permet d'éditer des zones sélectionnées.

L'éditeur de rythme « RTM » (c) sert à visualiser des rythmes à l'aide d'une notation fractionnaire.

III.2.3. Exemples d'utilisation de PatchWork chez Tristan Murail

Les exemples choisis décrivent des processus mis en jeu dans la pièce « Gondwana » (1980) de Tristan Murail. Tristan Murail utilise couramment l'informatique pour la composition de ses pièces et il a beaucoup contribué à la mise au point du programme PatchWork, étant à l'origine des algorithmes spectraux de la bibliothèque « Esquisse ». Les principaux processus utilisés en 1980 par Tristan Murail pour la construction de « Gondwana » n'avaient pas à l'époque été écrits dans PatchWork qui n'existait pas encore. Par contre, ils ont été réécrits plus tard par le compositeur dans cet environnement à titre de documentation (cf. Annexe V).

A) L'algorithme de la FM pour générer l'harmonie

Tristan Murail a appliqué l'algorithme de la synthèse par modulation de fréquence pour la production des harmonies utilisées pour l'écriture de la partition d'orchestre de Gondwana. C'est le principe de la « **simulation spectrale** » consistant à décrire et à simuler les spécificités d'un spectre naturel ou artificiel[9]. L'algorithme de la modulation de fréquence permet la production d'accords complexes, non tempérés et inharmoniques qui ont été utilisés pour l'évolution harmonique de la pièce.

La modulation d'une fréquence f_p par une fréquence f_m produit des composants partiels de fréquence $f_p + if_m$, i étant un nombre entier positif ou négatif. Par exemple, la modulation de Fa#3 par la note Sol#2 donne la liste de notes qui suit[10] :

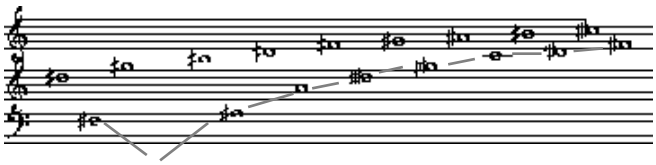


figure III-5 : hauteurs produites par une modulation de fréquence de Fa#3 par Sol#2 avec des phénomènes de repliement.

Le phénomène de double trajectoire observé dans la figure III-5 provient des phénomènes de repliement qui ont lieu lorsqu'un partiel généré par la modulation de fréquence prend une valeur de fréquence négative. Quand un partiel descend à une fréquence inférieure à zéro, il se replie sur la valeur positive opposée.

B) La production de séries harmoniques

Les notes de la figure III-6 sont obtenues en développant des séries harmoniques à partir de deux notes : Fa#2 et Sol#1. On a calculé les harmoniques dont le rang est donné par une série arithmétique — 3 5 7 9 11 13 15 17 19 — à laquelle on a ajouté la valeur 2. On obtient les harmoniques impairs des deux notes.

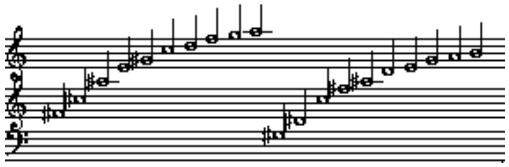


figure III-6 : séries harmonique défectives calculées pour « Gondwana » de Tristan Murail.

[9] Cette technique a été utilisée également par d'autres compositeurs comme Mesias Maiguashca pour sa pièce « FMelodies I » (1981), réalisée à l'Ircam avec la machine 4C ou Fausti Romitelli dans « En Trance » (cf. § IV-3-5),

[10] Dans PatchWork, les hauteurs de notes sont exprimées en MIDIcents, correspondant aux numéros de notes MIDI multipliés par cent afin d'obtenir une précision égale au centième de 1/2 ton ce qui peut paraître exagéré pour une écriture instrumentale mais se révèle parfaitement justifié dans le cas de la synthèse.

C) Les interpolations

Le rythme d'enchaînement des différents accords de la partition de Tristan Murail est calculé par un processus d'interpolation. Les durées des accords (en 1/100e de seconde) sont calculées par une interpolation dépendant de trois valeurs de référence — 300, 170 et 400 — fournies par le compositeur. L'interpolation est réalisée en utilisant le module de Lagrange qui calcule l'équation de la fonction polynomiale optimale qui passe par ces trois valeurs (cf. figure III-7). Il est ainsi possible de calculer les durées des accords intermédiaires.

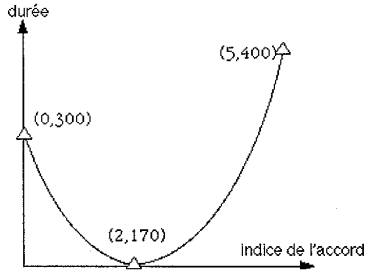


figure III-7 : tracé de la fonction de Lagrange passant par trois points.

Pour chacun des trois exemples, le résultat final peut être joué en MIDI dans l'éditeur d'accords « chordseq ». PatchWork permet de faire jouer des micro-intervalles en 1/4 et en 1/8e de ton, à condition de disposer d'un synthétiseur multitimbral acceptant des accordages fins sur les différents canaux MIDI. Dans ce cas, l'émission des notes se fait sur les différents canaux MIDI selon le tableau suivant :

hauteur	juste	+1/8e ton	+1/4 ton	+3/8e ton
canal MIDI	1	2	3	4

Nous allons maintenant nous intéresser à l'utilisation du programme PatchWork pour construire des sons de synthèse. Nous allons décrire de quelles façons le programme PatchWork peut être utilisé pour le contrôle de la synthèse réalisée avec le programme **Csound** puis nous étudierons ensuite une bibliothèque de PatchWork spécialisée dans la lecture et l'exploitation de **données spectrales** pour la synthèse et nous terminerons ce chapitre par l'étude de la bibliothèque **Chant** qui permet la synthèse de sons sur des modèles de voix chantée ou d'instruments résonants.

III.3. PatchWork et la synthèse sonore avec Csound

La production de sons à l'aide du programme Csound requiert l'écriture de deux documents, « l'orchestre » et la « partition » (cf. § II-1). **L'orchestre** décrit les fonctionnalités des « instruments » que l'on veut utiliser et la nature de leurs paramètres de contrôle. **La partition** contient des listes de valeurs numériques permettant le contrôle des instruments contenus dans l'orchestre.

Nous allons décrire l'utilisation du programme PatchWork pour la production de partitions pour Csound.[11].

Dans les partitions utilisées pour Csound, on rencontre principalement deux types de paramètres, les paramètres décrivant des tables et les paramètres des instruments appelés « pfields » ou « paramètres de notes ». D'autres paramètres, comme le tempo ou des variables globales, peuvent également y figurer.

Les tables sont définies par une déclaration « f » placée en début de ligne qui est suivie des paramètres décrivant la table. La nature et le nombre de ces paramètres varient avec le type de table utilisé.

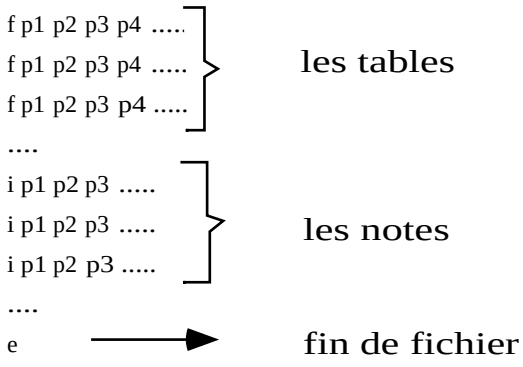
[11] Mikhail Malt et moi-même avons réalisé en 1995 une bibliothèque de PatchWork, intitulée « Csound / edit-sco », destinée à la production de partitions pour Csound [Malt, Pottier-1993].

```
f p1 p2 p3 p4 p5 p6 …
```

Les notes sont définies par une déclaration « i » située en début de ligne qui est suivie de paramètres dont la nature et le nombre dépendent du choix de l'instrument que l'on cherche à contrôler.

```
i p1 p2 p3 p4 p5 …
```

Une partition a généralement la structure suivante :



III.3.1. La production de partitions pour Csound avec PatchWork

A) Synthèse d'un spectre harmonique par synthèse additive

Nous avons choisi cette technique de synthèse en raison de son très large champ d'application qui, dans l'absolu, lui permet de reproduire tout type de son. Elle est basée sur l'assemblage de sons purs, sinusoïdaux, composants élémentaires du son. Pour produire des sons complexes, il est nécessaire de contrôler une grande quantité de paramètres. Pour la synthèse de sons harmoniques, le calcul peut être effectué avec une cinquantaine d'oscillateurs. Dans le cas de sons bruités, il est parfois nécessaire de faire appel à des milliers d'oscillateurs et d'exercer des contrôles très précis et très rapides sur les oscillateurs. La synthèse additive présente l'avantage de gérer des événements qui sont représentés dans un plan fréquence/temps correspondant à la fois à la perception auditive et à la représentation traditionnelle de la musique sur partition.

Pour additionner des ondes sinusoïdales, il suffit d'utiliser un instrument additif assez simple — *additif.orc*, cf. Annexe IV — et de lui adjoindre une partition décrivant la forme d'onde à utiliser et donnant les valeurs des paramètres - fréquence, amplitude, durées - de chaque partiel dans un tableau.

Avant de montrer comment PatchWork permet de produire des partitions pour Csound, nous indiquerons rapidement quels autres alternatives sont à la disposition du compositeur pour remplir cette tâche.

Lorsque le compositeur doit réaliser une partition d'une centaine de lignes de valeurs numériques, il est très contraignant pour lui d'avoir à fournir une par une toutes ces valeurs en utilisant un simple éditeur de texte. En dehors des outils spécialisés pour la musique comme PatchWork, les tableurs, outils de bureautique, permettent de réaliser des partitions de façon assez simple. Ils conviennent à l'écriture de partitions lorsque les relations liant les paramètres des partiels entre eux sont des relations arithmétiques simples. Ils permettent de créer et d'organiser les données en colonne.

