UNIVERSITE PARIS VI – PIERRE ET MARIE CURIE
Ecole Doctorale d'Informatique, Télécommunications et Electronique de Paris

Thèse de Doctorat en Informatique

Jean BRESSON

La synthèse sonore en composition musicale assistée par ordinateur

Modélisation et écriture du son

Thèse dirigée par Carlos AGON
Soutenue le 30 Novembre 2007

Jury :
- M. Philippe DEPALLE (Rapporteur)
- M. Miller PUCKETTE (Rapporteur)
- M. Claude CADOZ (Président)
- M. Carlos AGON (Directeur de thèse)
- M. Gérard ASSAYAG (Examinateur)
- M. Michel BEAUDOUIN-LAFON (Examinateur)
- M. Jean-Dominique POLACK (Examinateur)

Thèse préparée à l'IRCAM – CNRS UMR 9912
Equipe Représentations Musicales
1, place Igor Stravinsky 75004 Paris
http ://www.ircam.fr

Version révisée le 22/01/2008

Résumé

Les progrès réalisés dans le domaine de la synthèse sonore ont introduit des ouvertures significatives liées à la production des sons par le calcul, et permettent aujourd'hui d'envisager une réelle écriture, ou composition du son. Cependant, rare sont les outils qui permettent de détacher réellement le son de son caractère physique (en tant que signal) pour en proposer une représentation plus "musicale".

La représentation musicale du son est une question complexe mettant en jeu des aspects et problématiques pluridisciplinaires, et impliquant une réflexion fondamentale sur la pratique compositionnelle et ses rapports aux nouvelles technologies. La dualité entre le niveau symbolique de la composition musicale et celui du signal sonore se présente alors comme une question fondamentale, à laquelle s'ajoutent d'autres problèmes récurrents en informatique musicale, tels que la distinction entre discret et continu dans le domaine temporel.

Après avoir déterminé les termes et les enjeux du sujet, nous proposons une approche inspirée des principes de la composition assistée par ordinateur (CAO). La CAO s'intéresse aux formalismes musicaux et calculatoires mis en relation dans des environnements informatiques de composition. Elle met en avant une démarche interactive du compositeur vis-à-vis des processus de composition réalisés par ordinateur, que nous mettons en parallèle avec l'idée de modélisation. Dès lors, les langages de programmation se présentent comme des outils privilégiés pour le développement des modèles et processus compositionnels.

En reportant ces idées au domaine de la synthèse sonore, nous envisageons une nouvelle conception du son en tant qu'objet d'un modèle, représenté sous forme de programmes unifiant données et processus musicaux. Cette conception permet alors de reconsidérer le phénomène sonore sous un point de vue compositionnel.

Des propositions sont apportée avec un système intégré à l'environnement de CAO OpenMusic, permettant aux compositeurs d'y développer des processus et modèles compositionnels orientés vers le son et la synthèse. Les outils constituant cet environnement permettent d'une part de traiter et manipuler données et processus liés au domaine du signal, principalement à l'aide de la programmation visuelle, et d'autre part de contrôler des niveaux plus abstraits, impliquant notamment des structures et représentations symboliques, et permettant d'organiser des formes musicales dans des structures temporelles élaborées.

Abstract

Although many works have been carried out in the field of digital sound synthesis and processing techniques, which introduced significant openings related to the sound production by computer, few of them actually allow to dissociate the sound from its physical/signal character in order to propose a more "musical" representation.

The musical representation of sound is a complex problem, involving interdisciplinary aspects, and requires a fundamental reflection about music composition and its relations to new technologies. The duality between the symbolic level of music composition and the level of sound signals is for example an important question which underlies the principal problematic dealt with in this thesis. We will see that it is possible to put this matter in relation with other recurring problems in the computer music field, such as the distinction between discrete and continuous time representations.

After determining the key issues of the problem, we follow an approach inspired by the principles of computer-aided composition (CAC). CAC brings together musical and computing formalisms in computer composition environments, and puts forward an interactive approach from the composer regarding the computer composition processes, which we relate to the notion of compositional modelling. Programming languages thus represent privileged means for the development of compositional models and processes within the framework of CAC.

By translating these ideas in the field of sound synthesis, we propose a new conception of sound considered as the object of a model, and represented as a program uniting description data and musical processes. This approach, which constitutes one of the main contribution of this thesis, will then allow one to consider sound from a compositional standpoint.

A new framework that we developed is presented, which allows to deal with these problems in a practical implementation. This system is integrated in the CAC environment OpenMusic, and allows composers to extend the compositional domain of this environment to the field of sound signals, hence leading to the development of musical models and processes related to sounds and sound synthesis. The tools developed thus concern the low level of sound description structures and processing using visual programming techniques, and also integrate more abstract levels, including symbolic structures and representations, and elaborated time structures.

Merci...

... à Carlos Agon, sans qui cette thèse n'aurait certainement pas existé, et qui représente bien plus que le simple directeur de ce travail ;

... à Gérard Assayag pour son accueil au sein de l'équipe Représentations Musicales, ainsi qu'à Hugues Vinet pour la confiance qu'il m'a accordée en me permettant de réaliser cette thèse à l'Ircam ;

... aux membres du jury qui ont aimablement accepté d'évaluer mon travail ;

... à Karim Haddad, Marco Stroppa, qui ont été d'une aide précieuse, mais aussi à Claude Cadoz, Hugues Dufourt, Philippe Manoury, à Tolga Tüzün, Jean Lochard, Hans Tutschku et aux autres collaborateurs qui ont à un moment participé à ce travail ;

... à tous ceux qui m'ont aidé par leurs lectures et conseils avisés, Moreno Andreatta, Olivier Warusfel, Isabelle Viaud-Delmon, Mikhaïl Malt, Xavier Rodet, Maman et Gyl, Manouel Van Slaghmolen, Daniel Pressnitzer ;

... à ceux qui étaient là aussi pour le soutien, dont beaucoup ont été cités plus haut, mais aussi Coralie, Olivier et Elodie, Guillaume, Maria, Jeremy, Marion, Clara ; les doctorants de RepMus ; les toulousains (les vrais) Benoit, Manu, Manouel ; Marie et Miguel, los amigos granainos, Marga et Stéphane, quelque part loin d'ici ;

... et surtout à mes parents, grands-parents, et à toute ma famille.

Table des matières

Introduction **1**

I Contexte et aspects théoriques (1)
Synthèse sonore et applications musicales **13**

1 Synthèse sonore **15**
1.1 Son et synthèse numériques 15
1.1.1 Eléments d'historique 15
1.1.2 Le support numérique 16
1.2 Techniques et modèles de synthèse 17
1.2.1 Techniques basées sur des signaux numériques 18
1.2.2 Techniques spectrales 21
1.2.3 Modèles abstraits ou non linéaires 31
1.2.4 Modèles physiques 33
1.3 Conclusion 34

2 Systèmes de synthèse sonore et de contrôle de la synthèse **35**
2.1 Synthèse sonore et représentations du son 35
2.2 Systèmes de synthèse 37
2.3 Langages de synthèse 39
2.3.1 Csound et les langages Music N 39
2.3.2 Un langage pour le traitement des signaux 41
2.3.3 Langages pour les modèles physiques 41
2.3.4 Environnements temps-réel 42
2.4 Contrôle de la synthèse 44
2.4.1 Outils et environnements de contrôle 45
2.4.2 Contrôle de systèmes additifs et spectraux 46
2.4.3 Le contrôle avec les modèles physiques 49
2.4.4 Le contrôle dans les environnements temps réel 51
2.5 Conclusion 53

3 La synthèse sonore dans la composition **55**
3.1 Instruments et partitions 55
3.2 Notation 57
3.3 Interprétation 60
3.4 Conclusion 61

II Contexte et aspects théoriques (2)
Composition Assistée par Ordinateur **63**

4 Modélisation informatique pour la composition musicale **65**
4.1 Composition assistée par ordinateur 65
4.1.1 Origines et principes 65
4.1.2 Objets et processus 67
4.1.3 Modèles et représentations informatiques 68
4.1.4 Modèles compositionnels 69
4.1.5 CAO et Modélisation 70
4.2 Langages de programmation 71
4.2.1 Programmation et composition 71
4.2.2 Programmation visuelle 72
4.3 OpenMusic : un langage de programmation visuelle pour la composition musicale 74
4.3.1 Architecture du langage 74
4.3.2 Environnement de travail 75
4.3.3 Patches : programmes visuels 76
4.3.4 Fonctionnalités de programmation avancées 77
4.3.5 Intégration données/programmes 81
4.3.6 Applications musicales 84
4.4 Conclusion 85

5 Synthèse sonore et CAO : vers une modélisation compositionnelle **87**
5.1 Problématiques et aspects théoriques 87
5.1.1 Représentations symboliques 87
5.1.2 Ambivalence des objets musicaux 89
5.1.3 Entre symboles et signaux 90
5.1.4 Synthèse et modèles 91
5.1.5 Le sens des modèles 92
5.1.6 Caractéristiques pour une approche compositionnelle 92
5.2 Environnements de CAO pour la synthèse 94
5.2.1 Langages “classiques” 94
5.2.2 Langages visuels 96

5.2.3 Contrôle de la synthèse sonore dans OpenMusic : précédents travaux . . . 97
5.3 Conclusion . . . 101

III Outils de programmation pour la représentation et la synthèse des sons en composition assistée par ordinateur 103

6 Programmation visuelle avec les sons et la synthèse sonore 105
6.1 Architecture et utilitaires pour l'audio . . . 105
6.1.1 LibAudioStream : un système pour la manipulation et le rendu de ressources audio . . . 106
6.1.2 Nouvelles fonctionnalités audio dans OpenMusic . . . 107
6.2 Bibliothèque de fonctions pour le traitement des sons . . . 110
6.3 Intégration des systèmes de synthèse . . . 113
6.4 La bibliothèque OM-SuperVP . . . 116
6.5 Conclusion . . . 119

7 Descriptions sonores de bas niveau : formats de données et protocoles de transfert 121
7.1 Descriptions sonores de bas niveau . . . 121
7.2 Lecture et écriture de fichiers . . . 122
7.3 Modèle instrumental : le format MIDI . . . 123
7.4 Descriptions sonores généralisées : SDIF/OSC . . . 125
7.4.1 Le format SDIF . . . 125
7.4.2 Protocole OSC . . . 129
7.5 Outils pour la manipulation des descriptions sonores au format SDIF dans OpenMusic . . . 130
7.5.1 Structures de données SDIF . . . 130
7.5.2 Ecriture des données . . . 131
7.5.3 Lecture et interprétation des données . . . 133
7.5.4 Visualisation : SDIF-Edit . . . 134
7.6 Conclusion . . . 136

8 Analyse du signal sonore 139
8.1 Le son comme source de matériau musical . . . 139
8.2 L'abstraction du matériau sonore . . . 140
8.3 Outils pour l'analyse dans OpenMusic . . . 143
8.3.1 Les données de la forme d'onde numérique . . . 143
8.3.2 Analyse spectrale . . . 144
8.3.3 Fréquence fondamentale . . . 144

8.3.4 Segmentation . . . 146
8.3.5 Analyse additive . . . 146
8.4 Utilisation de l'analyse dans les processus de synthèse . . . 149
8.5 Conclusion . . . 153

9 Structures de données pour la synthèse sonore 155
9.1 Structures de données élémentaires . . . 156
9.1.1 Fonctions et enveloppes . . . 156
9.1.2 Matrices . . . 158
9.2 OMChroma : vers une intégration des aspects comportementaux dans les structures de données pour la synthèse . . . 159
9.2.1 Intégration données/programmes . . . 160
9.2.2 Aspects "macro"-compositionnels . . . 162
9.3 Un système pour la génération de formes musicales avec les descriptions sonores . . . 163
9.3.1 Structure . . . 163
9.3.2 Manipulation des données . . . 166
9.3.3 Connexion au domaine de la synthèse . . . 168
9.3.4 Règles d'appariement des paramètres . . . 169
9.3.5 Mise en oeuvre dans les modèles compositionnels . . . 172
9.4 Conclusion . . . 174

IV Structures musicales et organisations temporelles 175

10 Aspects temporels dans la composition et la synthèse sonore 177
10.1 Structuration temporelle en composition assistée par ordinateur . . . 177
10.1.1 Représentation des structures temporelles . . . 178
10.1.2 Calcul des structures temporelles . . . 181
10.2 Le temps et la synthèse sonore . . . 184
10.2.1 Niveaux d'intégration . . . 185
10.2.2 Temps discret et temps continu . . . 186
10.2.3 La notion d'évènement . . . 188
10.2.4 Objets et phénomènes continus . . . 189
10.2.5 Sur le temps dans les outils de synthèse sonore . . . 190
10.3 Conclusion . . . 191

11 Modélisation et contrôle des processus de synthèse dans le temps 193
11.1 La *maquette* . . . 194
11.1.1 Un séquenceur de programmes . . . 194

11.1.2 Un objet musical . . . 196
11.1.3 Un programme . . . 196
11.2 Application à la composition . . . 199
11.2.1 Une représentation temporelle des modèles . . . 199
11.2.2 Composer avec les sons . . . 200
11.3 Maquettes de synthèse : contrôle de la synthèse dans le temps . . . 203
11.3.1 Contrôle de la phase d'assemblage des composants . . . 203
11.3.2 Exemples . . . 205
11.4 Gestion et contrôle des phénomènes continus . . . 210
11.4.1 Création d'objets continus . . . 210
11.4.2 Articulations entre les évènements . . . 213
11.5 La maquette comme structure et interface du modèle . . . 219
11.6 Conclusion . . . 222

12 Temporalités mixtes et notation musicale 225
12.1 Partitions et programmes . . . 225
12.2 Le *sheet* : une nouvelle partition dans OpenMusic . . . 228
12.3 Systèmes temporels et représentation graphique . . . 229
12.3.1 Non-linéarité de la représentation temporelle . . . 229
12.3.2 Alignement temps/espace . . . 230
12.3.3 Systèmes temporels . . . 231
12.3.4 Notation rythmique et notation proportionnelle . . . 231
12.3.5 Objets en temps continu . . . 233
12.4 Partition programmable . . . 234
12.4.1 Aspects fonctionnels dans l'éditeur de *sheet* . . . 234
12.4.2 Intégration du *sheet* dans l'environnement de CAO . . . 235
12.5 Conclusion . . . 237

Conclusion 239

Annexe : Applications 245
Composition . . . 247
Analyse et musicologie . . . 254
Pédagogie musicale . . . 257

Bibliographie 263

Introduction

Les travaux présentés dans cette thèse se situent dans le domaine de l'informatique musicale, et plus particulièrement de la composition musicale assistée par ordinateur. Comme son nom l'indique, l'informatique musicale a pour but de réunir informatique et musique dans une même discipline, destinée à étudier des problématiques aussi bien musicales (pour lesquelles l'informatique est susceptible d'apporter des solutions) qu'informatiques (pour lesquelles la musique constitue un cas d'étude original).

Par composition musicale assistée par ordinateur, nous entendons traiter de l'ordinateur et de l'informatique en général comme outils de composition, comme moyens d'exprimer et d'actualiser des idées, formalismes ou intuitions musicales. Dans ce contexte, nous nous intéresserons plus précisément à la question de la synthèse sonore dans la composition musicale. Si la synthèse sonore permet en effet de travailler sur la construction de sons de manière inédite, son insertion dans les pratiques et les outils de composition, nous essaierons de le montrer, demeure souvent problématique.

Le son, objet de composition

Les innovations technologiques de la fin du XIX^{e} siècle ont radicalement changé la conception du son dans la musique occidentale. L'enregistrement a permis de fixer les sons sur des supports matériels tangibles (d'abord sur microsillon puis sur bandes magnétiques). Le son enregistré n'était dès lors plus un phénomène insaisissable, perceptible dans une conjonction de conditions particulières, mais devenait un objet concret, observable et manipulable. Si l'on savait naturellement que toute forme musicale se ramenait à un son, que faire de la musique c'était créer (directement ou indirectement) du son, cette "matérialisation" du son lui donnait cependant désormais la condition nouvelle d'objet (musical) "en soi"[1] [Schaeffer, 1966].

La transduction électroacoustique, passage de l'énergie électrique à une énergie mécanique (acoustique), fait également partie de ces conquêtes technologiques majeures pour la musique. Concrètement, il s'agit des microphones et haut-parleurs, donc d'une technologie également liée à l'enregistrement. Cependant, celle-ci implique aussi la capacité de convertir en signaux sonores tout signal électrique, toute onde créée par des moyens électriques ou électroniques.

Ces nouvelles possibilités ont permis de conduire de nombreuses expériences amenant un contrôle et une compréhension nouvelle des phénomènes sonores, et ont contribué à un profond renouvellement de la pensée et de l'écriture musicale. Selon la formule consacrée, il ne s'agissait dès lors plus de composer avec les sons, mais de composer le son lui-même ; *reconsidérer le matériau sonore, les sons mêmes qui avaient été utilisés jusqu'ici* [Stockhausen, 1988b].

[1] Cette condition sera par ailleurs l'un des fondements du courant de musique "concrète".

> "[...] les compositeurs [...] ne peuvent plus se contenter d'accepter les phénomènes sonores comme acquis, mais [...] aspirent à faire pénétrer autant que possible leurs conceptions formelles dans les sons, afin d'atteindre une nouvelle correspondance entre matériau et forme, entre une microstructure acoustique et une macrostructure musicale." [Stockhausen, 1988b]

Le domaine du timbre sonore s'ouvrait ainsi dans une conception globale du phénomène sonore, libérée des contraintes des moyens de production instrumentaux traditionnels. Les moyens électroacoustiques permettent de décrire des sons sans nécessairement se conformer aux paramètres traditionnels tels que les hauteurs, les durées, les intensités, mais de façon plus ouverte. Le timbre, considéré comme structure essentielle d'un son, prend alors une place centrale dans la composition [Barrière, 1991].

Les avancées technologiques ont ainsi guidé le développement de la musique électroacoustique,[2] dont les précurseurs tels que Edgard Varèse, Karlheinz Stockhausen, Iannis Xenakis et bien d'autres, explorèrent l'univers sonore à l'aide des premiers synthétiseurs et magnétophones analogiques. Elles ont permis également celui de la musique spectrale qui, si elle reste la plupart du temps instrumentale, exploite elle aussi cette connaissance du son et des phénomènes de la perception à l'aide des instruments de l'orchestre.

Le son et la forme se rapprochent donc dans l'activité comme dans la pensée musicale : il existe un *ordre sonore total* [Stockhausen, 1988c] sur lequel le musicien agit en composant.

Le numérique

L'apparition des technologies numériques et de l'informatique est une nouvelle étape décisive, d'une portée tout aussi importante sur la création musicale. Le son représenté par une suite de nombres sur un support numérique (par exemple sur un disque dur) se "concrétise" d'avantage, mais s'abstrait à la fois de son support. Il peut être fidèlement restitué, transmis, observé en précision sous la forme d'un document ou d'un flux numérique.

Avec le numérique, les manipulations sur les objets sonores ne sont par ailleurs plus limitées par les contraintes et la relative imprécision des supports physiques antérieurs. Celles-ci, réalisées sur les bandes magnétiques à l'époque de la musique concrète, sont aujourd'hui des opérations élémentaires avec les outils informatiques : on coupe, colle, mélange, et déforme les sons à l'aide de logiciels de plus en plus puissants et précis.

En principe, le son est donc intégralement "composable", au même titre que des structures musicales traditionnelles : il s'agit de déterminer les valeurs numériques dont il est constitué. Le support numérique permet la visibilité, l'édition, la correction du matériau sonore. Il révèle la temporalité des objets sonores et permet ainsi d'en envisager une structuration totale dans une activité de composition.

---

[2] Nous emploierons le terme de musique *électroacoustique* pour désigner les formes et pratiques musicales faisant appel à des modes de productions sonores synthétiques et non uniquement instrumentaux.

Cependant, la quantité des données nécessaires pour décrire un son sous forme numérique et la faiblesse structurelle de cette représentation (une suite de nombres) ont rapidement fait apparaître la nécessité d'une méthodologie plus avancée :

> "Etant donné que des millions d'échantillons sont nécessaires pour composer un son, personne n'a le temps d'entrer tous ces échantillons pour un morceau de musique. C'est pourquoi les échantillons doivent être synthétisés à l'aide d'algorithmes ou dérivés d'enregistrements naturels." [Smith, 1993]

Les techniques numériques d'analyse et de traitement du signal vont alors également permettre d'accéder à une connaissance plus approfondie encore, et à un contrôle inédit des structures et logiques internes du son. Ces connaissances, bien sûr, se répercuteront sur les possibilités de synthèse sonore.

> "[...] l'enregistrement et la transcription de l'information acoustique [ont produit] une nouvelle intelligence des choses en suscitant une écriture de programmes de simulation qui, par la formulation de modèles théoriques, permettaient d'en approfondir la formalisation." [Dufourt, 1991]

Sur ce terrain fertile se sont donc développées différentes techniques et esthétiques musicales. L'exploration du potentiel de la synthèse numérique en termes de création de timbres et de structures sonores est aujourd'hui encore un des défis posés à l'informatique musicale.

Applications musicales : des outils de synthèse aux outils de contrôle

Les algorithmes et techniques développés pour la synthèse sonore ont donné lieu à la création de nombreux outils logiciels permettant la mise en oeuvre de ces techniques dans un contexte musical [Moorer, 1977]. Nous appelons ces outils *systèmes de synthèse*.

Cependant, d'un point de vue musical, les potentialités offertes par ces systèmes sont difficilement accessibles pour les musiciens et compositeurs, en manque de repères dans le domaine et (souvent) de connaissances techniques suffisantes pour les manipuler de manière efficace. Leur utilisation nécessite en effet l'appropriation par le compositeur de systèmes puissants mais complexes, dont l'appréhension et le paramétrage, nécessaires à la production des sons de synthèse, peuvent détourner et priver son travail d'une portée réellement créative et musicale.

> "Mis à part quelques exceptions notables, les applications concrètes de la synthèse sonore restent bien en retrait par rapport à leurs intentions artistiques [...]. Bien qu'il ne subsiste guère de doute en théorie quand aux potentialités musicales de la synthèse sonore, on n'est pas encore parvenu à un matériau sonore synthétique suffisamment souple pour répondre aux exigences de la composition, c'est-à-dire de sa mise en œuvre dans un contexte de langage musical." [Eckel, 1993]

Ainsi la question du contrôle de la synthèse a-t-elle accompagné le développement des systèmes de synthèse. On appellera outil de *contrôle de la synthèse* un environnement réalisant une abstraction des systèmes de synthèse et de traitement du signal afin d'établir

des représentations de plus haut niveau permettant, le plus souvent à l'aide d'interfaces graphiques, de spécifier les paramètres nécessaires à leur mise en œuvre pour la production sonore. Les abstractions réalisées par les environnements de contrôle de la synthèse sont alors souvent synonymes d'une diminution du nombre et de la complexité des paramètres des systèmes de synthèse : c'est en effet sur des structures de données de complexité raisonnable que peut en principe se développer une pensée, une démarche musicale.

Nous essaierons de montrer que cette réduction engage également une réduction du degré de liberté dans un système, et une certaine rigidité dans les processus créatifs qui ne permet que partiellement d'exploiter le potentiel (musical, en particulier) de la synthèse sonore.

Aujourd'hui, les formes de musique électroacoustique se diversifient, intégrant plus étroitement les aspects calculatoires, l'interaction avec les technologies de temps réel et les interfaces gestuelles. Le développement très marqué de la musique "mixte" favorise également l'association des instrumentistes avec les technologies et les sons de synthèse. Dans ce renouveau des conceptions musicales, l'ordinateur intervient désormais à tous les niveaux de la création, depuis la composition formelle jusqu'à la synthèse des signaux sonores [Risset, 1996]. Le compositeur, par l'intermédiaire de l'ordinateur, s'empare donc du phénomène musical d'une façon inédite, redéfinissant son propre positionnement ainsi que celui des concepts musicaux auxquels il fait appel.

Ecrire le son

Dès lors que la production sonore est assurée par l'informatique et non plus par un interprète, l'écriture musicale doit permettre de déterminer des sons dans une démarche qui leur attribuerait une forme, un sens musical.

Une écriture pourrait en premier lieu amener à penser un système de notation symbolique, à l'intérieur duquel des signes seraient associés à des paramètres sonores, permettant d'organiser le son de manière formelle et abstraite, et au vu des nouvelles possibilités accessibles par la synthèse. Cependant, étant donné le faible recul dont nous disposons du fait de l'apparition relativement récente de la synthèse sonore dans la composition musicale, il n'existe pas de réel consensus, ni de pratique établie dans le domaine, qui permettrait de déterminer à la fois ces paramètres, et la nature des signes qui pourraient y être associés. Il est donc encore trop tôt, voire inapproprié, d'envisager la création d'un tel système. Pour l'heure, les aspects arbitraires et personnels dans les conventions d'écriture nous semblent importants à maintenir afin d'éviter un goulot esthétique pouvant être causé par les restrictions d'un système figé et préétabli.

Il est encore opportun de se demander comment un compositeur conçoit, ou devrait concevoir un son pris dans sa totalité : par reproduction, en essayant de s'approcher ou de transformer un son existant ou imaginé ? Par une conception logique de sa structure ? Par pure intuition ?

L'articulation du signal sonore et du signe musical

Le domaine de la composition relève principalement d'un niveau symbolique : les représentations symboliques permettent de considérer le son *abstrait* de sa réalité acoustique, et favorisent la réflexion formelle sur les structures sonores et musicales. C'est à ce niveau que travaille le compositeur de musique instrumentale lorsqu'il écrit une pièce sur une partition. La représentation symbolique du phénomène sonore, dans le sens d'une représentation ayant un sens pour un compositeur, lui permettra d'avoir une emprise formelle sur celui-ci ou sur ses processus de génération.

A l'opposé, le domaine sonore est en principe celui de la précision, de la continuité, qui sont plus difficiles à appréhender musicalement. Il relève de l'interprétation dans le schéma instrumental, mais doit désormais être intégralement contrôlé par le compositeur avec l'électroacoustique.

Si l'abstraction permet d'élever les représentations du son au rang de symboles, nous avons cependant noté qu'une abstraction masquant complètement le domaine du signal serait une restriction inacceptable dès lors que l'on a entrevu les possibilités offertes par la synthèse sonore. Entre signal et symbole, les représentations analytiques, les descriptions structurées de plus ou moins haut niveau, les objets musicaux plus ou moins classiques, constituent ainsi autant de *niveaux d'abstraction* intermédiaires et successifs permettant de positionner une réflexion (scientifique, musicale ou autre) sur le son, basée sur les représentations correspondantes. Le niveau d'abstraction, positionnement sur cette échelle allant du signal, du matériau brut vers les descriptions sonores structurées puis les objets musicaux symboliques, est ainsi une notion fondamentale, dont la maîtrise et la perméabilité, selon l'appréciation de l'utilisateur d'un système de composition, est en mesure d'assurer son potentiel musical.

Le problème de l'articulation du signal sonore et du signe musical se pose donc comme celui d'intégrer et de contrôler des données et structures musicales liées au signal sonore dans un espace de représentations symboliques. Une autre problématique, relative à ce lien entre les domaines du signal et du symbole, concernera les conceptions et représentations du temps, notamment à travers le traitement des structures temporelles discrètes et continues. Ces questions entrent en effet de plus en plus au cœur des problématiques liées aux environnements informatiques et aux pratiques de composition, et les outils actuels ne permettent que partiellement aux compositeurs d'explorer ce type de dualité dans les concepts et objets musicaux. Par ailleurs, elles doivent être envisagées en complémentarité avec celle du passage du symbolique vers le signal, qui permet finalement la mise en place effective d'un outil de composition.

Un environnement de composition musicale assistée par ordinateur orienté vers la synthèse sonore ne doit donc pas imposer de systèmes de représentations mais en permettre la construction, favorisant par là la constitution d'un niveau symbolique, porteur de sens et support formel pour l'écriture, et propre à chaque situation.

La modélisation en informatique musicale

La modélisation est un concept que l'on retrouve plus ou moins explicitement dans la plupart des applications et systèmes informatiques [Berthier, 2002]. Modéliser un objet (un phénomène, un problème, une situation, etc.) permet de mettre en évidence un point de vue formel sur celui-ci, et de mettre cette formalisation en œuvre par le calcul dans un objectif expérimental, créatif, ou autre.

Sous cette définition, on peut considérer l'informatique musicale comme une discipline visant à formaliser différents aspects du phénomène musical et à construire les modèles informatiques correspondants, dans un objectif de compréhension, ou d'analyse, dans un premier temps, puis souvent de création et de production musicale. Règles de l'harmonie, structures temporelles, synthèse et propagation des ondes acoustiques, geste instrumental, sont autant d'axes de recherches de l'informatique musicale qui ont permis, suivant les avancées technologiques, les pratiques musicales, et les intuitions de ces cinquante dernières années, de modéliser ce phénomène musical sous différentes formes dans le cadre de sous-disciplines comme l'analyse et la synthèse du signal sonore, le montage numérique, la spatialisation, ou encore la composition assistée par ordinateur. Parmi celles-ci, nous nous intéresserons dans cette thèse à l'analyse/synthèse du signal, dans un premier temps, puis à la composition assistée par ordinateur (ou CAO).

On parle souvent de modèles dans le domaine de la synthèse sonore, principalement pour catégoriser les différentes techniques et systèmes existants. Dans ce cas, c'est le son en tant que phénomène acoustique qui fait l'objet de la modélisation. De ce point de vue, les systèmes de synthèse proposent donc des représentations du son issues des modèles de synthèse et manipulables par le calcul, et les systèmes de contrôle de la synthèse sont des outils pour la construction de représentations adaptées à la mise en œuvre de ces modèles.

La CAO, quant à elle, met en avant cette notion de modélisation informatique appliquée spécifiquement à l'activité de composition musicale, en s'appuyant notamment sur des langages de programmation spécialisés [Assayag, 1998]. Le modèle n'est plus alors un simple outil, mais devient l'objet central d'un processus de composition, et par conséquent des systèmes informatiques correspondants. S'agissant de processus de composition, nous utiliserons dans ce cas, pour simplifier, le terme de *modélisation compositionnelle*. C'est en nous appuyant sur cette idée que nous avons essayé d'aborder la question de l'écriture du son.

Modélisation compositionnelle pour la synthèse sonore

L'informatique permet d'évoluer à toutes les échelles de la production sonore et musicale, à la fois dans la formalisation des structures musicales et dans la création de sons de synthèse. Les pratiques musicales contemporaines (utilisation de l'analyse et la synthèse sonore, musique électroacoustique, mixte, etc.) révèlent ainsi l'intérêt actuel d'un rapprochement de ces deux aspects, qui est aujourd'hui techniquement accessible (ce qui était

moins évident il y a encore quelques années, quand des dispositifs dédiés devaient être utilisés pour le traitement de données symboliques et pour la synthèse sonore). L'enjeu est donc de dépasser cette simple cohabitation pour aller vers des environnements capables d'intégrer les différents concepts, données et processus mis en jeu, dans une démarche musicale et un temps de composition commun, et à l'intérieur desquels les domaines du son et de la composition musicale pourraient communiquer et interagir. Pour reprendre [Stockhausen, 1988c], il est question de l'intégration de la pensée musicale, formelle et organisatrice, dans le matériau sonore.

Les connexions sont certes possibles entre les environnements de CAO et de synthèse sonore, notamment grâce aux protocoles de communication et au partage des données ; les premiers permettent alors de modéliser la partie compositionnelle, et les systèmes de synthèse sont utilisés pour interpréter, produire des sons correspondant aux formes créées dans la partie compositionnelle.

Cependant, en légère opposition à cette démarche qui maintient implicitement le schéma synthèse/contrôle, ainsi qu'une certaine rigidité dans le niveau d'abstraction adopté, l'approche que nous avons souhaité développer part de la CAO et étend ses paradigmes et outils à ce nouveau champ d'application que constituent le son et la synthèse sonore.

La musique peut en effet se manifester sous la forme d'une onde sonore (dans le domaine "physique"), mais aussi par des structures et processus formels (dans le domaine de la composition musicale) ; la première forme se trouvant être une expression de la seconde, par l'intermédiaire de l'interprétation ou de la synthèse sonore. Si les travaux liés à la synthèse sonore se sont donc traditionnellement (et naturellement) concentrés sur la première de ces deux manifestations, dans laquelle la démarche de modélisation informatique reste globalement extérieure à la composition, nous essaierons donc, dans le cadre de cette thèse, de traiter la seconde forme, et de considérer le son et la synthèse sonore dans le cadre des modèles compositionnels.

La démarche que nous avons suivie consiste donc à penser et représenter le son musical sous forme de structures et de programmes permettant sa modélisation au niveau compositionnel. Le problème de la synthèse sonore est alors abordé non pas à partir de considérations liées au domaine du signal et visant à en dégager des représentations abstraites (approche "*bottom-up*"), mais plutôt partant de considérations d'ordre compositionnel et de représentations de haut niveau visant plus ou moins explicitement une réalisation par la synthèse (approche "*top-down*").

Deux notions importantes dans cette approche peuvent déjà être soulignées :

– Une représentation informatique du son en tant que programme, qui unifie le son vu comme un ensemble de données et les processus musicaux conduisant à la production de ces données.

– Une conception du son en tant qu'objet d'un modèle, qui n'est pas un résultat, ou une cible mentalement préétablie et visée par un processus, mais qui se constitue par l'interaction du compositeur avec ce modèle.

En pratique

D'un point de vue applicatif, ce travail s'insère dans le cadre du développement de l'environnement de CAO OpenMusic [Agon, 1998] [Bresson *et al.*, 2005a], un langage de programmation visuel pour la composition musicale créé et développé à l'Ircam. Il peut donc être vu concrètement comme une extension de cet environnement, permettant de travailler sur le son comme objet compositionnel. Notre objectif n'est cependant pas de faire de OpenMusic un environnement de traitement du signal (des outils externes seront majoritairement utilisés pour cela), mais un espace dans lequel un compositeur pourrait travailler sur et avec les processus de traitement et de synthèse sonore suivant des niveaux de représentation et d'abstraction ouverts et suffisamment élevés. La manipulation des structures et des données liées au son dans ce nouvel environnement nous permettra alors d'établir les relations entre la composition musicale et le signal sonore.

Les problématiques de notre sujet sont donc abordées dans le cadre d'une architecture logicielle opérationnelle permettant l'expérimentation et la mise en œuvre des différentes propositions apportées. L'environnement correspondant, baptisé OMSounds, peut être divisé en deux parties complémentaires :

– La première partie concerne principalement les outils de bas niveaux permettant d'orienter ou de mettre en relation les processus compositionnels avec la synthèse sonore et le domaine du signal en général. Elle comprend diverses stratégies et protocoles de communication entre l'environnement de CAO et les systèmes d'analyse/synthèse sonore, ainsi que des structures de données pour la représentation des différentes entités mises en jeu dans la modélisation du son.
– La deuxième partie traite les questions des organisations temporelles et contient des interfaces et éditeurs de haut niveau, qui permettront de former des structures musicales à partir des processus de bas niveau évoqués ci-dessus. Une représentation visuelle du phénomène sonore en tant que modèle compositionnel est notamment proposée à travers l'éditeur de *maquettes*.

Différentes caractéristiques de cet environnement, que nous essaierons de souligner dans ce document, offriront aux compositeurs des nouvelles possibilités pour le traitement et la synthèse des sons dans le cadre de la CAO :

– L'intégration dans OpenMusic permet l'utilisation des outils de programmation pour la spécification et l'automatisation des tâches liées au traitement du signal.

– Les concepts informatiques comme l'abstraction, la modularité, ainsi que les interfaces correspondantes, permettent à l'utilisateur de structurer ses programmes et d'établir les différentes échelles d'abstraction du matériau musical.
– Le niveau d'abstraction établi comme frontière entre les deux parties de l'architecture que nous avons distinguées n'est pas imposé par l'environnement. Les processus et modèles compositionnels pourront ainsi être constitués suivant un positionnement choisi par le compositeur sur les échelles d'abstractions, favorisant un traitement personnalisé des ambiguïtés entre les domaines symbolique et signal, ou encore discret et continu.
– Une proposition de représentation symbolique du son (*maquette*) unifie les domaines micro- et macro-compositionnels. Une maquette peut rendre compte à la fois de structures internes aux phénomènes sonores et de l'organisation de formes musicales à grande échelle.
– Le caractère générique des outils proposés permettra le libre développement de processus et modèles les plus divers, en créant des objets sonores à partir de formes musicales abstraites, en soumettant du matériau sonore aux traitements musicaux, ou encore en intégrant différentes démarches dans un seul et même modèle.
– La possibilité d'établir constamment des relations étroites entre le son et la notation musicale traditionnelle est par ailleurs maintenue à tous les niveaux et dans les deux parties de cet environnement.

Nous espérons donc, à l'issue de ce travail, montrer que cet environnement constitue une réalisation originale, permettant une nouvelle approche musicale du son et de la synthèse sonore.

D'un point de vue méthodologique, nous avons essayé de mettre en avant théorie et pratique musicales, afin de nous approcher de la réalité du sujet par l'intermédiaire direct des utilisateurs. Ainsi l'expertise musicale de ce travail repose-t-elle en grande partie sur la collaboration et le dialogue avec des compositeurs. Complémentairement aux écrits théoriques, auxquels il sera souvent fait référence, nous nous sommes appuyés sur des collaborations étroites avec des compositeurs travaillant intensément sur les problématiques musicales de la synthèse sonore (notamment avec Marco Stroppa et Karim Haddad), ainsi que sur une série d'entretiens réalisés auprès d'autres compositeurs ou théoriciens attachés de manières diverses à ces mêmes problématiques (Hugues Dufourt, Philippe Manoury, Claude Cadoz). Des références à ces entretiens seront donc également rencontrées au long de cet exposé.

Plan de la thèse

Compte tenu du caractère pluridisciplinaire de cette thèse, on retrouvera dans les différentes parties de ce document des considérations relevant des domaines musicaux et informatiques, portant notamment sur les langages de programmation (visuels, en particulier), les interfaces homme-machine, ou encore sur le traitement du signal.

La partie I est consacrée à une présentation du contexte théorique et technologique lié à la synthèse sonore et à ses applications musicales. Dans un premier temps (chapitre 1), nous donnons un aperçu des techniques existantes dans le domaine du traitement du signal pour la représentation et la synthèse des signaux sonores. Le chapitre 2 traite de l'application de ces techniques pour la composition, avec les systèmes de synthèse, les langages de synthèse et les environnements de contrôle de la synthèse. Dans le chapitre 3, la question de la synthèse est reconsidérée d'un point de vue musical : les notions telles que l'instrument, ou la partition de synthèse, méritent en effet une réflexion et une approche spécifique dans ce nouveau contexte.

Dans la partie II, nous introduisons le schéma directeur de notre travail, qui trouve son origine dans le domaine de la CAO. Nous nous intéressons dans le chapitre 4 aux concepts de la CAO, en ce qu'elle permet d'approcher les problématiques musicales d'un point de vue compositionnel, et en particulier à l'idée de langages de programmation en tant qu'outils de modélisation et de création musicale. L'environnement OpenMusic, qui sera la plateforme de développement et d'intégration de nos travaux, sera alors présenté. Le chapitre 5 introduira ensuite l'idée d'une approche de la synthèse sonore inspirée des principes et techniques de CAO, et visant à modéliser le son selon des représentations et processus de haut niveau liés à la formalisation musicale. Des notions théoriques attenantes à cette démarche seront présentées, ainsi que des précédents travaux réalisés dans cette direction (notamment dans OpenMusic).

La partie III initie la description concrète de l'environnement de composition OM-Sounds, avec la présentation des outils de programmation visuelle relatifs à la première des deux parties que nous avons distinguées dans notre architecture. Les fonctionnalités directement liées aux processus de synthèse seront présentées dans le chapitre 6, puis le chapitre 7 détaillera les outils relatifs à la manipulation, au traitement et au transfert des données de description sonore de bas niveau. Le chapitre 8 sera ensuite concentré sur les outils d'analyse sonore, permettant d'extraire données et matériau musical de sons existants. Enfin le chapitre 9 traitera des structures de données entrant en jeu dans ces processus liés au son et à la synthèse sonore, et permettant de créer, visualiser et manipuler les structures utilisées pour la création d'un signal numérique.

Pour finir, la partie IV se placera plus spécifiquement au niveau de l'écriture musicale et de la structuration globale des modèles, c'est-à-dire à celui du deuxième aspect distingué dans notre architecture. Les différents repères et interactions temporelles pouvant intervenir à ce niveau, mais également en général dans les relations entre objets et programmes de

synthèse, introduisent alors une problématique fondamentale liée à la gestion et à l'organisation du temps. Cette partie est ainsi principalement dédiée à des aspects de notation et de représentation temporelle. Le chapitre 10 présentera les problématiques générales liées au temps dans la composition assistée par ordinateur, et plus particulièrement avec la synthèse sonore. Dans le chapitre 11, un système de représentation temporelle est proposé, intégrant synthèse sonore et liens entre symbolisme et signal dans un contexte et une structure de haut niveau. Enfin, nous présenterons dans le chapitre 12 un nouvel éditeur développé pour la création de partitions programmables mettant en jeu notation instrumentale et électronique.

Nous terminons avec quelques exemples d'applications dans les domaines de la composition, de l'analyse, et de la pédagogie musicale.

Première partie

Contexte et aspects théoriques (1) Synthèse sonore et applications musicales

Chapitre 1

Synthèse sonore

La synthèse numérique permet de créer des sons artificiels par le calcul, et les différents algorithmes et paradigmes de synthèse proposent autant de représentations des sons pour la création musicale.

Si les travaux que nous présenterons par la suite vont dans le sens d'une abstraction du domaine du traitement du signal pour la conception d'un environnement de composition de plus haut niveau, ces représentations resteront présentes et seront souvent déterminantes dans les démarches compositionnelles et les applications correspondantes. Dans ce premier chapitre, nous donnons donc un aperçu des principales techniques et paradigmes existants pour la synthèse.

1.1 Son et synthèse numériques

1.1.1 Eléments d'historique

Pour reprendre l'acception générale de la notion de synthèse, la *synthèse sonore* peut être considérée comme la construction de sons de manière artificielle, par combinaison d'entités élémentaires. Cependant, il n'existe pas pour le son de réel consensus quant à la nature d'un atome, ou d'éléments simples et indivisibles dont il serait constitué, ce qui laisse un champ ouvert à diverses interprétations. On peut donc difficilement dater les premières expériences de synthèse sonore, puisque (comme le rappelle par exemple [Risset, 1993]) depuis des temps bien antérieurs à l'apparition de l'électronique et de l'informatique, les principes de certains instruments anciens ou de pratiques musicales reposaient déjà sur la superposition d'éléments sonores élémentaires dans le but de constituer sons plus complexes.

A partir des années 1920, la technologie a permis de créer les premiers sons artificiels à l'aide de composants électroniques simples (principalement des oscillateurs). C'est alors

que l'on a réellement commencé à parler de synthèse sonore.[1] Oscillateurs, filtres, et autres technologies analogiques ont par la suite été largement utilisés pour la synthèse sonore, et ont ouvert un large champ d'investigation dans le domaine sonore et musical. La formalisation mathématique et la mise en œuvre des techniques de synthèse et de manipulation des ondes sonores ont alors permis de développer des outils de plus en plus complexes et performants.

Avec les technologies numériques, le signal sonore est représenté par une suite de valeurs numériques et peut donc être synthétisé par tout programme produisant en sortie une telle suite de nombres. C'est ce que l'on désignera par la *synthèse numérique*. Les premières expérimentations dans ce domaine datent des années 1950 : de nombreux travaux de recherches en informatique musicale se sont alors concentrés sur la mise en place de techniques et algorithmes permettant de générer ces formes d'ondes numériques par le calcul, à partir de formulations mathématiques et de structures de données, plus ou moins complexes, voire de sons préexistants (*traitement numérique du signal*).

1.1.2 Le support numérique

Dans un ordinateur, ou sur un support numérique, un son est stocké sous forme d'une suite de valeurs. Chacune de ces valeurs est appelée *échantillon* (ou *sample*, en anglais). Elle représente l'amplitude mesurée du signal sonore à un instant donné, déterminé par la fréquence d'échantillonnage, et est codée (quantifiée) sur un certain nombre de *bits*, dépendant de la précision souhaitée et des capacités de stockage disponibles. La figure 1.1 illustre schématiquement le processus de numérisation d'un signal. La norme actuelle (pour les Compact Discs, par exemple) effectue généralement une quantification sur 16 (voire 24) bits à une fréquence d'échantillonnage de 44,1 kHz (i.e. une seconde de son est codée sur 44100 échantillons de 16 bits chacun). Les dispositifs de conversion d'un signal analogique, capté par un microphone par exemple, en un signal numérique, et inversement de restitution d'un signal analogique à partir d'un signal numérique sont appelés respectivement Convertisseur Analogique/Numérique (CAN – ou ADC, pour *Analog/Digital Converter*) et Convertisseur Numérique/Analogique (CNA – ou DAC, pour *Digital/Analog Converter*).

Un son peut donc être crée (et enregistré) dans un ordinateur, soit par CAN, c'est-à-dire en numérisant un signal sonore analogique, soit par un calcul direct des valeurs des échantillons numériques (i.e. par synthèse numérique).

[1] Les ondes Martenot (1928), le Theremin (1924) sont ainsi parmi les premiers instruments électroniques utilisés en concert.

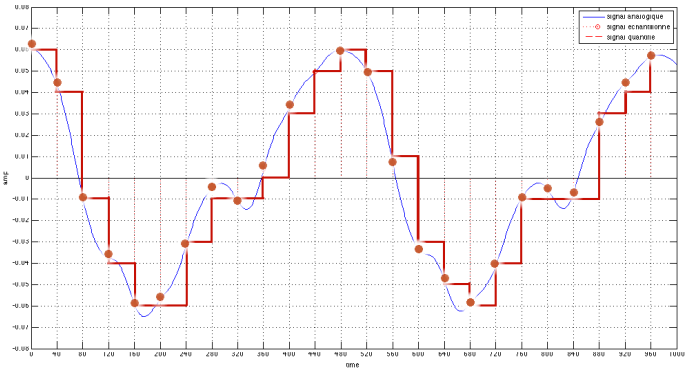


FIGURE 1.1: Numérisation d'un signal sonore. Les points représentent les échantillons relevés avec une période de 20 (fréquence d'échantillonnage $Fe = 1/20 = 0.05$). Le signal est ici quantifié sur 4 bits donc sur une palette de $2^4 = 16$ valeurs possibles.

1.2 Techniques et modèles de synthèse

Les différentes techniques existantes pour la synthèse sonore sont généralement classées par *modèles* : on trouvera régulièrement dans la littérature des références aux différents *modèles de synthèse* [De Poli *et al.*, 1991]. [Depalle et Rodet, 1993] définissent un modèle comme *une abstraction qui a pour but de condenser de manière formelle l'information véhiculée par le son.*

Les techniques de synthèse peuvent effectivement être rapprochées de la notion de modélisation évoquée en introduction, celle-ci étant entendue comme une activité visant à formaliser un objet dans un objectif donné (expérimental, de simulation, de création, etc.) Un modèle de synthèse sous-entend en effet une certaine conception de la structure formelle du son, de la manière dont il est implicitement constitué. Dans la mesure du possible, nous utiliserons cependant cette terminologie avec modération, afin d'éviter une possible confusion avec les "modèles compositionnels" qui feront l'objet ultérieur de notre travail.

Auparavant, notons que chaque type de synthèse sonore met en avant une certaine représentation du son, définissant un domaine dans lequel un éventuel utilisateur évoluera pour l'observer et le manipuler. Un modèle de synthèse pourra ainsi être utilisé dans une démarche d'analyse sonore, fournissant par le calcul une représentation donnée, ou point de vue particulier sur un signal sonore, ou dans une démarche symétrique de synthèse et de création sonore, s'agissant alors de créer un son à partir de cette représentation :

> "Un modèle sonore sert aussi bien à la représentation formelle d'un matériau sonore qu'à sa production." [Eckel, 1993]

Parmi les différentes techniques permettant de synthétiser ou d'analyser les sons, on identifiera quelques grandes "familles", ou types de modèles. On différencie en général fondamentalement les modèles de signaux et les modèles physiques. Parmi les modèles

de signaux, les techniques spectrales, basées sur une décomposition fréquentielle du son, sont les plus largement répandues (synthèse additive, "vocoder de phase", synthèse granulaire, modèles formantiques, etc.) D'autres techniques travaillent directement sur la représentation du signal numérique, soit par l'intermédiaire des mathématiques, avec la modulation de fréquence, par exemple, soit par lecture et échantillonnage de signaux préexistants. Les modèles physiques, quant à eux, mettent en avant les origines mécaniques de la production des sons.[2]

Cette catégorisation est discutable, et certaines techniques pourront parfois être considérées comme relevant de plusieurs types de modèles de synthèse. Sans insister sur leur classification, qui est donc sensiblement variable selon les auteurs et leurs finalités, nous essaierons dans les sections suivantes de donner une description générale des principales de ces techniques et paradigmes de synthèse.

1.2.1 Techniques basées sur des signaux numériques

Nous regroupons sous cette catégorie, parfois également qualifiée de "domaine temporel" (*time-domain models*), les techniques de synthèse s'appuyant sur des signaux numériques (variation d'une valeur dans le temps – voir figure 1.2) donnés initialement.

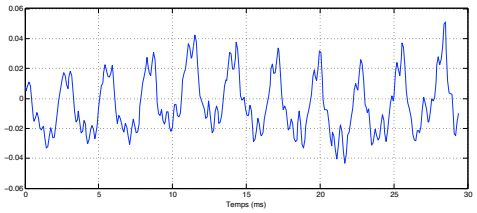


FIGURE 1.2: Représentation classique (amplitude / temps) d'une forme d'onde (ou d'un signal sonore).

Traitements numériques

Une forme d'onde numérisée (dans un fichier ou sous forme d'un flux d'échantillons) permet un certain nombre d'opérations et traitements sur le signal sonore (retards, traitements de la dynamique, spatialisation, etc.) Dans la mesure où nous nous intéressons à la synthèse de signaux, on pourra considérer tout type de traitement numérique sur un signal sonore comme une forme particulière de synthèse, dont l'un des paramètres est un signal donné initialement.[3]

[2] Afin de limiter le cadre de cette thèse, et même si nous en donns ici une brève description et l'évoquerons à quelques reprises dans les chapitres suivants, la synthèse par modèles physiques ne sera cependant pas spécifiquement traitée dans la suite de nos travaux.

[3] Les techniques spectrales nous donneront d'autres exemples dans ce sens.

Synthèse par table d'onde

La table d'onde est une zone de mémoire dans laquelle est stocké un signal échantillonné. Celle-ci peut être très simple (par exemple, les valeurs d'un sinus), ou contenir des signaux complexes (par exemple des valeurs aléatoires, ou des sons enregistrés). La lecture de cette table de valeurs, de manière circulaire et avec une vitesse (ou un pas de lecture) variable permet d'obtenir un signal périodique, dont la fréquence est un rapport entre la durée et la fréquence d'échantillonnage du signal contenu dans la table et ce pas de lecture. Cette technique est utilisée notamment pour créer des oscillateurs numériques (voir figure 1.3).

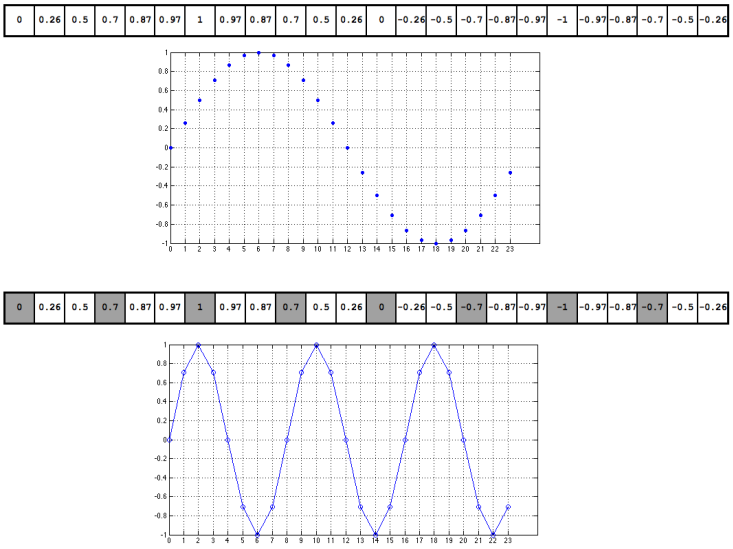


FIGURE 1.3: Table d'onde. Cette table contient 24 valeurs permettant d'approximer, en lecture cyclique, un signal sinusoïdal (figure du haut). Le parcours cyclique de la table avec différents "pas de lecture", ou intervalles entre deux valeurs prélevées dans le tableau (3, dans le cas représenté en bas), permet de contrôler la périodicité du signal produit en sortie.

La figure 1.3 montre que la lecture de la table avec différents pas de lecture permet de générer des signaux de différentes périodicités. Un simple facteur multiplicatif permet ensuite de contrôler l'amplitude de ces signaux : on dispose alors d'un oscillateur numérique, générateur de signaux périodiques, contrôlable par deux paramètres (amplitude et fréquence). L'utilisation simultanée de plusieurs oscillateurs permettra de générer des sons complexes par synthèse "additive" (voir section 1.2.2).

Sampling

Le *sampling*, ou échantillonnage, repose sur le principe de la table d'onde. Littéralement, échantillonner un signal signifie effectuer une discrétisation de celui-ci avec une période donnée (e.g. prendre un échantillon – une valeur – toutes les n microsecondes). On parle d'échantillonnage comme méthode de synthèse lorsque l'on utilise des tables d'ondes

contenant des sons numérisés (il s'agit donc généralement de tables de taille beaucoup plus importante que dans le cas de la figure 1.3).

Cette technique est notamment utilisée dans de nombreux synthétiseurs du commerce, permettant la reproduction de timbres à partir de sons pré-enregistrés, et répondant généralement à un contrôle de type MIDI[4] (par exemple à partir d'un clavier). Un son est alors produit par "variation" autour d'un timbre donné selon des hauteurs, durées, et nuances d'intensités voulues.

Dans ces systèmes, des sons de référence sont donc numérisés et stockés en mémoire. Ils sont ensuite transposés (variation de la vitesse de lecture de la table) et adaptés en amplitude pour atteindre les hauteurs et nuances visées. Généralement, les déformations produites par des transpositions excessives détériorent de façon notable la qualité du timbre (ou du moins sa ressemblance avec le timbre original). De même, les instruments traditionnels (acoustiques) présentent souvent des timbres très différents selon l'intensité avec laquelle ils sont joués. Dans la pratique, on utilise donc pour un même timbre un certain nombre de sons (correspondant à différentes plages de hauteurs et de nuances) à partir desquels un son particulier est calculé par interpolation. Les sons sont par ailleurs généralement segmentés en parties (attaque, partie stable, relâche). Les parties d'attaque sont caractéristiques de nombreux timbres instrumentaux, et leur durée est donc maintenue constante alors que la lecture de la table peut "boucler" sur la partie centrale et stable du son pour atteindre la durée souhaitée (ou attendre le relâchement de la touche s'il s'agit d'un clavier).

La synthèse par échantillonnage permet donc de simuler des sons instrumentaux ou de reproduire des sons préexistants en les contrôlant "note par note". Elle ne permet donc pas, en principe, d'appréhender des notions comme les transitions entre ces notes et la "prosodie" des séquences et des sons générés. Ceci peut être remédié partiellement par l'utilisation de "diphones", c'est-à-dire d'échantillons sonores correspondant aux transitions entre paires de phonèmes (à l'origine dans les applications pour la parole) ou de notes (dans les extensions aux sons musicaux).

Méthodes concaténatives

Egalement basées sur des signaux préexistants, les méthodes de synthèse concaténatives utilisent des bases d'échantillons sonores, stockées en mémoire ou générées à partir de segmentation de signaux donnés en entrée du processus de synthèse, pour reconstituer de nouveaux signaux. Ces méthodes sont donc généralement couplées à des techniques complexes pour la segmentation, l'analyse, la sélection des signaux (voir par exemple [Schwartz, 2004]).

---

[4] MIDI (*Musical Instrument Digital Interface*) est un format standard permettant la communication de données entre les dispositifs et programmes musicaux. Une description plus détaillée se trouve dans le chapitre 7, section 7.3.

La méthode de synthèse PSOLA (*Pitch Synchronous OverLap-Add*) [Charpentier et Stella, 1986], utilisée pour l'analyse et la synthèse de la parole principalement, se base sur des techniques avancées de segmentation de signaux afin d'en resynthétiser des nouveaux, par recombinaison, transformations et réintégration des segments.[5] Cette méthode a également été étendue et utilisée pour les sons musicaux [Peeters, 1998].

1.2.2 Techniques spectrales

Analyse de Fourier

Le théorème de Fourier montre qu'une fonction périodique peut être décomposée en une série de fonctions sinusoïdales (série de Fourier) de fréquences et amplitudes variables. Celui-ci est généralisé par la transformée de Fourier (TF) qui permet d'associer un *spectre* en fréquence à une fonction intégrable quelconque. Pour une fonction $s(t)$, on a une équation de la forme :

$$TF(s) : S(f) = \int_{-\infty}^{+\infty} s(t).e^{-i2\pi ft}dt \tag{1.1}$$

Cette transformation a la propriété d'être inversible, ce qui nous donne :

$$TF^{-1}(S) : s(t) = \int_{-\infty}^{+\infty} S(f).e^{+i2\pi ft}df \tag{1.2}$$

La transformée de Fourier permet donc de passer d'une fonction définie dans le domaine temporel à une fonction (ou une distribution) dans le domaine des fréquences : le spectre. Au lieu d'observer la variation d'une valeur (pression acoustique, ou amplitude) dans le temps, celle-ci indique l'énergie moyenne répartie sur différentes fréquences. L'analyse de Fourier a ainsi des répercutions directes sur les intuitions perceptives liées au signal (perception et discrimination des hauteurs). Elle sera fondamentale dans de nombreuses applications de traitement du signal sonore, notamment dans toutes les techniques spectrales.

En pratique, un signal numérique est défini en temps discret par $s[k] = s(nT)$, où T représente la période d'échantillonnage (l'inverse de la fréquence d'échantillonnage, soit $1/F_e$). On utilise alors la transformée de Fourier discrète (TFD) et son inverse, explicitées pour un signal défini sur N échantillons par les formules suivantes :

$$TFD(s) : S[k] = \sum_{n=0}^{N-1} s[n].e^{-i2\pi k\frac{n}{N}} \tag{1.3}$$

[5] La technique d'*overlap-add* consiste à réaliser des segmentations partiellement superposées afin d'adoucir les transitions entre les segments lors de la reconstitution (synthèse) des signaux, par moyennage sur les parties tuilées.

$$TFD^{-1}(S) : s[n] = \frac{1}{N} \sum_{n=0}^{N-1} S[k].e^{i2\pi n \frac{k}{N}} \tag{1.4}$$

Des algorithmes permettent de calculer cette transformée discrète et son inverse avec une complexité relativement faible (*Fast Fourier Transform* – FFT, et *Inverse Fast Fourier Transform* – IFFT).

La figure 1.4 montre l'exemple d'un signal pseudo périodique et de sa transformée de Fourier. Sur la transformée de Fourier, on observe un pic principal correspondant à la fréquence fondamentale F_0 (périodicité du signal) que l'on retrouve sur le signal, ainsi qu'une série de pics secondaires correspondant aux harmoniques de celle-ci ($F_0 \times 2$, $F_0 \times 3$, etc.). Dans un signal périodique, en effet, les fréquences des composantes sinusoïdales sont liées harmoniquement (i.e. elles sont les multiples d'une même fréquence fondamentale). Le troisième graphique est une représentation logarithmique du spectre (en dB), généralement utilisée pour son adéquation aux caractéristiques de la perception sonore.[6]

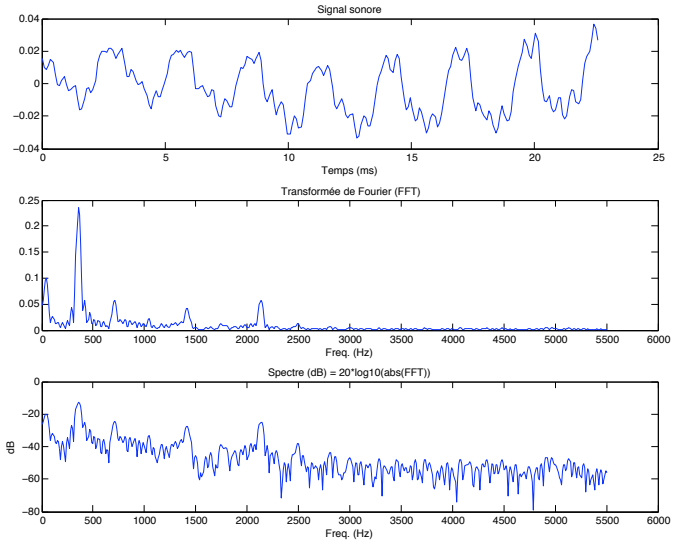


FIGURE 1.4: Transformée de Fourier d'un signal pseudo périodique. De haut en bas : signal (amplitude/temps), spectre (amplitude/fréquence), et spectre logarithmique (amplitude dB/fréquence).

[6]La sensibilité de l'oreille humaine est liée de façon proportionnelle au logarithme de l'amplitude des signaux sonores.

Transformée de Fourier à court terme

La transformée de Fourier constitue une analyse étendue sur une durée, perceptivement pertinente uniquement pour des signaux stationnaires sur celle-ci (c'est-à-dire dont la répartition énergétique par fréquence reste constante).

Pour calculer le spectre d'une fonction non stationnaire (qui est le cas général pour un signal sonore), on utilise la transformée de Fourier à court terme (TFCT, ou STFT pour *short-time Fourier transform*) [Allen et Rabiner, 1977]. Cette technique, schématisée sur la figure 1.5, consiste à multiplier le signal par une fenêtre d'analyse localisée sur un petit nombre d'échantillons (généralement entre 250 et 4000 environ, pour des signaux échantillonnés à 44100 Hz) sur lesquels il est supposé stationnaire.

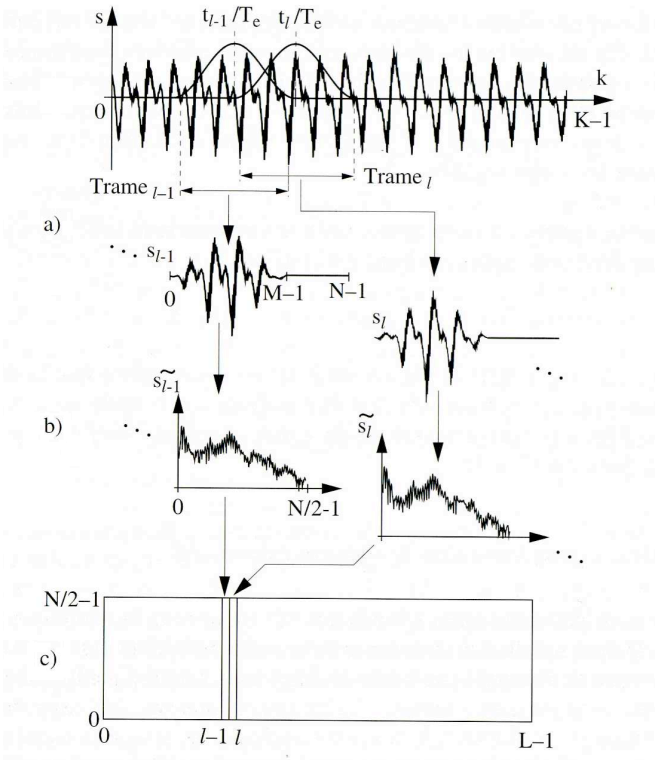


FIGURE 1.5: Schéma de principe de la Transformée de Fourier à Court Terme. a) Multiplication du signal par une fenêtre temporelle glissant successivement sur les différentes trames ; b) calcul du spectre à court terme par transformée de Fourier sur chacun des segments (trames) isolés ; c) concaténation des segments pour la constitution du spectrogramme sur la durée du signal.[7]

Sur cette courte fenêtre localisée dans le temps, l'analyse par TFD donne une description du signal dans le domaine fréquentiel (appelée spectre à court terme). Les fenêtres utilisées sont en général en forme de "cloche" (fenêtres de Hanning, Hamming, Blackman,

[7]Figure extraite de [Doval, 2004].

etc.) afin d'éviter, lors du calcul de la transformée de Fourier, les distorsions créées par la variation brutale qui serait causée par une fenêtre rectangulaire. La fenêtre est ensuite décalée (généralement par *overlapping*, c'est-à-dire d'une demi-fenêtre au plus, afin d'éliminer les pertes dues au fenêtrage) et l'analyse est ainsi effectuée successivement sur l'ensemble des segments du signal pour calculer une suite de spectres à court terme. Cette technique d'analyse du signal est détaillée par exemple dans [Doval, 2004].

On obtient donc par cette technique une information tridimensionnelle (amplitude × fréquence × temps) représentant l'évolution dans le temps du spectre du signal. Cette information peut être visualisée sous forme d'un diagramme 3D ou d'un spectrogramme (plan temps/fréquence sur lequel l'énergie du signal est indiquée par l'intensité de couleur, le plus souvent sur une échelle de niveaux de gris – également appelé sonagramme). Un exemple de spectrogramme est donné sur la figure 1.6.

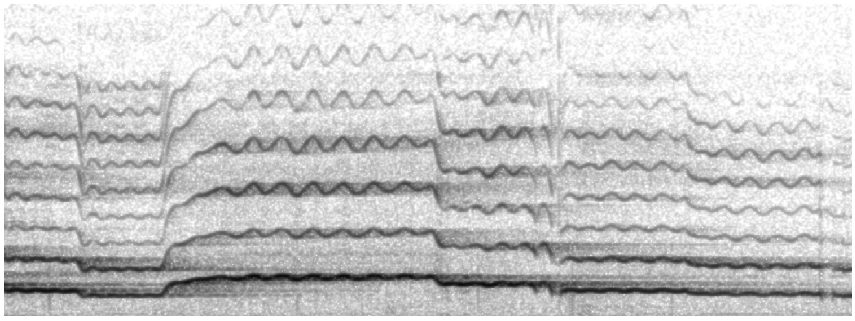


FIGURE 1.6: Représentation d'un signal sonore analysé par la TFCT (spectrogramme).

Une remarque importante à soulever à ce stade concernant l'analyse de Fourier est l'existence d'un principe d'incertitude, bien connu des physiciens, selon lequel les précisions de l'analyse en temps et en fréquence s'obtiennent forcement l'une au détriment de l'autre.

Rappelons d'abord que, d'après le théorème de l'échantillonnage, le taux d'échantillonnage d'un signal doit être deux fois supérieur à la composante de fréquence maximale contenue dans ce signal, sans quoi on observe une distorsion (appelée "repliement spectral") dans le signal échantillonné. En d'autres termes, pour être échantillonné avec une fréquence d'échantillonnage donnée Fe, un signal ne doit pas contenir de fréquences supérieures à $Fe/2$. On appelle ce seuil la fréquence de Nyquist. Afin de ne pas la dépasser, on filtre généralement les hautes fréquences d'un signal avant d'effectuer un échantillonnage.[8] Les fréquences présentes dans la TFD sont donc elles aussi comprises entre 0 et $Fe/2$. Par ailleurs, on peut montrer que le calcul d'une transformée de Fourier discrète sur un signal de longueur N permet d'obtenir un spectre de $N/2$ points (bandes de fréquences, appelées *frequency bins*). Une augmentation de la fréquence d'échantillonnage du signal permet donc d'élargir la plage des fréquences analysées, mais pas la précision de l'analyse.

[8]La norme de 44100 Hz pour les formats audio numériques se justifie ainsi comme correspondant au double de la fréquence maximale perçue par l'oreille humaine (aux alentours de 22000 Hz).

Il faut, pour obtenir une bonne précision en fréquence dans la TFD, calculer celle-ci sur un nombre suffisant de points (i.e. sur un segment sonore plus long).[9] Cependant de larges fenêtres ne permettront qu'une reconstruction approximative de la structure temporelle fine du signal. Pour obtenir une description avec une grande finesse temporelle, il faut au contraire découper le signal en segments très courts lors de l'analyse par TFCT.[10]

Ce problème de compromis entre précision temporelle et fréquentielle est traité par les techniques d'analyse multirésolution qui ont fait leur apparition par la suite, dont notamment la transformée en ondelettes [Kronland-Martinet *et al.*, 1987] [Kronland-Martinet et Grossman, 1991], qui fait également l'objet d'expériences dans des applications pour l'analyse et la synthèse sonores [Arfib et Kronland-Martinet, 1993].

A partir de cette analyse par TFCT, la synthèse consiste à effectuer une transformée de Fourier inverse sur chacune des fenêtres et à reconstituer le signal par *overlap-add*. C'est une méthode de synthèse assez efficace puisque la FFT et son inverse permettent d'effectuer ces opérations avec un coût de calcul relativement faible, et invariant selon la complexité du spectre. Différentes améliorations permettent en outre d'optimiser le calcul et le paramétrage de la synthèse par FFT inverse (voir par exemple [Rodet et Depalle, 1992]).

Cependant, la construction d'un spectrogramme complet dans une éventuelle démarche musicale reste une tâche complexe pratiquement impossible à réaliser de façon intrinsèque. On utilisera donc généralement cette technique pour la transformation de sons préexistants par analyse/resynthèse, ou en utilisant des données préalablement structurées grâce aux techniques additives.

Synthèse additive

Les modèles d'analyse/synthèse additifs découlent de la théorie de Fourier, considérant le son comme une somme de sinusoïdes élémentaires, mais privilégient une vision orthogonale à celle de la TFCT : le son est décomposé dans le domaine fréquentiel en premier lieu. A chaque élément de cette décomposition correspond un signal pseudo sinusoïdal d'amplitude et de fréquence variables dans le temps, appelé *partiel*. Un partiel représente donc l'une de ces composantes sinusoïdales (ou périodicités) contenues dans un signal. On suppose donc que celle-ci soit maintenue sur une certaine durée (on lui attribue en général un début et une fin, selon un seuil d'intensité minimale), et suivant des évolutions d'amplitude et de fréquence relativement lentes dans le temps.

---

[9] La technique du *zero-padding* (ajout de valeurs nulles au signal initial pour augmenter sa taille) permet d'augmenter le nombre de points du spectre (une sorte de sur-échantillonnage en fréquence) mais ne change pas la précision de l'analyse.

[10] En utilisant des grandes fenêtres décalées par un petit nombre d'échantillons les unes par rapport aux autres, on obtient un nombre plus important de trames lors de la TFCT, mais constituées des composantes moyennées par superposition sur les intervalles, donc finalement sans gain de précision temporelle.

Considérant un signal sonore donné, on peut donc envisager un dispositif d'analyse additive comme une banque de n filtres identiques et répartis entre 0Hz et la moitié de la fréquence d'échantillonnage du signal initial – fréquence de Nyquist. Les bandes passantes de ces filtres doivent être suffisamment étroites (c'est-à-dire qu'ils doivent être suffisamment nombreux) pour que leur sortie soit approximativement sinusoïdale (c'est-à-dire qu'ils ne contiennent qu'une composante périodique du signal chacun). Il n'y a alors en principe pas de perte d'information entre les données d'analyse et le signal numérique d'origine, qui peut être reconstitué à l'identique. Le vocoder de phase (*phase vocoder* [Portnoff, 1980] [Dolson, 1986]) est un dispositif permettant d'obtenir ce type de représentation.

Cette méthode est mathématiquement équivalente à l'analyse du signal par TFCT. Elle présente cependant l'avantage de fournir une information plus structurée, dans le sens ou les partiels peuvent être considérés et traités individuellement (modifications locales, filtrages, etc.) En considérant seulement les partiels les plus importants d'un signal sonore, on peut obtenir une description allégée et essentielle, par rapport à ce que représente un spectrogramme complet.

La TFCT, en revanche, présente l'avantage d'une plus grande généralité, car elle ne présuppose pas de la régularité des trajectoires des partiels du signal. Des méthodes de détection et suivi de partiels sont applicables aussi sur cette représentation, permettant d'obtenir une description structurée de sa constitution harmonique. Ainsi, la méthode d'analyse additive (suivi de partiels, ou *partial tracking*) la plus couramment utilisée, introduite dans [McAulay et Quatieri, 1986], consiste en une détection des maximums locaux dans les spectres instantanés calculés par FFT et en la sélection et le suivi des principaux d'entre eux. On obtient par cette méthode un ensemble de partiels, positionnés sur le plan temps/fréquence et caractérisés par l'évolution de leurs fréquences, amplitudes et phases.

Sur la figure 1.6, on observe clairement en foncé les partiels principaux du signal sonore analysé, qui constituent sa partie harmonique.

La synthèse additive consiste en la mise en œuvre simultanée d'un certain nombre d'oscillateurs correspondant aux différents partiels. Les paramètres de contrôle d'un oscillateur (amplitude et fréquence) peuvent être spécifiés par des valeurs numériques constantes ou variables (sous forme d'enveloppes, ou issues d'autres processus générateurs de signaux). A partir d'un ensemble d'oscillateurs sur lesquels on contrôle individuellement ces valeurs et évolutions de fréquence et d'amplitude, on est donc en mesure de recréer des signaux sonores complexes (figure 1.7).

Cette méthode est l'une des techniques de synthèse les plus utilisées car elle est intuitive musicalement : les paramètres temps, hauteur, intensité sont les mêmes que ceux utilisés traditionnellement pour la composition. Leur représentation (mentale et graphique) sur le "plan" temps/fréquence facilite leur manipulation : on pourra jouer sur les rapports de temps, de fréquences entre les partiels, travailler sur l'harmonicité et l'inharmonicité [Mathews et Pierce, 1980], etc.

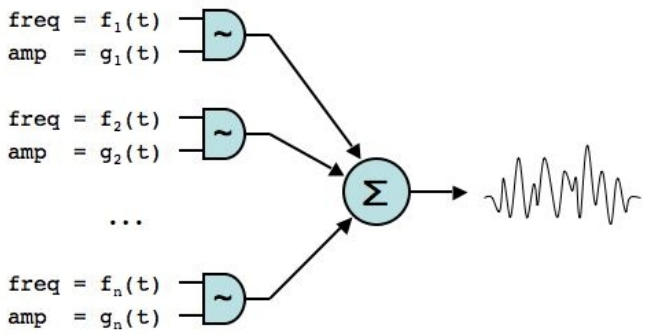


FIGURE 1.7: Schéma d'une synthèse additive à partir d'un ensemble d'oscillateur.

En revanche, elle se montre coûteuse lorsqu'un grand nombre d'oscillateurs doivent être activés en même temps, ce qui est souvent le cas lorsque l'on souhaite construire des sons riches (un simple son de piano peut être constitué de plus d'une centaine de partiels).[11] Une technique utilisée pour remédier à cela est de reconstituer à chaque instant un spectre instantané à partir des valeurs des différents partiels sur lequel on calcule une FFT inverse selon le procédé évoqué plus haut avec la TFCT.

Le modèle additif, dans ses différentes variantes, permet donc une décorrélation des informations temporelles et spectrales du signal. Il s'agit donc d'un mode de représentation particulièrement utile pour réaliser des mesures et transformations indépendamment sur ces deux composantes. Ces informations étant structurellement découplées, de nombreuses opérations peuvent être effectuées pour le traitement des sons : filtrages, mais aussi étirement ou compression temporelle (*time stretching*), transpositions ou *pitch shifting*, etc. [Dolson, 1989].

L'absence de modélisation des composantes non périodiques du signal sonore (c'est-à-dire le bruit, mais également les transitoires d'attaque, les frottements, et autres phénomènes non représentés par les partiels) est un inconvénient de la synthèse additive, principalement quand il s'agit de reproduire ou transformer des sons "naturels". La technique présentée dans [Serra et Smith, 1990] sous le nom de *Spectral Modeling Synthesis*, consiste à isoler la partie bruitée d'un signal analysé en soustrayant au signal original le signal issu de la resynthèse des partiels. Cette partie "non périodique" est alors traitée séparément de la partie modélisée par les partiels, les deux étant finalement réunies (additionnées) lors de la synthèse.

[11] Les capacités des processeurs de calcul actuels tendent à minimiser cette restriction.

Modèle source/filtre

Lorsque l'on analyse un signal sonore par la transformée de Fourier, on observe souvent une forme générale du spectre modulant les amplitudes des différents pics spectraux. C'est ce que l'on appelle l'*enveloppe spectrale* du signal. Cette enveloppe est caractérisée par une évolution "lente" en fonction de la fréquence. Notre oreille y est particulièrement sensible dans la perception et l'identification de certains timbres sonores, alors que la structure fine du spectre, caractérisée par les pics des différents partiels, porte l'information harmonique (hauteur) ou bruitée du son.

Dans la plupart des sons instrumentaux et naturels, ces deux composantes d'un signal se rapportent respectivement à la résonance et à la source d'un système acoustique. D'un point de vue physique, la source correspond à une vibration et la résonance à un milieu de propagation agissant sur la source comme un filtre, dont la réponse atténue ou amplifie plus ou moins certaines fréquences.[12]

La figure 1.8 reprend le spectre instantané mesuré dans l'exemple de la figure 1.4 sur lequel est superposé une estimation de cette enveloppe spectrale.

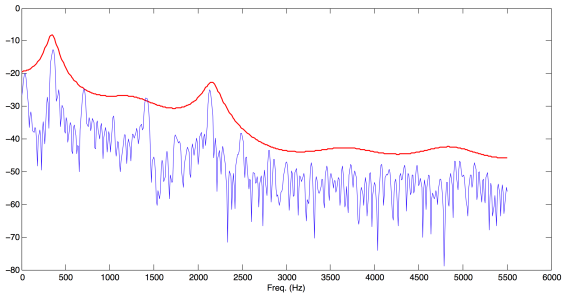


FIGURE 1.8: Spectre et enveloppe spectrale.

L'enveloppe spectrale d'un signal peut être calculée de différentes manières, dont les plus courantes sont la prédiction linéaire (LPC) [Makhoul, 1975] ou le calcul du *cepstre* (transformée de Fourier du spectre logarithmique). Les maximums locaux de cette enveloppe correspondent à des *formants*. Un formant est décrit principalement par une fréquence centrale, une amplitude, et une largeur de bande à mi-hauteur caractérisant sa forme plus ou moins étroite. En règle générale, un petit nombre de formants suffit à décrire correctement les parties basses fréquences d'une enveloppe spectrale. Un des avantages du modèle source/filtre vient donc du fait que le son y est représenté avec une quantité raisonnable de données (descriptions sommaires de la source et de l'enveloppe spectrale).

[12] De part leur forte corrélation avec les caractéristiques physiques des systèmes résonants, et même si leur implémentation relève de techniques spectrales, les modèles source/filtre sont également parfois répertoriés en tant que modèles physiques (voir paragraphe 1.2.4).

La synthèse source/filtre est donc mise en œuvre par l'application d'un filtrage linéaire (correspondant à l'enveloppe spectrale) sur un signal d'excitation. Cette méthode a surtout été utilisée pour la synthèse de la parole avant d'être étendue aux sons musicaux en général [Rodet *et al.*, 1984]. Le fonctionnement réel de production sonore de la voix est en effet assez bien reproduit par le modèle *source/filtre* (c'est-à-dire considéré comme constitué d'une source d'excitation et d'un filtre linéaire, supposés non couplés).[13]

L'enveloppe spectrale peut également être utilisée lors des transformations appliquées sur les spectrogrammes, afin de préserver des caractéristiques timbrales lors des modifications des fréquences des partiels, par exemple, ainsi que pour réaliser des synthèses croisées entre différents sons (moduler le spectre d'un son par l'enveloppe spectrale d'un autre).

La synthèse dite *soustractive*, utilisée principalement à l'époque des synthétiseurs analogiques, consiste à utiliser un son au timbre plutôt complexe (par exemple un bruit) et de le soumettre à des filtres passe-bande afin de remodeler ce timbre (par élimination ou atténuation de certaines bandes de fréquence), et constitue donc également une forme de synthèse source/filtre.

Fonctions d'Ondes Formantiques

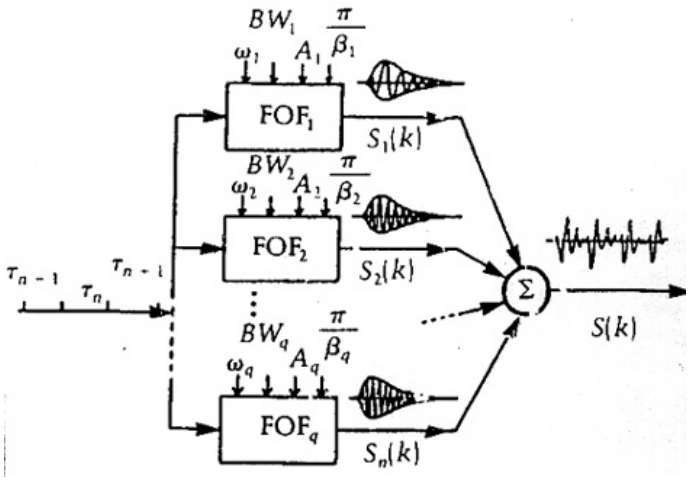


FIGURE 1.9: Synthèse par Fonctions d'Ondes Formantiques.[14]

La synthèse par Fonctions d'Ondes Formantiques (FOF) est un dérivé du modèle source/filtre consistant à remplacer les filtres numériques par des fonctions correspondant directement aux formants du signal, c'est-à-dire les résultantes d'impulsions ou de

[13]On distingue dans le signal de la voix la partie modulante que constitue cette enveloppe spectrale, qui caractérise le conduit résonnant de l'appareil vocal en tant que filtre, et permet d'identifier par exemple une voyelle prononcée, ou d'autres caractéristiques du timbre. La source d'excitation quant à elle provient de l'air expulsé et des impulsions des cordes vocales (dans le cas de sons dits "voisés").

[14]Figure extraite de [Rodet *et al.*, 1985], avec l'aimable autorisation des auteurs.

signaux traversant les différents filtres (voir figure 1.9). Chaque FOF modélise ainsi un signal dont le spectre constitue directement l'une des composantes formantiques.

Une fonction d'onde formantique est décrite pour l'essentiel par une équation de la forme :

$$s_i(k) = G_i e^{-\alpha_i k} sin(\omega_i k + \Phi_i) \tag{1.5}$$

Les paramètres de cette équation permettent de spécifier notamment une fréquence centrale, une largeur de bande, et une amplitude, le tout déterminant les caractéristiques et la forme d'un formant. Le signal est ensuite reconstitué par addition des différentes FOF :

$$h(k) = \sum_{i=1}^{n} s_i(k) \tag{1.6}$$

Cette méthode permet d'optimiser les calculs et de générer des enveloppes spectrales qui nécessiteraient l'utilisation de bancs de filtres beaucoup plus complexes.

Dans [Rodet *et al.*, 1984] sont détaillés les principes de la synthèse par FOF, et comparés à la méthode par filtres numériques.

Modèles résonants

Les modèles de résonances constituent une autre approche de type source/filtre. Des sons sont analysés afin d'extraire leurs caractéristiques de résonance, modélisées par un ensemble de filtres, généralement caractérisés par des enveloppes d'amplitude de pente décroissante variable. Dans une phase de synthèse, ces filtres sont appliqués à des sons, naturels ou synthétiques, afin de créer des timbres plutôt percussifs [Barrière *et al.*, 1985].

L'algorithme de Karplus-Strong est une méthode également basée sur les résonances pour la modélisation de sons de type cordes pincées ou percussifs [Karplus et Strong, 1983] [Jaffe et Smith, 1983]. Il s'agit d'un processus simple constitué d'une boucle récursive et d'un filtre passe-bas permettant, à partir d'une impulsion donnée (bruit, ou échantillon sonore), la génération d'un son dont la fréquence fondamentale est contrôlée et dont l'atténuation des partiels est d'autant plus importante que la fréquence est élevée. Cette méthode est très utilisée car elle permet la création de sons complexes aux timbres riches pour un faible coût de calcul.

Synthèse granulaire

En 1946, Dennis Gabor énonçait une théorie centrée sur l'idée d'une représentation des signaux composée de particules élémentaires (*quanta*) [Gabor, 1946]. Dans son application au signal sonore [Gabor, 1947], cette théorie, appuyée par des expériences perceptives, a permis d'approcher la notion d'éléments sonores élémentaires (que l'on appellera "grains"),

répartis dans l'espace temps / fréquence. La théorie de Gabor annonçait ainsi une nouvelle approche conceptuelle pour la synthèse sonore, basée sur l'organisation et la répartition de ces grains pour la création de sons complexes. Il s'agit en fait, au lieu de décomposer le signal en segments afin d'en faire une analyse en fréquence (approche "classique" de Fourier), de considérer de multiples occurrences d'une onde élémentaire, transposée et dilatée sur le plan temps / fréquence.

Iannis Xenakis fut le premier compositeur à s'emparer musicalement de cette idée, en concevant des structures sonores et musicales à partir de modèles de répartition (nuages) de grains sonores définis par leur durée, fréquence, et intensité [Xenakis, 1981]. Des implémentations logicielles ont été réalisées quelques années plus tard [Roads, 1985] [Truax, 1988].

La mise en œuvre d'une synthèse granulaire consiste en pratique dans un premier temps à choisir le grain que l'on va utiliser : une petite forme d'onde pouvant être par exemple sinusoïdale, aléatoire, ou encore issue d'un extrait sonore, ainsi qu'une enveloppe d'amplitude qui lui est appliquée (souvent de type gaussienne).[15] Dans un deuxième temps, il s'agit d'organiser les grains ainsi définis dans l'espace temps / fréquence (nombre, densité, répartition, etc.) Plusieurs centaines, voire milliers de grains sont à paramétrer pour quelques secondes de son ; on utilisera donc généralement pour cela des processus, probabilistes par exemple, permettant le contrôle d'amas granulaires importants.

Curtis Roads a publié différents textes et ouvrages sur la synthèse granulaire (voir par exemple [Roads, 1991], [Roads, 2002]), que l'on pourra consulter pour une description approfondie de cette technique.

1.2.3 Modèles abstraits ou non linéaires

On appelle modèles *abstraits* les méthodes de synthèse par lesquelles le signal est créé directement à partir de formulations mathématiques. On qualifie également ce type de technique de *non linéaire*, dans le sens où il n'existe pas de relation linéaire entre le nombre et les variations des paramètres donnés en entrée, et les caractéristiques du signal obtenu en sortie. [Risset, 1993] emploie le terme de synthèse *globale*.

La synthèse FM

L'exemple le plus célèbre est celui de la synthèse par modulation de fréquence (ou FM), inventée par John Chowning [Chowning, 1973]. Le principe de la synthèse FM, utilisé auparavant pour la transmission de signaux analogiques (pour les ondes radio, par exemple), consiste à faire varier la fréquence d'une onde *porteuse* en fonction d'une deuxième onde dite *modulante*. Un signal $X(n)$ est donc le produit de l'équation :

$$X(n) = A(n)sin(2\pi f_p nT + I(n)sin(2\pi f_m nT)) \tag{1.7}$$

[15] La taille de ce grain, généralement d'une durée inférieure à 50 ms, est proche de la durée minimale d'un signal permettant sa perception en tant que son (perception d'une hauteur, par exemple).

Ce signal a la propriété d'être très riche spectralement, c'est-à-dire composé de très nombreux partiels. La répartition spectrale des partiels est déterminée par le rapport entre la fréquence de l'onde porteuse (f_p) et celle de l'onde modulante (f_m) : des rapports rationnels produiront des sons harmoniques, alors que des rapports plus complexes (non rationnels) produiront des sons inharmoniques. On pourra ainsi jouer sur ces rapports pour modifier les degrés d'inharmonicité. L'énergie du spectre, centrée autour de la fréquence porteuse, est également répartie dans les partiels qui l'entourent en fonction de l'indice de modulation $I(n)$, ce qui offre un contrôle supplémentaire sur le timbre. Un grand éventail de timbres différents peuvent ainsi être obtenus par synthèse FM, pour un coût de calcul très faible et un petit nombre de paramètres de contrôle.[16]

On trouvera dans [Roads et Strawn, 1985] une réédition de l'article original de Chowning ainsi qu'une compilation d'autres textes décrivant des améliorations ou applications de la synthèse FM.

Autres techniques non linéaires

Suite à ces expériences, d'autres techniques mathématiques ont été explorées pour la synthèse de signaux sonores, mais les résultats sont restés jusqu'ici assez marginaux.

On pourra citer toutefois la technique de synthèse par distorsion non linéaire, également appelée *waveshaping synthesis* [Arfib, 1979] [LeBrun, 1979] [Roads, 1979]. Dans ce type de synthèse, un signal initial est traité par un système à fonction de transfert non linéaire, ce qui permet de produire une grande variété de timbres, dépendant des variations des quelques paramètres de cette fonction de transfert.

L'un des inconvénients majeurs des techniques de synthèse non linéaires, y compris de la synthèse FM, est qu'elles ne sont pas réversibles par un procédé d'analyse : il n'existe pas de moyen, par exemple, d'extraire les paramètres de l'équation de modulation de fréquence à partir d'un son donné. On ne sait donc pas non plus exactement corréler les différents paramètres de ces équations à des conséquences directement perceptibles sur le signal produit, sinon par l'expérimentation empirique, ce qui est un problème au niveau des applications musicales :

> "Il est difficile de trouver une grande variété de sons musicalement plaisants par exploration des paramètres d'une expression mathématique." [Smith, 1993]

[16] Cette synthèse a été utilisée et popularisée par les synthétiseurs Yamaha de la série DX, dont le fameux DX7.

1.2.4 Modèles physiques

La synthèse par modèles physiques s'attache aux causes des phénomènes (sonores, en l'occurrence) plutôt qu'à leurs effets [Cadoz, 1979] [Florens et Cadoz, 1991]. Sous cette définition, il est possible d'en rapprocher certaines techniques que nous avons vues précédemment, comme le modèle source/filtre ou les modèles résonants. Ces techniques sont en effet à mi-chemin entre modélisation physique et modélisation spectrale du signal.

S'il s'agit encore généralement des principes d'excitateurs et de résonateurs, cependant, on parlera de modèles physiques pour les méthodes consistant en la simulation de processus vibratoires produisant des signaux par la constitution d'objets physiques virtuels.

Dans la synthèse par modèles physiques, on définit donc généralement en premier lieu un système mécanique, une structure physique. Sur celle-ci sont effectuées des stimulations mécaniques (percussion, frottement, ou autres forces appliqués en des points donnés du système) et déterminés des points d'écoutes, sur lesquels on cherche à mesurer les ondes acoustiques produites par ces stimulations et transformées par propagation dans le système.[17] Un système d'équations établi pour le calcul de l'équation d'onde résultant de ces différents paramètres doit alors être résolu. Cette résolution doit généralement se faire itérativement par approximations successives de l'onde numérique (une solution mathématique explicite étant en général impossible à déterminer directement), ce qui peut représenter une charge de calcul importante.[18]

Les principales approches utilisées sont les systèmes de masses–interactions et la synthèse modale.

L'approche *masses–interaction* consiste en la définition d'objets "vibrants" à partir d'entités physiques élémentaires décrites par des paramètres tels que leurs dimensions, leur masse, leur élasticité. On peut en effet décrire (ou approximer) le comportement vibratoire de tout système physique par un ensemble de masses ponctuelles connectées les unes aux autres par des liaisons rigides, élastiques, ou autres. Les premiers travaux sur la synthèse par modèles physiques, menés par Lejaren Hiller et Pierre Ruiz [Hiller et Ruiz, 1971] utilisaient déjà ce principe pour la modélisation de cordes vibrantes, avec une série de masses liées entre elles par des ressorts. L'application d'une force en un point de cette structure se propage ainsi selon les attributs physiques de ses composants élémentaires (masses et élasticité des ressorts), simulant le principe de la propagation des ondes sur une ou plusieurs dimensions.

La *synthèse modale* [Adrien, 1991] se base sur des descriptions d'objets à plus grande échelle, partant du postulat que tout objet est composé de sous-structures, correspondant par exemple aux différentes parties d'un instrument. Soumises à une excitation, ces parties

[17] Des paramètres additionnels sont également nécessaires à la résolution ultérieure des équations d'ondes, tels que l'état initial, les conditions limites, le comportement transitoire, etc.

[18] Ce fut pendant longtemps l'un des principaux freins à l'utilisation de la synthèse par modèles physiques.

répondent selon ses modes de vibrations propres. Les sous-structures sont ainsi déterminées par un certain nombre de données caractérisant les fréquences, amortissements, etc. de ces résonances propres. Ces caractéristiques sont elles-mêmes déterminées par le calcul ou par l'expérimentation sur des objets réels, constituant une base de données de sous structures élémentaires pouvant être assemblées pour créer des systèmes complexes. Il s'agit donc d'une méthode d'un niveau d'abstraction plus élevé que celui des masses–interactions, la modélisation se situant au niveau d'objets à l'échelle "humaine".

D'autres méthodes entrent aussi dans le cadre des modèles physiques (par exemple la synthèse par guide d'ondes [Smith, 1992]). [Roads, 1993] donne un aperçu de ces différents types de synthèse par modèles physiques.

La simple imitation d'instruments existants reste évidemment d'un intérêt limité pour la musique ; en revanche la synthèse par modèles physiques permet de créer des structures mécaniques irréalistes (taille, densité des matériaux), ou hybrides (mélangeant des propriétés de différents instruments), ou même évoluant dans le temps. Les sons issus de modèles physique sont par ailleurs généralement caractérisés par leur côté naturel (même lorsqu'ils sont improbables naturellement). L'intuition et la culture auditive permettent ainsi d'associer plus facilement les variations des paramètres d'entrée des processus de synthèse avec leurs conséquences dans le domaine sonore. Ces paramètres sont assimilables à ce que pourraient être des gestes instrumentaux sur des instruments réels.

