Composition assistée par ordinateur : techniques et outils de programmation visuelle pour la création musicale

Jean Bresson

COMPOSITION ASSISTÉE PAR ORDINATEUR : TECHNIQUES ET OUTILS DE PROGRAMMATION VISUELLE POUR LA CRÉATION MUSICALE

Mémoire d'Habilitation à Diriger des Recherches

Université Pierre et Marie Curie
Sorbonne Universités
UFR 919 – Faculté d'ingénierie

Jean Bresson
UMR « Sciences et Technologies de la Musique et du Son »
IRCAM-CNRS-UPMC

Soutenue le 4 mai 2017

Jury :

– Philippe Codognet (Université Pierre et Marie Curie, Paris) – Examinateur
– Pierre Cointe (Université de Nantes / École des Mines de Nantes) – Rapporteur
– Myriam Desainte-Catherine (Université de Bordeaux) – Rapporteure
– José Manuel Iñesta (Universidad de Alicante) – Examinateur
– Marcelo Wanderley (McGill University, Montréal) – Rapporteur
– Gérard Assayag (IRCAM- STMS) – Invité

Historique des révisions :

03/11/2017	p.23	Réécriture : *de description → liées aux descriptions*
19/11/2017	p.59	Correction/typo : ~~*décrit*~~
08/05/2018	p.43	Typo, note de bas de page 7 : *les → le*
04/08/2018	p.67	Typo : *des* ~~*des*~~
16/08/2018	pp.117-127	Bibliographie (corrections mineures)
07/03/2022	pp.24, 88, 92-95, 98	OM7 → OM# et note de révision (p.24)
	p.59	Correction : *implications*
	p.2	Historique des révisions (ce tableau)

TABLE DES MATIÈRES

AVANT-PROPOS

Ce mémoire décrit mes principaux travaux et recherches dans le domaine de l'informatique musicale et de la composition assistée par ordinateur. J'ai essayé notamment d'en souligner le lien avec différentes disciplines : langages de programmation visuelle, création et recherche musicale, musicologie, traitement du signal et son spatialisé, IHM, représentation et visualisation des données, et plus récemment avec les techniques et formalismes calculatoires de l'interaction. Ancré au sein de l'équipe Représentations Musicales de l'IRCAM (Institut de Recherche et Coordination Acoustique/Musique) [1] mon parcours implique également un certain nombre de collaborations et de partenariats. Un accent est mis dans ce mémoire sur les projets auxquels j'ai contribué à titre personnel, mais aussi en tant qu'encadrant de travaux universitaires ou en tant que coordinateur. La liste des étudiants et des travaux académiques que j'ai supervisés est donnée en annexe de ce document. Ceux-ci sont signalés par le symbole [⋆] dans le manuscrit. En annexe est aussi proposée une brève description des différents chapitres constituant l'ouvrage *The OM Composer's Book 3* que j'ai publié en 2016, et auquel je ferai référence dans le manuscrit via le symbole [📖]. Mes publications sont listées à la fin de chaque chapitre, séparément de la bibliographie principale. Leurs citations sont également signalées par le symbole ✎.

1. Unité Mixte de Recherche « Sciences et Technologies de la Musique et du Son » : IRCAM / CNRS / UPMC Sorbonne Universités.

ABSTRACT

*This document is a report on my research activity in **computer-aided music composition**, carried out in the IRCAM "Science and Technology of Music and Sound" (STMS) laboratory.*

*The first chapters present a general perspective on this field of research, focusing on the OpenMusic environment: a visual programming language developed at IRCAM, which has been the main context and focus of my work during the last decade. Computer-aided composition is situated relative to other research in the fields of "**domain-specific languages**" (DSLs) for music and **visual programming**. Included is a discussion of the composer as "end-user programmer" in the context of visual programming environments.*

*Chapter 3 contains the presentation of several works carried out in OpenMusic. A first part of these works relate to the representation of **time structures** in the environment: time structures from a high-level compositional perspective, or considered more locally in the context of rhythm quantification and processing. A section is then dedicated to **music analysis and modelling**, and we conclude the chapter with a presentation of projects and collaborations in the domain of computer-aided orchestration and in the field of visual arts.*

*Chapter 4 describes achievements and projects related to the **synthesis and processing of sounds and signals**, that have been carried out in the context or as a continuation of my PhD (2007). In chapter 5, the discussion and tools are extended to the issue of organizing time structures while working with sound synthesis, hence bringing into play some additional questions about symbolic/discrete musical structures vs. continuous signals, and the compositional specification of sound properties and morphologies extended as timed evolutions. We conclude and illustrate this discussion with the description of a concrete application in Marco Stroppa's opera* Re Orso*.*

*Chapter 6 is dedicated to **spatialization**, where the compositional and signal processing tools are applied to the control of virtual acoustic or spatial properties of sounds. This chapter also introduces the combination of offline compositional software with real-time DSP or media frameworks, which arises when bringing together interactive control, formal compositional specification, and audio feedback or rendering.*

*These topics are the basis of the discussion in chapter 7, which focuses on my most recent works on **interactive compositional systems**. We begin with the specification and development of a reactive extension for OpenMusic programs. This is followed by the presentation of several steps towards structured and expressive communication between musical and media frameworks, and by the discussion of an integrated system for the computation, scheduling and rendering of dynamic musical structures. We conclude by drawing perspectives on the application of these works as a new bridge between signal and symbolic processing in computer music.*

Research works that occurred under my supervision are listed in the Appendix and indicated by the symbol [⋆] where they occur in the text. The document also frequently cites chapters of the OM Composer's Book *series as examples of compositional application of the works presented, particularly volume 3, which I published in 2016. The chapters of this volume are listed and briefly described in the Appendix as well, and are referred to in the document using the symbol 🕮. My publications are given at the end of each corresponding chapter, separate from the main bibliography. Their citations in the text are marked with the symbol ✎.*

1

INTRODUCTION

La musique a souvent motivé des travaux et innovations scientifiques dans les domaines de l'acoustique, du traitement du signal, des sciences cognitives ou des mathématiques, mais également dans des champs de recherche plus inattendus comme les langages de programmation, la recherche opérationnelle, l'apprentissage, ou les interfaces homme-machine. Parmi les disciplines qui constituent l'« informatique musicale », la *composition assistée par ordinateur* s'attache à l'étude des outils, langages et formalismes de programmation, dans l'objectif de fournir aux compositeurs et musiciens professionnels des outils de représentation et de calcul pour la production et le traitement des données musicales (partitions, structures harmoniques, rythmes, sons, etc.). Comme nous essaierons de le montrer, ce champ de recherche innove par des réalisations singulières dans les domaines du *end-user programming* et de la programmation visuelle, et constitue un terrain riche pour la recherche scientifique et musicale autour de questions telles que la dualité symbole/signal ou encore les formalismes et structures temporelles en informatique et en musique.