; table décrivant une fonction sinus pour l'oscillateur					
f1	0	1024	10	1	
; partiels ou notes					
		durée	amplitude	fréquence	attaque
		(sec.)	(linéaire)	(Hz)	(sec.)
i1	0	3	3276,7	220	1
i1	0	3	3276,7	440	1
i1	0	3	3276,7	660	1
i1	0	3	3276,7	880	1
i1	0	3	3276,7	1100	1
i1	0	3	3276,7	1320	1
i1	0	3	3276,7	1540	1
i1	0	3	3276,7	1760	1
i1	0	3	3276,7	1980	1
i1	0	3	3276,7	2200	1

tableau III-1 : écriture de la partition « add.sco » à partir d'un tableur.

Le tableau III-1 correspond à une partition destinée à la création d'un son harmonique par synthèse additive, d'une fréquence fondamentale de 220 Hz. La plupart des données ont été calculées par des formules répétées automatiquement comme le montre le tableau III-2.

	A	B	C	D	E	F
1	; table décrivant une fonction sinus pour l'oscillateur					
2	f1	0	=2^10	10	1	
3	; partiels ou notes					
4			durée	amplitude	fréquence	attaque
5			(sec.)	(linéaire)	(Hz)	(sec.)
6	i1	0	3	=32767/10	220	1
7	=A6	=B6	=C6	=32767/10	=E$6*2	=F6
8	=A7	=B7	=C7	=32767/10	=E$6*3	=F7
9	=A8	=B8	=C8	=32767/10	=E$6*4	=F8
10	=A9	=B9	=C9	=32767/10	=E$6*5	=F9
11	=A10	=B10	=C10	=32767/10	=E$6*6	=F10
12	=A11	=B11	=C11	=32767/10	=E$6*7	=F11
13	=A12	=B12	=C12	=32767/10	=E$6*8	=F12
14	=A13	=B13	=C13	=32767/10	=E$6*9	=F13
15	=A14	=B14	=C14	=32767/10	=E$6*10	=F14

tableau III-2 : affichage des fonctions permettant d'écrire la partition « add.sco ».

Le son produit par Csound à l'aide de l'orchestre « additif.orc » et de la partition « add.sco » est représenté par le sonagramme de la figure III-8.

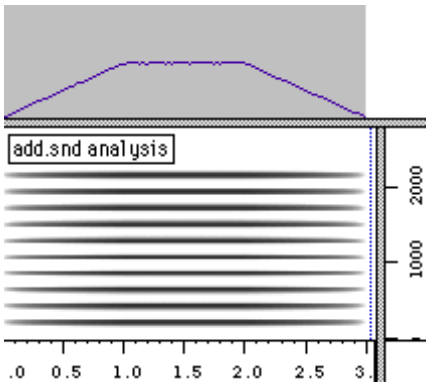


figure III-8 : « add.snd » réalisé avec l'orchestre « additif.orc » et la partition « add.sco » (exemple sonore n°22).

L'exemple présenté nous montre qu'un tableur permet la production de sons simples. Dans le cas de sons complexes, il est nécessaire d'utiliser des programmes spécialisés dans les applications musicales comme le programme PatchWork

Dans PatchWork, avec la bibliothèque *Csound/edit-sco*, le compositeur dispose d'un ensemble de modules destinés à programmer l'édition de partitions Csound, d'une façon conviviale et efficace. Cette bibliothèque permet de générer les tables et les valeurs des paramètres contenus dans les partitions par des **moyens**

graphiques, par le biais d'**algorithmes** et par l'édition de notes en **notation traditionnelle**.

Selon les utilisateurs de Csound, la partition et l'orchestre peuvent varier en degrés de complexité. Certains compositeurs préfèrent construire des orchestres très élaborés contrôlés par des partitions réduites. D'autres utilisent des orchestres simples qu'ils contrôlent dans la partition par de nombreuses informations. Nous décrirons des utilisations de Csound plutôt orientées vers le deuxième cas de figure.

PatchWork possède des modules qui permettent de formater des données numériques pour l'écriture de tables et de notes. Ainsi, la partition réalisée par un tableur dans le tableau III-1 peut être reconstruite à partir d'un patch simple dans PatchWork (cf. figure III-9). On part d'une note, La2, dont on calcule la fréquence, 220 Hz, avec l'objet *mc->f*, puis on calcule les dix premiers multiples de cette fréquence avec une fonction qui calcule des séries arithmétiques.

Le module *instrument2* est un module extensible dont chaque entrée correspond à une colonne dans la partition. L'entrée n°1 indique donc le numéro de l'instrument (1), l'entrée n°2 donne la date du début du son (temps = 0"), l'entrée n°3 donne la durée du son (3") qui sera la durée de chacun des partiels que l'on va additionner. Pour avoir des durées différentes pour chaque partiel, il faut donner une liste de valeurs, au lieu de donner une valeur unique. L'entrée n°4 donne l'amplitude des partiels (ici, l'amplitude maximum de l'onde divisée par le nombre de partiels), l'entrée n°5 donne la liste des fréquences et l'entrée n°6 donne le numéro de la table utilisée pour donner la forme d'onde de chaque partiels.

Le module *edit-sco2* produit le fichier partition en regroupant les données de type *note* et les données de type *table*. Les données de type *table* correspondent ici à la description d'une table sinusoïdale.

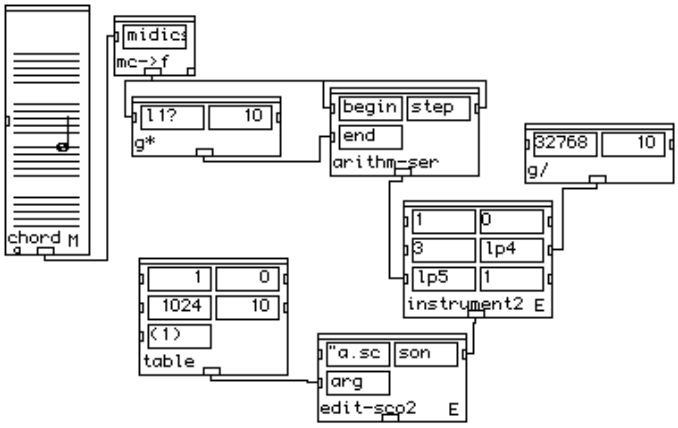


figure III-9 : patch pour l'écriture d'une partition Csound pour réaliser une synthèse additive avec l'instrument « additif.orc ».

B) Synthèse sonore à partir d'une partition traditionnelle

Le programme PatchWork est beaucoup plus souvent utilisé pour produire des partitions pour des instruments acoustiques que pour réaliser la synthèse de sons électroniques.

Lorsqu'un compositeur a réalisé une partition, par une quelconque programmation dans PatchWork, il peut l'écouter en la faisant jouer en MIDI sur un synthétiseur mais s'il a besoin d'une plus grande précision en fréquences, en amplitudes ou en durées, il peut synthétiser le son directement avec Csound en synthèse additive. Chaque note de la partition va correspondre à un partiel qu'il va synthétiser avec un instrument plus ou moins complexe.

Dans le but de tester rapidement des assemblages complexes de fréquences, nous avons conçu deux modules, (*xchds->sco-h* et *xchds-mn5sco*) qui permettent de

construire automatiquement des partitions pour la production de sons de synthèse à partir de partitions traditionnelles écrites dans PatchWork.

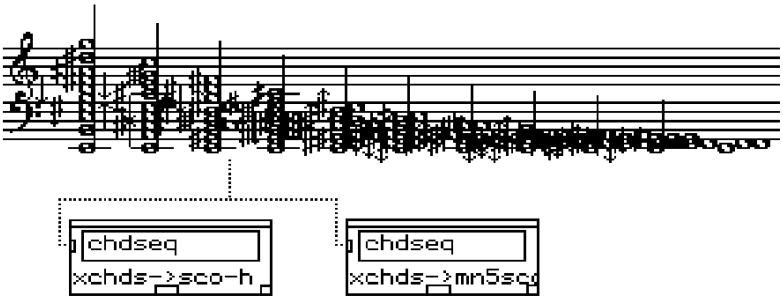


figure III–10 : modules pour l'écriture d'une partition Csound à partir d'une liste de notes.

Le premier module est destiné à l'écriture d'une partition permettant la synthèse d'un son pur pour chaque note figurant dans la partition. La synthèse réalisée par ce biais permet de s'affranchir de la notation tempérée. Les hauteurs peuvent être exprimées en MIDIcents avec une résolution du centième de demi-ton. L'orchestre « additif.orc », décrit au paragraphe précédent, peut être utilisé pour faire jouer cette partition. Le son produit est présenté dans la figure III–11.

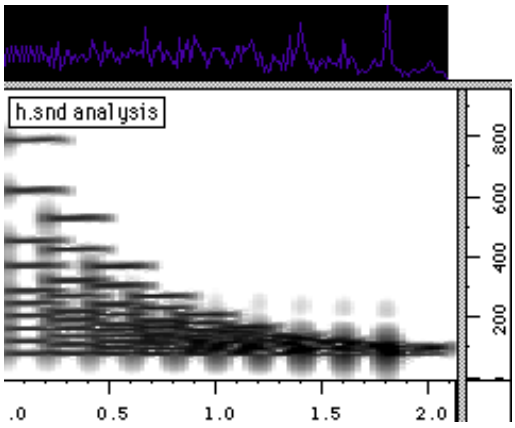


figure III–11 : son de synthèse produit par l'addition de sons sinusoïdaux à hauteurs fixes à partir d'une partition (exemple sonore n°23).

Le deuxième module (*xchds->mn5sco*) permet de créer une partition destinée à la synthèse d'un son continu à partir d'une partition. Ce son est formé de l'assemblage de plusieurs glissandos superposés dont les trajectoires passent par les différentes notes des accords successifs. Appliqué à la partition précédente (figure III–10), ce module produit une partition dont la synthèse donne le son représenté par la figure III–12.