1.3 Conclusion

Nous avons présenté dans ce chapitre un aperçu des principales techniques de synthèse sonore. Des informations complémentaires pourront être trouvées par exemple dans [Roads, 1996], ouvrage de référence en la matière, ou encore dans [De Poli *et al.*, 1991], [Moorer, 1977], [Battier, 1988], et de nombreux autres ouvrages dédiés au sujet.

Ces différentes techniques sont pour la plupart implémentées sous forme de programmes (*systèmes de synthèse*) permettant de synthétiser des sons à partir d'un ensemble de paramètres (ensemble de partiels, paramètres de FOF ou d'autres équations mathématiques, signaux préexistants et données de traitement ou de filtrage, etc.) Ces paramètres forment alors une représentation particulière du son, dépendante et donnée par la technique de synthèse utilisée.

Avec le développement d'outils et d'environnements logiciels multi-paradigmes, le cloisonnement entre les techniques que nous avons énoncées a tendance à s'estomper, et celles-ci s'utilisent couramment en complémentarité les unes avec les autres. Si nous irons progressivement dans le sens d'une abstraction de ces considérations pour aborder des problématiques musicales de plus haut niveau, la connaissance des techniques sous-jacentes n'en restera cependant pas moins une nécessité pour un contrôle approfondi des sons et processus de synthèse.

Chapitre 2

Systèmes de synthèse sonore et de contrôle de la synthèse

Ce chapitre propose un aperçu des travaux et systèmes informatiques visant à la mise en œuvre des techniques de synthèse vues dans le chapitre précédent dans un cadre musical.

Trois types de systèmes sont distingués : les *systèmes de synthèse*, qui implémentent des processus correspondant à des techniques de synthèse particulières, les *langages de synthèse*, qui permettent à l'utilisateur de créer lui-même ses processus de synthèse, et les environnements de *contrôle de la synthèse*, qui permettent de paramétrer ces processus à l'aide de représentations de haut niveau.

A travers les exemples présentés, nous souhaitons situer notre travail dans le contexte des outils existants et identifier les problématiques et enjeux liés à la composition musicale électroacoustique.

2.1 Synthèse sonore et représentations du son

[Loy, 1989] effectuait à la fin des années 1980 une revue des systèmes de composition et de synthèse de l'époque. Parmi les environnements de synthèse, il distinguait déjà les "compilateurs d'échantillons" et les programmes de paramétrage et de contrôle de la synthèse. Le contexte technologique ayant évolué depuis, les outils qui sont cités dans ce texte sont tombés pour beaucoup en désuétude ; cependant les différents types de systèmes, tout comme les problématiques évoquées dans ce texte et dans la bibliographie de l'époque, sont pour la plupart toujours d'actualité.

Nous nous intéresserons ici dans un premier temps (section 2.2) au premier type d'environnements ("compilateurs d'échantillons"), permettant d'engendrer des signaux sonores à partir des techniques de synthèse décrites dans le chapitre précédent. Nous les appellerons *systèmes de synthèse*.

Nous avons mentionné précédemment le fait qu'une technique de synthèse donnée était associée à une *représentation* correspondante et particulière du son (un ensemble de

partiels, une fonction mathématique, un signal source et un ensemble de filtres, ou une enveloppe spectrale, etc.) L'ensemble des paramètres d'un système de synthèse constituera ainsi une représentation du son spécifique et implicitement définie par la technique de synthèse adoptée. Cette représentation constitue la forme concrète que prend un son selon cette technique.[1]

> "L'information [véhiculée par un son par l'intermédiaire d'un modèle de synthèse] est représentée par un ensemble de valeurs évoluant dans le temps, appelées paramètres, qui identifient le son dans le modèle choisi." [Depalle et Rodet, 1993]

La représentation suggère ou détermine également l'ensemble des opérations possibles sur le son avec une technique de synthèse. Elle permet à l'utilisateur d'un système de se construire une image mentale de son travail. Pour [Risset, 1991], la représentation est ce qui rend une idée présente, *visible aux yeux ou a l'esprit*. Elle détermine le potentiel (musical) d'un système.

> "Cette représentation conditionne la manipulation du matériau sonore dans sa composition, car elle seule permet d'en exploiter les différents aspects et les soumettre aux procédures de composition." [Eckel, 1993]

Une technique de synthèse définit ainsi un espace multidimensionnel qu'il s'agira d'explorer : le processus de composition doit alors faire apparaître un objet sonore concret par la construction de la représentation associée.

> "Une fois le dispositif de synthèse réalisé, la production d'un son consiste à construire sa représentation dans le modèle." [Depalle et Rodet, 1993]

Cependant, il est souvent impossible pour un utilisateur de déterminer, suivre et contrôler l'évolution de tous les paramètres nécessaires à un contrôle total d'un système de synthèse. Les représentations découlant directement des techniques de synthèse et des systèmes correspondants sont généralement peu adaptées à la composition musicale : de par la complexité des données et des processus mis en jeu, celles-ci requièrent une grande capacité de mémoire et de calcul, accessibles par la machine mais rarement par la pensée musicale. Le choix de représentations plus abstraites devient un passage obligatoire : la réduction imposée par ces représentations permettra de structurer l'espace sonore initialement accessible par la synthèse.

> "Pour le compositeur, il importe que toute représentation établisse une distinction entre les dimensions essentielles et contingentes du monde représenté, pour n'en modéliser que les aspects les plus saillants." M. Stroppa, [Cohen-Levinas, 1993].

[1] *Représenter* X *par* Y*, c'est considérer la présence de* Y *comme substitut valable à la présence de* X*, relativement à une certaine situation* [Berthier, 2002].

Deux stratégies, éventuellement complémentaires, permettent ainsi d'aborder la synthèse d'un point de vue musical : la création de systèmes de synthèse personnalisés, définissant implicitement les représentations correspondantes, et la construction de représentations établies par un système de plus haut niveau, ce dernier cas se rapprochant du deuxième type d'environnements décrits par [Loy, 1989] et mentionnés précédemment. Respectivement, nous parlerons de *langages de synthèse* et d'environnements de *contrôle de la synthèse*. Nous donnons successivement quelques exemples dans les sections 2.3 et 2.4.

2.2 Systèmes de synthèse

Il est relativement abordable à l'heure actuelle pour n'importe quel logiciel d'implémenter des modules simples de génération et traitement de signaux numériques, et par conséquent de réaliser des synthèses additives, modulations et filtrages de toutes sortes. Il existe ainsi une grande diversité de systèmes de synthèse, dont une liste exhaustive serait difficile à établir. Nous en citons ici à titre d'exemple quelques-uns, correspondant à différentes "stratégies de contrôle", et auxquels il sera fait référence dans la suite de ce chapitre ou dans les suivants.

SuperVP [Depalle et Poirot, 1991] est une implémentation du vocoder de phase basée sur l'analyse de Fourier à court terme, à partir de laquelle sont applicables toute une série de traitements sur les signaux sonores (filtrages, étirement/compression temporelle sans transposition – *time stretching*, transposition sans modification temporelle, etc.) ainsi que d'autres types d'analyses comme l'estimation et le suivi de la fréquence fondamentale, la détection de transitoires, etc. La figure 2.1 illustre le schéma de contrôle de ce système.

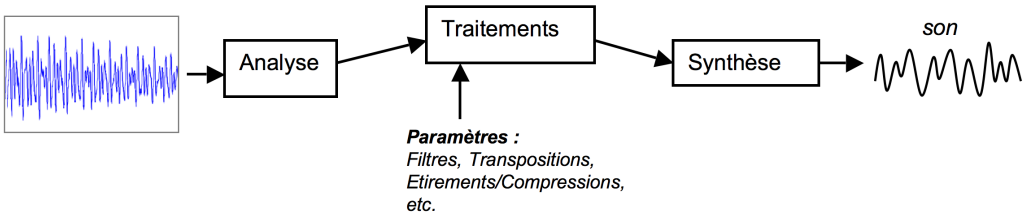


Figure 2.1: Schéma de contrôle du système SuperVP (ou d'un système par analyse-resynthèse) : spécifier un son et des paramètres de traitement.

ReSpect [Marchand, 2007] est un exemple de système de synthèse additive basé sur le système SAS (*Structured Additive Synthesis*) proposé dans [Marchand, 2000], permettant la mise en œuvre de centaines d'oscillateurs simultanément, et intégrant différentes techniques d'optimisation et de contrôle. Le contrôle de ce type de système s'effectue généralement par la spécification des paramètres individuels des différents partiels produits par chacun de ces oscillateurs, selon un mode "polyphonique" (figure 2.2).

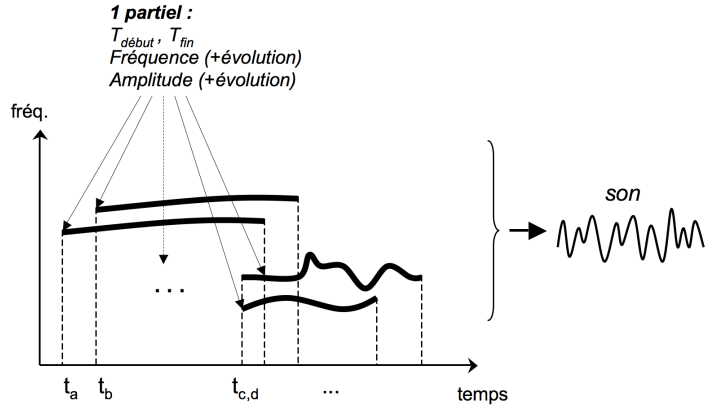


FIGURE 2.2: Schéma de contrôle d'un système additif classique : spécifier les positions, paramètres et évolutions des différents partiels.

Le synthétiseur CHANT [Rodet *et al.*, 1984] [Rodet *et al.*, 1985] intègre également un certain nombre de techniques liées aux modèles additifs dont celui des Fonctions d'Ondes Formantiques (ou FOF). Initialement dédié à la synthèse de la voix [Bennett et Rodet, 1989], son utilisation a été étendue à la synthèse sonore en général. Ce synthétiseur permet un contrôle des générateurs de FOF par des paramètres globaux (pouvant éventuellement être modifiés par des sous-programmes) : il s'agit de décrire l'état d'un "patch", ou combinaison de modules de génération de signaux (bruits, FOF, etc.) ou de filtrage, préalablement choisi, en spécifiant les paramètres de ces modules à différents moments (par exemple les fréquences, amplitudes, largeurs de bandes d'un ensemble de FOF). Le synthétiseur réalise alors une interpolation des paramètres entre ces états pour calculer le signal numérique en sortie (voir figure 2.3).

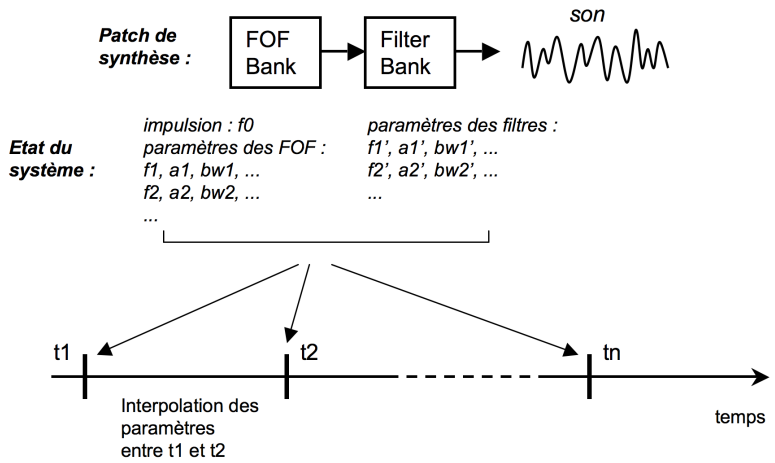


FIGURE 2.3: Schéma de contrôle du synthétiseur CHANT : spécifier l'état des différents modules d'un "patch" de synthèse (en haut) à différents instants.

2.3 Langages de synthèse

Les outils auxquels nous allons nous intéresser ici ont une vocation plus généraliste que les systèmes de synthèse précédents. Ils permettent une opérabilité sur le paramétrage mais également sur les algorithmes mêmes de la synthèse sonore. Ces outils constituent ainsi une première étape dans une approche compositionnelle de la synthèse, puisqu'ils rapprochent le travail du compositeur de la programmation des processus conduisant directement à la production du son.[2] De ces processus émergeront ainsi des représentations personnalisées qu'il s'agira ensuite de déterminer dans le cadre du contrôle de la synthèse.

S'agissant de systèmes dotés de primitives et d'une syntaxe bien définie, nous appellerons ces outils *langages de synthèse*. Nous verrons que certains de ces langages intègrent également des éléments de contrôle plus ou moins avancés.

2.3.1 Csound et les langages Music N

Initiée par Max Mathews dans les années 1950, la famille des MUSIC *N* regroupe un ensemble de langages dédiés à la synthèse sonore [Mathews, 1969]. Après la création de MUSIC I et II, MUSIC III lance réellement en 1960 le modèle qui inspirera cette famille de langages de synthèse, dont MUSIC IV et MUSIC V, du même auteur, mais également de nombreux autres comme CMUSIC [Moore, 1990] ou CSOUND (par Barry Vercoe, 1986), qui est aujourd'hui l'un des plus utilisés [Boulanger, 2000].

Il s'agit de langages permettant de construire des instruments de synthèse à partir de modules élémentaires de traitement ou de génération de signaux connectés les uns aux autres. Ceux-ci incluent des oscillateurs, tables d'ondes, générateurs de bruit, d'enveloppes, filtres, et autres modules plus complexes. Ensemble, ils définissent dans un *instrument* un algorithme répondant à des évènements et données de paramétrage externes, définis dans une partition (ou *score*), et produisant un signal numérique.

La définition d'instruments (dans un "orchestre" – *orchestra*) et le contrôle de ces instruments dans la partition (*score*) sont réalisés dans des fichiers de texte. Un instrument CSOUND (fichier *orchestra*) simple peut se présenter de la manière suivante :

```
instr 1
k1 oscil1 0, 1, p3, 2
asig oscil p4, p5, 1
out asig * k1
endin
```

[2]Cette démarche implique qu'un compositeur ait un minimum de connaissances relevant du domaine du signal et de la synthèse. On pourra rapprocher cette idée au fait qu'un compositeur de musique instrumentale doive savoir, ou au moins avoir idée des caractéristiques et des modes de jeu des instruments qu'il utilise. Les travaux sur les interfaces utilisateur auront par ailleurs pour but de faciliter cette démarche.

Cet instrument est créé par combinaison de modules élémentaires prédéfinis, et correspond au processus de synthèse illustré sur la figure 2.4.

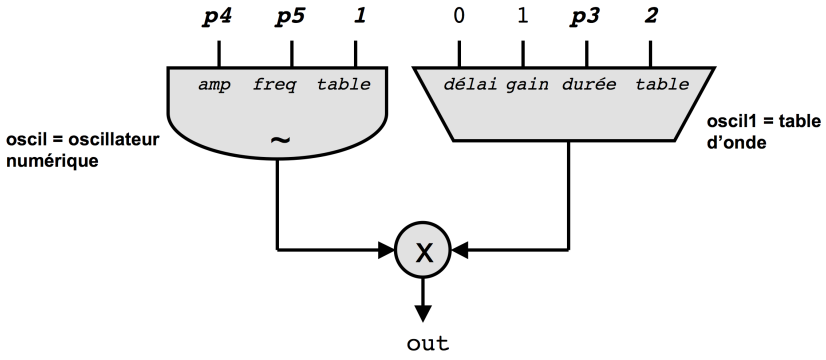


FIGURE 2.4: Représentation du processus de synthèse de l'exemple d'instrument CSOUND. La sortie d'un oscillateur d'amplitude $p4$ et de fréquence $p5$, lisant ses valeurs dans une table (*1*), est multipliée à la sortie de la lecture d'une autre table (*2*) de durée $p3$.

Il s'agit d'un oscillateur d'amplitude et de fréquence variables contrôlé par une enveloppe dynamique. Différents paramètres, indiqués par la notation `pi` ou correspondant aux indices de tables dans l'instrument CSOUND, ne sont pas spécifiés dans le processus : amplitude, fréquence, et table d'onde de l'oscillateur, ainsi que la durée et le profil de l'enveloppe dynamique. Ils seront déterminés lors de la mise en relation de cet instrument avec une partition (*score*).

Le fichier *score* reproduit ci-dessous permet de spécifier des instructions activant l'instrument construit précédemment. Il contient les données permettant le contrôle et le paramétrage de l'*orchestra*, par des tables (par exemple enveloppes, ou tables d'ondes pour les oscillateurs) et des commandes ponctuelles (*note statements*) destinées aux instruments qu'il contient :

```
f1 0 4096 10 1
f2 0 2048 7 0.000 614 1.000 1434 0.000
;p1 p2 p3 p4 p5
i1  0  1  1  440
i1  1  1  0.6  380
ee
```

Nous venons donc avec cet exemple de définir un système de synthèse additive permettant de synthétiser un son par l'assemblage et l'organisation d'un certain nombre de partiels. De la même manière, par l'association de modules pour la création d'instruments et la production des partitions correspondantes, CSOUND permet d'implémenter une grande variété de techniques de synthèse, allant de la synthèse additive, la synthèse granulaire, la synthèse par fonctions d'ondes formantiques, jusqu'à la modulation de fréquence et la synthèse de Karplus-Strong, voire de combiner ces techniques dans des instruments plus complexes.

Les deux fichiers sources, *orchestra* et *score*, sont ensuite traités par CSOUND pour produire un fichier audio. Pour cela, les évènements définis dans le *score* sont triés et le son est obtenu par le calcul des signaux issus des instruments correspondants, à partir des paramètres définis dans ces évènements. Un *taux de contrôle* permet d'optimiser ce calcul en fixant la fréquence de rafraîchissement des paramètres à une valeur inférieure au taux d'échantillonnage du signal. Le calcul du signal de sortie peut donc se faire par blocs et non échantillon par échantillon.[3]

2.3.2 Un langage pour le traitement des signaux

FAUST (*Functional AUdio STreams* [Orlarey *et al.*, 2004]) est un langage fonctionnel dédié au traitement du signal. Les langages fonctionnels considèrent les fonctions comme objets primitifs ; un processus défini dans FAUST est ainsi une fonction qui traite une autre fonction (un signal $s(t)$). Etant donné que les processus produisent des signaux sonores en sortie, on pourra considérer un tel processus comme processus de synthèse sonore à part entière (voir chapitre 1, section 1.2.1).

Des blocs de traitement sont assemblés dans le langage par la composition fonctionnelle d'un programme. Un nombre relativement petit d'opérateurs élémentaires permet cet assemblage : *parallel*, *sequence*, *split*, *merge*, et *recursion* (mise en boucle d'un bloc sur lui-même), ainsi que des fonctions classiques de génération et de traitement des signaux. Le programme décrit dans le langage à partir de ces éléments peut alors être interprété sous forme d'un *block-diagram*. Le compilateur FAUST en extrait la sémantique, c'est à dire détermine l'essence du calcul qui y est écrit afin de générer du code (C++) optimal. Des éléments d'interface utilisateur sont également définis de façon générique dans le langage, permettant ensuite au compilateur de déployer le code pour une plateforme (logicielle, graphique) choisie.

La figure 2.5 montre un exemple de traitement créé dans FAUST.

2.3.3 Langages pour les modèles physiques

MODALYS[4] [Morrison et Adrien, 1993] [Eckel *et al.*, 1995] [Ellis *et al.*, 2005] est un logiciel de synthèse modale développé à l'IRCAM permettant la construction de systèmes vibrants à partir d'objets simples (cordes, plaques, membranes, colonnes d'air, etc.) et d'excitateurs (plectres, archets, marteaux, etc.) liés les uns aux autres par des connections statiques ou dynamiques (percussions, frottements, pincement, etc.) Une interface en LISP permet de spécifier un tel système (objets, connexions, points d'accès), de le mettre en

[3] On retrouvera cette notion de taux de contrôle dans de nombreuses applications de contrôle de la synthèse. En effet les variations perceptibles des paramètres de contrôle sont pour la plupart d'un ordre de fréquences beaucoup plus basses que celui des fréquences audibles.

[4] Initialement appelé MOSAÏC.

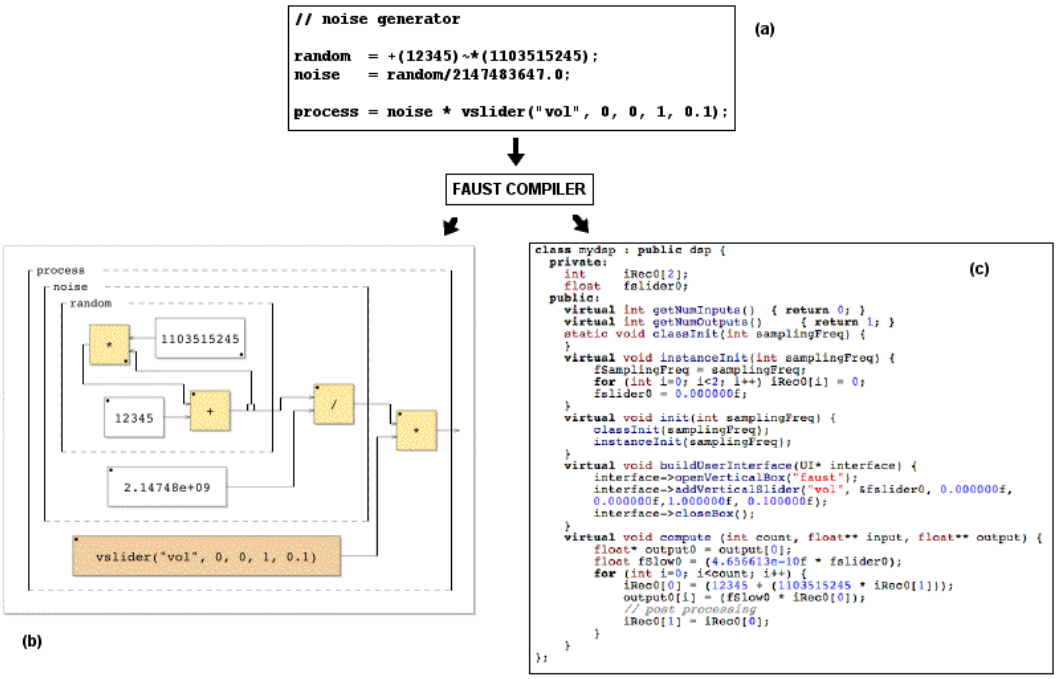


FIGURE 2.5: Création d'un traitement dans FAUST : (a) code FAUST, (b) interprétation sous forme de diagramme , et (c) interprétation sous forme de code C++.

résonance (activation des connexions), et d'"enregistrer" le signal produit en des points d'écoute donnés. Ce mode de création des processus de synthèse sous forme de programmes LISP permet donc de classer cet environnement parmi les langages de synthèse.

CORDIS-ANIMA [Cadoz *et al.*, 1993] est un autre type de langage pour les modèles physiques suivant le paradigme "masses–interaction". Ce langage permet la création des systèmes physiques par assemblage de masses ponctuelles et d'interactions physiques (élasticité, frottement, amortissement, butée, etc.) dans des espaces mono-, bi-, ou tri-dimensionnels. Il s'agit donc d'un langage très général permettant la modélisation et la simulation de modèles physiques aussi bien pour la synthèse sonore que pour la simulation de mouvements et l'image de synthèse.

2.3.4 Environnements temps-réel

En règle générale, on qualifie de temps réel un système permettant d'assurer le temps d'exécution d'un processus au même titre que le résultat de celui-ci. Pour la synthèse sonore, on parlera donc de temps réel lorsqu'un système est capable de synthétiser les échantillons numériques d'un signal avec une vitesse supérieure ou égale au taux d'échantillonnage de ce signal. Ce type de système permet de réaliser des processus de synthèse sonore activés par des flux de données et offre un rendu sonore immédiat et une certaine interactivité sur les paramètres de contrôle (ces données pouvant provenir d'actions dans le systèmes ou de captations en temps réel).

Les systèmes temps réel sont très couramment utilisés pour la création de processus de traitement du signal et de synthèse sonore, avec en particulier les environnements MAX/MSP [Puckette, 1991] et PUREDATA [Puckette, 1996]. Initialement créé pour la gestion de flux d'évènements de contrôle de type MIDI (c'est-à-dire des données de type *note on / note off*, ou des signaux à bande passante relativement basse), le système MAX fut par la suite étendu au traitement des signaux audio (sous l'appellation actuelle de MAX/MSP). La figure 2.6 montre un programme de synthèse (*patch*) dans l'environnement MAX/MSP.

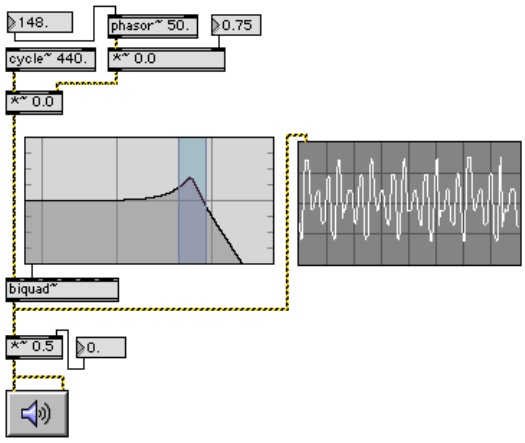


FIGURE 2.6: Un programme de synthèse dans MAX/MSP.

D'un point de vue formel, ces systèmes sont généralement assimilables à la partie *instrument* que nous avons vue avec CSOUND, à cela près que les modules de traitement sont soumis à une excitation continue par les flux de données entrants, et que ces données sont générées et transmises en temps réel. Les modules sont assemblés entre eux par des connections graphiques, et peuvent éventuellement être encapsulés dans des sous-patches pour une structuration des processus par l'intermédiaire d'*abstractions* (unités sémantiques pouvant renfermer des sous-processus plus ou moins complexes).

Des évènements et signaux traversent ces modules ; si l'un d'entre eux est un *DAC* (*Digital/Analogic Converter*), alors le signal se transforme en un son dirigé sur la sortie audio du système. L'exécution d'un programme défini dans ce type d'environnement suit donc un paradigme dit de *data flow* (flux de données). Le contrôle sur les données est immédiat : les messages transitent dans le programme pour activer des interrupteurs, des *sliders*, des filtres, et autres modules, et peuvent provenir de sources extérieures et/ou d'interfaces avec l'utilisateur.

En plus de leur caractéristique temps réel, une particularité des environnements cités précédemment, qui a fait leur popularité, est leur aspect graphique, permettant une relative accessibilité et une meilleure représentation mentale des processus de synthèse et de traitements sonores qui y sont créés.

Ces environnements offrent ainsi une grande liberté dans la construction et le paramétrage des processus de synthèse. Le caractère immédiat et interactif de leur utilisation est un autre avantage : l'expérimentation et l'exploration des différentes possibilités et combinaisons dans les programmes de synthèse seront facilitées et plus efficaces dans ce contexte :

> "Pour imaginer un timbre inexistant in abstracto [...], la seule autre manière est l'expérimentation ; [...] on regarde comment se comportent les sons dans différents types de régimes, on trouve des choses intéressantes, ou pas..." Ph. Manoury.

Cependant, nous essaierons de le montrer par la suite (dans le chapitre 10), les environnements temps réel peuvent aussi se montrer limités pour effectuer certains calculs plus complexes ou d'une portée plus avancée dans un développement musical à long terme.

2.4 Contrôle de la synthèse

Les représentations informatiques pour la musique peuvent être de natures très diverses. [Roads, 1996] liste différents aspects de ces représentations dans un système de composition. On y trouve en particulier la manière dont la musique est présentée (graphiquement) au musicien ; la manière dont elle est représentée dans l'ordinateur (structures de données internes) ; les possibilités de contrôle que cette représentation offre au musicien ; ou encore la façon de communiquer entre les différents programmes et appareils mis en jeu (protocoles). La première catégorie, concernant la représentation externe au musicien, découle plus ou moins directement des autres, qui sont généralement occultées à l'utilisateur. Elle s'avèrera particulièrement importante dès lors que l'on s'oriente vers des applications musicales et non uniquement expérimentales [Eckel, 1992], et est au centre des problématiques du contrôle de la synthèse.

On appellera outils de contrôle de la synthèse les dispositifs, environnements ou programmes destinés à créer ces représentations en vue de leur passage sous forme de paramètres dans un programme ou système de synthèse.

> "L'ensemble des techniques et des stratégies de génération des représentations des sons est appelé contrôle de la synthèse. " [Depalle et Rodet, 1993]

Les systèmes de contrôle de la synthèse ont pour objectif de traiter le problème de la création de représentations sonores en plaçant ces représentations à un niveau d'abstraction plus élevé, accessible au musicien. Le travail de synthèse à proprement parler est délégué à un processus distinct avec lequel est généralement établie une communication. On parle donc en principe d'un contrôle de "haut niveau", dans la mesure où il occulte (ou permet l'abstraction) à l'utilisateur des processus de traitement et de synthèse du signal (dits de "bas niveau").

2.4.1 Outils et environnements de contrôle

Entre les différents types de systèmes de synthèse (a priori des applications figées) et les langages de synthèse (environnements ouverts permettant la constitution de processus divers et personnalisés), les problématiques de contrôle sont sensiblement différentes.

Dans les systèmes de synthèse qui offrent des représentations de type "notes" (ensemble d'évènements de hauteurs, durées, intensités, fixes), la notion de contrôle de la synthèse n'a pas grand intérêt à être discutée, tant elle se confond avec le schéma musical traditionnel. Il s'agit d'écrire une partition, telle qu'on la connaît dans le domaine instrumental (et créée par moyens informatiques ou pas), qui doit alors simplement être transcrite dans le format de données adapté au programme de synthèse en question. C'est le cas des synthétiseurs par échantillonnage du commerce, mais également souvent de systèmes de synthèse additifs ou FM. La standardisation de ce type de contrôle calqué sur le mode instrumental, en particulier grâce au protocole MIDI [Loy, 1985], permet de mettre en œuvre facilement un tel système dans lequel le processus de composition, qui vise l'écriture d'une partition, et le processus de synthèse sonore, qui "lit" cette partition, sont relativement indépendants. Avec la plupart des systèmes de synthèse, cependant, un accès plus ouvert aux structures internes du son est possible, induisant l'utilisation de données complexes et hétérogènes pour la construction des représentations correspondantes. C'est ce que nous verrons ici avec des exemples d'environnements pour le contrôle de synthétiseurs ou processeurs de signaux basés sur les techniques spectrales et additives (section 2.4.2).

La flexibilité des langages de synthèse pose différemment le problème des environnements de contrôle. Une partie du contrôle, nous l'avons évoqué plus haut, s'insère à l'intérieur même de ces langages, dans les interfaces proposées pour la définition des processus. A celle-ci doivent s'ajouter des stratégies de contrôle à proprement parler pour le paramétrage de ces processus, pouvant aller de la création d'évènements, de la spécification de courbes et enveloppes diverses spécifiant les évolutions temporelles des paramètres, jusqu'à la mise en place de structures spécifiques plus complexes.

Ainsi, avec CSOUND, la séparation *score/orchestra* permet d'envisager des outils de contrôle non seulement pour la création et la représentation d'instruments (avec par exemple des interfaces graphiques permettant de connecter entre eux modules de traitement ou de génération de signaux), mais également pour la création des *scores* à l'aide de structures de données allant des courbes de contrôle jusqu'aux représentations plus classiques (évènements de type "notes"). De nombreux outils logiciels ont été développés dans cette optique.[5]

Nous verrons donc ensuite d'autres exemples de systèmes de contrôle spécifiques aux langages de synthèse, principalement autour des systèmes temps réel (section 2.4.4), et

[5]Le programme CECILIA [Piché et Burton, 1998] en est certainement l'un des plus complets. Plus d'informations et documentations sur Csound et les développements autour de ce langage sont disponibles en ligne sur `http ://www.csounds.com`.

passerons également par les langages que nous avons évoqués pour la synthèse par modèles physiques, qui encore une fois se distinguent par une approche singulière (section 2.4.3).

2.4.2 Contrôle de systèmes additifs et spectraux

Systèmes additifs

Les deux exemples suivants illustrent des stratégies possibles pour le contrôle de processus relevant principalement de techniques de synthèse additives, par la structuration (notamment temporelle) des représentations correspondantes.

DIPHONE [Rodet et Lefevre, 1997] est un logiciel permettant la mise en œuvre de différentes techniques de synthèse (synthèse additive, synthèse par FOF, etc.) selon une approche concaténative. DIPHONE utilise en particulier le synthétiseur CHANT : il s'agit d'assembler des "briques" (appelées *diphones*, du fait de la vocation initiale du logiciel à synthétiser la voix) les unes à la suite des autres, chacune contenant des valeurs de paramètres de synthèse correspondant à des transitions entre deux états stables ("phonèmes", dans le cas de la voix) du synthétiseur (voir figure 2.7). Le paramétrage CHANT se fait en effet par des paramètres globaux spécifiant des états du système à des moments particuliers (voir section 2.2), induisant une notion de continuité mais également de monophonie, par opposition aux possibilités polyphoniques d'autres synthétiseurs additifs. On constitue donc au préalable des dictionnaires de diphones (notamment à partir d'analyse de sons réels) dans lesquels on puise des valeurs pour reconstituer des sons. Les diphones assemblés sont de durées variables. A travers les transitions, définies par l'assemblage des diphones, ce système permet de contrôler la synthèse dans la durée et dans la continuité, et d'établir par là une certaine "prosodie" dans les sons de synthèse.

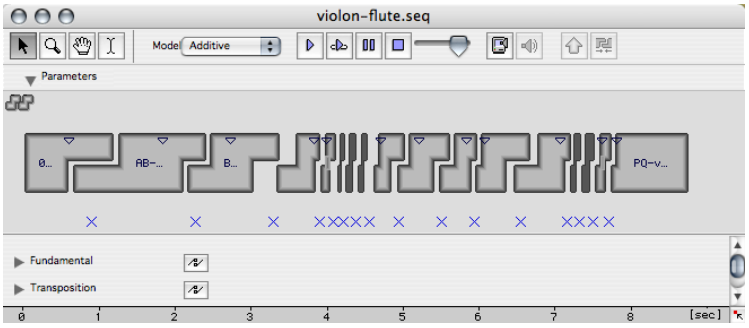


FIGURE 2.7: Le logiciel DIPHONE.

BOXES [Beurivé, 2000] est un environnement visuel qui propose une organisation temporelle hiérarchique des objets musicaux (voir figure 2.8). Ces objets (représentés par des boîtes rectangulaires) contiennent des descriptions sonores spectrales interprétées lors de la synthèse par le système additif SAS de [Marchand, 2000]. Complémentairement à l'organisation hiérarchique des données de représentation permise par cet environnement, l'un de ses principaux intérêts se situe au niveau de la spécification temporelle de celles-ci les unes

par rapport aux autres, réalisée par la déclaration de contraintes temporelles (relations de Allen [Allen, 1983]) établies entre les boîtes, et constituant un modèle d'organisation logique supplémentaire.

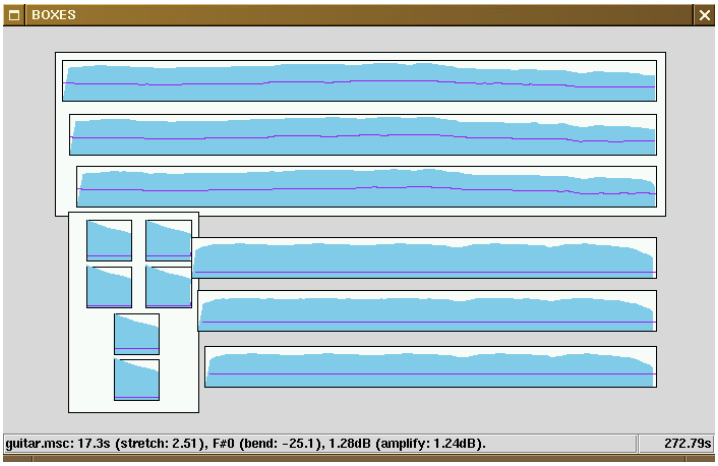


FIGURE 2.8: Le logiciel BOXES.[6]

Il s'agit ici, en opposition à l'exemple précédent, d'un système polyphonique et hiérarchisé, ne permettant cependant pas de contrôle avancé sur la structure interne et les articulations entre les objets.

Analyse-resynthèse, vocoder de phase

Différentes applications utilisent des données issues de l'analyse pour paramétrer des processus de traitement et de synthèse sonore. C'est le cas de la plupart de celles se basant sur le principe du vocoder de phase, qui fournit une représentation des signaux sur laquelle des transformations d'ordre fréquentiel ou temporel, potentiellement renforcées par une représentation graphique, sont applicables [Eckel, 1992].

Le logiciel AUDIOSCULPT [Bogaards et Röbel, 2004] [Bogaards, 2005] offre une interface graphique permettant le contrôle du programme SUPERVP. Cette interface, visible sur la figure 2.9, propose de visualiser parallèlement le signal sous sa forme classique (fonction du temps) et son analyse sous forme de spectrogramme. Elle permet de spécifier, à l'aide d'une palette d'outils, des traitements à appliquer sur cette dernière représentation (c'est-à-dire dans le domaine spectral, voir chapitre 1, section 1.2.2), localisés dans le temps et dans les fréquences (et représentées par des formes géométriques sombres sur l'exemple de la figure). Par ailleurs, un séquenceur de traitements (en bas sur la figure) permet d'organiser dans le temps la position et la durée de ces traitements.

Une fois tous les traitements définis et paramétrés, une commande "*process treatments*" appelle SUPERVP qui calcule un nouveau son à partir de toutes ces données.

[6]Image issue du site web `http ://scrime.labri.fr/logiciels/BOXES/`.

AudioSculpt fait également appel à un autre système de synthèse appelé pm2, qui implémente des processus d'analyse et de synthèse additives. Concrètement, cet outil permet de réaliser des suivis de partiels sur le signal, c'est-à-dire d'extraire des données harmoniques de plus haut niveau que la représentation en spectrogramme. Ces données peuvent également être modifiées et utilisées ensuite pour la synthèse.

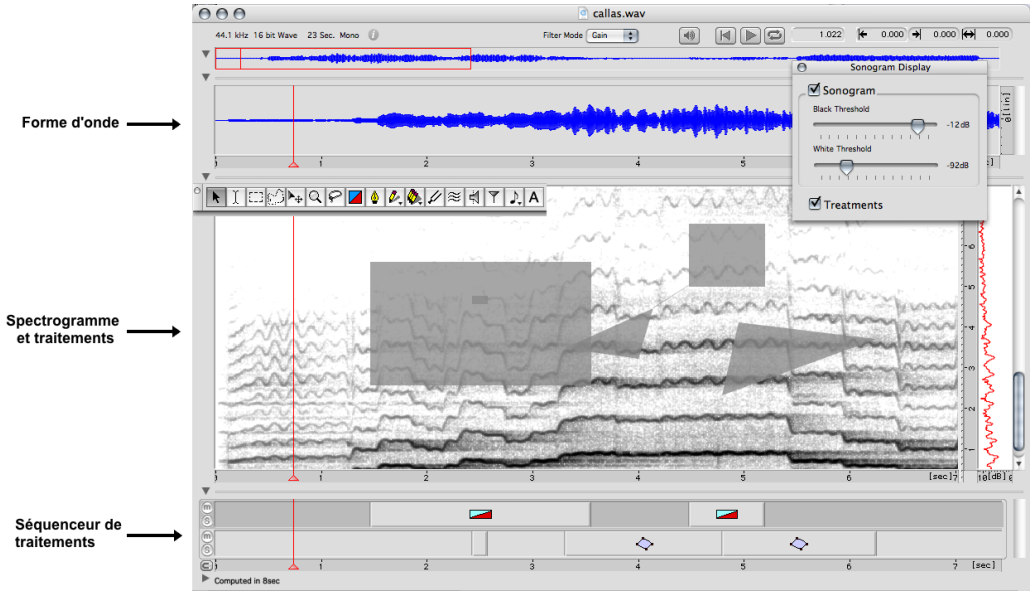


Figure 2.9: AudioSculpt : un logiciel pour l'analyse, le traitement et la synthèse sonore.

Spear [Klingbeil, 2005] est une autre réalisation récente permettant la représentation et le traitement des sons suivant le modèle du vocoder de phase, offrant elle aussi une interface graphique pour l'édition et la transformation des signaux. La représentation y est construite autour des partiels et de leur évolution individuelle, c'est-à-dire d'un point de vue orthogonal à celle proposée dans AudioSculpt (de type TFCT) qui consiste en une distribution d'énergie sur toutes fréquences et pour chaque segment (trame d'analyse/synthèse TFCT) du signal.

Le système Lemur [Fitz et Haken, 1996] propose une interface semblable pour la représentation des signaux, basée sur un système d'analyse/synthèse additive inspiré de [McAulay et Quatieri, 1986]. Phonogramme [Lesbros, 1996], ou plus récemment MetaSynth[7], présentent également des interfaces sous forme de plan temps/fréquence, mettant en avant le côté purement visuel et permettant de mettre en relation son et image par le biais d'outils d'édition inspirés du domaine du graphisme (sans partir nécessairement d'un son original).

[7]http ://www.uisoftware.com/

La synthèse par descripteurs de haut niveau

Même s'il n'existe pas d'outils réellement opérationnels dans le domaine, la notion de synthèse par descripteurs de haut niveau mérite d'être mentionnée dans cette section sur le contrôle de la synthèse. Celle-ci se base sur des techniques acquises dans le domaine de l'analyse du signal et de la perception sonore, consistant à décrire ce signal selon des qualificatifs perceptifs : brillance, rugosité, douceur, etc. [McAdams *et al.*, 1995]. Ces attributs sont associés à des caractéristiques des signaux (comme le centroïde spectral, l'étendue spectrale, etc.) que l'on appelle des descripteurs de haut niveau [Peeters, 2003]. Ces descripteurs peuvent par exemple être utilisés pour la classification et l'indexation des bases de données de sons. En revanche, ceux-ci ne s'obtiennent pas par un processus d'analyse réversible. En d'autres termes, on ne sait pas directement synthétiser un signal à partir d'un ensemble de descripteurs de haut niveau. Il est possible en théorie, cependant, de transformer un signal de façon à lui donner une caractéristique plus ou moins forte par rapport à un descripteur donné (par exemple en modifiant la répartition de l'énergie sur le spectre [Tardieu, 2004]).

En principe, il s'agirait donc de contrôler un processus de synthèse sonore par des attributs perceptifs, traduisibles dans le langage naturel, c'est-à-dire par des mots, plutôt que par des paramètres numériques. Une idée sous-jacente, en termes de contrôle de la synthèse, est en effet que l'on pourrait généralement mieux exprimer ce que l'on souhaite obtenir, que comment l'obtenir.[8] Le problème est donc de faire correspondre ces spécifications (de haut niveau) avec des paramètres de synthèse (de bas niveau). [Miranda, 1995] propose par exemple une approche de ce type dans le cadre d'un système d'apprentissage interactif, qui permettrait de contrôler un tel système tout en personnalisant à la fois le langage de spécification et la correspondance avec les paramètres du système de synthèse.

2.4.3 Le contrôle avec les modèles physiques

Genesis [Castagne et Cadoz, 2002a] [Castagne et Cadoz, 2004] est un environnement de contrôle pour la synthèse par modèles physiques basé sur le langage Cordis-Anima. Des éléments graphiques tels que les masses ponctuelles (définies par une masse, une position, une vitesse), et les liaisons (élasticité, frottement, etc.) permettent la création de réseaux de types masse-interaction, activés par la mise en mouvement de ces éléments (simulant une percussion, un frottement, etc.)

Genesis est basé sur la version monodimensionnelle de Cordis-Anima : si la représentation graphique est tridimensionnelle, une seule dimension est utilisée pour les interactions

[8] Cette idée ne s'accorde pas réellement à notre conception d'un système de composition tel que nous le décrirons par la suite, dans la mesure où la composition consisterait précisément en l'explicitation du processus conduisant à un résultat souhaité.

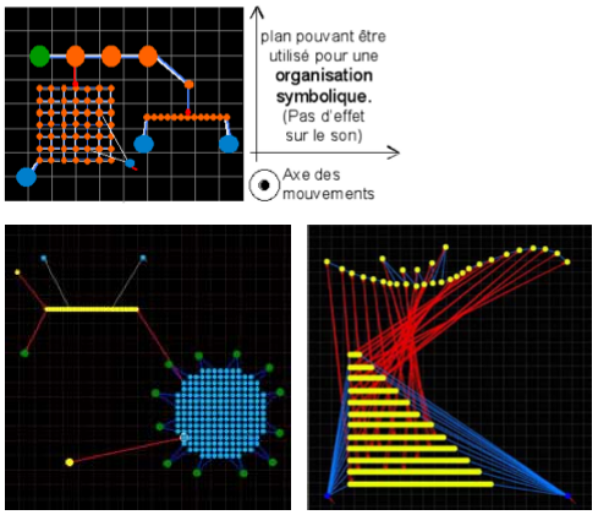


FIGURE 2.10: GENESIS : contrôle du système de synthèse par modèles physiques avec CORDIS-ANIMA. Constructions simples entre masses et interactions (en haut) et exemples de systèmes complexes créés dans l'environnement (en bas).[9]

et la synthèse sonore. Le plan orthogonal permet cependant de disposer et interconnecter les objets et d'offrir une représentation graphique librement structurée sur celui-ci (voir figure 2.10) et sans incidence sur le résultat physique et sonore. Cette conception de l'espace est un exemple de séparation claire du niveau topologique, que l'on peut rapprocher de celui de la construction, de l'écriture, et du niveau du résultat physique (sonore). Une projection sur la dimension utilisée pour la synthèse (orthogonale au plan de la représentation) permet d'observer le comportement "réel" du modèle.

La généralité du système CORDIS-ANIMA permet d'y définir selon un même formalisme des modèles sonores, acoustiques, et des modèles "basses fréquences", évoluant dans le domaine "macro-temporel". On obtient ainsi une approche intégrée du contrôle à différentes échelles. GENESIS permet ainsi, par exemple, de simuler un ordonnancement d'évènements de contrôle par la spécification de masses ayant des positions et vitesses données de telle sorte qu'elles activent (percutent) d'autres masses vibrantes à des moments déterminés. Des stratégies de contrôle élaborées intégrant systèmes physiques, évènements et gestes "instrumentaux" peuvent donc s'organiser dans ce système par la gestion des masses et des mouvements (voir par exemple [Cadoz, 2002]). Des mécanismes de hiérarchisation permettent également d'aborder la complexité des structures par l'intermédiaire de "capsules", objets rassemblant un certain nombre d'éléments ou de sous-capsules et présentant un nombre réduit de points d'accès pour des liaisons avec le contexte dans lequel ils sont utilisés.

[9]Illustrations extraites de [Castagne et Cadoz, 2002b], avec l'aimable autorisation des auteurs.

2.4.4 Le contrôle dans les environnements temps réel

Dans une certaine mesure, les environnements de temps réels, surtout lorsqu'ils sont visuels, peuvent contenir eux-mêmes leur propre système de contrôle. Des outils et objets spécialisés y sont en effet intégrés (notamment dans MAX/MSP) et permettent la visualisation et l'interaction sur les données transmises dans les processus.

Cependant le contrôle prédéfini ("écrit") d'un processus de synthèse dans ces environnements est en général problématique, ceux-ci étant en principalement destinés à répondre à des messages et signaux dans une interaction avec des dispositifs externes. La contrainte du temps réel oblige alors à une portée temporelle réduite, et le traitement du temps en tant que paramètre musical est très limité : il n'y a pas de réelle place pour les structures temporelles complexes (composées) dans ce contexte. Les objets de type *timeline*, ou séquenceurs rudimentaires, permettent en principe tout au plus de programmer le déclenchement de messages à des instants prédéterminés.

On notera cependant que l'une des motivations pour la création de PUREDATA était la conception d'un environnement intégrant temps réel avec gestion des structures de données plus complexes. Dans ce système, les *data structures* permettent ainsi l'ordonnancement et le contrôle de processus temps réel sur un axe temporel, grâce à une interface graphique (voir figure 2.11). Avec cette représentation, les évènements sonores peuvent être retardés et soumis à un séquencement contrôlé graphiquement. Dans la limite d'une chronologie linéaire, des relations de cause à effet peuvent être programmées entre les évènements. Si le contrôle est donc encore quelque peu limité, le rendu visuel et interactif de l'édition de programme utilisant cette interface, et surtout l'apparition du temps comme paramètre de contrôle, permettent d'envisager avec celle-ci de réelles partitions de synthèse en temps réel [Puckette, 2002].

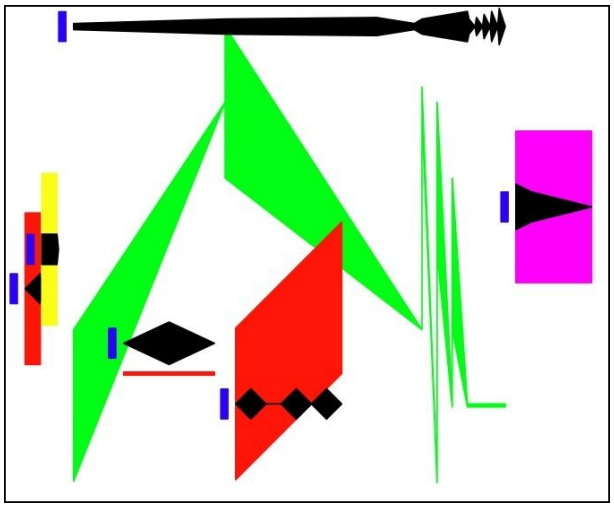


FIGURE 2.11: Editeur de *data structures* dans PUREDATA.[10]

La librairie FTM actuellement développée à l'IRCAM dénote également un effort dans le même sens, avec l'intégration dans MAX/MSP de structures musicales complexes et proches des données musicales [Schnell *et al.*, 2005].

Contrôle du déroulement temporel et de l'exécution : IanniX

Nous voyons petit à petit que la question du contrôle de la synthèse va généralement de pair avec des problématiques temporelles : organiser des objets, des paramètres dans le temps, contrôler leur déroulement, leurs enchaînements et évolutions.[11]

L'environnement IANNIX [Coduys et Ferry, 2004], sur lequel nous nous attardons un instant, présente une approche intéressante de ces problématiques. Celui-ci propose une représentation du temps correspondant aux deux axes du plan, sur lequel l'utilisateur définit des "trajectoires temporelles" (rectilignes, courbes, circulaires, etc.) Des curseurs peuvent alors être disposés et vont suivre ces trajectoires selon des directions et vitesses variables, et activeront des évènements de synthèse lorsqu'ils rencontrent des objets disposés sur ou autour de ces trajectoires. Ces objets peuvent correspondre à des valeurs simples (*triggers*), ou bien à des courbes définies graphiquement. La rencontre d'un curseur avec l'un de ces objets provoque donc l'envoi d'une (dans cas des *triggers* simples) ou d'une série de valeurs (dans le cas de courbes) vers un système de synthèse (le plus souvent, un *patch*, ou processus de synthèse, défini dans PUREDATA ou MAX/MSP). IANNIX se positionne ainsi clairement du côté du contrôle de la synthèse, comme outil de représentation et de séquencement d'évènements et de données de synthèse ; une sorte de partition offrant une représentation symbolique d'une œuvre (voir figure 2.12).

Les trajectoires peuvent être parcourues simultanément à des vitesses variables ou même dans des sens opposés par différents curseurs. L'espace $temps \times temps$ proposé permet la construction de structures temporelles diverses et entrelacées, différentes temporalités pouvant être développées et coexister dans un même espace et dans une même exécution. Les caractéristiques graphiques des objets du système (trajectoires, curseurs, *triggers*) sont donc étroitement liées au déroulement temporel et à l'exécution des processus : l'association de la partition (ensembles des objets) et des trajectoires définissent la ou les séquences musicales. La correspondance entre la morphologie des trajectoires, le comportement des curseurs (qui peuvent par exemple s'activer mutuellement), et le repère $temps \times temps$, permettent alors d'organiser l'ordonnancement temporel de manière inédite. Cette dualité entre temps et espace, qui fait l'originalité de cet environnement, est discutée par les auteurs dans [Coduys *et al.*, 2003].

---

[10] Figure extraite de [Puckette, 2002].

[11] Ces questions seront traitées spécifiquement dans la quatrième partie de la thèse, en particulier dans le chapitre 10.

[12] Figure extraite de [Coduys et Ferry, 2004], avec l'aimable autorisation des auteurs.

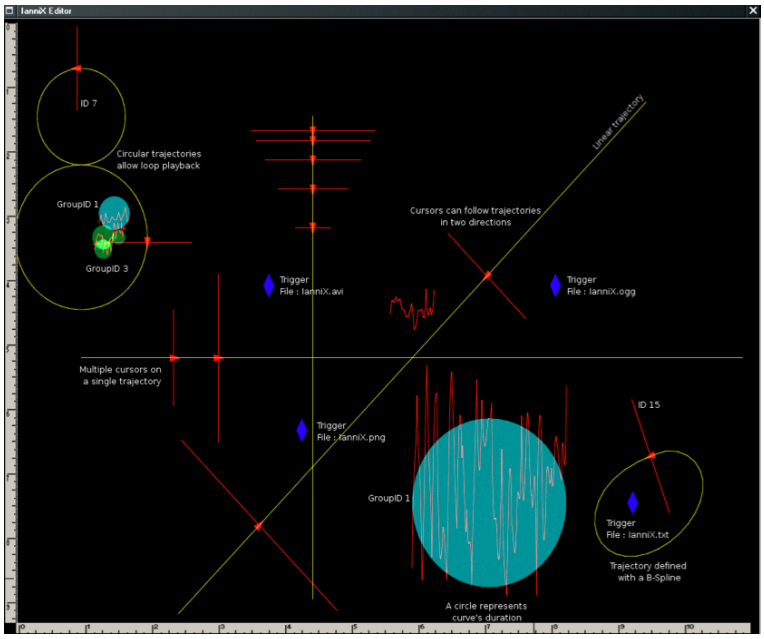


FIGURE 2.12: Le logiciel IANNIX.[12]

2.5 Conclusion

Nous avons présenté dans ce chapitre des notions et travaux relatifs à l'utilisation de la synthèse sonore dans un objectif musical, dont un certain nombre d'environnements de contrôle de la synthèse. Ceux-ci se basent sur des systèmes de synthèse et proposent généralement des interfaces permettant la construction de représentations adaptées à ces systèmes. Ils permettent ainsi de traiter et d'organiser les données de paramétrage des systèmes de synthèse en réduisant la complexité de ces données du point de vue de l'utilisateur. L'aspect graphique des systèmes de contrôle est aussi souvent prépondérant dans la manière dont cet utilisateur peut s'abstraire du domaine du traitement du signal par une symbolique visuelle, pour se concentrer sur des problématiques musicales.[13] D'une manière générale, il s'agit donc d'augmenter le potentiel musical d'une représentation du son par l'établissement d'abstractions ou de conventions graphiques.

Il y a ensuite un *mapping* plus ou moins explicite entre les représentations suggérées par les systèmes de synthèse et celles proposées par les systèmes de contrôle. Ainsi, chaque système de contrôle détermine des points d'entrée sur les processus de synthèse, comme autant de degrés de liberté donnés à l'utilisateur à travers les paramètres qu'il doit spécifier pour obtenir un son de synthèse. Dans cette logique, moins on a de paramètres de contrôle, plus le contrôle est dit (ou perçu) de "haut niveau". Un problème qui se

[13] Tous les environnements de contrôle ne sont pas pour autant visuels ; certains compositeurs expérimentés préféreront travailler directement avec les langages et structures de "bas niveau" afin de conserver le maximum des possibilités de contrôle sur les processus de synthèse.

pose alors est que l'on réduit ainsi le potentiel d'expressivité du système : on sera rapidement limité dans l'expérimentation et la découverte de sons avec une liberté de mouvement réduite. L'idée d'un dispositif de haut niveau réduisant (même judicieusement) paramètres et possibilités de contrôle semble donc trop contraignante au vu des idéaux de la musique électroacoustique. Les langages de synthèse sont quant à eux dotés d'une flexibilité supplémentaire, mais s'éloignent du niveau abstrait nécessaire à la composition musicale.

Gérard Assayag a déjà relevé l'importance de ce type de problème pour l'établissement de systèmes de représentations garantissant le potentiel musical et la liberté esthétique permise par un système :

> "Représenter sous une forme musicale une structure ou un processus abstraits – que l'on souhaite au départ aussi peu contraints que possible pour préserver la liberté d'imagination formelle – implique la maîtrise d'un système de représentations musicales, c'est-à-dire d'un ensemble cohérent de structures symboliques aptes à définir les propriétés des objets musicaux et les relations que ces derniers entretiennent à mesure qu'ils se combinent pour former des assemblages de plus en plus complexes. Le système de représentation devra rendre compte de tous les niveaux d'intégration [...]. La difficulté réside alors dans le repérage et l'extraction d'universaux qu'il sera pertinent de fixer en des représentations informatiques de référence. Une telle étude, fondamentale dans les deux sens du terme, est trop souvent négligée [...], cela conduisant à un déficit de généralité des modèles et donc à des goulots esthétiques [...]. Pourtant, seule la maîtrise de ce niveau profond serait en mesure de garantir et l'ouverture stylistique – la capacité à satisfaire des compositeurs adoptant des points de vue très éloignés les uns des autres – et une communication mieux contrôlée entre pensée musicale et systèmes de production sonore (notamment par synthèse)." [Assayag, 1993]

Avec la synthèse sonore, le matériau musical issu de la composition est converti en signal sonore par un processus informatique accessible au compositeur, et les paramètres de ce processus sont par ailleurs généralisables à tout type de données, comme l'a montré la diversité de systèmes de synthèse existants. L'intégration d'un contrôle de bas niveau sur ces données et processus est donc nécessaire, associé à des stratégies de contrôle de plus haut niveau, ce qui est difficilement envisageable dès lors que l'on se positionne formellement d'un côté ou de l'autre de la frontière. Une navigation flexible entre les domaines des processus de synthèse et celui des processus compositionnels de haut niveau, et entre les différents niveaux d'abstraction intermédiaires, semble une caractéristique nécessaire à une réelle emprise de la composition sur la synthèse sonore.

Chapitre 3

La synthèse sonore dans la composition

Au vu des nouveaux éléments apportés par la synthèse sonore dans la composition, il semble légitime de s'interroger sur la validité des notions relevant du schéma musical traditionnel dans ce nouveau contexte. La culture et l'histoire jouent en effet un rôle prédominant dans la manière d'aborder les techniques nouvelles dans la musique. Or dans la pratique, il existe peu de formalisation musicale relative à la synthèse sonore : les générations actuelles de compositeurs sont les premières à y être confrontées, et peu de repères théoriques solides existent. Tout cela pousse donc à la formation d'une pensée s'appuyant sur les catégories et les concepts traditionnels (issus de la musique instrumentale) pour décrire et expliciter, par analogie ou par opposition, les nouveaux concepts de la musique électronique.

> "Il me semble important de garder des catégories qui appartiennent au monde traditionnel des instruments, et de les porter dans le domaine de la synthèse, non pas pour retrouver des sons, mais pour établir la continuité d'une culture." Ph. Manoury.

Dans ce chapitre nous proposons une brève réflexion sur les répercussions de l'utilisation de la synthèse sonore dans la composition musicale, en abordant successivement celle-ci du point de vue de la distinction classique entre instruments et partitions (section 3.1), de la notation (section 3.2), et du concept d'interprétation (section 3.3). Cette réflexion a été en grande partie nourrie par les entretiens réalisés auprès de Hugues Dufourt, Philippe Manoury et Claude Cadoz, et dont sont rapportés quelques propos.

3.1 Instruments et partitions

Un compositeur crée de la musique en organisant des éléments (ensembles de paramètres musicaux) dans des structures composées, qu'il écrit ou note sur une partition. Ces structures sont interprétées par un ou des musicien(s) à qui il revient la responsabilité de produire le son par l'intermédiaire d'instruments. Suivant ce schéma, il semble donc que

les systèmes de synthèse tels que nous les avons décrits dans le chapitre précédent prennent avec la musique électroacoustique exactement la place de l'interprète/instrument dans le processus total de création sonore (c'est-à-dire dans le processus allant de la formalisation musicale, de la composition, jusqu'à la production d'un son). On parle en effet souvent d'instrument de synthèse pour qualifier un processus, système de synthèse donné a priori.[1] Les langages de synthèse présentés dans la section 2.3 du chapitre 2 introduisent donc la notion de lutherie électronique. S'il se distingue du niveau du contrôle (construction des représentations), l'"instrument" de synthèse est en effet fondamental dans la recherche musicale avec la synthèse sonore :

> "Le compositeur n'est plus seulement utilisateur d'instruments existants, mais il construit son instrument." C. Cadoz.

Ces systèmes, ou instruments, sont donc contrôlés par des paramètres spécifiés avec des outils de contrôle, qui se positionnent ainsi au niveau de la partition. Il s'agit avec ces outils de construire, dans un processus de composition, des représentations permettant de mettre en œuvre un système de synthèse donné pour produire des sons.

Ainsi la dualité entre les notions d'instrument (processus de synthèse présentant un certain nombre de paramètres) et de partition (ensemble de données, ou représentations du son, permettant la mise en œuvre d'un instrument) se retrouve dans de nombreux environnements de synthèse et de contrôle de la synthèse (à commencer par les langages de synthèse de la famille MUSIC N avec les notions d'*orchestra* et de *score*).[2]

A la frontière des domaines de l'écriture et du son, la partition décrit des formes musicales structurées à partir d'instructions d'interprétations. La partition constitue le support sur lequel le compositeur peut "étaler" la musique sous ses yeux. Elle est donc un support de communication de la musique, du compositeur vers l'interprète, ou vers un lecteur, mais également un support d'écriture très important, c'est-à-dire de réflexion, de communication du compositeur avec lui-même. La partition exprime de la musique ce qui est pensé par le compositeur ; elle contient *le substrat du sens de la composition* [Ebbeke, 1990].

Les objets et représentations sonores sont organisés au niveau de la partition par l'intermédiaire d'un système de *notation* et *interprétés* par les instruments. Nous considérerons successivement ces deux dernières notions (notation et interprétation) dans la suite de ce chapitre.

---

[1] [Miranda, 1995] propose une définition de l'instrument comme système identifiable par des caractéristiques timbrales. L'instrument produit des sons comme structures ayant des "caractères" communs, mais à l'intérieur desquelles émergent certaines "valeurs".

[2] Nous avons vu par ailleurs qu'une distinction trop marquée de ces concepts pourrait constituer l'un des problèmes de l'ouverture de la composition électroacoustique.

3.2 Notation

La notation musicale permet de décrire, transmettre et lire la musique par un ensemble de symboles ayant une sémantique définie, articulés dans une partition. [Bennett, 1990] cite trois fonctions principales d'un système de notation musicale : 1) indiquer à l'interprète ce qu'il doit jouer, et à quel moment ; 2) la transmission de la musique ; 3) la réflexion et l'analyse. Il note que les deux premiers aspects ne sont plus strictement indispensables avec la musique électroacoustique : il n'y a pas nécessairement d'interprète, et la transmission peut se faire par le biais de l'enregistrement. Opposons à cela, pour le premier point, que dans le cas de la musique mixte, la notation de la synthèse sonore permet aux instrumentistes et/ou au chef de suivre le déroulement de l'œuvre, par des repères visuels sur la partition ; ou encore qu'une forme de notation entre également en jeu lorsqu'un interprète est chargé, d'une manière ou d'un autre, du contrôle ou du déclenchement des processus de synthèse. De manière générale, il est en effet difficile de parler de musique "écrite" si l'on ne dispose pas d'une notation, sous quelque forme que ce soit.

Le troisième point souligné (la réflexion formelle sur la musique) reste cependant et sans aucun doute le plus problématique dans l'idée d'une notation pour partitions électroacoustiques. Par sa fonction de substitution du réel, la notation constitue en effet une forme de représentation musicale au sens où nous l'avons vue dans le chapitre précédent (section 2.1), et joue à ce titre elle-même un rôle important dans la manière de penser et formaliser des idées musicales. Pour [Malt, 2000], la transcription de l'imaginaire et des représentations musicales dans un mode de notation a pour but de les passer dans un univers symbolique pour pouvoir les manipuler. Le compositeur a en effet souvent besoin d'esquisse (graphique) pour fixer les concepts, et avoir une idée des résultats avant qu'ils ne soient réels. [Dufourt, 1991] analyse et soutient également ce rôle déterminant de l'écriture, en tant que projection de la musique et du son sur un espace (plan), que médiation graphique, dans la formation de la pensée musicale. La notation devient ainsi *un langage dans lequel s'expriment de nombreuses idées durant le processus de composition* [Bennett, 1990]. [Assayag, 1993] note que celle-ci opère *un lien cohérent entre des systèmes d'opérations combinatoires sur des ensembles de symboles et un univers sonore disposant de ses propres règles de fonctionnement perceptif et cognitif*, soulignant par là son influence sur les processus de composition. La notation ne sert donc pas uniquement à représenter la musique, elle permet, et oriente aussi la réflexion sur cette musique.[3]

Différentes stratégies sont ainsi adoptées pour noter une partition électroacoustique. En donnant des informations sur les processus mis en œuvre, ou sur les opérations à réaliser pour une interprétation de l'œuvre (déclenchement de processus, etc.), la notation décrit le "comment faire" de la synthèse : on se rapproche alors de la notion de tablature plus que

[3] [Ebbeke, 1990] parle d'un *système de signes [qui], dès lors qu'il avait atteint à une forme de validité [...], se répercutait sur les contenus musicaux que lui-même avait aidé à fixer.*

de partition. En utilisant des symboles graphiques se rapportant à des éléments perçus des structures sonores, on obtient une notation approximative qui ne peut généralement pas rendre vraiment compte du phénomène représenté. Ces notations schématiques et autres tablatures ne peuvent donc pas réellement décrire la réalité complexe ni l'intention compositionnelle portée par le son. Des descriptions plus complètes des sons et des processus de synthèse peuvent alors être données (généralement sur des supports distincts, "hors partition"), permettant parfois de reconstruire les processus correspondants, dans la majorité des cas de s'en faire une vague idée. Dans la mesure où elles seraient utilisées comme support pour l'écriture, l'une ou l'autre de ces formes de notation pourrait cependant avoir un sens dans une démarche compositionnelle.[4] Mais dans la plupart des cas, la partition a alors pour rôle celui de simple ordonnateur temporel des "objets" de synthèse, plutôt que de vrai support d'écriture.

Dufourt oppose également le problème de la lecture :

> "Quand on a de l'électronique sur une pièce instrumentale, on trouve soit une notation purement signalétique [...] (on ne sait pas ce qui va se passer, on a simplement l'élément de déclenchement), soit une représentation mimétique (on nous dessine en gros l'allure d'un phénomène [...]), mais en aucun cas il n'y a de réelle lecture possible du phénomène électroacoustique..." H. Dufourt.

Le problème de la notation musicale dans le domaine de la musique électronique se pose donc sous différents aspects. Celle-ci doit permettre la lecture, la compréhension de la musique, et l'identification des sons perçus ou imaginés à partir de symboles (visuels) qui les rendraient "manipulables" par la pensée.

> "Le problème est qu'une véritable écriture, ce n'est pas simplement une notation, mais quelque chose qui intègre toute la théorie. Or ces fonctions sont encore dissociées en informatique musicale. On contrôle un ordre, ou un autre, mais on n'a pas cette prise globale, à la fois intuitive et rationnelle. Il faudrait avoir l'intuition du son et en même temps pouvoir dire, en analysant l'écriture, ce qui est écrit et ce que cela signifie." H. Dufourt.

Une notation pour la musique électroacoustique signifierait donc une possibilité de substitution intelligible des phénomènes sonores permettant de signifier les intentions musicales et le résultat correspondant. Si l'on considère la notation musicale conventionnelle, et compte tenu de tous les paramètres entrant en jeu dans un phénomène sonore, cette notation n'en indique qu'une petite partie, mais qui permet d'en exprimer l'intention musicale (à l'aide de durées, de hauteurs et d'intensités). [Dufourt, 1991] note que depuis toujours, la notation est réorganisée et redéfinie à l'introduction de nouveaux paramètres dans la musique, ainsi sous-tendue par la notion de "champ fonctionnel à multiple facteurs". Complémentairement à la représentation complète d'un son que peut constituer un

[4] C'est par exemple le cas dans certaines paritions de Stockhausen, comme *Mikrophonie* (voir chapitre 12, figure 12.11).

programme de synthèse, une notation symbolique exprimerait par de nouveaux facteurs l'intuition et l'intention musicales portées par des sons créés par la synthèse.

Si la conception d'un système de notation semble donc dépasser le cadre de notre travail (celui-ci, s'il existe un jour, se formera certainement de lui-même, avec le temps et les expériences musicales) il n'est cependant pas sans intérêt d'essayer de replacer cette notion, primordiale dans la musique traditionnelle, dans ce nouveau contexte, pour tenter d'entrevoir ce que serait alors une partition rendant compte de processus de synthèse et de leurs paramètres.

> "Il faudra résoudre [le problème de la notation] parce qu'on ne peut pas s'en tenir à des représentations formalisées de type informatique pour faire de la musique." H. Dufourt.

L'absence d'un consensus sur un système de notation peut en effet être vue comme un obstacle à la constitution de discours théoriques sur la musique électroacoustique :

> "L'oreille est bien obligée de travailler avec des catégories qui ne font partie pour l'instant d'aucun consensus. Cela signifie qu'en définitive, ce qui est perçu échappe pour ainsi dire à l'analyse." [Ebbeke, 1990]

Elle peut cependant aussi constituer une ouverture, qui semble pour l'heure nécessaire à la constitution de démarches compositionnelles originales avec la synthèse sonore. Pour [Malt, 2000] la représentation symbolique choisie pour la notation doit en effet correspondre à une sémantique graphique personnelle, définissable par le compositeur. On trouvera ainsi dans la musique électronique ou mixte des systèmes et conventions variées utilisées par les compositeurs, manifestant chacun à leur manière leur travail et leur point de vue sur le son et la synthèse.

[Assayag, 1993] situe par ailleurs la notation en regard de la notion de langage, dans le cadre du développement des environnements de composition assistée par ordinateur :

> "La notation agit avec un logiciel de CAO non seulement comme matérialisation des informations circulant dans le système mais comme support de l'inventivité formelle. A ce titre, la notation devrait être dotée de la même souplesse, de la même ouverture (en termes d'extensibilité et de programmabilité) que le langage même et constituer, à terme, le "milieu" naturel d'expérimentation de l'utilisateur. Les niveaux du langage et de la notation tendront alors à se confondre du point de vue de l'utilisateur." [Assayag, 1993]

Différents environnements informatiques intègrent ainsi des systèmes de notation musicale programmables et personnalisables, généralement sur la base du système traditionnel (voir par exemple [Kuuskankare et Laurson, 2006]). Un des objectifs des environnements de composition actuels, dont il sera question dans les chapitres suivants, sera l'intégration de la notation et des aspects formels et procéduraux dans la construction des structures musicales.

3.3 Interprétation

Si instruments et partitions sont pris en charge par les systèmes informatiques de synthèse et de contrôle, le rôle traditionnel de l'interprète doit être lui aussi envisagé sous un nouveau jour. Dans la musique instrumentale traditionnelle, il est implicite qu'une partition écrite sera lue et interprétée, ce qui permettra de réaliser la musique sous sa forme "sonore". C'est cette interprétation, en fonction de l'information "écrite" contenue dans la partition, qui génère donc l'information sonore. Avec la synthèse, cette phase de production sonore est réalisée par un ordinateur et est donc susceptible d'entrer dans le champ de l'activité de composition.

Le son numérique une fois créé (écrit) est en effet déterminé à l'échantillon près : la dernière part de "liberté", ou d'indéterminisme dans le phénomène sonore produit relève en principe seulement de sa diffusion dans l'espace.[5] La perte d'incertitude et de variabilité liée à la suppression de l'intermédiaire que constitue l'interprète entre la musique écrite et la réalité sonore est ainsi une des problématiques fondamentales de la composition électroacoustique :

> "... le problème du rapport de l'électroacoustique à l'homme, à l'instrument, alors que pendant des millénaires et jusqu'à une époque récente, l'homme n'avait de rapport à la musique qu'à travers la mécanique." H. Dufourt.

Etant donné la complexité de l'établissement des instruments de synthèse et de la construction des représentations correspondantes, il résulte souvent de la synthèse sonore des sons très réguliers, révélant fortement leur caractère synthétique. Ceci peut être souhaité dans certains cas, mais bien souvent les compositeurs ont recours à diverses techniques pour rendre les sons de synthèse plus "vivants". On pourra, par exemple, insérer artificiellement des variations, de l'aléatoire dans certains paramètres des programmes de synthèse, ou dans les représentations qui leur sont associées.

Le compositeur Marco Stroppa, par exemple, introduit par la programmation cette part d'indéterminisme dans le processus de transfert des données de contrôle vers les systèmes de synthèse, afin que deux sons issus d'un même processus et avec les mêmes paramètres (c'est-à-dire deux représentations identiques) soient systématiquement différents, bien qu'identifiables comme étant le "même son".[6] Dans cette démarche, le traitement de certains paramètres des programmes de synthèse relèveraient donc du domaine de l'interprétation, par opposition à ceux qui sont purement compositionnels ("écrits").

Philippe Manoury développe quant à lui le concept de partition virtuelle pour la synthèse sonore. Dans son travail, des processus de synthèse sont préprogrammés, mais

[5] La diffusion du son dans l'espace peut elle aussi faire l'objet de traitements musicaux, notamment par les techniques de spatialisation du son.

[6] Ce procédé nous rapproche ici encore de l'idée d'instrument en tant que système englobant une classe de sons identifiables par leur timbre.

certains des paramètres sont indéterminés a priori et proviennent de l'analyse de captations en temps réel du jeu d'instrumentistes (*"on connaît la nature des transformations mais pas les valeurs exactes qui vont les définir"* [Manoury, 1990]). A l'origine du temps réel, il y avait en effet une recherche visant à se rapprocher du rapport instrumental à la musique, disparu pour un temps avec la synthèse sonore : les paramètres provenant d'un jeu instrumental réel sont infiniment plus "naturels" que s'ils avaient été spécifiés manuellement (ou même à l'aide de programmes). Ainsi, l'interaction possible des processus de synthèse avec le jeu instrumental (notamment par l'analyse en temps réel, le suivi de partition, etc.) offrent des ouvertures sur des nouveaux types de programmes, enrichis par des données musicales provenant du monde réel.

Ces exemples nous montrent donc que la notion d'interprétation, *synonyme d'absence de prédétermination totale* [Manoury, 1990], garde un sens avec la synthèse sonore :

> "A partir du moment où la musique qui est codifiée est incomplète, elle a besoin d'être interprétée pour exister." Ph. Manoury.

L'utilisation de données provenant de sons naturels enregistrés est également un moyen efficace d'obtenir des données riches et/ou naturelles pour le paramétrage des systèmes de synthèse (nous avons parlé par exemple des techniques d'analyse/resynthèse dans les chapitres précédents). Nous reviendrons plus précisément sur cette idée dans la suite de ce travail (chapitre 8).

Enfin, toute l'approche de la synthèse par modèles physiques vise à remédier à un "excès de synthèse" en mettant en avant les notions d'instruments et de geste instrumental, calquées sur le modèle du monde physique.

3.4 Conclusion

Si les problématiques traitées dans ce chapitre demeurent ouvertes, elles nous auront permis de mettre en avant quelques idées sur la question de la composition musicale électroacoustique. Dans le nouveau schéma musical que nous avons essayé d'esquisser, nous avons pu rapprocher les systèmes de synthèse du chapitre précédent de la notion d'instrument, les outils de contrôle à celle de partition. Les catégories traditionnelles (instruments, interprétation, partition, notation) ont donc toujours lieu d'être considérées dans ce contexte, mais sont quelque peu transformées ou déplacées. Nous retiendrons surtout que ces notions interfèrent plus étroitement avec la composition. Les frontières entre ces catégories et la mesure dans laquelle elles entrent dans le cadre de l'activité de composition sont moins marquées : la notion d'instrument, ainsi que celles de notation et d'interprétation, sont réévaluées dans la mesure ou elles correspondent à des domaines désormais susceptibles d'être soumis aux processus de composition. La partition, en tant que support central, fédérateur des différents domaines, a ainsi vocation à se doter d'une flexibilité nouvelle afin de permettre l'établissement de démarches de composition cohérentes et personnelles avec la synthèse et les techniques électroacoustiques.

Deuxième partie

Contexte et aspects théoriques (2) Composition Assistée par Ordinateur

Chapitre 4

Modélisation informatique pour la composition musicale

Dans ce chapitre, nous laissons un instant de côté la synthèse sonore pour nous intéresser à la *composition assistée par ordinateur* (CAO). Cette discipline est centrée sur les aspects compositionnels, donc a priori plutôt en amont des questions d'interprétation, de synthèse ou de rendu sonore en général. Elle met en avant la formalisation des processus de composition en relation avec les systèmes et paradigmes informatiques, principalement à l'aide de langages de programmation. Ces notions et principes seront détaillés dans un premier temps ; nous verrons qu'elles s'articulent autour du principe de modélisation.

Nous donnerons ensuite un aperçu de quelques précédents travaux réalisés dans le domaine de la CAO, et décrirons en particulier l'environnement OpenMusic. OpenMusic est un langage visuel basé sur les paradigmes de programmation fonctionnelle et orientée-objet dans lequel les compositeurs peuvent développer des processus et modèles compositionnels faisant appel à divers formalismes musicaux et calculatoires. Cet environnement est la base de nos travaux sur la synthèse et la représentation du son. Son fonctionnement et son architecture seront présentés, afin de faciliter la compréhension ultérieure de notre travail et de la façon dont il s'intègre dans ce contexte.

4.1 Composition assistée par ordinateur

4.1.1 Origines et principes

Parmi les différentes disciplines que compte l'informatique musicale, la CAO propose une approche centrée sur la formalisation des processus de composition musicale.

L'idée d'une conception formalisée des processus musicaux, descriptibles par des procédures logiques ou algorithmiques, existe depuis longtemps dans la théorie et la composition musicales, mais se retrouve particulièrement mise en avant par les courants musicaux du XX^{e} siècle. Les nouvelles possibilités offertes par l'informatique, qui s'est développée à

cette époque, en ont par ailleurs permis l'exploration et la réalisation grâce à des outils de représentation et de calcul de plus en plus performants.

Avant que les technologies numériques ne soient capables de synthétiser des signaux, la manipulation des structures musicales par des méthodes et formalismes calculatoires furent les premières applications d'informatique musicale. Les travaux de Lejaren Hiller sont certainement parmi les premiers pas significatifs dans ce sens. Sa *Suite Illiac* pour quatuor à cordes, réalisée avec Leonard Isaacson en 1956, est considérée comme la première pièce musicale écrite par ordinateur. Cette suite a été générée (composée ?) par un programme sélectionnant et combinant notes et accords à partir de règles et contraintes prédéfinies, issues notamment de la formalisation de règles classiques (d'harmonie, en particulier) et d'expérimentations avec des modèles mathématiques.

Avec les débuts de la synthèse numérique, la plupart des travaux en informatique musicale, menés principalement aux Etats-Unis dans les années 1970-80, s'orienteront vers la synthèse des sons. C'est en Europe que seront alors soutenus les efforts axés sur la modélisation des processus musicaux, notamment par Pierre Barbaud [Barbaud, 1968], Iannis Xenakis [Xenakis, 1981], André Riotte [Riotte et Mesnage, 2006], et bien d'autres. Depuis cette époque, chercheurs et compositeurs (notamment à l'IRCAM) se sont ainsi attachés à bâtir les bases théoriques de la CAO, mettant l'accent sur une étude formelle de la musique en relation aux technologies et formalismes informatiques.

Les premières applications musicales constituèrent une approche généralement qualifiée de *composition algorithmique*, en ce sens qu'il s'agissait de créer des programmes qui seraient ensuite capables de composer "seuls", suivant des procédures algorithmiques définies. La CAO telle que nous l'entendons actuellement se différenciera cependant de cette idée, qui sous-entendait la possibilité d'implémenter l'intégralité d'une composition dans un programme, pour favoriser une vision selon laquelle le processus de composition implique une interaction et une intervention constante de l'utilisateur dans la création et la mise en oeuvre des programmes. [Laske, 1981] est l'un des premiers articles suggérant cette distinction de la CAO par rapport à la composition algorithmique, et soulignant l'importance de l'interaction entre le compositeur et la machine.

> "Nous pouvons envisager l'interaction entre un compositeur et un programme informatique sur un axe allant d'un contrôle exclusivement manuel à un contrôle exercé par un algorithme. La zone de plus grand intérêt pour la théorie de la composition est la zone intermédiaire de cet axe, puisqu'elle permet une grande flexibilité d'approche. Les pouvoirs de l'intuition et du calcul peuvent être combinés.[1]" [Laske, 1981], cité dans [Malt, 2003].

[1] *"We may view composer-program interaction along a trajectory leading from purely manual control to control exercised by some compositional algorithm (composing machine). The zone of greatest interest for composition theory is the middle zone of the trajectory, since it allow a great flexibility of approach. The powers of intuition and machine computation may be combined."* [Laske, 1981]

Comme toute activité artistique, la composition musicale présente en effet la particularité d'impliquer des processus créatifs qui ne sont réellement ni formalisés ni compris. En conséquence, un système de composition développé sous la forme d'un programme qui produirait de la musique ne fait pas réellement sens. La CAO proposera donc plutôt des environnements *d'aide* à la composition, permettant aux compositeurs d'utiliser l'ordinateur pour expérimenter, formaliser tel ou tel champ compositionnel, mais rarement pour produire automatiquement l'intégralité d'une pièce musicale [Laurson, 1996]. Le problème est donc considéré dans une optique légèrement différente : un système de CAO se doit de permettre aux compositeurs de créer eux-mêmes des systèmes informatiques reflétant leurs propres idées, intuitions ou formalismes et leur permettant de résoudre par le calcul les problèmes correspondants [Assayag, 1998].

4.1.2 Objets et processus

La dualité entre objets et processus musicaux, dans l'écriture comme dans la représentation de la musique, est une idée récurrente dans la pensée musicale contemporaine. Gérard Grisey évoquait par exemple un tel rapprochement :

> "Objets et processus sont analogues. L'objet sonore est seulement un processus contracté, et le processus n'est autre qu'un objet dilaté. [...] Le processus rend perceptible ce que nous cache la rapidité de l'objet : son dynamisme interne.[2]" [Grisey, 1987]

Cette idée est fondamentale dans l'approche proposée par la CAO. Comme nous venons de le voir, celle-ci vise en effet la conception d'environnements informatiques permettant la formulation, la représentation et le calcul de structures musicales, mettant l'accent sur la création des processus conduisant à ces structures.

Selon la terminologie utilisée dans [Girard *et al.*, 1989] (dans un tout autre contexte), nous abordons ici un problème de dualité entre sens et dénotation d'un programme, la dénotation s'approchant de la notion de résultat alors que le sens aurait plutôt trait à la façon d'obtenir ce résultat. Traditionnellement, la sémantique a favorisé le côté opératoire en négligeant le côté syntaxique des programmes : la sémantique d'un programme est déterminée par son résultat (c'est l'approche de la composition algorithmique que nous avons évoquée précédemment). Dans le cas de processus créatifs, cependant, l'absence d'un résultat "correct" (dénotation) favorise une focalisation sur le processus lui-même (sens). La conception du processus, dans ce contexte, se trouve porteuse d'un sens musical important. Si un objet, une forme musicale en tant que résultat d'un calcul, dénote une pièce, le processus conduisant à la création de cet objet lui donne un sens.

---

[2] *"Object and process are analogous. The sound object is only a process that can be contracted, the process is nothing more than a dilated sound object. [...] The process makes perceptible what the rapidity of the object hides from us : its internal dynamism."* [Grisey, 1987]

Favorisant la considération du processus (créatif) plus que du résultat, nous essaierons de montrer que l'approche de la CAO met ainsi particulièrement en avant la notion de *modèle*.

4.1.3 Modèles et représentations informatiques

Avant d'aller plus loin dans la CAO, il est nécessaire de revenir un instant sur la notion de modélisation, que nous avons évoquée quelques fois dans les chapitres précédents. Les définitions utilisées sont pour la plupart issues de [Berthier, 2002].

Par modéliser, nous entendons considérer un objet à travers un ensemble d'éléments (concrets ou abstraits) se référant à cet objet. Modéliser, c'est donc introduire un objet dans un champ de discours. Il pourra s'agir d'un objet mental, matériel, d'un phénomène, d'une situation, d'une activité ou de tout autre concept que le modèle vise à comprendre, étudier, expliciter. L'objet de la modélisation est donc avant tout un objet intentionnel, qui n'est pas nécessairement donné ou constitué a priori.

> "L'activité de modélisation a pour but de rassembler en un discours cohérent un certain nombre d'expériences ou d'observations considérées comme liées entre elles d'une manière qui reste à éclaircir ou à déterminer au cours du processus même de modélisation." [Berthier, 2002]

Un modèle est par ailleurs une formalisation de certains aspects choisis de son objet. Modéliser consiste ainsi à poser les termes et les structures à travers lesquels on interprétera les observations sur l'objet, ce qui confère un fort caractère partiel à la visée modélisatrice. [Berthier, 2002] parle de "grille de lecture", ou de "filtre cognitif".

Les éléments prenant part au processus de modélisation d'un objet sont ce que nous appellerons les *composants* d'un modèle. Leur organisation et relations formelles constitueront la *structure* du modèle (voir figure 4.1).

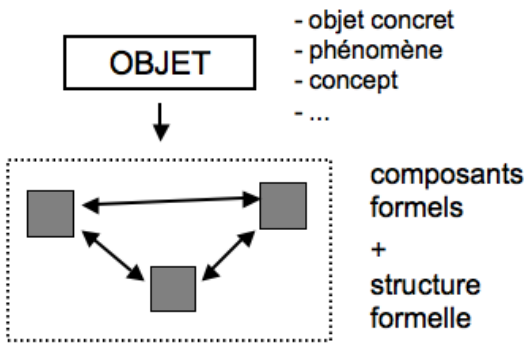


FIGURE 4.1: Modélisation d'un objet : composants et structure.

Un modèle reflète ainsi une certaine manière de concevoir un objet. Pour mettre en œuvre cette conception dans l'un des objectifs fixés par le modèle (simulation, expérimentation, création, etc.), le processus de modélisation fait appel à des *représentations*, éléments

opératoires utilisés comme substituts aux composants et structures mis en jeu dans la modélisation. Nous avons évoqué précédemment (dans le chapitre 2) le rôle déterminant des représentations quant à leur capacité à orienter les applications (musicales) réalisables dans un système.

> "[La représentation d'un modèle est] une expression de ce modèle (une manière de l'exprimer, peut être partiellement) qui introduit des éléments supplémentaires [...] qui supportent des modes opératoires effectifs [...] et destinés à faciliter le traitement pragmatique des questions qui ont conduit à construire le modèle." [Berthier, 2002]

Par la possibilité qu'il offre de créer des représentations symboliques et de traiter ces représentations par le calcul, l'ordinateur se présente ainsi comme un outil privilégié pour la modélisation. S'agissant de structures de données, de classes, de fonctions, de spécifications mathématiques ou vectorielles, de processus ou de règles de calcul, les représentations des composants et structures constituant les modèles, principaux acteurs dans les programmes et environnements informatiques, informent sur des aspects du monde réel et permettent d'interagir avec celui-ci au cours du processus de modélisation.

C'est donc à travers cette notion de modèle que la question de la composition musicale est abordée implicitement par la CAO.

4.1.4 Modèles compositionnels

[Malt, 2000] propose une analyse intéressante des processus de composition musicale dans le cadre de la composition assistée par ordinateur. L'auteur y décrit les principales phases de ces processus, principalement avec la phase conceptuelle, durant laquelle sont établies les grandes lignes et principes directeurs d'une composition, et la phase d'écriture durant laquelle les concepts prennent forme, sont présentés (actualisés) sur un support.

Ces deux phases ne sont pas réalisées séquentiellement mais en concurrence : il y a tout au long de la composition d'une œuvre des mouvements et une circulation de pensée entre le domaine des concepts et celui de l'écriture, jusqu'à atteindre une forme d'équilibre ou l'écriture se conforme aux concepts, et où les concepts se conforment aux contraintes de l'écriture. Le lien permettant d'actualiser le monde conceptuel, en même temps qu'il le façonne au cours du processus de composition, est assuré par les *modèles*.

La notion de modèle est ainsi décrite comme une représentation conceptuelle constituant le lien entre l'abstrait et le monde réel (voir aussi [Malt, 2003]). Le modèle permet de développer les concepts, mais également de les fixer sous une forme plus ou moins tangible. Il est la concrétisation, à la fois en tant que *figuration* et que *schéma directeur* des idées musicales du compositeur. Nous utiliserons le terme de *modèle compositionnel* pour désigner un tel modèle, relatif à un processus de composition.

4.1.5 CAO et Modélisation

Au sens que lui donne la CAO, la notion de modèle compositionnel fusionne avec celle du modèle informatique [Assayag, 1993]. Il est vu comme un ensemble de programmes, de données musicales ou extra-musicales, réalisant une idée musicale. De ce point de vue, un environnement de composition par ordinateur peut être considéré comme ce que [Malt, 2000] qualifie d'"espace de modélisation". Il doit permettre le développement et la manipulation des modèles, par l'expérimentation sur les données et processus, et par des allées et venues entre les domaines conceptuel et réel. Les techniques et formalismes informatiques et de programmation permettront au compositeur d'évoluer dans cet espace de modélisation :

> "L'objectif général est la définition de modèles informatisés utilisables dans des situations où le compositeur désire préparer des matériaux musicaux complexes et structurés, relativement à une certaine formalisation ou un certain ensemble de contraintes qui lui sont propres et qu'il est en mesure d'exprimer de façon cohérente."
> [Assayag et Cholleton, 1994]

La démarche de modélisation, à travers laquelle le compositeur développe et actualise des idées, prend ainsi une place significative dans le processus de composition. Avec la CAO, les modèles sont créés, corrigés, affinés par le compositeur durant la recherche musicale qui constitue la composition d'une pièce. Le modèle est ainsi observé, stimulé, et prend forme progressivement dans une démarche expérimentale autour d'une idée, d'un concept, d'une composition. Il est donc susceptible d'évoluer, d'être ajusté et complété au fur et à mesure de cette expérimentation [Assayag, 1998]. On retrouve cette idée dans la discussion proposée par [Berthier, 2002], qui précise que dans un processus de modélisation, ni les composants ni l'objet du modèle ne sont forcément donnés d'emblée, mais *se constituent conjointement avec le modèle.*

> "Le modèle (en cours de construction) vise un objet (en cours de définition) supposé commun à un ensemble (en cours d'affinement) de certains types d'expériences ou d'observations." [Berthier, 2002]

La création d'un modèle pourrait ainsi constituer le cœur du processus de composition. Dès lors qu'il est réalisé avec un ordinateur, les programmes informatiques deviennent donc un moyen privilégié par lequel les modèles sont représentés et mis en pratique dans ce processus.

[Malt, 2000] décrit par ailleurs la représentation des modèles compositionnels comme étant fondamentalement liée à une pensée musicale ; son choix étant déterminant dans le travail d'un compositeur. Nous avons en effet mis en avant plus haut le caractère partiel de l'activité de modélisation ; un même objet pouvant être modélisé de différentes manières, correspondant à autant d'approches subjectives de cet objet.

4.2 Langages de programmation

Nous avons vu que l'informatique et la programmation pouvaient être des outils privilégiés pour le développement et l'expérimentation sur les modèles. Le compositeur, utilisateur d'un système de CAO, se sert de l'ordinateur pour développer un modèle, actualiser une idée musicale sous la forme d'un programme. A ce titre, il doit être considéré comme un programmeur plutôt que comme un simple utilisateur de programmes. Les environnements de CAO, dans cette optique, sont ainsi généralement proposés sous forme de langages de programmation.

> "Nous concevons un tel environnement [de CAO] comme un langage de programmation spécialisé que les compositeurs utiliseront pour construire leur propre univers musical. [...] Ceci nous amène à réfléchir aux différents modèles de programmation existants, aux représentations, interne et externe, des structures musicales qui seront construites et transformées par cette programmation." [Assayag, 1998]

4.2.1 Programmation et composition

MUSICOMP [Hiller, 1969] est considéré comme l'un des premiers langages informatiques dédiés à la composition musicale. Il s'agit d'un langage dans le sens où ce n'est pas un programme en charge de produire une pièce mais un système permettant à son utilisateur de spécifier des règles pour la production de cette pièce, c'est-à-dire une interface permettant à celui-ci de formuler des instructions destinées à la machine. De nombreux autres environnements de CAO ont été développés par la suite, sous la forme de langages de programmation spécialisés.

Les mécanismes d'abstraction et d'application des langages fonctionnels permettent notamment d'utiliser les données comme source de programmes, qui génèrent à leur tour des données, et de simuler la plupart des notions de programmation (structures de données, structures de contrôle, récursivité, etc.) Ils peuvent ainsi être mis en œuvre de manière à centrer l'interaction dans un système (de composition) sur l'écriture des programmes [Orlarey *et al.*, 1997]. Le langage LISP [Steele, 1990] a souvent été utilisé comme base des environnements de CAO. Son caractère interprété en permet l'extensibilité et l'utilisation interactive, et sa relative simplicité syntaxique, son expressivité et sa puissance dans la manipulation des structures symboliques permettent d'effectuer des opérations avancées sur le matériau musical (voir par exemple [Dannenberg *et al.*, 1997]).