UN COUP D'ŒIL EN ARRIÈRE : ORIGINES DE LA CAO

Les travaux pionniers de Max Mathews sur la synthèse sonore numérique dans les années 50 sont généralement considérés comme l'origine de l'informatique musicale (Mathews, 1963). Ils ont en effet fondé dans ce domaine des principes encore suivis de nos jours, et initié des directions de recherche plus que jamais d'actualité. Un point remarquable de ces travaux fut notamment l'utilisation de langages de programmation à des fins artistiques : une anticipation d'un principe que l'on appelle aujourd'hui *end-user programming* (Burnett et Scaffidi, 2014), désignant le fait de permettre à l'utilisateur d'un logiciel (*end-user* : une personne qui n'est pas un programmeur professionnel) de créer ou de modifier lui-même des programmes. [1] À la même époque, inspiré par le développement de l'intelligence artificielle, Lejaren Hiller initiait avec l'ordinateur *Illiac* les premières expérimentations *compositionnelles* dans le domaine — un programme-compositeur capable de produire une œuvre musicale (Hiller et Isaacson, 1959).

La recherche en informatique musicale s'est ensuite majoritairement concentrée sur le traitement numérique du signal, qui faisait des progrès spectaculaires et laissait entrevoir des possibilités inédites dans le domaine de la création sonore. C'est en Europe, où les approches « formelles » de la composition musicale étaient peut-être plus solidement ancrées (bien que

1. Un des exemples les plus communs de *end-user programming* est par exemple l'usage des feuilles de calcul (ou *spreadsheets*) comme Microsoft Excel.

peu portées alors sur l'informatique) que Pierre Barbaud, Iannis Xenakis, André Riotte et bien d'autres poussèrent plus avant cette idée de composer avec un ordinateur (Barbaud, 1968; Xenakis, 1971; Riotte et Mesnage, 2006). Ces travaux, qui structureront le principe d'aide informatique à la composition, tendent à considérer la musique via des représentations *symboliques*, relativement abstraites du domaine du signal sonore et plus proches de celui des idées. Cette distinction entre les domaines « signal » et « symbolique » dans la recherche et les applications musicales, ainsi décelable dès les premiers balbutiements de la discipline, s'est perpétuée jusqu'à nos jours. Nous y reviendrons plus loin.

Le terme de « composition algorithmique » (Cope, 1996; Nierhaus, 2008) qui s'est popularisé par la suite dénote, de manière parfois un peu floue, toute forme d'application d'algorithmes ou de programmes (qu'ils soient complètement déterministes ou aléatoires) pour produire de la musique. Ce terme évoque généralement, souvent à juste titre, une certaine prise de contrôle de l'ordinateur sur la création : il s'agit de programmer un ordinateur de sorte à lui faire produire de la musique (que ce soit sous forme sonore, ou sous forme de partition) de façon autonome. Cette vision laisse ainsi peu de place à l'interaction, à la décision et au jugement humain ; son renversement sera fondamental à l'établissement de la notion de composition assistée par ordinateur telle que nous la concevons aujourd'hui, se positionnant à un point intermédiaire entre l'application de procédures algorithmiques et un contrôle plus personnel du compositeur.

STRUCTURATION D'UN DOMAINE

La composition assistée par ordinateur (CAO) en tant que discipline autonome s'est essentiellement forgée à l'IRCAM dans les années 80-90, autour du groupe de recherche Crime (Claudy Malherbe, Michèle Castellengo, Gérard Assayag, 1983-86) (Assayag *et al.*, 1985) puis de l'équipe Représentations Musicales (de 1992 à nos jours) (Assayag et Rueda, 1993; Assayag, 2009). Elle met en avant une approche à la fois créative et formelle, basée sur les modèles de calcul et de programmation, mais également sur la représentation et l'interaction avec les structures et la notation musicales (Assayag et Timis, 1986). Depuis la machine-compositeur autonome de Hiller, les systèmes de composition assistée par ordinateur ont ainsi évolué dans le sens d'une plus forte interaction du musicien dans la conception et dans l'exécution des processus musicaux. Cette évolution est bien sûr liée à des évolutions culturelles et esthétiques de l'époque, [2] mais aussi à la recherche et aux avancées technologiques qui l'ont caractérisée : augmentation de la puissance de calcul, développement de l'informatique personnelle, des interfaces utilisateurs et des dispositifs d'entrée/sortie, etc.

La notion de *end-user programming* devient également déterminante avec le développement de langages de *programmation visuelle* (Burnett, 1999) qui marquent une étape importante quant à la dissémination de la CAO et des pratiques musicales faisant appel à la programmation. Dans le chapitre 2 nous présenterons et mettrons nos travaux en perspective dans ce domaine à travers le développement de l'environnement OpenMusic.

2. Le développement de la CAO fut notamment connecté à celui du courant de musique *spectrale* en France, et à la collaboration des chercheurs de l'IRCAM avec des compositeurs tels que Claudy Malherbe, Tristan Murail, Joshua Fineberg, Magnus Lindberg et bien d'autres.

POINT DE DÉPART

Notre contexte de recherche et le point de départ de nos travaux se situent donc dans ce domaine de la composition assistée par ordinateur et des langages de programmation dédiés à la musique. Dans le chapitre 3 nous présenterons dans un premier temps un ensemble de projets réalisés dans ce contexte. Comme nous le verrons, leur particularité tiendra souvent en un équilibre spécifique entre la programmation, les interfaces utilisateurs, et les différentes contraintes liées au domaine d'application visé (la composition, l'analyse musicale, la pédagogie...).