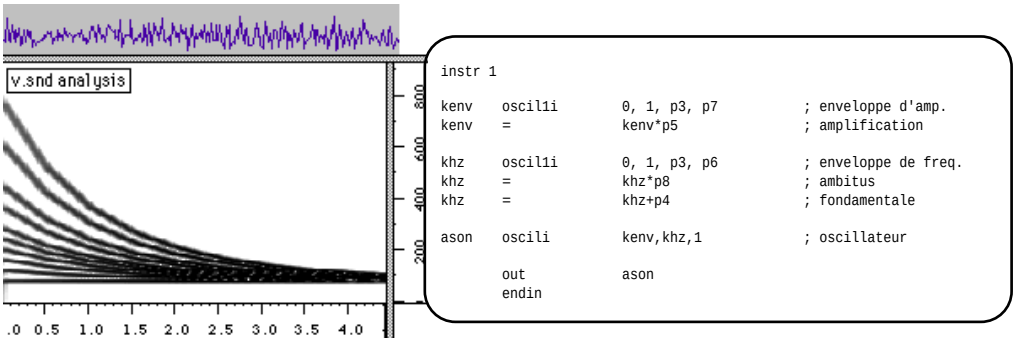


figure III–12 : à gauche, son de synthèse produit par l'addition de sons sinusoïdaux à hauteurs variables d'après une partition, à droite, orchestre « mn5.orc » correspondant (ex. son. n°24).

Avec ces deux modules, on dispose de moyens simples et rapides pour synthétiser des enchaînements d'accords ou des trajets de fréquences. Dans le

deuxième exemple, les accords donnés dans l'éditeur de partition doivent tous avoir le même nombre de notes afin de définir des trajectoires — note 1 de l'accord 1 vers note 1 de l'accord 2, ... Pour réaliser des sons de synthèse complexes, il est possible d'utiliser des partitions très élaborées.

La bibliothèque Csound/edit-sco offre beaucoup d'autres possibilités et permet de produire tous types de partitions, quel que soit le nombre de paramètres à contrôler. Il faut alors créer des patchs spécifiques en fonction des applications recherchées.

C) Synthèses sur un modèle percussif

Dans un but pédagogique, nous avons réalisé un exemple pour lequel ont été produits différents sons de synthèse, tous réalisés à partir de l'analyse d'un son de vibraphone effectuée par la méthode des modèles de résonance. Pour ces synthèses, nous avons écrit une sorte de partition indiquant différentes utilisations des données fournies par l'analyse. Ici, un ensemble de patchs simples permettent à partir de PatchWork de réaliser diverses transformations des données du modèles et d'aboutir à des ensemble de sonorités variées.

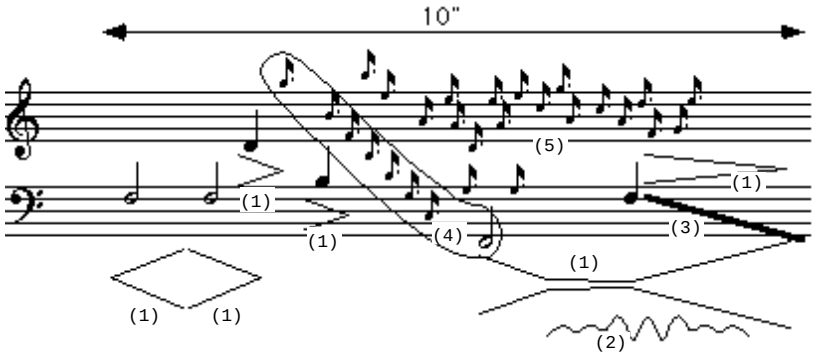


figure III-13: partition destinée à la production d'un son de synthèse
(1) : enveloppes d'amplitude ; (2) : forme du vibrato ; (3) : glissando ;
(4) : désynchronisation de partiels (arpège) ; (5) : partiels joués individuellement
(exemple sonore n°25).

Dans la figure III-13, les notes blanches ou noires indiquent des sons pour lesquels les partiels sont synchronisés et reproduisent un son de vibraphone, à l'exception de la première note pour laquelle les enveloppes sont retournées et donnent un son joué à l'envers. Des enveloppes d'amplitude ont été modifiées, raccourcies ou allongées. Dans le cas de la note Fa1, au lieu d'une enveloppe percussive, nous avons placé une enveloppe comportant une partie entretenue accompagnée d'un vibrato avec un crescendo vers la fin. La note noire Fa2 suit un glissando qui aboutit à la note Fa1. Les petites notes indiquent des partiels assimilables à des résonances qui sont extraites du fichier d'analyse et jouées séparément. Les notes groupées et entourées indiquent que les partiels d'un même son sont désynchronisés en arpège.[12]

[12] Ce types de variations à partir d'un modèle spectral est inspiré des variations réalisées par Jean-Claude Risset dans sa pièce « Passages » à partir de modèles de cloches (cf. § IV-1).

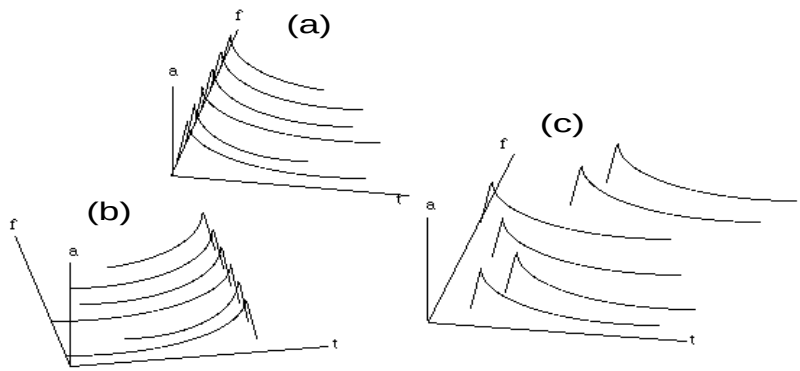


figure III–14 : enveloppes des partiels du modèle, (a) normale, (b) inversée, (c) asynchrone.

Pour répondre aux contraintes de la partition, nous avons du écrire quatre instruments pour Csound donnés dans l'orchestre « resons.orc » (cf. Annexe IV—B).

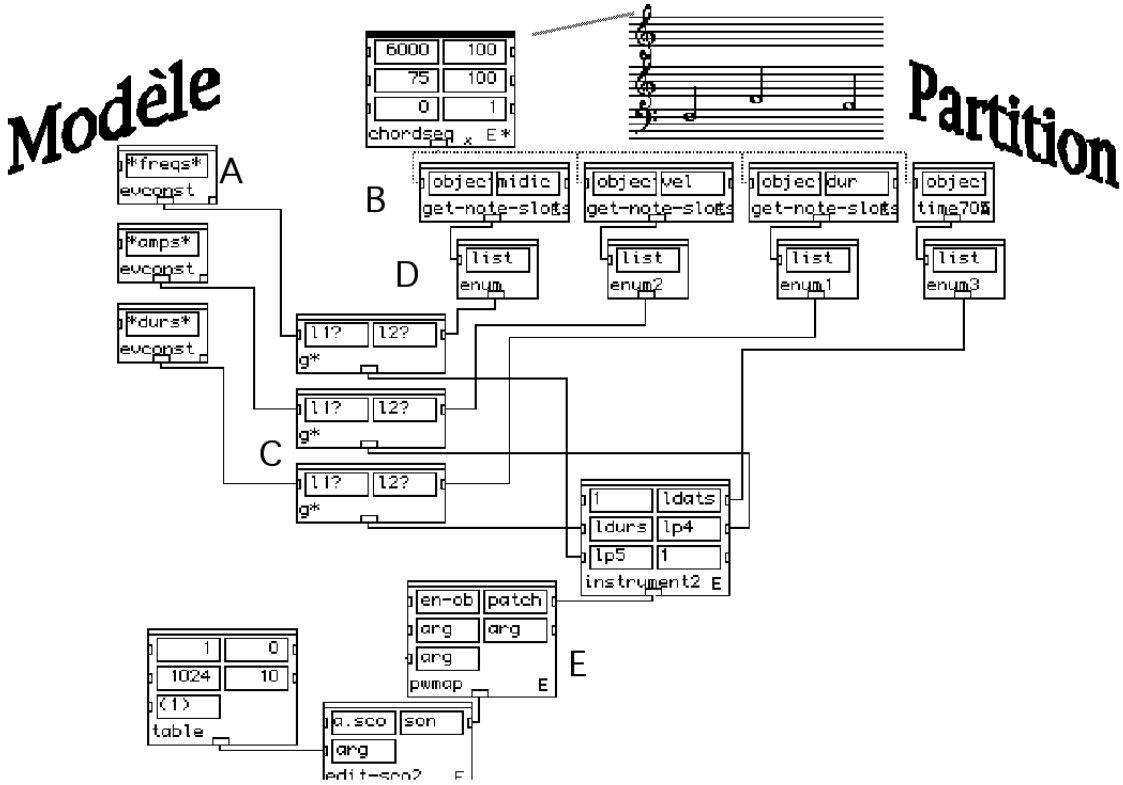


figure III–15 : patch pour la synthèse de sons d'après un modèle instrumental et une partition.

Le patch produisant les notes percussives jouées avec transposition et changement des durées est donné dans la figure III-15. Les données provenant du modèle sont délivrées par les modules *evconst* (A). Les valeurs sont modifiées (C) en fonction des paramètres donnés par la partition. Ces paramètres sont extraits par les modules *get-note-slots* et *time* (B) puis une boucle est mise en place pour les envoyer l'un après l'autre (modules *enum* (C) pour le début de la boucle, module *pwmap* (E) pour la fin de la boucle) vers les modules de multiplication *g** (C). Les données sont alors envoyées au module *instrument2* pour être formatées avant la production de la partition avec le module *edit-sco2*.

Des patchs similaires permettent de produire les partitions pour les autres instruments afin de réaliser la synthèse des différents sons constituant la partition de la figure III-13. Une partition unique regroupant toutes les données peut être créé à partir d'un patch général dans PatchWork. Lorsqu'on lance ensuite le

programme Csound en appelant cette partition, Csound produit un fichier—son unique contenant l'ensemble des sons de la partition de la figure III-13.