D'autres paradigmes de programmation se sont également prêtés à des applications pour la composition musicale, en particulier la programmation par objets ([Pope, 1991] donne un ensemble d'exemples), ou encore la programmation déclarative et les langages par contraintes (voir par exemple [Rueda *et al.*, 2001]).

Une série d'environnements de composition dédiés à la formulation de situations et au calcul des structures musicales se sont succédés à l'IRCAM. Parmi ceux-ci, on pourra citer FORMES, système orienté-objet permettant la création et l'organisation de processus de

synthèse sonore [Rodet et Cointe, 1984], ou CRIME, environnement pour l'élaboration de structures musicales symboliques [Assayag *et al.*, 1985], et surtout PATCHWORK [Laurson et Duthen, 1989], qui introduisit un changement radical dans le domaine avec la notion de programmation visuelle.

COMMON MUSIC [Taube, 1991] est un autre langage de programmation dédié à la composition. Basé lui aussi sur le langage LISP, il permet également la construction de séquences musicales à l'aide des outils de programmation offerts par le langage, interprétables pour différents dispositifs ou programmes de rendu sonore (port MIDI, programmes de synthèse, etc.)

4.2.2 Programmation visuelle

Le pouvoir d'expression offert par les langages de programmation est donc mis à disposition des compositeurs par les environnements de CAO. Cela dit, ce pouvoir d'expression est également assujetti à certaines compétences en programmation. Les interfaces homme-machine sont le moyen par lequel l'utilisateur interagit avec le système, et sont certainement la clé de nombreux problèmes concernant cet accès aux ressources de l'ordinateur pour un utilisateur, voire pour un programmeur. Les travaux menés autour des environnements de programmation visuelle vont dans ce sens.

On parle de programmation visuelle dès lors qu'un langage propose une sémantique à plus de une dimension, c'est-à-dire lorsqu'il ne s'agit pas (syntaxiquement) de texte purement linéaire mais que d'autres indications (visuelles) entrent en compte dans cette sémantique.

Les environnements de programmation visuelle pour la composition améliorent l'accessibilité, et par conséquent (dans notre cas) le potentiel créatif des langages de programmation. Ils permettent au compositeur de programmer, de visualiser, d'éditer, de maintenir et réutiliser des processus et des structures de données par l'intermédiaire d'interfaces plus conviviales et relativement simples syntaxiquement.

ELODY [Orlarey *et al.*, 1997] est un exemple d'environnement pour la composition basé sur un langage de programmation visuel, dérivé du Lambda Calcul [Barendregt, 1984]. Il permet la création d'objets musicaux par la combinaison de données simples et de règles de construction élémentaires. Les règles d'abstraction et d'application permettent de rendre variables certains éléments des objets, conduisant à la définition de fonctions applicables sur d'autres objets. La figure 4.2 montre un exemple de création et d'application d'une telle fonction avec ELODY.

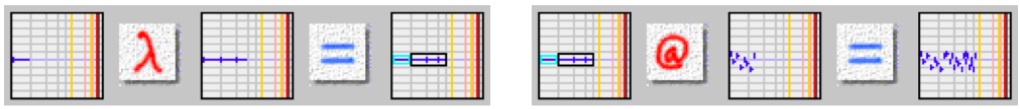


FIGURE 4.2: Programmation visuelle dans l'environnement de composition ELODY : (a) Création de la fonction *triple* à partir d'une séquence de notes particulière et du constructeur lambda (abstraction). L'objet *note* est rendu variable dans la séquence. (b) Application de la fonction *triple* créée à un autre objet.[3]

Comme nous l'avons vu avec les langages visuels pour la synthèse sonore en temps réel dans le chapitre 2, les graphes (composants graphiques interconnectés) constituent une manière commode et intuitive de représenter visuellement un processus. PATCHWORK [Laurson et Duthen, 1989] et ses successeurs directs PWGL [Laurson et Kuuskankare, 2002] et OPENMUSIC [Agon, 1998] utilisent ce même type de représentation. Il s'agit de langages fonctionnels construits sur la base du langage LISP, qui en constituent des extensions à la fois visuelles et musicales. Ceux-ci sont parmi les langages visuels les plus complets existant à l'heure actuelle dans le domaine. La figure 4.3 montre un programme visuel (ou *patch*) dans le langage PATCHWORK.

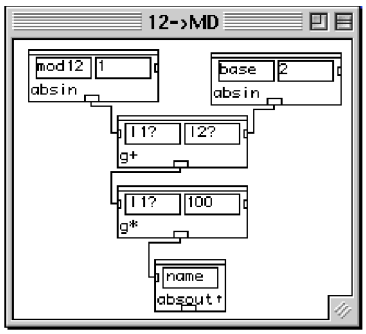


FIGURE 4.3: Un *patch* (programme visuel) dans PATCHWORK.

Il est important de distinguer ce type de langage de programmation des environnements de synthèse du type MAX/MSP qui, s'ils présentent une interface comparable, répondent à des paradigmes très différents. Dans les langages de type PATCHWORK les programmes visuels correspondent à des expressions fonctionnelles, équivalentes à des programmes écrits en LISP (on parle de *s-expressions*). Une telle expression est évaluée pour produire un résultat par réductions successives suivant le flux de contrôle défini par les connexions du programme visuel (on parle de *control flow*, par opposition au *data flow* des environnements de temps réel). Plus de précisions seront données dans la section suivante, dédiée au langage OPENMUSIC.

[3]Illustrations extraites de [Orlarey *et al.*, 1997].

4.3 OpenMusic : un langage de programmation visuelle pour la composition musicale

OpenMusic [Agon, 1998] [Assayag *et al.*, 1997a] est l'environnement de CAO actuellement développé par l'équipe Représentations Musicales de l'Ircam. Conçu par Gérard Assayag et Carlos Agon dans le milieu des années 1990, il hérite d'une grande partie des concepts de PatchWork (programmation visuelle, fonctionnalités musicales, etc.)

OpenMusic est un langage de programmation complet, étendant le langage Lisp et doté d'une spécification visuelle. Le paradigme dominant y est donc celui de la programmation fonctionnelle.

A l'aide d'outils graphiques proposés par OpenMusic, l'utilisateur peut créer des fonctions et des programmes faisant appel à des opérations arithmétiques ou logiques et à des notions informatiques comme l'abstraction, l'itération, ou encore la récursivité. Il peut par ailleurs tirer profit des potentialités de la programmation orientée objet de CLOS (*Common Lisp Object System* [Gabriel *et al.*, 1991]) en créant des classes, des relations d'héritage, des méthodes, ou même aller plus loin avec des outils pour la programmation par meta-objets, la programmation par contraintes, etc.

4.3.1 Architecture du langage

Les éléments du langage peuvent être distingués selon deux catégories : les *meta-objets* sont les primitives du langage : fonctions, programmes (patches), classes, instances, types, etc. ; et les *composants graphiques* assurent la représentation graphique et les interactions de l'utilisateur avec ces éléments.

Dans CLOS, le protocole de meta-objets (*Meta-Object Protocol* – MOP) établit les éléments du langage objet (classes, fonctions génériques, méthodes, etc.) comme étant elles-mêmes des classes de meta-objets, instances de la classe `standardclass` [Kiczales *et al.*, 1991]. Celles-ci (respectivement `standardclass`, `standardgenericfunction`, `standardmethod`, etc.) peuvent ainsi être sous-classées et étendues par de nouvelles classes. C'est ce que fait OpenMusic, avec `OMClass`, `OMGenericFunction`, `OMMethod`, etc. `OMClass`, par exemple, est définie comme sous classe de `standardclass` et étend ainsi cette classe par différents comportements et propriétés liés aux aspects visuels du langage (identification d'icônes, documentation, persistance, comportement dans l'interface graphique, etc.) Toutes les classes créées dans OpenMusic pourront alors être définies comme étant des `OMClass` et non des `standardclass`, et s'intègreront ainsi dans le langage visuel. Le même principe est utilisé pour les fonctions génériques et les méthodes.

Les meta-objets créés dans OpenMusic sont donc dotés d'attributs et comportements relevant de l'aspect visuel du langage. Ceux-ci se manifestent principalement à travers les composants graphiques, notamment avec les *boîtes* (classe `OMBox`) et les *éditeurs* (classe `Editor`). Ceux-ci sont cependant des objets dits "non graphiques" dans l'environnement,

le niveau graphique à proprement parler (éléments de l'interface) recouvrant ce premier niveau par le biais de la classe `OMFrame` et de ses sous-classes (`BoxFrame`, `InputFrame`, `EditorFrame`, etc.)

Une boîte est donc une spécification syntaxique du langage visuel. Elle fait référence à un autre élément (par exemple, à une fonction) et représente sa manifestation dans le langage visuel. Elle a la propriété d'avoir des entrées et sorties, dépendant de l'objet auquel elle fait référence, et ainsi de pouvoir être connectée à d'autres boîtes à l'intérieur d'un programme. La plupart des objets de l'environnement sont également associés à un éditeur, permettant leur construction et leur édition "manuelles".

Les aspects visuels sont donc étroitement intégrés dans l'architecture du langage : il ne s'agit pas d'une simple interface sur celui-ci mais de propriétés des meta-objet, et de composants graphiques faisant partie de la spécification même du langage.[4] Une description plus détaillée de cette architecture est donnée dans [Agon, 1998].

4.3.2 Environnement de travail

L'environnement général de OpenMusic est semblable au "bureau" que l'on trouve dans la majorité des systèmes d'exploitation actuels. Dans cet espace, appelé *workspace*, l'utilisateur se crée un environnement de travail en organisant programmes, objets, librairies, et autres composants du système. La figure 4.4 montre un exemple de *workspace*.

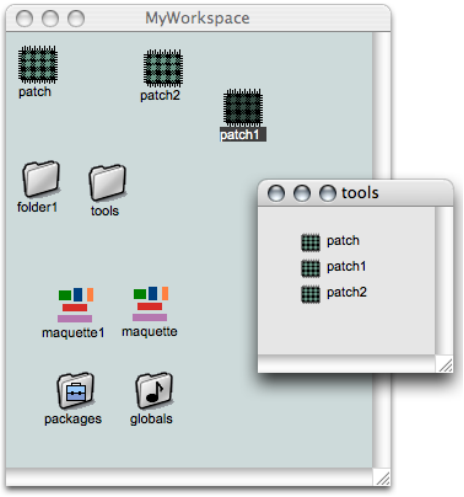


Figure 4.4: Un *workspace* OpenMusic.

Les icônes disposées sur le workspace représentent des instances de ces différents composants : *patches*, dossiers, *maquettes*. Leur organisation reflète une organisation réelle de fichiers sur le disque dur : il s'agit d'objets de l'environnement dits "persistants". Les

[4] L'intérêt d'une telle intégration du visuel au niveau de la construction formelle du système est discutée par exemple dans [Balaban et Elhadad, 1999].

patches représentent des programmes et sont les principaux points d'entrée dans le domaine la programmation.

4.3.3 Patches : programmes visuels

Un patch est un programme visuel. Il est associé à un éditeur, dans lequel l'utilisateur (compositeur/programmeur) dispose des unités fonctionnelles sous forme de boîtes graphiques, présentant chacune un certain nombre d'entrées et de sorties. Un ensemble de boîtes reliées les unes aux autres par l'intermédiaire de connexions établies entre ces entrées et sorties compose ainsi un programme. La figure 4.5 représente un patch réalisant des opérations arithmétiques sur des entiers.

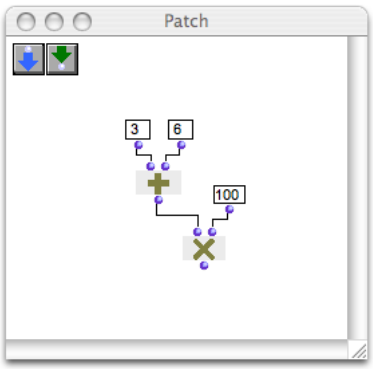


FIGURE 4.5: Un *patch* implémentant l'opération arithmétique $(3 + 6) \times 100$.

Chaque boîte fait référence à un objet fonctionnel : dans l'exemple de la figure 4.5, les boîtes font référence à des fonctions ($+$ et $\times$), ou à des valeurs constantes (3, 6 et 100). Les fonctions sont définies dans l'environnement ; il peut s'agir de primitives du langage Common LISP, de fonctions prédéfinies dans OPENMUSIC, ou des fonctions créées par l'utilisateur. Les entrées des fonctions (et des boîtes en général) sont situées en haut de celles-ci, et leurs sorties en bas. Les données dans un patch circulent donc toujours globalement de haut en bas.

Les connexions entre les entrées de boîtes et les sorties d'autres boîtes définissent la composition du programme représenté dans un patch. Boîtes et connexions constituent un graphe acyclique définissant la sémantique de ce programme.

Ce graphe est équivalent à une expression LISP, et peut être, suivant le paradigme de ce langage, évalué en n'importe quel point. Le déclenchement de l'*évaluation* d'une boîte, par une action de l'utilisateur, consiste en l'appel à la fonction correspondante avec pour arguments les résultats des évaluations des boîtes connectées à ses entrées. Une série d'évaluations se produit alors récursivement pour réduire cette expression suivant la composition du programme, ce qui correspond finalement à l'exécution de ce programme.

Le patch représenté sur la figure 4.5 correspond donc à l'expression LISP suivante :

```
(* (+ 3 6) 100)
```

L'évaluation de la boîte × réduit donc cette expression : le résultat de la boîte + est multiplié à celui de la boîte 100, et ainsi de suite. Le *listener*[5] affiche le résultat :

```
>> 900
```

Ainsi, si nous avons vu que les données circulent de haut en bas, nous voyons que le sens de l'éxecution de ce programme est quant à lui globalement orienté du bas vers le haut. Cette caractéristique due au paradigme fonctionnel sous-jacent est une différence fondamentale par rapport aux langages temps réel comme PureData ou Max/MSP (voir chapitre 2, section 2.3.4) ; la relation entre données et processus est en effet inversée dans ces deux types d'environnements [Puckette, 2004].

4.3.4 Fonctionnalités de programmation avancées

Structures de contrôle

Diverses structures de contrôle communément employées dans la programmation (conditionnelles, itérations, etc.) sont implémentées dans le langage visuel. La figure 4.6 montre un exemple de leur utilisation dans un patch : une boîte `omloop` (représentant une itération) est présente sur la gauche, associée elle aussi à un éditeur permettant de définir le comportement du programme dans cette itération.

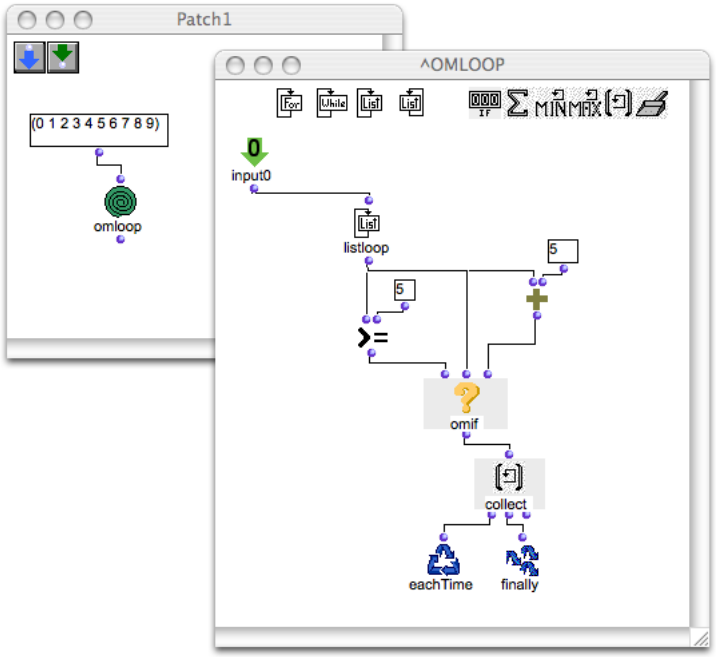


FIGURE 4.6: Structure de contrôle `omloop`.

[5]En Lisp et dans les langages interprétés du même type, le *listener* est une fenêtre dans laquelle peuvent être entrées et évaluées des expressions, et qui affiche les résultats.

Dans cet exemple l'itération est opérée sur la liste donnée en entrée de la boucle par l'intermédiaire de l'itérateur `list-loop` (différents itérateurs sont disponibles : `on-list`, `for`, `while`, etc.) A chaque étape de l'itération, donc pour chaque élément de la liste, intervient une autre structure de contrôle (la structure conditionnelle `omif`) pour incrémenter la valeur de celui-ci si elle est inférieure à un seuil donné. Les résultats successifs sont collectés dans une nouvelle liste qui est renvoyée comme résultat de l'itération. Ce patch correspond donc à l'expression LISP suivante :

```
(loop for x in '(1 2 3 4 5 6 7 8 9) collect (if (>= x 5) x (+ x 5)))
```

Le résultat affiché si l'on évalue la boîte `omloop` est donc :

```
>> (5 6 7 8 9 5 6 7 8 9)
```

Abstraction

Dans les langages à fermeture comme LISP, issus du formalisme du Lambda Calcul, données et programmes sont des structures équivalentes. L'*abstraction fonctionnelle* consiste ainsi à faire devenir variables certains éléments d'une structure statique, ce qui conduit à la définition d'objets fonctionnels.

Dans OpenMusic, des entrées et des sorties, également représentées par des boîtes, peuvent être introduites dans les patches. Elles vont permettre de rendre variables certaines données du programme défini dans le patch.

A partir de l'exemple de la figure 4.5, il est donc possible par exemple de créer la fonction $f(x) = (x+6)*100$, en ajoutant une entrée et une sortie connectées au programme comme le montre *patch1* sur la figure 4.7 - a.

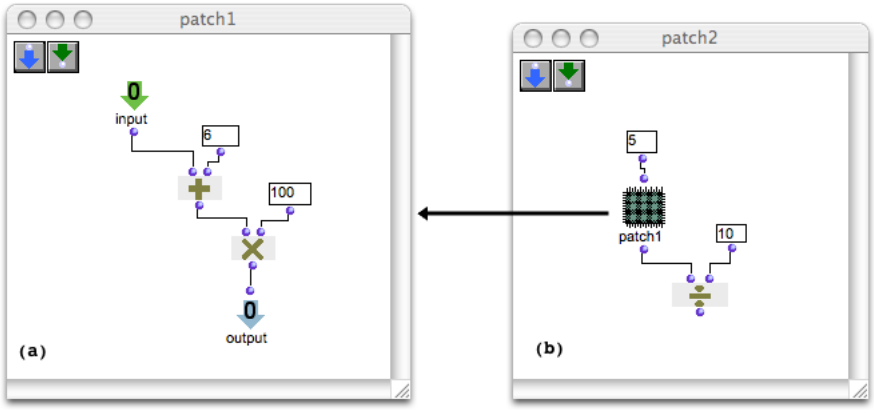


FIGURE 4.7: (a) Définition et (b) application d'une fonction dans un patch. On réalise une abstraction en rendant variables certains paramètres du programme. L'abstraction représentée par la boîte *patch1* dans le patch *patch2* reste liée et permet l'accès et l'édition du programme original.[6]

Cette fois, le patch tout entier correspond à la définition d'une fonction. L'expression Lisp correspondante serait la suivante :

```
(lambda (x) (* (+ x 6) 100))
```

Sous forme de définition fonctionnelle, on peut également écrire cette expression de la manière suivante :

```
(defun myfunction (x) (* (+ x 6) 100))
```

Un patch peut donc de cette manière être utilisé dans un autre patch. Il est alors vu comme une fonction à n entrée et m sorties, comme illustré dans *patch2* sur la figure 4.7 - b. A l'intérieur de ce patch pourront être assignées des valeurs aux variables de l'abstraction (*application fonctionnelle*).

Dans cet exemple, un nouveau type de boîte apparaît, faisant référence au patch (la boîte appelée *patch1*). Son évaluation correspond à l'expression Lisp :

```
(funcall (lambda (x) (* (+ x 6) 100)) 5)
```

ou, de façon équivalente :

```
(myfunction 5)
```

Le nouveau programme (*patch2*) correspond donc à l'expression :

```
(/ (myfunction 5) 10)
```

A l'intérieur d'un patch, le programme peut donc être modifié et partiellement exécuté. De l'extérieur, dans un autre programme, il est une boîte correspondant à une fonction abstraite. Une abstraction définit ainsi une unité générique de calcul ou de traitement (une fonction), susceptible d'être utilisée dans différents contextes. On retrouve là une pratique répandue en programmation, ainsi que dans la composition musicale : un procédé développé par un compositeur dans le cadre d'une pièce peut ainsi être réutilisé dans une autre, ou dans une autre partie de cette pièce, sans nécessiter une nouvelle implémentation (de même que la définition de fonctions permet leur réutilisation dans les programmes). Les multiples occurrences d'un patch dans d'autres programmes feront donc toutes référence au même programme initial.

L'abstraction permet ainsi d'organiser les niveaux structurels d'un processus compositionnel et de travailler et progresser dans la complexité de ces processus.

Une abstraction peut également être appelée dans la propre fonction qu'elle définit, implémentant le principe de la récursivité.

---

[6] Dans la suite, nous symboliserons toujours la présence de l'éditeur, ou du contenu d'un objet, par une flêche comme celle visible sur cette figure. Ici : "la fenêtre *patch1* à gauche correspond au contenu de la boîte *patch1* visible à droite."

Fonctions *lambda*

Nous avons dit plus haut qu'en LISP une fonction pouvait être considérée au même titre qu'une donnée dans le calcul, c'est-à-dire être inspectée, ou même construite dans ce calcul. Cette caractéristique donne au langage un pouvoir d'expression étendu, notamment par la possibilité de définir naturellement des fonctions d'*ordres supérieurs*, c'est-à-dire prenant d'autres fonctions comme arguments, ou produisant des fonctions en tant que résultats.

Dans OPENMUSIC, un patch utilisé dans un autre peut être mis par l'utilisateur en mode "lambda", et alors être vu dans le programme (d'ordre supérieur) qui le contient non plus comme un appel fonctionnel au programme qu'il représente, mais comme un "objet" fonctionnel.

Un patch comme *patch1* dans la figure 4.7 - b, par exemple, s'il est évalué en mode "lambda", ne produira plus comme résultat une valeur (`1100` dans notre exemple), mais une fonction `(lambda (x) (* (+ x 6) 100))` qui pourra être utilisée comme telle dans la suite du programme.

La curryfication (transformation d'une fonction à n arguments en une fonction à $n-1$ arguments) peut également être réalisée grâce à ce mode "lambda", en affectant explicitement (par des connexions) des valeurs à certaines entrées de la fonction.

Fonctions locales

Complémentairement à la démarche d'abstraction que nous venons de voir, il est possible dans un patch OPENMUSIC de créer un sous-programme (sous-patch) qui est une fonction locale, définie à l'intérieur de ce patch uniquement.

Les utilisateurs de OPENMUSIC font la distinction entre les patches "rouges" qui représentent ces fonctions locales, et les patches "bleus" qui sont les abstractions fonctionnelles globales discutées précédemment.

Ainsi, l'exemple de la figure 4.7, correspondait aux déclarations :

```
; patch1
(defun myfunction (x) (* (+ x 6) 100))

; patch2
(defun myprogram (x)  (/ (myfunction x) 10))
```

Ici, `myfunction` est définie dans l'environnement, alors qu'avec une fonction locale (patch "rouge"), on aurait eu l'équivalent de :

```
(defun myprogram (x)
  (let ((myfunction (lambda (x) (* (+ x 6) 100))))
    (/ (funcall myfunction x) 10)))
```

La fonction `myfunction` n'existe alors pas en dehors de `myprogram`. Ce mécanisme, moins modulaire que celui des abstractions globales, permet cependant d'encapsuler les

programmes dans une organisation hiérarchique et compacte, utile et parfois nécessaire, notamment pour la lisibilité des processus. Il est également possible de détacher les patches de leur abstraction (créer une fonction locale à partir d'une abstraction donnée) ou inversement de définir une abstraction à partir d'une fonction locale.

Programmation par objets

OpenMusic est également un environnement "orienté-objet", basé sur la couche objet du Lisp, CLOS (*Common Lisp Object System* [Gabriel *et al.*, 1991]). CLOS propose une intégration puissante et élégante de la programmation par objet et de la programmation fonctionnelle. Cette intégration se retrouve dans les possibilités de programmation offertes à l'utilisateur. Le protocole de meta-objets (MOP) permet la réflexivité du langage (voir la discussion précédente sur les meta-objets dans OpenMusic), c'est-à-dire que les éléments qui constituent ce langage peuvent devenir les objets de traitement des programmes [Agon et Assayag, 2003]. En d'autres termes, les classes, fonctions, méthodes qui constituent un programme peuvent être créées, inspectées et manipulées dynamiquement pendant l'exécution du programme.

Il est ainsi possible dans OpenMusic de définir de manière graphique de nouvelles classes, de paramétrer les *slots* de ces classes, et d'établir des relations d'héritage avec les classes existantes. Le polymorphisme des fonctions génériques dans CLOS est également intégré au langage visuel, avec la possibilité de créer des fonctions génériques et leurs différentes méthodes, ou d'ajouter des nouvelles méthodes à des fonctions génériques existantes, afin de les spécialiser pour de nouveaux types d'arguments.

Autres types de programmation

De nombreux travaux ont également été menés dans OpenMusic concernant la programmation par contraintes : différents moteurs de résolution de contraintes y ont été intégrés (Screamer [Siskind et McAllester, 1993], Situation [Bonnet et Rueda, 1998], OMClouds [Truchet *et al.*, 2003]).

[Agon, 2004] fait le point sur ces différents paradigmes de programmation et leurs applications musicales.

4.3.5 Intégration données/programmes

Objets et éditeurs

De nombreuses classes d'objets (note, accord, séquence, polyphonie, courbes, sons, etc.) sont prédéfinies dans OpenMusic et permettent de manipuler le matériau musical dans les programmes visuels à différentes échelles de précision et de complexité. La figure 4.8 montre un patch faisant appel à différentes de ces classes. Elle donne également une idée du type de processus musicaux pouvant être créés dans OpenMusic.

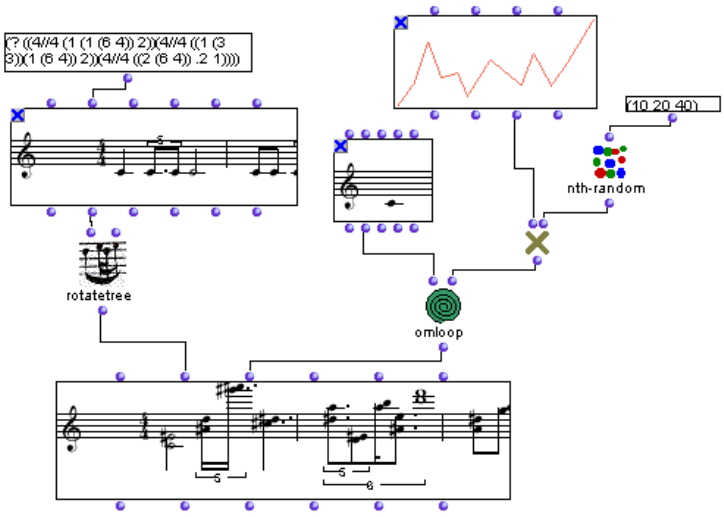


FIGURE 4.8: Un patch utilisant des objets musicaux. Dans cet exemple, une instance de la classe `voice`, représentant une séquence musicale en notation rythmique, est créée par le calcul d'un pattern rythmique d'une part (à gauche) et d'un profil mélodique d'autre part (à droite). La fonction `rotate-tree` applique une rotation circulaire à un arbre rythmique donné. Les valeurs en ordonnée de la fonction (`BPF`) visible en haut à droite sont multipliées par un facteur tiré aléatoirement, et traitées itérativement dans la boîte `omloop` afin de générer la séquence d'accords.

On trouve dans ce patch un nouveau type de boîtes, appelées *factories*, qui permettent de créer des instances d'une classe donnée, à laquelle elles font référence. Les entrées et sorties de ces boîtes représentent respectivement l'instance créée (`self`) et les accès en écriture (entrées) et lecture (sorties) sur les différents *slots* publics de cette classe (par exemple, liste des hauteurs, des *onsets*, des durées, etc. pour une séquence d'accord – classe `chord-seq`). Ces boîtes permettent donc d'instancier les objets correspondants en spécifiant ces valeurs, et/ou d'utiliser ces valeurs dans des processus et calculs ultérieurs.

Chaque *factory* est également associée à un éditeur, spécifique à la classe correspondante, et permettant d'éditer manuellement ou simplement de visualiser l'instance courante contenue dans cette boîte. OPENMUSIC propose donc divers éditeurs dédiés aux structures de données, les plus avancés étant les éditeurs de type partition (voir figure 4.9), mais aussi des éditeurs pour fichiers MIDI, audio, pour les courbes et *breakpoint functions* (BPF), etc.

Ces boîtes *factory* permettent donc de générer et/ou stocker et visualiser l'état d'un ensemble de données à un moment donné du calcul (i.e. à une position donnée dans le graphe défini dans le patch), et offrent en même temps un accès direct à ces données par le biais de l'éditeur [Agon et Assayag, 2002].

Une notion importante en CAO, que nous avons énoncée précédemment, est en effet la possibilité d'agir sur le matériau musical à la fois par des processus (ce qui permet

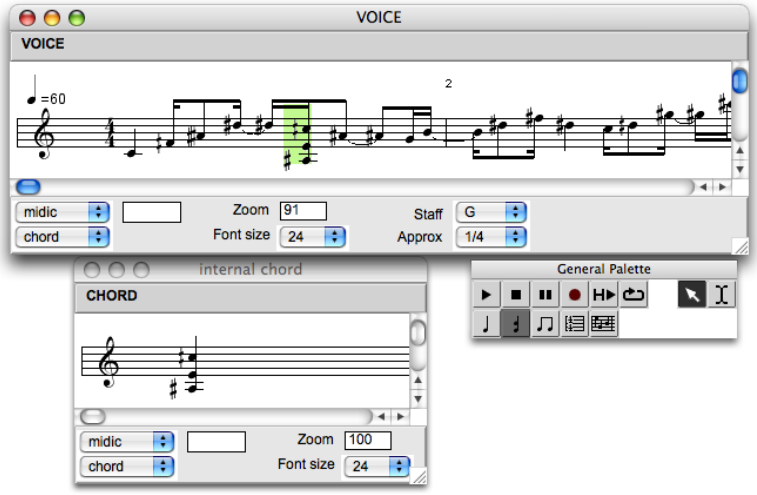


FIGURE 4.9: Editeur graphique d'un objet de type `voice`.

un contrôle plus puissant et un niveau d'organisation plus élevé qu'une édition directe des données) et manuellement (pour spécifier des données initiales, vérifier et corriger les données ou résultats intermédiaires, etc.)[7] L'intégration étroite des éditeurs graphiques dans le langage visuel répond à cette double nécessité ; la représentation conjointe des structures musicales et des processus dont il font partie (ou desquels ils sont issus) offre un contrôle étendu, à la fois sur le matériau brut et sur les processus musicaux.

L'idée initiale dans OPENMUSIC était celle d'une intégration de composants (données et applications) à l'intérieur du programme [Assayag et Agon, 1996]. Dans cet objectif, les *factories* constituent ainsi un point d'entrée privilégié pour l'introduction de données et l'interaction de l'utilisateur (compositeur–programmeur) dans le programme.

Les objets musicaux peuvent également être joués (dans les patchs ou par l'intermédiaire de leurs éditeurs respectifs), ce qui permet un accès et un contrôle supplémentaire sur les données et résultats des processus.[8]

Représentation du temps : maquette

Une des grandes nouveautés proposées dans OPENMUSIC par rapport aux générations précédentes d'environnements de CAO tient dans un objet appelé *maquette*, permettant de mettre en jeu le temps dans les processus de composition.

Une maquette est un document qui étend la notion de programme visuel d'une dimension temporelle et spatiale. Comme un patch, elle représente un espace à deux dimensions,

[7] Cette idée est également reprise dans [Roads, 1996].

[8] Les objets musicaux sont joués par l'intermédiaire d'un processus qui distribue ceux-ci sur différents *players* spécialisés (MIDI et audio, principalement). Ces mêmes *players* sont également sollicités lorsque l'on utilise les commandes de la palette flottante visible aux côtés de l'éditeur sur la figure 4.9.

mais dont l'axe horizontal correspond au temps, selon le modèle des séquenceurs musicaux classiques. La maquette est donc en mesure de "jouer" son contenu (i.e. les objets disposés et éventuellement calculés à l'intérieur). La figure 4.10 est un exemple de maquette.

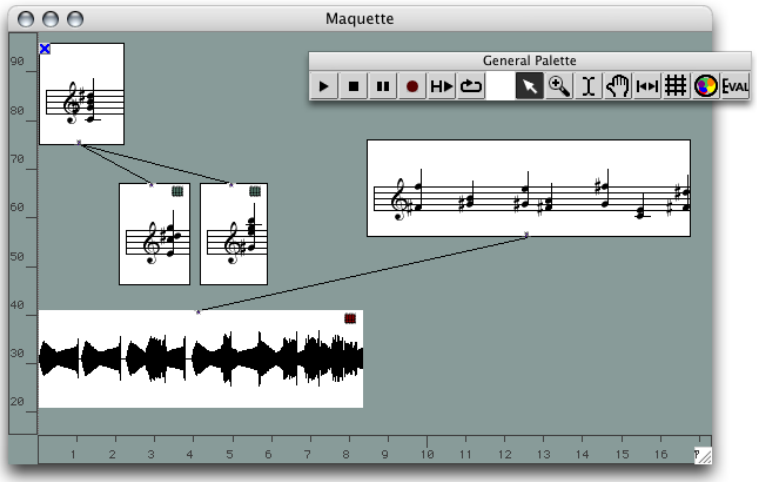


FIGURE 4.10: Une maquette dans OPENMUSIC.

Les objets contenus dans une maquette sont appelés *TemporalBoxes*. De manière générale, on peut considérer que chacun se réfère à un programme (patch) produisant un résultat musical visible dans le contexte global. Ces programmes peuvent notamment intégrer dans le calcul les caractéristiques temporelles et graphiques des *TemporalBoxes* auxquelles ils sont associés, et être liés les uns aux autres, ce qui permettra l'implémentation de relations à la fois temporelles et fonctionnelles entre les objets musicaux.

La maquette constitue ainsi une approche originale pour la composition musicale assistée par ordinateur, permettant de mettre les éléments de l'environnement de composition (structures de données et processus) en relation étroite avec les structures temporelles. Une description détaillée sera donnée dans le chapitre 11.

4.3.6 Applications musicales

Complémentairement aux outils de programmation, l'environnement OPENMUSIC fournit une bibliothèque importante de classes, de structures de données et de fonctions prédéfinies, permettant d'orienter la programmation dans des applications musicales et compositionnelles. Certaines, à vocation généraliste, sont intégrées dans les noyau du langage ; d'autres, plus spécifiques à des esthétiques ou formalismes particuliers, s'y ajoutent dynamiquement par le biais de bibliothèques externes. Le système modulaire des bibliothèques a ainsi permis au fil des années à différents chercheurs et compositeurs d'intégrer à cet environnement des classes et fonctions dédiées à des applications et esthétiques musicales particulières.

De nombreuses applications et œuvres musicales ont vu le jour, partiellement ou entièrement créées à l'aide de OpenMusic. *The OM Composer's Book* [Agon *et al.*, 2006] est un recueil de textes et de témoignages de compositeurs concernant l'utilisation de la CAO dans leur travail de composition. Chaque chapitre décrit, à travers une œuvre créée à l'aide de OpenMusic, la place et l'utilité de cet environnement dans la modélisation et la création musicale. Un deuxième volume, à paraître en 2007-2008, est actuellement en préparation [Bresson *et al.*, 2008].

4.4 Conclusion

Nous avons souhaité montrer dans ce chapitre la pertinence de l'approche proposée par la CAO, et en particulier de l'utilisation des langages de programmation pour une modélisation des processus liés à la composition musicale. A travers les recherches et applications autour des environnements de CAO et leur utilisation dans des projets compositionnels, l'expérience a en effet montré que le discours musical contemporain pouvait être mis, au moins partiellement, en relation avec un cheminement et une pensée informatiques. Au delà du simple développement processus informatiques, l'intérêt est donc de permettre la description et le développement de processus musicaux à l'aide de concepts et paradigmes informatiques, c'est-à-dire de mettre en correspondance pensée informatique et pensée musicale.

Le terme de composition assistée par ordinateur est peut-être donc source de confusion ; l'ordinateur est en effet aujourd'hui plus un outil qu'un assistant à la composition. Comme tout outil, en outre, il est susceptible d'influencer et d'orienter le travail du compositeur. Certains environnements favoriseront la mise en œuvre de certains types de processus, ou même plus généralement la création de certains modèles par rapport à d'autres, en fonction des outils de programmation proposés et des possibilités des langages.

L'environnement OpenMusic a donc été présenté dans ce contexte, comme un langage ouvert et puissant pour la programmation et la composition musicale. Pendant la durée de cette thèse au sein de l'équipe Représentations Musicales de l'Ircam, nous avons été très impliqués dans le développement de cet environnement. La version OpenMusic 5 [Bresson *et al.*, 2005a] a été créée dans un objectif d'ouverture sur de nouvelles plateformes[9] et applications.[10] Elle intègre également divers remaniements au niveau de l'organisation et des interfaces, ainsi que la plupart des travaux réalisés dans le cadre de cette thèse.

Le travail que nous avons réalisé se situe donc dans la continuité de l'approche de la CAO, reconsidérée dans le cadre des problématiques posées par la synthèse sonore. Comme

[9] L'architecture multi-plateforme est assurée par une API (*Application Programming Interface*), actuellement implémentée sur MacOSX, Windows XP, et Linux.

[10] Notamment pédagogiques, voir [Bresson *et al.*, 2006].

nous l'avons mentionné au début de ce chapitre, la CAO en général, et OpenMusic en particulier, se positionnent globalement (mais nous verrons qu'il existe des exceptions) en amont des questions de synthèse et de rendu sonore (OpenMusic est généralement utilisé pour l'écriture de parties ou de pièces instrumentales ; les parties de synthèse étant réalisées dans d'autres environnements). L'enjeu de ce travail sera donc d'étendre le champ de la CAO à ce domaine de la synthèse sonore, afin de traiter les différentes problématiques que nous avons entrevues dans les premiers chapitres.

Chapitre 5

Synthèse sonore et CAO : vers une modélisation compositionnelle

Après les notions présentées dans le chapitre précédent, nous souhaitons ici revenir vers les questions relatives à la synthèse sonore pour introduire une démarche visant à insérer celle-ci dans le cadre de la modélisation compositionnelle. Cette démarche va donc dans le sens d'un traitement de la question de la synthèse sonore à partir de considérations musicales de haut niveau inspirées des principes de la CAO.

Après une discussion sur certaines problématiques spécifiques à cette intégration de la synthèse sonore dans la composition musicale, et relatives cette fois aux environnements informatiques correspondants, quelques travaux existants dans le domaine seront présentés, ainsi que des développements préliminaires réalisés par le passé dans OpenMusic.

5.1 Problématiques et aspects théoriques

5.1.1 Représentations symboliques

L'écriture (musicale, en ce qui nous concerne) fait appel à des systèmes de représentation symboliques, et les représentations informatiques d'un système de composition n'échappent pas à cette nécessité.

Selon [Chazal, 1995], une première définition du symbole pourrait être un objet mis à la place d'un autre objet, ou d'une réalité quelconque. Le symbole réunit un signe et son sens, signifiant et signifié. Il a donc en soi une fonction de représentation, mais également une fonction pratique, permettant la mémorisation, la condensation, l'ordonnancement de l'information. De par leur fonction, les symboles doivent être reconnus et interprétés dans leurs différentes occurrences : c'est par l'interprétation des symboles que l'on est capable d'appréhender le sens d'une représentation.

Comme le signalent [Malt, 2000] ou [Manoury, 1990] l'abstraction de la réalité permise par les symboles, dégageant les objets de leur contexte, est fondamentale dans la mise en place de la dialectique des idées et des concepts dans un processus de création.

> "Le pouvoir d'abstraction est la faculté de pouvoir travailler sur des objets abstraits, c'est-à-dire dégagés de leur contexte final, ce qui rend possible plusieurs niveaux de structuration pouvant, soit s'interpénétrer, soit fonctionner de manière plus indépendante. [...] La condition de cette abstraction réside grandement dans le fait que la notation musicale est tout à fait symbolique." [Manoury, 1990]

L'ordinateur effectue lui aussi des calculs sur des symboles : de même qu'une note est interprétée sur une partition, qu'une lettre manuscrite est reconnue comme telle, un *bit* (valeur binaire) dans un système informatique a valeur de symbole dès lors qu'il s'agit d'une tension électrique interprétée comme signifiant un 0 ou un 1. Le symbole, en principe et sous son acception la plus générale, se prête donc aussi bien au traitement mental qu'au traitement informatique ; l'informatique renforçant par ailleurs son caractère opératoire.

Il n'est donc pas inutile de faire la distinction entre symbolisme au niveau informatique et symbolisme au niveau musical (ou mental en général). Pour [Chazal, 1995], le traitement mental ou informatique des symboles est réalisé selon deux modalités distinctes :

> "Le sens conféré au symbole est traité par l'esprit qui le décode ; la machine traite son support physique." [Chazal, 1995]

L'utilisation d'un outil et d'un matériau symboliques (l'ordinateur et le signal numérique) ne suffit donc pas à assurer le caractère symbolique des représentations musicales.

> "Le signal numérique en soi ne permet pas de parler de symbolisme. Ce qui va vraiment permettre de parler de symbolisme, en revanche, ce sont toutes les opérations que l'on peut effectuer sur le phénomène grâce au numérique." C. Cadoz

Une valeur binaire est un symbole pour l'ordinateur, nous venons de le voir. Elle peut être combinée à d'autres symboles pour constituer des valeurs décimales, des caractères qui sont des symboles de plus haut niveau, plus proche du domaine du compositeur, ou de l'utilisateur d'un système informatique en général. Cependant, ni un nombre ni un caractère ne sont forcément porteurs de sens dans un processus de composition. Leur association dans une structure composée (par exemple dans la description d'une note – valeurs de hauteur, d'amplitude, de durée, etc.) pourra atteindre ce niveau symbolique : une condensation suffisante de l'information lui permet d'être intégrée dans ce processus tout en portant un sens (musical) qui lui donnera un rôle dans une structure donnée.

Les nombres qui constituent les échantillons d'un signal numérique n'ont donc pas non plus de valeur symbolique propre a priori. Ils peuvent difficilement être mis en relations les uns aux autres, et ne portent pas de sens individuellement dans un processus de composition. Un échantillon prend du sens s'il est associé aux autres échantillons du signal pour former un objet (un son). L'ensemble des échantillons constitue une forme d'onde sonore dont la valeur sémantique est donc beaucoup plus grande que la totalité de ces

valeurs isolées. A l'image des échantillons d'un son numérique, les données de description sonore (par exemple des analyses spectrales vues dans le chapitre 1), ou les signaux et fonctions de contrôle des algorithmes de synthèse, sont aussi difficilement interprétables et assimilables dans un processus musical. C'est certainement de là que vient la difficulté de l'écriture du son telle que nous l'avons envisagée : construire et manipuler un tel objet est problématique en ce sens qu'à la différence des objets musicaux traditionnels, il n'est pas naturellement décomposable en éléments symboliques.

5.1.2 Ambivalence des objets musicaux

Il est possible de ramener le son à un niveau symbolique en s'appuyant sur les catégories musicales traditionnelles, c'est-à-dire en s'intéressant à des valeurs telles que la hauteur, la durée d'évènements auxquelles sont associées des valeurs de paramétrages pour un système de synthèse. Nous avons vu précédemment que cette démarche n'avait pas grand intérêt dans l'objectif d'un contrôle musical de la synthèse sonore, passant ainsi à côté des possibilités d'une spécification plus fine des structures sonores. De même, la perte d'information due à un système "trop" symbolique n'est pas souhaitable : un tel système doit également garder une fonction de représentation ancrée dans le réel afin de permettre une translation du domaine compositionnel vers la réalité sonore (par l'intermédiaire des processus de synthèse).

[Stroppa *et al.*, 2002] évoquent le problème d'un même objet sonore, considéré selon le cas comme "une entité logique musicalement relevante", et utilisée comme telle dans un processus de traitement symbolique, ou comme "résultat d'un processus de synthèse" lorsqu'il s'agit de créer ce son. Ce ne sont pas là deux types d'objets différents mais bien un même objet vu sous deux points de vue différents. Au-delà de ces conversions systématiques du symbolique vers le signal qui ne font qu'éviter le problème, une représentation symbolique d'un son en tant qu'un objet compositionnel implique donc une réflexion sur des structures intermédiaires flexibles, voire une navigation entre le son indivisible et l'échantillon sonore insignifiant.

Il y a donc une question d'échelles de grandeurs dans la définition du symbole : une simple segmentation d'un son en entités significatives (par exemples des notes, ou des syllabes) peut constituer une forme de décomposition symbolique, contrairement à une décomposition en petits segments de quelques échantillons.[1] Mais il y a aussi une question de choix et d'appréciation : des segments de 1000 échantillons utilisés dans une analyse TFCT, ou les unités élémentaires utilisées dans un processus de synthèse concaténative peuvent être vus ou pas, selon le cas, comme des entités symboliques. La question de la représentation symbolique dépend donc aussi de critères subjectifs.

[1] L'exemple de la segmentation d'un son numérique est un cas particulier de ce que nous avons appelé la "décomposition" du son ; le cas général pourra faire appel à toute sorte de procédés d'analyse ou d'extraction de données.

5.1.3 Entre symboles et signaux

Cette question des représentations symboliques a été souvent traitée dans la littérature d'informatique musicale.

Dans [Miranda, 1995], les niveaux de structuration de l'information sont ceux des connaissances (*knowledge*), du symbolique, et de l'*engineering* (que l'on pourrait traduire comme niveau du traitement du signal). Les éléments musicaux sont ainsi mis en correspondance avec des symboles, utilisés pour penser, écrire et noter la musique. Le niveau sonore est sous ce niveau symbolique : il est atteint par un *mapping* des symboles sur des gestes musicaux (dans la musique instrumentale) ou sur des paramètres de synthèse (pour la musique électroacoustique). [Loy, 1989] distingue les représentations qu'il qualifie d'iconiques des représentations symboliques ; l'iconique étant ce qui crée un lien avec le réel, soit ce qui touche aux sons numériques, par opposition aux données symboliques (notes, etc.) La description des étapes de production musicale avec un système de composition tourné vers la synthèse se résume aussi dans ce cas en la traduction de données symboliques, manipulées au niveau compositionnel, en données ou instructions conduisant à la production de données iconiques.

Une autre manière d'expliciter cette ambivalence est d'utiliser la distinction entre les domaines symbolique et "subsymbolique" utilisée par [Leman, 1993] ou [Camurri, 1990]. [Leman, 1993] distingue dans les descriptions musicales (manières d'exprimer ou de formaliser un sens musical) différentes représentations. La représentation acoustique relève de l'ordre iconique, avec une unité de la forme et du contenu, dans lequel il n'est pas question d'interpréter des symboles. C'est le cas des signaux, et par extension des représentations analytiques du signal. Dans une représentation symbolique, au contraire, on utilise des signes, des étiquettes se rapportant à des objets, et dont la lecture et l'interprétation (compréhension de ce à quoi ils font référence) est nécessaire à l'accès au sens. Dans ce contexte, un système de synthèse transforme (encore une fois) une représentation symbolique en une représentation acoustique. Entre les deux, le domaine subsymbolique implique des mécanismes automatiques qui établissent une causalité entre ces représentations et l'apparition (mentale ou concrète) de l'objet. [Camurri, 1990] décrit le domaine symbolique comme un niveau qui fait abstraction de la nature de la machine qui traite les symboles, alors que le subsymbolique est plus proche des structures microscopiques (dans ce texte, il s'agit de représentation et de perception musicale, ce niveau subsymbolique se rattache donc aux domaines neuronaux et connexionnistes).

En adaptant ces notions à notre sujet, on pourra considérer que les données dans le domaine symbolique sont structurées et porteuses d'information sémantique : elles ont une signification pour l'utilisateur (compositeur) et sont donc susceptibles d'être mises en œuvre dans des opérations musicales. A l'opposé, le domaine subsymbolique rassemblerait donc plutôt les processus et représentations dont la signification relève du traitement informatique.

Les structures musicales passent par ce niveau subsymbolique pour accéder à une réalité sonore (même si ce n'est pas forcément sous forme informatique). Sa considération dans le cadre des outils de composition orientés vers la synthèse sonore permettra donc certainement d'assurer leur flexibilité, tout en maintenant un lien avec les représentations symboliques.

5.1.4 Synthèse et modèles

La notion de modèle à été évoquée dans le chapitre 1 pour qualifier les différentes techniques de synthèse sonore. Cette terminologie a été justifiée dans la mesure où un modèle de synthèse décrit une certaine manière de concevoir la structure du son, et où celui-ci devient un objet de formalisation.

> "La modélisation informatique a appris à transcrire la réalité empirique des sons sous la forme d'équivalents abstraits qui en formalisent les éléments constitutifs." [Dufourt, 1991]

Elle doit cependant être distinguée de celle de modèle compositionnel qui nous intéresse à présent, entendu comme modèle lié à un processus musical de composition. Prenons le cas d'un modèle particulier : celui du système musical traditionnel. Celui-ci correspondrait à une conception du son musical divisé en évènements localisés dans le temps et essentiellement définis par une hauteur (c'est-à-dire des notes). Utilisée depuis des siècles, cette conception permet d'écrire, de lire et de penser la musique par la combinaison et l'arrangement de ces composants (au sens de composants d'un modèle), organisés (particulièrement, dans le temps) en structures plus complexes (accords, mélodies, polyphonies, etc.) dans le cadre de la partition. Le modèle compositionnel est donc le processus qui aboutit à la construction de cette partition. Il est l'objet des systèmes de CAO, et dans la mesure où il a trait au calcul, il est donc aussi susceptible d'être implémenté et traité dans des programmes informatiques (voir chapitre 4).

Un son vu comme un objet de composition, comme une finalité musicale, se trouve donc modélisé à la fois par un processus de synthèse, qui engendre certaines représentations du son, mais surtout (du point de vue qui nous intéresse) par le processus qui va permettre la construction de ces représentations. Comme finalité de la musique, on peut donc voir le son comme l'objet des modèles compositionnels, la modélisation étant alors entendue de manière générale comme le processus permettant d'abstraire la musique de sa forme acoustique brute en des structures formelles. La représentation informatique du son prend alors une forme plus générale. Elle pourra inclure les techniques utilisées, mais également le choix des paramètres variants et invariants, des degrés de liberté, la description des parties déterministes ou non-déterministes du calcul, des règles prédéfinies dans le paramétrage de ce calcul. Sans négliger le processus de synthèse, qui reste cependant une variable sous-jacente au modèle, on se positionne alors plus directement dans la logique et les problématiques musicales de la création sonore.

5.1.5 Le sens des modèles

Le sens musical dans un modèle compositionnel est explicité par une interprétation permettant de recouvrer (mentalement et/ou physiquement) l'objet musical en tant que structure formelle (pendant la composition, ou l'analyse d'une pièce) ou en tant que son (lors de l'exécution de cette pièce).

A priori, les représentations statiques, compactes et structurées semblent plus à même de porter un sens musical. Cependant, nous avons vu que leur caractère proprement symbolique n'est pas assuré pour autant. L'information contenue dans un signal peut être réduite et structurée, par exemple sous la forme d'un spectrogramme ou d'un ensemble de partiels (représentation additive), mais ne permet pas nécessairement un traitement symbolique au niveau compositionnel tel que nous l'avons décrit (une seconde de son peut nécessiter la spécification de milliers de composants, liés à des fonction du temps complexes).

C'est l'organisation de composants dans un modèle pourvu d'une structure décrivant de manière symbolique leurs relations et comportements qui permettra alors de constituer *à la fois une entité permettant de créer des sons et un objet logique portant un sens musical* [Eckel et Gonzalez-Arroyo, 1994],[2] c'est-à-dire de formuler des intentions musicales. Le sens musical s'exprimera donc au travers de symboles, mais aussi de relations établies entre ces symboles.

Nous retrouvons ici l'approche défendue dans le chapitre précédent avec la CAO, mettant en avant la dualité, sinon la prédominance des processus sur les objets. Pour [Malt, 2000], la construction des modèles pose en effet principalement le problème de *substituer les symboles aux processus qu'ils représentent.* Le niveau symbolique doit ainsi être cherché autant dans les données que dans les processus, et pourra être atteint par la mise en œuvre d'abstractions diverses et à différentes échelles de ces processus.

Ici encore, la programmation va donc nous permettre de contrôler le sens musical des processus compositionnels liés au son et à la synthèse à travers la structuration des modèles.

> "La conquête du sens par des signes, voire des signaux, réside dans la réification du sens, son objectivation dans des entités matérielles. [...] Le symbole est aussi un noeud dans un système réticulaire auquel participent d'autres symboles. Le sens est diffusé dans les relations existant dans les systèmes informatiques." [Chazal, 1995]

5.1.6 Caractéristiques pour une approche compositionnelle

Nous avons donc vu jusqu'ici qu'un système de composition lié à la synthèse sonore doit permettre de travailler avec un certain niveau d'abstraction, sans lequel les processus de synthèse restent inaccessibles à la pensée musicale ; et à la fois présenter un caractère

[2] *"[...] an object capable of being viewed both as a sound producing entity and as a logical object musically meaningful."* [Eckel et Gonzalez-Arroyo, 1994]

précis, ouvert et programmable, sans lequel les applications sont restreintes au détriment du potentiel créatif des systèmes.

> "Le contrôle de la synthèse doit satisfaire à une double exigence : autoriser l'accès à des paramètres efficaces en terme de commande des dispositifs de synthèse (contrôle appelé de "bas niveau") et aider le musicien à effectuer des manipulations sur le matériau sonore qui ont un effet perceptif évident ou une intension compositionnelle directe (contrôle appelé de "haut niveau")." [Depalle et Rodet, 1993]

Entre ces deux niveaux, il est nécessaire d'établir le plus de connexions possible afin de permettre une perméabilité dans les modèles, ouverts à la fois sur les représentations symboliques, subsymboliques, et sur le domaine du signal.

Les précédentes considérations nous inspirent ainsi la nécessité d'une certaine continuité dans les niveaux d'abstraction mis en place dans un système de composition intégrant la synthèse sonore. Nous entendons par là qu'un palier fixe entre le processus de composition et celui de la synthèse sonore constitue un obstacle au développement de modèles compositionnels. La subjectivité nécessaire dans ce développement met ainsi en avant la nécessité d'un contrôle étendu, intégrant les différentes notions structurelles et comportementales de ces processus dans des niveaux d'abstraction successifs, qui permettront alors de créer par la modélisation des objets sonores musicalement significatifs.

Dès lors, les interfaces de programmation, les différents éditeurs de données ou de structures musicales, ainsi que la notion d'intégration entre données et processus que nous avons décrite dans le chapitre précédent, seront pour nous des éléments clés permettant cette intervention du compositeur dans le calcul lors de la construction des modèles.

> "Même en supposant que le sens prenne uniquement son origine chez l'homme, celui-ci prend place dans le réseau de relations informatiques par le biais des IHM [...] Le symbole acquiert un nouveau pouvoir dans un ordinateur grâce aux interfaces. [...] Il n'est plus le support passif du sens mais un élément de sa diffusion et de son efficience sur le monde." [Chazal, 1995]

[Risset, 1977] mettait déjà sur un même plan techniques d'écriture et recherche de timbres sonores, et se positionnait en faveur d'une intervention humaine dans les calculs, pour *une communication, un dialogue efficace avec la machine* afin *de contrôler ou agencer détails ou grandes lignes* et finalement de permettre au musicien de se *construire son propre monde*. Par là, le compositeur se positionnait donc aussi dans cette logique, et même parmi les précurseurs de l'approche de la CAO présentée dans le chapitre précédent, appliquée au domaine de la recherche sonore.

5.2 Environnements de CAO pour la synthèse

Comme dans le cas de la CAO en général, les langages de programmation prennent donc aussi du sens dans les systèmes de contrôle et de paramétrage des processus de synthèse sonore, garantissant à la fois liberté de décision et potentiel créatif de ces systèmes.

S'ils traitent de synthèse sonore, les travaux que nous présentons ici se distinguent des langages de synthèse du chapitre 2 dans le sens où ils permettent, au delà de la création de processus de synthèse sonores, d'implémenter des processus de plus haut niveau pour la production des paramètres de synthèse (en cela ils se positionnent d'ailleurs plus dans une logique de composition et se rapprochent des langages de CAO).

5.2.1 Langages "classiques"

GROOVE [Mathews et Moore, 1970] fut certainement le premier langage permettant de formuler avec un niveau d'abstraction relativement élevé des instructions pour un synthétiseur. Il s'agissait d'un système destiné à la génération de fonctions (données de paramétrages évoluant dans le temps) à partir d'entrées capturées en temps réel. Le langage SCORE [Smith, 1972] permettait lui aussi la saisie et le calcul de données pour le contrôle d'un dispositif de synthèse selon un modèle proche de celui de la notation musicale traditionnelle.

FORMES a été l'un des précurseurs des projets de CAO menés à l'IRCAM à partir des années 1980 [Rodet et Cointe, 1984], et constitue aujourd'hui encore une approche originale dans le domaine de la synthèse sonore. Cet environnement LISP est basé sur l'idée de processus–objets, créés notamment à partir de "copies différentielles". Il permettait d'établir une hiérarchie de ces processus dans laquelle les processus de bas niveaux contrôlant les paramètres de synthèse étaient successivement encapsulés dans des processus de niveaux de plus en plus élevés, spécifiant des règles formelles avancées, telles que la prosodie, l'expression, ou la structuration globale d'un extrait sonore. La hiérarchie de processus constituait ainsi un graphe de calcul évalué périodiquement, modifiant l'état de signaux ou de paramètres de synthèse. FORMES a été utilisé principalement pour contrôler le synthétiseur CHANT (voir chapitre 2, section 2.2) et permettait de faire face à la complexité des données tout en exploitant les potentialités de ce synthétiseur. Le caractère instantané dans l'évaluation des processus de contrôle dans un tel systeme limite la portée temporelle et musicale du calcul ; cependant la notion de règles (fonctions, ou sous-programmes, régulant l'évolution d'un ou plusieurs paramètres du synthétiseur dans la durée des processus) offrait un contrôle dynamique de l'organisation et de l'exécution des processus de synthèse, une idée qui a aujourd'hui quasiment disparu avec l'abandon du projet au profit de nouvelles interfaces de contrôle graphiques (en particulier avec DIPHONE – voir chapitre 2, section 2.4.2), dont le potentiel est finalement plus limité.

```
(de rest? (in-n-notes)
  (when (eq master (fself name))
    (let ((notes (fself?'active-sons)) (t-sum 0))
       (let ((a-note (nthcdr in-n-notes notes)))
          (when (memq a-note'(short-rest'long-rest))
            (repeat in-n-notes (+=t-sum ((nextl notes)?'duration)))
            (ajuste (the other master) in-n-notes (plus time t-sum))
            (when (gt (random 06) 3)
            (setq master (alter master ))))))))
(de the-other (voice)
  (if (eq voice 'VOICE1) 'VOICE2 'VOICE1 ))
(de ajuste (voice in-n-notes the-time)
  (let ((reste ((car (voice?'active-sons))?etime))
       (tsum () ) (list-t () ) (list-o () )
         (r-s (cdr (voice?'active-sons) )))
     (setq tsum reste)
     (let ( (r-s r-s))
        (when (and (lt tsum the-time) r-s)
          (setq tsum (plus tsum (newl list-t
            (apply (get (newl list-o (nextl r-s)) 'duration:) ) ))))
          (self r-s)))
     (if (setq aux (sub tsum reste) )
     (setq fact (div (sub the-time reste) (-tsum reste) )))
     (mapc list-o (lambda (o) (put o'duration:[lambda()
                (mul fact (nextl list- t))])))
)))
```

FIGURE 5.1: Extrait d'un programme dans FORMES.

Dans les langages pour la composition et le contrôle de la synthèse développés par Roger Dannenberg (ARCTIC [Dannenberg *et al.*, 1986], CANON [Dannenberg, 1989], NYQUIST [Dannenberg, 1997]), le paradigme fonctionnel et l'intégration données–programmes permettent de manipuler les objets sous forme de fonctions et de gérer les aspects de synchronisation et de transformation des variables dans le temps par des prototypes fonctionnels. Une des principales problématiques traitées dans ces environnements est en effet celle du contrôle des variations continues (ou pseudo continues) des paramètres de synthèse dans le temps (nous reviendrons sur ces problématiques dans la quatrième partie de la thèse). Une notion fondamentale dans ces langages est l'abstraction comportementale (*behavioral abstraction*), qui permet à l'utilisateur de définir le comportement des objets en fonction du contexte dans lequel ils se trouvent. Dans [Dannenberg *et al.*, 1986] l'auteur introduit également la notion de programmation déclarative dans les spécifications temporelles des paramètres de synthèse : les relations entre fonctions peuvent être déclarées dans le programme et non établies par le biais d'une procédure. Avec NYQUIST, qui est le plus récent de ces langages, la notion de *lazy evaluation*, ou évaluation paresseuse, est mise en avant pour implémenter des opérations sur des objets–signaux, évalués en dernière instance pour la synthèse [Dannenberg, 1997].

L'environnement FOO [Eckel et Gonzalez-Arroyo, 1994] est un autre travail sur les langages pour le contrôle de la synthèse (également en LISP) dans lequel les auteurs essaient d'intégrer le temps linéaire à petites unités dans le cadre du traitement du signal avec un temps plus logique et hiérarchisé dans le domaine symbolique. L'encapsulation des modules de synthèse sonore dans des abstractions fonctionnelles permet une structuration hiérarchique du temps. Cet environnement est cependant resté à l'état de prototype et n'a pas vu d'applications concrètes.

Avec Varèse, [Hanappe, 1999] propose un environnement alliant synthèse sonore en temps réel et composition musicale. Cet environnement, écrit en Java, est assorti d'un système puissant de gestion de la mémoire et des processus, contrôlé par un interpéteur Scheme offrant une interface de haut niveau (envoi de commandes au système de synthèse temps réel) et une interactivité de programmation (création de processus plus complexes et personnalisation de l'environnement). La représentation du temps permet d'appréhender spécification d'évènements ou de groupes d'évènements en relation aux fonctions du temps continues (évolution de paramètres), et d'envisager dans un même système la création de formes de haut niveau contrôlant, par un mécanisme d'abstractions successives, des processus allant jusqu'au niveau de la synthèse sonore.

5.2.2 Langages visuels

Le caractère "textuel" des précédents langages les rend certainement quelque peu hermétiques, ou du moins nécessite de la part de l'utilisateur une certaine aisance en programmation pour la mise en œuvre des processus les plus élémentaires (voir figure 5.1). Avec des langages visuels, les possibilités offertes par la programmation sont assorties d'une spécification graphique permettant une représentation explicite et plus intuitive (voire musicale) des processus et des données (voir chapitre 4, section 4.2.2).

PWGL [Laurson et Kuuskankare, 2002], que nous avons déjà mentionné précédemment (chapitre 4), mérite également d'être cité parmi les langages orientés vers la synthèse. Celui-ci est en effet doté, en supplément de la partie de programmation visuelle de type Lisp/PatchWork, d'un module complet dédié à la synthèse sonore (PWGLSynth [Laurson *et al.*, 2005]). Ce module contient un processeur de signal écrit en C (plus efficace que Lisp pour le traitement du signal), séparé mais étroitement intégré au langage. La figure 5.2 montre un exemple de patch utilisant PWGLSynth.

Avec PWGLSynth, il s'agit donc dans un premier temps de créer des instruments de synthèse, sur le modèle d'un langage visuel de synthèse : les aspects temps réel du système permettent une interaction et un test pratique et immédiat. L'aspect "contrôle" relève de la partie Lisp, plus proche des concepts de CAO, avec laquelle sont générées des partitions représentées grâce à un système de notation lui aussi programmable (ENP [Kuuskankare et Laurson, 2006]), et dont les informations sont mises en relations aux paramètres des instruments de synthèse. PWGLSynth constitue ainsi un mélange original de programmation fonctionnelle et de système temps réel pour la composition et la synthèse sonore. Les paradigmes de CAO et de synthèse sonore restent cependant distincts et relativement indépendants ; nous essaierons dans nos travaux, au détriment de la partie "purement synthèse", qui sera déléguée à des outils externes, de favoriser l'intégration de l'aspect compositionnel dans les processus de synthèse.

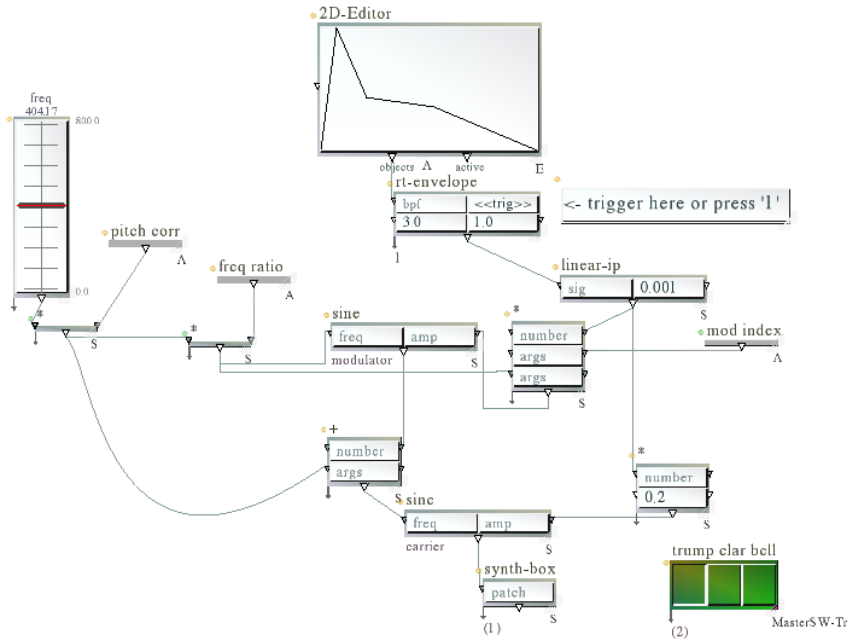


FIGURE 5.2: Un programme de contrôle et de synthèse sonore dans PWGL.[3] Le graphe de calcul définit la sémantique fonctionnelle du programme et permet d'envoyer des messages en temps réel au moteur de synthèse (par exemple avec le *slider* sur la gauche.)

5.2.3 Contrôle de la synthèse sonore dans OpenMusic : précédents travaux

Si OPENMUSIC est un environnement plutôt spécialisé dans le traitement de données musicales liées à la musique instrumentale (notes, accords, etc.), quelques précédents travaux visant à travailler spécifiquement avec la synthèse sonore y ont cependant été réalisés. Pour la plupart liés à des systèmes et environnements évoqués dans le chapitre 2, ces travaux constituent ainsi une première étape dans un dialogue entre l'environnement de composition et le domaine du signal sonore.

Génération de données de contrôle

Une première idée pour une utilisation d'un système de CAO orientée vers la synthèse sonore est l'utilisation des potentialités de calcul de l'environnement pour la génération de données de paramétrage destinées à des outils logiciels externes de synthèse et de traitement du son. Cette démarche s'effectue principalement par l'écriture de données préalablement calculées et formatées dans des fichiers, destinés à être lus par les programmes de synthèse. En supplément des possibilités qu'offre un langage de programmation en termes de génération de ces données par rapport à une spécification directe des

[3] Patch issu du tutorial de l'application.

paramètres, un attrait important consiste ici en la mise en relation (conversions, visualisation, etc.) des données musicales traditionnelles (par exemples des accords, séquences musicales, rythmes, etc.) et des données formatées pour la synthèse.

OM2Csound [Haddad, 1999] est une bibliothèque de fonctions permettant la génération automatique de *scores* pour Csound.[4] Comme nous l'avons vu dans le chapitre 2, un *score* (ou partition) dans Csound active les instruments de synthèse définis dans l'*orchestra* à des instants donnés et pendant une durée déterminée, attribuant des valeurs aux données déclarées comme variables dans ces instruments (paramètres numériques, tables et fonctions utilisées, etc.) De tels paramètres peuvent ainsi être générés par des programmes visuels et traduits en partition pour le synthétiseur. En particulier, les classes d'objets musicaux ou extra-musicaux (séquences d'accords et *breakpoint functions*) peuvent être utilisées comme structures de données initiales, et leurs éditeurs respectifs comme interfaces pour la représentation et la saisie des données pour le contrôle des instruments (voir figure 5.3).[5]

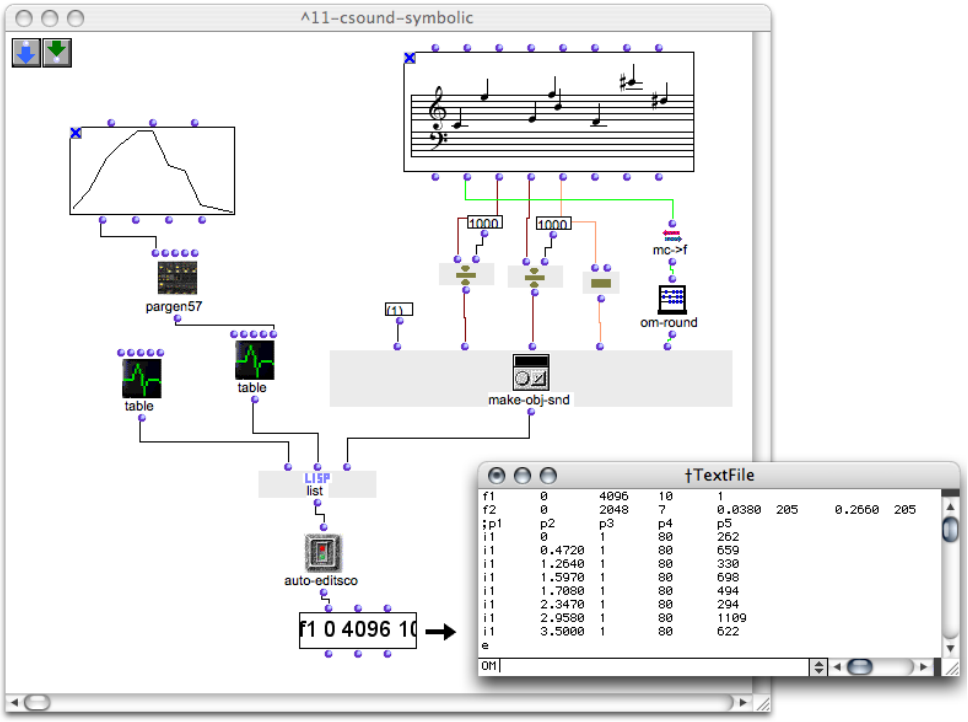


Figure 5.3: Génération de partition Csound dans OpenMusic : écriture d'un fichier *score* à partir de processus et données musicales.

[4]La bibliothèque OM2Csound a été développée dans PatchWork par Laurent Pottier avec l'aide de Mikhail Malt et adaptée dans OpenMusic par Karim Haddad.

[5][Pottier, 2008] donne un exemple d'application musicale concrète de ces outils pour le contrôle d'un processus de synthèse granulaire.

OM-AS [Tutschku, 1998] est une autre bibliothèque OpenMusic dédiée à la génération automatique de paramètres pour le programme SuperVP. L'idée est encore une fois de tirer parti des possibilités de calcul et des objets et éditeurs disponibles dans l'environnement de CAO pour les traduire sous forme de paramètres destinés à SuperVP. Il s'agit donc ici aussi d'écrire un fichier (au format texte), que le programme pourra lire et utiliser pour le paramétrage des traitements. Ce fichier peut être utilisé dans AudioSculpt (d'où le nom OM-AS), permettant d'appliquer des filtrages par bandes, transpositions ou étirements variables dans le temps [Tutschku, 2003] ; alors qu'il aurait été compliqué et fastidieux de spécifier ceux-ci manuellement ou à l'aide des outils de dessin dans AudioSculpt. La figure 5.4 montre des exemples avec la transposition d'un son. Dans l'exemple de gauche, les paramètres de la transposition sont spécifiés et écrits dans un fichier de paramètres à partir d'une courbe sinusoïdale, ce qui produira une sorte de vibrato sur le son resynthétisé par SuperVP. Dans la partie de droite, les données proviennent d'une séquence de notes dans OpenMusic : le son sera transposé suivant cette mélodie.

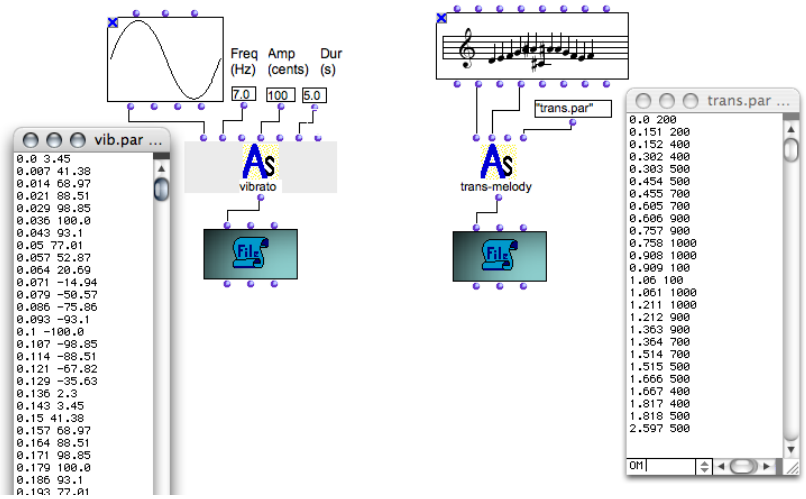


Figure 5.4: Génération de paramètres pour SuperVP dans OpenMusic avec la bibliothèque OM-AS.

Avec la librairie OMDiph,[6] il est également possible de générer depuis OpenMusic, et donc par l'intermédiaire de programmes, des dictionnaires et des scripts de paramétrages pour le logiciel Diphone.

Ecriture des processus de synthèse

Dans le cas de systèmes offrant un accès au contrôle et à la spécification des processus de synthèse (ceux de la catégorie des langages de synthèse du chapitre 2), OpenMusic peut

[6] Créée par Jean Lochard.

également servir d'interface graphique pour la description et l'écriture de ces processus. Différents travaux ont également été menés dans cette direction.

OM2Csound a été incrémentée d'une partie dédiée à la génération des instruments. Un éditeur de patches particulier (voir figure 5.5) permet de constituer graphiquement des instruments de synthèse à l'aide de boîtes représentant les modules de génération et de traitement des signaux. Suivant le principe de Csound (voir chapitre 2, section 2.3.1), certains des paramètres sont explicités alors que d'autres (tables et autres paramètres de contrôle) sont ouverts en vue d'une spécification dans la partition (*score*) qui sera utilisée avec ces instruments.

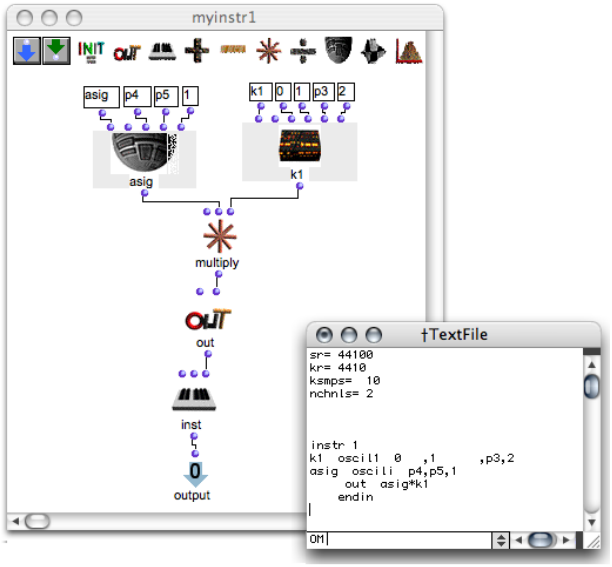


Figure 5.5: Création d'un instrument Csound dans OpenMusic. Contrairement à ce que l'on a vu avec les *scores*, le graphe fonctionnel n'effectue pas des calculs à proprement parler mais produit sa propre interprétation dans le langage des orchestres Csound.

La librairie OM-Modalys[7] permet le contrôle du synthétiseur par modèles physiques Modalys depuis OpenMusic [Ellis *et al.*, 2005]. La partie contrôle de Modalys étant elle-même en Lisp (voir chapitre 2, section 2.3.3), les deux environnements vont cette fois s'intégrer l'un à l'autre : l'évaluation d'une boîte dans OpenMusic correspond directement à une commande de Modalys, reflétant fidèlement le mode de contrôle initial. Un système physique est donc construit incrémentalement suivant l'ordre d'évaluation du patch : il s'agit finalement d'une suite d'instructions terminée par un appel à la fonction de synthèse proprement dite. Ici encore, les structures de données et les éditeurs correspondants disponibles dans OpenMusic permettent de faciliter le contrôle des processus de synthèse, principalement pour la spécification de leurs données initiales (mouvements, forces, caractéristiques physiques des structures, etc.)

[7]Créée par Nicholas Ellis.

MODALYS-ER est un autre système pour la création et le contrôle d'instruments virtuels MODALYS, porté dans OPENMUSIC avec la bibliothèque MFOM [Polfreman, 2002]. Celle-ci permet un contrôle de plus haut niveau, les instruments (systèmes physiques) y étant définis sous forme de classes, dotées d'un éditeur graphique. Les instances créées sont ensuite interprétées et transmises vers le synthétiseur. Cette librairie n'est aujourd'hui plus maintenue dans les versions récentes de OPENMUSIC.

5.3 Conclusion

Nous avons présenté un certain nombre de notions et travaux allant dans le sens d'une intégration de la synthèse sonore dans le paradigme de programmation proposé par les environnements de CAO. Dans ce domaine, des avancées significatives ont été réalisées dans la compréhension de ses enjeux, mais peu d'outils sont réellement utilisés par les compositeurs, hormis, et sauf exceptions, par leurs propres auteurs. Souvent, la forme joue beaucoup dans ce sens : les langages de programmation peuvent (à juste titre) sembler hermétiques aux musiciens. La programmation visuelle est en cela susceptible de résoudre une partie de ces difficultés. Ainsi, nous entendons dans la suite de ce travail montrer que les principes d'un environnement comme OPENMUSIC, étendus au domaine de la synthèse, peuvent conduire à un système permettant le développement de modèles liés au son et porteurs de sens musical.

Parmi les principales questions soulevées alors, nous avons particulièrement mis en avant celle de l'élaboration de représentations symboliques, à travers les possibilités d'abstraction, de hiérarchisation, et d'interfaces utilisateur adaptées. La problématique du temps est un autre aspect fondamental, que nous laissons volontairement de côté dans un premier temps pour nous y concentrer dans la quatrième partie de la thèse.

Des outils existant dans OPENMUSIC permettent de réaliser certaines tâches liées à la synthèse sonore, souvent spécifiques et ponctuelles dans un projet compositionnel. Ils représentent néanmoins des premiers pas dans l'idée du système que nous souhaitons mettre en place. Dans [Bresson *et al.*, 2005b], des premières propositions sont faites visant à relier ces outils aux modèles développés dans OPENMUSIC. L'objectif suivi dès lors a été de permettre à OPENMUSIC de devenir un centre de contrôle d'un projet compositionnel complet, permettant au calcul, à la CAO et au formalisme musical d'intégrer le domaine de la synthèse et du traitement du signal.

Dans cette approche, le son ne sera donc pas considéré comme un signal, représenté par un ensemble de données issues d'un programme, mais comme l'objet d'un processus (ou modèle) compositionnel visant à produire un tel signal.

Nous nous concentrerons dans les prochains chapitres sur le développement et l'intégration dans OPENMUSIC d'outils dédiés à l'écriture de musique électroacoustique, dans l'idée de favoriser cette connexion du domaine de la pensée et de la formalisation musicale avec celui du signal sonore.

Troisième partie

Outils de programmation pour la représentation et la synthèse des sons en composition assistée par ordinateur

Chapitre 6

Programmation visuelle avec les sons et la synthèse sonore

OMSounds est un environnement, ensemble d'outils et d'interfaces, permettant la création de processus électroacoustiques dans OpenMusic, et par conséquent le développement des démarches compositionnelles correspondantes [Bresson et Agon, 2006a].

Dans ce chapitre, nous présentons une première partie des fonctionnalités de cet environnement, correspondant aux outils permettant d'intégrer le traitement et la synthèse sonore dans les programmes visuels [Bresson, 2006b]. Nous essaierons également d'envisager des premières possibilités offertes par cette extension de l'environnement de CAO.

6.1 Architecture et utilitaires pour l'audio

Les sons sous forme numérique sont les objets de nos modèles étendus à la synthèse. Ils peuvent aussi entrer en jeu dans les processus compositionnels liés à ces modèles, en tant que matériau initial ou intermédiaire. Une première étape dans une optique de composition électroacoustique est donc de disposer d'outils pour la manipulation de ces objets. Dans OpenMusic, la classe `sound` représente un fichier audio. Celle-ci est instanciée à partir d'un *pathname* spécifiant la localisation de ce fichier sur le disque dur (voir figure 6.1).

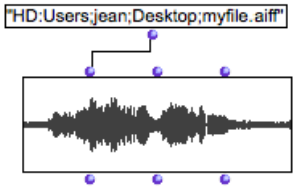


Figure 6.1: La classe `sound` dans OpenMusic.

Nous avons implanté une nouvelle architecture audio dans OpenMusic, basée sur la bibliothèque LibAudioStream[1] développée à Grame (Lyon). Cette restructuration a permis de disposer d'un support audio multi-plateformes et de faciliter les opérations de rendu et de manipulation des fichiers audio.

6.1.1 LibAudioStream : un système pour la manipulation et le rendu de ressources audio

LibAudioStream permet de gérer le contrôle de ressources audio par l'intermédiaire de *streams*.

Un *stream* est un pointeur désignant initialement une source audio (fichier ou entrée temps réel). Celui-ci pourra être manipulé à l'aide d'une série de fonctions renvoyant systématiquement de nouveaux *streams* audio : `MakeCutSound` (renvoie un *stream* correspondant à un extrait de la source d'un *stream* initial), `MakeSeqSound` (renvoie un *stream* correspondant à la mise en séquence de deux autres *streams*), `MakeMixSound` (renvoie un *stream* correspondant à la superposition de deux *streams*), `MakeLoopSound` (renvoie un *stream* correspondant à la répétition d'un autre *stream*), etc.

Ce système permet de créer et manipuler les sons de façon abstraite et de définir des processus de traitements en cascade, appliqués en dernière instance lors du rendu d'un *stream*. L'exemple ci-dessous montre de tels processus réalisés à l'aide des fonctions de LibAudioStream :

```
// Met en séquence un segment silencieux de 40000 échantillons
// et un extrait du fichier "sound1.wav"
AudioStream s1 = MakeSeqSound(
                   MakeNullSound(40000),
                   MakeRegionSound("sound1.wav", 0, 20000));

// Superpose le stream précédent avec le fichier "sound2.wav"
// et applique un fadein/fadeout sur le stream obtenu
AudioStream s2 = MakeFadeSound(
                   MakeMixSound(
                     s1,
                     MakeReadSound("sound2.wav"),
                     44100, 44100));

// etc..
```

Des transformations sur le son peuvent être appliquées de la même manière : la fonction `MakeTransformSound` renvoie un *stream* à partir d'un *stream* initial et d'un objet `AudioEffect` (des effets élémentaires sont disponibles : volume, panoramique, etc.) Pour

[1] http ://libaudiostream.sourceforge.net

des effets plus avancés, il est possible d'utiliser le langage FAUST (voir chapitre 2, section 2.3.2) pour créer un `AudioEffect` à partir d'un traitement défini (et compilé) dans ce langage.

Le rendu des *streams* dans LIBAUDIOSTREAM est effectué grâce à un *player* dont les fonctionnalités sont similaires à celles des séquenceurs multipistes traditionnels : création, chargement et contrôle de l'exécution (avec possibilité d'appliquer des effets) sur les différentes pistes et/ou sur l'ensemble du *player*.

Ce rendu peut également être redirigé sur un fichier, de sorte à enregistrer le son-résultat correspondant aux différentes opérations.

Intégration de la bibliothèque audio

Afin d'être utilisable dans OPENMUSIC, les fonctions de la bibliothèque doivent être encapsulées dans des appels en LISP. Des liens dynamiques (*bindings*) peuvent être créés de différentes manières selon les implémentations LISP et les FFI (*foreign function interface*) dont elles disposent. Ceux-ci définissent des fonctions LISP correspondant aux fonctions définies dans l'interface de la bibliothèque (comme `MakeMixSound`, `MakeTransformSound`, etc.)

6.1.2 Nouvelles fonctionnalités audio dans OpenMusic

Avec les possibilités offertes par LIBAUDIOSTREAM, on est à présent en mesure d'allouer aux objets `sound` de nouvelles caractéristiques. La classe `sound` se voit donc affectés de nouveaux *slots*, dont principalement un `volume`, un `pan` (panoramique), et un `track` (piste audio). Ces valeurs sont assignables dans l'éditeur audio (voir figure 6.2).

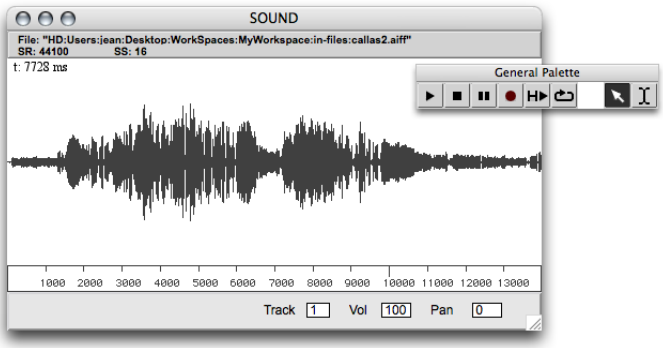


FIGURE 6.2: L'éditeur de fichier audio (classe `sound`).

Au moment de jouer ces objets `sound` (dans l'éditeur audio, dans les patches, ou dans une maquette), on utilise les fonctions citées plus haut pour effectuer un rendu sur plusieurs pistes (si différents *tracks* sont spécifiés), et selon les attributs de chaque son. Pour une séquence donnée chaque son est d'abord correctement positionné à l'aide des fonctions

`MakeSeqSound` et `MakeNullSound` (pour les silences), et modifié selon ses attributs (volume, panoramique) propres avec `MakeTransformSound`. Tous les sons appartenant au même track sont ensuite regroupés avec la fonction `MakeMixSound`. Il ne reste ainsi qu'un *stream* par piste, qui est chargé dans le *player* de la bibliothèque.

Mixage

La possibilité de contrôle par pistes dans le *player* permet d'appliquer des effets et transformations sur les sons par groupes (en fonction des différents *tracks*).

Une nouvelle interface que nous avons créée (visible sur la figure 6.3) permet ainsi d'envoyer des appels au *player* (`SetVolChannel` et `SetPanChannel`) au moment de la lecture d'un ou de plusieurs fichiers simultanés (par exemple lors de la lecture d'une maquette contenant plusieurs objets `sound`).

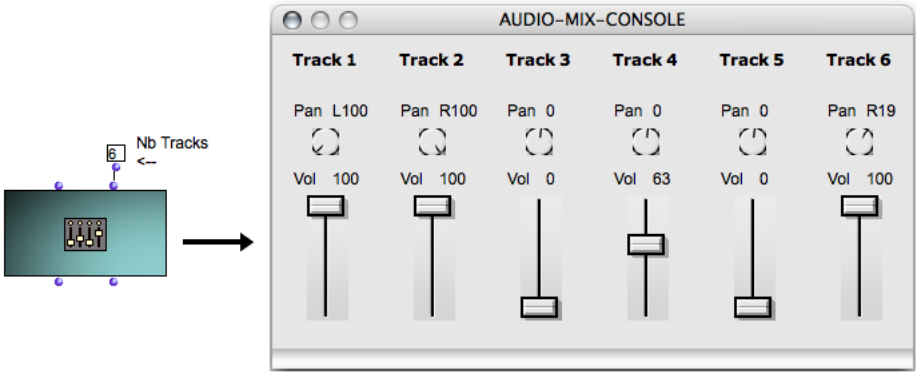


FIGURE 6.3: `audio-mix-console` : objet musical dans OPENMUSIC, et éditeur pour le mixage des pistes audio.

Cette interface est en réalité l'éditeur associé à la classe `audio-mix-console`, qui est un objet musical au même titre que les autres (comme `note`, `chord`, `sound`, etc.) Il contient un certain nombre de valeurs (volumes et panoramiques de différentes pistes audio), éditables par le biais de cet éditeur, qui est capable de les envoyer au *player* systématiquement lorsqu'elles sont modifiées (i.e. lorsque l'on change la position des *faders* ou des potentiomètres). En conséquence cet objet, correspondant à un ensemble fixé de paramètres audio (*settings*), peut exister dans un patch sous forme d'une boîte *factory* (également visible sur la figure 6.3), ou même être introduit dans une maquette.

L'exécution (*"play"*) d'un `audio-mix-console` consiste donc en l'envoi des données qu'il contient (éventuellement réglées manuellement) au *player* audio. Plusieurs peuvent exister simultanément, dans une maquette, par exemple, modifiant ces réglages aux instants correspondant à leur position (voir figure 6.4).

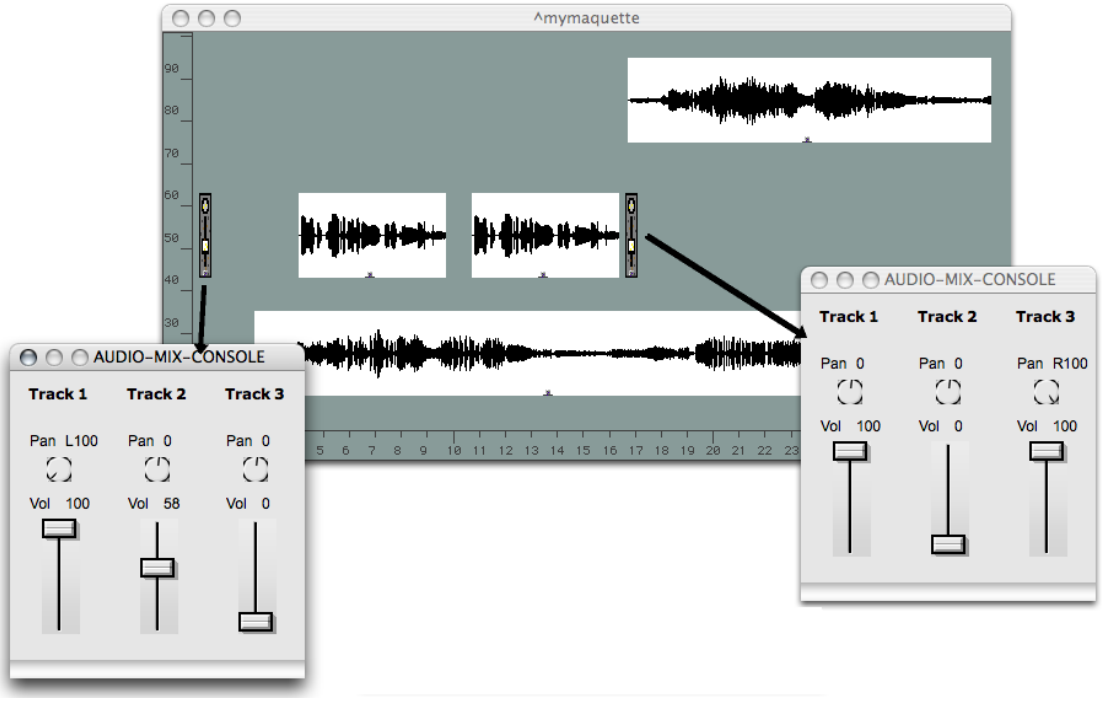


FIGURE 6.4: Utilisation d'objets `audio-mix-console` dans une maquette. Les réglages de la première instance sont utilisés jusqu'à l'arrivée du curseur de lecture à la position du second, qui envoie alors ses valeurs au *player*.

Segmentation

La classe *sound* possède également un nouveau *slot* `markers`, qui constitue une simple liste d'instants associée au fichier. Ces *markers* peuvent être spécifiés et édités manuellement dans l'éditeur de fichier audio, ou calculés dans un patch et connectés à la boîte (*factory*) `sound` (voir figure 6.5).

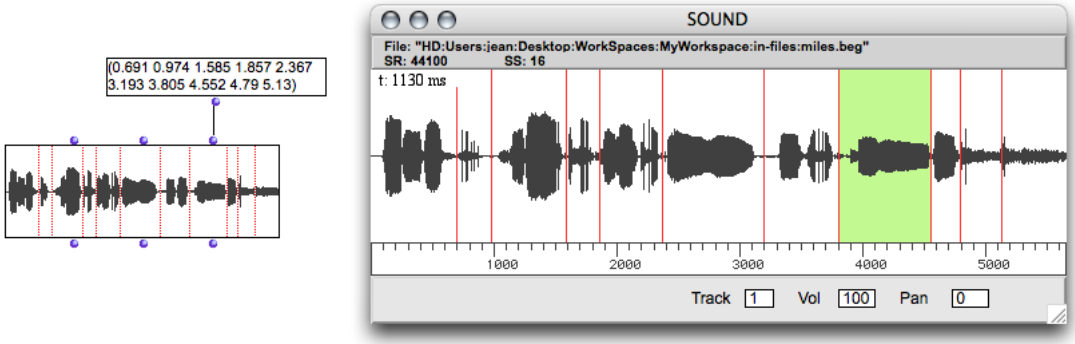


FIGURE 6.5: *Markers* associés à un objet `sound`.