Nous partirons également du constat de « bipolarisations » du domaine autour de plusieurs oppositions structurantes, entre notamment :

- Les approches « extensionnelles » et les approches « intensionnelles », où les structures musicales (et en particulier les séquences temporelles) sont spécifiées respectivement soit de manière descriptive (sous forme de listes d'évènements et de paramètres musicaux), soit suivant une logique procédurale (ou « prescriptive » — sous forme de d'algorithmes, de programmes, ou de diagrammes de flot de données).
- Les approches « signal » et les approches « symboliques », où le signal regroupe les domaines de l'acoustique et du son, alors que le symbolique se réfère aux structures musicales de plus haut niveau liées à la pensée compositionnelle et aux techniques afférentes de représentation et de calcul.
- Les approches « temps-réel » et les approches « temps-différé », où le temps réel se réfère aux systèmes réactifs principalement utilisés en situation de performance ou d'exécution musicale, alors que les systèmes temps-différé se rapportent aux processus compositionnels dans lesquels les structures musicales sont produites, manipulées ou analysées « hors-temps » (*offline*) en amont de leur exécution. [3]

Le travail présenté dans ce mémoire a pour objectif général d'étendre le champ d'action de la CAO (plutôt axé initialement sur les aspects symboliques de la musique, et suivant un mode de calcul « temps différé ») sur l'ensemble de ces territoires, afin de dépasser ces antagonismes et d'ouvrir la voie à de nouvelles dimensions créatives.

Le chapitre 3 nous aura montré une première étape de convergence entre les approches intensionnelles et extensionnelles dans la spécification des structures temporelles. Dans les chapitres suivants (4 et 5) nous abordons la convergence du domaine symbolique et de celui du signal en composition assistée par ordinateur, rapportée ensuite à ces mêmes problématique de construction des structures temporelles. Le chapitre 6 traitera plus particulièrement de la question de la spatialisation sonore et du croisement des spécifications spatiales et temporelles dans ce même contexte. Nous verrons aussi comment la division paradigmatique initiale entre signal et symbole dans les représentations musicales peut être traitée à travers la convergence du modèle de calcul temps-différé de la CAO avec celui des systèmes réactifs ou temporisés. De là nous glisserons enfin vers les questions d'interaction dans le calcul et dans les représentations temporelles, pour esquisser dans le chapitre 7 une nouvelle génération d'outils de CAO en phase avec les pratiques et problématiques les plus actuelles de la création musicale contemporaine.

3. A l'intérieur de cette distinction, nous verrons qu'il est possible de déceler encore d'autres clivages, par exemple concernant les modes d'exécution des programmes *data-driven* vs. *demand-driven*.

2

CAO/OPENMUSIC : LE COMPOSITEUR-PROGRAMMEUR

Les notions de programmation et de *end-user programming* sont souvent présentes de manière plus ou moins explicite dans la recherche informatique liée à la création musicale. La maîtrise d'un langage et la pratique de la programmation permettent en effet d'optimiser le potentiel calculatoire, mais aussi *expressif* de l'informatique pour les utilisateurs. Les compositeurs seraient donc des programmeurs en puissance : des techniques de contrepoint de la musique baroque, aux processus formels de la musique contemporaine, leur capacité à manipuler les concepts liés au calcul formel ne sont plus à démontrer. En interagissant avec eux on est ainsi frappé par la facilité avec laquelle ils s'approprient des outils et notions informatiques souvent complexes. Attirés par leur puissance exploratoire, et parfois par leur caractère mystérieux, les compositeurs ne reculent pas nécessairement devant cette complexité et acceptent la nécessité d'un apprentissage parfois difficile — l'apprentissage de la pratique musicale est à cet égard certainement plus exigeant.

2.1 LANGAGES DÉDIÉS À LA MUSIQUE

De nombreux environnements informatiques d'aide à la composition musicale ou de création sonore ont été conçus comme des « langages dédiés » — *domain-specific languages* (DSL). Parmi les projets marquants dans le domaine de la composition musicale, on pourra citer par exemple les langages Arctic (Dannenberg *et al.*, 1986), Common Music (Taube, 1991) ou encore Haskore (Hudak *et al.*, 1996). Dans le domaine de la synthèse sonore les travaux de Max Mathews ont donné lieu à une famille de langages appelés « Music *N* » (Mathews, 1969) dont Csound (Boulanger, 2000) est le dernier représentant en date. Plus récemment, Faust (Orlarey *et al.*, 2004) a proposé une alternative fondée sur une sémantique fonctionnelle forte pour la conception de modules de traitement et de synthèse sonore.

Des langages de plus haut niveau pour le contrôle des processus de synthèse ont permis d'étendre les possibilités offertes par la synthèse sonore numérique au domaine de la formalisation et de la structure musicale. Parmi ceux-ci, citons le langage Formes développé à l'IRCAM dans les années 80 (Rodet et Cointe, 1984), les travaux de Roger Dannenberg avec notamment la langage Nyquist (Dannenberg, 1997) ou encore SuperCollider (McCartney, 2006). Enfin, l'essor des systèmes dédiés à l'interaction et au *live coding* a fait émerger récemment une nouvelle génération de langages dynamiques comme Chuck (Wang *et al.*, 2015), Impromptu (Sorensen, 2005), ou Antescofo (Echeveste *et al.*, 2013).

Les langages de programmation fonctionnelle — et en particulier les dialectes de LISP (Steele, 1990) — ont souvent été utilisés pour répondre aux exigences du domaine musical (Dannenberg *et al.*, 1997).[1] Les mécanismes d'abstraction et d'application fonctionnelles mis en

1. Les langages Arctic, Common Music, Formes, Nyquist ou Impromptu que nous avons cités précédemment — et d'autres encore (Eckel *et al.*, 2004) — furent tous implémentés dans divers dialectes de LISP.

avant par ces langages impliquent une vision spécifique des données, considérées comme une concrétisation (application) d'un processus, plutôt que comme une structure statique — vision manifestement pertinente du point de vue compositionnel.