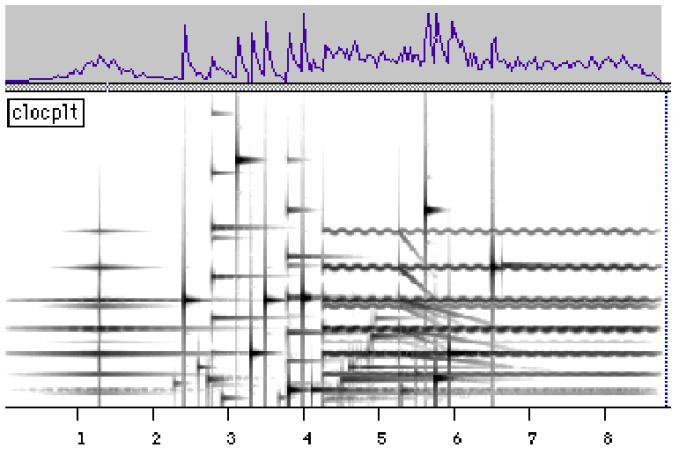


figure III-16 : sonagramme du son de synthèse construit sur un modèle de cloche (ex. son. n°25).

III.3.2. La construction de tables pour Csound

En utilisant la bibliothèque Csound/edit-sco, il est possible de contrôler les paramètres de la synthèse avec des graphes. Ces graphes peuvent être dessinés manuellement en utilisant la souris ou une tablette graphique. Ils peuvent également être obtenus par des moyens algorithmiques ou par la lecture de données provenant d'analyses de sons.

Dans PatchWork, les éditeurs de courbes permettent de placer des points sur un plan. Les informations contenues dans le graphe peuvent être obtenues soit sous forme d'une double liste contenant les coordonnées *XY* des points de la courbe, soit par un échantillonnage régulièrement espacé de la courbe, soit en fournissant une liste de valeurs d'ordonnées et en calculant les valeurs des abscisses correspondantes. La figure III-17 indique comment ces valeurs peuvent être obtenues à partir d'une courbe et à quel découpage de la courbe elles correspondent.

Selon la courbe, on choisira l'une ou l'autre des méthodes d'échantillonnage. Pour une courbe dessinée manuellement, on préfèrera l'utilisation des coordonnées des points. Pour une courbe complexe, l'échantillonnage à pas constant est mieux adapté. Pour une courbe qui présente des zones à variations rapides et d'autres à variations plus lentes, il est préférable de calculer les images par la courbe des valeurs choisies.

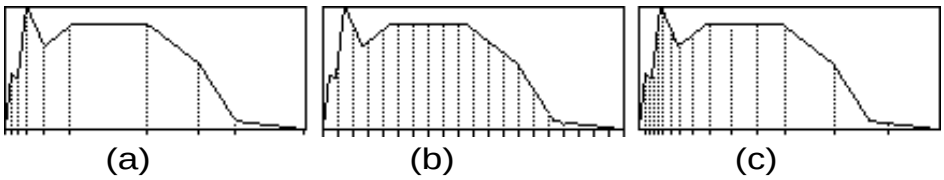


figure III-17 : échantillonnage de courbes par segments.
(a) coordonnées des points de la courbe, (b) échantillonnage à pas constant, (c) images de valeurs choisies

Lorsque l'on a obtenu les coordonnées d'un ensemble de points, un module de la bibliothèque Csound/edit-sco permet de construire automatiquement la table correspondante.

La construction graphique de tables est une technique souvent utilisée dans le domaine de la composition comme dans celui de la synthèse.

A) Enveloppes d'amplitude

Pour la synthèse des sons, le contrôle de l'enveloppe d'amplitude est un élément primordial. Sur la plupart des synthétiseurs du commerce, ce paramètre est limité à quelques segments : l'attaque, la tenue et la chute du son. Il est impossible de réaliser des microvariations de l'amplitude du son et de programmer des évolutions temporelles sur de longues durées.

Avec Csound, il est possible de déterminer des trajectoires complexes évoluant sur de longues durées. L'analyse de sons instrumentaux permet de constituer des banques de données contenant les enveloppes d'amplitude de différents instruments. Ces enveloppes peuvent ensuite être appliquées, avec ou sans transformation, à la synthèse de sons.

Pour les sons instrumentaux, on peut distinguer l'enveloppe d'amplitude globale du son, obtenue en effectuant la moyenne des amplitudes instantanées sur des fenêtres successives, des enveloppes d'amplitude calculées séparément pour chaque harmonique du son par analyse additive. L'enveloppe d'amplitude globale caractérise l'évolution dynamique du son alors que les enveloppes des harmoniques, dans leurs évolutions relatives, caractérisent le timbre. Ces enveloppes peuvent être utilisées pour des synthèses additives ainsi que pour des synthèses par forme d'onde ou par filtrage.

L'oreille perçoit peu distinctement les légères variations que l'on peut observer sur l'enveloppe d'amplitude globale. Seules les variations très contrastées de la dynamique du son sont décelables. Par contre, l'oreille est sensible aux variations relatives des partiels du son.

B) Trajets de fréquence

Sur beaucoup de synthétiseurs traditionnels, lorsqu'une note est jouée, la hauteur du son reste constante pendant la durée de la note. Seul le vibrato permet de produire des oscillations autour de cette hauteur. Cependant, avec les instruments acoustiques, la hauteur des sons instrumentaux n'est jamais parfaitement stable.

Lorsque l'on étudie le trajet de la fondamentale d'un son instrumental ou vocal, on observe des inflexions, des microvariations et du vibrato. Les variations diffèrent fortement d'un instrument à un autre (cf. figure III-19). Le vibrato, les glissandos et les enchaînements de notes sont des éléments importants pour la production de sons de synthèse intéressants.

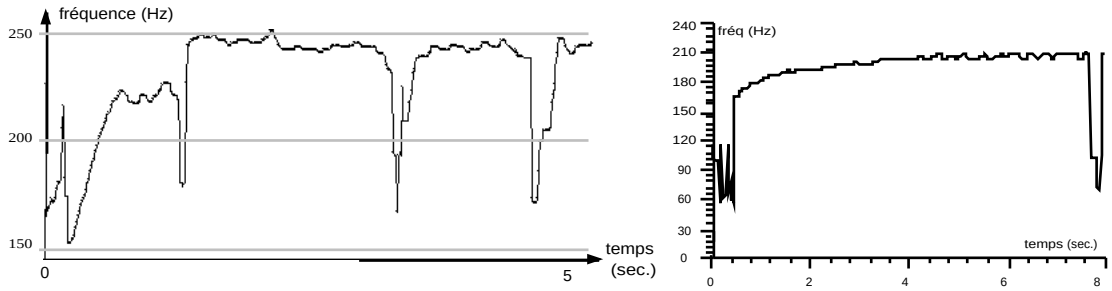


figure III-18 : courbes de l'évolution de la fondamentale d'une voix chantée asiatique (à gauche) et d'une voix chantée bouddhique (à droite) présentant des inflexions de fréquence (exemples sonores n°8).

Alors que le trajet de la fondamentale comporte des inflexions extrêmement significatives de la qualité du jeu instrumental, les trajets de fréquence des différents harmoniques apportent peu d'informations supplémentaires. Ces trajets ont en général une progression parallèle à celle de la fondamentale et les quelques irrégularités que l'on observe ne semblent jouer qu'un rôle mineur dans la qualité du son.

En utilisant PatchWork, Csound et une technique d'analyse additive, nous avons effectué des échanges de paramètres entre les fréquences et les amplitudes des harmoniques de plusieurs sons différents – sons de flûte, de guitare, de violoncelle et voix. Nous n'avons pas noté de différences importantes entre le son original et le même lorsque les trajets de fréquences avaient été échangés.

Par contre, pour étudier les différentes formes que peut prendre un vibrato de fréquence, nous avons comparé et échangé les trajets des fondamentales de plusieurs sons : un son de flûte, un son de violon, une voix féminine de Java et un son de shakuhachi. Nous avons obtenu des sons hybrides au rendu peu musical, possédant à la fois des caractéristiques des deux sons. Les formes de ces fondamentales sont données dans la figure III-19.

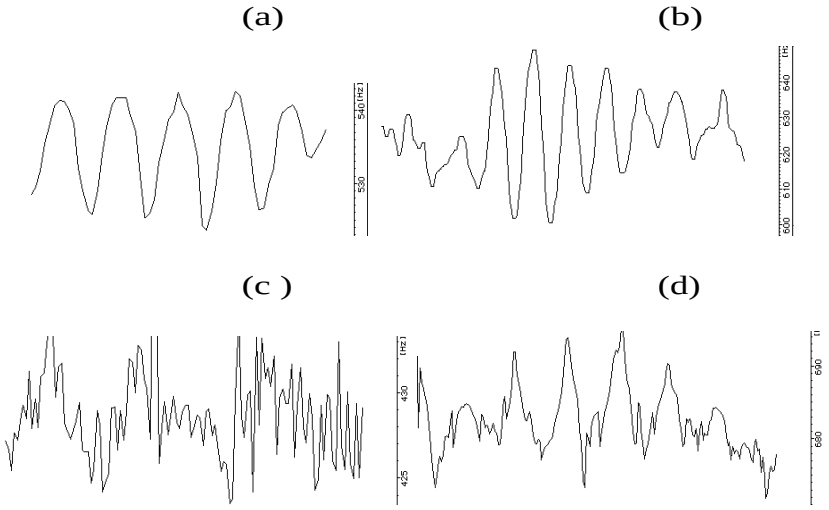


figure III-19 : trajets de fondamentales munies de vibratos pour différents sons : (a) violon (0,9"),(b) voix de java (1,6"), (c) shakuhachi (0,7"), (d) flûte (0,9")(exemples sonores n°26).