Les *markers* constituent une segmentation du son, selon laquelle il pourra être découpé, depuis l'éditeur, encore une fois (glisser une région sélectionnée entre deux *markers* dans

un patch crée une nouvelle instance `sound` correspondant à cette région) ou à l'aide de fonctions dans un patch, comme nous le verrons dans la section suivante.

6.2 Bibliothèque de fonctions pour le traitement des sons

Les différentes fonctions de traitement et de transformation des *streams* de LibAudioStream sont également adaptées sous forme d'une bibliothèque de fonctions graphiques, permettant de manipuler les sons dans le langage visuel.

Les principales boîtes de cette librairie sont illustrées sur la figure 6.6 :

FIGURE 6.6: Fonctions audio.

- `sound-silence` : crée un *stream* "silencieux" (nul) pendant une durée donnée ;
- `sound-cut` : renvoie un *stream* correspondant à une région d'un son ;
- `sound-mix` : renvoie un *stream* correspondant à la superposition de deux sons ;
- `sound-seq` : renvoie un *stream* correspondant à la séquence de deux sons ;
- etc.

La figure 6.7 montre des exemples simples d'utilisation de ces fonctions avec des fichiers audio.

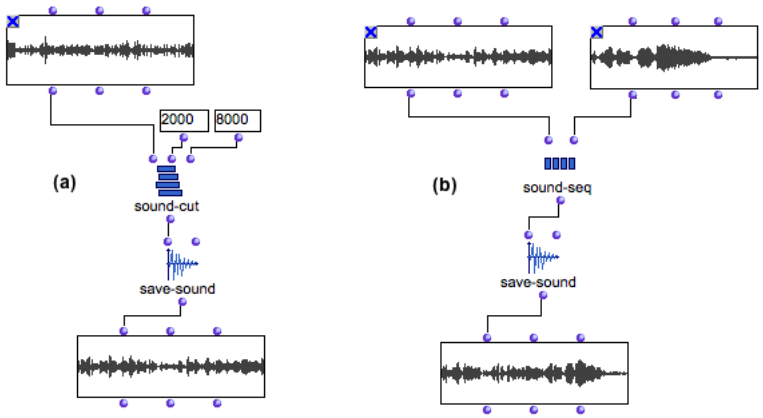


FIGURE 6.7: Exemples d'applications de fonctions de la bibliothèque LibAudioStream dans OpenMusic : (a) `sound-cut` renvoie un extrait du son (de 2000ms à 8000ms) ; (b) `sound-seq` met deux sons en séquence.

La possibilité d'utiliser des effets compilés par le langage Faust pour la transformation de *streams* audio est également intégrée dans cette bibliothèque (voir figure 6.8). Faust définit des modules de DSP présentant éventuellement un certain nombre de paramètres et transformant un flux d'échantillons audio en un autre.

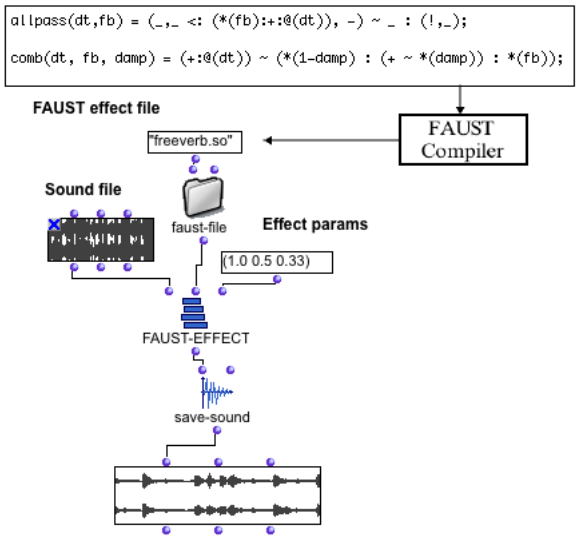


FIGURE 6.8: Application d'un effet défini et compilé par FAUST sur un *stream* audio.

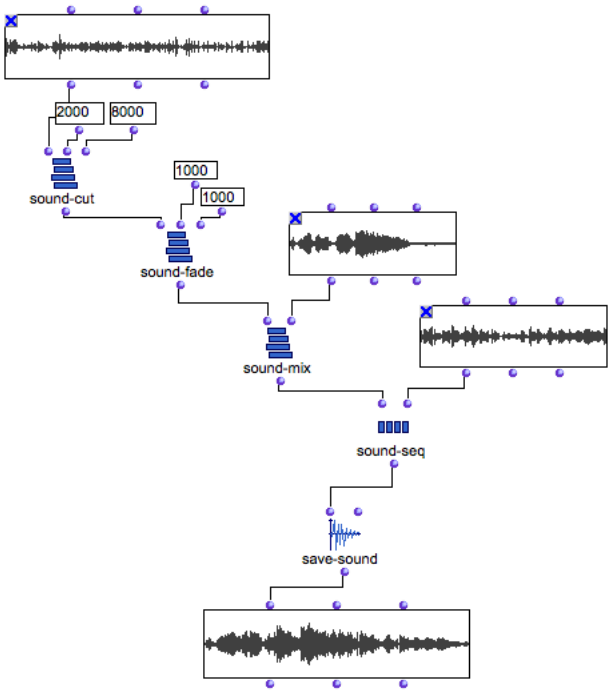


FIGURE 6.9: Application d'une succession de traitements sur des sons. Les *streams* sont des états intermédiaires transmis entre les fonctions ; les opérations sont appliquées au moment de l'écriture du fichier (`save-sound`).

Les fonctions de cette bibliothèque renvoient toutes un *stream*, faisant une référence au *stream* initial et à l'opération correspondante : l'application effective des opérations est retardée au moment du rendu ou de l'écriture du flux audio (on passe par la fonction `save-sound` pour écrire un *stream* dans un fichier audio). Les sons donnés en entrée

peuvent donc se présenter soit sous forme de fichiers (objet `sound`), que l'on convertit alors en *stream* en interne, soit directement sous forme de *stream*. Si les données des sons (échantillons numériques) sont initialement stockées sur des fichiers, donc externes au système, ces fonctions permettent ainsi de les manipuler de façon abstraite dans les patches. Il est ainsi possible de les appliquer les unes à la suite des autres sans passer nécessairement par l'intermédiaire de nouveaux fichiers audio : le traitement est appliqué uniquement au moment de l'écriture du fichier (avec `save-sound`). La figure 6.9 montre un exemple de séquence de traitements appliqués ainsi sur un son.

Les sons peuvent donc êtres crées, coupés, mixés, séquencés selon des procédés définis algorithmiquement et grâce aux outils de programmation visuelle (voir figure 6.10). Associés à des itérations, par exemple, ils permettront de créer des assemblages de sons à grande échelle et complètement automatisés, un "montage algorithmique" qui aurait demandé un temps et une précision considérables avec des outils de montage classiques.

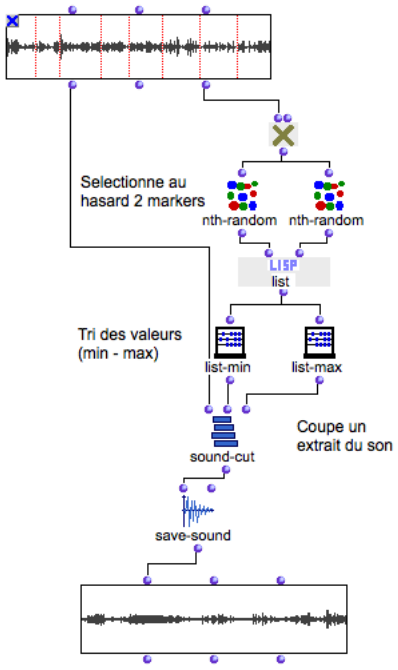


FIGURE 6.10: Exemple d'utilisations des fonctions de la librairie audio pour la manipulation de sons en relation à des processus algorithmiques : les paramètres de la fonction `sound-cut` sont calculés par un processus sélectionnant aléatoirement des markers dans le fichier audio afin d'en tirer un extrait au hasard.

L'exemple illustré sur la figure 6.11 est inspiré d'un patch créé par le compositeur Hans Tutschku. Il s'agit d'un programme effectuant une série de tirages aléatoires de segments d'un fichier audio (sur le modèle de l'exemple de la figure 6.10), mis itérativement en séquences les uns à la suite des autres pour recréer un nouveau fichier.

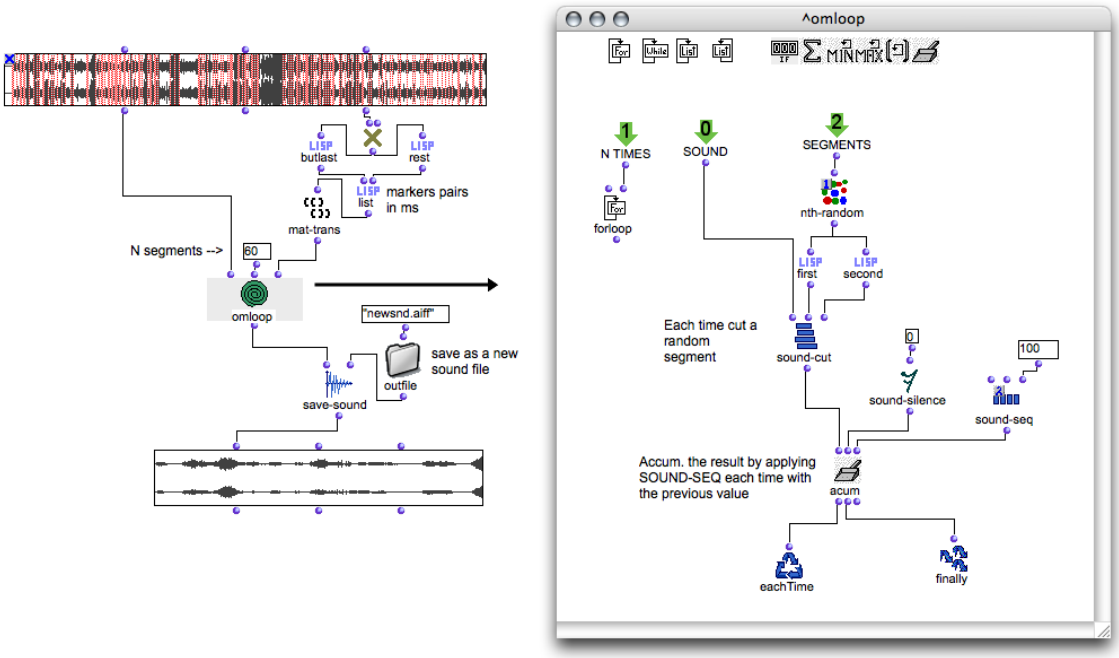


FIGURE 6.11: Manipulation algorithmique d'un signal audio : le processus `omloop` découpe et met en séquence itérativement 60 segments tirés aléatoirement dans le son initial.

6.3 Intégration des systèmes de synthèse

Nous avons évoqué précédemment l'avantage de considérer les processus de synthèse sonore comme des éléments variables des modèles, afin de permettre la construction de processus compositionnels non contraints par ces systèmes. Dans le système que nous mettons en place, rappelons qu'il n'est pas question de traiter les tâches de synthèse et de traitement du signal directement, mais d'organiser un environnement permettant de contrôler et coordonner ceux-ci à l'intérieur d'un processus musical.

Différents protocoles permettent ainsi d'utiliser des systèmes de synthèse à l'intérieur de cet environnement. Des interfaces ont été créées, avec des outils de synthèse spécifiques, permettant de convertir des données provenant de OPENMUSIC dans des formats de paramétrage adaptés (nous avons déjà vu quelques exemples dans le chapitre précédent) et d'appeler ces systèmes à l'intérieur même des processus définis dans l'environnement.

Parmi les différents systèmes et outils de synthèse, nous utiliserons principalement CSOUND, qui comme nous l'avons vu (chapitre 2, section 2.3.1) permet une grande flexibilité dans la création de processus de synthèse, mais également le synthétiseur CHANT de l'IRCAM, ainsi que les outils d'analyse/synthèse SUPERVP (vocoder de phase) et PM2 (analyse et synthèse additive). Ce panel est évidemment extensible, l'objectif étant de fournir des outils suffisamment généraux et souples pour s'adapter à de nouveaux programmes et systèmes de synthèse.

Les programmes que nous avons cités se présentent sous forme d'exécutables appelés par ligne de commande dans un terminal. Depuis un interpréteur LISP, il est généralement possible de lancer ce type de commande en invoquant un *shell* (terminal virtuel) par un appel système. On peut ainsi créer des fonctions dédiées à l'appel de certains programmes avec des paramètres adaptés. Il s'agit alors de formater la ligne de commande adéquate et de la lancer dans le *shell*.

La figure 6.12 montre par exemple la fonction `csound-synth` qui permet d'appeler le compilateur CSOUND en lui spécifiant un fichier *orchestra* et un fichier *score*. Cette fonction renvoie le chemin vers le fichier audio créé, de sorte qu'il est possible de la connecter directement à un objet `sound`.

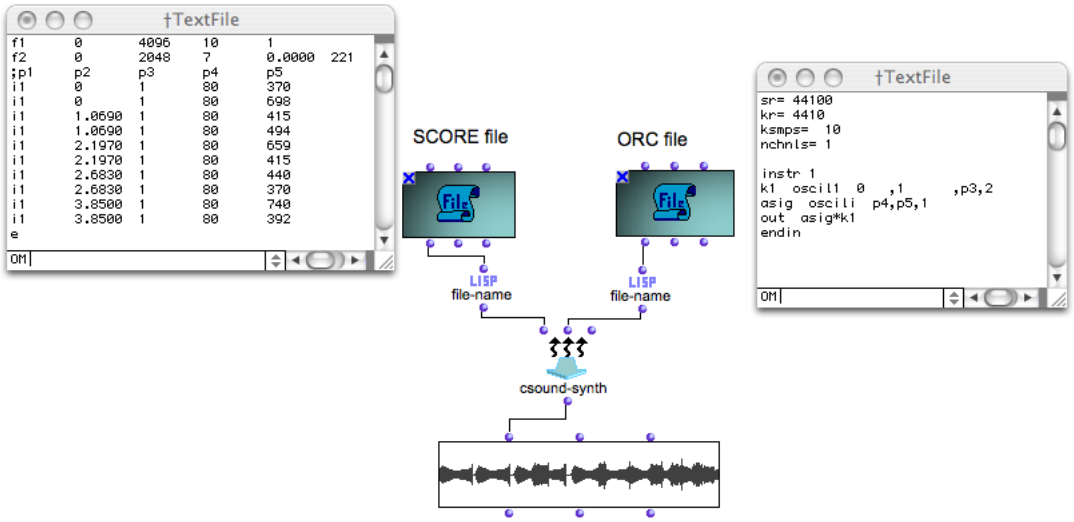


FIGURE 6.12: Synthèse CSOUND dans un patch.

En adaptant les fonctions de la bibliothèque OM2CSOUND (voir chapitre 5, section 5.2.3), on peut donc désormais relier directement les processus de création d'instruments et de partitions CSOUND que cette bibliothèque permet de réaliser avec la fonction `csound-synth`, et les rattacher à un objet `sound` dans la continuité d'un même programme visuel. C'est ce qui est illustré sur la figure 6.13.

Suivant l'exemple donné dans le chapitre 5 (figure 5.3), ce type de programme pourra être directement rattaché à des structures de données musicales symboliques, converties dans le langage de CSOUND par l'intermédiaire des outils de la bibliothèque.

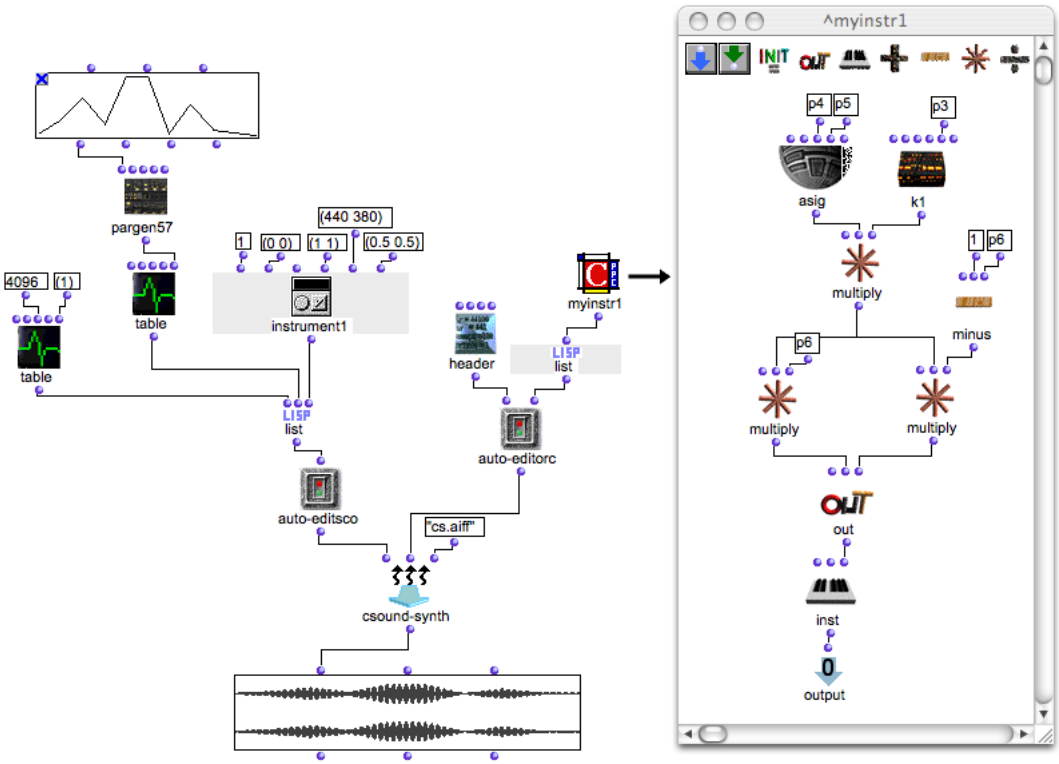


FIGURE 6.13: Synthèse CSOUND avec la bibliothèque OM2CSOUND.

Sur le modèle de `csound-synth`, d'autres fonctions sont donc définies, permettant d'exécuter les programmes CHANT (synthèse par FOF), SUPERVP (synthèse source/filtre, synthèse croisée), ou PM2 (synthèse additive) depuis OPENMUSIC. La figure 6.14 montre un exemple de synthèse croisée réalisée à partir de deux sons (enveloppe spectrale d'un son appliquée sur la partie harmonique du second) avec SUPERVP.

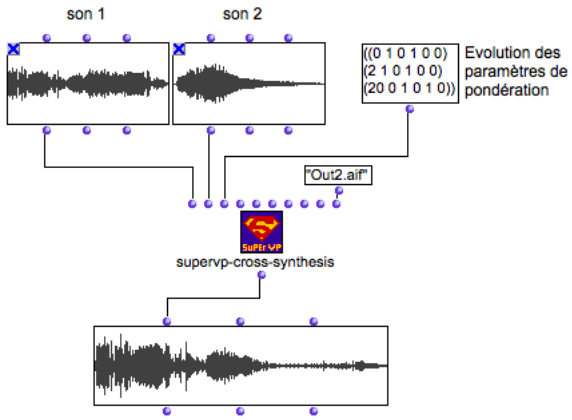


FIGURE 6.14: Synthèse croisée : un appel au programme SUPERVP.

6.4 La bibliothèque OM-SuperVP

La bibliothèque OM-SuperVP[2] a été créée suivant le modèle d'analyse/traitement/resynthèse proposé par les logiciels AudioSculpt/SuperVP (voir chapitre 2, sections 2.2 et 2.4.2). SuperVP permet d'appliquer aux sons analysés différents traitements, éventuellement simultanés et suivant un paramétrage évoluant dans le temps, avant de resynthétiser un son. C'est le principe qui est utilisé dans le séquenceur de traitements de AudioSculpt.

Sur le même principe, OM-SuperVP propose une fonction `supervp-processing`, pour laquelle doit être spécifié un son initial, une série de paramètres d'analyse (taille de la FFT, taille, forme et *overlapping* de la fenêtre d'analyse TFCT, etc.) ainsi qu'un ou plusieurs traitements. Les traitements disponibles sont représentés par différentes boîtes. Parmi ceux-ci, on trouvera les fonctions de *time stretching*, *pitch shifting*, transposition, filtre passe-bande, filtre par formants, *break-point filter*, *clipping*, et *freeze* (figure 6.15).

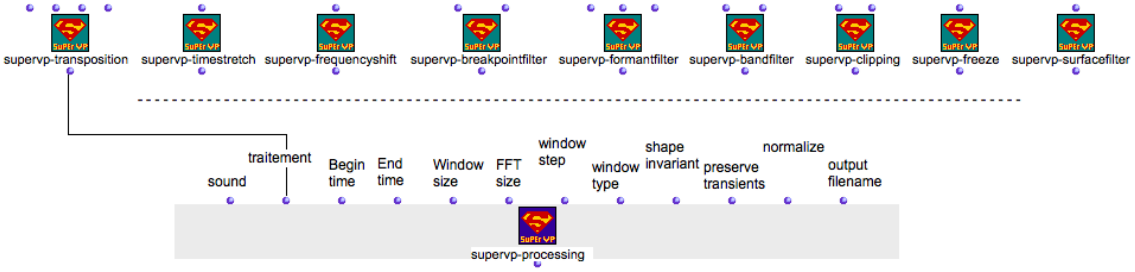


Figure 6.15: Bibliothèque de traitements dans OM-SuperVP.

Les traitements peuvent être utilisés seuls, ou combinés, connectés à la fonction `supervp-processing` sous la forme d'une liste de traitements (voir figure 6.16).

Chacun de ces traitements doit être paramétré de façon particulière. Les paramètres sont spécifiables par de simples valeurs numériques (comme dans l'exemple de la figure 6.16) ou avec des données variant dans le temps spécifiées par des listes temps / valeur(s) (éventuellement importées de fichiers externes), ou même à l'aide de *breakpoint functions* (objet `bpf` dans OpenMusic). La figure 6.17 illustre ce dernier cas avec l'utilisation d'une `bpf` pour la spécification du facteur d'étirement dans un traitement de *time stretching*.

Ces paramètres peuvent ainsi être générés par des algorithmes développés en amont des processus dans OpenMusic. On pourra, par exemple, utiliser pour cela les fonctions de la librairie OM-AS présentée dans le chapitre 5 (section 5.2.3), qui permettent de générer des fichiers de paramètres pour SuperVP. Sur la figure 6.18, un facteur de transposition est calculé et formaté à partir d'une mélodie spécifiée sous forme d'un objet `chord-seq`, et directement appliqué en tant que paramètre dans le traitement d'un objet `sound`.

[2]OM-SuperVP a été réalisé en collaboration avec Jean Lochard, du département Pédagogie de l'Ircam.

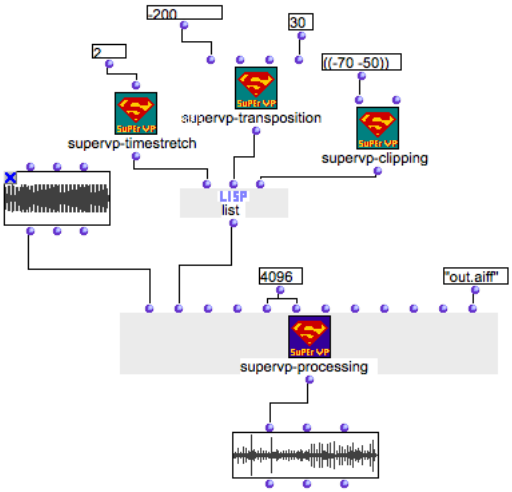


FIGURE 6.16: Application simultanée des traitements *time stretch*, *transposition*, et *clipping* avec OM-SUPERVP.

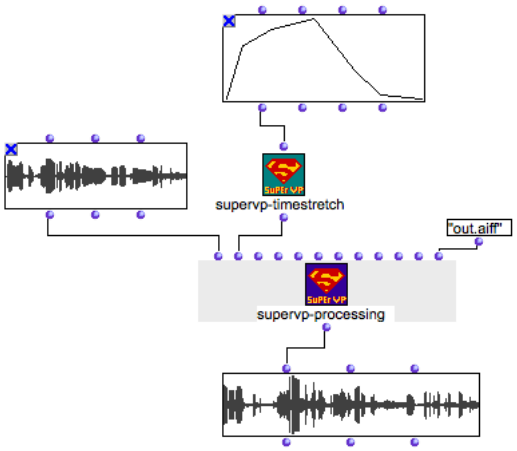


FIGURE 6.17: Paramétrage dynamique d'un traitement SUPERVP à l'aide d'une `bpf` (évolution dans le temps du facteur d'étirement/compression).

La figure 6.19 est un autre exemple de mise en relation des fonctions de traitement de la bibliothèque avec les structures de données musicales. Un rythme est utilisé pour déterminer les ratios qui permettront d'adapter le facteur de *time stretching*, afin de faire correspondre des segments déterminés dans le fichier audio initial avec ce rythme.

Enfin, les outils de programmation visuelle peuvent être mis à profit pour réaliser des opérations de traitements itératifs. Un son, ou une banque de sons contenus dans un dossier peuvent être traités itérativement grâce aux structures de contrôle `omloop`, par exemple, afin de générer d'autres ensembles de fichiers audio créés par synthèse selon des paramètres fixes ou générés dynamiquement durant l'itération. La figure 6.20 illustre ce type de procédé avec un son transposé plusieurs fois suivant les intervalles d'un accord.

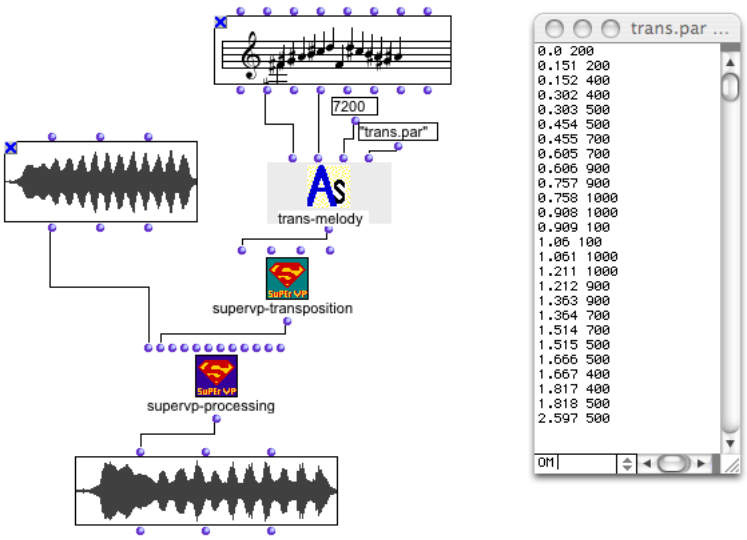


FIGURE 6.18: Utilisation de OM-SUPERVP en association avec la librairie OM-AS pour la génération de paramètres de transposition : le fichier de paramètres décrivant l'évolution du facteur de transposition (visible à droite) est calculé par la fonction `trans-melody` à partir d'une séquence musicale. Il est ensuite directement transféré à SUPERVP avec les fonctions `supervp-transposition` et `supervp-processing`.

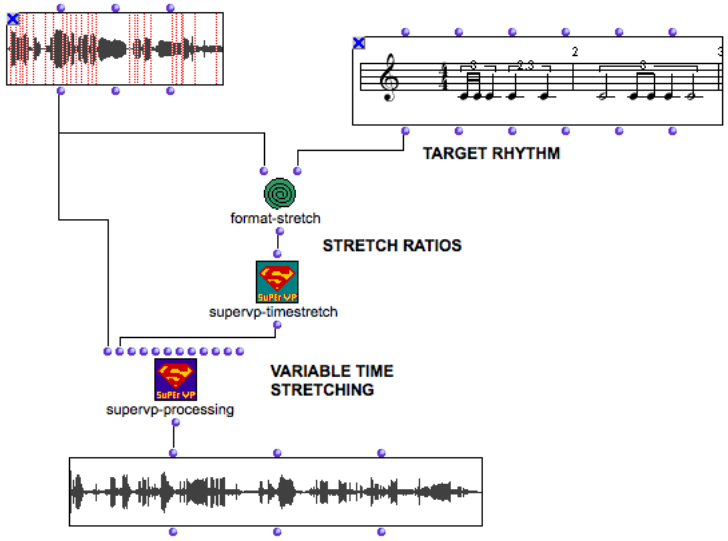


FIGURE 6.19: Traitement d'un signal audio par *time stretching*. L'itération `format-stretch` (dont le contenu n'est pas détaillé ici) calcule des rapports entre les durées des segments du fichier audio et celles du rythme donné à droite. Ces valeurs sont utilisées pour paramétrer la fonction `supervp-timestretch`, de sorte que les mêmes segments reproduisent le rythme donné dans le son synthétisé.

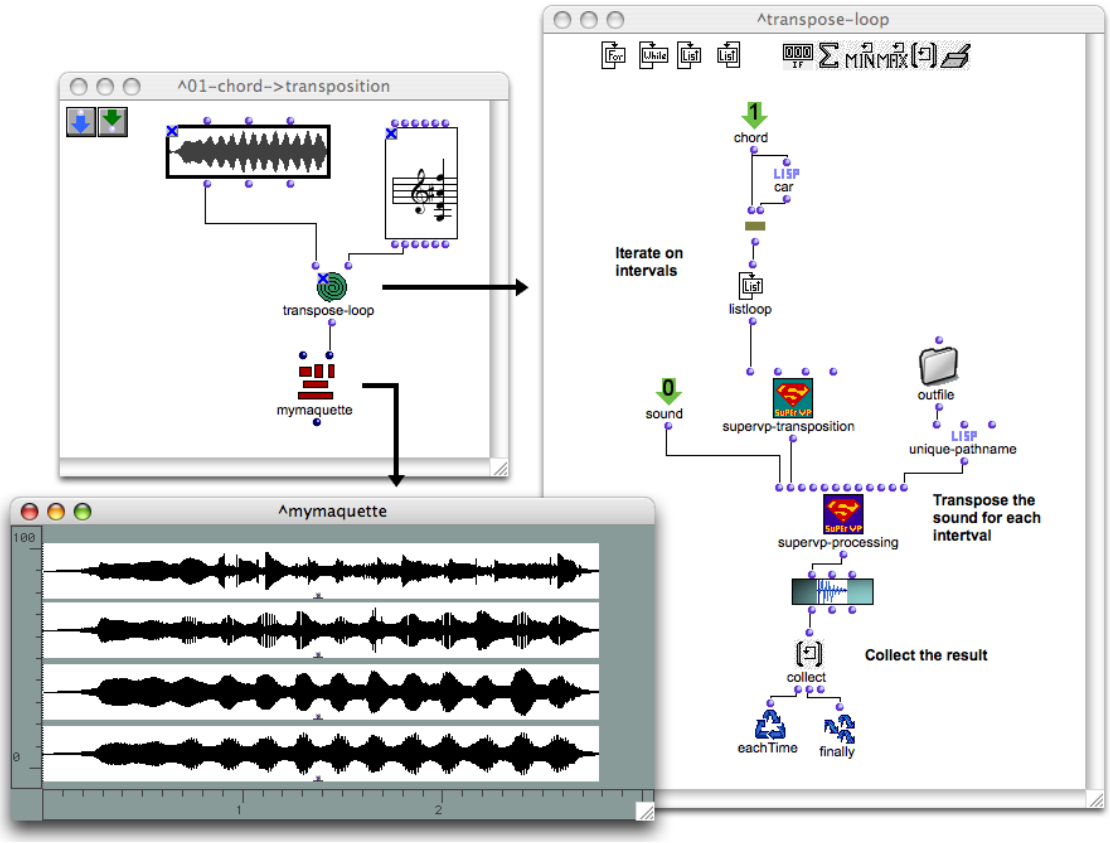


FIGURE 6.20: Traitement itératif d'un fichier son. La boucle `transpose-loop` réalise une itération sur la liste des intervalles de l'accord pour déterminer les valeurs correspondantes du facteur de transposition. A chaque itération un son transposé est créé, et collecté dans une maquette (visible en bas). Les 4 sons visibles dans la maquette correspondent ainsi aux 4 notes de l'accord de référence.

6.5 Conclusion

L'utilisation des outils de programmation que nous avons présentés dans ce chapitre permet de lier le traitement et la synthèse sonore à des processus plus ou moins complexes mis en œuvre dans OPENMUSIC et aux structures de données musicales de l'environnement. L'approche fonctionnelle unifie ainsi les différents processus dans une même logique de calcul et dans une même exécution fonctionnelle.[3] L'aspect visuel facilitera également l'insertion de ces outils dans un contexte compositionnel. Hans Tutschku décrit par exemple dans [Tutschku, 2008] l'utilisation des bibliothèques OM-SUPERVP et OM-AS pour la composition de sa pièce *Distance liquide*.

[3]Seule l'utilisation fréquente de références à des fichiers externes peut provoquer des effets de bord dans ce calcul (un même fichier modifié à différents endroits dans ce calcul).

Les mécanismes d'abstraction favoriseront ensuite la mise en relation des processus de traitement et de synthèse sonore avec les processus compositionnels et les structures musicales de plus haut niveau développés dans le même environnement. Sur la figure 6.21, on voit un patch similaire à celui de la figure 6.13, transformé en une fonction à deux entrées (une enveloppe dynamique et une séquence d'accords) et produisant un son. Cette fonction est appliquée dans un autre patch, symbolisant un niveau d'abstraction supérieur dans un processus de composition, dans lequel elle est utilisée avec des structures de données symboliques.

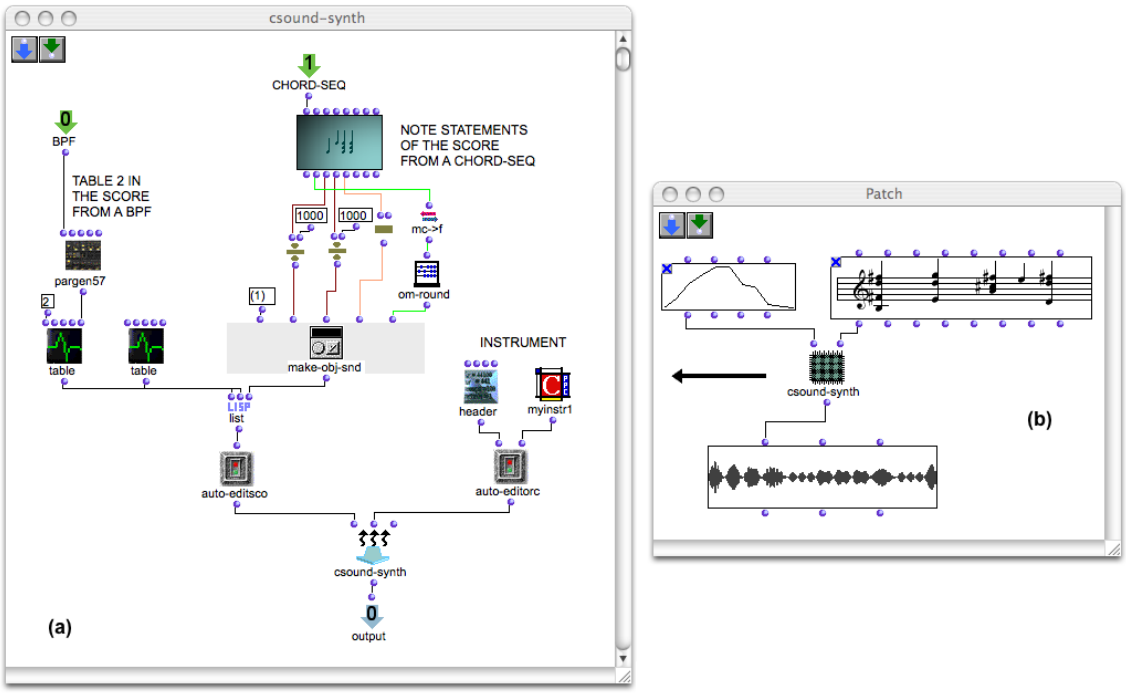


FIGURE 6.21: (a) Définition d'un programme de synthèse dans un patch OPENMUSIC et (b) utilisation de celui-ci en abstraction fonctionnelle. Le patch *csound-synth* est considéré comme une fonction, prenant en arguments une séquence de notes (`chord-seq`) et une `bpf` spécifiant une enveloppe dynamique, et renvoyant un fichier audio synthétisé.

Dès lors, la structure fonctionnelle des programmes permettra de faire face à la complexité des processus en maintenant une hiérarchie entre les abstractions musicales et les processus de bas niveau.

Chapitre 7

Descriptions sonores de bas niveau : formats de données et protocoles de transfert

Dans un environnement comme OpenMusic, privilégiant le calcul symbolique sur le traitement du signal, le passage par des fichiers ou des protocoles de communication est nécessaire du moment qu'il s'agit de transmettre à un programme de synthèse ou de rendu sonore des données de bas niveau formatées par le système.

Dans des situations prédéterminées (par exemple celles que nous avons vues dans le chapitre précédent), ce passage est réalisé par des fonctions spécifiques. C'est le cas lorsque l'on écrit des fichiers pour Csound, ou même lorsque l'on envoie des données provenant de séquences musicales à des synthétiseurs par l'intermédiaire du protocole MIDI et du format MIDIFile.

Dans une optique plus générale, pour aider l'utilisateur à mettre en place ses propres processus, des outils de programmation visuelle permettent de travailler sur le formatage, l'écriture et la transmission de ces données. Le traitement des données formatées dans les formats standards, et le contrôle des modalités de transfert et de communication sont deux aspects complémentaires permettant ce traitement des données musicales et/ou sonores de bas niveau.

7.1 Descriptions sonores de bas niveau

Dans les travaux compositionnels ayant trait à la synthèse ou l'analyse de sons, on passe régulièrement par la manipulation de descriptions sonores. Le terme de *description sonore* que nous utilisons se réfère à tout ensemble de données (a priori de bas niveau) utilisées pour représenter un son en fonction d'un formalisme ou d'un modèle donné. Il pourra s'agir de données issues de l'analyse d'un son (une FFT, un ensemble de partiels, des paramètres formantiques, etc.) ou destinées au paramétrage d'un synthétiseur.

Pour un système de composition assistée par ordinateur, la manipulation de données de description sonore représente un problème d'une part du point de vue compositionnel, puisqu'il s'agit d'embrasser une grande quantité d'information à la fois, ce qui implique la nécessité d'outils de visualisation, d'édition, voire de filtrage ou de sélection de ces données, mais également au niveau informatique, avec les problèmes de stockage, de mémoire occupée, de compatibilité et d'échange inter-applications. Ces données circulent intensivement dans le système de composition, et dès lors que nous avons choisi de proposer un système ouvert à l'utilisateur sur les différents niveaux de représentation et d'abstraction du matériau sonore et musical, il est nécessaire de fournir des outils de programmation permettant de les manipuler et de contrôler cette circulation. Les formats de stockage et de transfert peuvent alors fournir des cadres standardisés pour la représentation des descriptions sonores.

7.2 Lecture et écriture de fichiers

L'objet `file-box` est une extension de `omloop` permettant de réaliser des itérations avec un accès en lecture et en écriture sur un fichier. Il constitue une première modalité de contrôle du transfert de données et de la communication entre différents systèmes et composants des processus compositionnels.

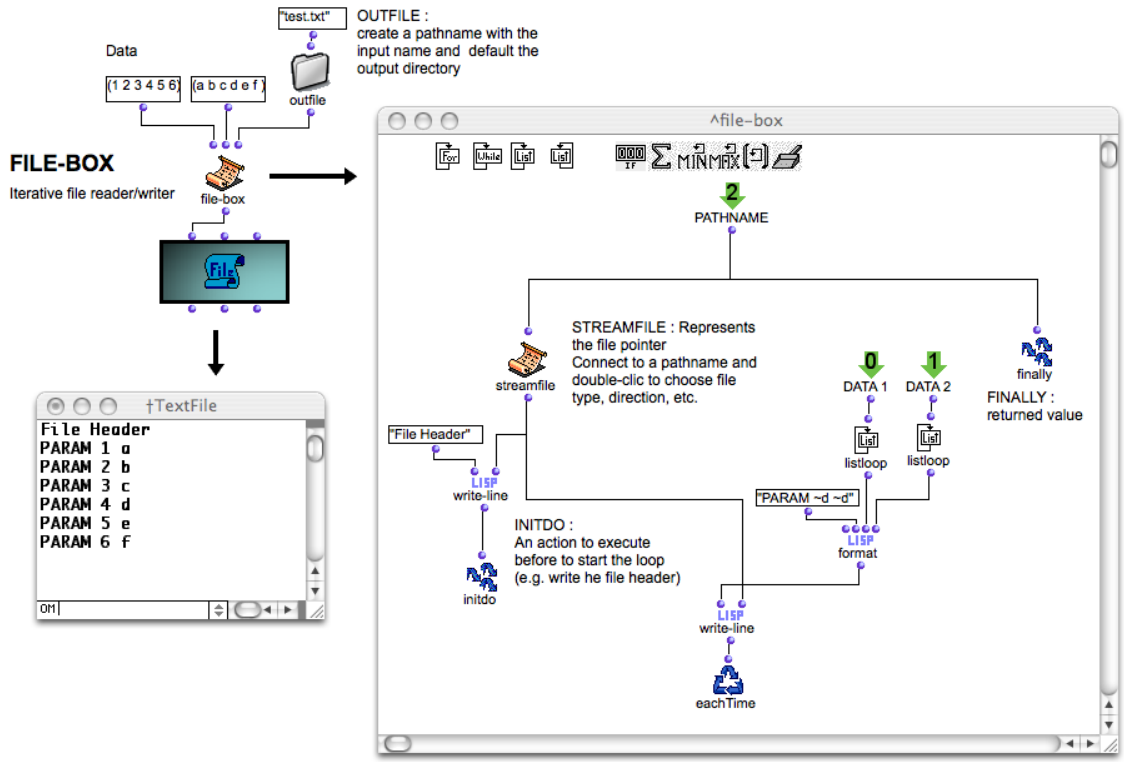


FIGURE 7.1: Un patch de type `file-box` : formatage et écriture de données dans un fichier.

Dans l'éditeur d'un `file-box` (sur la figure 7.1), la boîte (`streamfile`) représente ce flux d'entrée/sortie ; elle est initialisée avec un chemin accédant au fichier destiné à être lu ou écrit. Ce pointeur sur le fichier peut ensuite être utilisé pour les opérations de lecture ou d'écriture dans le programme (ou l'itération) décrite dans le patch `file-box`. Les opérations de lecture et d'écriture se font en principe au format texte ASCII par l'intermédiaire des fonctions LISP standard (`write-line`, `write-char`, etc..) mais pourront être étendues à d'autres formats (voir section 7.5).

Cet outil permettra ainsi à un utilisateur de réaliser ses propres entrées et sorties du système par l'intermédiaire de fichiers dans lesquels il est en mesure de lire ou écrire ce qu'il souhaite et au moment où il le souhaite dans le calcul.

7.3 Modèle instrumental : le format MIDI

MIDI (*Musical Instrument Digital Interface*) [MIDI Manufacturers Association, 2001] définit à la fois un format de données et un protocole de communication entre instruments électroniques, permettant de faire passer des messages entre dispositifs de contrôle, instruments électroniques et autres séquenceurs musicaux. La communication est établie par l'envoi d'évènements de différents types sur des ports MIDI. Parmi les types les plus utilisés, on trouve des évènements appelés *NoteOn* et *NoteOff*, qui indiquent les début et fin d'évènements sonores (correspondant à l'appui et au relâchement virtuels d'une touche sur un clavier), ou encore des évènements de type *ControlChange*, qui permettent de spécifier une valeur pour différents contrôleurs génériques (par exemple volume, panoramique, *pitchbend*, etc.), ou dépendant du récepteur. Les évènements sont récupérés par les dispositifs ou programmes connectés aux ports correspondants et utilisés pour paramétrer la synthèse (la norme General MIDI constitue un standard spécifiant des types de programmes et de contrôles prédéfinis).[1]

La création de données MIDI constitue ainsi un moyen de contrôle simulant un modèle instrumental, à partir de notes et de contrôleurs d'expression élémentaires). Elle est permise dans OPENMUSIC par l'intermédiaire du système MIDISHARE [Orlarey et Lequay, 1989], qui gère l'instanciation d'évènements et leur ordonnancement temporel. Les objets musicaux classiques (accords, et autres ensembles de notes) sont joués de cette manière (conversion en évènements MIDI et transmission sur un port MIDI).

Dans l'objectif de permettre la création de structures dépassant le cadre de ces objets, nous avons développé dans OPENMUSIC une série de classes permettant la manipulation des données MIDI dans les programmes visuels. La structure de description élémentaire est la classe `MIDIEvent`, qui représente un évènement MIDI. Les paramètres (*slots*) de cette classe sont les informations nécessaires à la construction du message MIDI correspondant : type d'évènement, valeurs, date, etc. L'objet `MIDIEvent` créé par instanciation de

[1] On trouvera dans [Loy, 1985] ou [Letz, 2004] des descriptions plus approfondies du format MIDI.

cette classe peut être considéré comme un objet musical dans OpenMusic. Il peut être édité, écouté (ou simplement envoyé sur un port MIDI : un évènement MIDI ne correspond pas nécessairement à l'émission d'un son), ou encore utilisé dans une maquette OpenMusic. A partir de cette structure, d'autres classes sont construites, représentant des réalités musicales plus conséquentes. Elles correspondent chacune à un certain ensemble d'évènements MIDI : la classe `EventMIDI-seq` représente une séquence d'évènements, et les classes de contrôleurs regroupent soit un ensemble d'évènements de contrôle simultanés (classe `midi-mix-console`), soit une séquence d'évènements correspondant à l'évolution dans le temps d'un contrôleur continu (classe `midicontrol`, extension de la classe `bpf`).

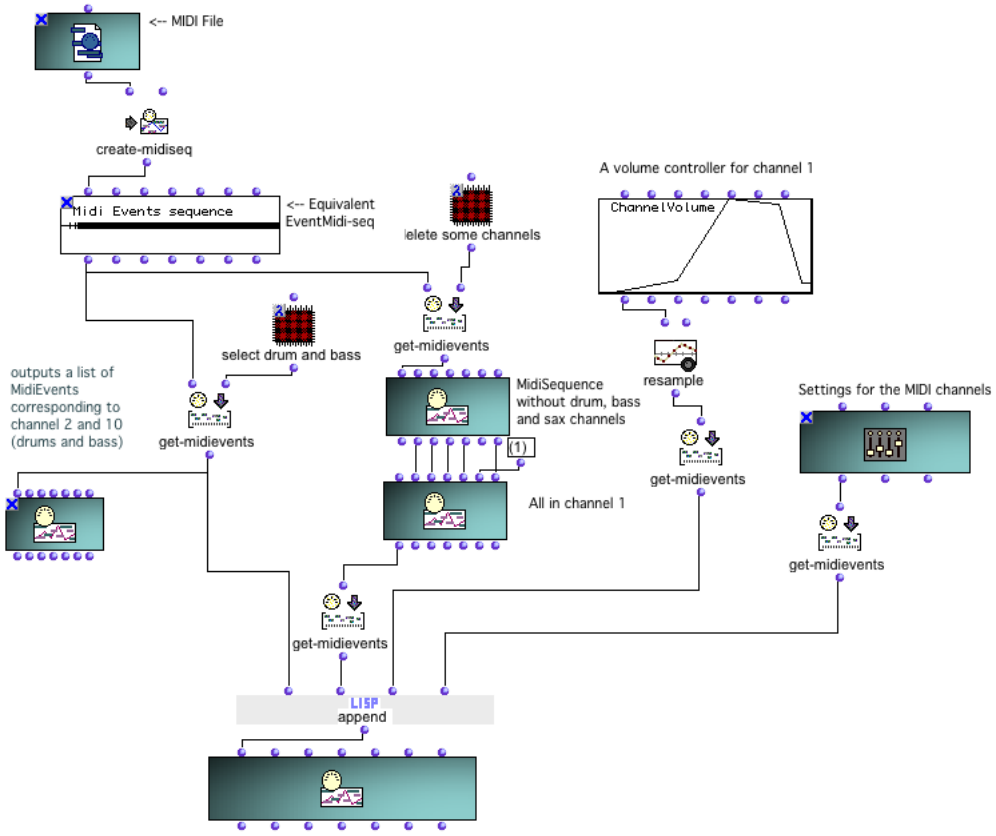


Figure 7.2: Manipulation de données MIDI dans OpenMusic.

Différentes fonctions sont disponibles pour un traitement précis des données de contrôle (filtrage, transformations, etc.) au format MIDI, permettant de développer graphiquement des programmes de traitement ou de génération de flux de données dans ce format (voir par exemple la figure 7.2).

Si la norme MIDI reste aujourd'hui très généralement utilisée dans les applications musicales, nous avons vu cependant qu'elle était surtout adaptée à un modèle de contrôle de type instrumental, que nous souhaitons dépasser dans l'utilisation généralisé des techniques de synthèses sonores. MIDI comporte également d'importantes limitations : impossibilité

d'encoder des informations musicales plus structurées (comme les rythmes, les relations entre les évènements musicaux, etc.), précision fréquentielle limitée à une division en demi-tons (soit l'équivalent d'un clavier classique, alors qu'il est de plus en plus fréquent dans la musique électroacoustique d'utiliser les micro-intervalles, voire des systèmes de hauteurs absolues, indépendantes de toute subdivision), limitation à 16 canaux, sont autant de raisons qui laissent présager que ce format soit remplacé un jour. De nouveaux formats (par exemple autour de la norme MPEG [Vercoe et Scheirer, 1998] [Ronfard, 2004]) tentent de remédier à certaines de ces limitations pour une description plus complète des objets sonores ou musicaux. Dans le cas qui nous intéresse (transfert et encodage des données de description sonores généralisées) le format SDIF, que nous étudierons dans les sections suivantes, se présente comme un standard mieux adapté aux pratiques de la musique électroacoustique.

7.4 Descriptions sonores généralisées : SDIF/OSC

Si le format MIDI s'est imposé comme standard pour les descriptions proches du système instrumental (notes, etc.), il est plus compliqué d'aboutir à un tel consensus dans le domaine sonore en général. C'est un objectif visé avec le format SDIF, dont nous proposerons l'utilisation en tant que support générique des descriptions sonores dans notre environnement de composition [Bresson et Agon, 2004].

Le protocole de transfert OSC vise également des objectifs similaires.

7.4.1 Le format SDIF

Le format SDIF (*Sound Description Interchange Format*) a été développé conjointement par l'IRCAM et le CNMAT de Berkeley afin de disposer d'un standard ouvert et multi-plateformes permettant la codification, le stockage et l'échange de données de description sonore. Il permet de conserver ces données de manière à la fois uniforme et ouverte [Wright *et al.*, 1998] [Wright *et al.*, 1999] [Schwartz et Wright, 2000].

Ce format est utilisé et supporté par un nombre croissant d'applications liées à l'analyse et la synthèse sonore, parmi lesquelles DIPHONE, le synthétiseur CHANT (pour les analyses et synthèses par fonctions d'ondes formantiques – FOF), et les programmes d'analyse ADDAN (additive) et RESAN (modèles de résonances), AUDIOSCULPT/SUPERVP/PM2 (pour les données de TFCT, de suivi de fréquence fondamentale, les ensembles de partiels), MAX/MSP (notamment à travers la librairie FTM), le système CLAM/SMS de l'UPF Barcelona, SPEAR, l'environnement OPENSOUNDWORLD du CNMAT, etc.

Le caractère générique de ce format permettra donc d'unifier la représentation des différents types de descriptions sonores dans un environnement comme OPENMUSIC, et ainsi de les manipuler avec les mêmes outils. Il garantira également la compatibilité des données dans les éventuelles communications et échanges avec ces différentes applications.

SDIF se présente sous la forme d'une bibliothèque multiplateforme définissant une interface optimisée pour l'écriture et la lecture des données dans le format.[2] Une interface LISP que nous avons développée permet d'effectuer ces opérations par des appels externes à la bibliothèque. Les outils de programmation visuelle développés pour l'utilisation du format SDIF dans OPENMUSIC seront détaillés dans la section 7.5.

Structure du format

Un flux de données SDIF (appelé *stream*) est constitué d'une séquence de trames (appelées *frames*). Tous les types de données sont codés de façon uniforme dans ces *frames*, identifiées par une signature, qui en détermine justement le type, par une date permettant de gérer la dimension et l'ordonnancement temporels des données, et par un identifiant de flux (*streamID*). Nous avons donc une description évoluant dans le temps, incluant éventuellement des données hétérogènes sous forme de flux entrelacés (voir figure 7.3).

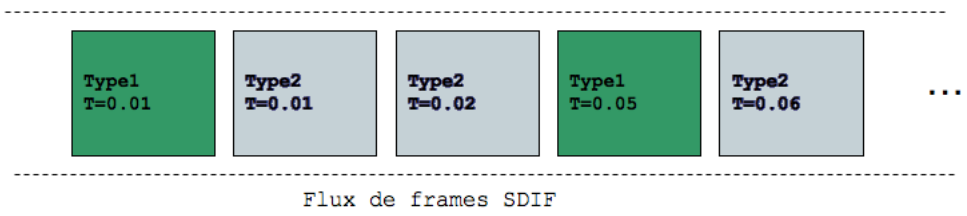


FIGURE 7.3: Une séquence de *frames* SDIF. Deux flux de types différents sont entrelacés.

Chaque *frame*, qui constitue donc un échantillon temporel du ou d'un des flux contenu(s) dans le fichier SDIF, contient à son tour une ou plusieurs *matrices* (également identifiées par des types), qui sont les spécifications de stockage élémentaires du format. Une *matrice* est un tableau bidimensionnel ayant pour première dimension (colonnes) les champs de description du son (par exemple : fréquence, amplitude, phase, pour une description spectrale) à l'instant spécifié par la *frame*. On dira qu'une matrice représente une structure, dont les colonnes sont les champs (*fields*), et les lignes sont les éléments de la description. La figure 7.4 montre la structure en profondeur d'un fichier SDIF.

En résumé, un fichier SDIF décrit de façon standardisée l'évolution de différents ensembles de données stockées sous forme de matrices. Les fonctions de la bibliothèque SDIF parcourent les fichiers par *frames*, puis récursivement par matrices dans lesquelles sont lues et écrites les données.

Types

Il existe des types de *frames* et de *matrices* prédéfinis dans les spécifications du format, correspondant aux principales descriptions sonores existantes (enveloppes spectrales,

[2] http ://www.ircam.fr/sdif/

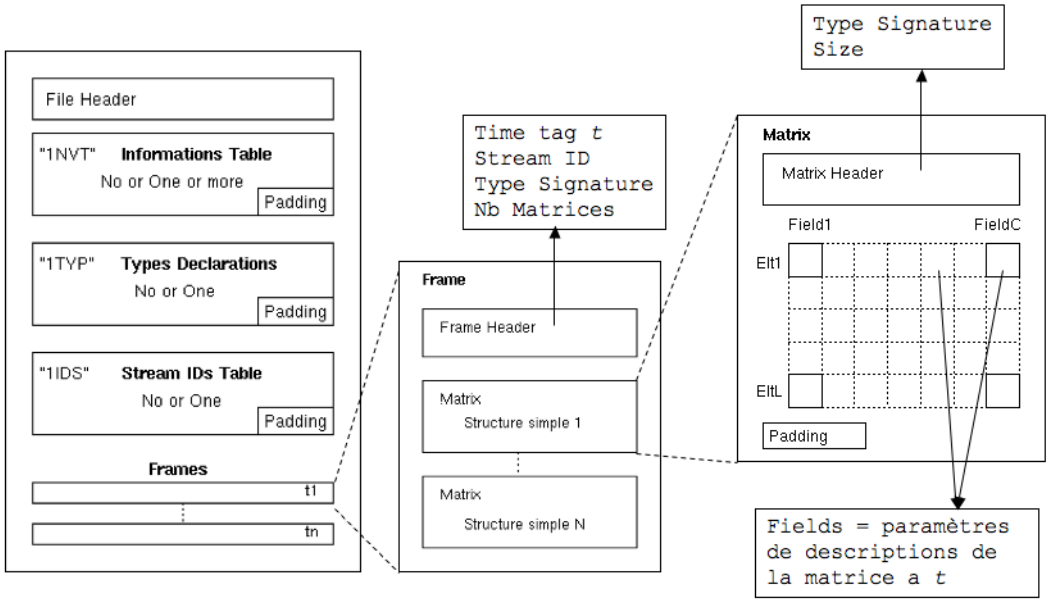
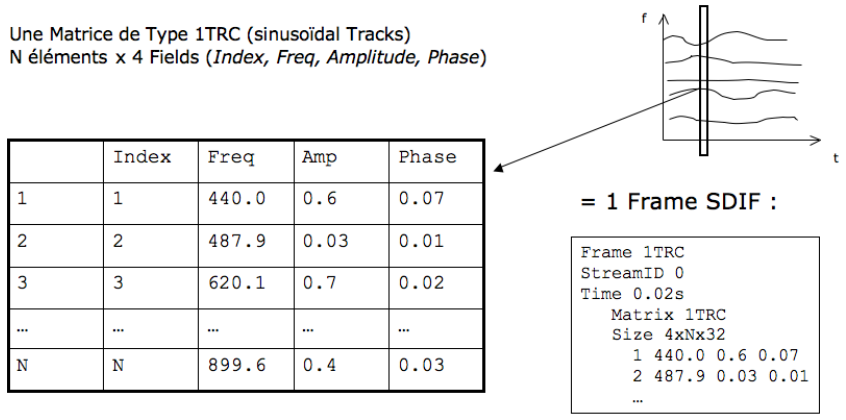


FIGURE 7.4: Structure d'un fichier SDIF.

filtres, résonances, FFT, estimation de fréquence fondamentale, énergie, modèles additifs, marqueurs, etc.) Ces types sont identifiés par une signature de la forme `"1xxx"`. Une application lisant un fichier SDIF peut ainsi parcourir la séquence de *frames* et ne considérer que les données contenues dans celles dont elle reconnaît (ou cherche à traiter) le type.

La figure 7.5 représente le codage d'une *frame* d'une description sonore additive (type `"1TRC"`) : à un instant donné sont spécifiées pour chaque partiel (identifié par le champ `index`) les valeurs de fréquence, amplitude, et phase.



	Index	Freq	Amp	Phase
1	1	440.0	0.6	0.07
2	2	487.9	0.03	0.01
3	3	620.1	0.7	0.02
…	…	…	…	…
N	N	899.6	0.4	0.03

FIGURE 7.5: Exemple d'une *frame* SDIF représentant une description additive.

Sur la figure 7.6, on a à présent une description de type suivi de fondamentale (type SDIF `"1FQ0"`). On n'a donc plus qu'une seule valeur à stocker à chaque instant. La même structure est cependant respectée. Sur cette figure nous avons ajouté la dimension temporelle par rapport à l'exemple précédent.

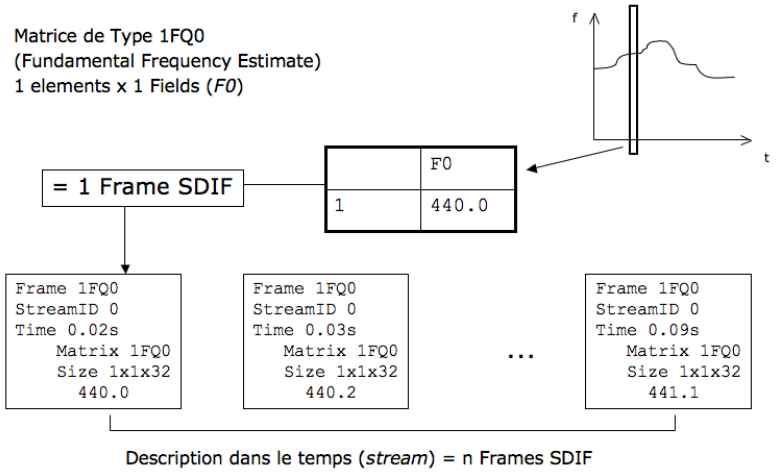


FIGURE 7.6: Exemple de *frames* SDIF représentant un suivi de fréquence fondamentale.

Enfin, la figure 7.7 représente une *frame* décrivant l'état d'un banc de FOF. Une succession de *frames* de ce type est par exemple utilisée pour paramétrer une synthèse FOF avec le synthétiseur CHANT. Dans ce cas, les *frames* (de type `"1FOB"`) contiennent chacune plusieurs matrices de types différents.

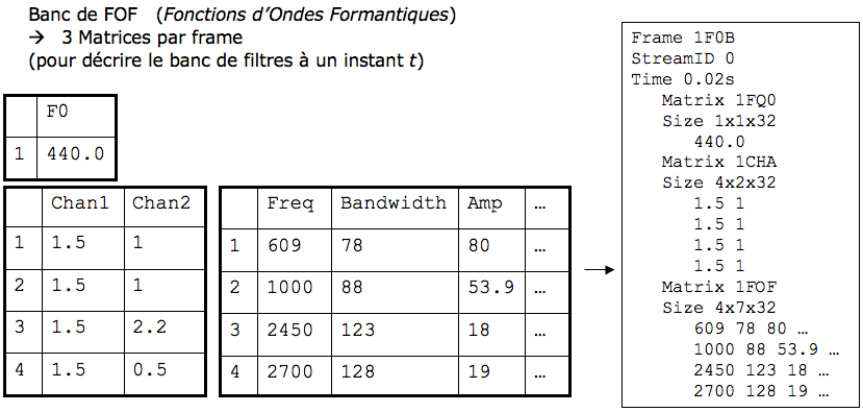


FIGURE 7.7: Exemple d'une *frame* SDIF représentant un banc de FOF.

L'un des objectifs de SDIF étant sa flexibilité, il est possible pour un utilisateur ou une application d'ajouter ses propres types de données, ou d'étendre les types existants par des nouveaux champs. Pour utiliser une *frame* d'un type donné dans un fichier SDIF, il faut que celui-ci soit défini soit parmi les types prédéfinis par le format, soit dans une table

de déclaration de types contenue dans le fichier lui-même (voir figure 7.4). Des données de types quelconques peuvent donc être codés dans un fichier SDIF selon le principe de la figure 7.8.

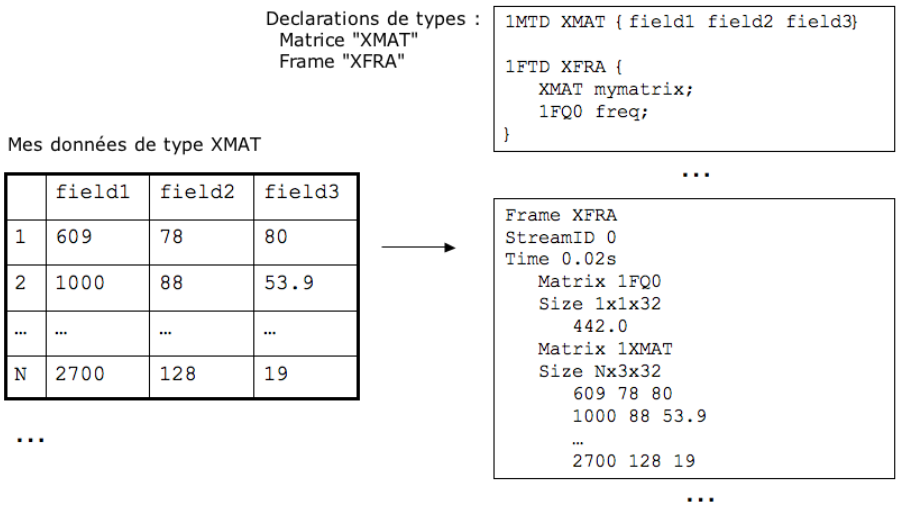


FIGURE 7.8: Utilisation de nouveaux types dans un fichier SDIF.

7.4.2 Protocole OSC

En ce qui concerne le transfert de données (autres que MIDI, qui est à la fois un format et un protocole de transfert), le protocole OSC (*Open Sound Control* [Wright et Freed, 1997] [Wright *et al.*, 2003]) semble se standardiser. Celui-ci permet de transmettre des données de manière libre et ouverte entre dispositifs ou systèmes de synthèse et environnements de contrôle. Encapsulé lui-même dans le protocole UDP, OSC est utilisé par un nombre croissant d'applications, d'environnements de synthèse ou d'informatique musicale en général.

Les structures OSC que nous avons implémentées dans le langage visuel OPENMUSIC sont limitées à l'évènement OSC (classe `OSCEvent`), et à la séquence d'évènements. Ceux-ci sont instanciés sur le modèle des évènements MIDI et peuvent être envoyés grâce à des fonctions d'émission et de réception via UDP.

OSC permet donc à un patch OPENMUSIC d'échanger des données musicales structurées ou de bas niveau avec des environnements de synthèse et de temps réel tels que PUREDATA ou MAX/MSP. Des évènements OSC placés dans une maquette permettent de mettre en relation l'exécution de celle-ci et le déclenchement programmé d'évènements dans ces environnements.

7.5 Outils pour la manipulation des descriptions sonores au format SDIF dans OpenMusic

En utilisant le format SDIF comme support des descriptions sonores de bas niveau, notre objectif est de considérer cette notion de description sonore comme un objet en soi, indépendant de sa nature et de son origine. Il sera alors possible d'utiliser des outils de traitements génériques sur une telle structure de données, et d'approcher la notion d'objet compositionnel qu'elle est susceptible de représenter dans les processus musicaux.

7.5.1 Structures de données SDIF

Un flux de *frames* SDIF est stocké dans un fichier. Comme la classe `sound`, la classe `SDIFFile` correspond à un fichier enregistré sur le disque dur. Elle est donc également instanciée à partir d'un *pathname*, comme le montre la figure 7.9.

Cette classe permettra de faire le lien avec les outils d'analyse et de synthèse sonore supportant le format SDIF, et correspondra alors à un objet intermédiaire utilisé pour stocker les données provenant ou destinées à ceux-ci. L'utilisation d'un fichier externe permet alors dans un premier temps de résoudre des problèmes d'espace et de stockage des données qui peuvent atteindre des tailles importantes.

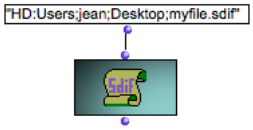


FIGURE 7.9: La classe `SdifFile`.

Afin de permettre la création et la manipulation de données au format SDIF dans OpenMusic, des classes spécifiques reconstituent de façon modulaire la structure d'un fichier (voir figure 7.10).

`SDIFMatrix` est un tableau à deux dimensions représentant une matrice SDIF, dont le type est spécifié sous forme d'une chaîne de caractères. Sur le modèle des matrices OpenMusic (qui seront détaillées dans le chapitre 9), un slot définit le nombre de lignes (*éléments* de matrices SDIF), et les colonnes (*fields*) sont ajoutées selon le type de données. `SDIFFrame` permet de modéliser une *frame* SDIF d'un type donné et localisée dans le temps, contenant une ou plusieurs matrices ; `SDIFStream` rassemble une liste de *frames* dans une description temporelle ; et `SDIFType` représente une déclaration de type. Enfin, `SDIFBuffer` rassemble des *streams* et/ou des *frames* SDIF et des déclarations de types dans une structure intermédiaire complète représentant l'intégralité d'un fichier SDIF.

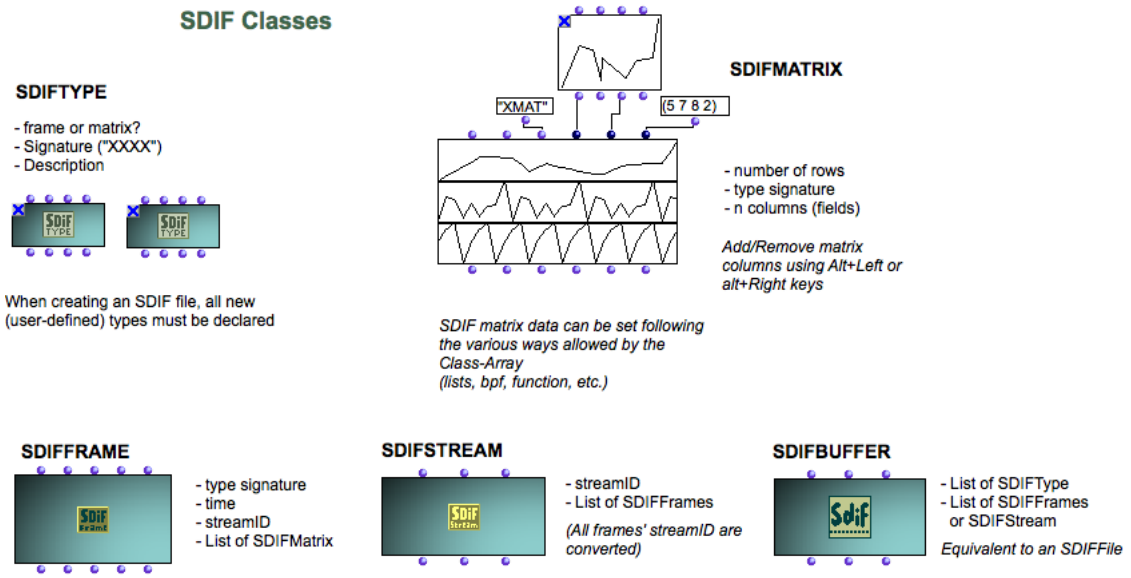


FIGURE 7.10: Classes SDIF dans OPENMUSIC.

7.5.2 Ecriture des données

Une structure SDIF créée par l'assemblage et l'ordonnancement des objets mentionnés ci-dessus peut ensuite être écrite dans un fichier par la fonction générique `save-sdif` :

- `save-sdif(SDIFBuffer)` = `save-sdif(`*liste de SDIFTypes, SDIFFrames et SDIFStreams*`)` ;
- `save-sdif(SDIFType)` = déclare les types dans le fichier ;
- `save-sdif(SDIFStream)` = `save-sdif(`*liste de SDIFFrames*`)` ;
- `save-sdif(SDIFFrame)` = écrit l'entête (*frame header*) et `save-sdif(`*liste de SDIFMatrices*`)` ;
- `save-sdif(SDIFMatrix)` = écrit l'entête (*matrix header*) et écrit les données de la matrice.

La figure 7.11 montre un exemple d'une structure SDIF créée de cette manière dans un patch.

Les opérations de lecture et d'écriture dans l'itérateur `file-box` (voir section 7.2) sont également disponibles avec le format SDIF (`sdif-write-frame`, `sdif-write-header`, etc.), ce qui permet de disposer d'un accès personnalisé sur les fichiers dans ce format (voir figure 7.12).

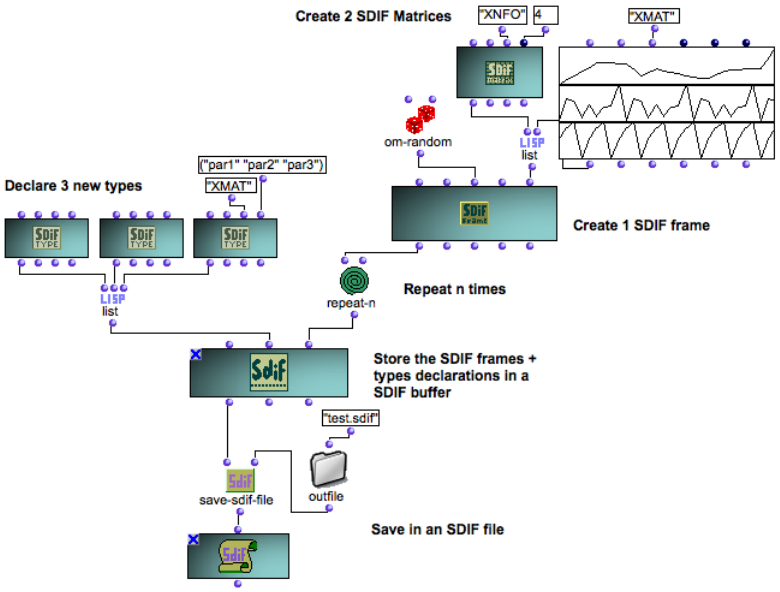


FIGURE 7.11: Création de structure de données SDIF dans un patch OPENMUSIC et sauvegarde dans un fichier.

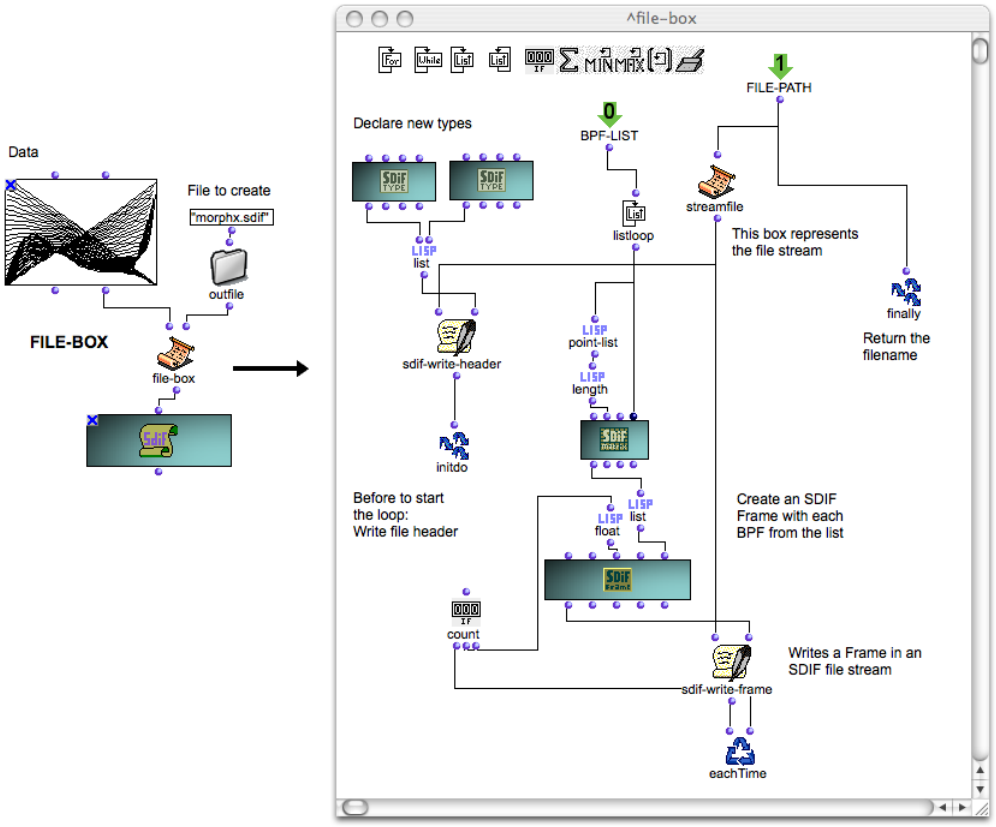


FIGURE 7.12: Formatage et écriture de données au format SDIF dans un fichier avec l'itérateur `file-box`.

7.5.3 Lecture et interprétation des données

Lorsqu'un fichier SDIF est chargé dans OpenMusic (instanciation de la classe `SDIFFile`), une lecture globale et superficielle est effectuée (lecture des en-têtes des *frames* et matrices uniquement) afin de constituer une "carte" structurelle de celui-ci, informant sur son contenu. On connaît alors les positions des *frames* et matrices successives dans le fichier, ainsi que la nature des données qu'elles contiennent. Une inspection globale peut donc être effectuée depuis un patch, sans nouvel accès au fichier.

L'accès aux données numériques se fait à partir de cette information de manière explicite ou non (du point de vue de l'utilisateur). On pourra dans le premier cas (figure 7.13) demander les valeurs contenues dans la colonne n de la matrice m de la *frame* f du fichier.

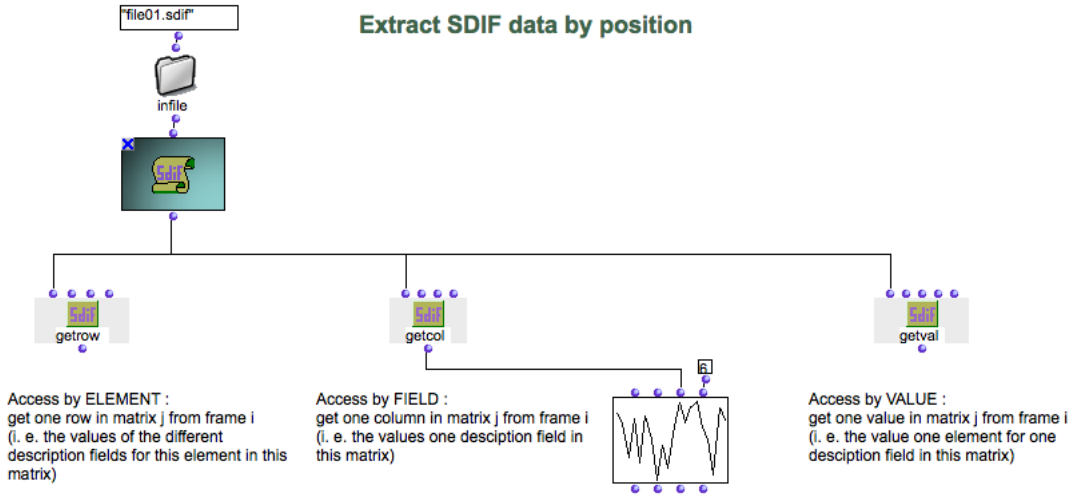


Figure 7.13: Lecture de données localisées dans un fichier SDIF.

Une autre manière d'accéder aux données contenues dans le fichier est permise par la boîte `getsdifdata`. Celle-ci permet de spécifier un type de données visé (signature des types de *frame* et de matrice), ainsi que, optionnellement, le champ de données, une fenêtre temporelle et une sous-sélection des éléments. Le résultat est renvoyé sous forme d'une liste de valeurs, qu'il est possible de traiter dans le patch et/ou de traduire sous forme d'objets divers (figure 7.14).

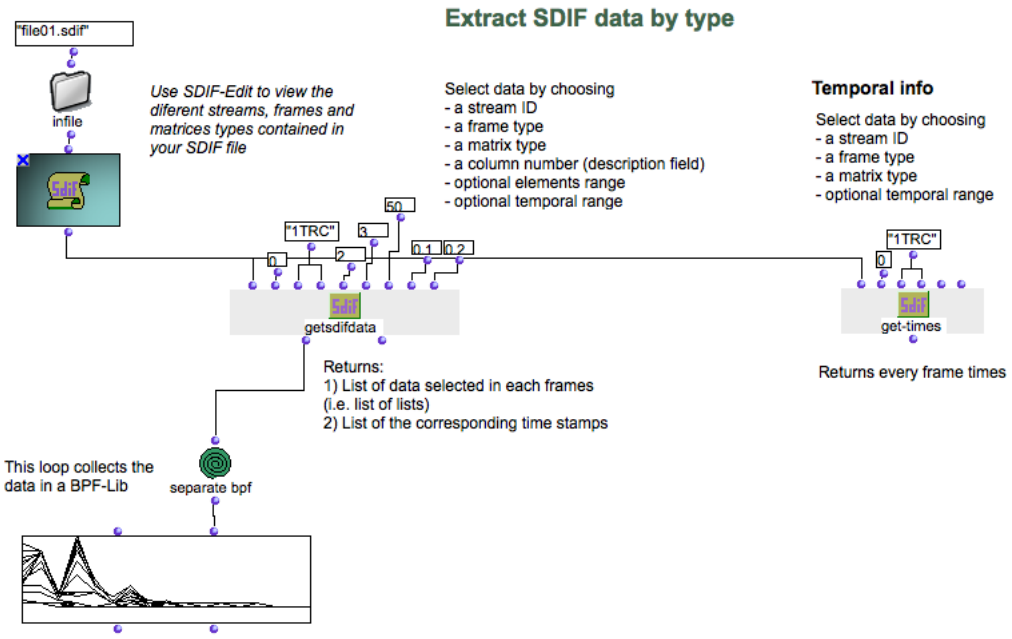


FIGURE 7.14: Lecture de données identifiées par types dans un fichier SDIF.

7.5.4 Visualisation : SDIF-Edit

Afin de pouvoir utiliser les fichiers SDIF dans l'environnement de composition, la possibilité de visualiser les données qu'il renferme est nécessaire. Elle permettra un contrôle et une évaluation des résultats des processus d'analyse, et constituera une étape préalable au traitement de ces données. La grande quantité d'information contenue dans les analyses et autres descriptions sonores implique en effet des méthodes d'extraction et de réduction nécessitant une connaissance en précision de la structure et de la localisation des données.

SDIF-EDIT [Bresson, 2003][3] est une application permettant la visualisation et l'édition de fichiers SDIF. Initialement, il s'agit d'une application indépendante écrite en C++ avec OPENGL, qui a été intégrée dans OPENMUSIC en tant qu'éditeur pour les objets `SDIFFile`.

Les problématiques de la représentation des données SDIF sont diverses ; la principale étant la flexibilité du format de laquelle résulte la nécessité de s'abstenir de tout a priori quant au contenu (nature des données, interprétation, taille) du fichier que l'on va visualiser. SDIF-EDIT propose ainsi une représentation générique (ne tenant pas compte du type des données visualisées), lui assurant de pouvoir traiter n'importe quel fichier. Cette représentation est une "grille" tridimensionnelle qui permet une visualisation des différents paramètres et de leur évolution temporelle (voir figure 7.15). Dans cet espace, l'utilisateur peut se déplacer, se focaliser sur des zones d'intérêt, éditer les données selon diverses procédures. Cette représentation générale en 3D a également été complétée par

[3] http ://recherche.ircam.fr/equipes/repmus/bresson/sdifedit/sdifedit.html

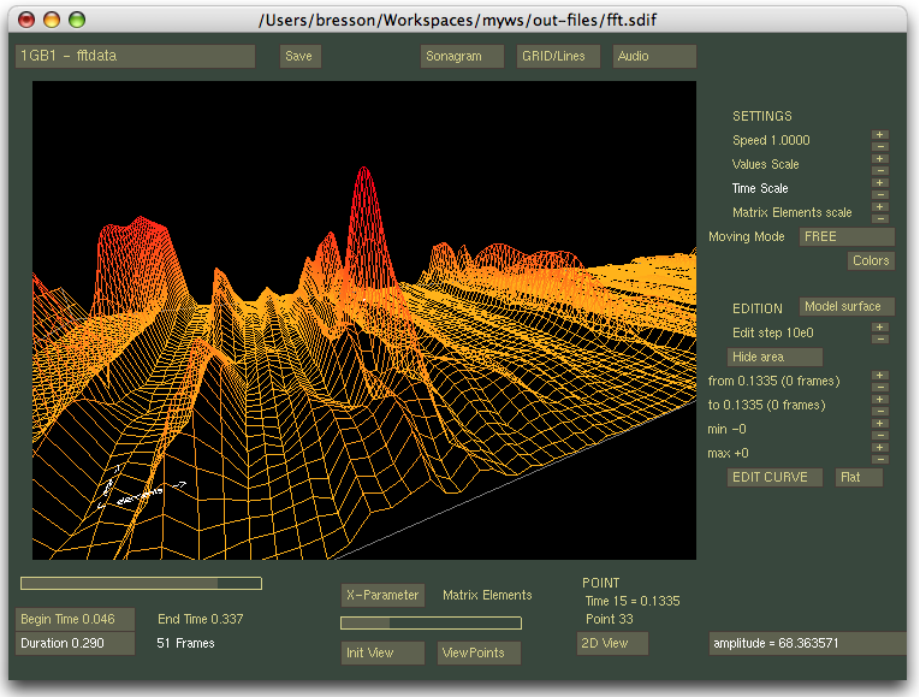


FIGURE 7.15: SDIF-EDIT : une application pour la visualisation de fichiers au format SDIF.

une navigation de plus haut niveau permettant la sélection d'un type de données parmi celles que peut renfermer un même fichier, et par diverses propositions de représentations alternatives, parfois plus adaptées à certains types de données (courbes, mini-sonagramme, etc.)

Si les possibilités de traitement des données sont limitées à l'intérieur de cet éditeur (tout au plus une modification manuelle des valeurs), il semble que l'outil de simple visualisation qu'il représente s'avère utile dans le contrôle des processus d'analyse et de synthèse sonore. On pourrait également s'avancer en supposant que la visualisation et la navigation en trois dimensions dans cet espace virtuel de données, au moment d'un passage du domaine du signal vers celui du symbolique, puisse inspirer des choix esthétiques basés sur une appréciation visuelle.

Les caractéristiques et fonctionnalités de SDIF-EDIT sont traitées plus en détail dans [Bresson et Agon, 2004].[4]

[4]L'application est distribuée et documentée par l'intermédiaire du forum IRCAM [Bresson, 2006a]. Elle a reçu en 2006 le prix spécial du jury au concours de logiciels musicaux LOMUS organisé par l'Association Française d'Informatique Musicale.

7.6 Conclusion

Nous avons présenté dans ce chapitre un certain nombre d'outils permettant la manipulation de données de description sonores de bas niveau, notamment à l'aide des formats standards. En particulier, le format SDIF nous semble proposer une généralité suffisante pour lier ces données relatives à des processus d'analyse et de synthèse sonore avec des processus musicaux de plus haut niveaux.

La figure 7.16 illustre cette idée. Il s'agit d'un processus de synthèse par FOF utilisant des outils évoqués dans le chapitre précédent (une fonction de synthèse appelant le synthétiseur CHANT). Des données musicales symboliques y sont converties en paramètres de synthèse sous la forme de structures de données SDIF : l'état d'un banc de FOF à différents moments déterminés est ainsi décrit par des courbes, et la fréquence fondamentale est issue d'une séquence musicale. Ces données formatées sont ensuite transférées au système de synthèse par l'intermédiaire d'un fichier SDIF pour produire un signal numérique.

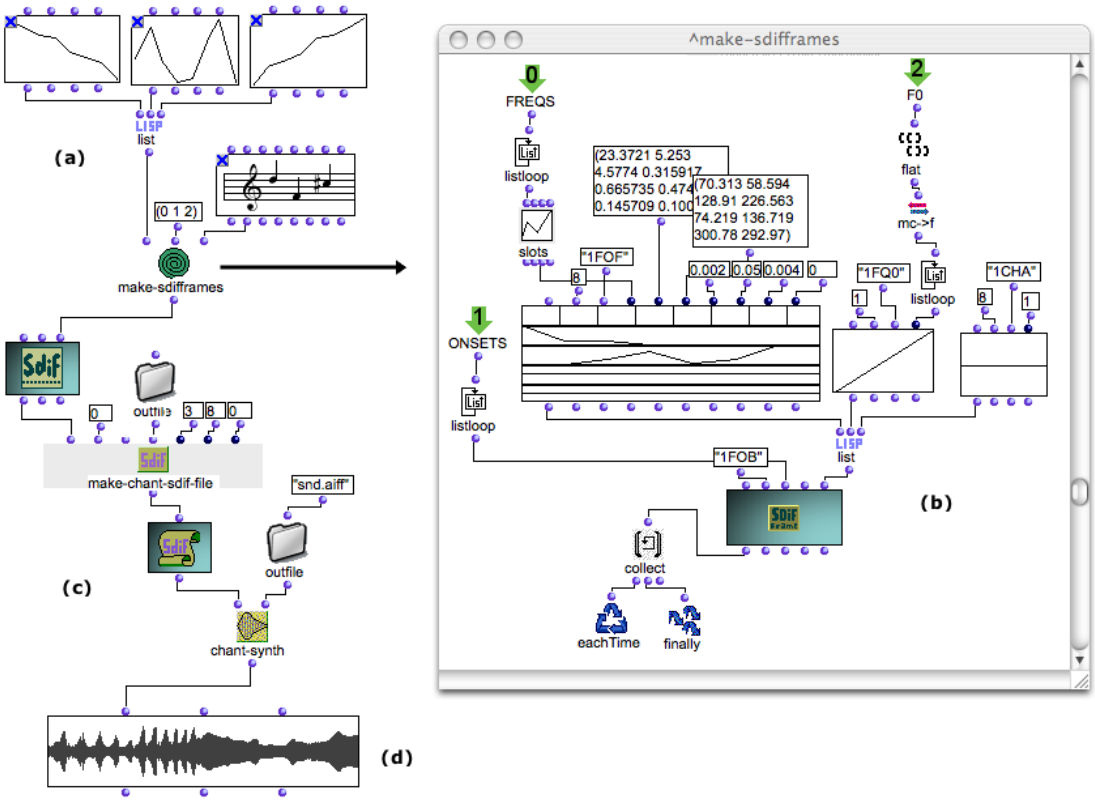


FIGURE 7.16: Un processus de synthèse par FOF dans OPENMUSIC. Des structures de données symboliques (a) sont converties en paramètres de synthèse sous la forme de structures SDIF (b) puis transférées au programme de synthèse (c) pour calculer un son (d).

Complémentairement à la démarche consistant à convertir systématiquement les données de bas niveau en structures symboliques, simples et structurées, les outils présentés dans ce chapitre permettent la programmation visuelle et le calcul symbolique sur ces mêmes données, constituant une modalité de navigation supplémentaire entre le domaine symbolique et celui du signal sonore.

Chapitre 8

Analyse du signal sonore

Nous avons évoqué dans le chapitre 3 le problème de l'information "perdue" par la suppression de l'intermédiaire humain et mécanique (interprète/instrument) dans la chaîne de production sonore, qui est un aspect important du problème général du contrôle de la synthèse sonore. Pour obtenir des sons satisfaisants, il faut en effet fournir aux dispositifs de synthèse des données suffisamment riches et structurées ; or ces données, généralement proches du niveau du signal (ou de celui que nous avons qualifié de "subsymbolique") et représentant souvent des fonctions du temps complexes et précises, peuvent difficilement être appréhendées entièrement dans une pensée musicale.

Une méthode répandue pour l'obtention et la génération de valeurs complexes pour les paramètres de synthèse sonore consiste à les extraire de signaux naturels. Ces signaux assurent en effet la richesse et la diversité du matériau, qui pourra être manipulé et transformé avant d'être redirigé vers la synthèse sonore. Cette insertion des données issues de sons ou d'analyses sonores dans les processus compositionnels fera l'objet des outils proposés dans ce chapitre.

8.1 Le son comme source de matériau musical

[Boesch, 1990] distingue deux approches dans les systèmes informatiques de composition (et par extension de synthèse sonore) : une première qui s'appuierait sur des algorithmes (dans l'optique des systèmes de composition automatique, puis de CAO), et une deuxième sur des données. Cette deuxième approche consiste à tirer parti de matériau (musical) existant (de sons, par exemple) pour le travailler et produire des formes dérivées de ce matériau. L'auteur se référait alors principalement à l'approche de la musique concrète ; c'est cependant une idée analogue qui dirige de nombreux processus utilisés dans la musique électroacoustique contemporaine.

Aux premières heures de l'électroacoustique, la musique concrète s'est en effet intéressée à la manipulation de sons naturels pris dans leur totalité (ce qui était l'unique possibilité avec les moyens de l'époque) pour construire des sons nourris par la richesse et la

complexité naturelles des enregistrements. Les méthodes de synthèse modernes ont par la suite permis une plus grande ductilité du matériau, mais il est alors devenu plus difficile de créer des structures sonores complexes *in abstracto*, sans matériau initial.

Dans ce contexte, la structure interne de sons naturels peut donc encore s'avérer utile, et les différentes techniques de traitement et d'analyse des signaux sont capables de fournir des quantités d'information et de connaissances sur les sons qu'il est possible d'introduire dans des processus de synthèse [Risset et Mathews, 1969].

Nous avons vu dans le chapitre 1 que les techniques de synthèse "spectrales" étaient fréquemment liées plus ou moins directement à des méthodes d'analyse du signal. Outre l'intérêt scientifique de la démarche d'analyse/resynthèse pour l'identification et la compréhension du rôle des différents paramètres, ce type de procédure est peut donc également être suivie par les musiciens pour pallier la complexité du phénomène sonore et des problèmes qui en retournent dans la création de sons de synthèse. Les systèmes comme le vocoder de phase, l'analyse additive ou l'analyse source/filtre, permettent d'extraire des données, de les traiter numériquement, de créer éventuellement des hybridations avec des données extérieures, et de resynthétiser enfin des sons nouveaux à partir de ces données [Risset, 1991] [Arfib et Kronland-Martinet, 1993].

Cette méthodologie d'analyse/transformation/synthèse s'est donc largement développée dans les pratiques compositionnelles, l'analyse fournissant des données sonores naturelles (réservoir de formes temporelles, d'organisations spectrales, ou autres) assurant, même à l'issue d'éventuels traitements musicaux, la richesse et la cohérence des sons synthétisés [Risset, 1993] [Stroppa *et al.*, 2002].

Dans les systèmes temps réel, la captation sonore ou gestuelle permet également une intégration directe de données naturelles dans les processus de synthèse. L'approche de la CAO, cependant, aura tendance à privilégier une vision plus globale sur le matériau et considérera ces données comme des structures statiques utilisées comme matériau initial dans les processus de composition.

Enfin, notons que sans même parler de synthèse, l'analyse sonore peut aussi simplement être utilisée comme réservoir de formes, hauteurs, structures temporelles, réinsérées dans une composition instrumentale : c'est l'une des idées principales du courant de musique spectrale, illustrée notamment par Gérard Grisey, Tristan Murail, et bien d'autres.

8.2 L'abstraction du matériau sonore

La question des représentations symboliques dans la composition a été évoquée dans le chapitre 5. Certaines formes d'analyse du signal, comme l'analyse additive, permettent de se rapprocher directement de ce type de représentation, par analogie aux structures musicales "traditionnelles" (temps/fréquence/amplitude), et constituent ainsi des représentations intuitives tout en gardant un pied dans la description réelle du son. De manière plus générale, on pourra cependant considérer aussi que toute autre procédure d'extraction de

données à partir d'un son, suivant une démarche plus ou moins scientifique ou musicale, peut constituer une étape d'abstraction qui permettra une manipulation musicale de ce son, et donc une considération symbolique sur celui-ci. Aux différentes techniques d'analyse et de représentation des signaux existantes correspondent donc autant de points de vue depuis lesquels les sons peuvent être considérés symboliquement dans la composition. Nous retrouvons là le côté subjectif et personnel du caractère symbolique que nous avions évoqué plus tôt.