D'autres paradigmes de programmation ont été adaptés dans les langages musicaux, comme celui de la programmation orientée objet (Pope, 1991), ou encore les approches déclaratives comme la programmation par contraintes (Anders et Miranda, 2011). Ces différents paradigmes ont aussi souvent été implémentés ou intégrés au sein d'environnements Lisp. Le langage Formes, par exemple, fut conçu comme extension du paradigme de programmation orientée objet au domaine temporel (Cointe et Rodet, 1984) et constitua une implémentation pionnière de ce paradigme dans le langage Lisp — de même que par la suite, son extension PreFORM (Boynton *et al.*, 1986). [2]

Il est intéressant de constater cependant que ces différents paradigmes ont été adoptés par les compositeurs de manière inégale : si l'approche par contraintes a suscité un intérêt remarquable de leur part, comme en témoigne la diversité des applications musicales qui ont été produites et documentées (Truchet et Assayag, 2011), l'approche orientée-objet en revanche, malgré les applications pionnières citées plus haut [3] et malgré sa prédominance dans l'industrie et la conception logicielle, a rarement fait l'objet à notre connaissance d'applications explicites en composition musicale.

2.2 PROGRAMMATION VISUELLE

L'apparition des langages de programmation visuelle (Burnett, 1999) a marqué un pas décisif dans le développement des langages de programmation dédiés à la musique. Ceux-ci permettent à leurs utilisateurs de développer des programmes en manipulant des éléments graphiques suivant une syntaxe et une sémantique spatiale ou visuelle. Supposés faciliter la programmation (par exemple en favorisant la correction syntaxique des programmes), mais également la compréhension et/ou la résolution des problèmes (via des représentations plus intuitives, ou plus proches des représentations mentales) ils contribuent à un usage plus aisé des systèmes informatiques pour certaines communautés d'utilisateurs, intéressés par la conception de modèles calculatoires ou de systèmes automatisés, mais pas nécessairement formés à la programmation textuelle « classique ». Une littérature importante a été produite dans les années 90, où ces hypothèses furent largement discutées (Boshernitsan et Downes, 1997; Whitley, 1997; Whitley et Blackwell, 1997; Green et Petre, 1992).

Dans le domaine de la composition assistée par ordinateur et de la création musicale en général (qui ont rarement été pris en compte dans les études citées ci-dessus), force est de constater que l'interaction et la conception graphique de programmes ont nettement fait progresser les possibilités et pratiques compositionnelles. Les langages de programmation visuelle offrent aux compositeurs le pouvoir d'expression d'un langage, tout en leur permettant d'entrer intuitivement dans la complexité des processus musicaux et d'en visualiser ou manipuler les données de manière interactive.

2. Introduit quelques années plus tard, CLOS — Common Lisp Object System (Gabriel *et al.*, 1991) — constitue aujourd'hui une implémentation remarquable du paradigme de programmation par objets en Lisp, assorti d'un puissant protocole de « méta-objets » (Kiczales *et al.*, 1991).

3. Par exemple les premières utilisations du système de composition Formes pour le contrôle du synthétiseur Chant, citées dans (Rodet *et al.*, 1982).

Parmi les langages de programmation visuelle, on compte notamment les feuilles de calculs de type Microsoft Excel, [4] les langages basés sur la manipulation d'« icônes », utilisant des objets graphiques et leurs relations spatiales pour définir des programmes, [5] et les langages basés sur les diagrammes (ou graphes) (Burnett et Baker, 1994). La plupart des langages visuels développés pour la musique font partie de cette dernière catégorie : leur syntaxe consiste en un agencement de modules de traitement (appelés communément « boîtes ») connectés les uns aux autres par des arcs (ou connexions) représentant les données transitant d'un module à l'autre. On parle de *patching* pour désigner la création de ces programmes, et par extension, on nomme généralement *patch* un programme visuel réalisé dans ce type de langage. [6]

La représentation sous forme de diagramme constitue une représentation simple et intuitive des processus, des flux de données et des expressions fonctionnelles (Larkin et Simon, 1987), et se prête particulièrement bien à la conception de systèmes *data-flow* réactifs. Dans de tels systèmes, les calculs sont initiés par une donnée d'entrée — on parlera de modèles de calcul *data-* ou *event-driven* [7] — et chaque module ou fonction du programme s'exécute, propageant ses valeurs de sortie en aval du graphe de flot de données. À l'inverse cependant, dans certains langages visuels dits *demand-driven*, les programmes s'exécutent sur requête d'une valeur dans le graphe, se propageant en amont afin d'exécuter les modules nécessaires à la détermination des cette valeur (Lau-Kee *et al.*, 1991). Ce modèle est similaire à la façon dont une expression fonctionnelle est évaluée dans un langage interprété comme Lisp ; c'est également celui implanté traditionnellement dans le domaine de la composition assistée par ordinateur — nous reviendrons sur cette partition des modes d'exécution dans le chapitre 7.

Un certain nombre de langages visuels ont donc aussi été développés pour des applications musicales (Desain, 1986), dont certains furent largement adoptés par les musiciens et compositeurs professionnels. L'environnement Max (Puckette, 1991) constitue l'une des principales réalisations dans le domaine. Initialement conçu pour le contrôle de la station audio-numérique 4X de l'Ircam dans les années 80 (Puckette, 1988), il est aujourd'hui distribué commercialement et utilisé pour la conception et la spécification de toutes sortes de traitements temps-réel d'évènements et de signaux. Une grande partie des systèmes développés à l'heure actuelle dans la création musicale contemporaine repose sur cet environnement, ou sur son pendant open-source, PureData (Puckette, 1996). Le modèle « flot de données » temps-réel rend ces environnements de programmation visuelle particulièrement adaptés aux applications interactives et aux dispositifs *live* utilisés lors de performances musicales (synchronisation d'évènements avec le jeu de musiciens, traitement temps réel de sources sonores, etc.).