C) Fonctions graphiques diverses

Diverses fonctions mathématiques peuvent être utilisées pour décrire les tables qui serviront au contrôle des paramètres de la synthèse. Nous allons citer celles qui sont le plus fréquemment utilisées.

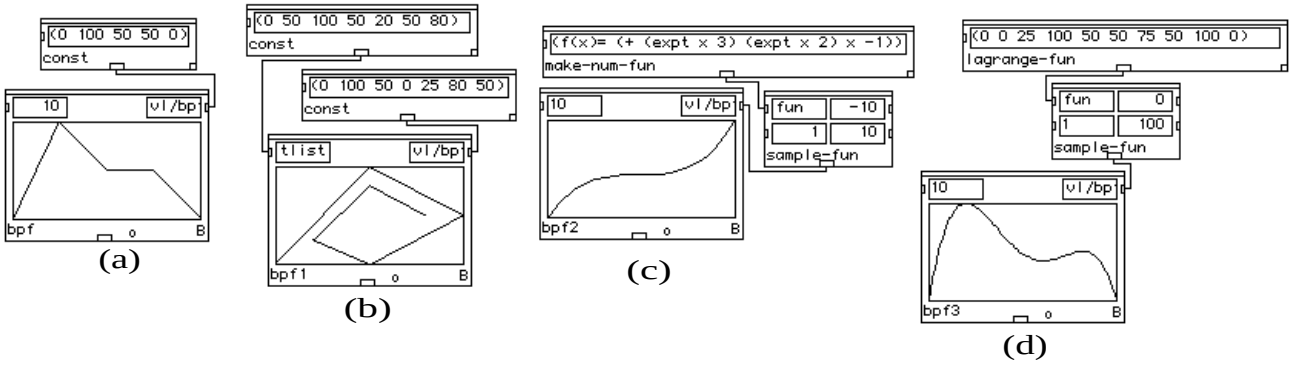


figure III-20 : création de courbes dans PW : (a) fonction linéaire par segments, seules les valeurs de Y sont fournies, (b) fonction linéaire par segments, les valeurs de X et de Y sont fournies, (c) fonction polynomiale échantillonnée pour X∈[-10, 10], (d) fonction polynomiale calculée par une fonction de Lagrange d'après la courbe (a).

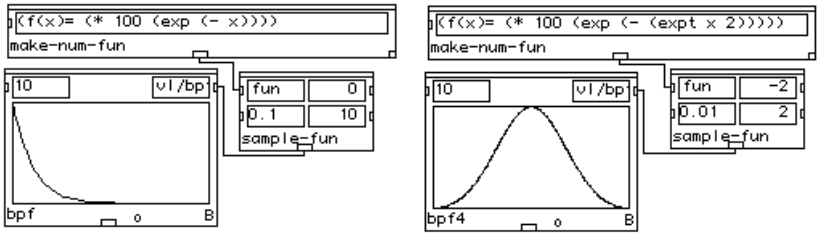
La fonction linéaire est une fonction qui permet de passer d'un point à un autre en ligne droite. Elle trouve ses limites dans le cas d'un parcours passant par plusieurs points non alignés. En effet, dans ce cas, le fait d'utiliser des fonctions linéaires par segments crée des ruptures de direction qui peuvent être gênantes à

l'audition. Par exemple, un glissando servant à l'enchaînement entre deux notes successives ne doit pas être produit de cette façon.

Les fonctions polynomiales pallient à cet inconvénient puisqu'elles permettent de décrire des trajets courbes ou « splines » passant par plusieurs points non alignés. Les fonctions de Lagrange permettent de calculer les coefficients des polynômes à partir des coordonnées des points.

La fonction exponentielle est souvent utilisée pour décrire les paramètres du son comme l'amplitude et la hauteur qui peuvent être représentées sur des échelles logarithmiques. Comme nous l'avons indiqué dans le chapitre I, l'amortissement d'un son varie selon une fonction exponentielle du temps.

Les courbes en cloche ou courbes de Gauss sont souvent utilisées pour fabriquer des enveloppes très douces. Ce type de courbes a été utilisé par Jean-Claude Risset pour les trajets d'amplitude des partiels constituant ses illusions auditives car elles permettent des attaques et des chutes très progressives.



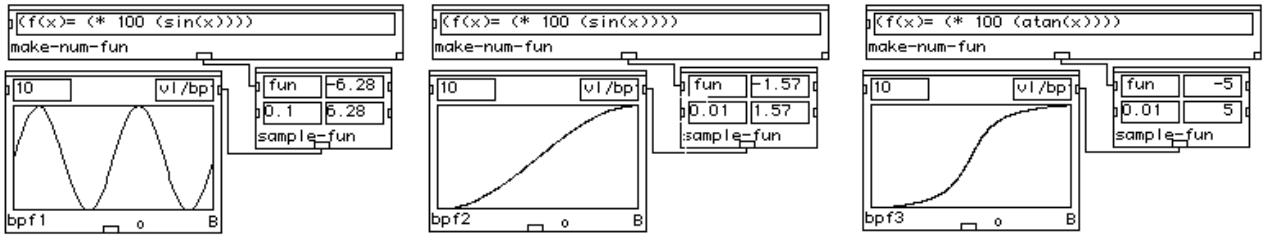
(a) (b)

figure III–21 : : création de courbes dans PW : (a) fonction exponentielle décroissante, (b) courbe de Gauss.

La fonction sinus correspond à la forme d'onde des sons purs. C'est également la principale fonction utilisée pour réaliser des modulations de type trémolo et vibrato.

On la retrouve également pour décrire des enveloppes d'amplitude ou de fréquence : **une moitié d'onde cosinus** permet d'avoir une attaque très douce et d'arriver rapidement à une valeur maximale. Dans le synthétiseur Chant, c'est la méthode employée pour les enveloppes d'amplitude ainsi que pour les trajets de fréquence lors d'une transition entre deux notes de hauteurs différentes.

La fonction Arc—tangente permet de décrire les variations de fréquences correspondant à un effet Doppler produit par une source sonore se déplaçant en ligne droite à vitesse constante.



(a) (b) (c)

figure III–22 : création de courbes dans PW.
(a) fonction sinus sur 2 périodes, (b) fonction sinus sur 1/2 de période, (c) fonction Arc—tangente.

Les fonctions browniennes permettent de réaliser des trajets aléatoires dont les écarts entre deux valeurs successives ne peuvent excéder un seuil donné. Elles sont utiles pour créer facilement des mouvements auquels on ne veut pas donner trop de régularité sans toutefois avoir à réaliser de formalisation excessive.

Les fonctions circulaires sont très employées pour la spatialisation du son. La diffusion de la musique électroacoustique en concert s'effectue souvent sur un grand nombre de haut-parleurs qui parfois entourent complètement le public. Les sons peuvent ainsi se déplacer autour des spectateurs selon des trajectoires qui sont définies en direct par le compositeur ou qui peuvent avoir été programmées à l'avance. Le programme **Holophon** du GMEM est un outil spécialement conçu pour la programmation graphique et algorithmique de la spatialisation de multiples sources sonores [Pottier—2000].

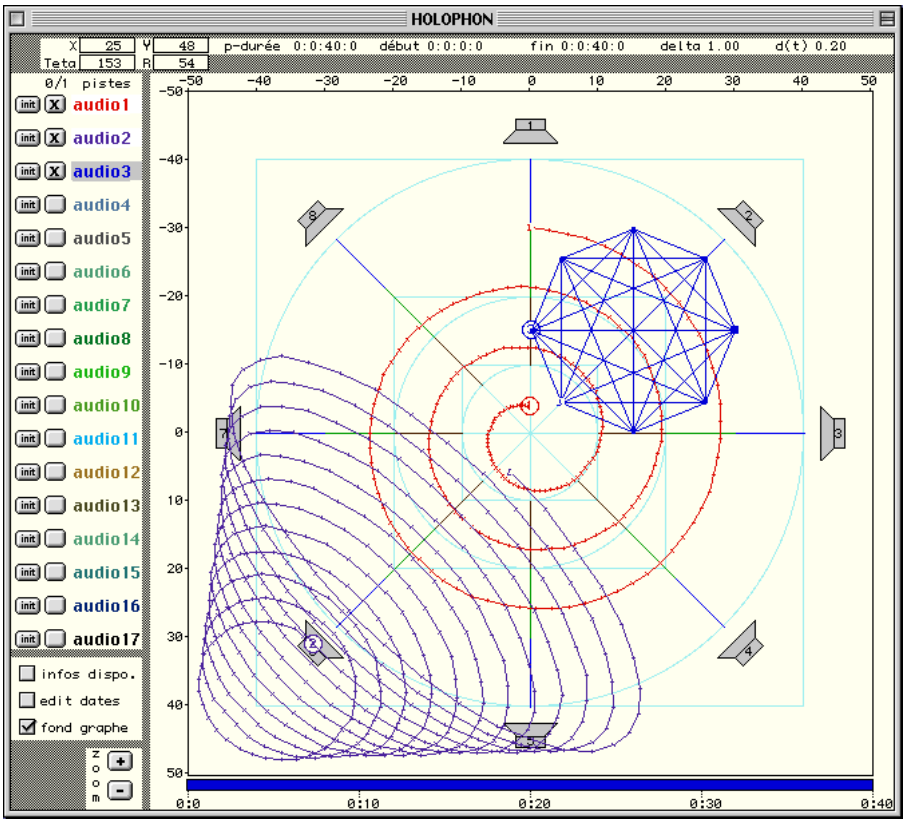


figure III-23 : éditeur graphique de trajectoires de spatialisation (Holophon—GMEM).

III.4. Manipulation de données d'analyses spectrales avec Patchwork

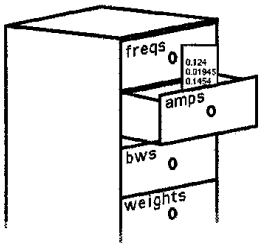
L'analyse des sons instrumentaux fournit d'importantes quantités de données. Des outils spécialisés sont nécessaires pour accéder à l'information utile contenue dans ces données. La bibliothèque « SpData » de PatchWork a été conçue dans ce but. Ecrite par Xavier Chabot en 1996 puis complétée en 1997 [Pottier—1997a], elle permet la lecture, l'écriture, la création, l'édition, la transformation et l'étude d'analyses spectrales du son.