> "[...] avant de devenir des symboles, les régularités extraites [des analyses du signal] sont des candidats potentiels à être des symboles, car cette réduction et cette manipulabilité n'engendrent pas nécessairement une fonction symbolique. Parmi ces formes, les régularités, il y a celles qui ont une résonance avec notre cognitif et celles qui n'en ont pas du tout." C. Cadoz.

L'"abstraction compositionnelle" opérée sur les données de description sonore se manifeste spécialement dans les utilisations qui en seront faites : une certaine description pourra être utilisée, selon les cas, suivant une correspondance directe entre l'origine et la destination des données (par exemple une analyse de suivi de fondamentale utilisée pour générer des hauteurs), ou suivant une démarche dans laquelle elle devient une forme abstraite, utilisée dans un autre contexte (par exemple, utiliser la forme de l'analyse de fondamentale pour créer une enveloppe d'amplitude). Cette dualité, schématisée sur la figure 8.1, n'implique aucunement une incompatibilité des deux approches.

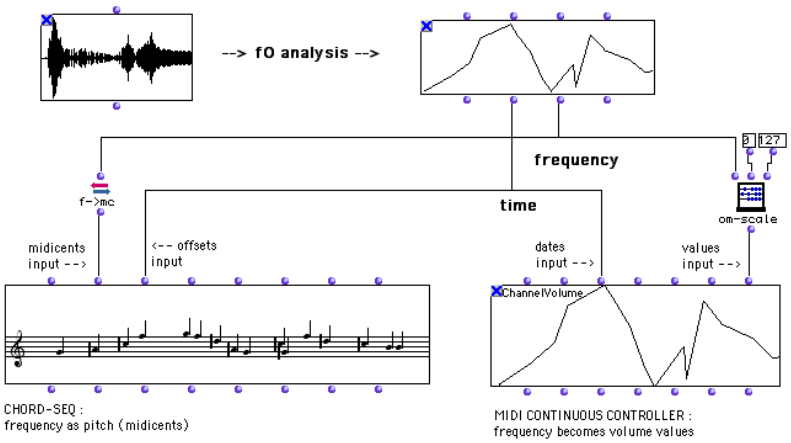


FIGURE 8.1: Abstraction des données d'analyse pour l'utilisation de la description sonore dans différents contextes.

C'est notamment ce caractère abstrait et générique donné par la composition à ce type de données qui nous a incité à utiliser le format SDIF, afin de reproduire ce caractère dans les outils de programmation (voir chapitre 7). L'importation d'un fichier SDIF contenant des données d'analyse permet en effet une utilisation libre de ces données dans les processus compositionnels.

[Haddad, 2006] illustre clairement cette démarche. Pour sa pièce *no one to speak their names (now that they are gone)*, l'auteur utilise des données d'analyse de résonance obtenues par le programme ResAn. Ces mêmes données sont utilisées dans la pièce avec différents objectifs : pour le paramétrage d'un processus de synthèse, et pour la génération de structures rythmiques. Le second cas met particulièrement en avant cette idée selon laquelle les données issues de l'analyse représentent une forme abstraite applicable à différents niveaux de la composition ; les formes temporelles à grande échelle étant issues de données d'analyses sonores "atemporelles". La valeur subjectivement accordée à ces données est exploitée dans une démarche considérant leur forme "en soi", indépendamment de leur fonction initiale. La figure 8.2 est tirée d'un patch créé pour la composition de cette pièce. Un fichier SDIF est utilisé comme intermédiaire entre le programme d'analyse et le processus de composition.

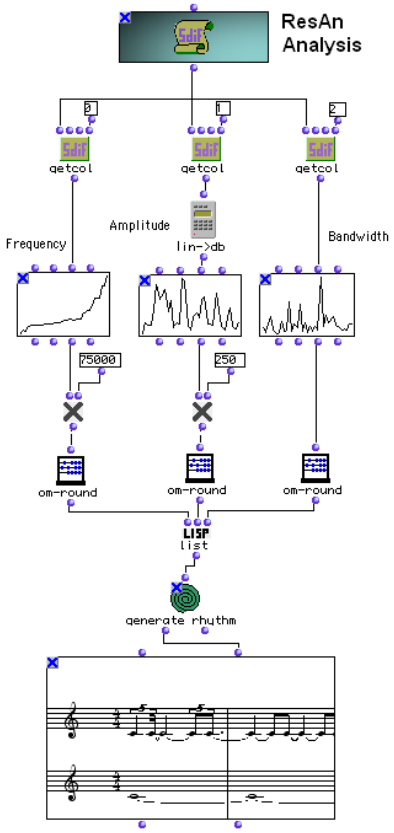


Figure 8.2: Utilisation de données d'analyse de résonances pour la création de rythmes dans *no one to speak their names (now that they are gone)* de Karim Haddad.

Le sens musical des données dans ce type de démarche, ou au contraire leur simple utilisation comme prétexte, comme source de matériau sur lequel viendra se greffer *a posteriori* une démarche musicale, est encore une fois une question esthétique relevant de la décision du compositeur, et sur laquelle il n'est fait aucune présupposition au niveau des outils de composition tels que nous les envisageons.

8.3 Outils pour l'analyse dans OpenMusic

Complémentairement aux outils de synthèse et de traitement du son présentés dans le chapitre 6, OMSounds contient un certain nombre d'outils et fonctionnalités pour l'analyse et l'interprétation des données d'analyse sonore.

Les analyses sont réalisées par des programmes externes de traitement du signal que nous avons déjà rencontrés dans le chapitre 6. Il s'agit des programmes SuperVP et pm2, utilisés notamment pour l'analyse et la synthèse dans AudioSculpt (voir chapitre 2), et pour lesquels des fonctions d'interface avec le langage de programmation visuel ont été réalisées.

Les données issues des analyses sont généralement transférées par l'intermédiaire de fichiers SDIF, et des fonctions génériques ou spécialisées en permettent l'extraction et l'utilisation ultérieure dans les processus musicaux développés dans OpenMusic.

8.3.1 Les données de la forme d'onde numérique

Les échantillons isolés d'une forme d'onde numérique peuvent être extraits et exploités dans un processus de composition. Considérer les valeurs de ces échantillons comme données musicales, ou matériau compositionnel, est en effet une première étape d'abstraction du matériau sonore. La figure 8.3 montre une fonction permettant d'extraire des échantillons d'un fichier audio, avec un échantillonnage variable.

Ce procédé a été également utilisé par Karim Haddad pour sa pièce *Adagio for String Quartet*, dans laquelle il utilise une forme d'onde sous-échantillonnée de la sorte pour créer, encore une fois, des polyphonies de durées [Haddad, 2006].

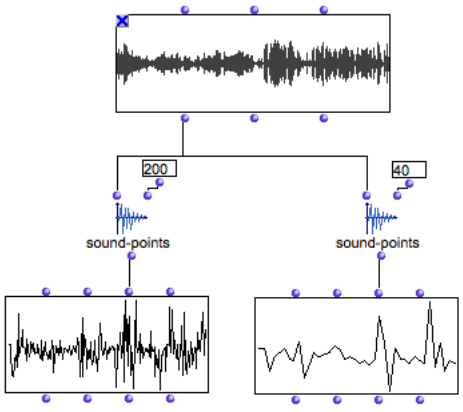


Figure 8.3: Extraction d'échantillons d'un fichier audio selon un taux d'échantillonnage déterminé.

Les outils d'analyse vont cependant nous permettre d'obtenir des représentations plus complètes et significatives du son.

8.3.2 Analyse spectrale

La plupart des algorithmes de traitement dans SuperVP sont basés sur l'analyse de Fourier à court terme, produisant une représentation temps/fréquence du son (voir section 1.2.2). Le résultat de celle-ci est écrit dans un fichier au format SDIF. Partant d'un fichier audio (`sound`), la fonction `FFT` permet d'appeler SuperVP depuis OpenMusic et d'instancier un objet `SDIFFile` à partir du fichier SDIF créé (figure 8.4).

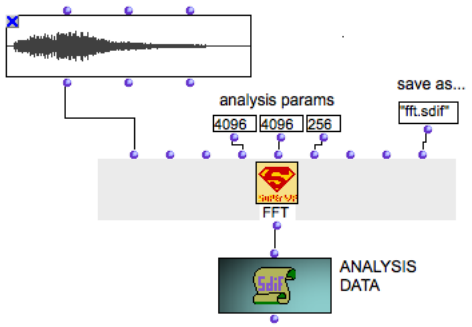


Figure 8.4: Analyse FFT.

Les principaux paramètres de la boîte `FFT` sont la taille de la FFT, la taille, la forme, et le pas d'*overlapping* de la fenêtre d'analyse, et le format des données en sortie. Des valeurs par défaut pour ces paramètres permettent d'obtenir un premier résultat rapidement. Ils peuvent éventuellement être ajustés ensuite, en fonction du son initial et des caractéristiques souhaitées de l'analyse.[1]

Les fonctions SDIF permettent ensuite d'extraire des données localisées dans l'analyse et de les exploiter dans la suite d'un processus. La figure 8.5 montre des exemples d'utilisation de la fonction générique `getsdifdata` pour l'extraction de données spectrales d'un fichier SDIF obtenu par la fonction `FFT`.

8.3.3 Fréquence fondamentale

L'estimation de fréquence fondamentale (ou F0) [Doval et Rodet, 1991] est une analyse réalisable dans SuperVP ou dans pm2, fréquemment utilisée aussi bien dans les systèmes de traitement du signal que dans les processus de composition : sa nature (évolution d'une fréquence dans le temps) permet des interprétations musicales directes. La fonction `f0-estimate` (figure 8.6) permet de réaliser cette analyse et de récupérer encore une fois le résultat sous forme d'un fichier SDIF.

Des paramètres supplémentaires apparaissent avec cette analyse, comme des bornes en fréquence pour l'analyse, un seuil de détection, un ordre de lissage (*smoothing*), etc.

[1] Voir par exemple les remarques de la section 1.2.2 concernant le compromis entre les précisions en temps et en fréquence.

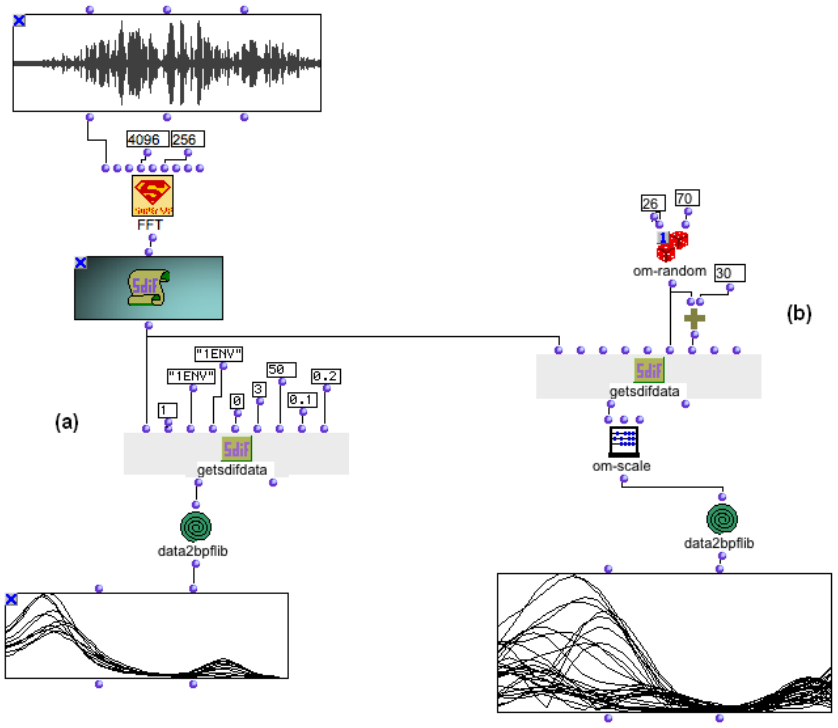


FIGURE 8.5: Extraction de données numériques, (a) localisées ou (b) tirées au hasard, dans un fichier SDIF provenant d'une analyse. La fonction `getsdifdata` permet de spécifier un type de données (ici, frames et matrices de type `"1ENV"`, première colonne) et de sélectionner un sous-ensemble localisé dans le temps et dans les éléments des matrices. Ces derniers paramètres sont soumis à un processus aléatoire dans le cas (b).

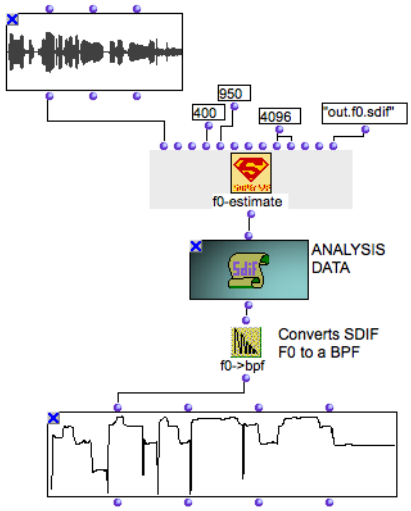


FIGURE 8.6: Estimation de fréquence fondamentale.

La fonction `f0->bpf` permet de convertir directement ce résultat d'analyse en un objet de type `bpf`, visualisable, éditable et utilisable dans OPENMUSIC.

8.3.4 Segmentation

Avec la détection des transitoires d'attaques, SuperVP nous permet d'obtenir un découpage temporel automatique des sons [Röbel, 2003]. Le résultat de cette analyse, réalisée par l'intermédiaire de la boîte `transient-detection`, est encore un fichier SDIF dont nous pouvons lire les informations temporelles par la fonction `get-mrk-onsets` (figure 8.7). Le résultat de celle-ci est une liste de valeurs qui peut par exemple être utilisée pour spécifier les *markers* d'un objet `sound`.

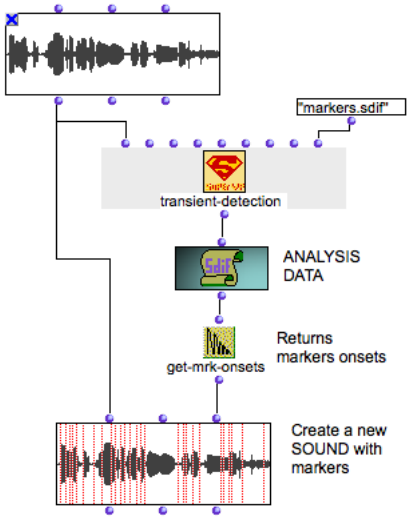


Figure 8.7: Détection de transitoires. Les données extraites du son sont réinjectées en tant que *markers* sur le même fichier audio.

8.3.5 Analyse additive

Avec Pm2, nous allons pouvoir réaliser des analyses additives des fichiers audio, c'est-à-dire en extraire les partiels les plus importants (on parle de suivi de partiels, ou *partial tracking* – voir chapitre 1, section 1.2.2) selon diverses méthodes.

La méthode la plus générale (dite "inharmonique") extrait des partiels indépendants, caractérisés par un numéro d'identification, une date de départ, et des évolutions de fréquence, d'amplitude, et éventuellement de phase.

Avec l'analyse "harmonique", en revanche, il faut fournir une information préalable sur la fréquence fondamentale du signal analysé. Les partiels sont alors considérés comme étant les multiples de cette fréquence fondamentale.

La figure 8.8 illustre la différence entre ces deux types d'analyses, réalisées dans AudioSculpt.

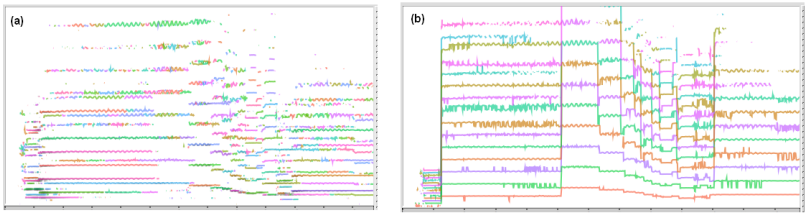


FIGURE 8.8: Représentation des résultats d'analyse additive d'un son dans AUDIOSCULPT (*partial tracking*) (a) inharmonique et (b) harmonique.

La fonction `as->om` écrite par Peter Hanappe [Hanappe et Assayag, 1998] permet de convertir des données issues de suivis de partiels en séquences musicales (`chord-seq`[2]) dans OPENMUSIC. En adaptant cette fonction au format SDIF, dans lequel ces données sont écrites aujourd'hui, nous avons pu ainsi connecter les sons à une interprétation symbolique à l'intérieur d'un même patch. C'est ce qui est illustré sur la figure 8.9.[3]

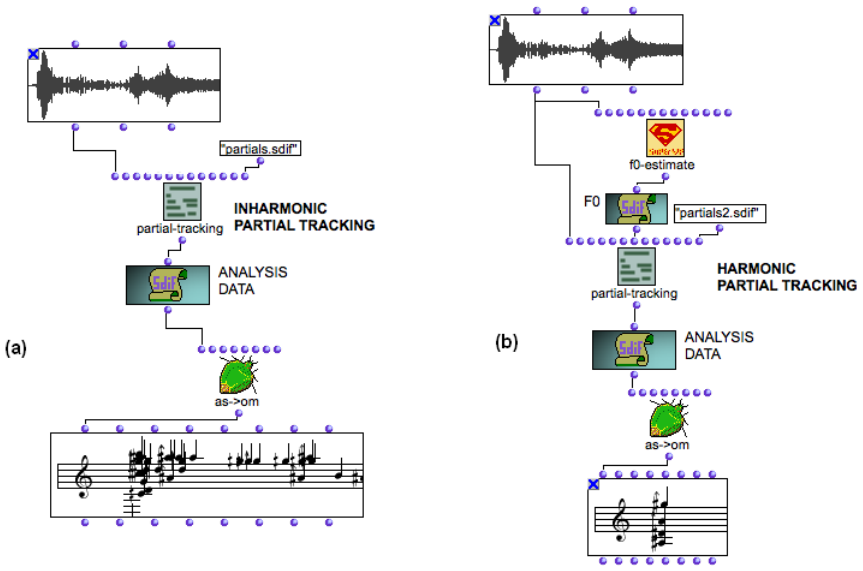


FIGURE 8.9: Analyses additives (a) inharmonique et (b) harmonique avec la fonction `partial-tracking` et conversion en données musicales symboliques avec la fonction `as->om`.

Cette figure met en évidence une différence entre les deux analyses : avec l'analyse harmonique (b), on n'obtient avec `as->om` qu'un accord : l'information sur l'évolution des partiels (visible sur la figure 8.8) est perdue dans la conversion en notation symbolique.

---

[2]Le `chord-seq` représente une séquence d'accords ou de notes dont les attributs temporels (*onset*, durée) sont exprimés en temps proportionnel (i.e. en millisecondes).

[3]On remarquera également sur cette figure que la fréquence fondamentale utilisée pour le paramétrage de l'analyse harmonique provient d'un appel préalable à la fonction `f0-estimate` présentée précédemment.

L'analyse *chord sequence* est une autre méthode d'analyse additive, basée cette fois sur une information préalable sur la segmentation temporelle du fichier audio. Un ensemble de *markers* temporels doivent donc être fournis avant de réaliser l'analyse, qui effectuera une réduction de données sur les intervalles correspondants. La figure 8.10 montre le résultat de cette analyse dans AudioSculpt. L'analyse *chord sequence* constitue donc une représentation plus compacte, basée sur un apport supplémentaire d'information initiale. Elle a par ailleurs une correspondance directe (sans perte, cette fois) avec des structures musicales comme le `chord-seq` dans OpenMusic.

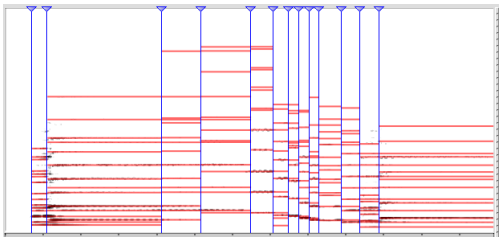


Figure 8.10: Analyse *chord sequence* dans AudioSculpt.

Depuis OpenMusic, on appellera donc la fonction `chord-sequence-analysis` en fournissant au programme une segmentation temporelle a priori (figure 8.11 - a). Si l'objet `sound` analysé contient lui-même des *markers*, ceux-ci seront utilisés à cet effet (figure 8.11 - b). Ici encore, une analyse *transient-detection* peut être utilisée en amont pour déterminer ces *markers*. La fonction `as->om` permet de convertir le résultat d'analyse en un `chord-seq`.

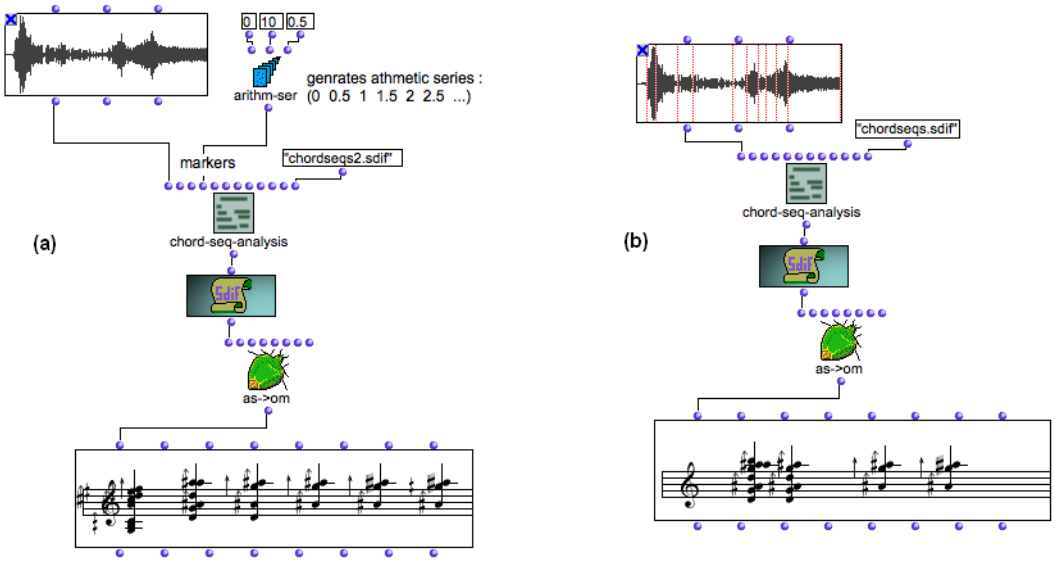


Figure 8.11: Analyse *chord sequence* et conversion en données musicales symboliques. La segmentation de l'analyse est spécifiée (a) par un processus décrit dans le patch et (b) par les *markers* dans le son analysé.

8.4 Utilisation de l'analyse dans les processus de synthèse

Les fonctions d'analyse présentées dans la section précédente pourront être associées aux outils de programmation de l'environnement et intervenir à différents niveaux dans les programmes et traitements compositionnels.

L'utilisation des boucles d'itérations (*loops*) permettra notamment d'effectuer des traitements d'analyse à la chaîne dans OpenMusic, pour analyser des banques de sons entières, ou encore créer des banques de données d'analyses à partir de sons (en variant les paramètres d'analyse) dans un même processus.

Elles pourront également, comme le montrent les quelques exemples qui suivent, être utilisées en complémentarité avec les outils de traitement et de synthèse sonore présentés dans le chapitre 6.

Les exemples des figures 8.12 et 8.13 reprennent respectivement ceux des figures 6.11 et 6.19 en mettant en relation dans un même programme les outils d'analyse et de traitement audio. Une segmentation est réalisée grâce à la détection de transitoire avant d'appliquer les algorithmes de montage ou de traitement par *time-stretching* sur les sons.

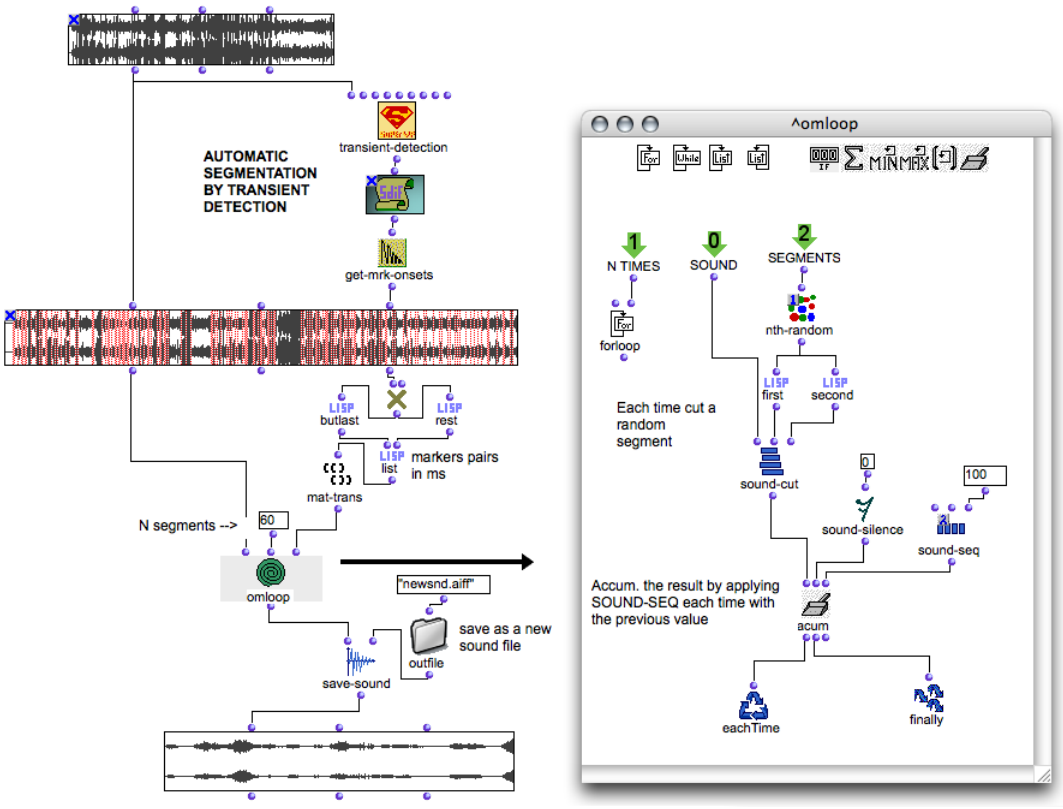


FIGURE 8.12: Manipulation algorithmique d'un signal audio basée sur des données d'analyse. Le patch présenté dans la figure 6.11 effectue un découpage et remontage aléatoire d'un fichier audio. Ici, la segmentation de ce fichier est issue d'une analyse de détection de transitoires réalisée dans le même patch.

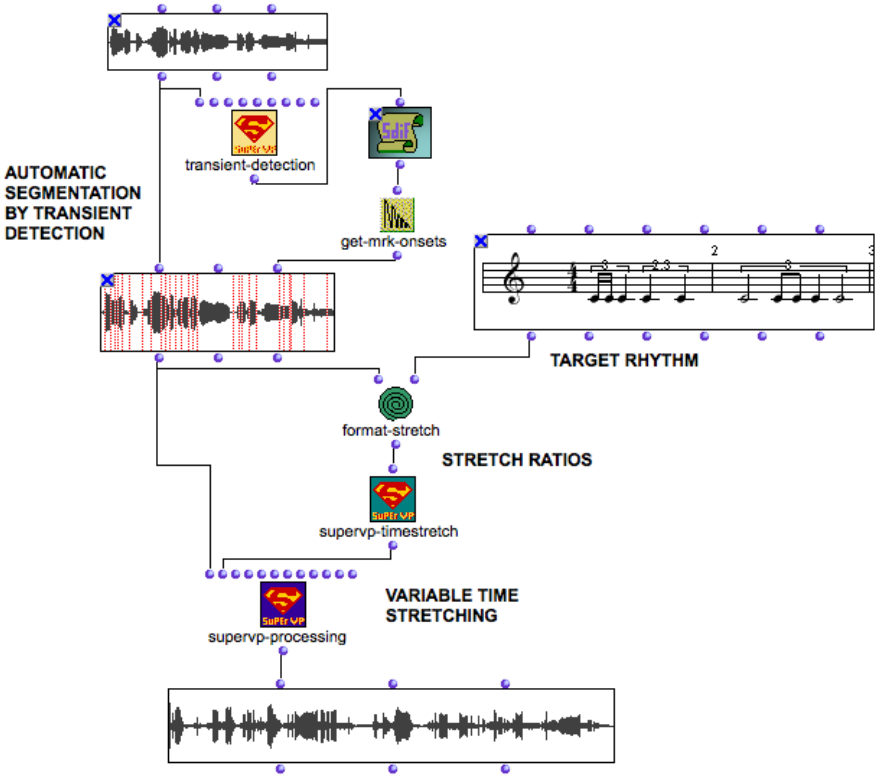


FIGURE 8.13: Traitement d'un signal audio (*time stretching*) à partir de données musicales (rythmiques) symboliques. La segmentation du son est calculée par une analyse de détection de transitoires.

Sur la figure 8.14, la fonction de transposition de la librairie OM-SuperVP transpose un son selon un paramétrage provenant du suivi de fréquence fondamentale de ce même son.

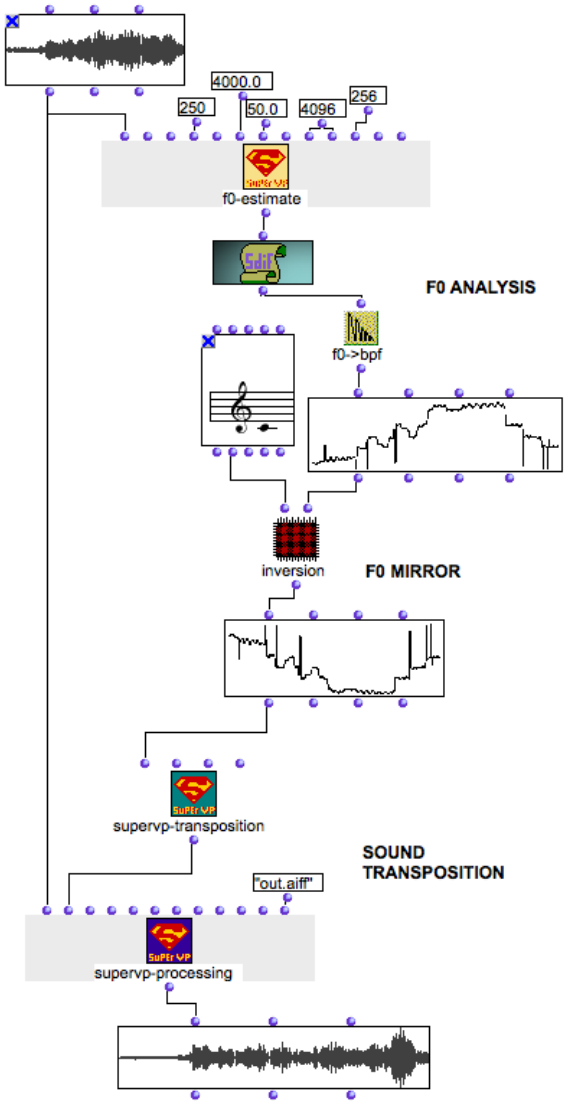


Figure 8.14: Transposition d'un signal audio à partir d'une transformation sur des données d'analyse. Les données de paramétrage de la transposition sont calculées par une opération de symétrie entre les données d'analyse initiales et une hauteur donnée. La transposition produit ainsi un son à fréquence fondamentale constante (la valeur de l'axe de réflexion), tout en maintenant les autres caractéristiques (enveloppe spectrale, transitoires, bruit, etc.) du son initial.

La figure 8.15 reprend l'exemple de la figure 7.16 du chapitre précédent, en assurant cette fois le paramétrage de la synthèse par FOF (plus précisément, du paramètre "fréquence fondamentale") à l'aide d'une analyse de fondamentale d'un signal audio. Une fonction d'interpolation entre deux BPF est utilisée pour générer les valeurs successives des fréquences des FOF.[4] Ces processus conjugués permettent ainsi un paramétrage beaucoup plus complexe que celui effectué dans l'exemple original.

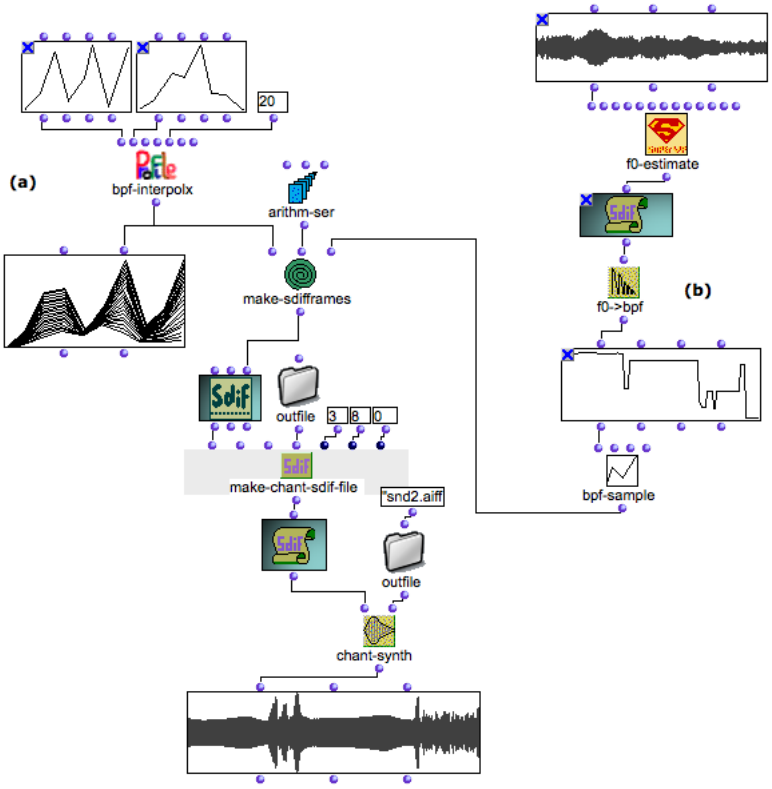


FIGURE 8.15: Synthèse par FOF avec CHANT. Les paramètres principaux de la synthèse sont obtenus par un processus d'interpolation (a) et par l'analyse F0 d'un son initial (b).

Enfin, les différentes analyses additives, converties en données symboliques, pourront être traitées et redirigées vers divers processus de synthèse. Le patch de la figure 8.16 utilise par exemple la fonction de synthèse définie en conclusion du chapitre 6 (figure 6.21), et utilise les données issues de l'analyse additive d'un son et converties en notation musicales pour paramétrer cette fonction.

[4]Le chapitre suivant traitera de cet autre type de procédé (algorithmique) utilisé pour la génération des données.

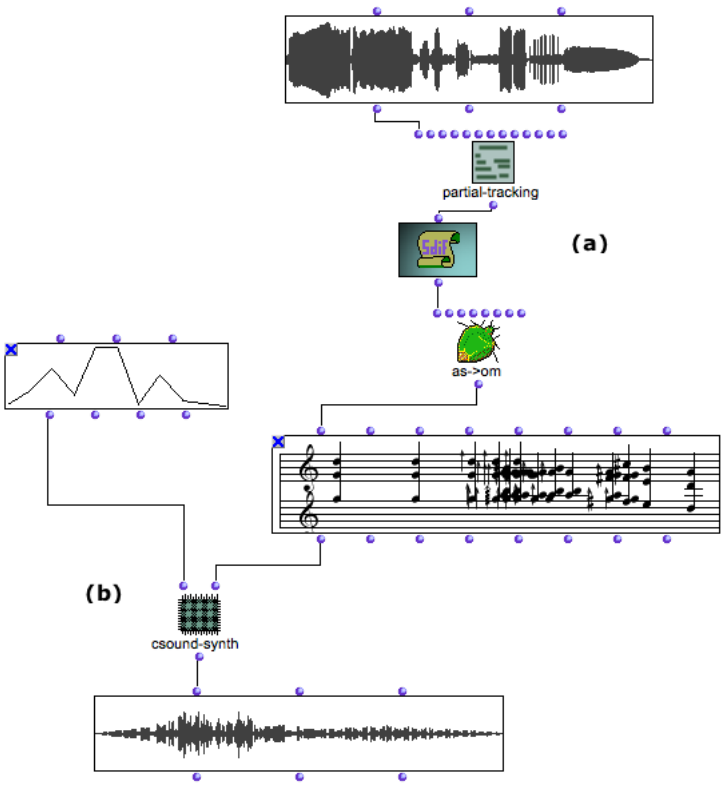


FIGURE 8.16: Processus (a) d'analyse et (b) de resynthèse sonore dans OPENMUSIC.

L'intérêt de passer ainsi par des données musicales dans une telle démarche consiste alors principalement à tirer parti du niveau symbolique atteint par ces données pour les soumettre à des procédés et traitements compositionnels avant de revenir vers la synthèse.

8.5 Conclusion

Les outils présentés dans ce chapitre permettent l'obtention de matériau musical issu de sons préenregistrés, destiné à être manipulés dans les processus musicaux développés dans OPENMUSIC. Comme nous l'avons vu avec les outils de synthèse (chapitre 6), ceux-ci s'insèrent au sein même et dans la même logique de calcul et d'exécution que ces processus musicaux. Ils permettent ainsi au compositeur d'intégrer le domaine du signal à celui de la CAO dans un projet compositionnel à l'intérieur duquel différents cycles du matériau musical, du son vers les données symboliques et des données symboliques vers le son, peuvent être développés.

Le chapitre suivant traitera plus particulièrement des structures de données utilisées pour la synthèse, qui pourront être mises en relation avec ces mêmes outils d'analyse pour la création de processus à grande échelle intégrant analyses et traitement symboliques de ces données.

Chapitre 9

Structures de données pour la synthèse sonore

Dans ce chapitre nous nous intéressons aux structures de données mises en jeu dans les modèles compositionnels tournés vers le son et la synthèse sonore. Celles-ci pourront inclure ce que nous avons appelé les descriptions sonores dans le chapitre 7, c'est-à-dire des données de bas niveau issues de l'analyse des sons et/ou destinées au paramétrage de synthétiseurs, mais également s'étendre à des représentations d'ordre plus général et de plus haut niveau pouvant intervenir à un moment donné dans un processus compositionnel lié, dans un sens ou dans l'autre (celui de l'analyse ou celui de la synthèse), au signal sonore.

Nous avons noté précédemment que ces données constituaient des abstractions (partielles) du son, dont l'affectation à des valeurs précises permettait de définir des sons concrets, suivant un point de vue donné dans l'espace défini par les processus de synthèse sonore et les modèles compositionnels. En ce sens, elles sont la matérialisation de l'abstraction du signal sonore opérée par le processus de modélisation, et reflètent une manière de penser et concevoir les sons à travers les modèles.

Particulièrement, nous nous concentrerons sur les moyens de générer ces données dans le cadre de processus compositionnels. Après l'utilisation de matériau provenant de sons préexistants, une nouvelle stratégie est envisagée à travers les outils de calcul et de représentation utilisés pour la génération et le traitement des données.

9.1 Structures de données élémentaires

Deux principaux objets, que nous avons déjà rencontrés dans les chapitres précédents, permettent de manipuler les descriptions sonores dans OpenMusic : les fonctions (ou enveloppes) et les matrices.

Nous essaierons de montrer ici les différentes possibilités permettant de spécifier ces données à partir des moyens de programmation et des structures de données disponibles dans cet environnement, l'idée étant d'intégrer une composante symbolique, permettant leur insertion dans un contexte musical, et une composante "subsymbolique" complémentaire par laquelle elles pourront être utilisées pour synthétiser des signaux avec finesse et précision.

9.1.1 Fonctions et enveloppes

Une fonction représente l'évolution de la valeur d'un paramètre par rapport à un autre. Celle-ci peut être exprimée mathématiquement ou par des données discrètes.

Nous appelons *enveloppe* un contour que l'on applique comme valeur ou comme pondérateur des valeurs d'un paramètre. Ce type de structure est fréquemment utilisé pour décrire les variations de paramètres de contrôle des synthétiseurs, dans le temps (le plus souvent) ou dans d'autres dimensions (nous avons parlé en particulier d'enveloppes spectrales dans le chapitre 1). Il s'agit donc d'une fonction particulière, relativement simple mais fondamentale dans la manipulation des sons.

Si leur précision est généralement déterminante, une information réduite peut cependant suffire à la description de ces objets, leur profil pouvant parfois être défini par certains points particuliers. Les enveloppes dynamiques sont par exemple souvent décrites par les couples de points (temps, amplitude) définis en quelques instants significatifs des sons. Une fonction ou une enveloppe définies par les valeurs de leurs points d'inflexion correspondent alors à ce que l'on appelle une *breakpoints function* (BPF, ou "fonction par segments"). Nous avons vu à plusieurs reprises dans les chapitres précédents ce type d'objet utilisé dans OpenMusic pour paramétrer des processus de synthèse (par exemple dans les exemples donnés figures 6.21 et 7.16).

Dans certains cas cependant, une description plus précise doit être spécifiée, suivant un taux d'échantillonnage plus fin (celui-ci restant cependant, dans la plupart des cas, largement inférieur au taux d'échantillonnage audio). On parle parfois de fonctions "continues".[1] Les valeurs peuvent alors être données "par extension", ce qui sera souvent compliqué et fastidieux, ou bien par des moyens algorithmiques, ou mathématiques, c'est à dire "par intension". Le calcul et la programmation permettront alors de générer des courbes complexes, ou de transformer les valeurs de courbes simples données initialement. Une fonction définie avec un certain taux d'échantillonnage peut en effet être obtenue à partir

[1] Nous reviendrons sur cette notion dans le chapitre 10.

d'une BPF simple sur laquelle une fonction d'interpolation est appliquée pour déterminer les différentes valeurs entre les points d'inflexion donnés. Les algorithmes d'interpolation polynomiale permettent également la génération de profils à partir de "points de contrôles" donnés dans une BPF. Ces différentes méthodes ont été implémentées dans OpenMusic.

Des outils dédiés au traitement des courbes et enveloppes fournissent ainsi des techniques pour la création ou la transformation de ces objets dans les programmes visuels (par exemple avec la bibliothèque Profile [Baboni-Schilingi et Malt, 1995]). Différentes techniques d'interpolation (par exemple linéaire, ou par *B-Spline* [Bartels *et al.*, 1987]) sont également disponibles sous forme de fonctions (voir figure 9.1).[2]

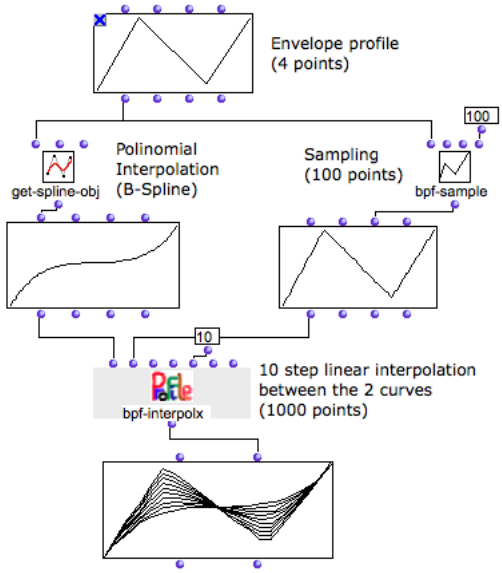


Figure 9.1: Création et transformations de courbes par le calcul dans un patch.

L'éditeur de `BPF` permet également une visualisation préalable de profils lissés, calculés par un algorithme de calcul de *B-Splines* étant donnés une résolution et un degré d'interpolation souhaités (voir figure 9.2).

Les problématiques liées à la génération et la manipulation des fonctions continues sont diverses. [Desain et Honing, 1992] soulèvent en particulier le problème des transformations sur les enveloppes selon les cas d'utilisation : l'enveloppe d'un vibrato par exemple (oscillations de la fréquence), si elle est prolongée dans le temps, ne doit pas être étirée (c'est-à-dire maintenir le même nombre d'oscillations sur un intervalle de temps plus long) mais être prolongée par répétition pour atteindre la durée souhaitée. Au contraire une courbe d'évolution de fréquence qui décrirait une transition continue (par exemple un

[2]Nous avons déjà rencontré dans le chapitre 8 (figure 8.15) un cas d'utilisation d'une fonction d'interpolation pour la génération d'une série de courbes décrivant des états intermédiaires dans une synthèse par FOF.

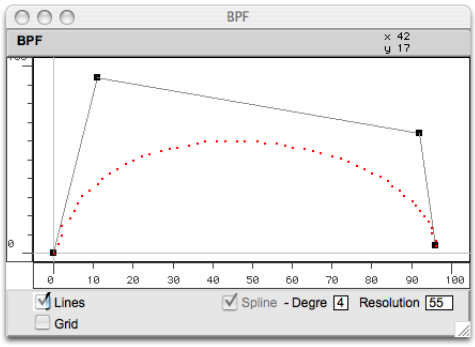


FIGURE 9.2: Prévisualisation des résultats d'algorithmes d'interpolation polynomiale (*B-Splines*) dans l'éditeur de `BPF`.

glissando) doit dans la même situation être étirée littéralement. Dans les deux cas cependant, on a affaire à un même objet (une fonction spécifiant une variation de fréquence), dont le comportement nécessite une spécification adaptée aux différentes situations. Dans OPENMUSIC, le parti pris est donc de déléguer ce type de spécification au compositeur par l'intermédiaire des outils de programmation.

9.1.2 Matrices

Une matrice dans OPENMUSIC est une classe représentant un tableau de deux dimensions ou plus, correspondant en principe à la description conjointe de plusieurs paramètres d'une description sonore. Le regroupement de ces paramètres dans une même structure constitue alors une réalité sonore plus conséquente, et permet surtout d'établir des relations fortes entre ces paramètres (par exemple incluant des relations de causalité entre leurs évolutions).

Les matrices OPENMUSIC sont instanciées avec un nombre fixe d'éléments (colonnes) et un ensemble de champs de description (lignes). Ces paramètres sont eux aussi spécifiés, soit par extension, soit par intension, à l'aide de moyens plus "symboliques". La figure 9.3 illustre différentes possibilités pour l'instanciation des champs de description d'une matrice. Le nombre d'éléments (colonnes) étant fixé (30 dans cet exemple), les différentes lignes sont remplies en fonction des types de paramètres donnés en entrée des *slots* correspondants aux champs de la matrice. La ligne 1 est calculée à partir d'une valeur constante (23), les lignes 2 et 4 à partir de listes de valeurs (spécifiées textuellement ou provenant d'un objet `bpf`) répétées jusqu'à atteindre le bon nombre d'éléments, la ligne 3 à partir d'une enveloppe (BPF) échantillonnée de sorte à recouvrir tous les éléments, et la ligne 5 à partir d'une fonction mathématique, évaluée sur ce nombre d'éléments (ce dernier cas pouvant être généralisé à toute fonction de type $y = f(x)$ définie par l'utilisateur).

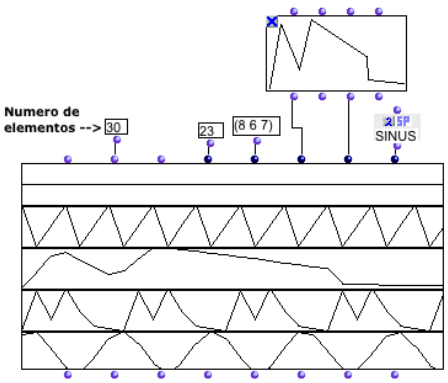


FIGURE 9.3: Instanciation d'une matrice à l'aide de différents types de données.

Une forme de symbolisme est donc maintenue dans le paramétrage de ces matrices, dont les valeurs numériques "exactes" sont calculées en dernière instance, selon différentes stratégies correspondant au type des paramètres. Ce principe est encore renforcé dans le système OMCHROMA, qui en est l'une des applications principales.

9.2 OMChroma : vers une intégration des aspects comportementaux dans les structures de données pour la synthèse

Le système CHROMA a été créé par Marco Stroppa dans le but de disposer d'un niveau de contrôle abstrait et musicalement pertinent pour le paramétrage de processus de synthèse. En séparant ce niveau de celui de la synthèse, le compositeur a souhaité généraliser un environnement de contrôle sous forme d'un langage qui lui permettrait d'exprimer ses propres idées et structures indépendamment des processus et dispositifs de synthèse utilisés. Il met ainsi en avant le concept de "synthétiseur virtuel", programme chargé de la transformation et du formatage des données de contrôle vers différents types de synthèse, et/ou vers différents synthétiseurs [Stroppa, 2000].

Le noyau de calcul CHROMA[3] est à l'origine d'une adaptation dans OPENMUSIC sous le nom de OMCHROMA [Agon *et al.*, 2000]. Le contrôle de la synthèse dans OMCHROMA est essentiellement basé sur des sous-classes des matrices OPENMUSIC présentées précédemment, auxquelles sont affectés des attributs comportementaux particuliers.

[3]Porté en Common LISP par Serge Lemouton.

9.2.1 Intégration données/programmes

Chaque classe dans OMChroma est une matrice de paramètres correspondant à un processus de synthèse implicite, vers lequel seront dirigées les données qu'elle contient lors de l'appel d'une fonction générique de synthèse (`synthesize`). La figure 9.4 montre un exemple de ce procédé. Il s'agit d'une synthèse additive simple avec tout d'abord un seul élément (partiel), puis (en allant vers la droite sur la figure) avec un nombre croissant de partiels à des fréquences, amplitudes, durées, ou attaques différentes. Les *slots* de la matrice correspondant à ces champs de description sont optionnels : il est possible de choisir ceux sur lesquels on souhaite travailler, les autres se voyant affecté des valeurs par défaut.

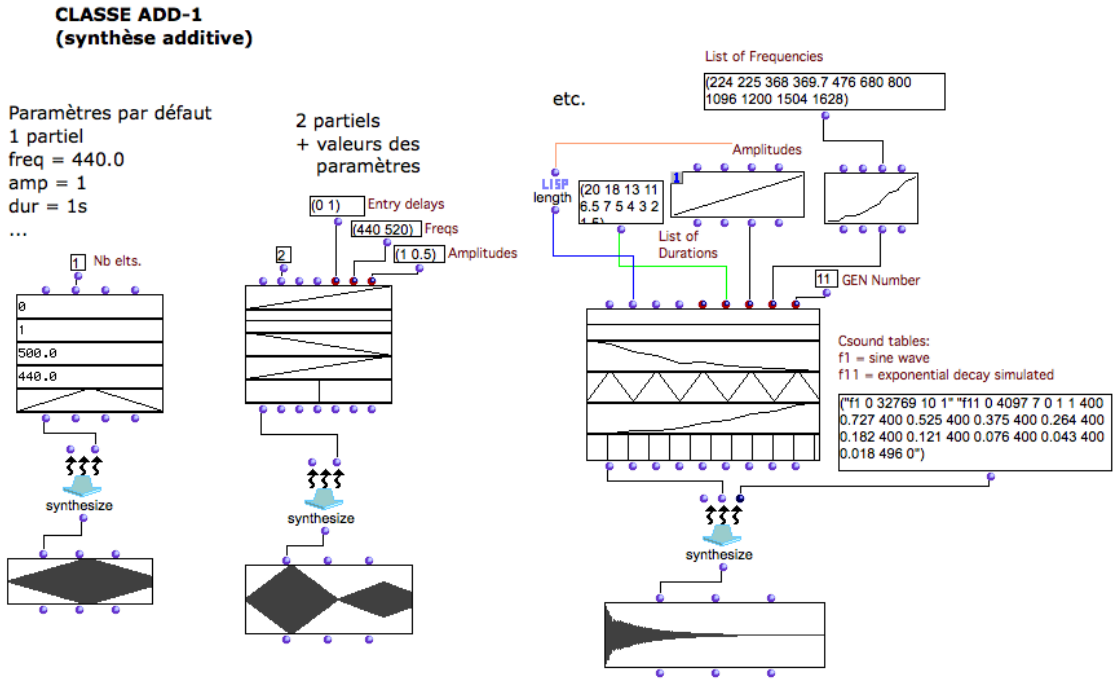


FIGURE 9.4: Synthèse sonore avec OMChroma. Utilisation des valeurs par défaut et complexification des processus d'instanciation de la matrice. La classe `add-1` est interprétée par la fonction `synthesize` sous forme de paramètres pour une synthèse additive réalisée dans CSound.

La plupart des classes existant à l'heure actuelle dans OMChroma correspondent à des processus réalisés dans CSound (la définition textuelle d'un *orchestra* est incluse à l'intérieur de ces classes). Certaines classes visent aussi des processus de synthèse réalisés avec le synthétiseur CHANT. On peut donc envisager d'instancier différentes classes avec les mêmes valeurs, si tant est qu'elles ont des champs de description similaires ou homogènes, et on obtiendra un résultat sonore différent, selon le processus de synthèse visé par celles-ci (voir figure 9.5).

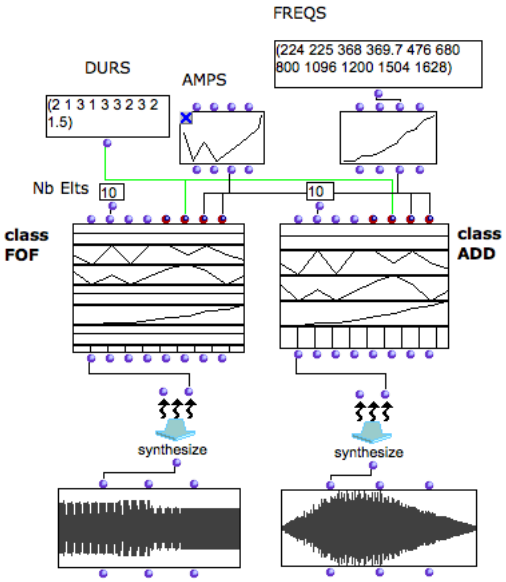


FIGURE 9.5: Synthèse sonore avec OMCHROMA. Deux classes différentes instanciées par les mêmes valeurs produisent deux sons différents, selon les processus de synthèse implicites auxquels elles correspondent respectivement.

OMCHROMA permet ainsi à l'utilisateur de tirer parti des possibilités de programmation par objets de OPENMUSIC pour étendre par héritage les classes existantes en leur attribuant de nouveaux paramètres et comportements.

Les matrices possèdent par ailleurs un *slot* appelé `user-fun`, par lequel une fonction optionnelle peut être spécifiée, qui déterminera la manière dont chaque composant de la matrice sera traité au moment du transfert des données vers le processus de synthèse. Pour chacun de ces composants, cette fonction offre un accès à l'état de la matrice entière, et permet d'effectuer des tests, d'en modifier les éléments, de les éliminer ou d'en créer des nouveaux. Cette fonction permettra ainsi de modifier et éventuellement de compléter ces données dynamiquement, induisant la notion de *règles*. Une règle pourra par exemple statuer que les éléments inférieurs à un certain seuil dans une dimension soient éliminés, que ceux supérieurs à une limite donnée soient ramenés à cette limite, ou que chaque composant soit complété par un certain nombre de *sous-composants* annexes. De par leur généralité, de telles règles, issues ou adaptées de situations musicales particulières, peuvent ensuite être réutilisées et combinées dans d'autres processus de synthèses. Elles peuvent notamment être définies graphiquement sous forme d'un patch (en mode *lambda*, c'est-à-dire interprété en tant que fonction – voir chapitre 4, section 4.3.4).

Un aspect comportemental est donc intégré dans ces structures de données, à travers le processus de synthèse implicitement défini par la classe, le système d'évaluation de champs de description, et la spécification d'un processus de traitement dynamique des données au moment de la synthèse.

9.2.2 Aspects “macro”-compositionnels

L'utilisation des matrices dans OMChroma constitue un positionnement dans une démarche de composition donnée : la définition de classes relèvera plutôt du domaine de la lutherie électronique, puis l'activité compositionnelle à proprement parler s'attachera à paramétrer et contrôler ces instruments pour créer des sons.

Un *slot* général (`action-time`) permet de plus de spécifier une date allouée à une matrice, qui jouera le rôle d'un *onset* dans un processus plus global. Une telle structure de données regroupant un certain nombre de paramètres liés entre eux, et dotée d'une information temporelle se rapproche ainsi de la notion d'*évènement* pour la synthèse. La synthèse d'un ensemble de matrices permettra de mettre plusieurs évènements en relation dans le temps (voir figure 9.6).

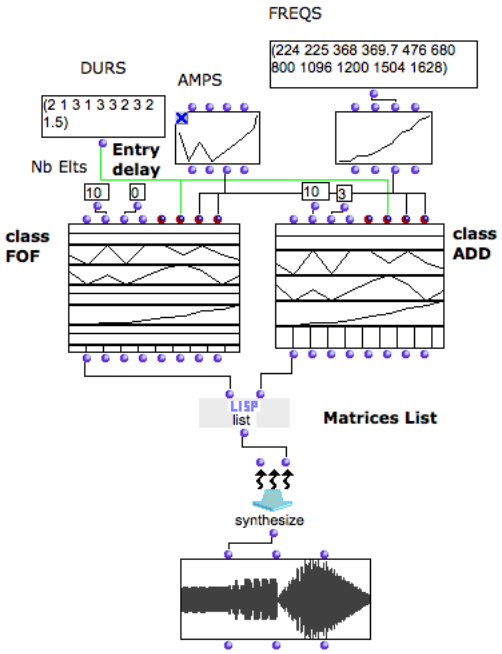


Figure 9.6: Synthèse à partir d'une liste de matrices (celles de la figure 9.5) localisées dans le temps par l'attribut *entry-delay* (à 0 et 3 secondes, respectivement).

Au niveau des matrices, un évènement est donc considéré comme un simple ensemble de données liées entre elles et localisées dans le temps. Un niveau de contrôle plus élevé, se rapprochant de ce que serait l'organisation d'une phrase musicale à partir de ces évènements, sera envisagé dans la prochaine section, puis traité spécifiquement dans la quatrième partie de la thèse.

9.3 Un système pour la génération de formes musicales avec les descriptions sonores

Le système que nous présentons ici est une extension du système OMChroma, également dérivée d'outils et procédures développés par Marco Stroppa dans son activité de composition. Il s'agit avec ce système de générer des matrices telles que celles que nous avons pu voir dans la partie 9.2, selon des formes et des structures à plus grande échelle. Après l'idée d'évènement à laquelle nous avons assimilé les matrices, nous essayons donc de nous approcher de la notion de phrase, ou de forme musicale.

Ce système rassemble d'une part l'idée de l'utilisation de données provenant d'analyses sonores comme matériau musical (développée dans le chapitre 8, section 8.1), et d'autre part celle de structures de données hybrides intégrant les aspects statiques et comportementaux (vue dans la section précédente).

Une description de ce système est également donnée dans [Bresson *et al.*, 2007].

9.3.1 Structure

La structure principale dans ce système est une classe appelée `cr-model`,[4] constituant un nouvel objet musical dans OpenMusic.

Celle-ci constitue une représentation abstraite d'un son selon un modèle spectral (temps/fréquence/amplitudes). Elle est construite à partir d'une structure de hauteurs (*pitch structure*) et d'une structure temporelle (*time structure*) indépendantes. La figure 9.7 montre l'objet `cr-model` instancié dans un patch, ainsi que l'éditeur associé à cet objet.

Cette structure de hauteurs est construite à partir d'une description spectrale, issue d'une analyse sonore ou créée intrinsèquement. Différents types d'analyse sont compatibles, pouvant être réalisées dans OpenMusic (avec les outils présentés dans le chapitre 8) ou dans un système extérieur (par exemple AudioSculpt) et importées sous forme de fichiers SDIF. Quatre types d'analyses sont actuellement supportées : le suivi de partiel harmonique et non harmonique, l'analyse *chord-sequence*, et l'estimation de fréquence fondamentale. Les données de la structure de hauteurs sont spécifiées en connectant un fichier SDIF (boite `SDIFFile`) à l'un des *slots* de la classe `cr-model` et en sélectionnant le type d'analyse visé (à condition que celui-ci figure dans le fichier SDIF en question).

La structure de hauteurs est alors créée sous forme d'un ensemble de sous structures appelées *Vertical Pitch Structures* (VPS). Les VPS sont des structures de données polymorphes unifiant les notions d'accord et de spectre, organisées à l'intérieur d'un ensemble d'outils pour le traitement et la manipulation génériques des données dans la dimension des hauteurs (conversions, filtrages, et autres transformations).

[4] Le terme *model* est ici choisi pour des raisons historiques dues aux antécédents de ce type de structure dans le système Chroma. Il ne doit pas cependant induire un amalgame avec le modèle (compositionnel, en particulier) tel que nous l'avons défini dans les chapitres précédents.

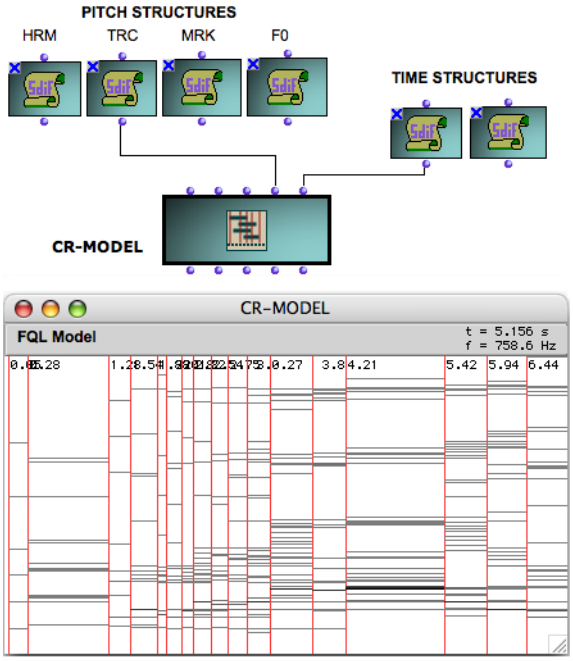


FIGURE 9.7: `cr-model` : une représentation sonore abstraite construite à partir d'une structure de hauteurs et d'une structure temporelle.

Sur la figure 9.7, la structure de hauteurs est construite à partir d'une analyse additive inharmonique (format SDIF "1TRC"). Sur la figure 9.8, le même objet est reconstruit, cette fois avec une structure de hauteurs issue d'une analyse de suivi de fondamentale ("F0").

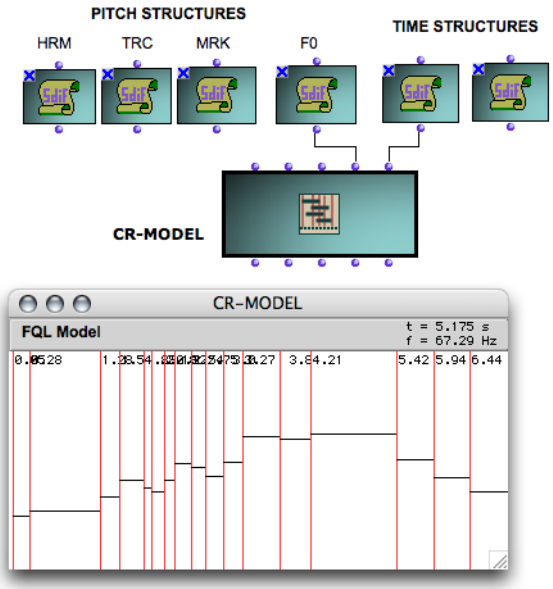


FIGURE 9.8: `cr-model` : structures de hauteurs alternatives (suivi de fondamentale).

La structure temporelle est une simple liste de valeurs spécifiant des *markers* et délimitant des segments. Cette liste peut aussi être déduite d'une analyse "temporelle" comme la détection de transitoires, ou encore la segmentation d'une analyse *chord-sequence*, importée sous forme d'un fichier SDIF également connecté au `cr-model`, tel que le montrent les deux précédentes figures.

Comme le montrent les précédents exemples, les structures de hauteurs et temporelle sont bien indépendantes : on peut sans problème créer une représentation abstraite sous forme de `cr-model`, en utilisant l'analyse spectrale d'un son et la structure temporelle issue d'un autre son, ou d'un processus quelconque. La figure 9.9 illustre ce dernier cas, avec la structure de hauteurs de la figure 9.7 associée à une structure temporelle régulière calculée par une simple série arithmétique (0, 0.5, 1, ..., 5.5, 6).

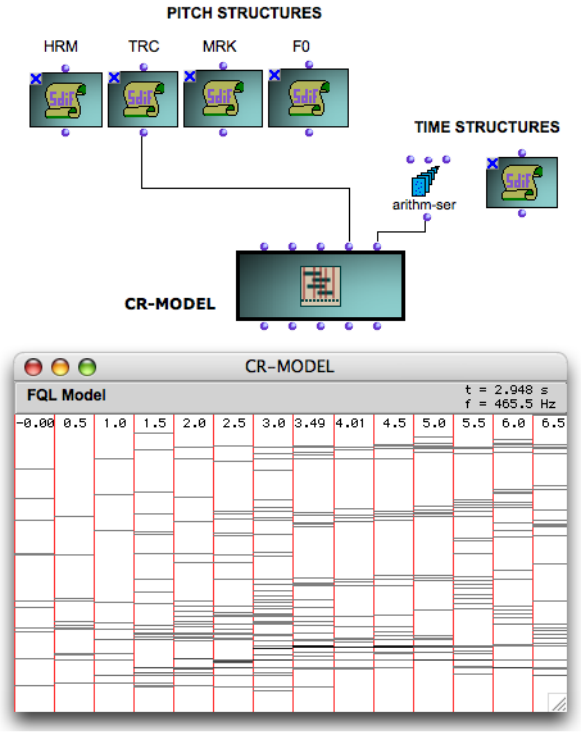


FIGURE 9.9: `cr-model` : structures temporelles alternatives.

La structure temporelle définit ainsi des segments à l'intérieur desquels les données (fréquentielles, principalement) seront considérées comme relativement stables, ou du moins comme relevant d'une même entité. Le sens donné à ces entités est encore une fois laissé à la discrétion du compositeur, et dépendra de ses choix ou stratégies compositionnelles. Si cette segmentation peut ne pas être toujours pertinente (une structure continue ne s'y prêtera pas nécessairement), elle s'avèrera cependant utile lorsqu'il s'agira d'organiser des structures et processus de synthèse à grande échelle. Elle nous rapproche en effet, par le haut cette fois, de la notion d'évènement. (Le lien sera ensuite établi entre ces segments et les matrices–évènements de synthèse telles que nous les avons vues précédemment.)

L'instanciation du `cr-model` découpe donc littéralement les données d'analyse de la structure de hauteurs selon la structure temporelle, forçant parfois cette segmentation en évènements (s'ils n'étaient pas présents dans les données). Dans chaque segment, une structure VPS est créée. Ce système de représentation permet ainsi de stocker et manipuler les différents types de données fréquentielles (analyses spectrales, accords, fréquence fondamentale, etc.) sous un format unique qui facilitera leur traitement ultérieur.

9.3.2 Manipulation des données

Les processus de composition consistent alors en principe à transformer des données structurées à l'intérieur du `cr-model` pour créer de nouvelles structures, plus ou moins proches ou déconnectées des données originales.

Nous avons dit qu'un `cr-model` pouvait être construit intrinsèquement, c'est-à-dire à partir d'un ensemble de VPS précalculées et non nécessairement d'une analyse sonore. Cet ensemble de VPS peut donc être issu d'un autre `cr-model`, puis subir divers traitements, avant d'être associé à une nouvelle structure temporelle arbitraire (ou à la même). La figure 9.10 montre un exemple simple.

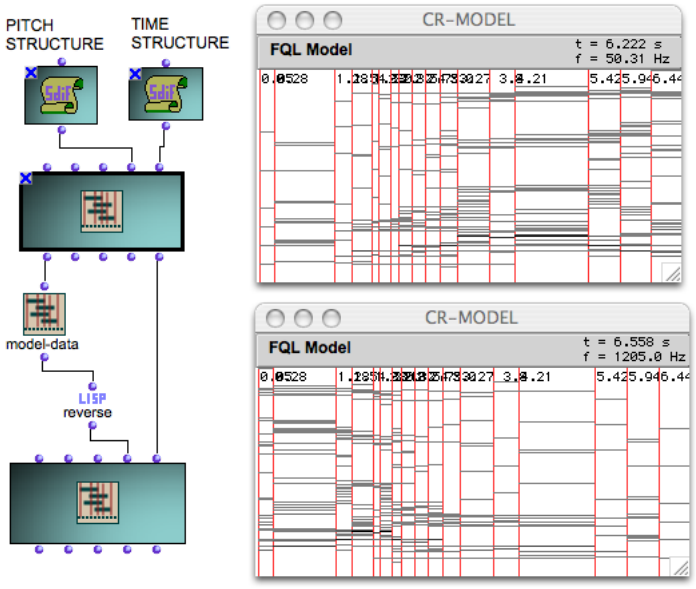


FIGURE 9.10: Traitement des données d'un `cr-model` : la structure de hauteurs d'un `cr-model` est extraite et inversée pour constituer la structure de hauteurs d'un nouveau `cr-model`, associé à la même structure temporelle.

La fonction `model-data` située entre les deux `cr-model` sur la figure 9.10 renvoie une liste de VPS issue du premier. Les éléments de cette liste peuvent être traités individuellement ; un deuxième argument optionnel de cette fonction (visible sur la figure 9.11) permet de spécifier une fonction auxiliaire qui sera appliquée à chacun. La fonction `model-data` ac-

cepte donc pour arguments 1) des données et 2) une fonction (en mode *lambda*) à appliquer sur ces données. Des fonctions prédéfinies plus ou moins élaborées sont disponibles à cet effet : filtres passe-haut, passe bas, transpositions, *frequency-stretching*, etc. (la figure 9.11 utilise par exemple un filtrage passe bas). Des fonctions définies par l'utilisateur sous forme de patches ou de combinaisons de plusieurs de ces fonctions prédéfinies peuvent également être utilisées pour spécifier localement le comportement de la fonction `model-data`.

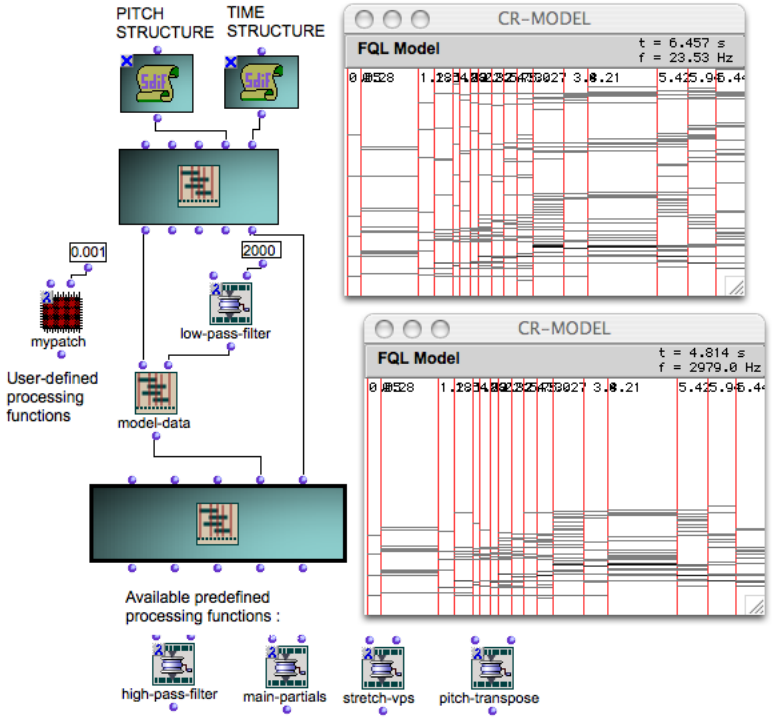


FIGURE 9.11: Manipulation des données de la structure de hauteurs : un filtre passe-bas (fonction `low-pass-filter`, réglée à 2000Hz) est appliqué à chaque VPS de la structure initiale lors de leur extraction. On voit en effet que les données du deuxième `cr-model` (en bas) ont été tronquées dans les hautes fréquences. D'autres traitements disponibles sont visibles en bas de la figure, et peuvent remplacer celui qui est connecté à la fonction `model-data`, tout comme le sous-patch visible sur la gauche.

Indépendamment des manipulations dans le domaine des hauteurs, la structure temporelle peut également faire l'objet de transformations. Celles-ci peuvent être appliquées directement sur la liste de valeurs constituant la structure temporelle (permutations, compressions, distorsions, etc.) Les *markers* peuvent aussi être convertis en durées, ou quantifiés en rythmes, sur lesquels seraient appliqués des traitements spécifiques, et ensuite reconvertis en valeurs temporelles absolues. Des fonctions spécifiques sont également prédéfinies, permettant d'appliquer par exemple des facteurs de compression variant dans le temps, ou d'insérer des variations aléatoires. La figure 9.12 montre des exemples de telles opérations sur la structure temporelle.

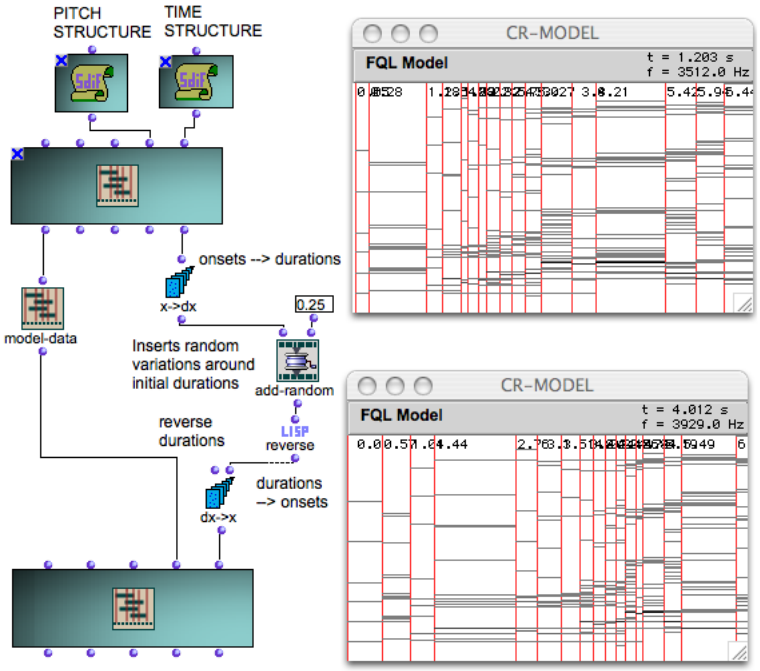


FIGURE 9.12: Manipulations sur la structure temporelle : les durées de la structure initiale sont renversées et associées à des petites variations aléatoires.

9.3.3 Connexion au domaine de la synthèse

La connexion de la structure du `cr-model` avec le domaine de la synthèse est assurée par l'intermédiaire des matrices présentées précédemment avec OMCHROMA (section 9.2), qui jouent le rôle d'évènements dans les processus de synthèse. La fonction `expand-model` établit une relation entre les données du `cr-model` avec les champs de description d'une classe de matrice donnée, qui déterminera donc le processus de synthèse visé. Chaque intervalle de la structure temporelle est successivement converti en une instance de cette classe, selon des règles d'appariement discutées dans la section suivante. La liste de matrices obtenue est synthétisée avec la méthode `synthesize` que nous avons rencontrée dans la section 9.2 (voir figure 9.13).

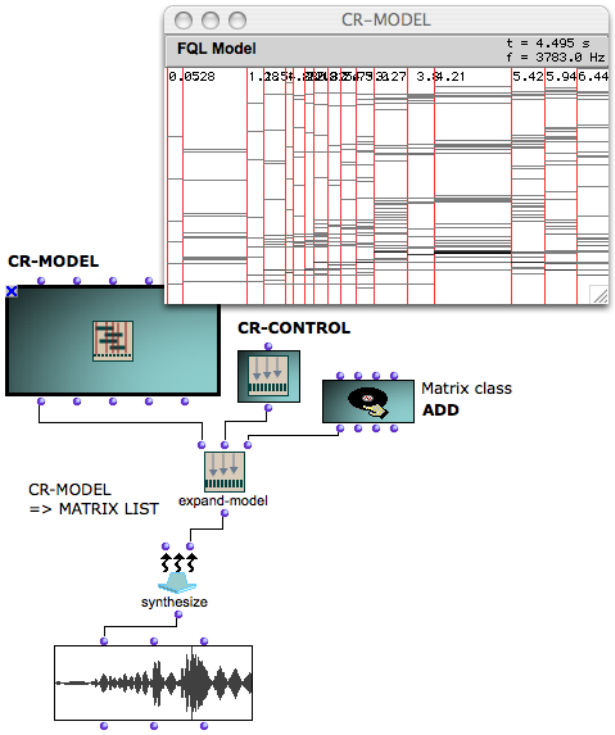


FIGURE 9.13: Conversion du contenu d'un `cr-model` en une liste de matrices, connectée directement à la fonction générique de synthèse. Le type de matrice produit est déterminé par l'objet connecté à l'entrée de droite de la boîte `expand-model`, et le processus d'appariement est défini dans l'objet `cr-control` connecté à la deuxième entrée.

9.3.4 Règles d'appariement des paramètres

Cette conversion des VPS en matrices est un autre point sur lequel le compositeur peut intervenir de façon significative. Il s'agit en effet de décider comment les paramètres du processus de synthèse (ou *slots* des matrices) sont calculés à partir des données d'un `cr-model`.

Sur la figure 9.13, nous avons pu remarquer les deux arguments supplémentaires de la fonction `expand-model`, complétant l'information fournie par le `cr-model` pour permettre la création des matrices. L'objet connecté au premier est appelé `cr-control`, et celui connecté au second est une matrice OMCHROMA. La matrice détermine la classe visée lors de la conversion ; le `cr-control` détermine les modalités de cette conversion.

Le `cr-control` définit donc une fonction, ou un ensemble de règles abstraites qui seront appliquées à chaque étape de la conversion du `cr-model` en une liste de matrices, c'est-à-dire successivement pour chaque segment décrit par sa structure temporelle. Il est associé à un éditeur particulier, permettant de définir et d'éditer graphiquement ces règles. La figure 9.14 en montre un premier exemple élémentaire.

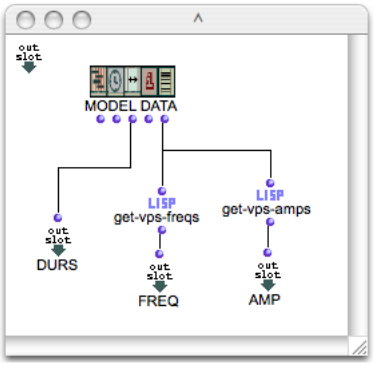


FIGURE 9.14: Spécification des correspondances entre les données d'un `cr-model` et les champs d'une matrice dans un éditeur de `cr-control`.

Dans cet éditeur, la boîte `MODEL DATA` contient des données issues du `cr-model`, destinées à être utilisées pour la génération des instances de matrice à chacune des itérations. On peut y trouver (par les sorties, de gauche à droite) une référence sur l'objet entier, sur sa durée totale, ansi que la durée, le rang et le contenu du segment courant. Pendant la conversion des segments en matrices, ces données seront donc mises à jour dynamiquement à chaque itération.

Les sorties de la fonction définie dans le `cr-control` sont représentées par des boîtes `out-slot`, ajoutées par l'utilisateur, et donc sont en nombre indéterminé. Sur la figure 9.14 nous avons trois sorties, correspondant aux trois paramètres pris en compte dans cet exemple : `DURS`, `FREQ`, et `AMP` (respectivement : durées, fréquences, et amplitudes). Les noms de ces sorties sont donnés par l'utilisateur, et doivent correspondre à des champs de description de la classe de matrice visée (plus précisément, aux noms des *slots* correspondants). Les valeurs qui y sont connectées seront donc considérées et appliquées lors de l'instanciation de cette classe, si celle-ci possède un *slot* du même nom.

La figure 9.14 représente donc des règles de conversion simples et directes entre le `cr-model` et la liste de matrices. Les fréquences des VPS sont assignées au *slot* `freq` de la matrice, les amplitudes au *slot* `amp`, et les durées des segments au *slot* `durs`.

Sur la figure 9.15 les règles sont plus élaborées. Dans cet exemple, un des *slots* (`amp`) est déterminé par une structure de donnée statique, indépendamment des données du `cr-model`. Les durées sont quant à elles étirées avec une multiplication systématique par 2.

Enfin, la figure 9.16 utilise des outils spécifiques pour des règles d'appariement encore plus complexes. En particulier, celles-ci considèrent la position relative (ou rang) du segment courant par rapport à l'ensemble du `cr-model` afin de calculer les valeurs des paramètres suivant des interpolations entre des données statiques prédéfinies.

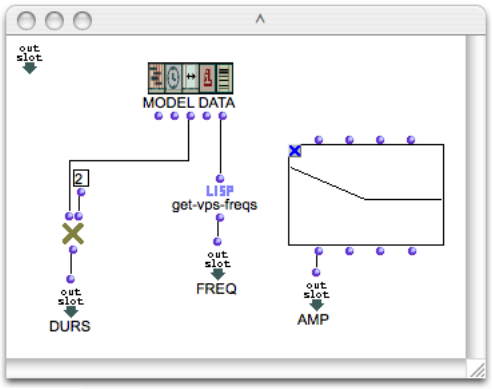


FIGURE 9.15: Appariement personnalisé entre les données du `cr-model` et les *slots* d'une matrice

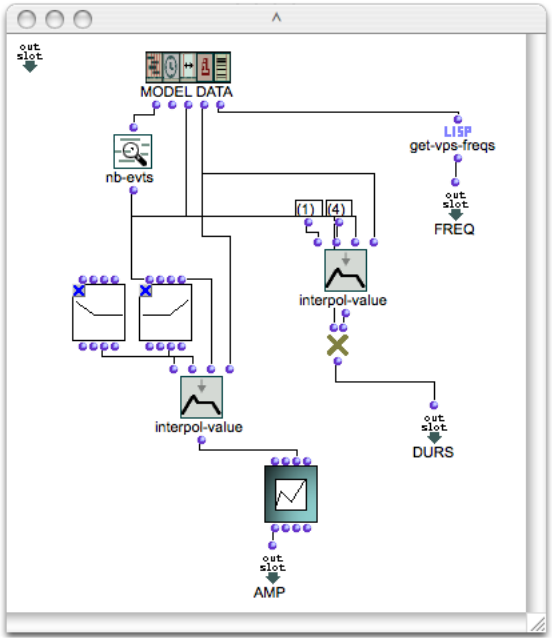


FIGURE 9.16: Utilisation de la position relative des segments pour le calcul d'interpolations : les durées sont multipliées par facteur évoluant de 1 à 4 suivant la progression du processus itératif, et les amplitudes évoluent d'un profil "passe-bas" vers un profil "passe-haut".

Etant donné que les *slots* des matrices OMCHROMA respectent une convention de nommage (par exemple `freq` pour les fréquences, `durs` pour les durées, `amp` pour les amplitudes, etc.) il est possible de connecter un `cr-model` à différents types de matrices, et avec le même processus d'appariement `cr-control`. De cette manière, il est fait abstraction du processus de synthèse visé, qui devient un élément variable du modèle compositionnel. La figure 9.17 illustre ce dernier point.

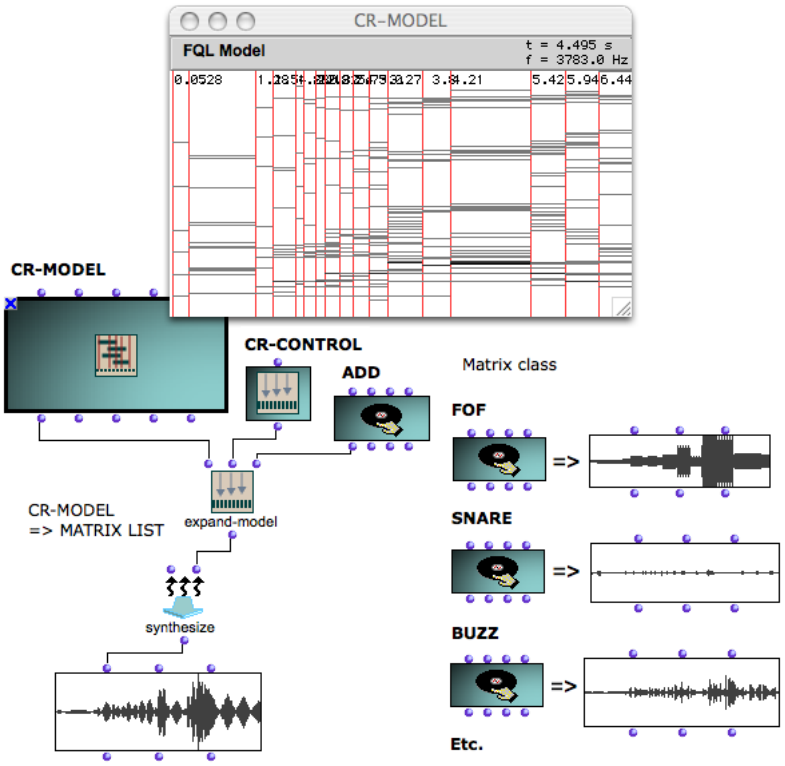


FIGURE 9.17: Conversion des données d'un `cr-model` vers différents types de matrices.

9.3.5 Mise en oeuvre dans les modèles compositionnels

Le système que nous venons de présenter offre de nouvelles possibilités pour la génération et la manipulation des données pour la synthèse sonore, et s'insère ainsi dans l'environnement de composition que nous avons décrit jusqu'ici. Associé aux outils d'analyse sonore présentés dans le chapitre 8, il pourra par exemple être intégré dans un processus complet d'analyse, traitement et resynthèse des données, structuré et homogénéisé par l'environnement de programmation (voir figure 9.18).

Notons qu'il ne s'agit pas d'essayer de resynthétiser fidèlement les sons analysés, ni d'en automatiser aucune extraction de connaissances intrinsèques utilisées pour la synthèse. L'intérêt de ce système est plutôt de permettre la structuration personnalisée des données et des processus, ainsi qu'une intervention constante du compositeur, à travers l'utilisation de fonction et de règles, dans les différentes étapes de traitement et de transfert du matériau musical.

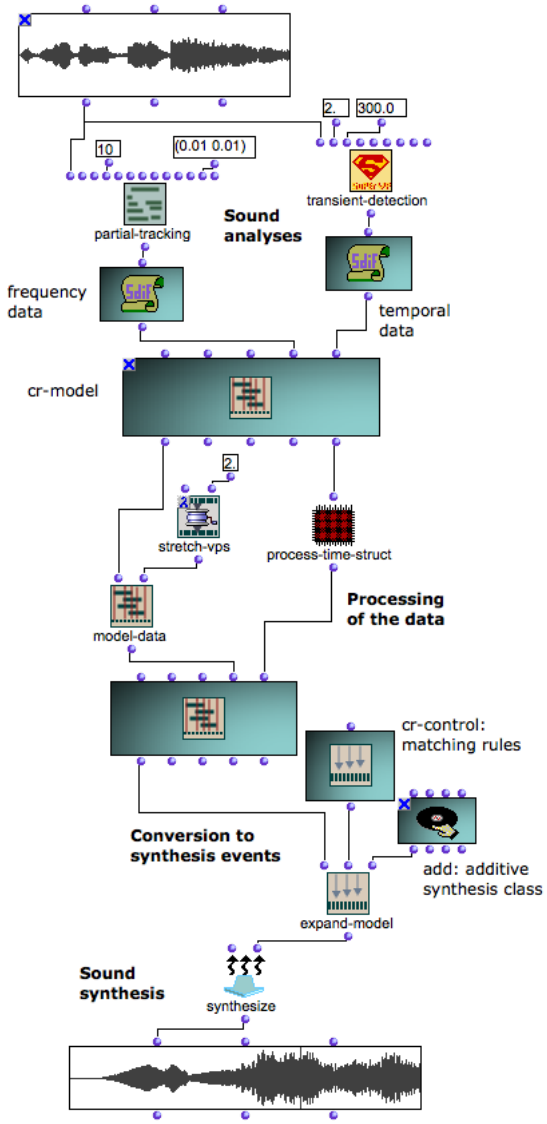


FIGURE 9.18: Un exemple de processus complet d'analyse / traitements compositionnels / synthèse sonore dans un même patch OPENMUSIC.

9.4 Conclusion

En essayant de recenser les différents types de données pouvant être utilisées pour la description et la manipulation des sons, nous avons rencontré au cours des derniers chapitres les échantillons sonores, les flux d'échantillons, les enveloppes, les matrices, jusqu'à arriver à des objets musicaux de plus haut caractère symbolique ou permettant de structurer les descriptions dans des formes plus conséquentes. Les structures et outils présentés dans ce chapitre permettent la création et la manipulation de ces données, principalement à l'aide du paradigme de programmation visuelle.

Aussi bien à travers ces outils que ceux du chapitre précédent, concernant l'utilisation des données issues de l'analyse, on notera que l'interaction directe de l'utilisateur avec les données (souvent complexes) est quasiment inexistante, mais opérée indirectement par des spécifications algorithmiques et/ou comportementales sur ces données. Finalement, il ne s'agit pas tant de spécifier des données que de définir le processus de génération, de traitement puis de conversion de ces données vers la synthèse et le son. Le processus de synthèse lui-même, nous l'avons vu, peut même être considéré comme un paramètre variable dans un processus de composition. Programmes et données sont ainsi intégrés dans des structures permettant d'appréhender à différents niveaux (forme globale – `cr-model`, évènement – matrice, paramètre, etc.) les propriétés structurelles et comportementales des objets sonores.

A mesure que nous avons avancé dans l'utilisation de ces structures, les questions relatives à la gestion et à l'organisation du temps se sont faites de plus en plus présentes. La notion d'évènement nous a permis dans un premier temps de nous orienter dans le temps, d'établir des relations de précédence et de synchronisation élémentaires entre les données à organiser dans le temps. Nous essaierons de pousser plus loin ces questions dans la quatrième et dernière partie de cette thèse qui traitera de l'organisation et de la représentation temporelles dans les processus et modèles compositionnels.

Quatrième partie

Structures musicales et organisations temporelles

Chapitre 10

Aspects temporels dans la composition et la synthèse sonore

La question de l'écriture avec la synthèse sonore rejoint fréquemment des problématiques liées au temps. Nous avons noté en particulier la nécessité d'un contrôle de l'évolution des paramètres sur la durée des processus, ainsi que l'importance des repères temporels fixes (évènements) permettant la structuration des formes musicales. Ces problématiques se retrouvent en certains points dans le domaine de la composition en général ; en d'autres points, elles sont spécifiques au domaine électroacoustique.

Nous essaierons dans ce chapitre d'identifier ces différentes problématiques, avant d'envisager dans le chapitre suivant des solutions pour une maîtrise des processus de synthèse dans le temps.

10.1 Structuration temporelle en composition assistée par ordinateur

La musique et ses différentes manifestations (sonore, en particulier) sont des phénomènes qui se déploient dans le temps. De ce fait, on considère souvent l'activité de composition comme étroitement liée à la manipulation et la structuration du temps. Celui-ci reste cependant problématique et fait l'objet d'interprétations et de conceptions diverses.

L'écriture du rythme et du contrepoint dénotent un effort datant du XIII^{e} siècle visant à formaliser le temps sous des formes plus abstraites, plus évoluées que le simple écoulement d'instants successifs. Depuis, le temps est resté une question centrale dans la recherche musicale, largement discutée aussi bien d'un point de vue musical que scientifique ou informatique (voir par exemple [Boulez, 1963], [Stockhausen, 1957], [Xenakis, 1981], [Grisey, 1987], [Accaoui, 2001] ou [Honing, 1993], [Desainte-Catherine, 2004])

10.1.1 Représentation des structures temporelles

[Accaoui, 2001] définit la "synthèse temporelle" comme une capacité à rassembler ou à retenir en une seule actualité un certain nombre de moments distincts :

> "Pour apparaître à la conscience [...] tout écoulement temporel doit tomber sous le coup d'une synthèse temporelle, d'un ensemble."

Pour l'auteur, cette synthèse temporelle nécessite l'existence préalable d'objets constitués dans le temps :

> "[...] on ne synthétise pas des unités temporelles sans synthétiser des unités substantielles, des unités objectives constituées." [Accaoui, 2001]

Nous nous appuierons dans un premier temps sur cette idée d'un temps constitué d'"unités objectives", pour nous focaliser sur des concepts relevant de l'organisation de ces unités (succession, simultanéité, et autres relations temporelles).

La formalisation (informatique) du temps va ainsi pouvoir aider par le calcul à décrire et former des structures temporelles complexes à partir d'objets élémentaires. Les différents formalismes utilisés à cet effet correspondent à des représentations temporelles particulières et constituent autant de conceptions de l'organisation sonore qui permettent de formuler et d'implémenter des relations et des opérations dans le temps sur ces objets, c'est-à-dire de construire avec ceux-ci un développement musical.

De nombreux formalismes et représentations temporelles ont été avancés (voir par exemple [Balaban et Murray, 1998], [Barbar *et al.*, 1993], [Bel, 1990], ou [Allen, 1991]), chacun permettant d'exprimer avec plus ou moins de facilité, d'élégance ou de puissance certains types de situations. En contrepartie, chacun peut aussi se montrer limité pour en exprimer d'autres. Il est difficile d'établir une classification, mais on peut en revanche isoler certains types de relations temporelles que l'on retrouvera dans les uns ou les autres de ces différents formalismes.

Séquence

La représentation la plus élémentaire d'une organisation temporelle consiste simplement à affecter à chaque élément (qu'on appellera *évènement*) une étiquette temporelle, une donnée de temps absolue (appelée généralement *onset*) indiquant sa date d'apparition dans un référentiel, ou dans une séquence (par exemple $t = 3$ secondes). A cette date est généralement associée une durée. L'organisation temporelle se trouve ainsi réduite à une succession chronologique d'évènements.