4. Voir aussi par exemple (Burnett *et al.*, 2001).

5. Voir par exemple (Monden *et al.*, 1984) ou encore Elody (Orlarey *et al.*, 1997), un langage musical entièrement basé sur le paradigme fonctionnel et sur des opérations simples de drag-and-drop entre icônes.

6. Cette catégorie de langages visuels a été popularisée par des systèmes comme Prograph (Cox *et al.*, 1989) ou LabView (Vose et Williams, 1986), et répandue dans des domaines comme le traitement expérimental de données (Hils, 1991; Helsel, 1998) ou encore le traitement d'image (Rasure et Williams, 1991; Young *et al.*, 1995).

7. Nous nous concentrons ici sur les couches dites « évènementielles » des environnements de programmation visuelle. Cependant la plupart des systèmes musicaux incluent également une couche temps-réel dont les calculs sont dirigés par des requêtes périodiques (*callbacks*) émanant des dispositifs de sortie audio — on parlera alors de calculs *time-driven* (Kopetz, 1991).

Durant la même période, Patchwork (Laurson et Duthen, 1989) introduisit l'usage de la programmation visuelle du côté des représentations « symboliques » et de la composition assistée par ordinateur. Patchwork était en quelque sorte une extension graphique de Lisp, donc fortement lié au paradigme de programmation fonctionnelle qui avait été prédominant dans la plupart des langages musicaux développés jusqu'alors. Le langage — dont le principal mode d'exécution est cette fois *demand-driven* — intégrait un certain nombre de fonctions et structures de données musicales avancées, ainsi que des modalités de visualisation et d'édition de ces structures sous forme de notation musicale. Utilisé par un nombre relativement important de compositeurs, il fut à la fois l'objet et le support de la plupart des recherches en CAO menées à l'IRCAM pendant de nombreuses années.[8] Les principaux systèmes de CAO actuels, OpenMusic (dont il sera principalement question dans ce mémoire) et PWGL (Laurson et Kuuskankare, 2002) héritent de ce langage et en suivent aujourd'hui encore les préceptes.

2.3 OPENMUSIC

L'essentiel de mon activité de recherche se déploie autour de l'environnement OpenMusic (Bresson *et al.*, 2011) 🖉. OpenMusic (OM) est un langage de programmation visuelle dédié à la composition musicale né à l'IRCAM à la fin des années 90 (Assayag *et al.*, 1997), succédant à Patcwhork.[9] Le langage visuel fut principalement développé et formalisé dans la thèse de Carlos Agon (Agon, 1998). Il étend les fonctionnalités de Patchwork, notamment avec une sémantique visuelle spécifique, une couche « objets » (définition de classes et de méthodes intégrées dans le langage visuel), l'utilisation des « méta-objets » (Agon et Assayag, 2003), et avec une extension temporelle des processus musicaux implémentée via le concept de *maquette* (voir chapitre 3, section 3.1). Une idée importante dans la conception du langage fut celle de composants (éditeurs) intégrés dans les programmes, qui deviennent ainsi des modèles interactifs réunissant le processus algorithmique, la visualisation des données et la notation, et permettant l'interaction de ces différents composants d'un projet musical.

Tout comme Patchwork, OpenMusic peut donc être vu comme une interface graphique avec le langage Common Lisp et son système d'objets CLOS, donnant une contrepartie visuelle à la plupart des constructions du langage : abstraction et application fonctionnelle, fonctions d'ordre supérieur, itérations, récursion, structures conditionnelles, définitions de classes, de fonctions génériques et de méthodes, etc. (Agon *et al.*, 2008) 🖉 (Bresson *et al.*, 2009) 🖉.[10] Il propose par ailleurs aux utilisateurs un nombre important d'outils, d'éditeurs graphiques et de bibliothèques de fonctions spécialisées dans le domaine musical.

La figure 2.1 montre un exemple de programme visuel (ou « patch ») dans OpenMusic. Plusieurs types de boîtes sont visibles sur cette figure : les principales sont les boîtes « objets » permettant d'instancier et d'éditer des structures musicales (courbes, partitions, etc.), et

8. Comme nous l'avons évoqué en introduction à propos de la CAO, cet ancrage des langages visuels en musique est corrélé au développement de l'informatique personnelle, des interfaces graphiques, mais fut également porté par des projets phares et des percées notables dans le domaine musical ; par exemple dans le cas de Max, via la collaboration de son concepteur Miller Puckette avec le compositeur Philippe Manoury, en quête d'une approche compositionnelle du « temps réel », ou pour Patchwork via les recherches menées par les compositeurs de l'école dite « spectrale » — voir par exemple (Malherbe, 2008).

9. Une autre branche héritière de Patcwhork fut également développée par Mikaël Laurson et ses collègues de la Sibelius Academy de Helsinki avec le logiciel PWGL (Laurson et Kuuskankare, 2002).

10. L'article (Bresson *et al.*, 2009) 🖉 publié dans *Higher Order and Symbolic Computation* détaille ce parallèle entre les fonctionnalités du langage visuel et les constructions du langage Common Lisp.

les boîtes « fonctionnelles » réalisant des opérations à partir de ces données. Des boîtes d'entrée/sortie (non visibles sur cette figure) permettent également de spécifier des variables connectées comme entrées dans les programmes, ou les valeurs de retour produites lors de l'exécution de ces programmes. Un *patch*, ou programme visuel, peut ainsi représenter une définition de fonction qui pourra être utilisée dans d'autres *patches* sous forme d'« abstraction ». [11]

En programmation fonctionnelle, les *fonctions d'ordre supérieur* sont des fonctions prenant en paramètres d'autres fonctions, ou produisant des fonctions comme valeurs de retour. Cette notion est implémentée dans le langage visuel via un état appelé *lambda*, qui peut être attribué aux boîtes et composants du programme. Lorsqu'elle est dans cet état (cf. par exemple l'icône λ sur la boîte $+$ en haut de la figure 2.1) une boîte retournera comme valeur une fonction équivalente, qui pourra être utilisée comme valeur d'entrée par d'autres boîtes. [12]