III.4.1. L'organisation de la bibliothèque « Spdata »

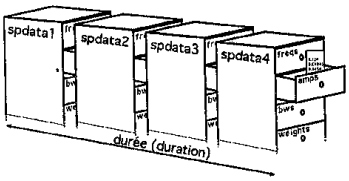
En raison de la taille considérable des fichiers contenant des données spectrales, en particulier lors d'analyses dynamiques, le traitement des données sous forme de listes dans PatchWork s'avère impossible. C'est pourquoi la technique de « programmation par objet » s'avère indispensable, les données n'étant plus lues et recopiées à chaque opération. Dans la bibliothèque SpData, les données fournies par l'analyse spectrale des sons sont stockées dans des emplacements organisés selon une architecture d'objets. On distingue deux catégories : les analyses statiques et les analyses dynamiques.

Dans le cas d'**analyses statiques** les données ne contiennent pas d'informations temporelles. Il s'agit de listes de valeurs décrivant des paramètres dont le nombre et la nature varient selon l'analyse choisie. Une première classe d'objets intitulée *C-spdata* permet de représenter les analyses statiques : les valeurs des fréquences sont stockés dans un emplacement nommé *freqs*, celles des amplitudes dans l'emplacement *amps*, etc.. En tout, onze emplacements ont été prévus pour stocker les données en provenance de divers types d'analyses.

size : nombre de partiels
frame : temps de l'analyse dans le son
partials : numérotation des partiels
freqs : fréquences
amps : amplitudes
normalized-amps : amplitudes réduite (la somme vaut 1)
bws : largeurs de bande
phases : phases
weights : poids perceptifs
resfact : durée de résonance globale
patch : nom d'un patch Max associé



Pour les **analyses dynamiques**, une classe intitulée *C-spdata-seq* sert à stocker les données d'analyses dynamiques. Une analyse dynamique correspond à une succession d'analyses statiques effectuées en divers endroits du son. Un objet *C-spdata-seq* contient un assemblage d'objets *C-spdata* correspondant aux analyses successives ainsi qu'un paramètre indiquant la durée du son analysé.



Cette technique de programmation par objets permet de manipuler plus de fonctions et moins de valeurs numériques, réduisant ainsi les temps de calcul de façon considérable.

III.4.2. Les fonctionnalités de la bibliothèque « SpData »

III.4.2.1. Lecture, visualisation et écriture des données spectrales

La bibliothèque fournit de nombreuses fonctions. Elle permet la lecture et l'écriture automatiques de données aux formats variés, correspondant à divers types d'analyses. Elle dispose également de modules permettant la visualisation et la modification des données obtenues par lecture de fichiers d'analyse.

En combinant cette bibliothèque avec la bibliothèque Csound/edit-sco, on peut produire des partitions pour des resynthèses d'après l'analyse.

Il est aussi possible de traduire les données d'analyse en messages MIDI et, de cette façon, d'avoir une écoute immédiate à partir de PatchWork en les faisant jouer en temps réel sur un synthétiseur.

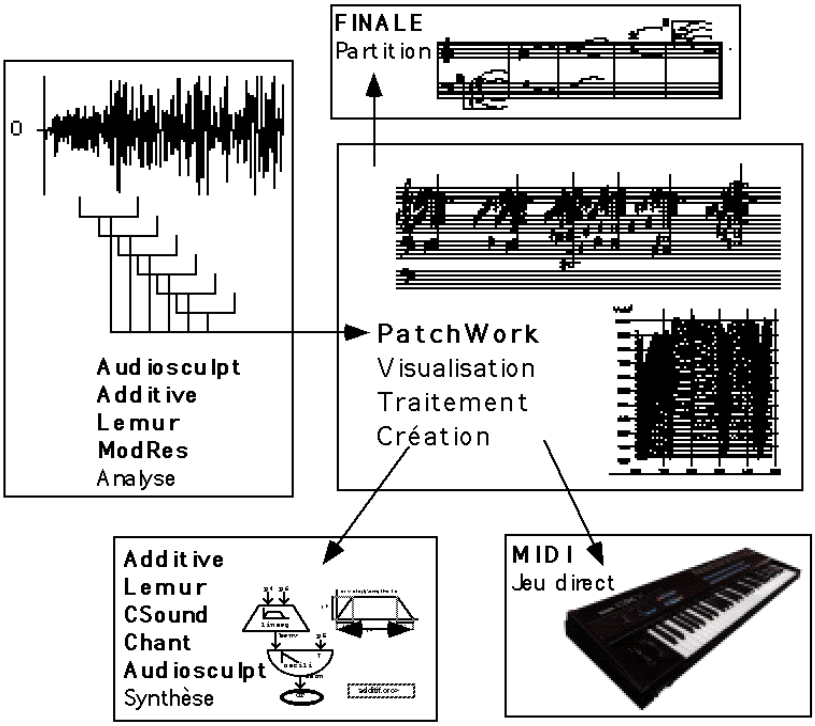


figure III-24 : entrées et sorties de données spectrales avec la bibliothèque SpData de PatchWork.

III.4.2.2. Traitement des données spectrales

Plusieurs modules permettent d'effectuer des traitements sur les données d'analyse. Parmi ceux-ci, le module *filter-spdata* réalise des **filtrages** en supprimant des partiels en fonction de critères portant sur un paramètre choisi.

Certaines modifications classiques peuvent être réalisées comme, par exemple, la transposition ou la modification de l'échelle du temps. Le module *scale—spdata* permet d'effectuer des **changements d'échelle**s sur tous les paramètres des objets *C-spdata* ou *C-spdata-seq*.

Des modifications plus complexes permettent également d'affecter le timbre. On peut ainsi augmenter ou diminuer la **brillance du son** en déplaçant le barycentre des fréquences partielles avec le module *l-stat-modif* qui sert à modifier de façon statistique les zones d'énergie dans le spectre. Il laisse les fréquences inchangées mais agit sur les amplitudes des partiels.

Le barycentre est égal à :

$$B = \frac{\sum freq_i \cdot amp_i}{\sum amp_i}$$ pour des données additives

$$B = \frac{\sum \frac{freq_i \cdot amp_i}{bw_i}}{\sum \frac{amp_i}{bw_i}}$$ pour des données de modèles de résonance

La figure III-25 montre comment la fonction *l-stat-modif* modifie un spectre en déplaçant le barycentre – la moyenne - de ses fréquences ou en changeant leur dispersion – l'écart-type.

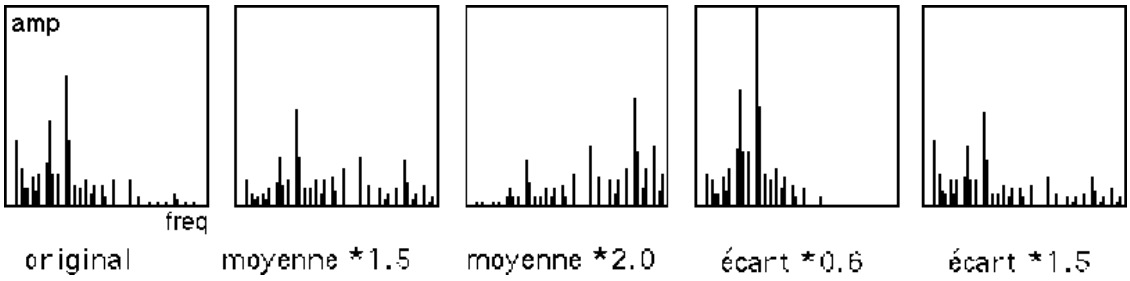


figure III-25 : modifications statistiques des amplitudes des pics d'un spectre.

Un algorithme a également été mis au point pour **amplifier ou réduire les composants harmoniques** d'un son par rapport aux composants non harmoniques.

Pour les analyses par modèle de résonance, un autre algorithme permet d'**accentuer l'attaque ou les résonances** d'un son.

Toutes ces méthodes de traitement sont spécifiques de l'environnement PatchWork-SpData car il n'existe pas d'autre outil permettant à la fois un travail sur la microstructure du son et sur des paramètres reliés de façon très sensible à la perception.

A partir du moment où l'on dispose d'une technique d'analyse-synthèse qui permet de reproduire fidèlement le son original, la bibliothèque SpData permet d'obtenir des sons transformés qui gardent la cohérence d'un modèle original tout en présentant une grande variété de sonorités. Ceci en fait un outil unique.

III.4.2.3. Synthèse sonore à partir de données spectrales structurées

Les données contenues dans les analyses peuvent être utilisées pour décrire des paramètres utiles à la synthèse, par exemple, piloter des oscillateurs en synthèse additive ou des filtres en synthèse soustractive. Des exemples en seront donnés au chapitre IV.

Pour réaliser la synthèse à partir de données provenant d'analyses dynamiques, nous indiquerons deux exemples de démarches possibles.

La première façon de procéder consiste à utiliser des oscillateurs dont on contrôle au cours du temps les fréquences et les amplitudes, que ce soit par des courbes, par le calcul ou par un geste instrumental.

La deuxième démarche correspond à une synthèse granulaire, on utilise des grains sonores qui se recouvrent de façon à produire un son continu. Chaque grain correspond à un partiel prélevé sur une analyse statique.

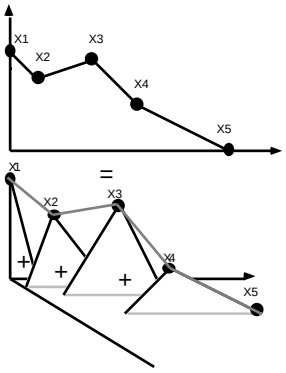


figure III-26 : décomposition en grains d'une courbe par segments.