Dans cette représentation, chaque évènement est indépendant. Calculs et déductions ne sont donc pas nécessaires pour connaître leurs positions dans le temps (voir figure 10.1 - a). C'est donc une représentation relativement simple, précise et facilement éditable (c'est également celle qui est utilisée dans la plupart des séquenceurs musicaux actuels). Elle permet d'effectuer facilement des comparaisons, des tests de précédence entre les évènements : il suffit de faire des comparaisons entre les valeurs des *onsets* et/ou des

durées. Il est en revanche délicat d'y exprimer des relations entre ceux-ci (synchronisation, *offsets* relatifs, etc.) On ne peut pas, par exemple, décrire correctement une situation simple telle que "A commence en même temps que B" : il est possible de leur donner les mêmes valeurs d'*onset*, mais si l'un des deux est déplacé, cette assertion ne sera plus vérifiée. Il est donc difficile d'y construire des formes musicales complexes.

Les représentations cycliques du temps peuvent être vues comme des cas particuliers de cette représentation séquentielle, présentant un degré supplémentaire de structuration du fait d'une périodicité donnée.

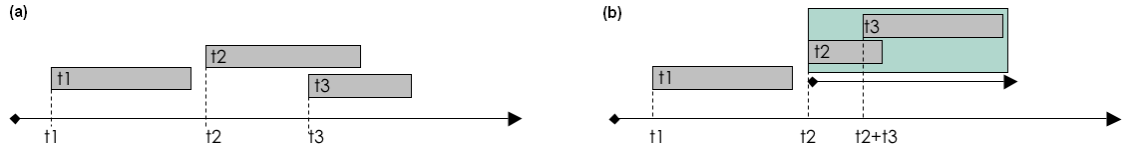


FIGURE 10.1: Représentations temporelles (a) séquentielle et (b) hiérarchique.

Hiérarchie

Certains inconvénients de la représentation séquentielle peuvent être résolus en introduisant la notion de hiérarchie. Les représentations hiérarchiques proposent des structures capables d'encapsuler d'autres structures. Une situation donnée est ainsi exprimée sous forme d'une arborescence, dont les noeuds peuvent être soit des évènements "terminaux", soit des structures composées, pouvant elles-mêmes contenir des évènements terminaux et/ou d'autres structures composées.

Dans le formalisme proposé par [Balaban et Murray, 1998], le temps est organisé par des *Elementary Structured Histories* (*ESH*) – objet (par exemple une action, un évènement) associé à une durée – et des *Structured Histories* (*SH*) – ensemble d'occurrences d'*ESHs* ou de sous-*SHs* à des positions temporelles données.[1]

Les positions des évènements deviennent donc relatives à une super-structure. On les exprime non plus de manière absolue mais par rapport à cette structure, qui représente un référentiel (voir figure 10.1 - b). C'est donc une représentation qui, tout en restant intuitive, permet de créer des formes plus structurées et mieux adaptées aux situations musicales.

Dans une certaine mesure, on peut en effet considérer la notation musicale traditionnelle comme une représentation hiérarchique du temps, avec les encapsulation de mesures, pulsations, et subdivisions. Dans OPENMUSIC, les rythmes sont d'ailleurs exprimés au moyen d'arbres rythmiques, qui sont des structures de données hautement hiérarchisées

[1]Ce formalisme a été proposé dans le cadre d'applications en intelligence artificielle, mais est également extensible aux situations musicales.

[Agon *et al.*, 2002]. Les objets musicaux en général y sont également construits sur un formalisme hiérarchique basé sur la notion de conteneur (*container*) [Assayag *et al.*, 1997b]. Un conteneur (polyphonie, séquence, mesure, accord, etc.) est un objet musical élémentaire ou contenant un ensemble d'autres conteneurs. Ses caractéristiques temporelles sont spécifiées par trois valeurs entières : un *onset* (o) – position de l'objet dans le référentiel de son propre conteneur, une durée (e), et une unité de subdivision (u) dans laquelle sont exprimées la durée et les *onsets* des objets qu'il contient. Ce système permet d'encapsuler de manière transparente les objets dans des niveaux hiérarchiques successifs. L'unité de subdivision (u) permet de faire coexister des entités et des systèmes d'unités temporelles hétérogènes, et de les ramener si besoin à une même échelle (i.e. dans une même unité).

La figure 10.1 met en évidence les différents rapports entre la date des évènements et leur position dans le flux temporel global : ils sont égaux dans la représentation séquentielle, mais pas nécessairement dans la représentation hiérarchique. Dans ce dernier cas, c'est la date relative de l'évènement, ajoutée récursivement à celle de la structure hiérarchique le contenant, qui donne la date "absolue".

Causalité

Les positions des évènements dans le temps peuvent également être déterminées par le calcul à l'intérieur d'un formalisme donné, introduisant la notion de causalité dans la représentation temporelle. Une telle représentation permet des interactions temporelles entre les objets, donc une modélisation puissante des situations musicales. Il est par exemple possible de définir les relations entre évènements selon un ordre de calcul indépendant du déroulement temporel. Dans l'exemple de la figure 10.2 - a, la position du bloc 1 est une fonction de celle du bloc 2. Le bloc 2 est donc antérieur au bloc 1 dans l'ordre du calcul, et lui est postérieur dans l'ordre temporel "physique". Ce type de représentation temporelle met donc en avant un temps d'écriture et de formalisation plus proche de la construction musicale (indépendant de l'ordre chronologique). En revanche, l'expression des relations (par l'utilisateur d'un système, par exemple) peut rendre son implémentation plus délicate.

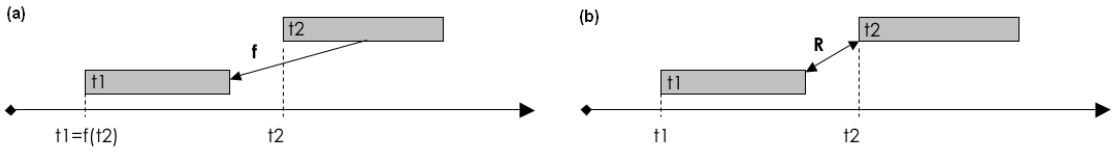


FIGURE 10.2: Relations temporelles (a) causale et (b) logique.

Logique

Les relations logiques permettent également d'envisager des interactions temporelles entre les objets, mais dans lesquelles le calcul n'est pas orienté : il n'y a pas la causalité que l'on a mise en évidence précédemment (voir figure 10.2 - b).

Parmi les travaux les plus célèbres sur la logique temporelle, James F. Allen a notamment énuméré 13 relations logiques élémentaires pouvant représenter l'ensemble des situations entre intervalles temporels (*before/after meets/met-by overlaps/overlapped-by starts/started-by during/contains finishes/finished-by equal*) [Allen, 1983].

Ce type de représentation peut être mis en place efficacement par des systèmes de programmation par contraintes. Dans un tel système, un CSP (*Constraint Satisfaction Problem*) représente un ensemble de contraintes (e.g. A doit commencer avant B, C doit se terminer en même temps que B, etc.) traitées par un système de résolution.

Une notion d'indéterminisme apparaît alors, du fait qu'une spécification de relations temporelles peut être satisfaite par un nombre variable de solutions (ou même ne pas être satisfaite, en cas d'inconsistance de cette spécification).

Dans [Beurivé et Desainte-Catherine, 2001] est par exemple discuté un système de résolution par contrainte pour les hiérarchies temporelles.[2] Le NTCC (*nondeterministic temporal constraint programming calculus*, [Palamidessi et Valencia, 2001]) est un autre exemple de calcul permettant la spécification et la résolution de situations temporelles exprimées sous formes de contraintes.[3]

10.1.2 Calcul des structures temporelles

"En-temps" – "Hors-temps"

La distinction *en-temps / hors-temps* mise en avant par Xenakis trouve un écho important dans le domaine de la composition assistée par ordinateur. [Xenakis, 1981] définit les structures "hors-temps" comme des objets possédant leur propre système d'engendrement, "par loi de composition interne", indépendant du temps et de l'ordonnancement qui sera établi. Le "en-temps" est une application entre ces structures "hors-temps" et des structures temporelles, par le biais d'une algèbre temporelle.

C'est la démarche qui est suivie implicitement lors de la création de matériau musical (selon des formalismes divers) et de l'organisation ultérieure de ce matériau dans le temps (selon des formalismes indépendants). Précisons cependant que si Xenakis avait plutôt tendance à insister sur la séparation des parties "en-temps" et "hors-temps", nous pourrons admettre qu'il existe en réalité des relations formelles entre ces parties (i.e. entre les structures "hors-temps" et l'organisation temporelle).

---

[2] Nous avons vu dans le chapitre 2 (section 2.4.2) une réalisation concrète utilisant ce système avec le logiciel BOXES.

[3] Un exemple d'utilisation pour le traitement du signal audio est proposé dans [Rueda et Valencia, 2005].

Une distinction similaire est reprise dans différents autres essais de formalisation du temps. [Balaban et Murray, 1998] distinguent par exemple *Atemporal Objects* (objets, évènements, processus, propriétés hors contexte temporel), et *Temporal Objects* (informations temporelles : dates ou intervalles) ; les *Structured Histories* étant considérées comme des associations entre objets atemporels et informations temporelles.

Le calcul du temps

Dans un environnement de programmation comme OpenMusic, les objets musicaux ont une temporalité propre, traitée comme un paramètre numérique dans le calcul (calcul et traitement des structures rythmiques, ou des valeurs d'*onsets* et de durées, par exemple). En s'appuyant sur la distinction formulée par Xenakis, ces objets, calculés dans les patches, pourraient donc être assimilés à des structures musicales "hors-temps", c'est à dire possédant leurs propres règles de composition interne.

La simple exécution d'un objet musical (par l'intermédiaire d'un instrument, d'un synthétiseur) constitue ainsi une première forme de "mise en temps" élémentaire.

En règle générale, ceux-ci devront cependant être insérés a posteriori dans un contexte temporel, dans une organisation globale, afin de faire partie d'une forme musicale. L'organisation externe des objets, leur mise en relation, suivant un formalisme temporel donné, dans un même déroulement (i.e. leur mise en temps), peut alors aussi être traitée par le calcul, dans les mêmes programmes. On pourra par exemple concaténer, superposer des objets, réaliser des structures hiérarchiques (en utilisant par exemple les mécanismes d'abstraction pour cumuler des *onsets*), utiliser des algorithmes divers, ou des CSP, pour calculer des formes rythmées, ou de simples valeurs temporelles assignées aux différents objets.

Cette approche posera cependant un problème de représentation ; les structures temporelles semblent en effet nécessiter une explicitation par des interfaces spécifiques pour pouvoir être exprimées et interprétées musicalement. Dans le chapitre suivant, l'utilisation des maquettes dans OpenMusic nous permettra de mettre les processus compositionnels dans une relation plus étroite avec leur contexte temporel, et d'améliorer notre système de modélisation par la représentation des structures temporelles.

Réduction séquentielle

Il est à noter que les représentations et structures temporelles que nous avons vues précédemment doivent toujours être ramenées à une représentation séquentielle chronologique au moment de l'exécution de la forme musicale créée.

Dans la description du système hiérarchique de [Balaban et Murray, 1998], on retrouve par exemple les *Performance Histories*, générées à partir d'une *Structured History* en ramenant celle-ci à une séquence d'évènements élémentaires.

Cette réduction en une représentation séquentielle est donc établie par le calcul, selon des méthodes déterministes ou non déterministes plus ou moins complexes dépendantes du formalisme choisi pour la représentation des situations temporelles.

Temps réel – Temps différé

Nous avons mis en évidence dans le chapitre 4 (section 4.2.2) la distinction entre les logiques de calcul des systèmes temps réel et des systèmes fonctionnels. Dans le cas d'un environnement de CAO comme OpenMusic, la logique est en effet opposée à celle du temps réel dans la mesure où on réalise l'évaluation d'un objet (plus précisément, d'une fonction) par évaluation récursive d'un graphe fonctionnel. Dans ce contexte, aucun délai maximum n'est garanti dans la terminaison des calculs (la latence entre l'appel d'une fonction et l'obtention du résultat) comme dans les environnements de temps réel. Les itérations, algorithmes de recherche, et autres processus pouvant y être implémentés peuvent prendre un temps indéterminé. On parle de calcul en temps "différé" ; les données musicales ne sont pas produites dans le même temps qu'elles peuvent être jouées et entendues.

Contrairement à ce que l'on pourrait imaginer, les environnements de temps "différé", dont fait partie OpenMusic, ne constituent cependant pas une catégorie "inférieure" aux environnements temps réel. Ceux-ci permettent en effet de mettre en œuvre des processus plus complexes contrôlant le son dans ses moindres détails (ces mêmes processus qui imposent le calcul en temps différé). Un tel système permet de mettre en place des structures élaborées, notamment à l'aide de formalismes temporels avancés dans une logique d'écriture indépendante de la chronologie (voir plus haut) et à plus grande échelle. Le temps réel, quant à lui, est à cet égard contraint à des représentations séquentielles.

L'indépendance du calcul vis-à-vis du temps qui passe, du temps "réel" de la musique, fait ainsi évoluer le calcul en temps différé dans une temporalité propre (celle de la composition) et lui permet de contrôler cette temporalité. Cette propriété est valable dans le cas de la CAO en général, mais également, et tout particulièrement, avec la synthèse sonore :

> "Le temps réel permet l'utilisation du dispositif en concert via un contrôle gestuel immédiat. Il facilite l'interaction du compositeur avec le synthétiseur et rend le choix et la détermination des paramètres plus intuitifs [...] A contrario le temps différé permet de développer des dispositifs plus complexes et de travailler plus finement le son." [Depalle et Rodet, 1993]

Malgré les intérêts que nous avons soulignés dans le contrôle de la synthèse en temps réel (interactivité, expérimentation sur les processus, compensation des pertes de variabilité et d'imprévu dues à l'électronique, etc.), la caractéristique de temps différé des systèmes auxquels nous nous intéressons constitue donc probablement ce qui permettra de donner une portée musicale aux processus de synthèse sonore.

> "La composition se situe typiquement dans le domaine du "temps non réel"." [Boesch, 1990]

Les travaux de Jean-Claude Risset se sont notamment positionnés dans ce sens :

> "C'est le temps différé qui permet d'étendre à la microstructure sonore des possibilités de notation et de travail proprement compositionnel." [Risset, 1993]

On trouvera également dans [Stroppa, 1999] ou [Bennett, 1990] des analyses des possibilités comparées, d'un point de vue musical, des systèmes de temps réel et de temps différé concernant la synthèse sonore.

Les liens entre les environnements interactifs temps réel (d'une utilité certaine pour la synthèse et l'interaction) et les environnements de calcul en temps différé constituent donc très certainement un champ d'investigation prometteur. On peut imaginer ces liens selon diverses modalités : un environnement de temps réel effectuant un rendu sonore de données calculées par un environnement de temps différé, ou inversement un environnement de temps réel sollicitant l'environnement de calcul en temps différé pour lui déléguer le calcul de structures musicales (temporelles) complexes.[4]

10.2 Le temps et la synthèse sonore

Dès les premières expérimentations dans le domaine de la synthèse sonore, le temps a été identifié comme un paramètre structurant fondamental. Stockhausen, en 1957, décrivait le timbre sonore comme étant *le résultat d'une structure temporelle* [Stockhausen, 1988a]. Il allait déjà loin dans cette idée, en assimilant les hauteurs à des "phases de temps". Selon ce principe, toute activité relevant de la création de timbres ou de sons de synthèse doit traiter essentiellement de temps.

Plus tard, les recherches portant sur les timbres synthétiques, menées notamment par Max Mathews, Jean-Claude Risset [Risset et Mathews, 1969] [Mathews et Kohut, 1973], John Grey [Grey, 1975], et bien d'autres, ont mis en évidence l'importance de descripteurs temporels, tels que les transitoires d'attaques ou les relations entre les évolutions temporelles de paramètres spectraux, dans la perception des différentes classes de timbres.

> "Le son se conçoit comme un processus temporellement orienté dont toutes les phases sont interdépendantes, dont les éléments interagissent entre eux au cours de l'évolution du processus." [Dufourt, 1991]

Parmi ces correspondances temporelles, des résultats significatifs ont par exemple été apportés concernant la perception des timbres "cuivrés", dont l'étendue spectrale varie de manière parallèle à la dynamique (plus le son est fort, plus le son se déploie dans le domaine des hautes fréquences), ou encore des sons d'instruments comme le violon, dont les résonances harmoniques dépendent fortement (et sont très variables en fonction) de

[4] C'est l'un des axes des recherches menées actuellement dans l'équipe Représentations Musicales, notamment autour de l'improvisation [Assayag *et al.*, 2006].

la fréquence, singularisant les gestes de type "vibrato". Adressant les problèmes de la description du son selon ses caractéristiques perceptives, [Wessel, 1978] considère aussi la distribution spectrale de l'énergie mais surtout la variation temporelle de cette distribution comme déterminants acoustiques de la perception des sons.

Les problèmes de la manipulation du temps dans les processus de synthèse sonore, et du déploiement même de ces processus dans le temps, se sont donc établis parmi les préoccupations des compositeurs de musique électroacoustique ou mixte :

> "Le principal mode de communication [entre instruments et électronique], le principal problème, me semble être dans le domaine temporel." Ph. Manoury

Pour [Risset, 1993], l'enjeu (musical) est de *"faire jouer le temps dans le son et pas seulement le son dans le temps"*.

10.2.1 Niveaux d'intégration

La translation de l'activité compositionnelle vers le son et la synthèse sonore apporte un certain nombre de questions spécifiques au niveau de l'approche et de la conception du temps. L'une des premières est celle des niveaux d'intégration : il existe dans les processus de synthèse un niveau dans lequel le contrôle des infimes variations des paramètres sonores doit être assuré ; ce niveau devant lui-même être intégré dans un ou plusieurs niveaux supérieurs permettant la structuration de formes musicales à la fois plus abstraites et plus conséquentes. Des relations structurelles existent entre ces niveaux macro- et micro-compositionnels (et les éventuels niveaux intermédiaires), mais leurs caractéristiques propres peuvent rendre ces relations problématiques :

> "Il y a des niveaux d'intégration qui créent des réalités spécifiques, et qui ont leurs propres lois" H. Dufourt

Le plus évident de ces niveaux d'intégration est celui de l'organisation d'objets sonores sur un axe temporel : ce niveau n'est pas spécifique à la synthèse sonore et peut être assimilé aux questions des structures temporelles utilisées pour la composition musicale en général (y compris instrumentale), comme celles que nous avons citées au début de ce chapitre. Avec l'écriture des processus de synthèse sonore se pose cependant de manière plus récurrente la question de la nature des objets sonores qu'il est question d'organiser. Comme le rappelle [Honing, 1993], la notion même de structuration dépend de la possibilité de décomposer une représentation en entités significatives, que l'auteur appelle *primitives*. Cette décomposabilité nécessite donc la détermination des primitives de représentation, qui ne sont plus nécessairement, avec la synthèse, des objets musicaux ou sonores symboliques et constitués (par exemple des notes), mais pourront être n'importe quel aspect d'une représentation du son.[5]

[5] Nous avons vu dans le chapitre précédent (section 9.3) un cas particulier d'une telle décomposition en primitives, correspondant à une démarche compositionnelle spécifique.

On rentre alors dans un autre niveau, moins symbolique et plus proche de celui du signal. Les éléments qui s'y rapportent doivent cependant être intégrés dans la démarche et le modèle compositionnels, à travers la structure temporelle.

Dans [Boulez, 1987], Pierre Boulez souligne par ailleurs l'importance de la cohérence interne et externe dans le matériau sonore, entre les niveaux qui sont celui de l'intérieur des objets ("*within objects, in order to build them*"), et celui de l'extérieur de ces objets ("*from the outside, in order to organize them*") :

> "Les critères externes peuvent agir sur sur les critères internes afin de les lier dans un développement cohérent et de les positionner dans un contexte formel." [6]

On perçoit donc la nécessité de la mise en place de structures permettant d'aborder les différents niveaux d'intégration des processus de synthèse sonore et de composition musicale.

> "[...] il ne doit pas y avoir d'un côté des sons figés et de l'autre une structure externe qui les organise. [...] au contraire, les sons doivent se former en fonction de la place qu'ils occupent dans un certain contexte et alors la structure naît de leurs différentes actualisations dans ce contexte, au point que chacun peut influer en retour sur la mise en jeu de tous." [Rodet *et al.*, 1985]

La richesse et la cohérence du matériau sonore produit passe ainsi par une cohérence dans les processus mêmes de modélisation de ce matériau dans son contexte temporel.

> "Ce qui rapproche le son de l'automate, c'est bien ce modèle d'autorégulation par lequel le processus favorise sa propre reproduction, maintient sa propre coordination et permet de stabiliser un processus dynamique dans la permanence d'une forme." [Dufourt, 1991]

10.2.2 Temps discret et temps continu

Les données utilisées pour paramétrer les processus de synthèse sont pour la plupart des valeurs évoluant dans le temps, dont la précision est par ailleurs d'une grande importance. Les processus de synthèse traitent du temps au niveau des échantillons sonores : il s'agit de générer les valeurs d'une forme d'onde numérique, représentant une vibration acoustique continue. Le taux d'échantillonnage de représentation du phénomène acoustique doit être aussi élevé que possible afin d'atteindre une qualité sonore satisfaisante, c'est-à-dire de simuler la continuité de cette onde acoustique. Même si on utilise généralement des taux d'échantillonnage inférieurs pour les paramètres de synthèse, l'approche des objets sonores nécessite donc une précision temporelle allant au delà des seuils de perception.

En supposant que la continuité puisse être convenue à partir du moment où la perception ne discerne pas d'entités discrètes à l'intérieur d'un phénomène, une problématique

[6] *"External criteria can act on internal criteria and modify the objects in order to link them in a coherent development and place them in a formal context."* [Boulez, 1987]

majeure liée au temps et à la synthèse sonore pourrait donc relever d'une opposition entre les paradigmes du temps discret et du temps continu.

Il n'est cependant pas toujours évident de discerner dans les phénomènes sonores et leurs processus constitutifs ce qui relève du temps discret et du temps continu : si un son en tant qu'onde acoustique est continu, le signal numérique est par essence et en principe un objet relevant du domaine discret (une suite d'échantillons numériques). C'est donc pour traiter ce type d'ambiguïté que nous considèrerons le domaine discret comme celui dans lequel les objets manipulés pour la création de structures musicales sont délimités et ont une signification propre (pour celui qui les utilise), alors que dans celui du continu, ces objets ne sont ni délimités ni individuellement discernables.

> "Une vibration constitue bien une évolution, mais la perception me la donne comme une chose intégrée. Le système perceptif sépare effectivement ce qui est évoluant mais perçu comme continuum, et ce qui est une évolution de ce continuum." C. Cadoz

On déplace par là le propos scientifique dans la distinction discret/continu, selon lequel le son numérique relèverait donc déjà du domaine discret, vers le domaine musical. Dans ce domaine, la continuité des objets devient problématique, dès lors que l'on souhaite les structurer dans un but musical.

Boulez a parlé d'opposition entre espaces striés et espaces lisses : le strié organise des formes distinctes à l'aide de repères et de valeurs quantifiables, alors que le lisse a trait à la variation, au développement continu [Boulez, 1963]. Le temps strié se rapproche ainsi des constructions rythmiques, en référence à une pulsation, alors que le temps lisse correspondrait à un temps absolu, sans repères particuliers. Cette opposition s'apparente donc, à une autre échelle, de celle qui nous intéresse ici, en ce que le temps lisse y est associé à la notion de continuum. Entre les deux, Boulez introduit la notion de coupure, permettant de décomposer le continu en sous-continus distincts. Un continuum auquel est associée la possibilité de couper l'espace contient ainsi, potentiellement, à la fois le continu et le discontinu. L'aspect discret (ou discontinu) des structures fournit ainsi des repères pour les transitions et les séparations, et permet l'élaboration de formalismes et processus compositionnels.[7]

L'opposition discret/continu se rapproche donc d'une certaine manière aux oppositions symbolique/subsymbolique ou symbolique/signal que nous avons évoquées dans le chapitre 5. La discrétisation implique en effet la discrimination d'éléments cernés parmi un continuum, le traitement de ces éléments étant rendu possible par leur éventuelle fonction symbolique.

Avec la définition de ces éléments (comme les notes dans le domaine instrumental), la discrétisation des phénomènes sonores (continus) en objets symboliques permet à l'écriture de se positionner dans sa fonction organisatrice du temps. Composer avec la synthèse sonore ne se limite donc pas au positionnement dans le temps d'objets musicaux ; il s'agit

[7] *"[discontinuity] gives rise to composition in its many dimensions."* [Boulez, 1987]

également d'accéder à la structure continue, à la texture, afin d'établir les éléments constitutifs d'un niveau symbolique.

> "Cette dichotomie [entre la structuration discrète, symbolique, des paramètres musicaux, et les effets continus inhérents a la synthèse et au traitement des sons] implique des systèmes de spécification et de contrôle inédits, et représente un enjeu majeur de la recherche musicale." [Assayag et Cholleton, 1994]

Notons enfin que dans une forme musicale, discret et continu ne forment pas nécessairement un ensemble bipolaire, mais peuvent être mis en abîme l'un dans l'autre. Une texture sonore qui relèverait du domaine continu peut constituer un élément discret si elle est cernée ou délimitée (par exemple dans le temps) ou simplement en présentant un point d'ancrage (pivot) permettant de manipuler cette structure continue comme un objet fini. De même un ensemble important d'éléments discrets peuvent s'assembler et former une texture perçue comme continue.

10.2.3 La notion d'évènement

La notion d'évènement que nous avons utilisée dans le début de ce chapitre nécessite quelques précisions dès lors que l'on se place dans un contexte de musique électroacoustique. Nous avons vu qu'elle gardait en effet toute sa pertinence au niveau musical, reflétant la conception du temps discret favorable à la composition ; les évènements sont les éléments que l'on peut manipuler dans le temps à un niveau symbolique.

Un évènement est donc, par sa fonction symbolique, l'équivalent de ce que pourrait être une note en musique instrumentale traditionnelle. Mais il pourra, selon les cas, prendre des formes diverses.[8] [Allen et Ferguson, 1994] définissent un évènement comme *la manière dont un agent classifie un changement.*[9] Un évènement implique alors un phénomène ou un changement d'état localisé dans le temps. Il peut donc s'agir du début d'un son dans une forme à grande échelle, d'une note (un élément sonore ayant une hauteur et une durée perceptible), d'une variation d'énergie dans une région spectrale, du début d'une transition continue, etc.

La définition de la nature de l'évènement devient donc floue. Surtout, elle devient un choix dans le positionnement du compositeur par rapport au processus de synthèse : *un évènement capture une certaine manière de penser un son*[10] [Stroppa, 2000].

[Bel, 1990] définit également un évènement comme un segment du son auquel serait assigné (subjectivement) un sens musical.[11] Cette notion, liée à celle de segmentation, est étendue dans la mesure où les évènements peuvent se recouvrir partiellement, voire se

[8] Nous retrouvons le problème de la décomposition en entités significatives, ou primitives évoqué précédemment avec [Honing, 1993].

[9] *"the way by which agents classify certain patterns of change"* [Allen et Ferguson, 1994]

[10] *"Each "event" captures a certain way of thinking about a sound"* [Stroppa, 2000]

[11] *"A musical event is a segment of sound to which some musical 'meaning' is assigned."* [Bel, 1990]

superposer. L'auteur définit par ailleurs des structures de plus haut niveau qu'il appelle *components* (composants) correspondant à des ensemble d'évènements liés les uns aux autres, et participant ensemble à une structure temporelle. Un composant peut également contenir des sous-composants : on retrouve la structure hiérarchique, caractéristique primordiale des représentations temporelles pour la musique.

Il existe cependant un aspect parallèle à cette description hiérarchique qui est la dualité entre l'évènement et la structure composée, autrement dit entre un objet et son contexte.

> "Il se joue une dialectique entre la complexité de l'objet et celle de son contexte. [...] Lorsque l'objet est trop complexe pour pouvoir être intégré dans un contexte en tant qu'élément de celui-ci, il devient lui-même ce contexte." [Manoury, 1990]

Nous avons pu rencontrer ce problème dans OpenMusic avec l'utilisation des matrices dans OMChroma, par exemple (Chapitre 9). Dans une matrice correspondant à une synthèse additive, par exemple, on pourra considérer les éléments de celle-ci comme des évènements (partiels) que l'on organise par le paramétrage de la matrice (section 9.2). A partir d'un certain niveau de complexité, cependant, c'est la matrice elle même qui doit être considérée comme un évènement et insérée dans une structure de plus haut niveau (section 9.3).

10.2.4 Objets et phénomènes continus

Considérons un objet continu comme une fonction du temps ; cette fonction pouvant simplement décrire l'évolution d'un paramètre de synthèse, ou bien définir une relation complexe entre plusieurs paramètres ou autres structures d'un programme.

Nous avons vu que si les évènements favorisaient le traitement musical symbolique, la synthèse sonore nécessitait cependant des structures plus fines et plus précises afin de générer des morphologies sonores complexes. Les phénomènes continus peuvent alors intervenir à l'intérieur ou entre les évènements discrets (respectivement comme expression d'un mouvement ou d'une transformation interne, ou comme expression d'une articulation, d'une transition d'un évènement vers un autre). Ils sont la clé, pour peu que l'on soit en mesure de les contrôler, d'une grande partie du potentiel de la synthèse sonore en termes d'écriture du son.

Dans ces phénomènes pourront cependant se démarquer des repères discrets, ou discontinuités, permettant de les manipuler, pour contrôler leur position, ou leur comportement par rapports à d'autres objets dans une structure donnée. Ainsi [Stroppa et Duthen, 1990] proposent par exemple un système d'organisation d'objets musicaux basé sur les pivots temporels, points d'ancrage pour la structuration temporelle d'objets non (explicitement) délimités dans le temps. On essaie ainsi de maintenir une certaine idée de discontinuité, qui tend vers celle de décomposition en évènements (ou de coupures).

> "L'acoustique numérique est amenée à considérer que les situations intermédiaires entre des états stables ont plus d'intérêt que ces états eux-mêmes. [...] Les états stabilisés deviennent alors des repères idéaux permettant de délimiter la réalité des états transitoires du son." [Dufourt, 1991]

Le problème est donc d'établir une relation entre la continuité des processus temporels et une représentation symbolique (discrète) permettant sa structuration et sa modélisation. En d'autres termes, être capable par des moyens (données et processus) symboliques d'engendrer des objets temporels continus à partir de repères (évènements) discrets.

10.2.5 Sur le temps dans les outils de synthèse sonore

Comme le souligne [Garnett, 1991], les architectures logicielles des programmes de synthèse les rendent généralement plutôt adaptés à l'un ou l'autre des paradigmes temporels discret ou continu.

[Loy, 1989] différencie les deux voies qui se sont historiquement développées dans l'exploration du timbre et de la microstructure du son : dans celle engagée par Mathews avec les MUSIC N [Mathews, 1969], le concept de note est préservé en tant que spécification acoustique ; il y a donc une séparation de la microstructure (au niveau du signal, des "instruments" de synthèse) et de la partition (même si les deux peuvent influer l'un sur l'autre). Les problématiques relèvent alors plutôt du temps discret, de l'organisation d'évènements, semblables à celles énoncées dans la section 10.1.1, mais à une échelle interne au son, donc beaucoup plus petite. Les environnements basés sur les évènements font toutefois appel a des fonctions du temps continues (dynamique, variations de tempo, etc.) Cependant, étant donné que ces fonctions n'ont pas besoin d'être évaluées en permanence, elles sont généralement spécifiées par des structures discrètes (dans les cas les plus simples, par des fonctions par segments, ou *breakpoint functions*). Pour spécifier des fonctions du temps plus élaborées, [Anderson et Kuivila, 1989] proposent par exemple, dans le système FORMULA, la notion de concaténation procédurale, qui permet de spécifier des fonctions complexes par composition de fonctions élémentaires définies individuellement dans le domaine continu.

A l'opposé (pour reprendre l'analyse de [Loy, 1989]), dans la voie héritée du sérialisme européen, le son est considéré comme un flux unique indifférencié soumis à un travail compositionnel global et uniforme (allant des échantillons sonores jusqu'à la macrostructure musicale). Il y a alors une notion de continuité, avec des transitions entre états qui créent une forme sonore globale. Les architectures logicielles correspondantes sont alors basées sur des processus plus ou moins complexes calculant périodiquement les valeurs d'un signal en fonction d'états de référence spécifiés par un niveau de contrôle. C'est le cas notamment du système FORMES/CHANT [Rodet *et al.*, 1984] [Rodet et Cointe, 1984], ou des langages fonctionnels comme ARCTIC [Dannenberg *et al.*, 1986].

10.3 Conclusion

La question du temps en informatique comme en musique est une question délicate, et à ce titre a souvent été controversée. Si les thèmes abordés dans ce chapitre sont donc sujets à discussion, ils nous auront permis d'identifier quelques points importants pour la continuation de nos travaux.

Le déploiement et l'organisation dans le temps des processus compositionnels faisant appel à la synthèse sonore se présentent en effet comme des problématiques majeures, compte tenu des différents repères temporels et des interactions (transitions, échanges de données, etc.) pouvant intervenir dans les relations entre les objets et programmes de synthèse. La modélisation temporelle de ces processus se situera donc au croisement de plusieurs domaines. Il s'agira de définir des composants sonores au niveau microscopique (domaine continu), et de les articuler à un niveau macroscopique (domaine discret, logique et hiérarchisé). Il y a donc avec la synthèse une question d'échelle dans le champ d'action compositionnel (allant du niveau des formes musicales jusqu'à celui des phénomènes sonores) qui s'accompagne d'une transition entre les domaines discret et continu, distincts en termes de traitement et de formalisation.

L'utilisation de la *maquette* comme outil de représentation et de contrôle de la synthèse dans OpenMusic nous permettra de mettre en place, dans le calcul et la synthèse des sons, des structures temporelles avancées tenant compte de ces différentes observations.

Chapitre 11

Modélisation et contrôle des processus de synthèse dans le temps

Malgré l'importance des structures temporelles dans la synthèse sonore et ses applications en composition musicale, que nous avons discutées dans le chapitre précédent, les programmes de synthèse et de contrôle de la synthèse existant permettent pour la plupart des manipulations assez primitives dans ce domaine. Dans ce chapitre, nous proposons l'utilisation de la *maquette* dans OpenMusic, que nous avons présentée rapidement dans le chapitre 4 (section 4.3.5), pour traiter les questions de temps dans les processus compositionnels, et en particulier dans ceux liés à la synthèse sonore.

Les concepts de la maquette ont été présentés initialement dans [Assayag, 1995] puis développés dans [Assayag *et al.*, 1997a] et [Agon, 1998]. Il s'agit d'une extension de la notion de patch dans laquelle la position des objets prend un sens, en particulier un sens temporel. Nos travaux dans OpenMusic nous ont amené à finaliser cet outil ainsi que son insertion parmi les autres composants de l'environnement de programmation. Nous présentons donc dans un premier temps les principes généraux et le fonctionnement de la maquette (section 11.1) ainsi que des remarques préalables concernant son utilisation avec la synthèse sonore (section 11.2). Les sections 11.4 et 11.3 traiteront ensuite des fonctionnalités étendues créées dans le cadre spécifique de cette thèse afin de traiter le problème de la synthèse sonore.

La maquette nous permettra ainsi de structurer les modèles compositionnels en y incluant une dimension temporelle. Elle concrétisera le positionnement d'un niveau de contrôle symbolique intégrant la notion d'évènement, en relation à une gestion des phénomènes continus à l'intérieur ou entre ces évènements.

11.1 La *maquette*

11.1.1 Un séquenceur de programmes

Dans une première approche, une maquette peut être vue comme un séquenceur dans lequel sont organisés dans le temps des objets, mais aussi les programmes (patches) produisant ces objets. Il s'agit donc sous ce point de vue d'un séquenceur de programmes. Chacun de ces programmes est représenté par une boîte (appelée *TemporalBox*) dont la position sur l'axe horizontal de la maquette définit la position dans le temps (*onset*) de l'objet correspondant, et dont l'extension définit la durée.

Les *TemporalBoxes* peuvent donc se référer soit à un programme (patch produisant une sortie musicale) ; soit à un simple objet (correspondant au cas d'un programme renvoyant une valeur constante). Chaque *TemporalBox* peut ainsi être considérée (représentée, éditée, etc.) soit en tant que programme, soit en tant qu'objet (résultant de ce programme). La dualité entre objets et processus, au cœur de l'environnement de CAO, est ici encore renforcée par cette double représentation des programmes insérés dans le contexte temporel.

Les patches présents dans une maquette ont donc un statut particulier puisqu'ils sont censés produire un résultat musical à intégrer dans le contexte temporel du séquenceur (on les appelle *TemporalPatches*). Leur première caractéristique est l'existence d'une sortie spéciale (appelée `TempOut`) chargée de transmettre ce résultat musical à la maquette pour une intégration dans ce contexte. La figure 11.1 montre un exemple de *TemporalPatch* dans une maquette.

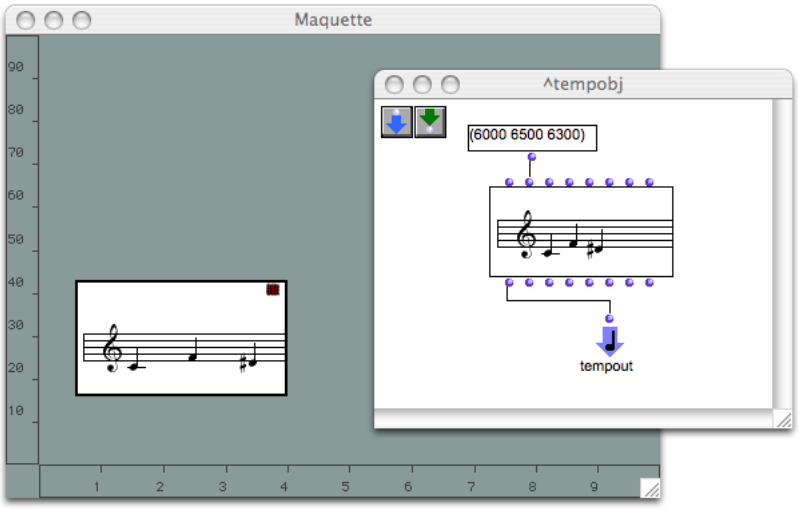


FIGURE 11.1: *TemporalPatch* : un patch présentant une sortie spéciale (*tempout*) intégrée dans le contexte temporel d'une maquette.

Une deuxième caractéristique propre aux programmes placés dans le contexte d'une maquette est le fait pour ces programmes de disposer, à l'intérieur même du calcul, d'un accès en lecture et/ou écriture sur les caractéristiques (position, taille, couleur, etc.) de la

boîte (*TemporalBox*) à laquelle ils sont associés. En d'autres termes, le résultat du calcul d'un patch (par exemple l'objet musical renvoyé) peut dépendre de la position (aussi bien dans le temps, sur l'axe horizontal, que sur l'axe vertical, qui prend ici son sens) ou des autres propriétés de la boîte correspondante. Ceci est illustré sur la figure 11.2.

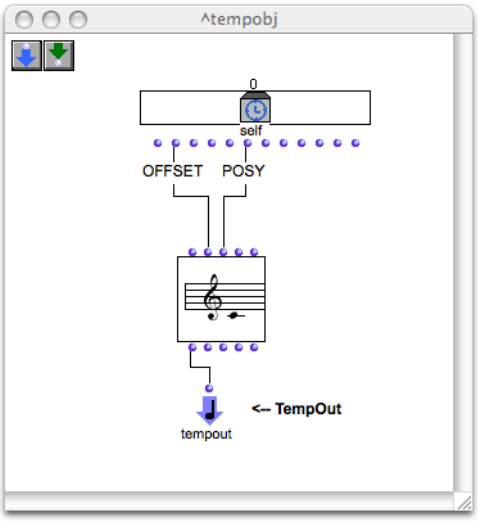


FIGURE 11.2: Utilisation des propriétés des boîtes dans le calcul des objets musicaux. Dans cet exemple, l'*offset* de la boîte (position dans le temps) détermine la hauteur de la note renvoyée par le programme, et la position verticale (*posy*) détermine son amplitude.

La boîte `self` sur la figure 11.2 représente cet accès du programme aux différents attributs (notamment graphiques et temporels) de la *TemporalBox* appelant ce programme : `offset` (position de la *TemporalBox* sur l'axe temporel de la maquette qui la contient), `extend` ("durée" de la *TemporalBox*), `posy` (position sur l'axe vertical), `sizey` (taille selon ce même axe), `colorframe` (couleur de la *TemporalBox*), `reference` (patch ou objet auquel la *TemporalBox* fait référence), `value` (résultat musical créé en évaluant l'objet de référence), etc. Cette boîte `self` est optionnelle et pourra être insérée dans n'importe quel patch pour lui donner un accès sur son (éventuel) contexte temporel.

Ceci constitue une caractéristique particulièrement intéressante du modèle fonctionnel des maquettes, qui va permettre d'une part au calcul de décider de l'organisation temporelle, et d'autre part à l'organisation temporelle (et graphique) d'influencer le calcul des objets. Les programmes sont donc insérés dans un contexte temporel mais peuvent aussi traiter l'information temporelle, faisant interagir le temps du calcul des processus, le temps musical des objets, et la structure temporelle globale [Assayag *et al.*, 1997b].

L'abstraction des patches permet la réutilisabilité des programmes ou des composants fonctionnels. Il est important de permettre aussi bien l'utilisation de patches locaux ou abstraits (voir la section 4.3.4 du chapitre 4) à l'intérieur des maquettes. Les transferts et inclusions des différents types de programmes les uns vers les autres sont par ailleurs transparents et sans restrictions.

La possibilité d'utiliser les abstractions comme fonctions génératrices d'objets musicaux, nous le verrons dans des exemples à suivre, permettra la création de "*templates*" d'objets temporels dans les programmes de synthèse, c'est-à-dire des objets fonctionnels pouvant être dupliqués et éventuellement produire des résultats différents selon les positions et autres propriétés des copies, toutes ces copies étant contrôlées par un programme unique (cela s'avèrera particulièrement pertinent lors de l'utilisation de nombreux éléments de comportement et nature similaires dans des structures temporelles plus complexes).

11.1.2 Un objet musical

L'axe temporel de la maquette lui donne une propriété d'objet musical en soi : elle est une sorte de *conteneur*, intégrant les résultats de tous les programmes qu'elle contient. Lors de l'évaluation d'une maquette, toutes les boîtes contenues dans la maquette sont évaluées, et ce *conteneur* (appelé `maquette-obj`) est construit, fusionnant les objets résultants dans une structure séquentielle linéarisée. Cet objet a un statut d'objet musical au même titre que les autres (`chord-seq`, `sound`, etc.)

Une maquette peut ainsi figurer à l'intérieur d'un programme (patch) sous forme d'une boîte. Celle-ci permettra sa construction, admettant une liste d'objets et une liste d'*onsets* comme arguments, et renvoyant comme résultat le conteneur musical construit par l'évaluation de la maquette (voir figure 11.3).

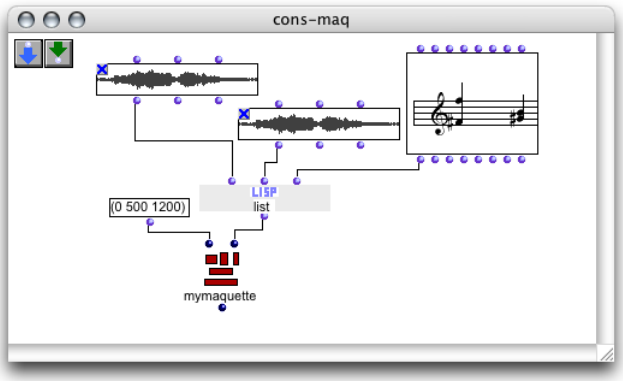


FIGURE 11.3: Construction d'une maquette à l'intérieur d'un patch.

Un troisième type de *TemporalBox* peut ainsi être envisagé, se référant cette fois à une maquette. Cette possibilité permet de créer des structures hiérarchiques (voir figure 11.4).

11.1.3 Un programme

Une maquette est avant tout une extension du patch. Elle a donc également (et essentiellement) les caractères d'un programme. Les patches associés aux différentes *TemporalBoxes* d'une maquette peuvent en effet, suivant le modèle fonctionnel des programmes visuels, présenter des entrées et des sorties afin de constituer des fonctions pouvant être

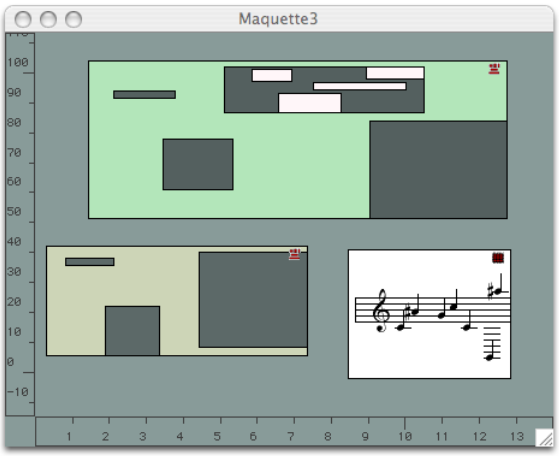


FIGURE 11.4: Imbrication de maquettes : construction de structures temporelles hiérarchiques.

connectées les unes aux autres dans ce programme. La figure 11.5 montre un exemple simple, dans lequel un objet musical dans une *TemporalBox* est calculé à partir de données provenant du calcul d'une autre *TemporalBox*.

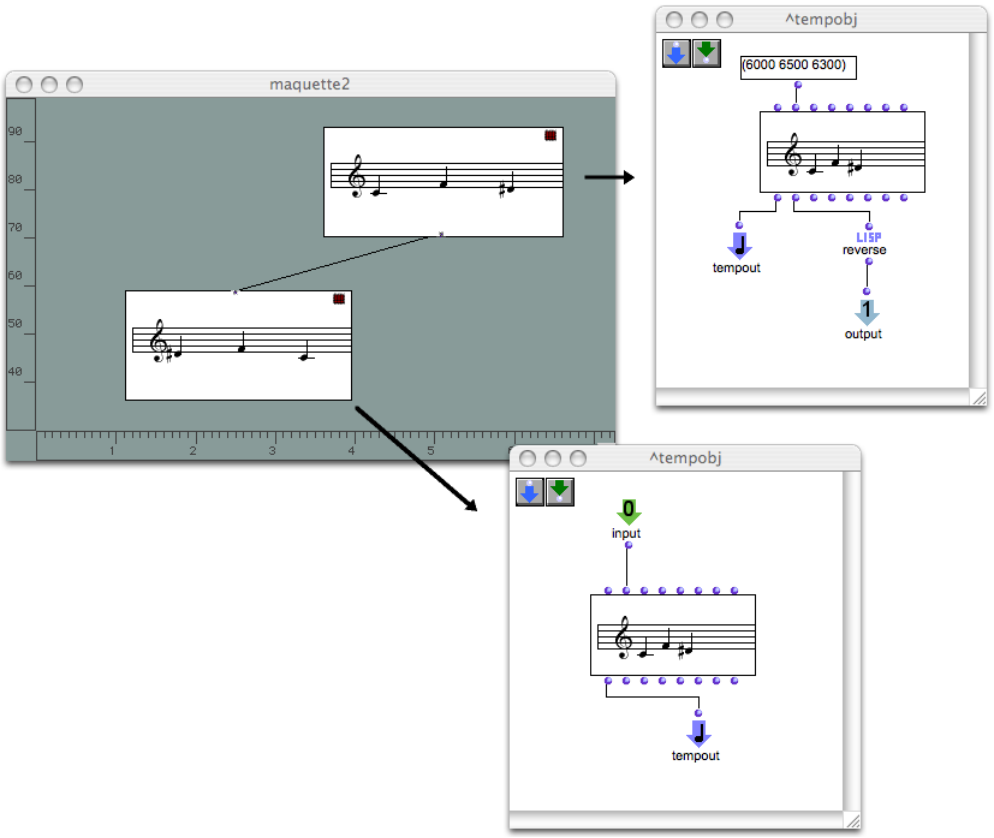


FIGURE 11.5: Relations fonctionnelles dans une maquette.

L'*évaluation* de la maquette déclenchera donc l'évaluation des boîtes terminales du graphe fonctionnel défini dans celle-ci, ces évaluations entraînant récursivement l'évaluation de toutes les autres boîtes connectées. La maquette se comporte alors effectivement comme un programme visuel, dont les boîtes contenues représentent les unités sémantiques de calcul, et dont les connexions définissent l'ordre et le sens du calcul. Un patch n'ayant pas de sortie `TempOut` pourra aussi exister dans une maquette (sous forme de *TemporalBox*) : il aura alors une fonction dans le calcul mais sans être inclus dans le conteneur `maquette-obj`.

Une maquette devient donc elle-même l'équivalent d'un *TemporalPatch*, qui calcule son résultat musical d'une manière particulière, tenant compte des sous-objets qu'elle contient et de leur disposition.

L'insertion d'entrées et de sorties dans les maquettes a permis de compléter ce statut de programme, leur permettant d'intervenir en tant que fonctions, aussi bien dans des patches que dans d'autres maquettes. Ainsi, les possibilités d'abstraction et d'application des maquettes étendent l'organisation et la hiérarchie temporelles au niveau fonctionnel. La figure 11.6 montre une maquette dotée d'entrées et sorties. Si elle est utilisée – appelée – dans un patch ou dans une autre maquette, les valeurs des objets connectés aux entrées de la maquette seront assignées aux boîtes représentant ces entrées dans le corps de la maquette, puis les boîtes représentant ses éventuelles sorties seront successivement évaluées.

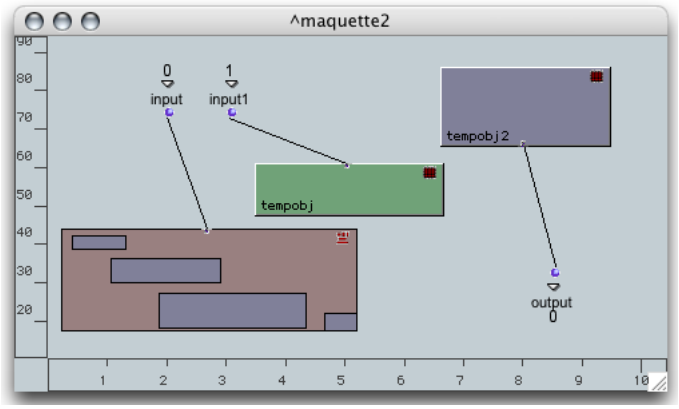


FIGURE 11.6: Entrées et sorties dans l'éditeur de maquette.

Les entrées et sorties permettent ainsi d'établir un flux de calcul entre les différents niveaux hiérarchiques. Dans un patch, voire dans une autre maquette, une maquette peut désormais être vue comme un appel fonctionnel du programme qu'elle définit. Une nouvelle représentation "fonctionnelle" de la boîte `maquette` vient compléter la version "objet" de la figure 11.3 (voir figure 11.7). Par ailleurs, et de la même manière que pour les patches classiques, il est possible d'utiliser des boîtes représentant soit des maquettes "locales", soit des abstractions de maquettes.

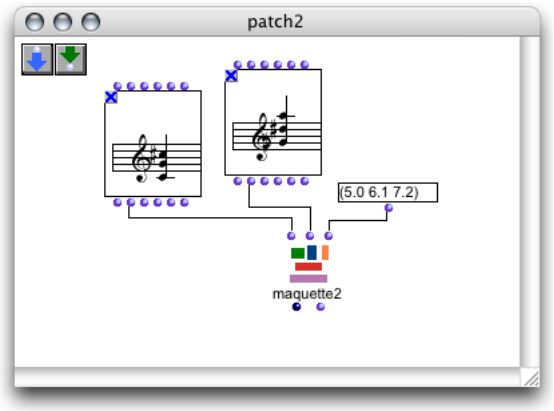


FIGURE 11.7: Version fonctionnelle de la boîte maquette utilisée dans un patch : les objets ne sont pas destinés à créer des *TemporalBoxes*, mais sont les arguments passés à la fonction définie par cette maquette.

Voici donc comment se résume le calcul d'une maquette :

1) Recherche des boîtes terminales
2) Evaluation des boîtes terminales (et des éventuelles sorties de la maquette).
   L'évaluation d'une boîte déclenche :
   - L'évaluation des boîtes auxquelles elle est connectée ;
   - L'évaluation du contenu de la boîte (patch ou maquette) en fonction du résultat éventuel de ces évaluations ;
   - L'enregistrement de la valeur issue de la sortie `TempOut` (pour une boîte faisant référence à un patch) ou de l'objet `maquette-obj` (pour une boîte faisant référence à une maquette) dans la boîte.

   (A l'issue de cette phase toutes les boîtes ont été évaluées.)
3) Collecte des résultats enregistrés dans les boîtes pour construire le conteneur `maquette-obj`.

11.2 Application à la composition

11.2.1 Une représentation temporelle des modèles

La maquette permet donc d'implémenter un certain nombre des différents types de relations temporelles que nous avons vues dans le chapitre précédent (section 10.1.1). Les organisations séquentielle et hiérarchique s'inscrivent naturellement dans la structure même de cet outil, ainsi que les structures temporelles faisant intervenir des relations de causalité (on voit par exemple sur la figure 11.5 que la situation de dissociation de l'ordre du calcul et de l'ordre chronologique décrite dans cette précédente section est reproductible dans une maquette). En ce qui concerne les liens logiques, des relations simples sont spécifiables à l'aide de marqueurs fixes placés sur l'axe temporel, pouvant

être attachés aux commencements ou fins des différents objets.[1]

Il est donc possible de composer sur différents niveaux de l'organisation musicale : organisation temporelle, spatiale, hiérarchique, sur les relations fonctionnelles, sur les programmes, sur les objets.

La maquette peut être considérée comme un objet musical, ou plus exactement un conteneur musical, et comme un programme étendu portant à la fois une sémantique fonctionnelle et temporelle. L'objet de la maquette, qui est à la fois son propre résultat, est concrétisé par le `maquette-obj` issu du calcul de la maquette.

Selon la vision que nous en avons donné dans le chapitre 4, la CAO vise à la construction de modèles compositionnels à l'aide des ressources informatiques, et principalement par la programmation. Considérant la maquette comme une extension de cette représentation des modèles compositionnels dotée d'une dimension temporelle, on peut donc rapprocher cet objet, issu du calcul de la maquette, de l'objet d'un modèle, c'est à dire de la finalité d'une configuration de composants intégrés dans une structure (celle du modèle). Dès lors, révélant leur structure temporelle, la maquette prend un rôle d'interface privilégiée dans le développement de ces modèles.

Les structures hiérarchiques permettent par ailleurs d'envisager de regrouper différents modèles "partiels" dans des formes et modèles de plus haut niveau, à l'intérieur desquels ils seront considérés comme simples composants.[2]

L'objectif que nous suivrons à partir de là sera donc de favoriser la construction de sons comme objets de ces modèles compositionnels (c'est-à-dire comme résultats musicaux des maquettes).

11.2.2 Composer avec les sons

Pour composer avec les sons et la synthèse sonore dans la maquette telle que nous l'avons vue jusqu'à présent, la seule possibilité est la suivante : les *TemporalPatches* peuvent produire des sons (par le biais des outils de programmation présentés dans les chapitres précédents), et ces sons, organisés dans une structure temporelle à l'intérieur de la maquette, sont ensuite intégrés (mixés) dans la construction du `maquette-obj`. Cette option permet donc déjà, dans un premier temps, d'organiser les processus de synthèse selon les différentes possibilités que nous avons détaillées dans la section 11.1.

[1] Un système plus avancé, basé sur la résolution de contraintes temporelles est actuellement développé dans le cadre d'un projet de partition interactive [Allombert *et al.*, 2006].

[2] Nous l'avons vu par exemple qu'un son (éventuellement objet d'un modèle) pouvait faire partie du matériau initial d'un processus compositionnel (i.e. être un composant dans un autre modèle).

Voici un petit exemple, réalisé autour de la pièce *Traiettoria... deviata* de Marco Stroppa.[3] Dans la partition originale des groupes de sons sont identifiables (voir figure 11.8 et [Chadel, 2000]).

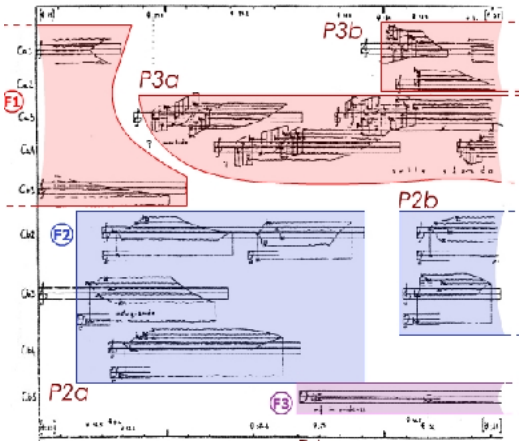


FIGURE 11.8: Extrait de la partition *Traiettoria... deviata* de Marco Stroppa. Les groupes de sons ont été mis en évidence, informant sur leur organisation formelle.[4]

La partie électronique de cette pièce a été recréée par le compositeur dans OPENMUSIC. La structure hiérarchique de la partition (mise en avant par des groupes de sons identifiés sur la figure 11.8) est reproduite dans les patches avec l'utilisation d'abstractions (voir figure 11.9). Des *onsets* s'additionnent à travers les patches inclus les uns dans les autres, selon une organisation hiérarchique, pour finalement s'appliquer en tant que valeur d'*entry delay* (ou *action-time*) sur des matrices de paramètres (voir les matrices OMCHROMA, chapitre 9 section 9.2.2) dans les ramifications finales de cette organisation.

Avec cet exemple, on voit la relative complexité de la création d'une structure temporelle (même assez simple) sans une représentation explicite du temps.

La reconstruction de cette même situation à l'aide de maquettes (figure 11.10) fournit une représentation plus intuitive et plus juste (et favorable aux expérimentations "manuelles") de cette organisation temporelle.

On voit cependant que contrairement à la version réalisée à l'aide des patches (figure 11.9), dans la maquette chacun des sous programmes doit réaliser une synthèse sonore, produire un son, qui sera intégré dans le résultat final. Les évènements, ou objets compositionnels manipulés et organisés dans le temps, sont forcément des sons individuellement constitués. Des éventuelles transitions ne peuvent par exemple pas être envisagées dans ce contexte ; l'intégration des données et paramètres de synthèse dans le processus temporel ne pouvant avoir lieu que localement (à l'intérieur de chaque *TemporalBox*).

---

[3] *Traiettoria* (1982-84), cycle de trois pièces (*Traiettoria... deviata*, *Dialoghi*, *Contrasti*) pour piano et électronique. Enregistré par Wergo : WER 2030-2, 1992.

[4] Illustration fournie par Jérôme Chadel.

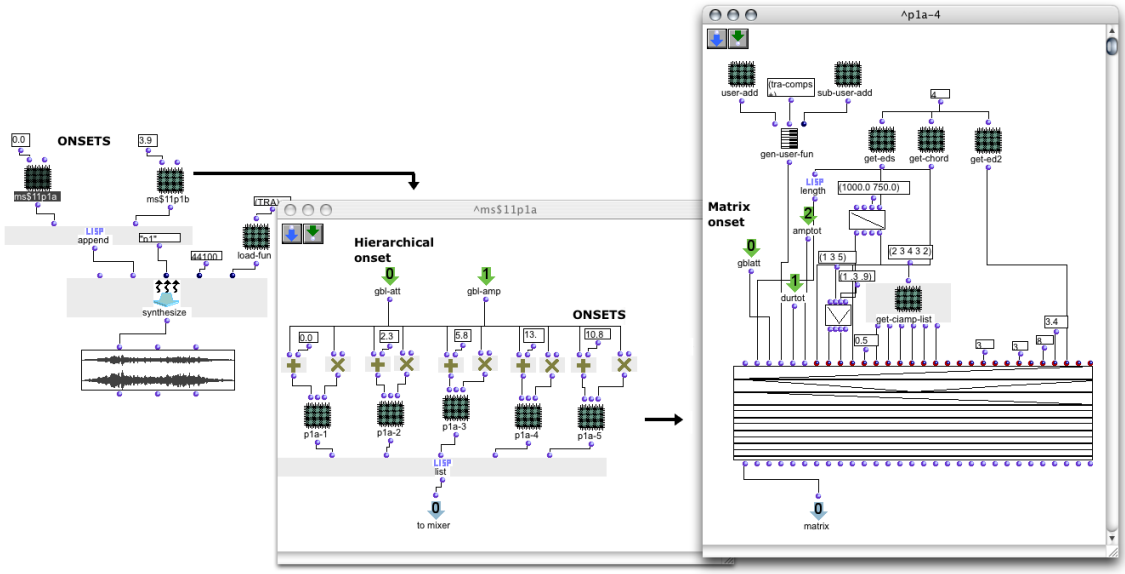


FIGURE 11.9: Reconstitution d'une partie de *Traiettoria... deviata* dans des patches OPEN-MUSIC. La hiérarchie temporelle est reproduite à l'aide des imbrications et abstractions successives, produisant un ensemble de matrices de paramètres synthétisées à la racine de la hiérarchie (fonction `synthesize`).

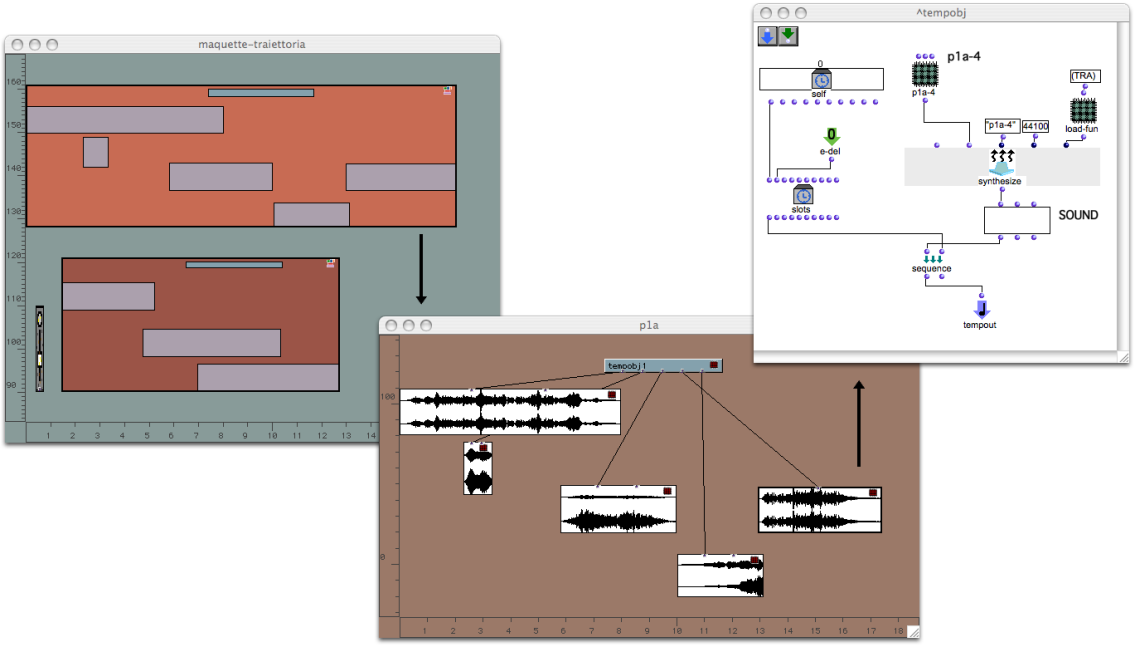


FIGURE 11.10: Reconstitution d'un extrait de *Traiettoria... deviata* à l'aide d'une maquette. Les groupes de sons sont représentés par des maquettes incluses dans une maquette globale. Chaque boîte renferme un programme produisant un élément sonore primitif de la bande électronique.

La même structure temporelle, implémentée successivement sous forme de patches et sous forme d'une maquette, nous a donc montré que le patch permettait une souplesse supplémentaire dans le calcul (par exemple celle de travailler jusqu'au bout sur des matrices de paramètres, synthétisées en dernière instance) et que la maquette offrait quant à elle une meilleure représentation de l'organisation temporelle. Les extensions présentées dans la suite de ce chapitre auront pour objectif principal de rassembler ces deux types de représentation en proposant un niveau de contrôle supplémentaire dans le calcul de la maquette.

11.3 Maquettes de synthèse : contrôle de la synthèse dans le temps

Comme précisé dans [Agon, 1998], le système temporel dans OpenMusic permet une représentation des structures temporelles discrètes. Une réflexion spécifique s'impose donc dans le cas de la synthèse sonore, qui implique comme nous l'avons vu la coexistence de ces structures discrètes avec des structures continues (évolutions de paramètres, etc.) Il est en effet nécessaire de pouvoir définir des objets, ou évènements, sous forme de structures de données abstraites calculées par des programmes, et de considérer leur interprétation dans le domaine sonore dans un niveau distinct du calcul et de l'organisation temporelle discrète opérés dans la maquette.

Durant le calcul de la maquette, nous avons vu que dans un premier temps chaque *TemporalBox* était calculée et produisait un résultat (en principe, un objet musical). A l'issue de cette première phase, la création d'un résultat global pour la maquette consiste à collecter (dans une deuxième phase) les résultats des boîtes afin de constituer un objet musical. Cet objet, en outre, représentera la maquette dans cette phase du calcul si elle est incluse dans une autre maquette (la deuxième phase étant l'équivalent du processus de calcul de la sortie `TempOut` dans un *TemporalPatch*).

C'est donc cette deuxième phase *d'assemblage*, qui ramène la structure temporelle à une séquence linéaire (voir chapitre précédent, section 10.1.2), à laquelle nous nous intéressons à présent.

11.3.1 Contrôle de la phase d'assemblage des composants

La création de ce que nous avons appelé l'objet d'une maquette (`maquette-obj`) dans la phase d'assemblage des composants se fait jusqu'ici en collectant et mixant les sorties des programmes correspondant aux *TemporalBoxes*. L'extension que nous proposons ici consiste à donner à l'utilisateur le contrôle sur cette phase du calcul, lui permettant de générer un objet musical selon le processus de son choix et étant donné le contenu (objets, mais également propriétés et relations graphiques et temporelles) de la maquette.

Cette possibilité sera primordiale dans le développement des processus liés à la synthèse : les objets musicaux disposés sur la maquette, ou résultant des programmes contenus dans cette maquette, pourront dès lors être des objets non nécessairement "finalisés" (comme des notes ou des sons, qui portent un sens musical individuel et déterminé) mais également tous types de paramètres ou de données de contrôle. De tels paramètres sont plus à même de permettre un contrôle fin sur les processus de synthèse, mais doivent être traités en aval afin de révéler ou d'acquérir leur contenu et leur signification complète ; en d'autres termes, être *interprétés*.

Les problèmes des transitions entre les éléments de contrôle discrets (représentés par les boîtes dans la maquette) et les phénomènes continus (modélisables par le calcul) peuvent être plus facilement appréhendés dans ce contexte. Nous en ferons la démonstration par la suite avec des exemples concrets.

Un nouvel attribut de la maquette (que nous appellerons *fonction de synthèse*) permettra de spécifier ce comportement du calcul dans la phase d'assemblage, ou d'interprétation.

Il s'agira à l'intérieur de celle-ci de prendre en compte tous les objets calculés dans la structure temporelle d'une maquette et de les intégrer dans un processus global (synthèse sonore, en principe) afin de renvoyer un objet musical. Un tel processus pourra faire appel aux outils de synthèse de bas niveau pour calculer le son à partir des évènements symboliques organisés temporellement dans la maquette. Il constituera donc le lien entre les boîtes, le contenu de la maquette, et son résultat sonore.

Le calcul de la maquette, que nous avons résumé dans la section 11.1.3, se présentera donc désormais de la manière suivante :

1) Recherche des boîtes terminales.
2) Evaluation des boîtes *(voir section 11.1.3)*.
3) Phase d'*assemblage* :
   - Si une fonction de synthèse est spécifiée : application de cette fonction à la maquette (traitement du contenu de la maquette après évaluation des boîtes).
   - Sinon, les résultats enregistrés dans les boîtes sont collectés pour construire le conteneur musical `maquette-obj` (comportement par défaut).

La fonction de synthèse se présente sous forme d'un patch et peut être assignée à la maquette soit par la programmation, soit en glissant le patch ou la fonction correspondante (i.e. leurs icônes dans l'environnement graphique) dans une zone prévue à cet effet sur l'éditeur de maquette (dans le coin inférieur gauche, voir figure 11.11).

La boîte nommée `MAQUETTE DATA` sur le patch de la figure 11.11 représente l'accès aux données de la maquette dans la fonction de synthèse (respectivement de gauche à droite dans ses sorties, la maquette elle-même, sa durée, les *TemporalBoxes* qu'elle contient, et les objets/résultats correspondants).

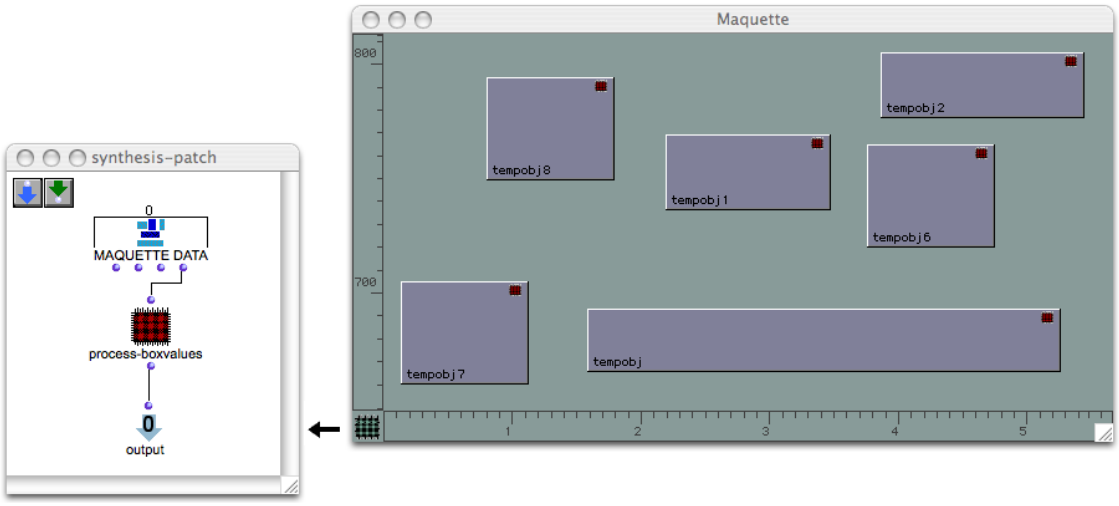


FIGURE 11.11: Une maquette et sa fonction de synthèse. L'icône dans le coin inférieur gauche de la fenêtre de la maquette représente cette fonction. En double-cliquant sur cette icône, on ouvre le patch correspondant et on accède au contrôle de la phase d'assemblage.

Lors de l'évaluation de la maquette, à la suite de la phase de calcul des boîtes constituant le "programme–maquette" (étape 2 dans le calcul d'une maquette), la fonction de synthèse est appelée avec les objets résultant de cette première phase et le résultat de cet appel devient donc la valeur de l'"objet musical–maquette" (étape 3 du calcul). Les évènements, ou objets calculés par les différents composants (*TemporalBoxes*) d'une maquette peuvent ainsi produire n'importe quel type de données dédiées à être interprétées dans la phase d'assemblage. L'accès par la programmation (visuelle) à ce processus permettra une interprétation spécifique et personnalisée de ces données. L'objet de la maquette n'est donc plus nécessairement un objet de type `maquette-obj` ; il pourra s'agir d'un son, ou de tout autre type d'objet.

La possibilité d'utiliser des patches comme fonctions de synthèse permet le prototypage et la réusabilité de ces fonctions dans différents contextes : on pourra utiliser une même fonction de synthèse avec différentes maquettes. Inversement, on pourra utiliser successivement sur une même maquette différentes fonctions de synthèse, correspondant à autant d'interprétations sonores de son contenu.

11.3.2 Exemples

Factorisation des processus de synthèse

Revenons sur la reconstitution de *Traiettoria... deviata* évoquée dans la section 11.2.2 (figures 11.9 et 11.10). Sur la figure 11.10, la maquette permettait un contrôle visuel sur la structure temporelle de la pièce, mais elle n'était pas en faveur de la souplesse et de l'efficacité du calcul étant donné qu'elle obligeait chaque boîte terminale du programme à synthétiser un son, tous les sons étant finalement mixés en dernière instance.

En utilisant le nouveau système de maquette et la fonction de synthèse, il est possible de déléguer le calcul du son à un niveau global et retardé. Dans la troisième et dernière version de la reconstitution de la pièce, illustrée sur la figure 11.12, chaque programme contenu dans les maquettes produit une matrice de paramètres (représentant un "évènement", voir chapitre 9, section 9.2). Ces matrices sont collectées par la fonction de synthèse (visible sur la figure 11.13) qui réalise un processus de synthèse unique pour chacune des maquettes.

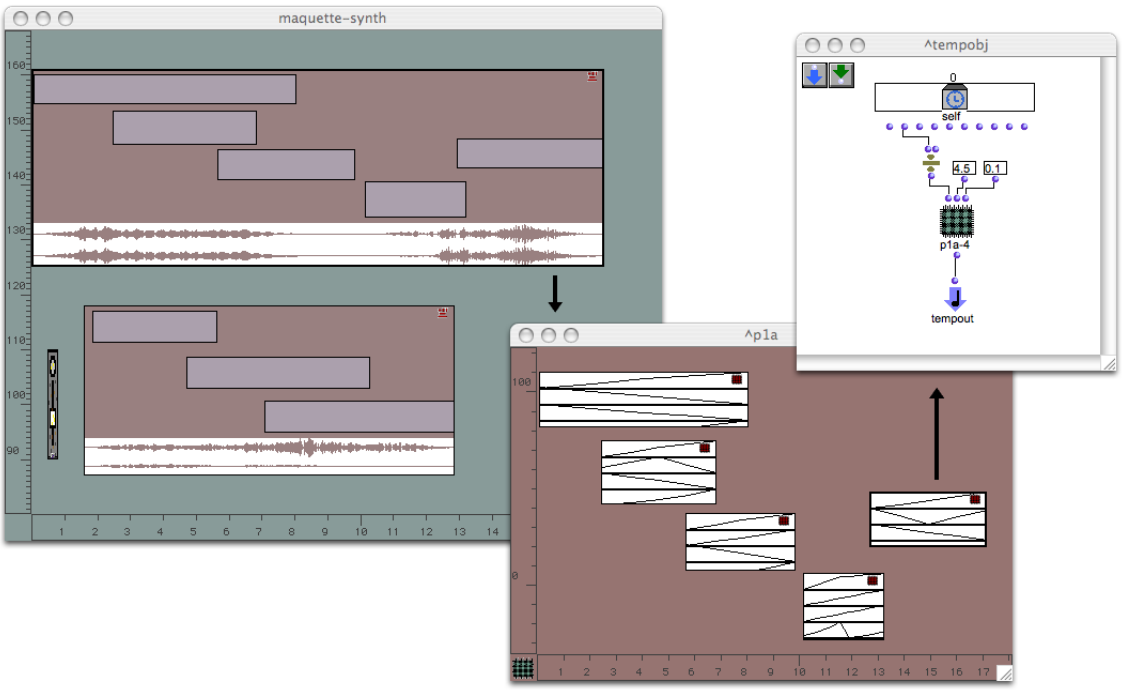


FIGURE 11.12: Reconstitution de l'extrait de *Traiettoria... deviata* dans une maquette utilisant le nouveau système de synthèse.

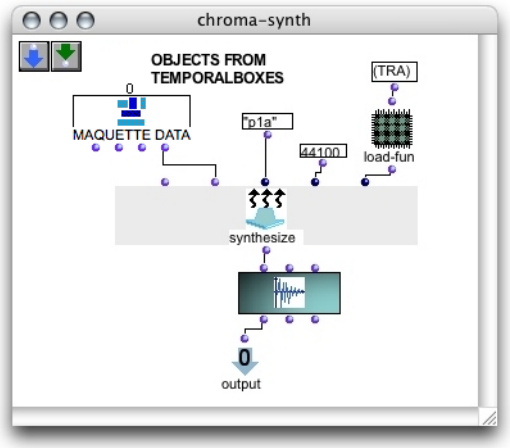


FIGURE 11.13: Fonction de synthèse utilisée dans les maquettes de la figure 11.12.

Cette méthode permet ainsi d'améliorer l'efficacité du calcul (en préservant la structure et la représentation temporelle), mais également d'envisager des interactions plus fortes entre les données en amont du processus de synthèse. Les objets représentés et manipulés dans cette maquette sont proches d'un niveau symbolique, interprété sous forme d'un son dans une phase ultérieure.

Abstraction du processus d'assemblage

Comme nous l'avons dit, la séparation de la fonction de synthèse sous forme d'un patch permet d'appliquer des interprétations diverses à un même ensemble de données. C'est ce que se propose d'illustrer cet exemple, avec la maquette de la figure 11.14.

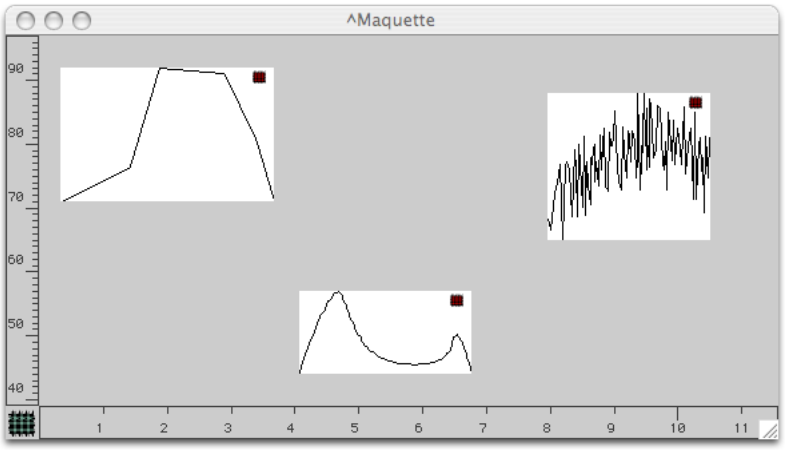


FIGURE 11.14: Une maquette pour notre exemple d'interprétations multiples.

Suivant une démarche proche de celle de l'exemple précédent, nous pouvons interpréter cette maquette à l'aide d'un processus de synthèse, comme celui illustré sur la figure 11.15. Avec ce patch, chaque boite est considérée comme un partiel dans une synthèse additive. Sa position donne les informations temporelles et la fréquence du partiel ; et la BPF qu'elle contient représente une enveloppe dynamique.

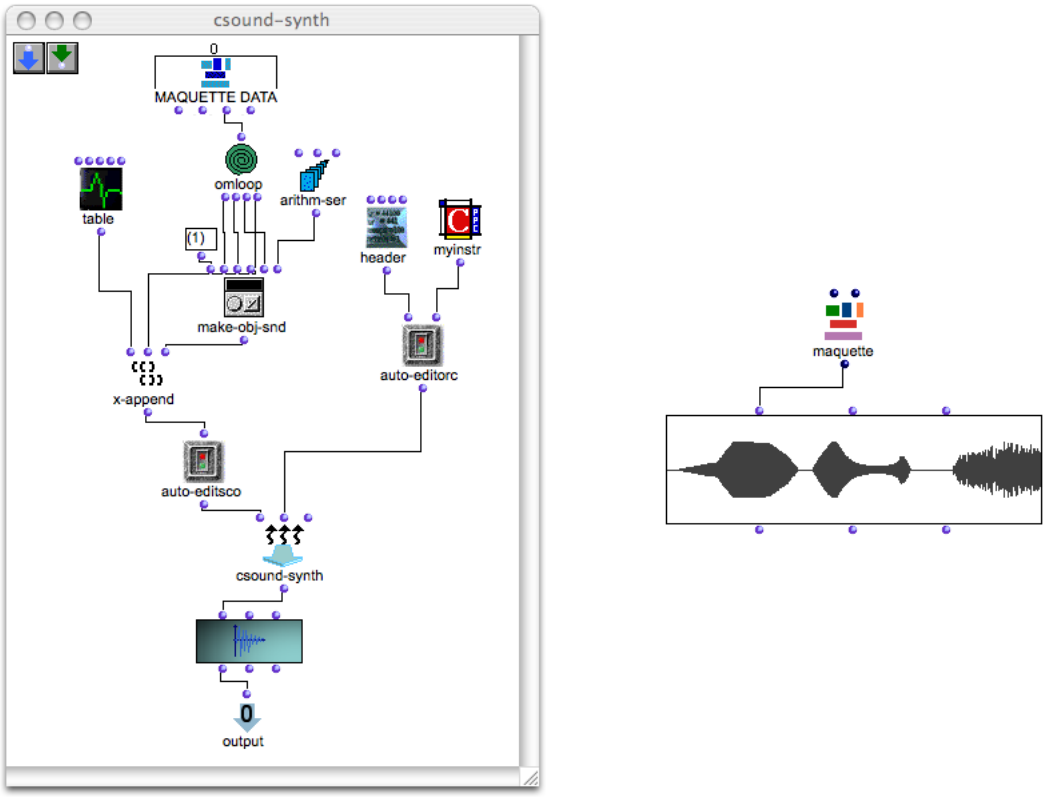


FIGURE 11.15: Interprétation de la maquette de la figure 11.14 avec un programme de synthèse (CSOUND). A droite, la maquette et son résultat sous forme d'objet `sound`.

Le patch de la figure 11.16 interprète quant à lui les mêmes données sous forme d'une séquence de notes, et construit un objet de type `chord-seq`.

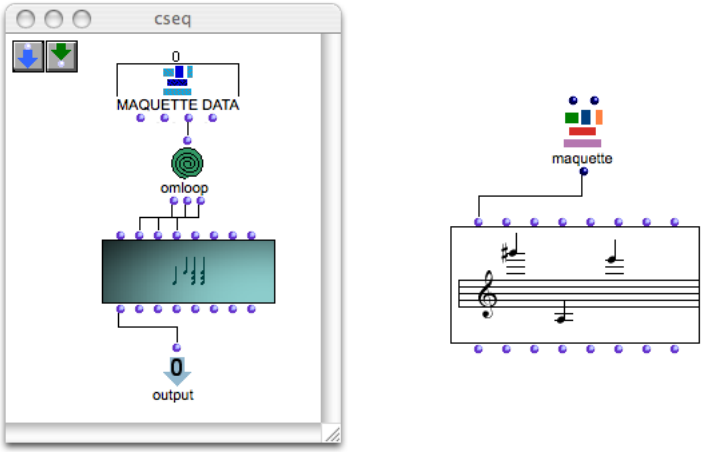


FIGURE 11.16: Interprétation de la maquette de la figure 11.14 en une séquence de notes. A droite, la maquette et son résultat sous forme d'objet `chord-seq`.

Enfin, une troisième interprétation (figure 11.17) utilise des outils de la bibliothèque OMIanniX, que nous avons créée pour établir un lien entre OpenMusic et l'environnement IanniX (voir chapitre 2, section 2.4.4). Les données sont interprétées et formatées sous forme d'une partition (fichier XML) pour cet environnement.

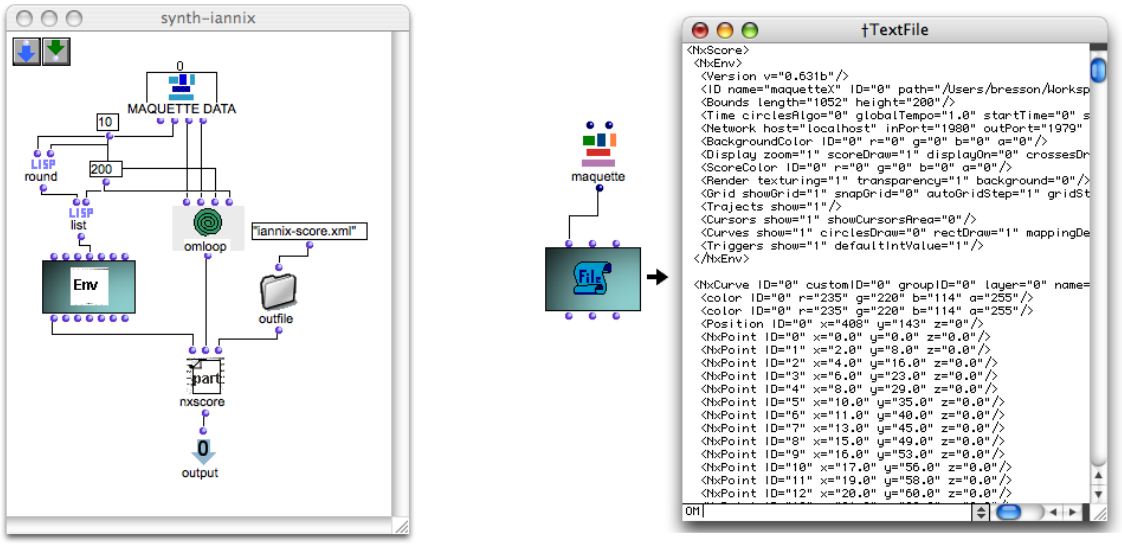


Figure 11.17: Interprétation de la maquette de la figure 11.14 en une partition IanniX. A droite, la maquette et son résultat sous forme de fichier texte.

La figure 11.18 montre cette partition chargée dans IanniX.

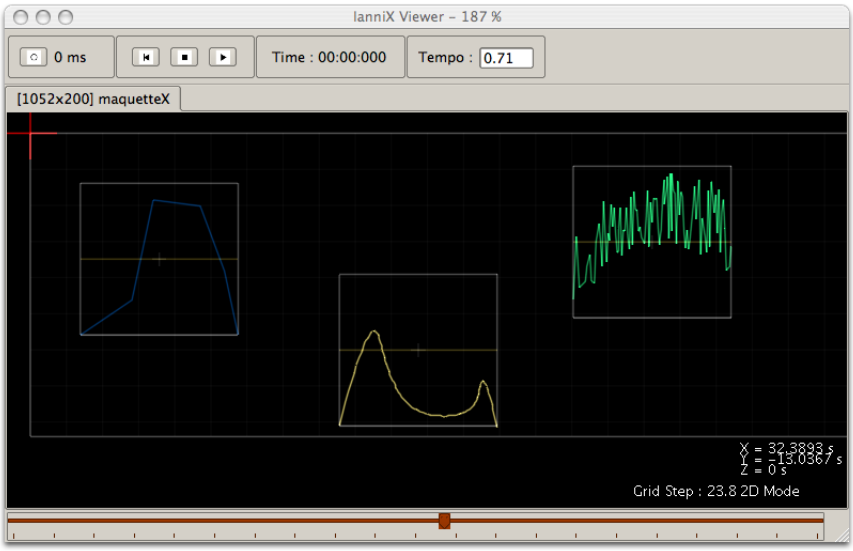


Figure 11.18: Partition créée avec la maquette de la figure 11.14 et la fonction de synthèse de la figure 11.17, chargée dans l'environnement IanniX.

11.4 Gestion et contrôle des phénomènes continus

La séparation des phases du calcul des objets (évènements) et du processus de synthèse dans l'évaluation des maquettes va permettre de considérer et d'établir des relations et comportements relevant du domaine continu tout en maintenant le niveau d'abstraction permis par cet outil. Le contrôle de la phase d'assemblage va nous permettre de systématiser le traitement des évènements dans le but de créer des phénomènes continus à l'intérieur ou entre ces évènements. Ceci peut en effet être réalisé au niveau de la fonction de synthèse de la maquette : il ne s'agit pas de spécifier ces phénomènes pour des évènements individuels, mais bien de décrire un comportement qui s'appliquera à tous, ou au moins à une classe donnée d'évènements. Partant d'objets de contrôle élémentaires, des descriptions sonores complexes peuvent ainsi être générées par le biais de transitions ou de prolongements programmés.

Dans cette partie, nous essaierons d'illustrer ceci par des exemples concrets en utilisant le synthétiseur CHANT. Celui-ci est basé sur la notion de "phrases" continues décrites par l'état d'un système de synthèse à différents instants (voir chapitre 2, figure 2.3). Nous utiliserons une configuration relativement simple du synthétiseur, constituée d'un module de synthèse par Formes d'Ondes Formantiques (FOF). Dans ce cas, le paramétrage consiste en la description de l'état d'une banque de FOF (fréquences, amplitudes, largeur de bandes, etc. des différentes fonctions) à différents instants. Ceci est réalisé par l'intermédiaire d'un fichier SDIF.

11.4.1 Création d'objets continus

Dans cet exemple nous allons reprendre un processus similaire à celui présenté en conclusion du chapitre 7, figure 7.16. Afin de simplifier l'exemple, nous ne considérons, au niveau du contrôle, que les fréquences des différentes FOF du système. La maquette représentée sur la figure 11.19 contient trois *TemporalBoxes*, chacune produisant une `BPF`, correspondant donc aux fréquences centrales de 8 FOF à trois instants donnés.

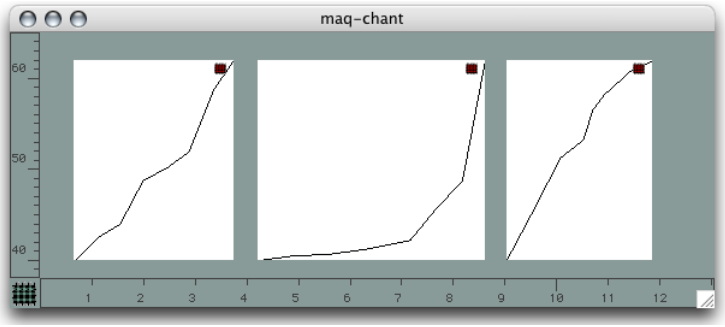


FIGURE 11.19: Une maquette pour le contrôle des fréquences dans une synthèse FOF avec CHANT.

Dans une première version, notre fonction de synthèse utilisera ces valeurs de fréquences et complétera les données de la banque de FOF pour inscrire son évolution dans un fichier SDIF, qui sera donné en paramètre au synthétiseur. Le patch correspondant est visible sur la figure 11.20.

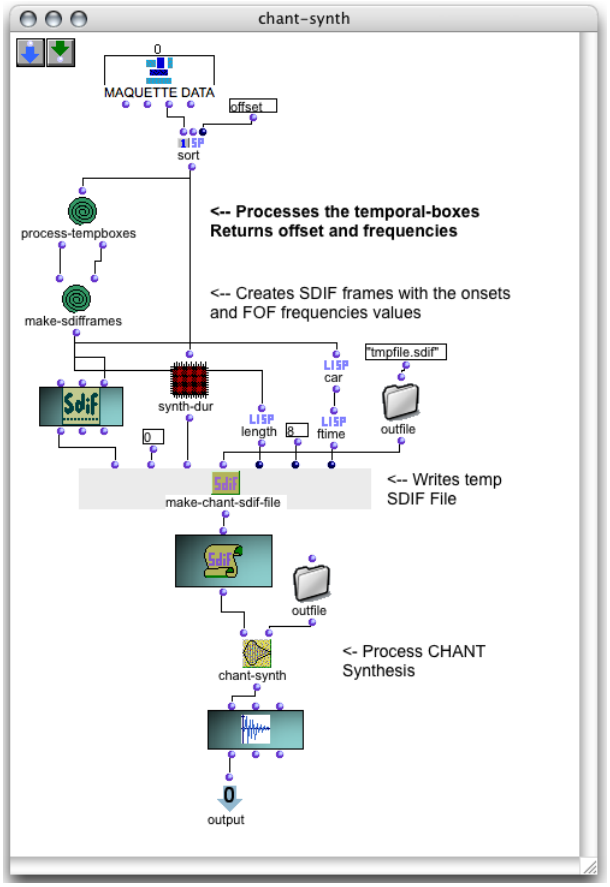


FIGURE 11.20: Une fonction de synthèse pour la maquette de la figure 11.19

La figure 11.21 montre l'itération `process-tempboxes` de la figure 11.20, qui extrait les données de paramétrage à partir d'une liste de *TemporalBoxes*. (L'itération `make-sdifframes`, qui construit les structures SDIF à partir de ces données, n'est pas détaillée ici.)

Avec cette fonction de synthèse, 3 *frames* SDIF `"1FOB"` ("*FOF bank*") sont créés et utilisées pour la synthèse. Ce processus est équivalent à celui qui est présenté dans le chapitre 7 (figure 7.16). La maquette permet cependant déjà une meilleure appréciation et un contrôle étendu de l'organisation temporelle des évènements.

Dans une deuxième version de notre fonction de synthèse, nous remplaçons l'itération `process-tempboxes` par celle représentée sur la figure 11.22, générant une continuation programmée pour les paramètres des FOF présents dans la maquette. Chaque *FOF Bank* est ainsi systématiquement complétée par une période de perturbations aléatoires calculées autour de l'état initial (et s'étendant sur la durée de la *TemporalBox*).

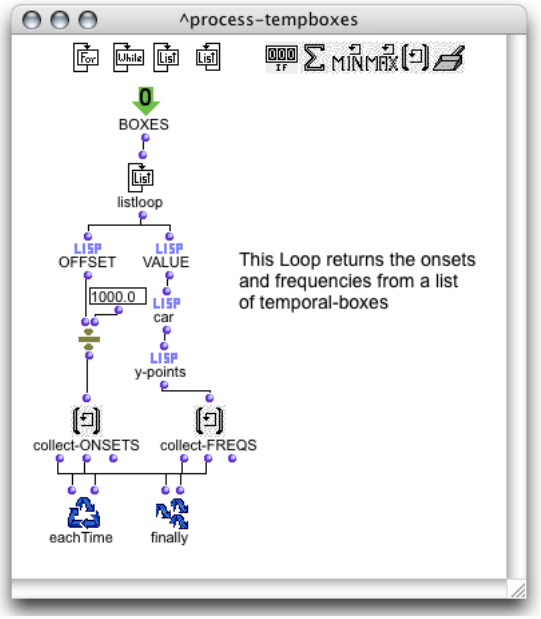


FIGURE 11.21: L'itération `process-tempbox` de la figure 11.20 : création de données à partir des valeurs et attributs des *TemporalBoxes*.