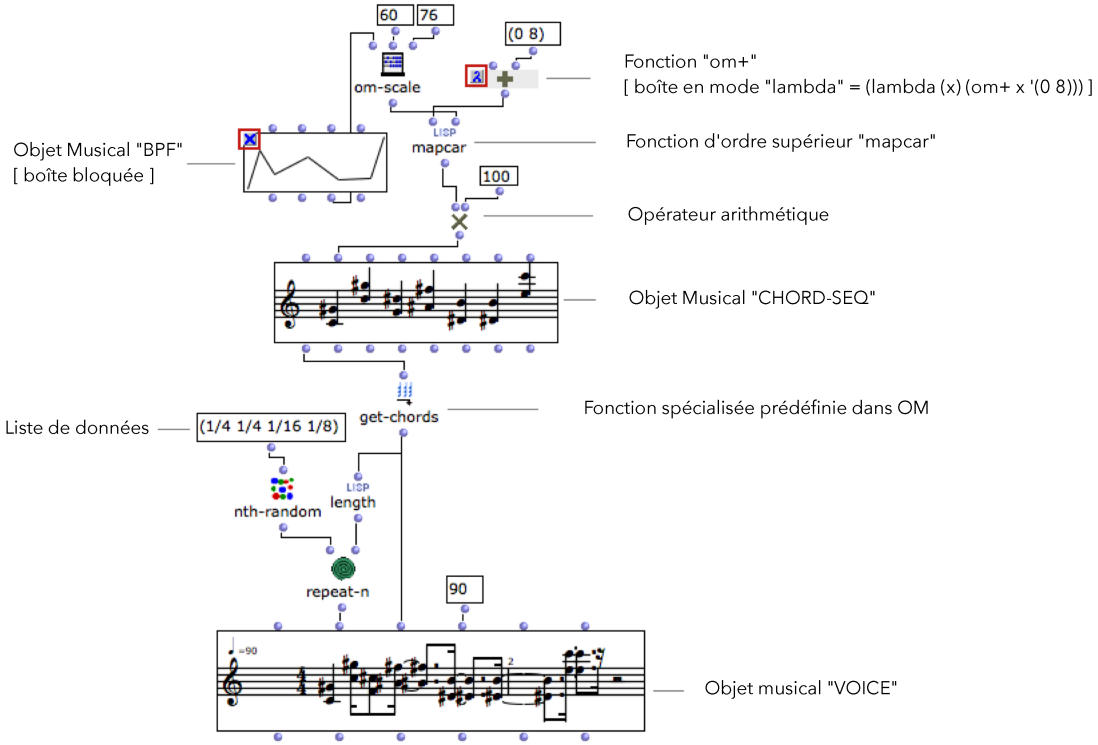


FIGURE 2.1: Un programme visuel (ou « patch ») OpenMusic, et description de quelques composants.

D'un point de vue opérationnel, le langage visuel combine le paradigme fonctionnel et la notion d'état local, avec la possibilité de geler et « stocker » des résultats partiels localisés sur le graphe de calcul. Lorsqu'une boîte est *bloquée*, sa valeur courante (résultat de la dernière évaluation) est enregistrée et réutilisée dans les évaluations suivantes (ce mécanisme permet par exemple de calculer des structures par étapes, évitant de multiplier

11. Le modèle d'évaluation dans OpenMusic produit pour chaque boîte l'équivalent d'une expression LISP correspondant à l'exécution du graphe ascendant à partir de cette boîte. Cette correspondance d'un *patch* avec une fonction LISP assure le passage à l'échelle du modèle lors de son utilisation (sous forme d'abstraction) dans un autre *patch*.

12. Parmi les primitives du langage LISP, des fonctions d'ordre supérieur comme *map* (et ses dérivés comme *mapcar*, cf. figure 2.1), sont fréquemment utilisées pour appliquer d'autres fonctions sur des ensembles de données.

des calculs parfois coûteux, ou permettant simplement de s'arrêter, voire éditer les données à des phases intermédiaires du processus). [13]

L'exécution des programmes est *demand-driven* : l'utilisateur décide, lorsqu'il le souhaite, de calculer ou recalculer une partie du graphe fonctionnel pour produire ou mettre à jour les valeurs de certaines boîtes. Les fonctions utilisent les données connectées à leurs entrées comme arguments, et les objets les utilisent comme valeurs d'initialisation. L'évaluation d'une entrée, lorsqu'elle connectée dans le graphe, entraîne l'évaluation du nœud auquel elle est connectée afin de déterminer sa valeur. À la différence de beaucoup de systèmes fonctionnels, un programme OpenMusic n'a donc pas un seul et unique résultat : il peut être évalué à tout moment et en tout point. Le résultat d'une évaluation (structure musicale, partition, son, ou tout autre type de donnée), même local, peut ensuite être restitué (joué), archivé, copié, exporté, et réutilisé dans des évaluations ultérieures. Il en résulte un workflow original, dans lequel le compositeur peut successivement programmer des fonctions et des processus musicaux, les exécuter, visualiser/écouter et éditer les données entrées ou les résultats à l'aide d'interfaces graphiques. En résumé, un programme OpenMusic n'a pas seulement vocation à produire des données algorithmiquement ; il peut constituer à lui seul la représentation opérationnelle d'une esquisse, « modèle » (Malt, 2003) ou processus/projet compositionnel interactif dans lequel des données musicales sont connectées par des relations fonctionnelles.