Cette méthode nous a permis de reproduire fidèlement des sons analysés par la technique additive et également de produire des déformations très intéressantes sur les sons.

Nous avons utilisé cette méthode pour la synthèse de sons analysés avec l'algorithme de Terhardt. Dans ce cas, les partiels obtenus dans une fenêtre d'analyse ne sont plus automatiquement corrélés avec ceux de la fenêtre précédente. Des pas d'avancement d'une taille supérieure à 1/10e de seconde doivent être utilisés pour éviter des modulations d'amplitude dues à des inversions de phase. Cette technique n'est pas destinée à la réalisation d'une synthèse reproduisant un son à l'identique. Les deux exemples suivants montrent des champs d'application de cette technique.

Gérard Assayag [Assayag, Castellengo, Malherbe-1985] a utilisé les algorithmes de Terhardt pour l'étude des multiphoniques produits par les instruments à vent. S'inspirant de ce travail, Eric Mauer a entrepris en 1994 une classification systématique et comparée des sons multiphoniques des instruments à vent en utilisant la bibliothèque SpData pour déterminer les hauteurs perçues. La synthèse avec Csound lui a permis de comparer les hauteurs perçues dans les sons de synthèse et dans les sons originaux pour valider les résultats obtenus.

Joshua Fineberg a utilisé la bibliothèque SpData pour réaliser la partie électronique de sa pièce « Paradigmes » (1994) sur la Station d'Informatique Musicale de l'Ircam. Par l'algorithme de Terhardt, il a analysé des sons d'orchestre puis a interpolé les résultats avec divers accords préétablis. Il a utilisé Csound et la technique de synthèse par grains, sur Macintosh pour tester les paramètres de la synthèse. Ensuite, sur la station (SIM), il a pu contrôler en temps réel la vitesse d'exécution de la synthèse.

III.4.2.4. Interpolation de modèles instrumentaux

Nous avons programmé des modules et réalisé un patch pour réaliser des transitions entre deux modèles par l'interpolation des paramètres issus d'analyses additives (cf. Annexe V-G).

Les fréquences et les amplitudes des partiels de deux sons peuvent être modifiées et interpolées de façon indépendantes. C'est en utilisant ce patch que nous avons remarqué que les interpolations des amplitudes des partiels étaient beaucoup plus significatives que les interpolations des fréquences.

Lorsque les deux sons ne sont pas de même hauteur, l'interpolation des fréquences produit des glissandos désastreux, on peut remédier à cela en n'interpolant que les amplitudes.

Des techniques d'interpolation à partir de données additives ou de modèles de résonances sont maintenant également disponibles dans le programme Diphone [Dudas-1997] et [Pottier-1997b].

III.5. La bibliothèque PW-Chant

Le synthétiseur Chant a fait l'objet de nombreux travaux de développement qui ont permis son intégration dans le programme PatchWork en tant que bibliothèque.

Lorsque cette bibliothèque est chargée dans PatchWork, on dispose de modules qui permettent la construction d'un patch de synthèse de sons numériques. La création d'un tel patch s'effectue en deux temps : la création de l'instrument et l'écriture de listes de notes destinées à faire jouer cet instrument.

La synthèse est déclenchée directement à partir du programme PatchWork et aboutit à la production d'un fichier—son sur le disque de l'ordinateur.

III.5.1. La notion « d'instrument » dans PatchWork

Le programme PatchWork est un outil d'aide à la composition qui permet avant tout d'effectuer des manipulations et des calculs complexes sur des notes, des rythmes et des structures formelles. Grâce à la bibliothèque Chant, il permet également d'aboutir à la synthèse sonore en dernière phase de la formalisation.

Cette formalisation fournit un ensemble de notes – définies par leur hauteur, leur date, leur durée et leur amplitude – à un module *chordseq*.

Pour réaliser la synthèse sonore, il est nécessaire d'attribuer un « instrument » à chaque note sous la forme d'un patch, ou abstraction, plus ou moins complexe indiquant la nature et le paramétrage de la synthèse utilisée.

III.5.2. Les différents « synthétiseurs » disponibles

On retrouve avec PW-Chant les trois types de synthèse disponibles dans Chant.

La synthèse de la voix est réalisée par le module de synthèse *fof-synth* qui effectue une synthèse par Fof entretenues.

La synthèse par filtrage est obtenue avec le module de synthèse *filter-synth* qui permet le filtrage de bruit blanc ou de fichiers-son enregistrés sur disque.

La synthèse de sons percussifs par Fof est effectuée par le module de synthèse *reson-synth*.

Pour tous les modules de synthèse, l'entrée numéro 1 sert à indiquer le modèle de l'instrument ou de la voix grâce à un menu rotatif qui indique la liste des voyelles pour les différents types de voix (basse, ténor, soprano, ...) ainsi que les modèles instrumentaux (piano, marimba, ...) disponibles.

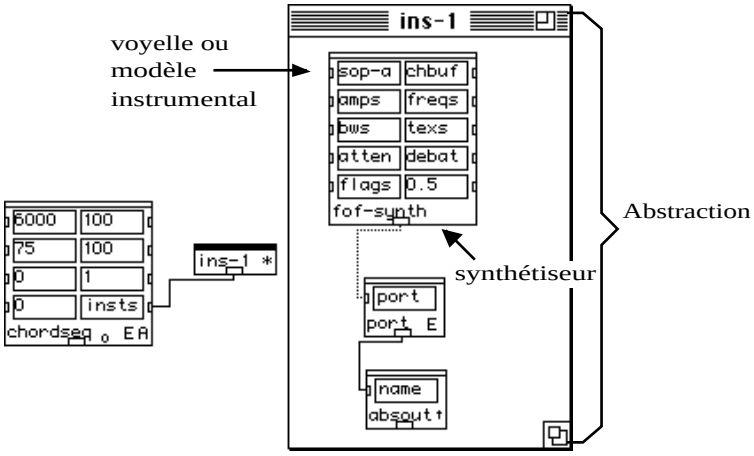


figure III-27 : instrument pour la synthèse de la voix

Pour les filtrages, la dernière entrée du synthétiseur *filter-synth* sert à connecter la source à filtrer. Celle-ci est donnée par le module *souffle-synth* pour le bruit blanc et par le module *source-synth* pour un fichier-son.

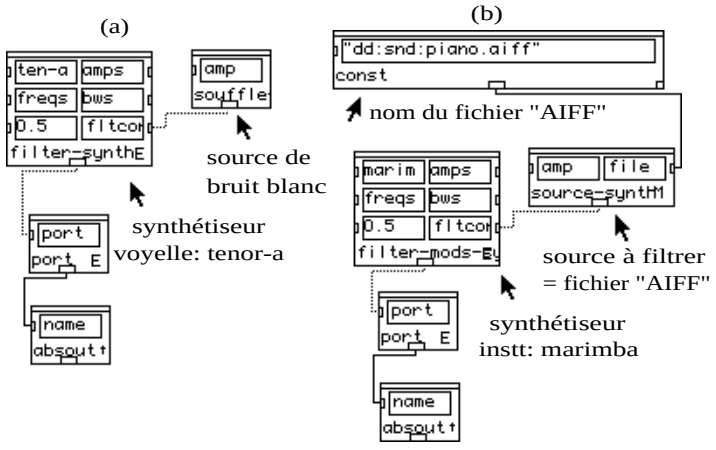


figure III-28 : instrument pour le filtrage: (a) bruit blanc filtré par une voyelle (b) son de piano filtré par un marimba

Les autres entrées du synthétiseur servent au contrôle des paramètres de la synthèse.

III.5.3. Le contrôle des paramètres de la synthèse

Il est possible d'enrichir les instruments de façon à permettre des contrôles variés sur la synthèse.

Divers patchs peuvent être branchés sur les différentes entrées des synthétiseurs pour modifier les valeurs de leurs paramètres. Ces paramètres correspondent à des données de la partition – hauteur, date, durée, amplitude, note précédente – ou relatives au modèle instrumental utilisé – fréquences, amplitudes, largeurs de bande des formants, *tex*, *atten*, *debatt* –. Plusieurs règles peuvent être appliquées à ces paramètres pour en modifier les valeurs au cours du temps.

Un protocole détaillé de programmation permet à l'utilisateur d'écrire en Lisp ses propres règles [Iovino, Laurson, Pottier-1994] qui vont se rajouter aux modules existant dont nous citerons les principaux.

Le vibrato et le « jitter »

L'adjonction d'un vibrato ou d'un « jitter » lors de la synthèse de la voix chantée est indispensable pour réaliser une synthèse réaliste.

On dispose d'un vibrato et d'un jitter simples ainsi que d'un vibrato vocal, comportant des microvariations. Les douze paramètres de contrôle du vibrato vocal sont réglés par défaut sur les valeurs correspondant à celles d'une voix chantée.

Les enveloppes

Il est possible d'utiliser des graphes pour représenter les variations dynamiques de chaque paramètre. Ces graphes sont des fonctions linéaires par segments (*BPF*) dont les valeurs en *X* et *Y* peuvent provenir d'un formalisme produit par des patchs, ou être dessinées à l'aide de la souris. Ils peuvent être placés directement sur l'entrée d'un module de contrôle pour en définir les variations.

Les transitions entre notes

Pour la synthèse de la voix, il est possible d'effectuer des transitions entre deux notes consécutives à l'aide du module *interpol-freq*. Il permet une transition entre deux voyelles par l'interpolation progressive des fréquences des formants.

Les règles automatiques

Les variations de hauteur et d'intensité des notes chantées s'accompagnent généralement de variations des différents paramètres des formants. Ces variations peuvent être contrôlées automatiquement par plusieurs modules : *auto-bend*, *auto-ampl*, *auto-bw*, *ampl-corr*.

Les interpolations

Pour les modèles instrumentaux, des fonctions d'interpolation permettent de créer des sons intermédiaires entre plusieurs instruments. Des trajets peuvent s'effectuer par l'interpolation des fréquences, des amplitudes et des résonances.