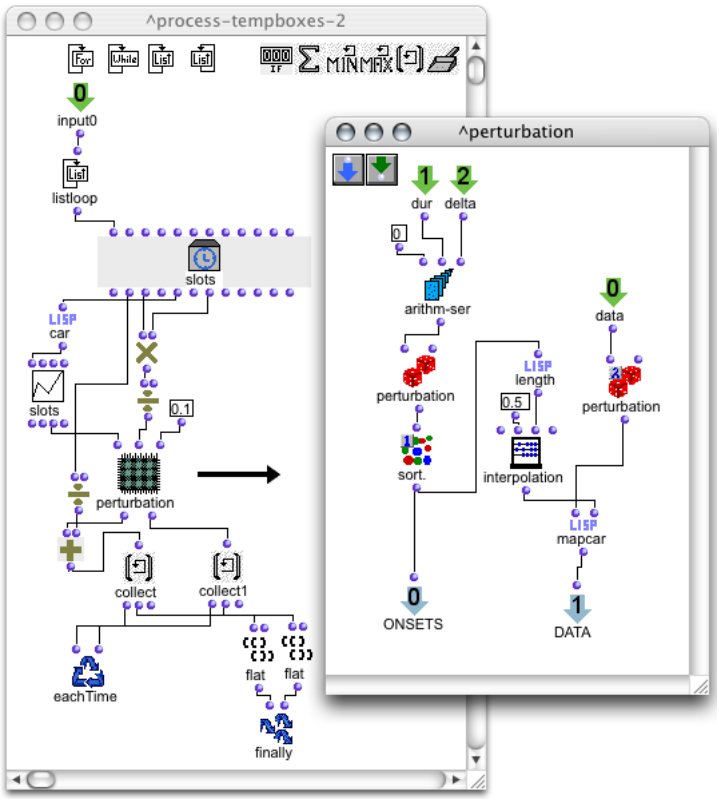


FIGURE 11.22: Deuxième version de l'itération `process-tempbox` de la figure 11.20 : génération de données supplémentaires suivant des variations aléatoires.

Des frames SDIF FOB supplémentaires sont donc générées, créant une évolution plus complexe de l'état des banques de FOF (avec nos 3 boîtes et un taux de contrôle de 0.1s, environ 100 frames sont créées dynamiquement).

La figure 11.23 est une représentation obtenue avec SDIF-Edit des données SDIF générées dans ces deux exemples. Elle donne une idée de la différence, en termes de quantité et de complexité, des flux de données générés dans chacun des cas.

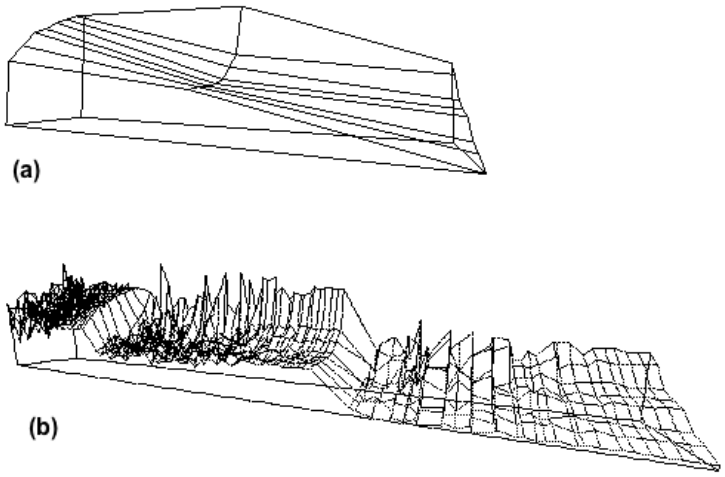


Figure 11.23: Flux de données générés dans les deux versions successives de notre exemple. (Visualisation avec SDIF-Edit.)

Ce processus nous a donc permis de construire une évolution complexe des FOF dans le temps sans considérer cette complexité au moment de créer et d'organiser les données de contrôle dans la maquette (les trois *TemporalBoxes*). Ces *TemporalBox* et leurs valeurs sont la partie externe, visible du processus, que le compositeur choisit de présenter et de soumettre au contrôle. La fonction de synthèse représente dans ce cas un niveau de traitement subsymbolique de ces données de contrôle, qui les intègre et les transforme en une réalité sonore plus complexe.

11.4.2 Articulations entre les évènements

Dans ce deuxième exemple, nous allons essayer, toujours avec le synthétiseur CHANT, de mettre en avant les possibilités d'articulation entre les évènements permises par notre système.

Partant de la maquette de la figure 11.24, nous souhaitons produire une séquence de matrices OMChroma pour une synthèse par FOF avec CHANT. Un autre principe est cette fois choisi : les notes des accords contenus dans les différentes boîtes représenteront les fréquences centrales des FOF, et les positions de ces mêmes boîtes sur l'axe vertical indiqueront la fréquence (en Hz) du signal excitateur de l'ensemble des FOF.

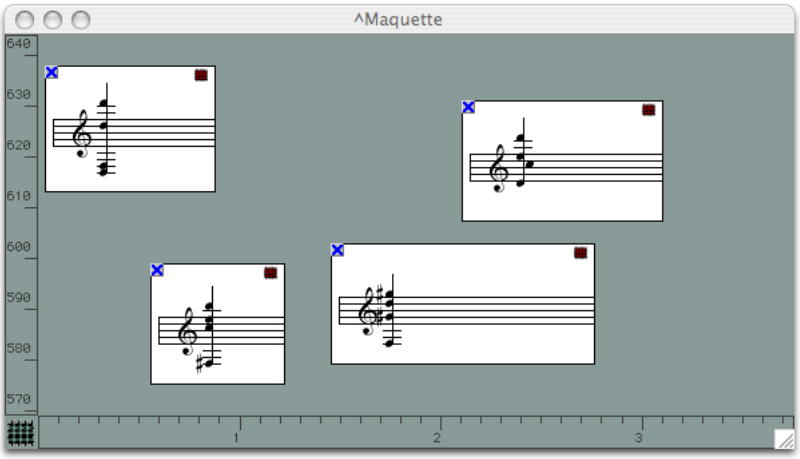


FIGURE 11.24: Une maquette pour le contrôle d'un processus de synthèse par FOF.

Une fonction de synthèse élémentaire pourrait être celle de la figure 11.25. Avec celle-ci, une matrice déterminant un état du banc de FOF est produite au début de chaque *TemporalBox*.

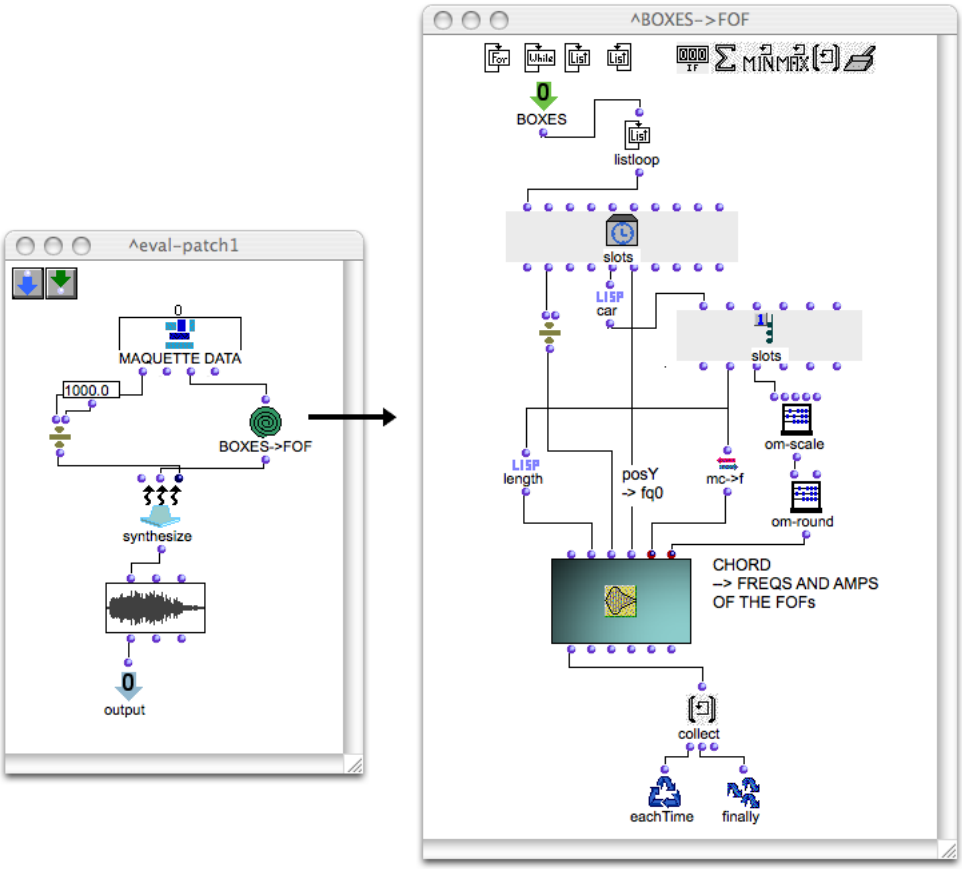


FIGURE 11.25: Une fonction de synthèse pour la maquette de la figure 11.24.

Dans ce cas, des interpolations linéaires sont réalisées entre ces états au moment de la synthèse, ce qui correspond au comportement par défaut du synthétiseur. Nous avons en effet parlé précédemment du caractère “continu” de CHANT, et de son contrôle par des évènements localisés dans le temps, entre lesquels le synthétiseur réalise des interpolations.

Afin d'intégrer la notion d'évènement (localisé dans le temps mais également associé à une durée) suggérée par la représentation de la maquette (boîtes), il faut considérer la représentation d'un évènement (en termes de contrôle, dans la maquette) par deux états au minimum (en terme de paramètres CHANT) : un début et une fin. La figure 11.26 est une illustration de cette idée : les états (valeurs des paramètres) doivent être spécifiées au moins à $t1$ et $t2$ afin de maintenir le contrôle pendant la durée de l'évènement. Avant et après, le synthétiseur effectue une interpolation entre ces états et les états précédent et suivant (c'est-à-dire entre l'état à $t1 - \epsilon$ et celui à $t1$, et entre l'état à $t2$ et celui à $t2 + \epsilon$).

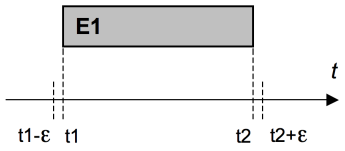


FIGURE 11.26: Représentation d'un évènement.

L'interprétation d'un évènement (passage d'une *TemporalBox* à des données de paramétrage pour CHANT) nécessite donc d'ores et déjà de déterminer :

- comment se manifeste cet évènement sur sa durée, c'est-à-dire ce qui se passe entre $t1$ et $t2$ (le cas le plus simple est de maintenir le même état, mais on peut également imaginer des continuations plus élaborées, comme dans le précédent exemple) ;
- comment se manifeste l'“absence” d'évènement, avant ou après celui-ci, c'est-à-dire les états à $t1 - \epsilon$ et à $t2 + \epsilon$, et la durée de ϵ (on peut aussi imaginer plusieurs solutions : maintient de l'état final, silence, déclin progressif, etc.)

Avec plusieurs évènements, on rencontre encore un autre comportement à déterminer, concernant les articulations et/ou transitions entre les évènements. La figure 11.27 montre deux configurations possibles entre deux évènements.

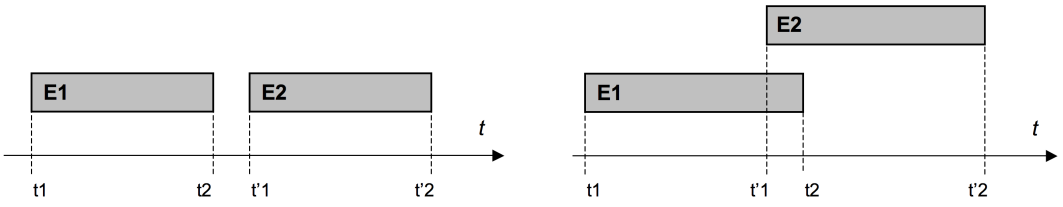


FIGURE 11.27: Interprétation des transitions entre évènements.

Il existe bien sûr d'autres configurations possibles (voir par exemple les 13 relations de Allen entre intervalles temporels [Allen, 1983]), ainsi que des cas supplémentaires quand plus de deux évènements sont mis en jeu. Nous nous en tiendrons à ces deux principales pour ne pas compliquer notre exemple.

Le premier cas de la figure 11.27 est assimilable à l'interprétation de l'absence d'évènement. On parlera d'intervalle entre deux évènements.

Dans le deuxième, la superposition nécessite un choix d'interprétation spécifique. En effet, étant donné le caractère monophonique du synthétiseur, ne pas considérer celle-ci reviendrait à partir des valeurs de E1 à $t1$ pour arriver par interpolations aux valeurs de E2 à $t'1$, puis revenir vers E1 entre $t'1$ et $t2$ pour revenir ensuite vers E2 à $t'2$. Ce comportement peut être un choix d'interprétation, mais n'est certainement pas le plus général ni le plus intuitif. Une solution pourrait être dans ce cas de maintenir les évènements E1 et E2 stables respectivement entre $t1$ et $t'1$ et entre $t2$ et $t'2$, de réaliser une interpolation de E1 vers E2 entre $t'1$ et $t2$.

Des choix d'interprétation doivent donc être faits, en fonction des configurations relatives entre les évènements. C'est précisément le rôle de la fonction de synthèse associée à la maquette de les réaliser.

Pour faciliter la mise en oeuvre d'un tel processus, nous utiliserons une fonction prédéfinie (`transitions`) qui va nous permettre de spécifier les comportements de notre fonction d'interprétation selon les différentes configurations trouvées entre les *TemporalBoxes* successives d'une maquette.

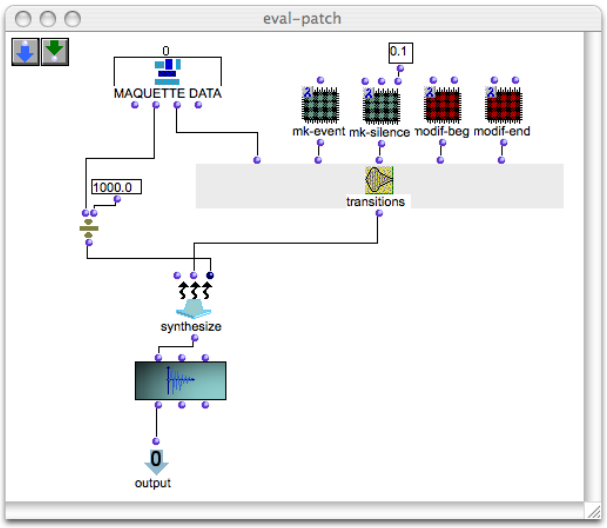


FIGURE 11.28: Une fonction de synthèse pour la maquette de la figure 11.24 : interprétation des transitions entre évènements.

Il s'agit d'une fonction acceptant en arguments une liste de *TemporalBoxes* et un ensemble de fonctions à appliquer selon les différents cas (déterminés par des comparaisons entre les débuts et fins des boîtes successives), et spécifiant :

1) Comment créer un évènement à partir d'une *TemporalBox* ;

2) Que faire quand il y a un "blanc" entre deux *TemporalBoxes* ;

3) Que faire quand le début d'une *TemporalBox* se situe avant la fin de la précédente ;

4) Que faire quand la fin d'une *TemporalBox* se situe après le début de la suivante.

La fonction `transition` renvoie ainsi une liste d'éléments (de type quelconque, à ce niveau) produits selon ces spécifications. Dans notre exemple il s'agira de produire des matrices correspondant à des états des FOF. Nous proposons donc une autre fonction de synthèse pour la maquette de la figure 11.24, utilisant cette fonction (voir figure 11.28).

A titre d'exemple, voici une possibilité d'interprétation des évènements, que nous avons utilisés dans cet exemple (patches en mode *lambda* sur la figure 11.28) :

1) `mkevent` : Un évènement est défini comme deux matrices localisées respectivement aux début et à la fin de la boîte correspondante, et dont les valeurs sont déterminées de la même manière que dans la première interprétation : les accords déterminent les fréquences et amplitudes des FOF, et la position sur l'axe vertical détermine la fréquence du signal d'excitation du banc de FOF. Les deux matrices assurent ainsi que ces valeurs resteront constantes sur la durée de la boîte. Cette fonction est détaillée sur la figure 11.29.

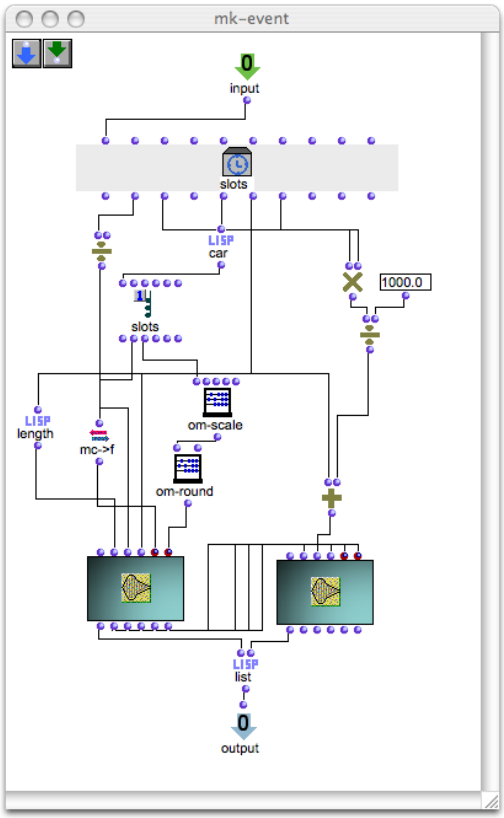


FIGURE 11.29: Création de deux matrices pour l'interprétation d'un évènement (début et fin d'une *TemporalBox*). Les valeurs des paramètres (calculées à partir du contenu et des propriétés de cette *TemporalBox*) sont maintenues entre les matrices de début et de fin.

2) `mksilence` : Un intervalle entre deux évènements (créés par la fonction précédente) est interprété comme un silence, et donc défini comme deux matrices aux amplitudes nulles, localisées respectivement juste après la fin du premier et juste avant le début du second. Les autres paramètres de ces matrices sont les mêmes que ceux de l'évènement auquel elles sont attachées. Cette fonction est détaillée sur la figure 11.30.

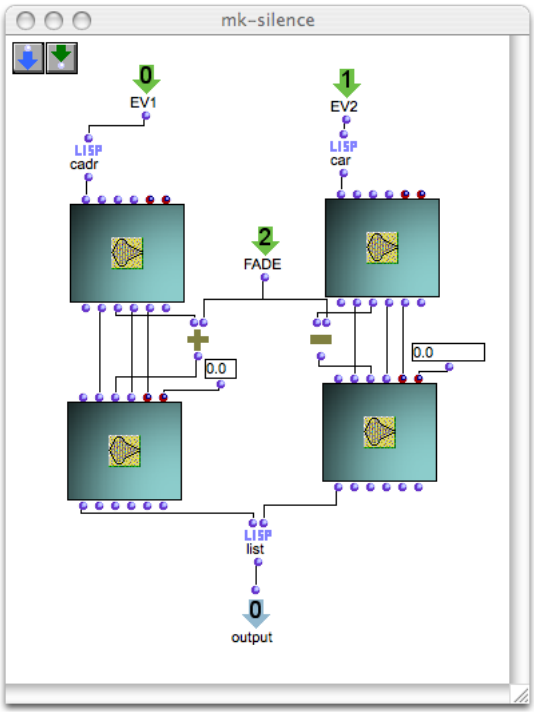


FIGURE 11.30: Création de deux matrices pour l'interprétation d'un intervalle entre deux évènement. Les amplitudes nulles des deux matrices produisent un silence sur la durée de cet intervalle.

3-4) `modifbeg` et `modifend` : ces deux fonctions similaires sont appliquées lors de la superposition d'un évènement avec la fin ou le début d'un autre, respectivement. Selon le processus expliqué précédemment, elles auront pour action de modifier les dates des matrices concernées afin de réduire la durée des évènements et de créer une zone d'interpolation correspondant à l'intervalle de superposition. La fonction `modifbeg` est détaillée sur la figure 11.31.

Encore une fois, ces fonctions et les patches correspondants sont donnés à titre d'exemple, l'intérêt du système étant de permettre la création de processus le plus librement possible, selon différents choix d'interprétations.

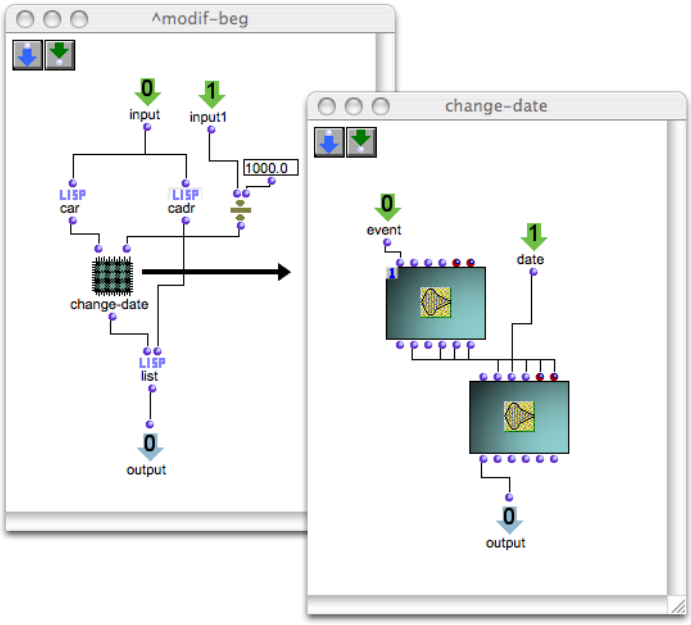


FIGURE 11.31: Comportement lors de la superposition de deux évènements : échange des dates de début et de fin entre les évènements. Les valeurs des paramètres sont donc stables sur la durée "non superposée" des évènements, et interpolées sur la durée de superposition.

11.5 La maquette comme structure et interface du modèle

Nous avons parlé dans la section 11.2.1 du rôle que pourrait jouer la maquette en tant qu'interface privilégiée pour le développement de modèles compositionnels. Dans cette dernière section nous essaierons, encore à travers un exemple, d'illustrer ce rôle structurant de la maquette étendue au domaine de la synthèse.

Cet exemple est inspiré d'expériences réalisées par Karim Haddad. Une maquette est utilisée pour contrôler de manière graphique et interactive un processus de synthèse réalisé avec CSOUND. Celle-ci est illustrée sur la figure 11.32.

Dans cette maquette, les *TemporalBoxes* représentent des occurrences de 6 programmes différents, chacun identifié par la couleur de la boîte. (L'un d'eux est visible sur la figure 11.32.) Nous qualifions ces programmes de *templates*, dans le sens où ils sont réutilisables et réinterprétés dans leurs différentes occurrences, bien que toutes ces occurrences se rapportent à un même programme.

Seuls ces 6 programmes, donc, permettent de contrôler le nombre important d'objets disposés sur la maquette. Chacun implémente d'une façon distincte la création d'un morceau de partition (*score*) pour CSOUND, à l'aide des outils de la bibliothèque OM2CSOUND. On peut voir également sur la figure que le résultat de ce programme dans ces différentes occurrences dépend des propriétés de la *TemporalBox* : la position sur l'axe vertical, notamment, déterminera dans certains cas une position dans l'espace panoramique (balance) ; dans d'autres une valeur de transposition, etc.

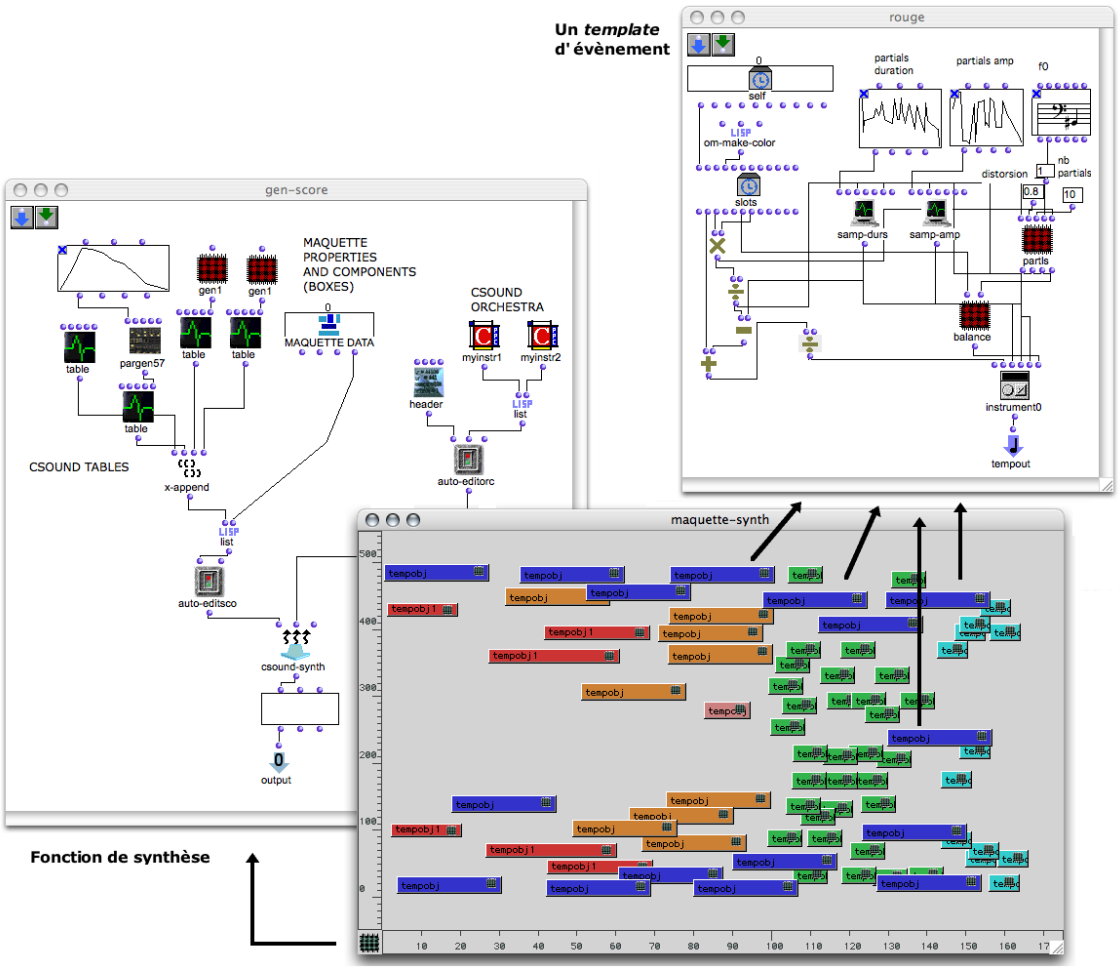


FIGURE 11.32: Une maquette pour le contrôle d'un processus de synthèse. Différents types d'*évènements* (*TemporalBoxes*) sont présents, identifiés par une couleur. Le patch en haut de la figure (intitulé "*rouge*") montre le contenu d'un type d'évènement. La fonction de synthèse de la maquette est visible sur la gauche.

Nous avons donc dans cet exemple un contrôle que l'on pourrait assimiler à l'organisation d'évènements de synthèse (éléments discrets) préalablement établis dans des programmes (*templates*). Ceux-ci, cependant n'ont pas de correspondance sonore individuellement, ils ne sont pas des objets directement "perceptibles." Ils doivent être intégrés et associés à un *orchestre* CSOUND afin de produire un résultat musical (un son). La fonction de synthèse (représentée également sur la figure 11.32) utilise aussi les outils graphiques de OM2CSOUND pour assembler les évènements et compléter le *score*, créer un *orchestre*, puis lancer une synthèse avec ces deux éléments afin de produire le son.

La caractéristique que nous souhaitons souligner avec cet exemple est le rôle structurant de la maquette en tant qu'interface sur ce processus de synthèse. La dimension de la représentation est réduite dans l'interface de la maquette : les boîtes permettent de positionner les évènements et d'en contrôler quelques paramètres seulement. Il s'agit d'une

représentation subjective d'une forme musicale, qui met ainsi en avant une partie plus importante ou musicalement pertinente d'un processus global (et généralement complexe), le soumettant aux manipulations directes par le biais de l'éditeur.

Par là, elle endosse le rôle d'une partition, dans laquelle des objets symboliques (évènements) sont générés par des programmes et organisés dans une structure accessible par une interface graphique relativement simple.

Les évènements, nous l'avons dit plus haut, ne sont pas nécessairement musicaux : il s'agit ici de valeurs numériques intégrées ultérieurement dans la fonction de synthèse de la maquette. Le contenu de chacun, ainsi que leur interprétation dans cette fonction de synthèse, sont des processus complexes relevant pratiquement d'un niveau "subsymbolique". La représentation symbolique, formelle, dans la maquette n'a pas simplifié le processus total ; elle a plutôt mis en avant un côté structurant de ce processus et permet à un moment donné de focaliser les manipulations sur ce niveau.

> "Une partition ne délivre qu'une partie du message que le compositeur considère comme essentielle en regard de ce qu'il veut exprimer." [Manoury, 1990]

Le compositeur est donc en mesure de fixer lui-même le niveau d'abstraction auquel il souhaite travailler pour construire dans la maquette une forme musicale donnée. Il peut choisir ce qui pour lui constitue l'"événement", entité manipulable au niveau symbolique, la manière dont celui-ci doit être interprété, et construit alors une représentation du son dans le temps à partir de ces évènements. A partir des mêmes éléments, du même contexte, il peut expérimenter, créer de nouvelles formes répondant aux mêmes principes et processus subsymboliques sous-jacents.

Une maquette peut donc être vue comme un programme (ou un modèle) produisant un son, dont la forme est contrôlée par une "partition" symbolique (figure 11.33).

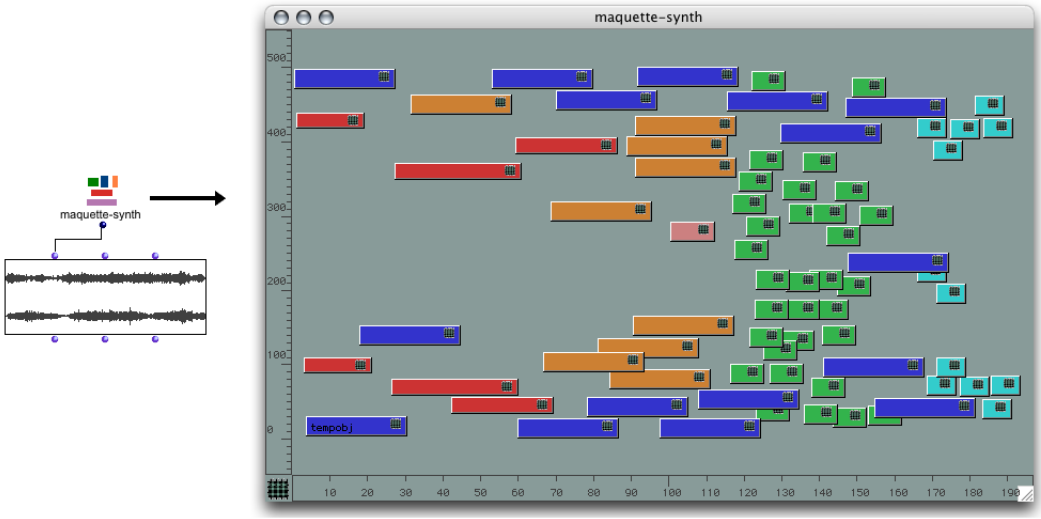


FIGURE 11.33: La maquette de la figure 11.32 vue simultanément comme un programme produisant un son dans programme de plus haut niveau, ou comme une "partition".

Cette “partition” capture ainsi la forme musicale, mais donne également accès au sens de cette forme par le biais des programmes, en même temps qu'elle intègre des registres “extra-partition” tels que celui de l'interprétation (transformation des données compositionnelles en son).

11.6 Conclusion

Le système de maquette que nous venons de voir permet d'intégrer les aspects temporels, symboliques, et calculatoires nécessaires au développement de modèles compositionnels orientés vers le son et la synthèse sonore. Avec une maquette, on est en effet capable de structurer une forme temporelle a partir d'éléments discrets, en distinguant et intégrant calcul des objets, organisation temporelle, et synthèse de la forme, et en assurant par la même la relation et la cohérence entre cette forme et la structure interne des objets. La maquette nous permet donc de déployer les processus dans le temps, mais également d'intégrer le temps dans les processus. Il s'agit d'une structure temporelle de haut niveau dont la sémantique ouverte permet différentes interprétations, notamment vers la synthèse sonore.

Ayant à l'esprit que les mécanismes d'abstraction permettent, à un moment donné d'un processus de composition, de réduire une description sonore à son essence symbolique et musicale, la maquette peut ainsi constituer un positionnement adopté par le musicien à ce moment, correspondant à un niveau d'abstraction qu'il définit pour structurer une forme musicale. Comme le rappelle [Honing, 1993], le choix d'un système de représentation et de ses entités implique un compromis entre des informations qui deviennent explicites au détriment d'autres informations qui sont repoussées en arrière plan.[5] Le choix de primitives (évènements) va alors conditionner la représentation, dans le sens qu'il décidera de l'information qui sera “perdue” en terme de contrôle direct, et de celle qui passera au premier plan de l'organisation. La maquette permettra ainsi à l'utilisateur de réaliser ce compromis. Les aspects déclaratifs et procéduraux de la représentation, évoqués par l'auteur dans ce même article, se reflètent également dans cette interface hybride de structuration “manuelle” des objets (édition et création de structures temporelles par le biais de l'interface, selon une approche de type déclarative), et dans son pendant fonctionnel, par les programmes (approche procédurale).[6]

La distinction entre les processus de haut niveau (organisation d'évènements) et les processus de plus bas niveau (constitution de ces évènements, et interprétation vers le sonore) fait finalement de ces aspects de bas niveau et de traitement du signal des éléments plus

[5] *“In deciding on any particular representational system and its entities, there is a trade-off; certain information will become explicit at the expense of other information being pushed into the background making it possibly hard to recover.”* [Honing, 1993]

[6] Nous retrouvons ici une notion fondamentale dans les principes de la CAO (cf. chapitre 4, section 4.1).

flexibles et variables dans un modèle compositionnel. Les phénomènes et évolutions continus sont traités dans les programmes produisant les évènements, ou lors de l'interprétation de ceux-ci (particulièrement dans le cas de transitions entre évènements). L'accès par la programmation visuelle à ces différents aspects des calculs n'est pas restreint et permet un contrôle intégral sur les structures fines des évènements et sur leur déploiement vers le domaine du signal. La maquette permet ainsi une vision et un contrôle multi-échelles des structures musicales. Le micro et macroscopique de la musique, particulièrement présents avec la synthèse sonore, peuvent y coexister et interagir.

Les principaux avantages que nous avons essayé de mettre en évidence avec ce système sont donc la modularité, avec l'utilisation d'abstractions, la mise en relation de la hiérarchie structurelle avec le calcul, des évènements entre eux dans ce calcul, ainsi que la possibilité d'appréhender les phénomènes continus.

Ce travail sur la représentation des programmes de synthèse dans le temps et le modèle temporel des maquettes est résumé dans [Bresson et Agon, 2006b].

Chapitre 12

Temporalités mixtes et notation musicale

Comme nous l'avons souligné dans le chapitre 3, la notation est un aspect déterminant dans la composition musicale, y compris lorsqu'elle est orientée vers la synthèse sonore et non vers une interprétation instrumentale.

La partition traditionnelle reste aujourd'hui encore un environnement privilégié pour développer cette notation. La représentation de la partie électronique d'une pièce, sa mise en regard avec les parties instrumentales (lorsqu'il s'agit de musique mixte), et surtout leur éventuelle interaction pendant le processus de composition amènent ainsi une réflexion sur la représentation du son et des processus de synthèse dans le cadre même de cette partition.

Dans ce chapitre nous présentons un objet développé dans OpenMusic proposant un nouveau type d'éditeur de partition. Celui-ci permet d'intégrer objets et processus musicaux hétérogènes (notamment liés à la synthèse) dans une représentation favorisant la forme traditionnelle de notation musicale [Agon et Bresson, 2007].

12.1 Partitions et programmes

Nous avons vu tout au long de cette thèse comment la composition musicale contemporaine (et électroacoustique en particulier) pouvait impliquer l'utilisation et la mise en relation d'objets et d'informations hétérogènes, aussi bien dans les structures de données elles-mêmes que dans leur traitement dans des processus de calcul, de visualisation, d'édition, etc. En particulier les différents paradigmes temporels auxquels sont susceptibles de se rattacher ces données (voir par exemple la discussion sur le temps discret et le temps continu, chapitre 10) ont mis en avant la nécessité de savoir intégrer et faire coexister différents systèmes temporels.

La partition, en tant que support d'écriture, est un endroit (l'endroit) où cette intégration doit avoir lieu en priorité. Plus qu'une simple représentation graphique de la musique, celle-ci doit en effet être également considérée en tant que support formel pour la compo-

sition, comme moyen de formuler et communiquer formes et structures musicales (voir la discussion dans le chapitre 3, section 3.1). Cette question concerne donc également le cas de la musique électroacoustique (qui n'est pas nécessairement destinée à être jouée mais doit bien en revanche être écrite, voire ensuite lue et analysée). La notation des éléments électroacoustiques dans une partition reste cependant un problème ouvert, auquel compositeurs et développeurs de systèmes informatiques tentent de remédier par divers moyens plus ou moins personnels et originaux (chapitre 3, section 3.2).

Dans la composition musicale contemporaine, il est par ailleurs fréquent de rencontrer des situations dans lesquelles des objets musicaux "traditionnels" sont utilisés en relation à d'autres types d'objets, destinés à être interprétés par des instrumentistes ou par des programmes de synthèse, et également présents sur la partition. Dans le cas de la musique mixte, par exemple, des signaux sonores, enveloppes, ou autres signes sont souvent insérés dans la partition afin de signaler des intentions ou de permettre aux musiciens de s'orienter lorsqu'ils jouent avec une bande enregistrée ou un ordinateur. Les compositeurs, cependant, doivent généralement effectuer cette édition, ou mise en forme de la partition à l'aide d'outils graphiques divers, après l'écriture de la partition, c'est-à-dire à la fin du processus de composition. Il n'y a donc pas, pendant la phase d'écriture, d'interaction formelle possible entre les objets hétérogènes qui composent cette partition.[1]

En proposant, dans le cadre de la CAO, de fédérer données musicales hétérogènes et processus de création de ces données, [Assayag, 1993] décrit le concept "partition potentielle" comme étant *un lieu où se superposent affichage des résultats, contrôle interactif des degrés de liberté, entrée des paramètres musicaux.* Le concept de notation est alors étendu et lié à celui du langage, permettant d'exprimer et de créer la musique par l'intermédiaire de modèles informatiques (voir chapitre 4). La notation en ce sens permet d'exprimer et d'interpréter ces formes et structures aussi bien dans un contexte instrumental traditionnel que dans un contexte électroacoustique.

C'est ainsi que le développement de OpenMusic, à travers les différentes interfaces et éditeurs proposés par cet environnement, s'est caractérisé par une volonté soutenue d'intégrer partition et programme. En premier lieu dans les éditeurs de *patches* : ces programmes visuels sont des documents fédérant différents types d'objets et de sous-programmes, parmi lesquels les éditeurs de partitions, dont nous avons relevé l'importance dans la création de processus musicaux (voir chapitre 4, section 4.3.5). Ensuite, dans les *maquettes* : celles-ci ont constitué une solution pour une intégration temporelle des objets créés par les programmes sous forme d'une interface se rapprochant de l'idée de partition, tout en constituant elle-même un environnement de programmation (voir chapitre 11, section 11.1).

[1] On notera cependant que si la plupart des éditeurs de partitions sont limités à la stricte notation traditionnelle, certains systèmes permettent à l'utilisateur d'étendre cette notation par des symboles et graphismes personnalisés (voir par exemple [Kuuskankare et Laurson, 2006]).

Cependant, complémentairement au rôle de partition (potentielle) que l'on pourrait ainsi attribuer à la représentation d'un processus musical dans cet environnement, il est important de ne pas négliger le rôle de la partition sous sa forme traditionnelle. Celle-ci reste en effet un support privilégié pour les musiciens et compositeurs, habitués à lire et écrire la musique sous cette représentation. Nous avons en effet pu remarquer qu'une certaine relation était souvent maintenue entre la synthèse sonore et la notation traditionnelle. Dans de nombreux exemples que nous avons donnés, les évènements ou autres types de données destinées au paramétrage de processus de synthèse étaient en effet calculés à partir de représentations musicales traditionnelles (accords, séquences, rythmes, etc.), en particulier pour les données temporelles et/ou liées aux hauteurs.[2]

Si elle présente l'avantage de permettre la mise en relation d'objets musicaux hétérogènes (notation rythmique, sons, etc.) dans un contexte temporel commun (et peut être vue à ce titre comme une forme de partition), l'une des faiblesse de la maquette, cependant, est justement celle de la notation musicale qui y est réduite à un aperçu approximatif du contenu des *TemporalBoxes*. En effet, celles-ci commencent et se terminent dans des points spécifiquement localisés dans le temps, et c'est même là l'une des principales potentialités de la maquette que de permettre d'intégrer précisément les informations temporelles et graphiques dans le calcul. Mais en conséquence, comme on peut le voir sur les figures du chapitre 11 (par exemple figures 11.1, 11.4, 11.5, etc.), la notation musicale à l'intérieur de la maquette est souvent inexacte. Le début des séquences, par exemple, occupe une certaine place (pour la clef, et autres symboles de la portée) qui fait que la première note se retrouve "graphiquement" ultérieure au début réel de la boîte. De même, les séquences exprimées en notation rythmiques ne seront plus cohérentes avec le déroulement linéaire global du temps dans la maquette. A cause de ce type de contraintes (et quand il est question de notation musicale), la maquette reste donc avant tout une interface de programmation étendue au domaine temporel plutôt qu'une réelle partition qui pourrait être lue.[3]

Les outils proposés par OpenMusic pour intégrer partition et programmes dans le cadre de la musique électroacoustique peuvent donc parfois s'avérer insuffisants, notamment si des données hétérogènes sont mises en jeu simultanément et/ou si une intervention humaine (lecture, interprétation, performance) doit s'effectuer en parallèle à l'exécution des parties synthétiques.

---

[2]La notation microtonale disponible dans OpenMusic permet alors de travailler dans des contextes micro-intervalliques.

[3]Dans la pratique, les maquettes utilisées pour la composition doivent généralement être converties en partitions (polyphonies, ou objet `poly` dans OpenMusic), à la suite du processus de composition, afin de se préter à une interprétation instrumentale.

12.2 Le *sheet* : une nouvelle partition dans OpenMusic

Le *sheet* est un objet créé dans OpenMusic dans le but de traiter les questions évoquées dans précédemment. Il peut être vu comme un nouveau type de document persistant dans l'environnement (comme un patch, une maquette) ou comme un objet musical. L'éditeur associé à cet objet, qui fait sa particularité, est un éditeur de partition ouvert, suivant les objectifs d'intégration partition–programme, ainsi que de représentation commune et cohérente des différents types de structures et d'objets musicaux.[4]

Le *sheet* est composé d'un nombre variable de voix, contenant chacune un ou plusieurs objets musicaux positionnés dans le temps. A ce titre, il est lui même un objet (conteneur) musical, au croisement de la maquette et de la polyphonie.

En principe, tout type d'objet musical peut être inséré dans un *sheet*, à partir du moment où il a une durée : accords, séquences, fichiers MIDI ou audio, enveloppes, etc. La figure 12.1 montre un éditeur de *sheet* contenant un certain nombre d'objets différents.

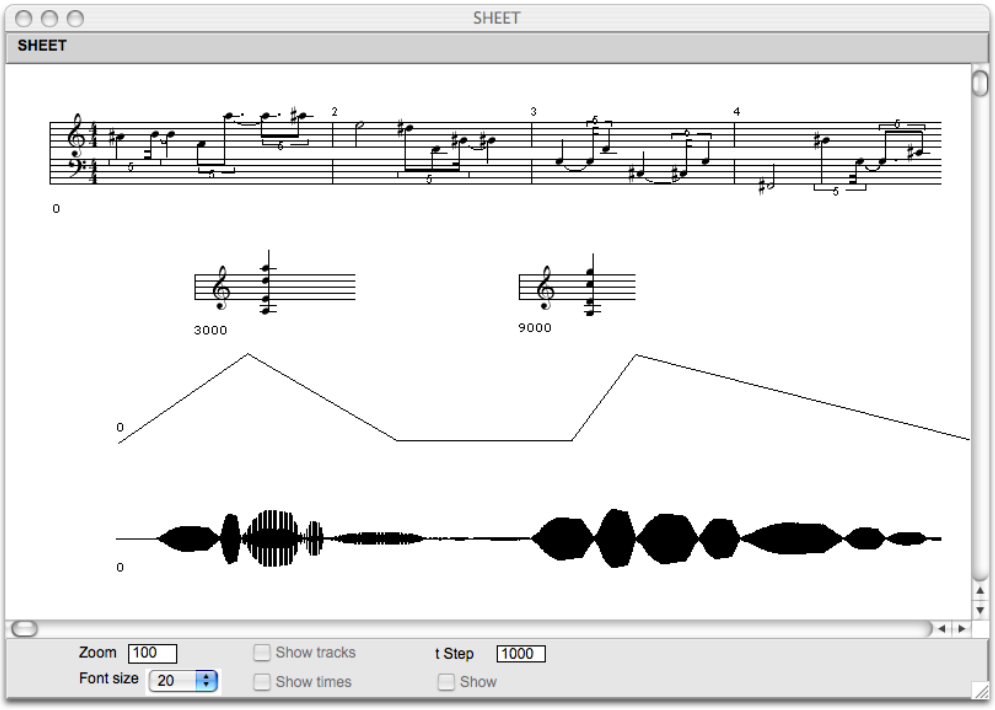


FIGURE 12.1: Editeur de l'objet *sheet* : chaque voix contient un objet ou une séquence d'objets musicaux. Les temps d'entrée (*onsets*) des différents objets sont affichés en bas à gauche de chacun.

[4]Le nom *sheet* se réfère à la feuille de papier blanche.

Les fonctionnalités élémentaires de cet éditeur sont donc la création et la suppression de voix, et l'ajout et la suppression d'objets à/de ces voix. L'affichage des différentes voix peut également être édité (taille, position, espacement, affichage d'une trame de fond, etc.) afin d'obtenir une disposition graphique souhaitée pour la partition.

Chacun des objets est par ailleurs éditable individuellement avec son propre éditeur (éditeur de partition, de BPF, de fichier audio, etc.)

12.3 Systèmes temporels et représentation graphique

L'un des problèmes principaux traités dans l'éditeur de *sheet* est la représentation graphique conjointe et cohérente des différents types d'objets.

12.3.1 Non-linéarité de la représentation temporelle

La notation musicale n'est pas linéaire vis-à-vis de la représentation du temps : il n'y a pas de relation linéaire entre le temps (les durées) et l'espace occupé par les symboles sur l'axe temporel de la partition. La figure 12.2 nous donne un exemple. La grille temporelle régulière (de 1 seconde) superposée à la partition met en évidence que l'espace occupé par une seconde varie constamment au cours du déroulement de cette partition.[5]

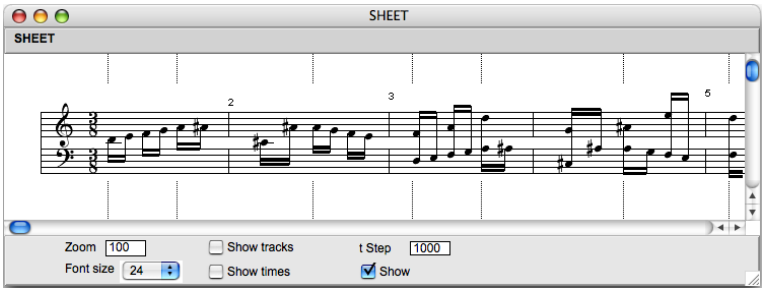


FIGURE 12.2: Un objet *voice* (notation musicale traditionnelle) dans l'éditeur de *sheet*, avec une grille temporelle régulière.

Les cas élémentaires de non-linéarité sont par exemple ceux des symboles de la partition qui n'occupent pas de temps du tout (par exemple les clefs, informations métriques, barres de mesures, altérations, etc.) Leur taille ne peut évidemment pas être adaptée à leur durée (ils auraient une dimension nulle) ; ainsi l'axe du temps est distordu afin de leur laisser une place dans la représentation.

Cette non-linéarité peut devenir problématique (en termes de représentation) lorsqu'il s'agit de représenter plusieurs objets simultanément. Sur la figure 12.3, par exemple, un canon est représenté, c'est-à-dire deux voix identiques dont l'une est décalée dans le temps

[5]L'affichage d'une grille temporelle régulière, permise par l'éditeur de *sheet*, permettra de mettre en avant ce type de distorsions dans les exemples donnés dans ce chapitre.

par rapport à l'autre (dans ce cas, de 1 seconde). Leurs contenus sont donc identiques, mais leurs représentations graphiques sont sensiblement différentes : elles ont été adaptées afin de respecter l'alignement graphique des évènements musicaux représentés par les symboles. Ainsi, par exemple, la deuxième note de la deuxième voix occupe presque trois fois plus d'espace (dans la dimension horizontale) que la même note de la première voix : de la place supplémentaire a été nécessaire à cet endroit afin de permettre l'affichage de la barre de mesure (et des espaces qui l'entourent) dans la première voix.

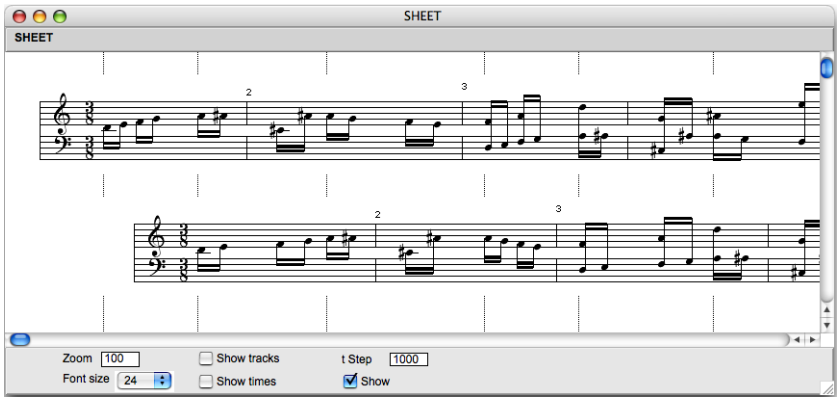


FIGURE 12.3: Deux objets *voice* en canon dans l'éditeur de *sheet* : phénomènes de distorsion dans la représentation du temps.

Malgré ces phénomènes de distorsion du temps dans la représentation graphique, une règle fondamentale doit être respectée : les évènements simultanés doivent se situer exactement à la même position sur l'axe temporel. Il faut donc satisfaire à la fois la contrainte d'affichage des symboles, et cette nécessité d'alignement visuel.

12.3.2 Alignement temps/espace

La cohérence de la représentation temporelle est assurée dans l'éditeur de *sheet* par un algorithme qui calcule une fonction reliant le temps et l'espace dans la partition, à partir de tous les objets présents sur celle-ci.

Les différents repères spatio-temporels contenus dans chaque objet sont considérés afin de déterminer des rapports entre le temps (durée entre deux repères) et l'espace (dimension des représentations graphiques correspondantes). En regroupant tous ces rapports, on construit une fonction discrète globale $x = f[t]$, dans laquelle chaque élément de la partition trouve une position dépendante des possibles contraintes propres à l'objet musical dont il fait partie (comme dans le cas de la figure 12.2) mais aussi des autres objets de la partition (cas de la figure 12.3).

En appliquant cette fonction sur l'ensemble de la représentation graphique des objets, on modifie (par étirement ou compression) la représentation de chaque objet individuellement (y compris ceux qui pourraient être représentés de façon linéaire, comme les courbes,

les fichiers audio, etc.) en fonction de tous les autres. Tous sont donc affectés par les distorsions entraînées par les symboles de notation "non-linéaires".[6]

Une méthode est donc définie pour chaque type d'objet permettant d'y collecter (lorsque c'est possible) ces différents repères utilisés pour construire la fonction liant temps et espace. Pour les objets musicaux traditionnels, par exemple, ces repères correspondent aux débuts et fins des notes, silences et autres symboles. D'autres objets, correspondant à des représentations temporelles distinctes, pourront cependant s'insérer dans le même système. La détermination des repères dans un objet apportera des contraintes non-linéaires dans la représentation le cas échéant, ou permettra simplement de créer des points d'intérêt dans la fonction temps/espace, dont la représentation sera conforme aux contraintes des autres objets.

12.3.3 Systèmes temporels

Les différents types d'objets pouvant se trouver dans le *sheet* peuvent correspondre à différents paradigmes temporels (ou systèmes temporels). Nous avons identifié trois principaux types de systèmes temporels parmi les objets existant dans OpenMusic : le temps pulsé (divisions régulières de temps ou pulsations, comme dans le cas de la notation musicale traditionnelle) ; le temps proportionnel (évènements musicaux spécifiés en millisecondes, par exemple) ; et le temps (pseudo-) continu (par exemple les signaux, enveloppes et autres fonctions).

Les objets relevant du temps pulsé ont été en partie traités dans les paragraphes précédents. Ils correspondent principalement à la notation musicale traditionnelle et se trouvent être les principaux objets responsables des phénomènes de distorsion temps/espace. Les deux autres systèmes sont abordés successivement dans les paragraphes suivants.

12.3.4 Notation rythmique et notation proportionnelle

La figure 12.4 montre un autre exemple dans lequel deux voix correspondent à présent à deux différents systèmes temporels : l'une est en notation musicale traditionnelle, comme dans les exemples précédents, et l'autre est un fichier MIDI dont les dates et durées des évènements sont spécifiées en millisecondes (temps proportionnel). Ce fichier MIDI est une transcription exacte de la voie supérieure. Suivant notre principe d'alignement, les dates et durées des évènements MIDI sont "délinéarisés" (i.e. étirés, ou compressés) en adéquation avec les contraintes graphiques de la notation musicale de la première voix, afin de se conformer aux positions des notes et symboles de celle-ci. Deux notes MIDI de durées égales (dans cet exemple, elles ont en réalité quasiment toutes la même durée) n'auront donc plus nécessairement la même taille sur la représentation graphique.

[6] Comme c'est le cas dans la plupart des éditeurs musicaux, le curseur de lecture de l'éditeur de *sheet* devra aussi varier sa vitesse de défilement selon cette fonction de distorsion temps/espace.

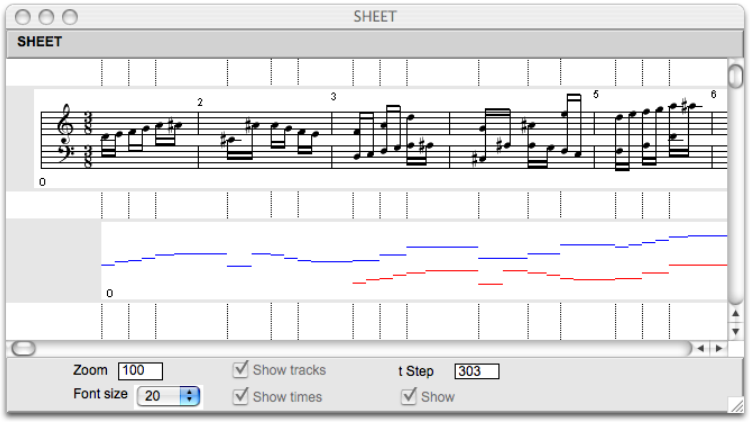


FIGURE 12.4: Intégration de systèmes temporels hétérogènes dans l'éditeur de *sheet* : un objet *voice* et un fichier MIDI.

Dans OPENMUSIC, le `chord-seq` est un autre objet relevant du temps proportionnel, mais il est représenté à l'aide d'une notation musicale (sur une portée). Bien qu'il ressemble à une partition, chaque accord ou note y est localisé par un temps spécifié en millisecondes, sans considération pour aucune autre unité métrique. La représentation graphique y est donc proportionnelle au temps. Cependant, au contraire du fichier MIDI de l'exemple précédent, le `chord-seq` apporte des propriétés non-linéaires dans la représentation graphique, du fait de sa propre représentation sur une portée (clef, têtes de notes, altérations, etc.)

Ici encore, l'éditeur de *sheet* nous permettra donc de représenter conjointement des objets musicaux exprimés en notation rythmique ou sous forme de *chord-seq*, comme le montre la figure 12.5. Dans cet exemple, on voit que c'est le *chord-seq* (pourtant en notation proportionnelle) qui est cette fois responsable des distorsions du rapport temps/espace.

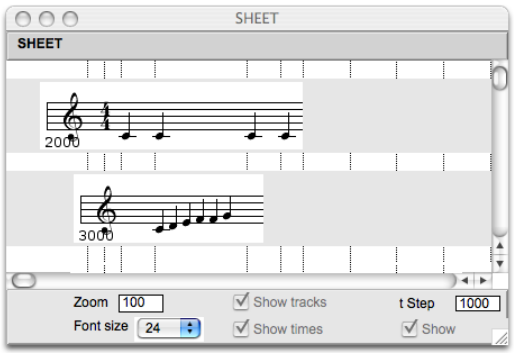


FIGURE 12.5: Intégration des notations rythmique et proportionnelle : chaque note de la voix supérieure (rythmique) dure 1 seconde. La deuxième seconde et étirée graphiquement afin de permettre aux notes du *chord-seq* de tenir entre la deuxième et la troisième note de cette voix.

12.3.5 Objets en temps continu

Le troisième système temporel que nous avons identifié est celui que nous avons appelé “continu”, dans le sens où les objets qu'il comprend ne sont pas composés d'évènements bien localisés (comme des notes). Cette catégorie inclut donc les courbes, fonctions, ainsi que les fichiers audio. Dans ces objets, seuls deux principaux repères temporels sont connus en principe : leur début et leur fin. Ceux-ci permettent déjà de les positionner et de déduire leur dimension dans la représentation graphique. Ces objets n'apportent généralement pas de contraintes quant à la linéarité de la représentation temporelle ; cependant, des moments additionnels peuvent être définis ou déduits dans ces objets afin d'affiner cette représentation. Ces moments mis en avant à l'intérieur des objets seront considérés comme repères dans l'algorithme de segmentation et de calcul des rapports temps/espace, de manière à ce que les distorsions soient appliquées indépendamment sur les différents segments définis.

Dans le cas des BPF ou enveloppes, on utilisera comme repères les points de contrôle qui définissent la courbe. Pour les fichiers audio, on utilisera les *markers* qui peuvent être insérés manuellement ou algorithmiquement sur l'axe temporel (voir chapitre 6, section 6.1.2).

La figure 12.6 montre un fichier audio dans lequel des *markers* ont été placés à chaque seconde, représenté dans l'éditeur de *sheet* avec une voix musicale en notation traditionnelle. Le début du son est donc synchronisé avec le début réel de la première voix (i.e. avec la première note), et les segments réguliers dans ce son sont successivement étirés ou compressés afin de respecter l'alignement avec les symboles de la partition.

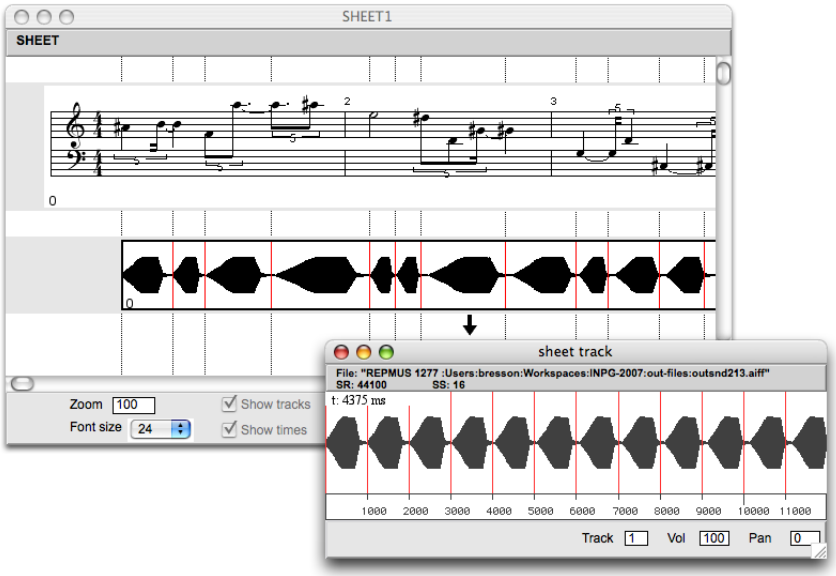


FIGURE 12.6: Intégration de systèmes temporels hétérogènes dans l'éditeur de *sheet* : une voix en notation traditionnelle et un fichier son. Les marqueurs dans le fichier son permettent d'aligner individuellement les segments avec la notation musicale.

12.4 Partition programmable

12.4.1 Aspects fonctionnels dans l'éditeur de *sheet*

Le deuxième principal aspect de l'éditeur de *sheet* concerne les possibilités de programmation qu'il propose. En mettant l'éditeur dans un mode particulier (appelé mode "patch"), l'utilisateur peut aussi décrire et mettre en avant à l'intérieur de la partition une autre partie de la sémantique musicale, concernant les aspects fonctionnels et génératifs, c'est à dire la manière dont sont créés et/ou liés les objets. Dans le mode "patch", l'éditeur devient semblable à un éditeur de patch habituel, dans lequel chaque objet est représenté avec une entrée et une sortie pouvant être connectées à des boîtes (représentant des programmes ou des appels fonctionnels), ainsi qu'aux autres objets de la partition. L'éditeur devient ainsi un cadre pour le développement de programmes dans lesquels les objets de la partition peuvent être reliés, créés ou traités algorithmiquement et fonctionnellement. Sur la figure 12.7, par exemple, deux voix sont connectées par un programme qui définit l'une comme étant l'inversion de l'autre.

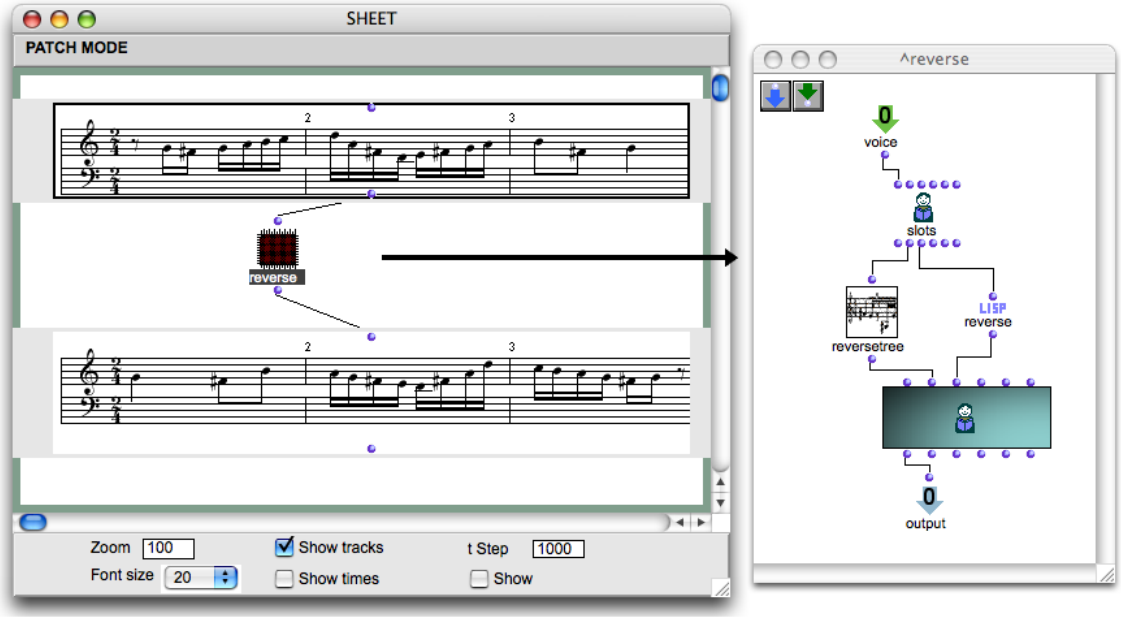


FIGURE 12.7: Relations fonctionnelles entre les composants d'un *sheet*, définies par la programmation à l'intérieur de l'éditeur (mode "patch").

La possibilité d'évaluer les objets ou la partition entière permet d'en construire ou mettre à jour le contenu selon ce type de dépendances : quelles que soient les modifications effectuées ensuite sur la première voix (dans l'exemple de la figure 12.7), l'évaluation de l'éditeur recalculera la seconde afin qu'elle soit son inversion.

Si l'on reprend à présent les exemples des figures 12.1 ou 12.6, ou pourrait supposer, ou vouloir, que la partie "électronique" de ces partitions (la voix contenant le fichier audio)

soit liée aux autres voix (supposées instrumentales). La figure 12.8 montre l'éditeur de la figure 12.1 en mode "patch" : le fichier audio est calculé par un programme, inclus dans cet éditeur, qui utilise des données provenant des notes de la première voix et de l'enveloppe de la troisième.

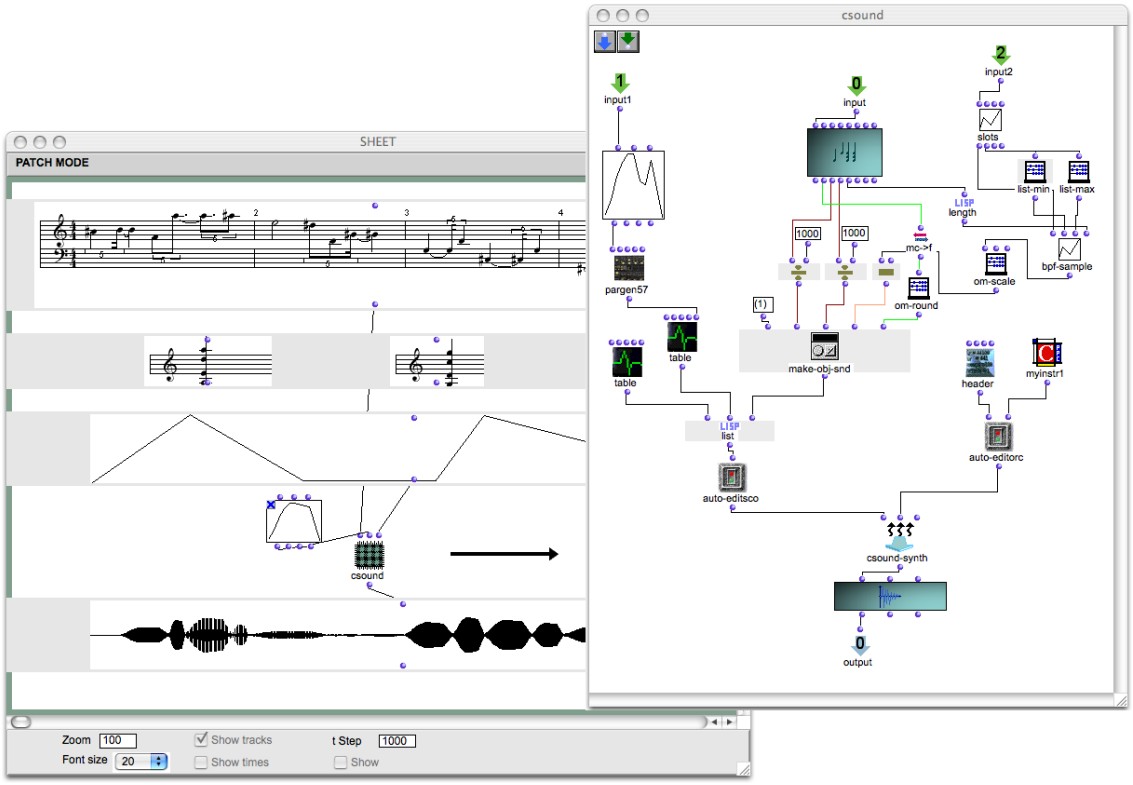


FIGURE 12.8: Processus de synthèse sonore dans un éditeur de *sheet* (mode "patch"). La boîte *csound* est une abstraction (détaillée sur la droite de la figure) qui effectue une synthèse sonore à partir d'une séquence musicale, donnée par la partition, et de deux enveloppes dynamiques (l'une est globale, venant également de la partition, et l'autre est locale et appliquée à chaque note individuellement).

La représentation peut ainsi être enrichie par différents éléments entrant en jeu dans les processus musicaux et/ou de synthèse sonore, ou simplement permettant une meilleure lecture et compréhension de la partition. En revenant au mode "score", les éléments de programmation sont cachés pour un affichage simplifié et plus proche d'une partition classique (figure 12.1).

12.4.2 Intégration du *sheet* dans l'environnement de CAO

Patches, maquettes et *sheets* constituent des supports complémentaires dans l'environnement OPENMUSIC, et peuvent naturellement interagir dans les processus et modèles qui y sont développés.

Nous avons vu en effet comment un patch pouvait être utilisé dans un éditeur de *sheet* pour définir des relations fonctionnelles entre les objets. Symétriquement, le *sheet* (tout comme la maquette), étant considéré comme un objet musical, peut être utilisé (créé, inspecté, etc.) dans un patch, comme élément d'un programme. Il existe en effet une boîte *factory* (voir chapitre 4, section 4.3.5) pour la classe `sheet`, à laquelle est associé l'éditeur que nous avons décrit dans ce chapitre. La figure 12.9 montre un *sheet* construit dans un patch à partir d'une liste d'objets musicaux créés dans l'environnement.

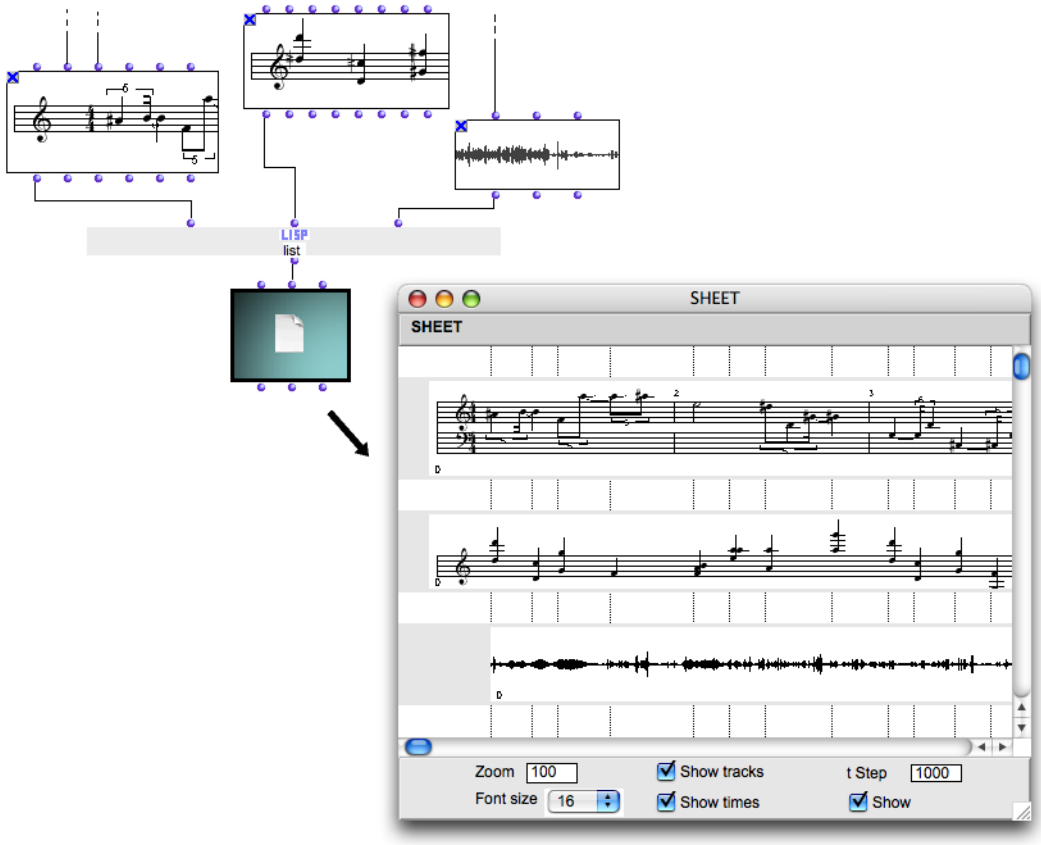


FIGURE 12.9: Construction d'un objet *sheet* dans un patch.

Si la maquette présente des possibilités de calcul plus avancées que le *sheet* (hiérarchie, abstraction fonctionnelle, etc.) elle peut également intervenir en complémentarité avec ce support. Sa double condition de programme/objet lui permet d'être incluse dans le *sheet* en tant que processus produisant un résultat musical et/ou directement en tant qu'objet musical, dans l'une des voix de la partition (voir figure 12.10). Dans ce cas, la maquette correspondrait à ce que nous avons appelé un système de temps proportionnel (avec des évènements positionnés sur l'axe temporel selon une spécification en millisecondes). Les boîtes qu'elle contient sont alors utilisées comme repères spatio-temporels dans l'algorithme d'alignement graphique, et sont ainsi étirées ou compressées dans la représentation afin de rester alignées avec les parties "non linéaires."

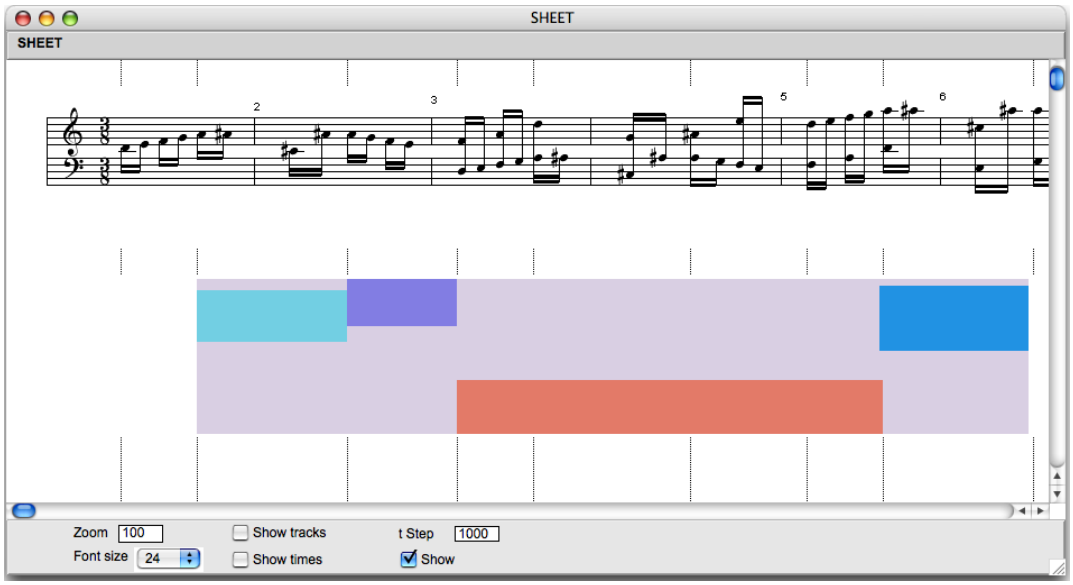


FIGURE 12.10: Une maquette dans l'éditeur de *sheet*.

En intégrant une maquette comme celles que nous avons vues dans le chapitre précédent dans un *sheet*, il est donc possible d'insérer un objet sonore "compositionnel", représenté sous forme d'une structure temporelle et fonctionnelle par cette maquette, dans le cadre d'une partition traditionnelle. Il est possible aussi d'établir des relations fonctionnelles entre partition instrumentale et partition de synthèse, par l'intermédiaire des entrées et sorties de la maquette (voir section 11.1.3).

12.5 Conclusion

Avec le *sheet*, nous avons essayé de réfléchir à une partition étendue aux différents types d'objets et de données musicales, principalement en termes de représentation (graphique) de programmabilité, qui sont deux caractéristiques primordiales dans un systèmes de CAO tel que nous l'avons défini.

Le *sheet* peut donc être utilisé selon diverses modalités en relation avec les autres documents existant dans l'environnement OPENMUSIC, particulièrement lorsqu'un intérêt particulier est porté sur la notation. Concernant la notation instrumentale uniquement, il offre des possibilités nouvelles, comme celle de mélanger notation rythmique et proportionnelle tout en maintenant la cohérence de la représentation.

Les problématiques liées notamment à la musique mixte, avec la représentation conjointe de parties instrumentales et électroniques, peuvent être traitées dans ce contexte, intégrant synthèse sonore et représentation du son dans le cadre d'une partition. Cette intégration est aussi renforcée par les aspects programmables, qui permettent de mettre en relation ces différentes parties.

Pour conclure sur cet éditeur, voici un exemple réalisé à partir de la partition de *Mikrophonie I*, de Karlheinz Stockhausen. Il s'agit d'une pièce écrite en 1964 pour tam-tam et six instrumentistes (dont deux tenant des microphones et deux jouant sur des filtres par l'intermédiaire de potentiomètres). Celle-ci à été écrite d'une manière assez originale, et peut être interprétée, selon les passages, plus ou moins librement. La figure 12.11 montre une page de cette partition reconstituée dans l'éditeur de *sheet*. On y trouve différents types d'objets graphiques, enveloppes, rythmes, et de nombreux détails qui font la richesse de la notation de cette partition.

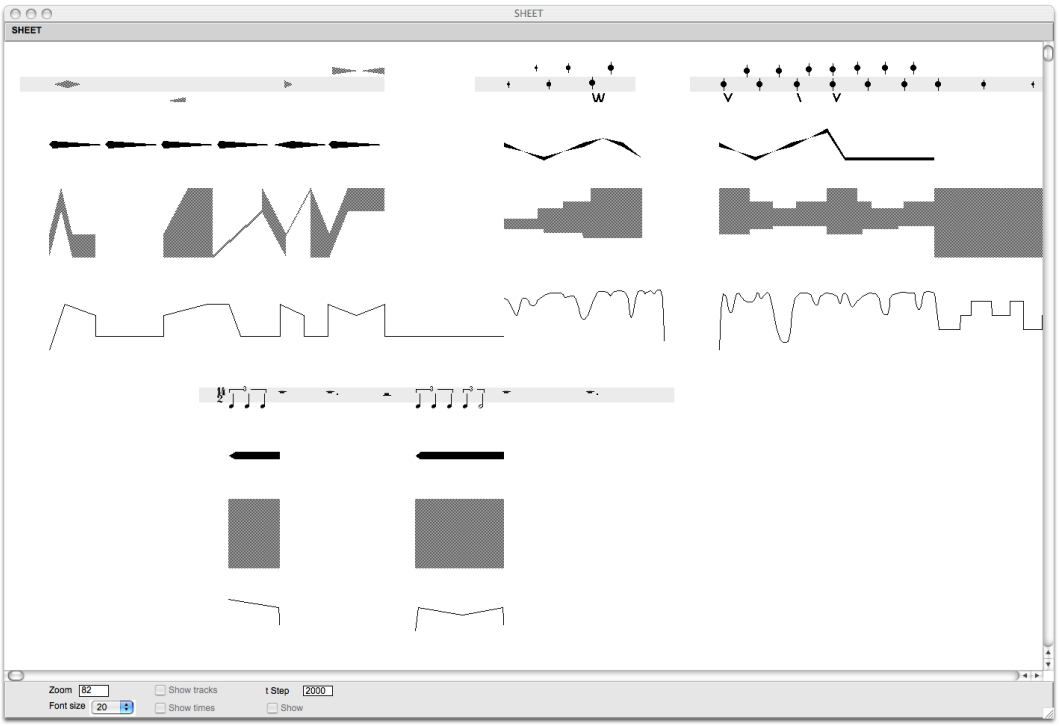


FIGURE 12.11: Une partie de *Mikrophonie I*, de K. Stockhausen, reconstituée dans l'éditeur de *sheet*.

Nous envisageons de poursuivre cette expérience en implémentant un processus de synthèse sonore, probablement en utilisant les modèles physiques avec MODALYS et son interface dans OPENMUSIC (voir chapitre 5, section 5.2.3). Celui-ci permettrait (selon le principe illustré sur la figure 12.8) de simuler approximativement la performance correspondante à partir des données contenues dans cette partition.[7]

[7]D'un point de vue musicologique, une telle implémentation pourrait offrir un modèle computationnel de la pièce et de ses processus d'interprétation. Etant donné son caractère ouvert et le fait que sa réalisation puisse varier selon les interprétations des instrumentistes, un tel modèle permettrait au musicologue d'analyser différentes des ses variantes, sans (trop) altérer l'intention du compositeur.

Conclusion

Le modèle compositionnel

Les techniques actuelles de synthèse sonore constituent de puissants outils proposant des représentations du son susceptibles d'être créées et manipulées par le calcul. Pour autant, leur pertinence et utilité musicales ne sont pas forcément assurées par ce seul potentiel, comme on a pu le croire aux premières heures de la synthèse sonore puis du numérique. Comme problème général de cette thèse, nous nous sommes posé celui de l'écriture du son, soit la question de savoir s'il était possible, et comment, d'utiliser ces techniques dans une démarche de composition musicale. Il s'agit d'une question large et assez ambitieuse, à laquelle il serait difficile d'apporter une réponse en deux mots.

Nous nous sommes placés dans le cadre et le paradigme de la composition assistée par ordinateur (CAO) pour traiter ce problème, à partir de méthodes éprouvées principalement dans le domaine de la musique instrumentale visant à construire des modèles compositionnels à l'aide de langages et d'outils de programmation adaptés. Nous avons par là souhaité dépasser la simple coexistence des technologies de synthèse sonore et des environnements de CAO pour aller vers un environnement informatique capable d'intégrer les concepts, données et processus relevant de ces deux domaines dans une approche unique et cohérente.

Après avoir parcouru les techniques et outils existants liés à la synthèse sonore, et essayé de replacer ceux-ci dans le contexte de la composition musicale (partie I de la thèse), le domaine de la CAO a été présenté comme proposant une vision plus à même de susciter une approche réellement musicale du problème (partie II). La CAO donnait alors une importance particulière à la notion de modèle, qui devenait le centre de la problématique compositionnelle. En passant du traditionnel modèle de synthèse, sur lequel sont basés la plupart des travaux existants, au modèle compositionnel (entendu comme modèle lié à une aspect donné d'une démarche compositionnelle), les problématiques musicales sont données *de facto* comme fondatrices et unificatrices de l'activité de modélisation. Le son (comme finalité du modèle) est alors abordé d'un point de vue plus abstrait, comme une forme musicale soumise par l'intermédiaire du modèle aux problématiques de composition et à la pensée musicale.

C'est donc à partir de cette notion de modèle compositionnel que nous avons traité notre problème, en travaillant sur des outils de représentation et de programmation lui permettant d'étendre sa portée au niveau de la synthèse et du signal sonore.

Les outils

La CAO met en avant le potentiel des langages de programmation pour le développement de modèles compositionnels, favorisant l'interaction du compositeur avec ces modèles. En particulier, les langages de programmation visuels constituent une représentation adéquate permettant à la fois de concrétiser et de mettre en lumière sens et structure des modèles. Les paradigmes traditionnels de la CAO ne sont cependant pas directement adaptés à une

utilisation dans le cadre de la synthèse sonore : différents modes de calculs, différentes échelles, différentes natures des objets et structures musicales mises en jeu, ou différentes temporalités, sont autant de caractéristiques qui engagent la nécessité d'outils et de supports spécifiques.

Sur de telles observations, nous avons construit une architecture logicielle (OMSounds) intégrée dans l'environnement de composition OpenMusic, et permettant d'orienter les processus musicaux créés dans cet environnement vers le son et la synthèse sonore. Dans un premier temps, les objets et données complexes impliqués dans la synthèse sonore sont abordés de manière symbolique par le calcul et la programmation visuelle, soit au niveau de leur génération par des programmes (éventuellement liée à des structures et données sonores existantes) soit par la description de leur propre comportement dans les traitements musicaux auxquels ils sont soumis. Leur connexion avec le domaine sonore (dans le sens de l'analyse ou de la synthèse) est par ailleurs assurée au sein même et dans la continuité des calculs. Cet environnement intégré, dont les principaux éléments ont été présentés dans la partie III de la thèse, permet ainsi de développer des démarches et modèles compositionnels complets assurant la connexion des processus de bas niveau liés au signal avec des processus ou structures musicales de plus haut niveau (y compris instrumentaux).

En abordant la question de l'écriture, nous en sommes naturellement venus à plusieurs reprises à nous interroger sur l'idée d'une partition électroacoustique. Celle-ci n'est pas uniquement entendue dans le sens d'une simple représentation (graphique), mais surtout en tant que support d'écriture : un endroit où l'on organise des objets, des symboles pour développer des idées musicales. Ceci étant, nous avons également vu que s'en tenir à une idée de partition trop stricte, constituant la limite entre les domaines de l'écriture et de la production sonore, ne nous permettrait que partiellement de résoudre notre problème. En ce sens le paradigme de la CAO nous a permis d'apporter des éléments de réponse supplémentaires, avec une conception de la partition constituée non seulement de résultats musicaux (séquences musicales), mais aussi de représentations des processus conduisant à ces objets. Les sons ne sont donc pas seulement représentés par un ensemble de données destinées à être transférées à un programme de synthèse, mais plutôt par une structure, un programme capable de produire ces données. Le schéma classique du contrôle de la synthèse est détourné dans le sens où la modélisation (compositionnelle) ne se base plus sur les processus de synthèse de bas niveau implicitement ou explicitement visés, mais utilise ceux-ci comme simples éléments des modèles.

Dans cet environnement qui se veut donc résolument orienté vers les problématiques compositionnelles, nous n'avons pas cherché à masquer la complexité ou les aspects techniques de la création sonore, puisque ce sont justement ces aspects qu'il s'agit d'exploiter et de maîtriser. La modularité permise par les mécanismes d'abstraction et de structuration des processus, les relations entre ces structures, le temps, et le calcul (dans la maquette, en particulier) sont autant de potentialités héritées de la CAO pouvant être appliquées au domaine de la synthèse, et permettant d'aborder celui-ci dans un contexte symbolique.

Ces premières idées et réalisations ont ensuite été rapportées au déploiement et à l'organisation des processus et structures de synthèse dans le temps, compte tenu des différents repères et interactions pouvant être mis en jeu dans ce contexte (partie IV). La *maquette de synthèse* présentée dans le chapitre 11 est finalement la réalisation la plus aboutie du concept de modèle compositionnel. Elle en constitue à la fois une représentation dynamique et malléable, reflétant une situation ou forme sonore abstraite ("essence symbolique et musicale d'un processus de composition"), et un positionnement adopté par le musicien lui permettant, à partir du niveau d'abstraction qu'il a établi, de structurer cette forme musicale. Le *sheet* du chapitre 12 complète le discours des chapitres précédents en mettant en relation la notation musicale avec les différentes représentations et outils de programmation liés au son, ou même avec les maquettes de synthèse.

L'écriture

Avec tout cela, dispose-t-on vraiment d'un système d'*écriture* musicale ? Pour [Veitl, 2007], un tel système aurait pour caractéristiques la matérialité (comme support), la visibilité (comme interface), la lisibilité (c'est-à-dire le fait qu'il y ait des liens évidents entre ce qui est vu et ce à quoi cela fait référence), un caractère performatif (potentiel d'action, d'expression, de création), et un caractère systémique (c'est-à-dire un système de signes permettant d'opérer pratiquement un travail de structuration). Matérialité et visibilité sont présentes dans notre système, par les supports et interfaces proposés. La lisibilité, ou possibilité de dénotation des unités élémentaires, nécessite un apprentissage des principes et codes utilisés : elle est ici possible, avec les limitations que posent l'ouverture d'un tel système ainsi que le nombre réduit de signes proposés. Le caractère performatif est assuré dès lors qu'on est en relation avec des processus de synthèse au sein du système ; cependant il faut encore définir les éléments à abstraire pour en faire des indications performatives [Veitl, 2007] (mais c'est justement l'un des intérêts d'un tel système que de permettre au compositeur le choix de ces indications). Enfin le caractère systémique sur lequel pourrait se fonder un langage et une pratique compositionnelle est encore une fois dépendant des aspirations et pratiques des différents compositeurs.

A ce sujet, nous avons abordé à plusieurs reprises le problème du symbolisme, nécessaire à la composition et à l'écriture, mais non nécessairement acquis au domaine du son et de la synthèse sonore. Il s'agit en fait d'un sous-problème de celui de l'écriture, auquel nous avons associé une idée de subjectivité. Le symbole, qui associe un savoir au signe, se constitue au fil des expériences ; et le positionnement du niveau symbolique devient variable dans la constitution des modèles compositionnels. Nous pouvons donc en cela espérer donner aux utilisateurs les moyens de créer une sémantique subjective des structures et un niveau symbolique qui leur sont propres, et à partir desquels pourrait se développer un système d'écriture personnel.

Perspectives et travaux futurs

Les expériences musicales réalisées avec les différents outils qui composent cet environnement ont montré comment la possibilité d'aborder la synthèse et le traitement du signal pouvait permettre aux utilisateurs de OpenMusic d'en étendre le champ d'utilisation dans ce domaine, ou inversement, permettre à des compositeurs concernés par l'électroacoustique d'utiliser la CAO pour une intégration de la synthèse dans des processus et modèles compositionnels plus avancés.

Différents projets sont envisagés autour des outils que nous avons développés ; les principaux concernent la composition, l'application concrète de ces outils comme mise à l'épreuve des principes du système. La collaboration avec Marco Stroppa sera prolongée dans une recherche de contrôle de la synthèse par règles étendue au niveau des structures temporelles continues (notamment grâce au système de maquettes). La communication avec l'environnement IanniX, l'intégration d'un système de synthèse granulaire (dans le cadre d'un projet du GMEM – Groupe de Musique Expérimentale de Marseille), ou encore la spécification de cibles sonores abstraites dans le cadre du projet Orchestration mené à l'Ircam, sont d'autres exemples de projets initiés en relation avec le travail présenté dans cette thèse.

Même si nous l'avons mentionnés à plusieurs reprises, la synthèse par modèles physiques n'a pas été directement traitée dans nos travaux. S'agissant d'une technologie puissante, et selon nous parmi les plus prometteuses pour l'avenir, une réflexion sur la spécificité de ce type de synthèse et sur la validité, avec celle-ci, des idées et outils que nous avons développés semble donc aussi une direction de recherche pertinente.

Les extensions possibles dans le domaine de la composition sont donc nombreuses de par la diversité des pratiques et technologies de synthèse existantes. Elles le seront aussi, par exemple, par la confrontation de ces principes à d'autres paradigmes de programmation que nous avons peu abordés, comme la programmation par contraintes.

L'analyse et la pédagogie musicales sont également évoquées dans l'annexe de ce document (*Applications*) comme domaines d'application possibles.

Enfin, la spatialisation est un autre domaine dans lequel nous souhaiterions essayer de transposer ou adapter les concepts développés dans le cadre de cette thèse. Il s'agit en effet d'un paramètre de plus en plus considéré dans la composition, mais encore une fois, rares sont les outils permettant de l'intégrer dans les processus compositionnels aux côtés des autres aspects musicaux qui les caractérisent. Des travaux préliminaires ont été menés dans ce domaine avec OpenMusic [Nouno, 2008]. Les notions et outils développés ici pourraient également apporter des éléments et propositions pour une intégration plus étroite de l'espace et du temps dans les modèles compositionnels.

Annexe

Applications

Dans cette annexe sont présentées quelques applications possibles des travaux réalisés dans le cadre de cette thèse, dans les domaines de la composition, de l'analyse, et de la pédagogie musicale.

Composition

La première et principale application concerne naturellement la composition musicale. C'est en effet celle qui a motivé et orienté ce travail.

De nombreux compositeurs ont manifesté un intérêt pour ces nouveaux outils intégrés dans OpenMusic concernant la synthèse et le traitement du son. Parmi eux, un certain nombre d'étudiants ayant suivi le cursus de composition de l'Ircam ou de différents conservatoires, où certains de ces outils ont fait l'objet d'enseignements ces dernières années.

A titre d'exemple, nous décrivons ici quelques aspects de la composition de *Metathesis*, de Tolga Tüzün. Il s'agit d'une pièce pour deux contrebasses et électronique, écrite et créée pour le cursus de composition de l'Ircam en 2006. Plus de détails pourront être trouvés dans [Tüzün, 2008], texte dont sont extraites les figures qui suivent.

Dans *Metathesis*, Tolga Tüzün réalise des analyses de séquences préalablement écrites et jouées à la contrebasse selon des modes de jeux particuliers (position de la contrebasse, disposition de l'archer, etc.) provoquant des caractéristiques timbrales singulières. La figure 13.1 montre une partie des esquisses préliminaires écrites pour les contrebasses.