Une intégration forte avec le langage Lisp sous-jacent permet de bénéficier de l'expressivité et de la compacité de la programmation textuelle lorsque cela s'avère utile, favorisant le passage à l'échelle et la modularité des programmes créés dans l'environnement (Erwig et Meyer, 1995). Fonctions et programmes sont organisés en *workspaces*, combinant programmes visuels, textuels et outils/ressources externes de manière à permettre la constitution de projets conséquents et la combinaison ou réutilisation du code et des outils développés. Des bibliothèques tierces, contenant généralement des outils spécifiques à certains domaines, procédés ou esthétiques musicales, peuvent enfin être intégrées, ajoutant à la versatilité de l'environnement. De nombreux utilisateurs ont pu ainsi développer et mettre leurs bibliothèques OpenMusic à disposition de la communauté. [14]

2.4 COMMUNAUTÉ ET UTILISATIONS DU LANGAGE

La qualité d'un langage dédié passe par sa capacité à supporter des projets réels (« grandeur nature ») dans le domaine pour lequel il a été conçu. De ce point de vue, OpenMusic compte aujourd'hui des centaines d'utilisateurs répertoriés dans le monde entier. L'application génère plusieurs milliers de téléchargements à chaque nouvelle version et rassemble également une communauté active sur le Forum de l'Ircam (plus de 300 membres inscrits au groupe d'utilisateurs en 2016). L'environnement de CAO est considéré comme l'un des principaux outils actuels dans le domaine de la création musicale contemporaine, et à ce titre a contribué à de nombreuses pièces et projets compositionnels majeurs, entrés pour certains dans le répertoire.

Les utilisateurs d'OpenMusic constituent une communauté relativement hétérogène en termes de connaissances et de facilités dans le domaine de la programmation ou de l'informatique en général. Comme nous l'avons relevé cependant, la création musicale semble être

13. Sur les figures 2.1 ou 2.2 par exemple, noter l'état « bloqué » des boîtes dans les parties supérieures des programmes (icône × dans le coin) : leur valeur a été calculée précédemment, importée ou éditée manuellement, et ne sera pas mise à jour lors de prochaines évaluations.

14. Voir par exemple `http://repmus.ircam.fr/openmusic/libraries`.

le terrain d'une vraie percée concernant l'utilisation des langages de programmation visuelle. Cette pratique, qui a largement dépassé le seul milieu des musiciens-*hackers*, souligne le rôle potentiel de la programmation visuelle comme vecteur d'apprentissage, susceptible d'introduire la programmation à une plus large communauté d'utilisateurs, et de déplacer les frontières classiques entre concepteurs et utilisateurs des systèmes informatiques.

Nous avons, avec Carlos Agon et Gérard Assayag, publié trois ouvrages entre 2006 et 2016 (*The OM Composer's Book*, volumes 1, 2 & 3) dans lesquels des compositeurs décrivent leur utilisation de l'environnent OpenMusic (Bresson *et al.*, 2016) ✎. [15] Les 58 contributions rassemblées dans ces ouvrages sont une excellente illustration de la diversité des applications et du potentiel de cet outil, et de la CAO en général. [16] Les figures 2.2, 2.3 et 2.4 sont extraites du volume 2. Le troisième volume de la série, publié en 2016, illustrera certaines évolutions récentes des usages et techniques de CAO dans la suite de ce mémoire : j'y ferai référence concernant certaines applications spécifiques de mes recherches. [17]

Il est difficile de présenter des exemples d'applications « typiques » du langage visuel ; en effet celles-ci ne seront jamais représentatives de l'ensemble de ses usages réels (nous renvoyons pour cela le lecteur aux ouvrages mentionnés plus haut). Ci-dessous sont toutefois rapportées quelques observations sur ces usages, reprenant les conclusions et exemples de (Bresson *et al.*, 2009) ✎.

RÔLE DE LA PROGRAMMATION. Le rôle de la programmation visuelle et de la composition assistée par ordinateur dans une approche ou au sein d'un projet compositionnel donné varie fortement d'un compositeur à l'autre. Si certains auront recours à un programme pour produire une partition entière en une passe d'exécution, d'autres, à l'opposé, considèreront l'outil de CAO comme un environnement expérimental dans lequel sont testées des idées, mais d'où aucun matériel musical concret ne sera nécessairement tiré. Entre ces deux extrêmes se situent la majorité des utilisateurs qui selon leurs projets, objectifs, ou niveaux d'expertise, déterminent une approche spécifique, focalisée sur des concepts plus ou moins formels et sur la génération ou le traitement de structures musicales plus ou moins concrètes (de formes abstraites, aux structures rythmiques, harmoniques, et jusqu'à la transformation, l'analyse ou la synthèse de signaux sonores).

PROGRAMMATION VISUELLE VS. TEXTUELLE. Il est intéressant également de constater la variété des approches concernant le recours à la programmation « classique » (textuelle) en complément de la programmation visuelle. On observe en effet, selon les cas, des utilisations purement graphiques du langage, où l'intégralité d'un processus est réalisé par des programmes visuels (depuis les traitements des données de bas niveau jusqu'à la connexion de modules à plus grande échelle), et à l'opposé, des utilisations du système comme simple environnement intégrateur où des modules programmés en Lisp sont rassemblés dans un cadre fonctionnel et dans un projet commun. Encore une fois, la plupart du temps la pratique se situe entre ces deux cas extrêmes ; tout du moins, l'utilisation des fonctions musicales et autres objets et éditeurs prédéfinis dans OpenMusic positionne l'utilisateur moyen dans un cas de figure intermédiaire. Dans les exemples présentés en figures 2.2 et 2.3, on observe des fonctions représentant des primitives Lisp (par exemple *length* sur la figure 2.2),

15. J'ai assuré la coordination éditoriale des volumes 2 et 3 (2008/2016).

16. Rozalie Hirs et Robert Gilmore ont également publié un ouvrage sur OpenMusic, focalisé plus spécifiquement sur l'héritage du compositeur Tristan Murail et de la musique spectrale (Hirs et Gilmore, 2009).