Une caractéristique de cette technique d'interpolation est de procéder par appariement : l'interpolation des fréquences n'est effectuée que pour les formants dont les fréquences sont voisines, de façon à éviter des glissandos indésirables.

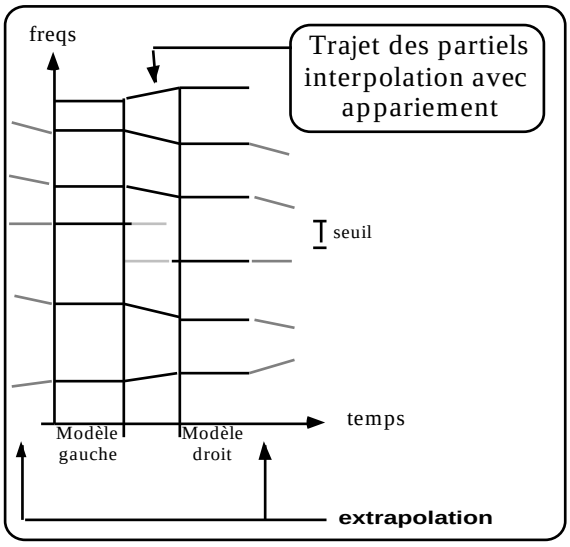


figure III-29 : interpolation des fréquences des partiels appariés

L'interpolation des largeurs de bande permet d'interpoler les durées des sons. On dispose d'une interpolation à deux dimensions, spectrale et temporelle, ce qui la rend très performante et la distingue d'un simple mixage entre instruments.

La constitution de bases de données

La base de données contenant les modèles de voix chantée et les modèles instrumentaux peut facilement être étendue par les utilisateurs. Pour cela, ils doivent disposer de données d'analyses obtenues par LPC pour la voix (cf. chapitre I) ou par modèle de résonance pour les instruments. Des modèles abstraits issus d'une formalisation quelconque peuvent également être utilisés. La bibliothèque SpData de PatchWork est adaptée à ce genre d'opération.

III.6. Conclusion sur PatchWork

Le programme PatchWork connaît un engouement manifeste de la part des compositeurs qui ont eu l'occasion de le mettre en pratique. On peut donc penser que ce programme constitue une réponse à leurs demandes et facilite leur activité compositionnelle. De plus, chaque compositeur arrive à trouver dans ce programme une utilisation qui lui est très personnelle.

On peut toutefois opposer deux démarches dans l'utilisation de PatchWork. Il est possible d'utiliser PatchWork uniquement pour effectuer des calculs qu'il aurait été long et laborieux de réaliser à la main, mais on peut aussi l'utiliser pour explorer des domaines dont la complexité ne pouvait être abordée sans le recours à l'ordinateur.

Carlos Agon souligne très justement dans sa thèse [Agon-1998, p. 133] que « la CAO a relativisé la fausse complexité de techniques qui étaient simplement compliquées ». Il désigne ainsi ceux qui utilisent PatchWork comme une calculatrice sophistiquée qui leur permet de réaliser des calculs compliqués, en particulier ceux qui suivent une démarche sérielle, s'appuyant sur la combinatoire.

D'un autre côté, il indique que « l'ordinateur permet d'explorer des champs expérimentaux d'une complexité réelle et qui étaient inaccessibles auparavant comme, par exemple, le champ du timbre dans sa représentation spectrale ». Les exemples que nous développons dans le chapitre suivant sont situés essentiellement dans ce champ.

Ce qui a fait le succès du programme PatchWork, c'est la facilité avec laquelle on peut l'aborder. Même si la réalisation d'un patch sophistiqué réclame une certaine compétence, PatchWork permet rapidement aux compositeurs de travailler et de construire leurs premiers patchs[13].

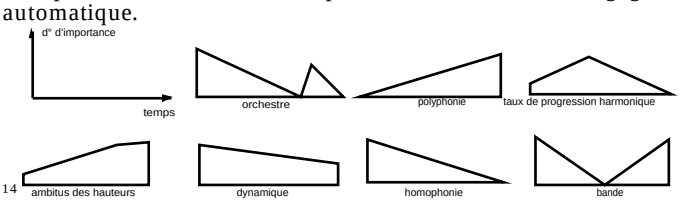
Les éditeurs graphiques et la notation musicale sont pour beaucoup dans la convivialité de ce programme. Les compositeurs ont des habitudes de pensée orientées vers le graphisme. Beaucoup d'entre eux utilisent un support graphique pour la préparation de leurs œuvres (Kaija Saariaho décrit par des courbes l'évolution de la plupart des paramètres liés à l'écriture de ses pièce[14], Iannis Xénakis. utilise des graphes pour programmer tous les paramètres du son - les formes d'onde, les enveloppes d'amplitude, les trajets de fréquences – avec le systèlme Upic, Klaus Huber réalise des courbes enchevêtrées dessinées à la main dont les intersections vont lui donner les rythmes, Jean-Claude Risset réalise souvent des esquisses de proportions pour définir la forme de sa pièce, Fausto Romitelli utilise des dessins pour définir les enveloppes des sons ou une allure globale d'évolution de certains paramètres au cours du temps etc…).

Patchwork dispose d'un ensemble de bibliothèques qui augmente au fur et à mesure de ses utilisations par divers compositeurs. Elles sont le fruit d'une expérience de composition assistée par ordinateur qui se renouvelle depuis près de vingt ans. On trouve parmi celles-ci les bibliothèques « Esquisse » pour l'harmonie et le traitement spectral (Magnus Lindberg, Marc-André Dalbavie, Jean-Baptiste Barrière, Kaija Saariaho) ; « Alea » pour la composition stochastique (Mikhail Malt) ; « Chaos » pour les algorithmes de la théorie du chaos (Mikhail Malt) ; « Gen-Utils » pour la construction et la transformation de courbes (Hans-Peter Stubbe) ; Situations (Antoine Bonnet et Camilo Rueda) et Contraints (Mikael Laurson) pour la programmation par contrainte, permettant par exemple de réaliser du contrepoint d'école ; « Profile » pour l'interpolation de fonctions ou de courbes (Jacopo Baboni).

Un des points forts de PatchWork tient aux possibilités données aux utilisateurs pour construire leurs propres bibliothèques de modules, écrites soit à partir de patchs compilés, soit à l'aide de fonctions écrites en Lisp. PatchWork dispose du potentiel d'un vrai langage de programmation en particulier avec les systèmes d'abstraction ou « encapsulation ». La programmation en Lisp est disponible à l'intérieur de PatchWork. Son apprentissage peut être très progressif et permet d'obtenir rapidement des résultats, par rapport à d'autres langages.

Les nombreuses bibliothèques mentionnées précédemment peuvent être utilisées pour la composition instrumentale ainsi que pour la synthèse. Couplées avec les bibliothèques « Chant », « Csound/Edit-sco » et « SpData », elles permettent de relier l'écriture instrumentale à la synthèse en prolongeant cette écriture jusque dans la microstructure du son.

[13] Avec PatchWork, on est bien sûr loin des programmes « Plug & Play » pour lesquels il suffit de lancer l'application puis de cliquer sur un bouton pour entendre immédiatement un résultat musical mais les possibilités ne sont pas du tout les mêmes non plus. PatchWork est un langage de programmation et non un logiciel de composition automatique.



[14] Courbes représentant des paramètres compositionnels de « Verblendungen » (1991) [Saariaho-1991, p. 418].

Le langage PatchWork comporte toutefois certaines insuffisances: l'impossibilité de programmer graphiquement des objets et des méthodes et la difficulté d'organiser des patchs dans le temps. C'est sur ce constat que le développement du programme PatchWork a été arrêté à l'Ircam pour laisser la place à un nouvel environnement de programmation graphique, **OpenMusic**, qui tout en conservant une très grande analogie avec PatchWork devrait permettre de compenser ses insuffisances [Assayag, Agon-1997] et [Agon-1998]. OpenMusic est distribué depuis 1998 et il semble que les versions actuelles aient acquis une fiabilité et une efficacité lui permettant de prendre la place de PatchWork. Notons que si les bibliothèques Chant et SpData n'ont pas encore été portées sur OpenMusic, on trouve depuis peu la bibliothèque « OM2Csound » (Mikhaïl Malt, Laurent Pottier, Karim Haddad) pour le formatage des données spécifiques aux partitions. Il a été annoncé une nouvelle bibliothèque, « Chroma », de Marco Stroppa qui est également destinée à la synthèse.

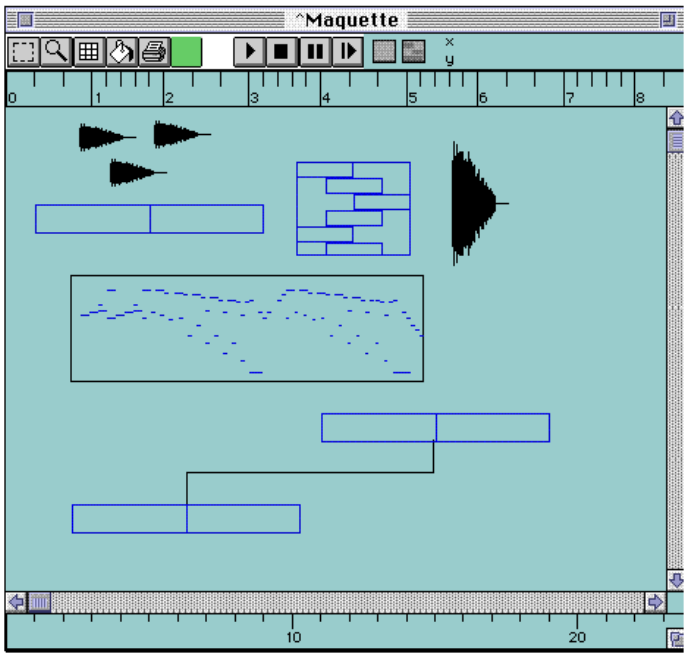


figure III-30 : maquette pour la représentation temporelle de l'organisation des patchs dans « Open-Music » [Assayag, Agon-1997].