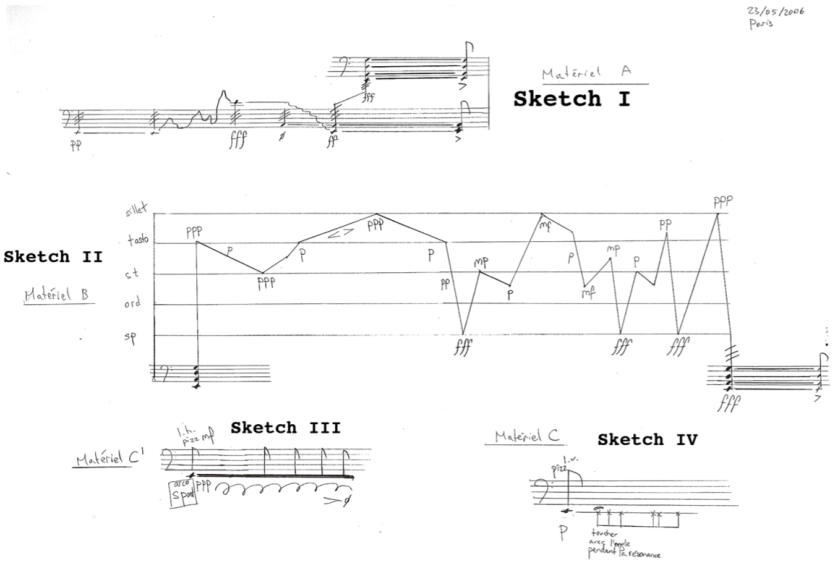


Figure 13.1: Esquisses préliminaires pour la composition de *Metathesis* par Tolga Tüzün.

Les enregistrements de ces esquisses sont ensuite analysés par suivi de partiels dans AudioSculpt (voir figure 13.2), selon différentes configurations des paramètres d'analyse, et les résultats des analyses sont enregistrés dans des fichiers au format SDIF.

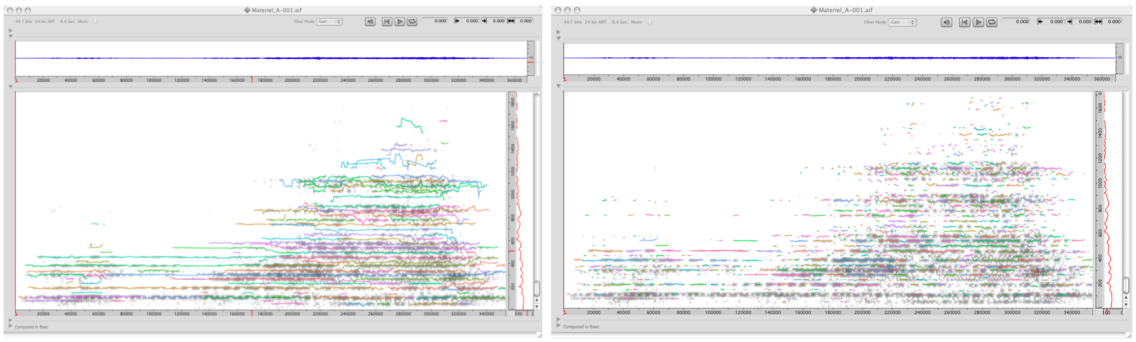


Figure 13.2: Analyses (suivi de partiels) des enregistrements de contrebasse dans AudioSculpt.

Dans un premier temps, le compositeur utilise ainsi les outils d'analyse et de traitement des données SDIF dans OpenMusic (voir chapitre 7, section 7.5) afin d'en extraire des informations harmoniques ou rythmiques. Un exemple de patch est illustré sur la figure 13.3.

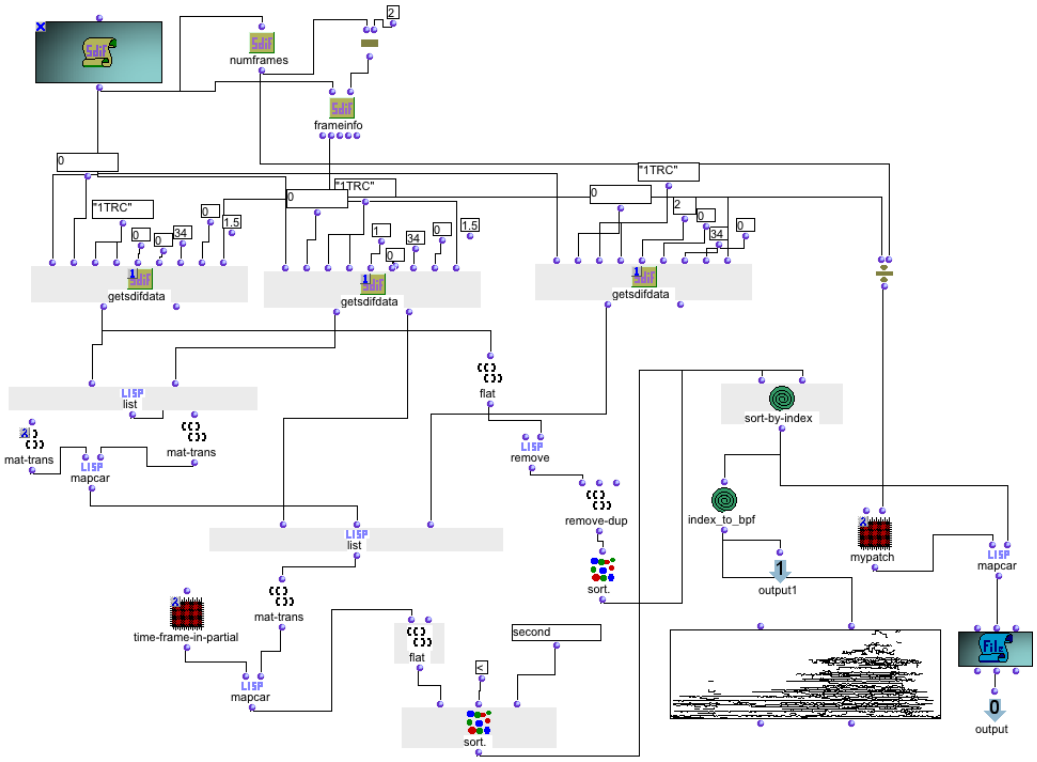


Figure 13.3: Extraction et traitement des données d'analyse dans OpenMusic.

Tolga Tüzün joue sur les paramètres de l'analyse pour obtenir les différents types de matériau : en modifiant les précisions en temps ou en fréquence, ainsi que les seuils de détection des partiels, il obtient par exemple des données avec une cohésion temporelle plus ou moins forte. Avec des résultats fragmentés (à droite sur la figure 13.2), il obtiendra, après transformation, des données rythmiques, alors que les analyses plus cohérentes temporellement seront utilisées comme réservoirs harmoniques.

Le compositeur utilise les données extraites de l'analyse pour construire des maquettes (figure 13.4), qui constituent pour lui un support mieux adapté aux manipulations et au traitement des données.

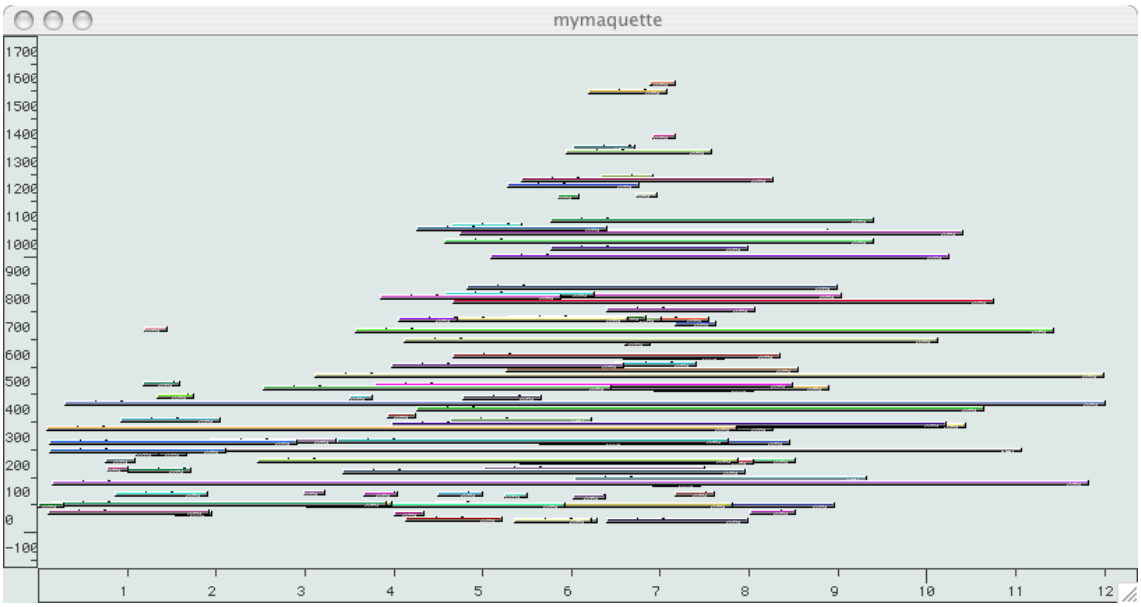


FIGURE 13.4: Conversion des données sous forme de maquette.

Les données contenues dans ces maquettes sont alors utilisées à plusieurs échelles de la composition de la pièce. Tout d'abord, pour l'écriture des parties instrumentales, en convertissant celles-ci en séquences harmoniques (suites d'accords) mises en relation avec les structures rythmiques évoquées plus haut. Le patch visible sur la figure 13.5 illustre cette démarche.

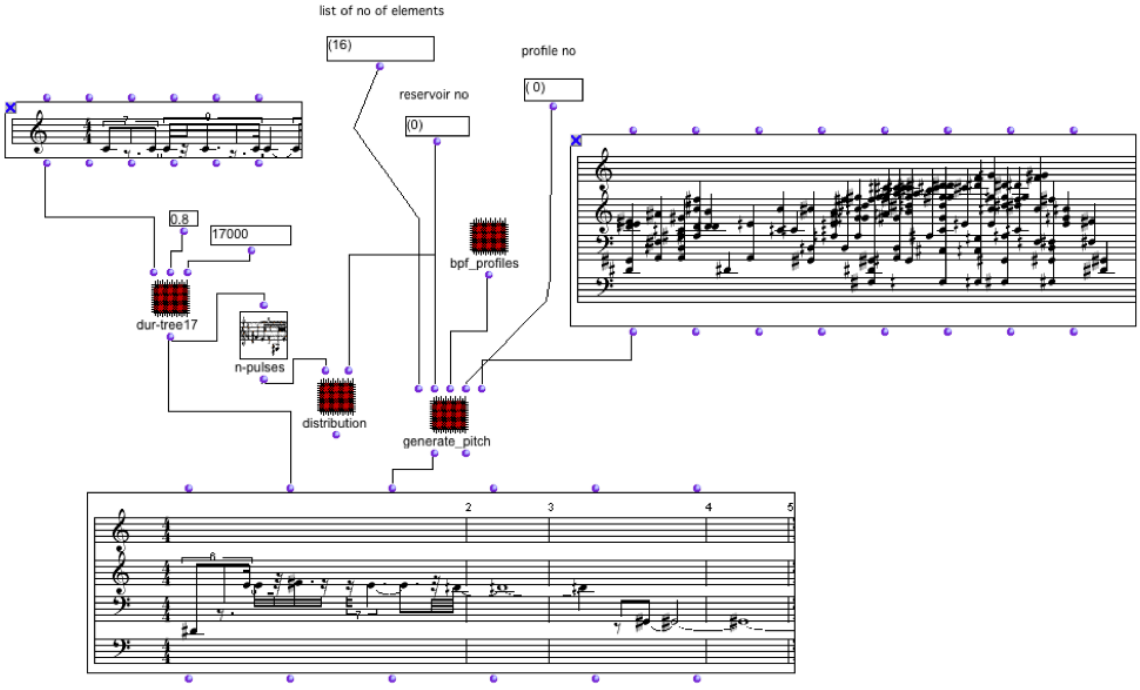


FIGURE 13.5: Utilisation des données harmoniques et rythmiques issues des analyses spectrales pour la création des parties instrumentales de *Metathesis*.

Ensuite, pour les parties électroniques de la pièce. Les données contenues dans les maquettes sont utilisées pour paramétrer des processus de synthèse réalisés dans OpenMusic avec les systèmes OMChroma ou OM-Modalys (voir respectivement les chapitres 9, section 9.2 et 5, section 5.2.3). Les sons obtenus sont ensuite eux-mêmes traités et transformés à l'aide des outils de OM-SuperVP par *time-stretching*, transposition, synthèse croisée, etc. (voir chapitre 6, section 6.4). La figure 13.6 montre un exemple d'un tel processus.

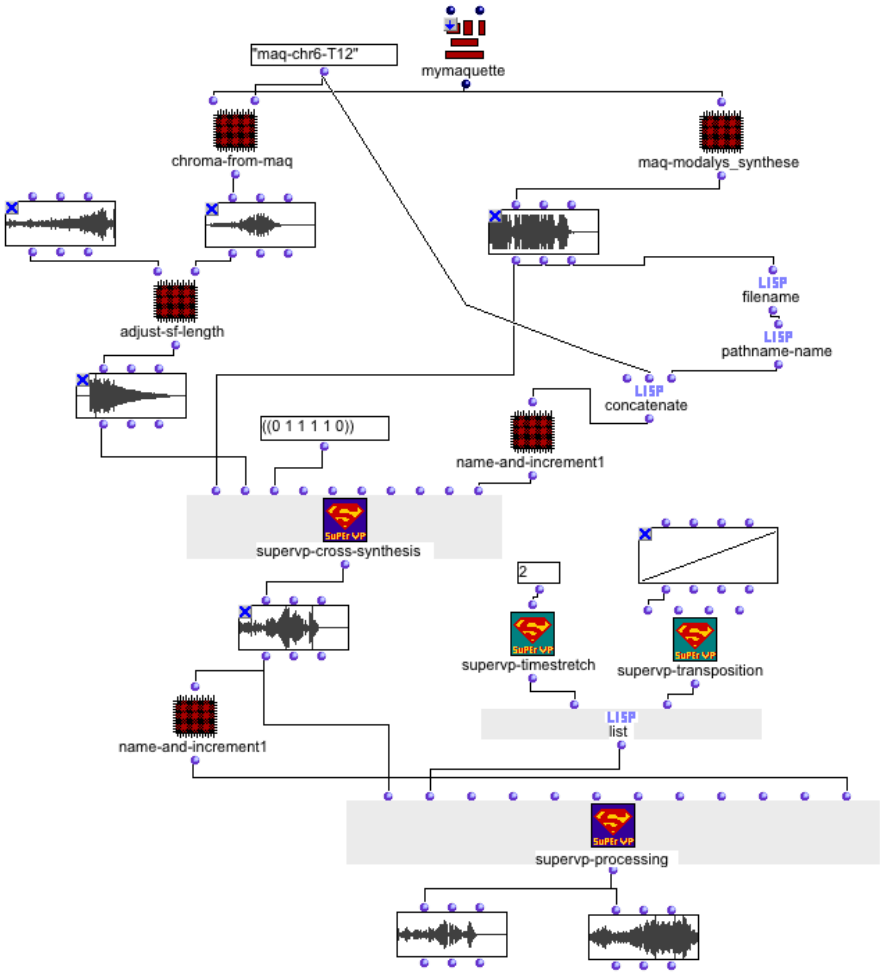


Figure 13.6: Construction des parties électroniques de *Metathesis*. Les données issues d'une maquette (en haut) sont utilisées comme paramètres pour la synthèse sonore. Les sons obtenus sont traités avec les fonctions de OM-SuperVP.

Les maquettes sont également utilisées à plus grande échelle par le compositeur afin d'intégrer les parties instrumentales et électroniques pendant la composition (voir figure 13.7). Comme on le voit sur la maquette de la figure 13.7, les parties électroniques sont elles-mêmes des maquettes utilisant les fonctionnalités de synthèse décrites dans le chapitre 11. Sur la figure 13.8 on peut voir le détail de l'une de ces maquettes, ainsi que le patch de la fonction de synthèse correspondante.

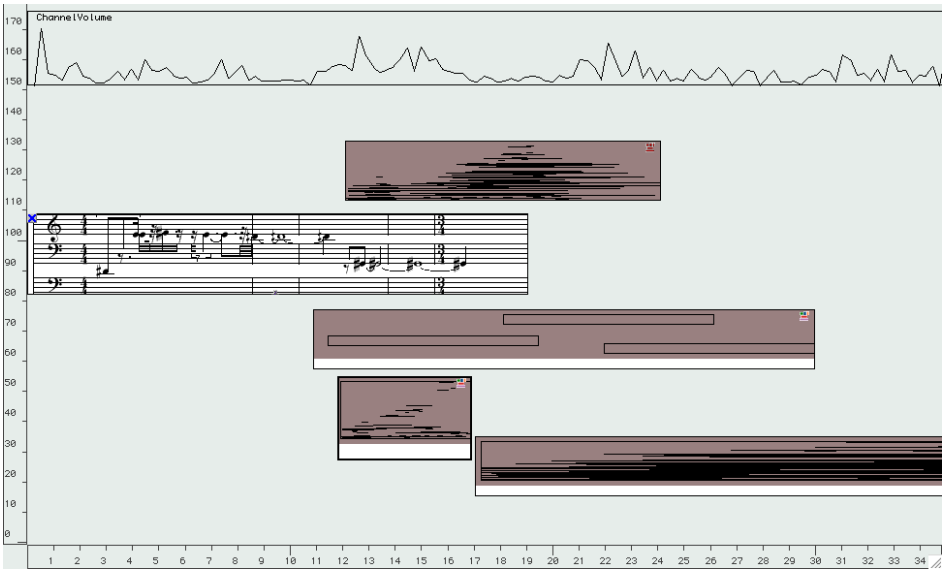


FIGURE 13.7: Intégration des différents éléments de la pièce dans une maquette.

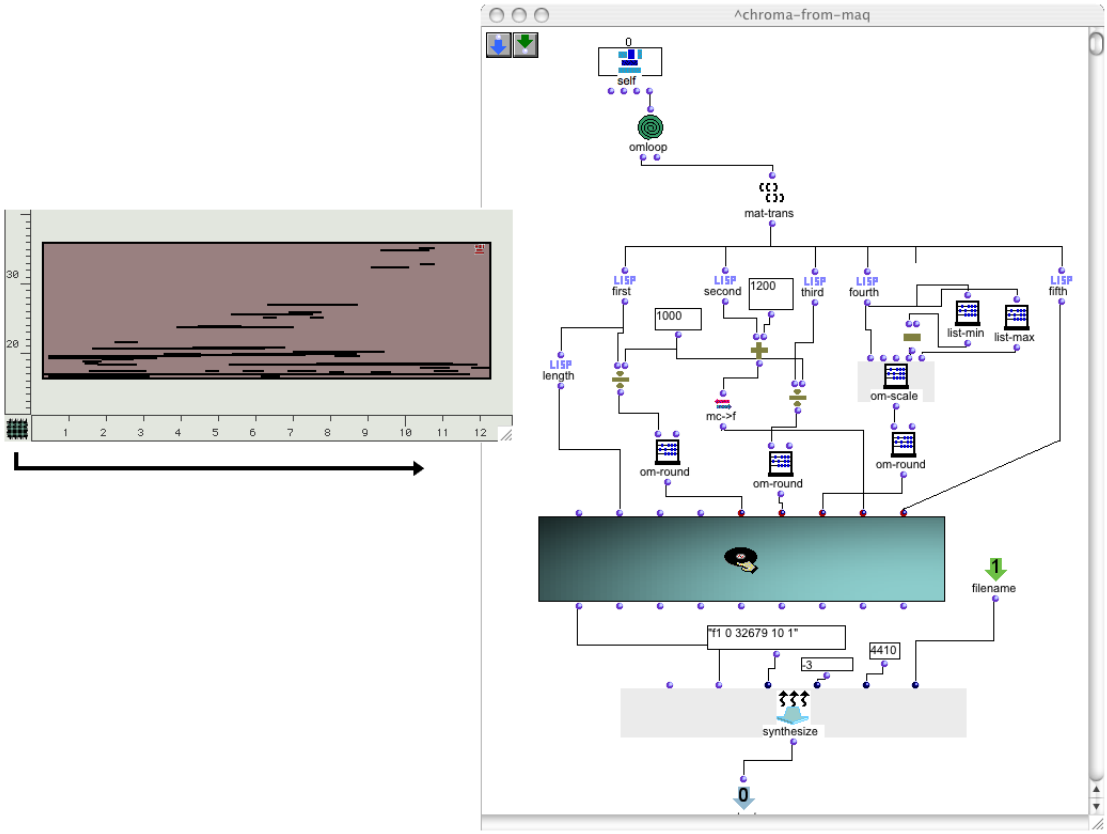


FIGURE 13.8: Maquette de synthèse : interprétation des données de la maquette (à gauche) par un processus de synthèse (à doite).

La figure 13.9 montre un extrait de la partition finale de la pièce.

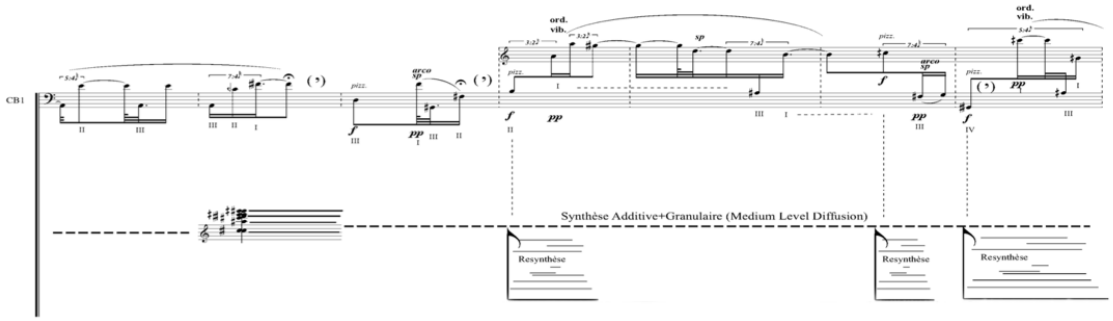


FIGURE 13.9: Extrait de la partition finale de *Metathesis* avec partie instrumentale, et parties électroniques suggérées en dessous.

Analyse / musicologie

Le développement des études sur les aspects calculatoires dans la théorie de la musique insère ceux-ci dans une dimension musicologique (voir par exemple [Mazzola *et al.*, 2004]) au sein de laquelle les modèles mathématiques appliqués à la musique peuvent se prêter à des implémentations informatiques, conduisant à une "analyse musicale assistée par ordinateur" [Riotte et Mesnage, 2006]. La "musicologie computationnelle" se positionne ainsi comme une discipline émergente entre les domaines de l'analyse et de l'informatique musicale, étendant l'étude des oeuvres à une dimension calculatoire, et donc susceptible d'être mise en oeuvre par des systèmes informatiques [Andreatta, 2003]. L'environnement OpenMusic peut être utilisé dans ce contexte pour l'implémentation de modèles computationnels de pièces, permettant la formalisation et l'analyse de celles-ci (voir par exemple [Agon *et al.*, 2004], [Noll *et al.*, 2006]).

Les outils réalisés dans le cadre de cette thèse permettent alors d'envisager une extension de cette démarche au domaine de la musique électroacoustique, pour lesquelles l'absence de partition pose des problématiques spécifiques pour l'analyse. Cette extension pourra ainsi viser l'analyse de pièces particulières,[1] ou la mise en place d'outils plus généraux. Ce dernier cas a fait notamment l'objet d'une collaboration avec Jean-Marc Chouvel[2] [Chouvel *et al.*, 2007].

Avec les outils d'analyse sonore et ceux de visualisation et traitement des données de description sonore, nous avons été en mesure de mener différentes expérimentations, principalement autour du concept d'analyse spectrale "différentielle", c'est-à-dire une analyse s'attachant aux variations de l'énergie sur le spectre comme éléments sonores sujets à l'analyse et à la formalisation.

La figure 13.10 montre un patch OpenMusic réalisant une analyse FFT (TFCT, plus exactement) d'un fichier audio, puis traitant automatiquement les données d'analyse obtenues (par l'intermédiaire d'un fichier SDIF) afin d'en extraire ces informations différentielles. Ce traitement, comme on peut le voir sur la figure, fait appel aux outils présentés dans le chapitre 7 pour le traitement itératif et l'écriture dans un nouveau fichier des données au format SDIF. Le résultat de ce traitement est un "spectre différentiel", que l'on peut visualiser dans SDIF-Edit (voir figure 13.11).

[1] Nous avons présenté dans le dernier chapitre un exemple possible de reconstitution d'une pièce de Karlheinz Stockhausen, et évoqué les ouvertures musicologique de ce type de démarche.

[2] Compositeur et musicologue, professeur à l'université de Reims.

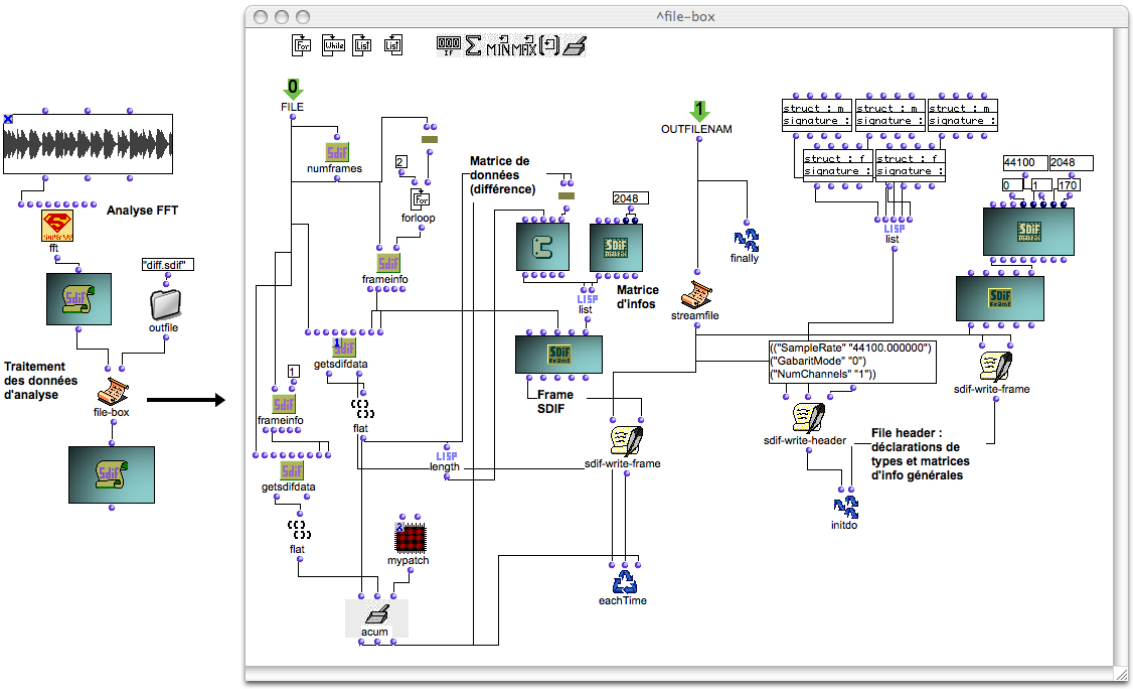


FIGURE 13.10: Implémentation d'un processus d'analyse à partir des outils de l'environnement OMSOUNDS.

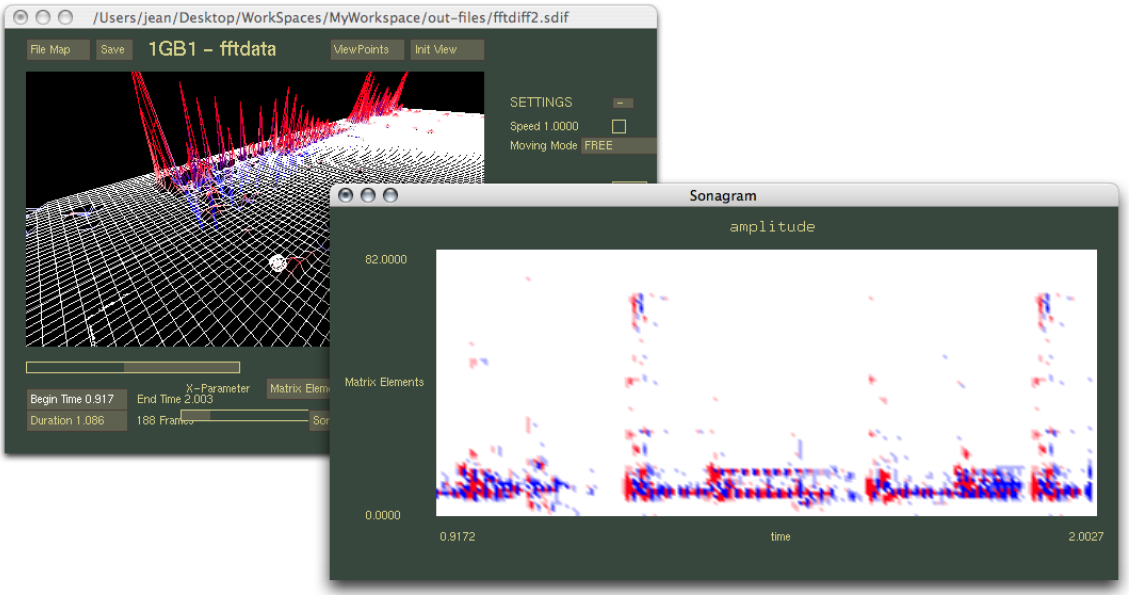


FIGURE 13.11: Visualisation des données issues de l'analyse dans la figure 13.10 avec SDIF-EDIT.

Sans prétendre avoir atteint un degré de généralité adapté à l'analyse de toute pièce électroacoustique, cette expérience met surtout en avant la possibilité de combiner librement les différents outils d'analyse, de traitement, voire de resynthèse des données dans des processus exploratoires d'analyse, d'une manière similaire aux processus de construction de modèles compositionnels (voir chapitre 4). Dans notre cas, une fois le processus validé, une implémentation directe de celui-ci en C++ nous a permis de réaliser un outil figé (également utilisable depuis OpenMusic) et plus performant (voir figure 13.12). Des traitements ultérieurs sur ces données, notamment sous formes de "statistiques" (valeurs moyennées sur l'ensemble du spectre), sont visibles sur cette dernière figure, et permettent d'obtenir des informations supplémentaires pour l'analyse.

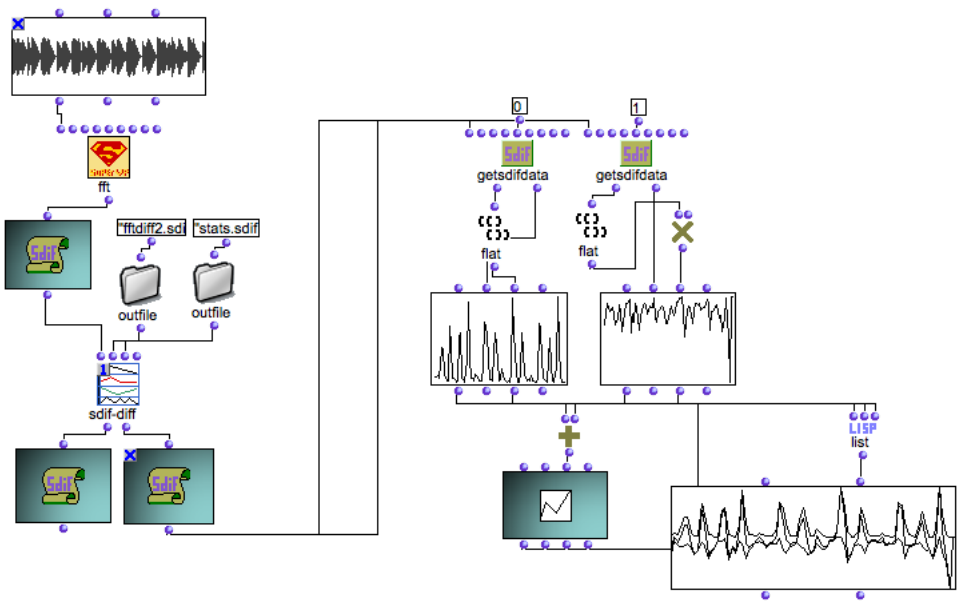


Figure 13.12: Encapsulation du processus d'analyse de la figure 13.10 dans une fonction générique (`sdif-dif`) et analyses statistiques des résultats dans OpenMusic.

Pédagogie musicale

OpenMusic a été utilisé comme plateforme pour le développement de l'application Musique Lab Maquette (ou ML-Maquette) [Bresson *et al.*, 2006], réalisée dans le cadre du projet Musique Lab 2 [Puig *et al.*, 2005] sur commande du Ministère de l'Education Nationale et du Ministère de la Culture et de la Communication.[3]

ML-Maquette est une application d'aide à la pédagogie musicale, centrée sur les principes de constructions musicales formelles et donc sur une approche compositionnelle de la musique. Par ce projet, les applications et le savoir-faire dont dispose OpenMusic dans le domaine de la CAO ont été étendus à un contexte pédagogique [Assayag et Bresson, 2006]. ML-Maquette permet ainsi à un utilisateur (professeur ou élève de classe de musique) de reconstituer et concrétiser dans un document dynamique diverses procédures et expériences dans le but d'illustrer et d'étudier des concepts ou extraits musicaux. Ces expériences pourront par exemple concerner la manipulation des différents paramètres des structures musicales (hauteurs, rythmes, dynamique), les transformations et opérations sur ces structures, l'harmonie, l'intégration temporelle des objets musicaux, etc.

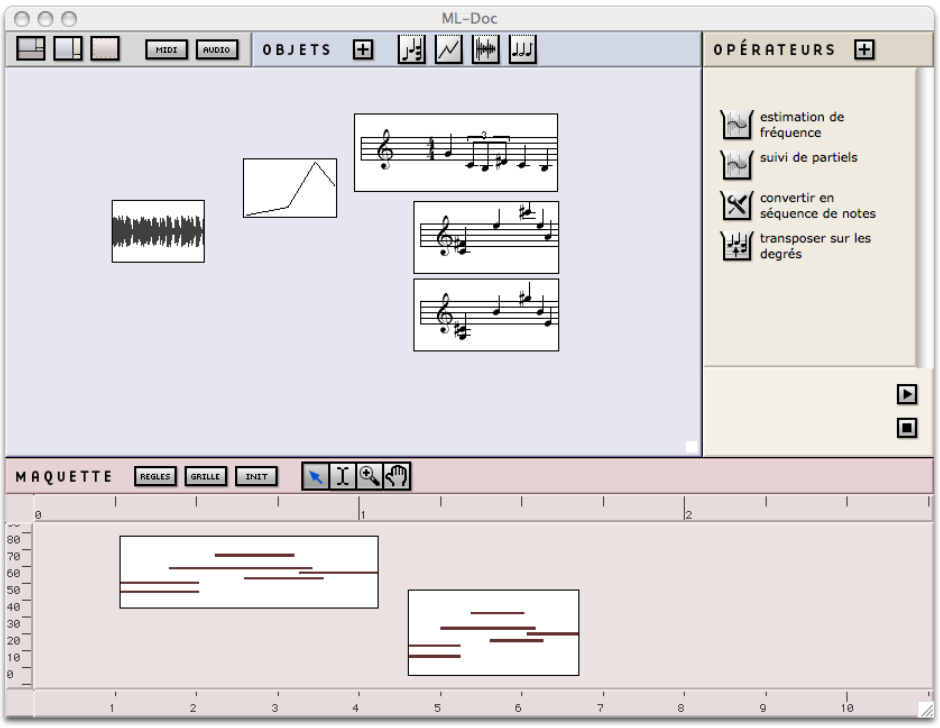


Figure 13.13: Un document dans ML-Maquette.

[3]Ce projet s'insère dans le cadre du dispositif de soutien aux ressources multimédia piloté par la Direction de la Technologie du Ministère de l'Education Nationale.

La figure 13.13 montre une fenêtre de l'application représentant un document ML-Maquette. Trois zones composent ce document. La zone *objets*, située dans la partie supérieure gauche, permet d'instancier différentes classes d'objets musicaux (accords, séquences, courbes, sons.) La zone *opérateurs*, située dans la partie supérieure droite, contient un ensemble de fonctions capables d'opérer des traitements sur les objets. Ces fonctions (opérateurs) sont sélectionnées par l'utilisateur parmi une bibliothèque, en fonction de l'application visée dans chaque document.

En glissant un objet sur l'un des opérateurs, et généralement après la saisie de différents paramètres nécessaires à sa fonction, on obtient un nouvel objet pouvant être à son tour inséré parmi les autres dans la zone *objets* (voir figure 13.14). Un cycle de création et de transformation d'objets musicaux peut ainsi être développé, selon une démarche expérimentale et pédagogique visée.

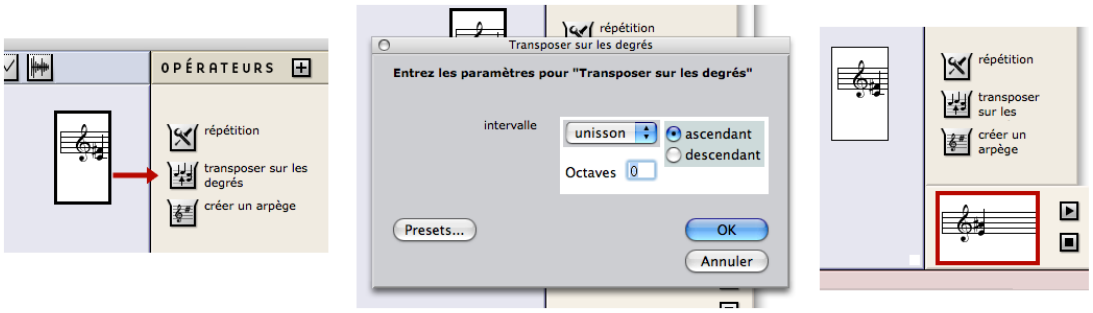


Figure 13.14: Application d'un opérateur sur un objet musical dans ML-Maquette : 1) l'objet (un accord) est glissé sur l'opérateur (une transposition) ; 2) une fenêtre de dialogue permet d'entrer les paramètres de la transformation (l'intervalle de transposition) ; 3) un nouvel objet est créé.

Enfin la zone *maquette*, située dans la partie inférieure du document visible sur la figure 13.13, permet d'insérer dans un contexte temporel les objets préalablement constitués. Cet espace permet, outre l'assemblage et le montage des objets musicaux, d'affecter à ceux-ci des comportements fonctionnels avancés selon l'opérateur qui les a générés (voir [Bresson *et al.*, 2006] pour une description plus détaillée). Dans l'ensemble d'un document peuvent ainsi se développer des scénarios et pédagogies variés liés à la création et la manipulation de matériaux et structures musicales [Guédy *et al.*, 2007].

La bibliothèque d'opérateurs disponibles dans ML-Maquette constitue donc un réservoir d'outils pour la réalisation des documents pédagogiques. Ces opérateurs sont classés en différents groupes, se rapportant à autant de domaines d'application.

En règle générale, un opérateur correspond à une méthode écrite en Lisp/CLOS et enregistrée dans l'application selon certaines règles syntaxiques. Le traitement de cette méthode permet alors à l'application d'intégrer automatiquement celle-ci parmi les opérateurs, en particulier pour la création des composants adaptés dans l'interface graphique. Ce

protocole est détaillé dans [Bresson, 2007]. Des processus de traitement et/ou de génération de matériau musical développés dans OpenMusic peuvent ainsi être intégrés facilement dans ML-Maquette en tant qu'opérateurs (soit par une réécriture en Lisp, soit en utilisant les possibilités de persistance et d'exportation des méthodes créées graphiquement).

Ainsi certains outils d'analyse, de traitement et de synthèse sonores (initialement créés dans un objectif de composition) ont-ils pu être adaptés et utilisés dans cet environnement pédagogique pour la création d'opérateurs relevant notamment de l'analyse du signal sonore et de la conversion des données d'analyse en structures musicales de haut niveau, ou de la démarche symétrique consistant à produire des signaux sonores à partir de structures musicales symboliques. Nous avons principalement utilisé pour cela le programme pm2, permettant d'effectuer des analyses et synthèses additives (voir chapitre 8, section 8.3).

A partir d'un processus comme celui donné en exemple sur la figure 13.15, réalisant l'analyse additive d'un son, et dont le résultat est converti en une représentation symbolique, il est possible de créer un opérateur pour ML-Maquette. Un tel processus représente en effet une unité de traitement cohérente, permettant de passer d'une structure de donnée initiale (un son numérique) à une autre (un séquence de notes et/ou d'accords), et se prête ainsi à une adaptation sous cette forme d'opérateur.

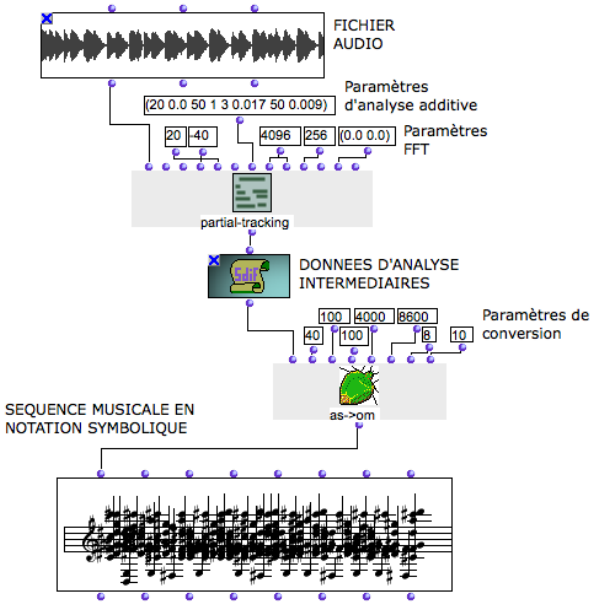


Figure 13.15: Un processus d'analyse du signal sonore (suivi de partiel) implémenté dans OpenMusic.

Après un choix des variables dans le processus, qui seront les paramètres présentés à l'utilisateur dans ML-Maquette, et une traduction selon la syntaxe des opérateurs, cette unité de traitement est rendue disponible dans le contexte pédagogique. Parmi toutes les variables (ou paramètres) possibles de ce processus (dont les principales visibles sur le patch), nous n'avons par exemple conservé ici, pour l'opérateur ML-Maquette corres-

pondant, que les bornes fréquentielles exprimées en hauteur symbolique pour le filtrage du résultat, ainsi que la polyphonie maximale, ou nombre de notes pouvant être considérées simultanément dans un même accord. Les paramètres restants sont donc fixés par des valeurs par défaut. L'utilisation de cet opérateur dans ML-Maquette est illustrée sur la figure 13.16.

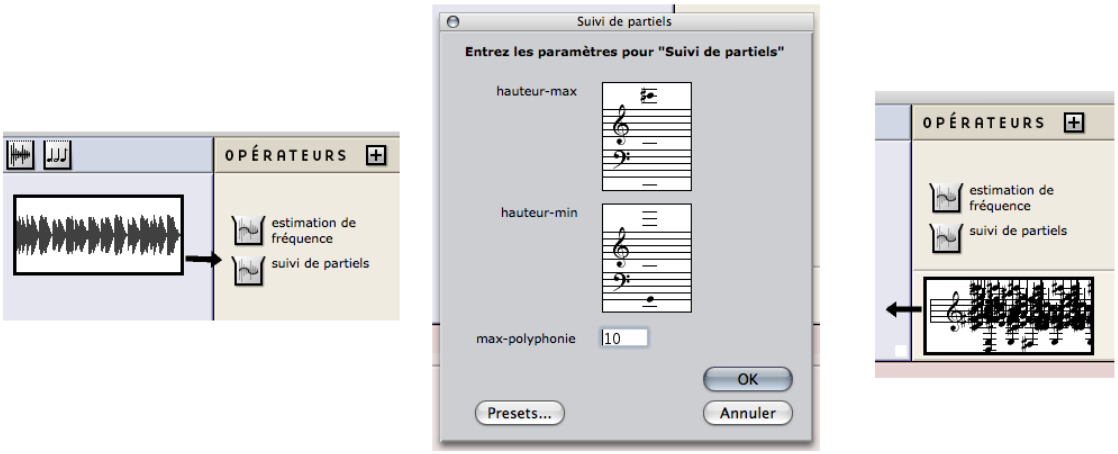


Figure 13.16: Utilisation de l'opérateur de suivi de partiels dans ML-Maquette : 1) un son est glissé depuis la zone *objets* sur l'opérateur ; 2) un formulaire demande d'entrer certains paramètres pour effectuer cette opération ; 3) un nouvel objet apparaît, qui peut être à nouveau glissé dans la zone *objets*.

Différents opérateurs ont ainsi été élaborés selon ce même principe. Parmi ceux-ci, on trouvera notamment le processus inverse, qui réalise une synthèse additive à partir des notes d'une séquence musicale ou d'un accord considérées comme des partiels. On trouvera aussi l'analyse par accord, qui utilise une segmentation préalable du signal sonore pour réaliser une analyse additive plus structurée (*chord sequence*), ou encore l'estimation et le suivi de fréquence fondamentale. Ces opérateurs sont illustrés sur la figure 13.17.

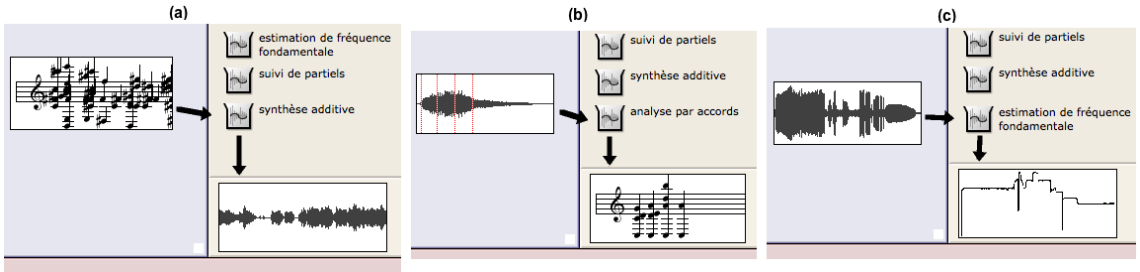


Figure 13.17: Opérateurs d'analyse/synthèse dans ML-Maquette : (a) synthèse additive, (b) analyse additive par accords (*chord sequence*), (c) estimation de fréquence fondamentale.

Les opérateurs de traitement du signal sont ainsi des outils très utilisés dans les expériences pédagogiques menées jusqu'ici avec ML-Maquette. Ils mettent en effet rapidement en évidence des notions fondamentales de la musique actuelle telles que les relation entre le son acoustique et sa représentation en hauteurs, ou encore entre le son naturel enregistré et le son de synthèse (dans une démarche d'analyse/resynthèse, par exemple). Les élèves ont ainsi la possibilité d'expérimenter et d'étudier ce type de concept par la construction et l'ecoute, donc avec un certain recul sur le matériau sonore et musical, complémentaire avec l'interaction directe permise par les environnements de traitement du signal en temps réel (dont notamment ML-Audio, le module correspondant dans le projet Musique Lab 2 [Guédy, 2006]). Ils peuvent ensuite réaliser eux-mêmes des séquences musicales (dans la zone *maquette*) en s'inspirant de pièces étudiées, se réappropriant ainsi l'écoute de ces pièces.

Ce type d'expérience pédagogique à été réalisé dans différents cours et ateliers de musique (en collège, conservatoire, ou autre), ainsi que dans des ateliers organisés par l'Ircam, autour d'oeuvres comme *Partiels* de Gérard Grisey, *Désintégrations* de Tristan Murail, ou encore *Le plein du vide* de Xu Yi, au programme du baccalauréat 2007. La figure 13.18 montre un exemple de document ML-Maquette créé par un élève au cours d'un atelier portant sur cette dernière pièce.[4]

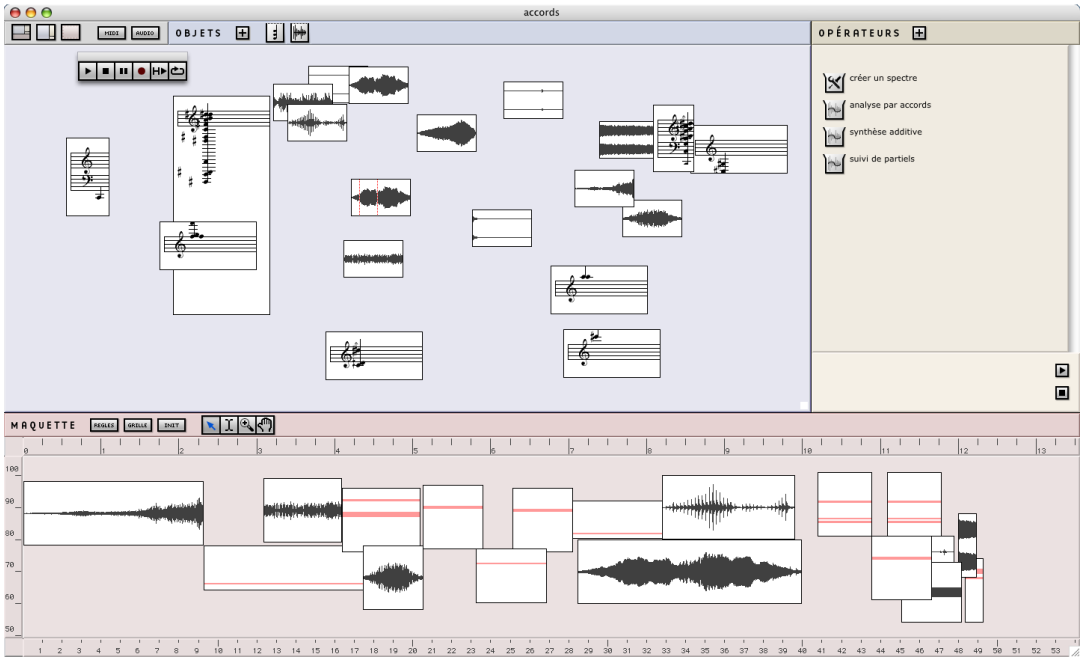


Figure 13.18: Exemple de travail réalisé par un élève dans ML-Maquette avec les outils d'analyse/synthèse sonore.

[4] Atelier encadré par Grégoire Lorieux à l'Ircam.

Bibliographie

[Accaoui, 2001] Accaoui, C. (2001). *Le temps musical*. Desclée de Brouwer.

[Adrien, 1991] Adrien, J.-M. (1991). The missing link : modal synthesis. *In* De Poli, G., Piccialli, A. et Roads, C., éditeurs : *Representation of Musical Signals*. MIT Press.

[Agon, 1998] Agon, C. (1998). *OpenMusic : Un langage visuel pour la composition musicale assistée par ordinateur*. Thèse de doctorat, Université Pierre et Marie Curie (Paris 6), Paris, France.

[Agon, 2004] Agon, C. (2004). Langages de programmation pour la composition musicale. Habilitation à diriger des recherches, Université Pierre et Marie Curie - Paris 6.

[Agon *et al.*, 2004] Agon, C., Andreatta, M., Assayag, G. et Schaub, S. (2004). Formal Aspects of Iannis Xenakis' "Symbolic Music" : A Computer-Aided Exploration of Compositional Processes. *Journal of New Music Research*, 33(2).

[Agon et Assayag, 2002] Agon, C. et Assayag, G. (2002). Programmation visuelle et éditeurs musicaux pour la composition assistée par ordinateur. *In 14ème Conférence Francophone sur l'Interaction Homme-Machine IHM'02*, Poitiers, France.

[Agon et Assayag, 2003] Agon, C. et Assayag, G. (2003). OM : A Graphical Extension of CLOS using the MOP. *In Proceedings of ICL'03*, New York, USA.

[Agon *et al.*, 2006] Agon, C., Assayag, G. et Bresson, J., éditeurs (2006). *The OM Composer's Book. 1.* Editions Delatour / Ircam.

[Agon et Bresson, 2007] Agon, C. et Bresson, J. (2007). Mixing Time Representations in a Programmable Score Editor. *In Proceedings of Sound and Music Computing Conference - SMC'07*, Lefkada, Greece.

[Agon *et al.*, 2002] Agon, C., Haddad, K. et Assayag, G. (2002). Representation and Rendering of Rhythmic Structures. *In Second International Conference on Web Delivering of Music*, Darmstadt, Germany.

[Agon *et al.*, 2000] Agon, C., Stroppa, M. et Assayag, G. (2000). High Level Control of Sound Synthesis in OpenMusic. *In Proceedings of the International Computer Music Conference*, Berlin, Germany.

[Allen et Rabiner, 1977] Allen, J. B. et Rabiner, L. R. (1977). A unified approach to short-time Fourier analysis and synthesis. *Proceedings of the IEEE*, 65(11).

[Allen, 1983] Allen, J. F. (1983). Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26.

[Allen, 1991] Allen, J. F. (1991). Time and Time Again : The Many Ways to Represent Time. *International Journal of Intelligent Systems*, 6(4).

[Allen et Ferguson, 1994] Allen, J. F. et Ferguson, G. (1994). Actions and Events in Interval Temporal Logic. *Journal of Logic and Computation*, 4(5).

[Allombert *et al.*, 2006] Allombert, A., Assayag, G., , Desainte-Catherine, M. et Rueda, C. (2006). Concurrent Constraints Models for Interactive Scores. *In Proceedings of the Sound and Music Computing Conference SMC'06*, Marseille, France.

[Anderson et Kuivila, 1989] Anderson, D. P. et Kuivila, R. (1989). Continuous Abstractions for Discrete Events Languages. *Computer Music Journal*, 13(3).

[Andreatta, 2003] Andreatta, M. (2003). *Méthodes algébriques en musique et musicologie du XX^{e} siècle : aspects théoriques, analytiques et compositionnels*. Thèse de doctorat, Ecole des Hautes Etudes en Sciences Sociales, Paris, France.

[Arfib, 1979] Arfib, D. (1979). Digital synthesis of complex spectra by means of multiplication of non-linear distorted sine waves. *Journal of the Audio Engineering Society*, 27.

[Arfib et Kronland-Martinet, 1993] Arfib, D. et Kronland-Martinet, R. (1993). Transformer le son : Modeler, Modéliser. *In Les cahiers de l'Ircam (2) - La synthèse sonore*. Ircam - Centre G. Pompidou.

[Assayag, 1993] Assayag, G. (1993). CAO : Vers la Partition Potentielle. *In Les cahiers de l'Ircam (3) - La Composition Assistée par Ordinateur*. Ircam - Centre G. Pompidou.

[Assayag, 1995] Assayag, G. (1995). Visual Programming in Music. *In Proceedings of the International Computer Music Conference*, Banff, Canada.

[Assayag, 1998] Assayag, G. (1998). Computer Assisted Composition today. *In 1st symposium on music and computers*, Corfu.

[Assayag et Agon, 1996] Assayag, G. et Agon, C. (1996). OpenMusic Architecture. *In Proceedings of the International Computer Music Conference*, Hong Kong.

[Assayag *et al.*, 1997a] Assayag, G., Agon, C., Fineberg, J. et Hanappe, P. (1997a). An Object Oriented Visual Environment for Musical Composition. *In Proceedings of the International Computer Music Conference*, Thessaloniki, Greece.

[Assayag *et al.*, 1997b] Assayag, G., Agon, C., Fineberg, J. et Hanappe, P. (1997b). Problèmes de notation dans la composition assistée par ordinateur. *In Actes des rencontres pluridisciplinaires musique et notation*, Lyon, France. Grame.

[Assayag et Bresson, 2006] Assayag, G. et Bresson, J. (2006). OpenMusic : de la composition à l'enseignement. *L'Inouï*, 2. Ircam - Léo Scheer.

[Assayag *et al.*, 1985] Assayag, G., Castellengo, M. et Malherbe, C. (1985). Functional Integration of Complex Instrumental Sounds in Music Writing. *In Proceedings of the International Computer Music Conference*, Burnaby, Canada.

[Assayag et Cholleton, 1994] Assayag, G. et Cholleton, J.-P. (1994). L'appareil musical. *Résonance*, 7.

[Assayag *et al.*, 2006] Assayag, G. (2006). Bloch, G., Chemiller, M., OMax - Ofon. *In Proceedings of the Sound and Music Computing Conference SMC'06*, Marseille, France.

[Baboni-Schilingi et Malt, 1995] Baboni-Schilingi, J. et Malt, M. (1995). Profile, librairie pour le contrôle de profils mélodiques. *Ircam.*

[Balaban et Elhadad, 1999] Balaban, M. et Elhadad, M. (1999). On the Need for Visual Formalisms for Music Processing. *Leonardo*, 32(2).

[Balaban et Murray, 1998] Balaban, M. et Murray, N. (1998). Interleaving Time and Structures. *Computer and Artificial Intelligence*, 17(4).

[Barbar *et al.*, 1993] Barbar, K., Desainte-Catherine, M. et Miniussi, A. (1993). The Semantics of Musical Hierarchies. *Computer Music Journal*, 17(4).

[Barbaud, 1968] Barbaud, P. (1968). *La musique, discipline scientifique.* Dunod.

[Barendregt, 1984] Barendregt, C. (1984). *The Lambda-Calculus. Volume 103.* Elsevier Science Publishing Company.

[Barrière, 1991] Barrière, J.-B., éditeur (1991). *Le timbre, métaphore pour la composition.* Ircam - Christian Bourgeois Éditions.

[Barrière *et al.*, 1985] Barrière, J.-B., Potard, Y. et Baisnée, P.-F. (1985). Models of Continuity between Synthesis and Processing for the Elaboration and Control of Timbre Structures. *In Proceedings of the International Computer Music Conference*, Vancouver, Canada.

[Bartels *et al.*, 1987] Bartels, R. H., Beatty, J. C. et Barsky, B. A. (1987). *An introduction to splines for use in computer graphics & geometric modeling.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Battier, 1988] Battier, M. (1988). *Elements d'informatique musicale.* IRCAM - Centre Pompidou.

[Bel, 1990] Bel, B. (1990). Time an dmusical structures. *Interface, Journal of New Music Research*, 19(2-3):107–135.

[Bennett, 1990] Bennett, G. (1990). Repères électro-acoustiques. *Contrechamps*, 11:29–52.

[Bennett et Rodet, 1989] Bennett, G. et Rodet, X. (1989). Synthesis of the Singing Voice. *In* Mathews, M. V. et Pierce, J. R., éditeurs : *Current Directions in Computer Music Research.* MIT press.

[Berthier, 2002] Berthier, D. (2002). *Le savoir et l'ordinateur.* L'Harmattan.

[Beurivé, 2000] Beurivé, A. (2000). Un logiciel de composition musicale combinant un modèle spectral, des structures hiérarchiques et des contraintes. *In Actes de Journées d'Informatique Musicale*, Bordeaux, France.

[Beurivé et Desainte-Catherine, 2001] Beurivé, A. et Desainte-Catherine, M. (2001). Representing Musical Hierarchies with Constraints. *In CP'2001 - Musical Constraints Workshop*, Paphos, Cyprus.

[Boesch, 1990] Boesch, R. (1990). L'électricité en musique. *Contrechamps*, 11:134–145.

[Bogaards, 2005] Bogaards, N. (2005). Analysis-assisted sound processing with audiosculpt. *In 8^{th} International Conference on Digital Audio Effects (DAFX-05)*, Madrid, Spain.

[Bogaards et Röbel, 2004] Bogaards, N. et Röbel, A. (2004). An interface for analysis-driven sound processing. *In AES 119^{th} Convention*, New York, USA.

[Bonnet et Rueda, 1998] Bonnet, A. et Rueda, C. (1998). Situation : Un langage visuel basé sur les contraintes pour la composition musicale. *In* Chemillier, M. et Pachet, F., éditeurs : *Recherches et applications en informatique musicale*. Hermes, Paris.

[Boulanger, 2000] Boulanger, R., éditeur (2000). *The Csound Book*. MIT Press.

[Boulez, 1963] Boulez, P. (1963). *Penser la musique aujourd'hui*. Gonthier.

[Boulez, 1987] Boulez, P. (1987). Timbre and composition - timbre and language. *Contemporary Music Review*, 2(1).

[Bresson, 2003] Bresson, J. (2003). Représentation et manipulation de données d'analyse sonore pour la composition musicale. Rapport de stage Ircam / Ecole Supérieure en Sciences Informatiques - Université de Nice-Sophia Antipolis.

[Bresson, 2006a] Bresson, J. (2006a). SDIF-Edit 1.4 – User Documentation. Ircam Software Documentation.

[Bresson, 2006b] Bresson, J. (2006b). Sound Processing in OpenMusic. *In Proceedings of the 9^th International Conference on Digital Audio Effects - DAFx-06*, Montreal, Canada.

[Bresson, 2007] Bresson, J. (2007). Processus compositionnels et opérateurs musicaux dans ML-Maquette – Les outils de traitement du signal. *In Actes de Journées d'informatique Musicale – JIM'07*, Lyon, France.

[Bresson et Agon, 2004] Bresson, J. et Agon, C. (2004). SDIF Sound Description Data Representation and Manipulation in Computer-Assisted Composition. *In Proceedings of the International Computer Music Conference*, Miami, USA.

[Bresson et Agon, 2006a] Bresson, J. et Agon, C. (2006a). Sound Writing and Representation in a Visual Programming Framework. *In Digital Music Research Network Doctoral Research Conference - DMRN-06*, Goldsmith College, University of London, UK.

[Bresson et Agon, 2006b] Bresson, J. et Agon, C. (2006b). Temporal Control over Sound Synthesis Processes. *In Proceedings of Sound and Music Computing Conference - SMC'06*, Marseille, France.

[Bresson *et al.*, 2005a] Bresson, J., Agon, C. et Assayag, G. (2005a). OpenMusic 5 : A Cross-Platform Release of the Computer-Assisted Composition Environment. *In Proceedings of the 10^{th} Brazilian Symposium on Computer Music - SBCM'05*, Belo Horizonte, MG, Brasil.

[Bresson *et al.*, 2008] Bresson, J., Agon, C. et Assayag, G., éditeurs (2008). *The OM Composer's Book. 2.* Editions Delatour / Ircam (à paraître).

[Bresson *et al.*, 2006] Bresson, J., Guédy, F. et Assayag, G. (2006). Musique Lab Maquette : approche interactive des processus compositionnels pour la pédagogie musicale. *Revue STICEF – Sciences et Technologies de l'information et de la Communication pour l'Education et la Formation*, 13.

[Bresson *et al.*, 2005b] Bresson, J., Stroppa, M. et Agon, C. (2005b). Symbolic Control of Sound Synthesis in Computer-Assisted Composition. *In Proceedings of the International Computer Music Conference*, Barcelona, Spain.

[Bresson *et al.*, 2007] Bresson, J., Stroppa, M. et Agon, C. (2007). Generation and Representation of Data and Events for the Control of Sound Synthesis. *In Proceedings of Sound and Music Computing Conference - SMC'07*, Lefkada, Greece.

[Cadoz, 1979] Cadoz, C. (1979). *Synthèse sonore par simulaion de mécanismes vibratoires.* Thèse de Docteur Ingénieur, Spécialité Electronique, Institut National Polytechnique de Grenoble, France.

[Cadoz, 2002] Cadoz, C. (2002). The Physical Model as Metaphor for Musical Creation "pico..TERA", a piece entirely generated by physical model. *In Proceedings of the International Computer Music Conference*, Goteborg, Sweden.

[Cadoz *et al.*, 1993] Cadoz, C., Luciani, A. et Florens, J.-L. (1993). CORDIS-ANIMA : a Modeling and Simulation System for Sound and Image Synthesis - The General Formalism. *Computer Music Journal*, 17(1).

[Camurri, 1990] Camurri, A. (1990). On the Role of Artificial Intelligence in Music Research. *Interface, Journal of New Music Research*, 19(2-3):219–248.

[Castagne et Cadoz, 2002a] Castagne, N. et Cadoz, C. (2002a). GENESIS : A Friendly Musician-Oriented Environment for Mass-Interaction Physical Modeling. *In Proceedings of the International Computer Music Conference*, Goteborg, Sweden.

[Castagne et Cadoz, 2002b] Castagne, N. et Cadoz, C. (2002b). L'environnement GENESIS : créer avec les modèes physiques masse-interaction. *In Actes des Journées d'Informatique Musicale, 9ème édition*, Marseille, FRance.

[Castagne et Cadoz, 2004] Castagne, N. et Cadoz, C. (2004). The mass-interaction scheme as a musical language : the GENESIS environment. *In Proceedings of ISMA*, Nara, Japan.

[Chadel, 2000] Chadel, J. (2000). Interactions Piano / Machine dans Traiettoria...deviata de Marco Stroppa. Mémoire de DEA, Université Marc Bloch de Strasbourg.

[Charpentier et Stella, 1986] Charpentier, F. et Stella, M. (1986). Diphone Synthesis Using an Overlap-Add Technique for Speech Waveforms Concatenation. *In ICASSP*.

[Chazal, 1995] Chazal, G. (1995). *Le miroir automate. Introduction à une philosophie de l'informatique*. Champ Vallon.

[Chouvel *et al.*, 2007] Chouvel, J.-M., Bresson, J. et Agon, C. (2007). L'analyse musicale différencielle : principes, représentation et application à la structuralisation des entités musicales pour la musique électroacoustique. *In EMS-07 Conference*, Leicester, Great Britain. Electroacoustic Music Studies Network.

[Chowning, 1973] Chowning, J. M. (1973). The synthesis of complex audio spectra by means of frequency modulation. *Journal of the Audio Engineering Society*, 21(7).

[Coduys et Ferry, 2004] Coduys, T. et Ferry, G. (2004). Iannix - Aesthetical/Symbolic Visualisations for Hypermedia Composition. *In Proceedings of Sound and Music Computing Conference - SMC'04*, Paris, France. IRCAM.

[Coduys *et al.*, 2003] Coduys, T., Lefevre, A. et Pape, G. (2003). Iannix. *In Actes des Journées d'Informatique Musicale*, Montbeliard, France.

[Cohen-Levinas, 1993] Cohen-Levinas, D. (1993). Entretien avec Marco Stroppa. *In Les cahiers de l'Ircam (3) - La Composition Assistée par Ordinateur*. Ircam - Centre G. Pompidou.

[Dannenberg, 1989] Dannenberg, R. B. (1989). The Canon Score Language. *Computer Music Journal*, 13(1).

[Dannenberg, 1997] Dannenberg, R. B. (1997). Machine Tongues XIX : Nyquist, a Language for Composition and Sound Synthesis. *Computer Music Journal*, 21(3).

[Dannenberg *et al.*, 1997] Dannenberg, R. B., Desain, P. et Honing, H. (1997). Programming Language Design for Music. *In* Roads, C., Pope, S. T., Piccialli, A. et DePoli, G., éditeurs : *Musical Signal Processing*. Swets and Zeitlinger.

[Dannenberg *et al.*, 1986] Dannenberg, R. B., McAvinney, P. et Rubine, D. (1986). Arctic : A Functional Language for Real-Time Systems. *Computer Music Journal*, 10(4).

[De Poli *et al.*, 1991] De Poli, G., Piccialli, A. et Roads, C., éditeurs (1991). *Representations of Musical Signals*. MIT Press.

[Depalle et Poirot, 1991] Depalle, P. et Poirot, G. (1991). A modular system for analysis, processing and synthesis of sound signals. *In Proceedings of the International Computer Music Conference*, San Francisco, USA.

[Depalle et Rodet, 1993] Depalle, P. et Rodet, X. (1993). De la voix aux instruments. *In Les cahiers de l'Ircam (2) - La synthèse sonore*. Ircam - Centre G. Pompidou.

[Desain et Honing, 1992] Desain, P. et Honing, H. (1992). Time Functions Function Best as Functions of Multiple Times. *Computer Music Journal*, 16(2).

[Desainte-Catherine, 2004] Desainte-Catherine, M. (2004). Modèles temporels pour la synthèse musicale. *In* Pachet, F. et Briot, J.-P., éditeurs : *Informatique musicale*. Hermes Science publications.

[Dolson, 1986] Dolson, M. (1986). The Phase Vocoder : A Tutorial. *Computer Music Journal*, 10(4).

[Dolson, 1989] Dolson, M. (1989). Fourier-Transform-Based Timbral Manipulations. *In* Mathews, M. V. et Pierce, J. R., éditeurs : *Current Directions in Computer Music Research*. MIT press.

[Doval, 2004] Doval, B. (2004). Méthodes d'analyse du signal musical. *In* Pachet, F. et Briot, J.-P., éditeurs : *Informatique musicale*. Hermes Science publications.

[Doval et Rodet, 1991] Doval, B. et Rodet, X. (1991). Estimation of Fundamental Frequency of Musical Sound Signals. *In Proceedings IEEE-ICASSP 91*, Toronto, Canada.

[Dufourt, 1991] Dufourt, H. (1991). *Musique, pouvoir, écriture*. Christian Bourgeois éditions.

[Ebbeke, 1990] Ebbeke, K. (1990). La vue et l'ouïe – Problèmatique des partitions dans la musique électro-acoustique. *Contrechamps*, 11:70–79.

[Eckel, 1992] Eckel, G. (1992). Manipulation of Sound Signals Based on Graphical Representation - A Musical Point of View. *In Proceedings of the International Workshop on Models and Representations of Musical Signals*, Capri, Italia.

[Eckel, 1993] Eckel, G. (1993). La maîtrise de la syntèse sonore. *In Les cahiers de l'Ircam (2) - La synthèse sonore*. Ircam - Centre G. Pompidou.

[Eckel et Gonzalez-Arroyo, 1994] Eckel, G. et Gonzalez-Arroyo, R. (1994). Musically Salient Control Abstractions for Sound Synthesis. *In Proceedings of the International Computer Music Conference*, Aarhus, Denmark.

[Eckel *et al.*, 1995] Eckel, G., Iovino, F. et Caussé, R. (1995). Sound Synthesis by Physical Modelling with Modalys, the Successor of the Modal Synthesis System Mosaïc. *In Proceedings of the International Symposium on Music Acoustics*, Le Normont, France.

[Ellis *et al.*, 2005] Ellis, N., Bensoam, J. et Caussé, R. (2005). Modalys Demonstration. *In Proceedings of the International Computer Music Conference*, Barcelona, Spain.

[Fitz et Haken, 1996] Fitz, K. et Haken, L. (1996). Sinusoidal Modelling and Manipulation Using Lemur. *Computer Music Journal*, 20(4).

[Florens et Cadoz, 1991] Florens, J. et Cadoz, C. (1991). The physical model : modeling and simulating the instrumental universe. *In* De Poli, G., Picciali, A. et Roads, C., éditeurs : *Representation of Musical Signals*. MIT Press.

[Gabor, 1946] Gabor, D. (1946). Theory of Communication. *Journal of Institue of Electrical Engineers*, 93.

[Gabor, 1947] Gabor, D. (1947). Acoustical quanta and the theory of hearing. *Nature*, 159(4044).

[Gabriel *et al.*, 1991] Gabriel, R. P., White, J. L. et Bobrow, D. G. (1991). CLOS : Integration Object-oriented and Functional Programming. *Communications of the ACM*, 34(9).

[Garnett, 1991] Garnett, G. (1991). Music, Signals and Representations : a Survey. *In* DePoli, G., Piccialli, A. et Roads, C., éditeurs : *Representation of Musical Signals*. MIT Press.

[Girard *et al.*, 1989] Girard, J.-Y., Taylor, P. et Lafont, Y. (1989). *Proofs and Types*. Cambridge University Press.

[Grey, 1975] Grey, J. (1975). *An Exploration of Musical Timbre*. Thèse de doctorat, Stanford University.

[Grisey, 1987] Grisey, G. (1987). Tempus ex Machina : A composer's reflexions on musical time. *Contemporary Music Review*, 2.

[Guédy, 2006] Guédy, F. (2006). Le traitement du son en pédagogie musicale. *L'Inouï*, 2. Ircam - Léo Scheer.

[Guédy *et al.*, 2007] Guédy, F., Bresson, J. et Assayag, G. (2007). Musique Lab 2 - Un environnement d'aide à la pédagogie musicale. *In Actes de Journées d'informatique Musicale – JIM'07*, Lyon, France.

[Haddad, 1999] Haddad, K. (1999). OpenMusic OM2CSound. *Ircam Software Documentation*.

[Haddad, 2006] Haddad, K. (2006). TimeSculpt in OpenMusic. *In* Agon, C., Assayag, G. et Bresson, J., éditeurs : *The OM Composer's Book. 1*. Editions Delatour / IRCAM.

[Hanappe, 1999] Hanappe, P. (1999). *Design and implementation of an integrated environment for music composition and synthesis*. Thèse de doctorat, Université Pierre et marie Curie - Paris VI, France.

[Hanappe et Assayag, 1998] Hanappe, P. et Assayag, G. (1998). Intégration des représenta-tions temps/fréquence et des représentations musicales symboliques. *In* Chemillier, M. et Pachet, F., éditeurs : *Recherches et applications en informatique musicale*. Hermes.

[Hiller, 1969] Hiller, L. (1969). Revised MUSICOMP manual. Rapport technique 13, University of Illinois Experimental Music Studio.

[Hiller et Ruiz, 1971] Hiller, L. et Ruiz, P. (1971). Synthesizing sounds by solving the wave equation for vibrating objects. *Journal of the Audio Engineering Society*, 19.

[Honing, 1993] Honing, H. (1993). Issues in the representation of time and structure in music. *Contemporary Music Review*, 9.

[Jaffe et Smith, 1983] Jaffe, D. A. et Smith, J. O. (1983). Extensions of the Karplus-Strong plucked string algorithm. *Computer Music Journal*, 7(2).

[Karplus et Strong, 1983] Karplus, K. et Strong, A. (1983). Digital Sythesis of plucked-strings and drum timbres. *Computer Music Journal*, 7(2).

[Kiczales *et al.*, 1991] Kiczales, G., des Rivières, J. et Bobrow, D. G. (1991). *The Art of the Metaobject Protocol*. MIT Press.

[Klingbeil, 2005] Klingbeil, M. (2005). Software for Spectral Analysis, Editing and Synthesis. *In Proceedings of the International Computer Music Conference*, Barcelona, Spain.

[Kronland-Martinet et Grossman, 1991] Kronland-Martinet, R. et Grossman, A. (1991). Application of the Time-Frequency and Time-Scale Methods (Wavelet Transforms) to the Analysis, Synthesis and Transformation of Natural Sounds. *In* DePoli, G., Picciali, A. et Roads, C., éditeurs : *Representation of Musical Signals*. MIT Press.

[Kronland-Martinet *et al.*, 1987] Kronland-Martinet, R., Morlet, J. et Grossman, A. (1987). Analysis of Sound Patterns through Wavelet Transforms. *International Journal of Pattern Recognition and Artificial Intelligence*, 1(2).

[Kuuskankare et Laurson, 2006] Kuuskankare, M. et Laurson, M. (2006). Expressive Notation Package. *Computer Music Journal*, 30(4):67–79.

[Laske, 1981] Laske, O. (1981). Composition Theory in Koenig's Project One and Project Two. *Computer Music Journal*, 5(4).

[Laurson, 1996] Laurson, M. (1996). Patchwork : A Visual Programming Language and some musical Applications. *Studia Musica*, 6.

[Laurson et Duthen, 1989] Laurson, M. et Duthen, J. (1989). Patchwork, a Graphic Language in PreForm. *In Proceedings of the International Computer Music Conference*, Ohio State University, USA.

[Laurson et Kuuskankare, 2002] Laurson, M. et Kuuskankare, M. (2002). PWGL : A Novel Visual Language Based on Common Lisp, CLOS, and OpenGL. *In Proceedings of the International Computer Music Conference*, Gothenburg, Sweden.

[Laurson *et al.*, 2005] Laurson, M., Norilo, V. et Kuuskankare, M. (2005). PWGLSynth : A Visual Synthesis Language for Virtual Instrument Design and Control. *Computer Music Journal*, 29(3).

[LeBrun, 1979] LeBrun, M. (1979). Digital waveshaping synthesis. *Journal of the Audio Engineering Society*, 27(4).

[Leman, 1993] Leman, M. (1993). Symbolic and subsymbolic description of music. *In* Haus, G., éditeur : *Music Processing*. Oxford University Press.

[Lesbros, 1996] Lesbros, V. (1996). From Images to Sounds, A Dual Representation. *Computer Music Journal*, 20(3).

[Letz, 2004] Letz, S. (2004). Les normes MIDI et MIDIFiles. *In* Pachet, F. et Briot, J.-P., éditeurs : *Informatique musicale*. Hermes Science publications.

[Loy, 1985] Loy, G. (1985). Musicians make a standard : the MIDI phenomenon. *Computer Music Journal*, 9(4):8–26.

[Loy, 1989] Loy, G. (1989). Composing with Computers, a Survey of Some Compositional Formalisms and Music Programming Languages. *In* Mathews, M. V. et Pierce, J. R., éditeurs : *Current Directions in Computer Music Research*. MIT press.

[Makhoul, 1975] Makhoul, J. (1975). Linear Prediction : A tutorial review. *Proceedings of the IEEE*, 63(4).

[Malt, 2000] Malt, M. (2000). *Les mathématiques et la composition assistée par ordinateur (concepts, outils et modèles)*. Thèse de doctorat, Ecoles des Hautes Etudes en Sciences Sociales, Paris, France.

[Malt, 2003] Malt, M. (2003). Concepts et modèles, de l'imaginaire à l'écriture dans la composition assistée par ordinateur. *In Séminaire postdoctoral Musique, Instruments, Machines*, Paris, France.

[Manoury, 1990] Manoury, P. (1990). La note et le son : un carnet de bord. *Contrechamps*, 11:151–164.

[Marchand, 2000] Marchand, S. (2000). *Modélisation informatique du son musical (analyse, transformation, synthèse) / Sound Models for Computer Music (anaysis, transformation, and synthesis of musical sound)*. Thèse de doctorat, Université Bordeaux I, France.

[Marchand, 2007] Marchand, S. (2007). ReSpect : A Free Software Library for Spectral Sound Synthesis. *In Actes des Journées d'Informatique Musicale – JIM'07*, Lyon, France.

[Mathews et Kohut, 1973] Mathews, M. et Kohut, J. (1973). Electronic simulation of violin resonances. *Journal of the Acoustical Society of America*, 53(6).

[Mathews et Pierce, 1980] Mathews, M. et Pierce, J. (1980). Harmony and Nonharmonic Partials. Rapport technique 28/80, IRCAM.

[Mathews, 1969] Mathews, M. V., éditeur (1969). *The Technology of Computer Music*. MIT Press.

[Mathews et Moore, 1970] Mathews, M. V. et Moore, F. R. (1970). GROOVE – a program to compose, store, and edit functions of time. *Communications of the ACM*, 13(12).

[Mazzola *et al.*, 2004] Mazzola, G., Noll, T. et Lluis-Puebla, E., éditeurs (2004). *Perspectives in Mathematical and Computational Music Theory*. Electronic Publishing Osnabrück.

[McAdams *et al.*, 1995] McAdams, S., Winsberg, S., Donnadieu, S., De Soete, G. et Krimphoff, J. (1995). Perceptual Scaling of Synthesized Musical Timbres : Common Dimensions, Specificities, and Latent Subject Classes. *Psychological Research*, 58.

[McAulay et Quatieri, 1986] McAulay, R. J. et Quatieri, T. F. (1986). Speech analysis/synthesis based on a sinusoidal representation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(4).

[MIDI Manufacturers Association, 2001] MIDI Manufacturers Association (2001). *Complete MIDI 1.0 Detailed Specification v96.1 (second edition)*. MIDI Manufacturers Association.

[Miranda, 1995] Miranda, E. R. (1995). An Artificial Intelligence Approach to Sound Design. *Computer Music Journal*, 19(2).

[Moore, 1990] Moore, F. R. (1990). *Elements of Computer Music*. Prentice Hall.

[Moorer, 1977] Moorer, J. A. (1977). Signal processing aspects of computer music : a survey. *Proceedings of the IEEE*, 65(8).

[Morrison et Adrien, 1993] Morrison, J. D. et Adrien, J.-M. (1993). MOSAIC : A Framework for Modal Synthesis. *Computer Music Journal*, 17(1).

[Noll *et al.*, 2006] Noll, T., Andreatta, M. et Agon, C. (2006). Computer-aided transformational analysis with tone sieves. *In Proceedings of Sound and Music Computing - SMC'06*, Marseille, France.

[Nouno, 2008] Nouno, G. (2008). Some Considerations on Brian Ferneyhough's Musical Language Through His Use of CAC – Spatialization As a Musical Parameter. *In* Bresson, J., Agon, C. et Assayag, G., éditeurs : *The OM Composer's Book. 2*. Editions Delatour / IRCAM (à paraître).

[Orlarey *et al.*, 1997] Orlarey, Y., Fober, D. et Letz, S. (1997). L'environnement de composition musicale ELODY. *In Actes de Journées d'Informatique Musicale*, Lyon, France.

[Orlarey *et al.*, 2004] Orlarey, Y., Fober, D. et Letz, S. (2004). Syntactical and Semantical Aspects of Faust. *Soft Computing*, 8(9).

[Orlarey et Lequay, 1989] Orlarey, Y. et Lequay, H. (1989). Midishare. *In Proceedings of the International Computer Music Conference*, Ohio State University, USA.

[Palamidessi et Valencia, 2001] Palamidessi, C. et Valencia, F. (2001). A Temporal Concurrent Constraint Programming Calculus. *In Proceedings of the 7^{th} International Conference on Principles and Practices of Constraint Programming.*

[Peeters, 1998] Peeters, G. (1998). Analyse et synthèse des sons musicaux par la methode PSOLA. *In Actes des Journées d'Informatique Musicale*, Marseille, France. LMA.

[Peeters, 2003] Peeters, G. (2003). Automatic Classification of Large Musical Instrument Databases Using Hierarchical Classifiers with Inertia Ratio Maximization. *In AES 115^{th} Convention*, New York, USA.

[Piché et Burton, 1998] Piché, J. et Burton, A. (1998). Cecilia : a Production Interface to Csound. *Computer Music Journal*, 22(2).

[Polfreman, 2002] Polfreman, R. (2002). Modalys-ER for OpenMusic (MfOM) : Virtual Instruments and Virtual Musicians. *Organised Sound*, 7(3).

[Pope, 1991] Pope, S. T., éditeur (1991). *The Well Tempered Object (Musical Applications of Object-Oriented Software Technology)*. MIT Press.

[Portnoff, 1980] Portnoff, M. R. (1980). Time-frequency representation of digital signals and systems based on short-time Fourier Transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(8).

[Pottier, 2008] Pottier, L. (2008). The control of microsound synthesis – *Transiciones de fase* by Manuel Rocha Iturbide. *In* Bresson, J., Agon, C. et Assayag, G., éditeurs : *The OM Composer's Book. 2*. Editions Delatour / IRCAM (à paraître).

[Puckette, 1991] Puckette, M. (1991). Combining Event and Signal Processing in the MAX Graphical Programming Environment. *Computer Music Journal*, 15(3).

[Puckette, 1996] Puckette, M. (1996). Pure Data : another integrated computer music environment. *In Proceedings of the Second Intercollege Computer Music Concerts*, Tachikawa, Japan.

[Puckette, 2002] Puckette, M. (2002). Using Pd as a score language. *In Proceedings of the International Computer Music Conference*, Göteborg, Sweden.

[Puckette, 2004] Puckette, M. (2004). A divide between 'compositional' and 'performative' aspects of Pd. *In 1st International Pd Convention*, Graz, Austria.

[Puig *et al.*, 2005] Puig, V., Guédy, F., Fingerhut, M., Serrière, F., Bresson, J. et Zeller, O. (2005). Musique Lab 2 : A Three-Level Approach for Music Education at School. *In Proceedings of the International Computer Music Conference*, Barcelona, Spain.

[Riotte et Mesnage, 2006] Riotte, A. et Mesnage, M. (2006). *Formalismes et modèles musicaux. 2 Volumes.* Editions Delatour / Ircam.

[Risset, 1977] Risset, J.-C. (1977). Musique, calcul secret ? *Critique*, (359).

[Risset, 1991] Risset, J.-C. (1991). Timbre Analysis by Synthesis. *In* De Poli, G., Piccialli, A. et Roads, C., éditeurs : *Representation of Musical Signals*. MIT Press.

[Risset, 1993] Risset, J.-C. (1993). Synthèse et materiau musical. *In Les cahiers de l'Ircam (2) - La synthèse sonore*. Ircam - Centre G. Pompidou.

[Risset, 1996] Risset, J.-C. (1996). Composing sounds, bridging gaps - the musical role of the computer in my music. *In* de la Motte-Haber, H. et Frisius, R., éditeurs : *Musik und Technik*. Schott, Mainz.

[Risset et Mathews, 1969] Risset, J.-C. et Mathews, M. (1969). Analysis of instrument tones. *Physics Today*, 22(2).

[Roads, 1979] Roads, C. (1979). A tutorial on non linear distorsion or waveshaping synthesis. *Computer Music Journal*, 3(2).

[Roads, 1985] Roads, C. (1985). Granular synthesis of sound. *In* Roads, C. et Strawn, G., éditeurs : *Foundations of Computer Music*. MIT Press.

[Roads, 1991] Roads, C. (1991). Asynchronous Granular Synthesis. *In* DePoli, G., Piccialli, A. et Roads, C., éditeurs : *Representation of Musical Signals*. MIT Press.

[Roads, 1993] Roads, C. (1993). Initiation à la synthèse par modèles physiques. *In Les cahiers de l'Ircam (2) - La synthèse sonore*. Ircam - Centre G. Pompidou.

[Roads, 1996] Roads, C. (1996). *The Computer Music Tutorial*. MIT Press.

[Roads, 2002] Roads, C. (2002). *Microsound*. MIT Press.

[Roads et Strawn, 1985] Roads, C. et Strawn, J., éditeurs (1985). *Foundations of Computer Music*. MIT Press.

[Röbel, 2003] Röbel, A. (2003). Transient detection and preservation in the phase vocoder. *In Proceedings of the International Computer Music Conference*, Singapore.

[Rodet et Cointe, 1984] Rodet, X. et Cointe, P. (1984). Formes : Compostion and Scheduling of Processes. *Computer Music Journal*, 8(3).

[Rodet et Depalle, 1992] Rodet, X. et Depalle, P. (1992). Spectral Enveloppes and Inverse FFT Synthesis. *In 93rd Audio Engineering Society Convention*, San Francisco, USA.

[Rodet et Lefevre, 1997] Rodet, X. et Lefevre, A. (1997). The Diphone Program : New Features, New synthesis Methods and Experience of Musical Use. *In Proceedings of the International Computer Music Conference*, Thessaloniki, Greece.

[Rodet *et al.*, 1984] Rodet, X., Potard, Y. et Barrière, J.-B. (1984). The CHANT project : From the synthesis of the singing voice to synthesis in general. *Computer Music Journal*, 8(3).

[Rodet *et al.*, 1985] Rodet, X., Potard, Y. et Barrière, J.-B. (1985). CHANT - De la synthèse de la voix chantée à la synthèse en général. Rapport technique 35, IRCAM.

[Ronfard, 2004] Ronfard, R. (2004). MPEG, une norme pour la compression, la structuration et la description du son. *In* Pachet, F. et Briot, J.-P., éditeurs : *Informatique musicale*. Hermes Science publications.

[Rueda *et al.*, 2001] Rueda, C., Alvarez, G., Quesada, L. O., Tamura, G., Valencia, F. D., Díaz, J. F. et Assayag, G. (2001). Integrating Constraints and Concurrent Objects in Musical Applications : A Calculus and its Visual Language. *Constraints*, 6(1).

[Rueda et Valencia, 2005] Rueda, C. et Valencia, F. D. (2005). A Temporal Concurrent Constraint Calculus as an Audio Processing Framework. *In Proceedings of the Sound and Music Computing Conference SMC'05*, Salerno, Italia.

[Schaeffer, 1966] Schaeffer, P. (1966). *Traité des Objets Musicaux*. Editions du Seuil.

[Schnell *et al.*, 2005] Schnell, N., Borghesi, R., Schwartz, D., Bevilacqua, F. et Muller, R. (2005). FTM – Complex Data Structures for Max. *In Proceedings of the International Computer Music Conference*, Barcelona, Spain.

[Schwartz, 2004] Schwartz, D. (2004). *Data-Driven Concatenative Sound Synthesis*. Thèse de doctorat, Université Pierre et Marie Curie - Paris 6.

[Schwartz et Wright, 2000] Schwartz, D. et Wright, M. (2000). Extensions and Applications of the SDIF Sound Description Interchange Format. *In Proceedings of the International Computer Music Conference*, Berlin, Germany.

[Serra et Smith, 1990] Serra, X. et Smith, J. O. (1990). Spectral modeling synthesis : A sound analysis/synthesis systembased on a deterministic plus stochastic decomposition. *Computer Music Journal*, 14(4).

[Siskind et McAllester, 1993] Siskind, J. M. et McAllester, D. A. (1993). Nondeterministic lisp as a substrate for constraint logic programming. *In Proceedings of the 11th National Conference on Artificial Intelligence*. AAAI Press.

[Smith, 1992] Smith, J. O. (1992). The Second-Order Digital Waveguide Oscillator. *In Proceedings of the International Computer Music Conference*, San Jose State University, USA.

[Smith, 1993] Smith, J. O. (1993). Observations sur l'histoire de la synthèse numérique du son. *In Les cahiers de l'Ircam (2) - La synthèse sonore*. Ircam - Centre G. Pompidou.

[Smith, 1972] Smith, L. (1972). SCORE–A Musician's Approach to Computer Music. *Journal of the Audio Engineering Society*, 20.

[Steele, 1990] Steele, G. L. (1990). *Common Lisp, the Language. Second edition*. Digital Press.

[Stockhausen, 1957] Stockhausen, K. (1957). ...wie die Zeit vergeht... *Di Reihe*, 3.

[Stockhausen, 1988a] Stockhausen, K. (1988a). ... comment passe le temps ... *Contrechamps*, 9.

[Stockhausen, 1988b] Stockhausen, K. (1988b). Musique électronique et musique instrumentale. *Contrechamps*, 9.

[Stockhausen, 1988c] Stockhausen, K. (1988c). Situation de l'artisanat (critères de la musique ponctuelle). *Contrechamps*, 9.

[Stroppa, 1999] Stroppa, M. (1999). Live electronics or... live music ? towards a critique of interaction. *Aesthetics of Live Electronic Music - Comtemporary Music Review*, 18(3).

[Stroppa, 2000] Stroppa, M. (2000). Paradigms for the high level musical control of digital signal processing. *In Proceedings of the Digital Audio Effects Conference DAFX'00*, Verona, Italia.

[Stroppa et Duthen, 1990] Stroppa, M. et Duthen, J. (1990). Une representation de structures temporelles par synchronisation de pivots. *In Colloque Musique et Assistance Informatique*, Marseille, France.

[Stroppa *et al.*, 2002] Stroppa, M., Lemouton, S. et Agon, C. (2002). OmChroma ; vers une formalisation compositionnelle des processus de synthèse sonore. *In Actes des Journées d'Informatique Musicale, 9ème édition*, Marseille, France.

[Tardieu, 2004] Tardieu, D. (2004). Synthèse et transformation sonore par descripteurs de haut-niveau. Mémoire de DEA, ATIAM, Université Aix Marseille II.

[Taube, 1991] Taube, H. (1991). Common Music : A Music Composition Language in Common Lisp and CLOS. *Computer Music Journal*, 15(2).

[Truax, 1988] Truax, B. (1988). Real-time granular synthesis with a digital signal processing computer. *Computer Music Journal*, 12(2).

[Truchet *et al.*, 2003] Truchet, C., Assayag, G. et Codognet, P. (2003). OMClouds, petits nuages de contraintes dans OpenMusic. *In Actes de Journées d'Informatique Musicale*, Montbeliard, France.

[Tutschku, 1998] Tutschku, H. (1998). OpenMusic OM-AS Library. *Ircam Software Documentation.*

[Tutschku, 2003] Tutschku, H. (2003). L'application des paramètres compositionnels au traitement sonore dans la composition mixte et électroacoustique. *In* Baboni-Schilingi, J., éditeur : *PRISMA 01.* Euresis Edizioni, Milano, Italia.

[Tutschku, 2008] Tutschku, H. (2008). The use of OpenMusic for sound composition. *In* Bresson, J., Agon, C. et Assayag, G., éditeurs : *The OM Composer's Book. 2.* Editions Delatour / IRCAM (à paraître).

[Tüzün, 2008] Tüzün, T. (2008). Maquette as data structure and synthesis agent in *Metathesis*. *In* Bresson, J., Agon, C. et Assayag, G., éditeurs : *The OM Composer's Book. 2.* Editions Delatour / IRCAM (à paraître).

[Veitl, 2007] Veitl, A. (2007). Notation écrite et musique contemporaine : quelles grandes caractéristiques des technologies numériques d'écriture musicale ? *In Actes de Journées d'informatique Musicale – JIM'07*, Lyon, France.

[Vercoe et Scheirer, 1998] Vercoe, B. L. et Scheirer, E. D. (1998). Structured Audio : Creation, Transmission, and Rendering of Parametric Sound Representations. *Proceedings of the IEEE*, 86(5).

[Wessel, 1978] Wessel, D. L. (1978). Timbre Space as a Musical Control Structure. *Computer Music Journal*, 3(2).

[Wright *et al.*, 1999] Wright, M., Chaudhary, A., Freed, A., Khouri, S. et Wessel, D. (1999). Audio applications of the Sound Description Interchange Format. *In 107th Audio Engineering Society Convention*, New York, USA.

[Wright *et al.*, 1998] Wright, M., Chaudhary, A., Freed, A., Wessel, D., Rodet, X., Virolle, D., Woehrmann, R. et Serra, X. (1998). New applications of the Sound Description Interchange Format. *In Proceedings of the International Computer Music Conference*, Ann Arbor, USA.

[Wright et Freed, 1997] Wright, M. et Freed, A. (1997). Open SoundControl : A New Protocol for Communicating with Sound Synthesizers. *In Proceedings of the International Computer Music Conference*, Thessaloniki, Greece.

[Wright *et al.*, 2003] Wright, M., Freed, A. et Momeni, A. (2003). Open SoundControl : State of the Art 2003. *In Proceedings of NIME-03 International Conference on New Interfaces for Musical Expression*, Montreal, Canada.

[Xenakis, 1981] Xenakis, I. (1981). *Musiques Formelles*. Stock Musique.