17. Les résumés des différents chapitres de l'ouvrage *The OM Composer's Book 3* sont donnés en annexe ✎.

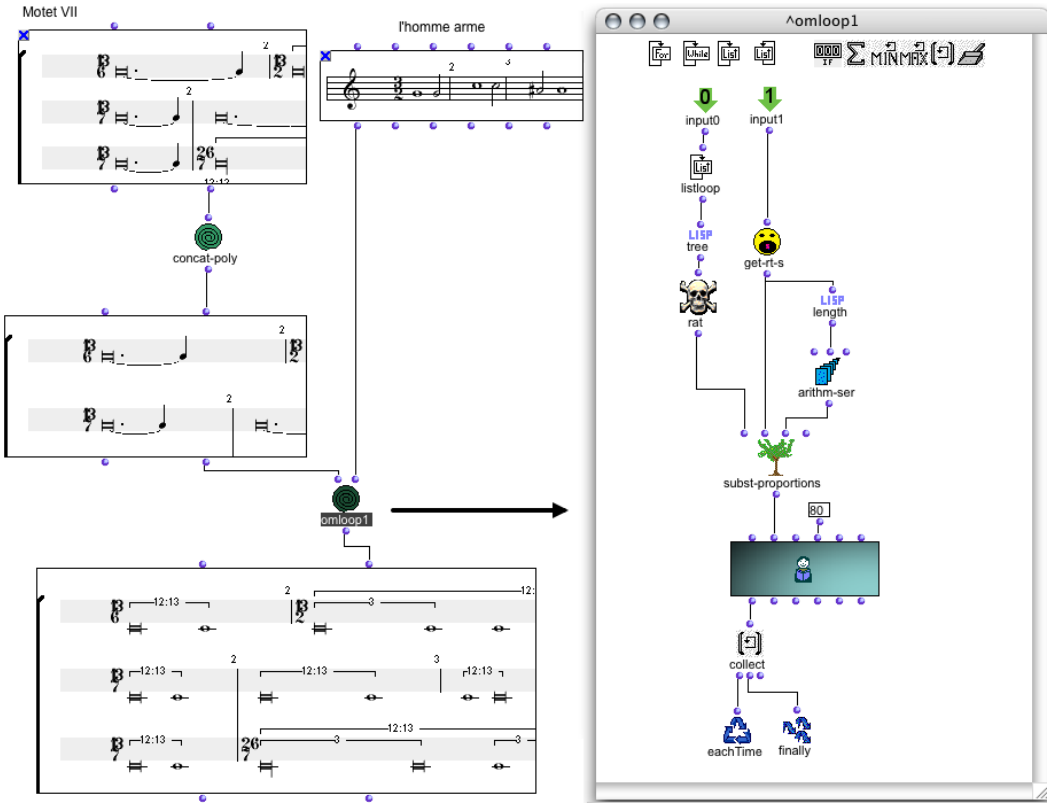


FIGURE 2.2: Karim Haddad : *Livre Premier de Motets* (2007). Calcul d'une partie rythmique. Sur la droite, une fenêtre OMLOOP premettant d'implémenter graphiquement une itération (*loop*). Extrait de *The OM Composer's Book 2* – (Haddad, 2008).

des fonctions OpenMusic de plus haut niveau (par exemple *arithm-ser* sur la figure 2.2, *om-scale*, *flat* sur la figure 2.3), ou des fonctions programmées par l'utilisateur ou chargées à partir de bibliothèques externes pour des opérations plus spécifiques (par exemple *subst-proportions* sur la figure 2.2, pour la substitution de motifs rythmiques par d'autres dans un contexte de structure hiérarchique, ou *interpol-prof* sur la figure 2.3, pour l'interpolation entre des lignes harmoniques suivant des profils et contraintes données). L'expérience tend à montrer que les utilisateurs expérimentent généralement avec les programmes visuels dans un premier temps, puis encapsulent les outils pertinents dans des modules et abstractions Lisp au fur et à mesure que ces outils deviennent plus génériques et matures, et qu'ils acquièrent eux-mêmes de l'expertise en programmation.

PARADIGMES DE PROGRAMMATION. Nous relevons pour finir quelques observations sur l'usage des différents paradigmes de programmation cités plus haut.

La programmation fonctionnelle, comme nous l'avons mentionné plus tôt, semble s'adapter naturellement aux démarches et aux modes de pensée des compositeurs, de même que le paradigme *data-flow* en général. La figure 2.4 est un exemple d'utilisation intensive de fonctions d'ordre supérieur pour l'implémentation de traitements successifs sur des structures musicales.

Bien que nous ne les ayons abordés que brièvement ici, les outils de programmation par contraintes trouvent aussi régulièrement des applications concrètes en composition musicale. OpenMusic, tout comme avant lui Patchwork et aujourd'hui PWGL, a été fréquemment utilisé comme environnement hôte pour l'implantation de ces systèmes, permettant de résoudre des problèmes de satisfaction de contraintes (CSP) liés à la construction de structures harmoniques, rythmiques, polyphoniques etc. (Bonnet et Rueda, 1998; Sandred, 2000; Truchet *et al.*, 2003). Des applications sont par exemple présentées dans *The OM Composer's Book* par Michel Amoric, Georges Bloch, Juan Camilo Hernández (à propos d'une pièce de Mauro Lanza), Johannes Kretz, Örjan Sandred, Killian Sprotte (dans le volume 1 / 2006), Christopher Melen et Permagnus Lindborg (dans le volume 2 / 2008), ou encore Julien Vincenot [🖉 6] (dans le volume 3 / 2016). Si cet aspect de la recherche et développement dans OpenMusic est moins actif aujourd'hui qu'il ne le fut dans les années 1990-2000 (on remarque en effet un net déclin de représentation sur la quinzaine d'années que couvrent les productions décrites dans les trois volumes cités ci-dessus), des projets continuent à voir le jour dans cette direction (Toro *et al.*, 2016) ; les derniers en date étant issus des travaux d'étudiants de l'Université Catholique de Louvain, comme Sasha Van Cauwalaert (Van Cauwelaert *et al.*, 2012) et plus récemment, Grégoire Zoetardt (2016) [⋆].

Comme nous l'avons aussi noté précédemment, la programmation par objets, contrairement à l'approbation générale qu'elle a suscité dans le domaine de l'ingénierie logicielle, est relativement peu mise à contribution par les compositeurs utilisant le langage visuel. Elle l'est indirectement par l'utilisation récurrente de classes prédéfinies dans le système, mais ne fait pas réellement l'objet d'une utilisation paradigmatique pour atteindre des objectifs ou implémenter des concepts compositionnels spécifiques. Une exception notable est la bibliothèque OMChroma, sur laquelle nous reviendrons dans le chapitre 5. Issue d'une approche développée par le compositeur Marco Stroppa, celle-ci utilise le concept de classe pour réifier via des modules de synthèse l'idée de « potentiels sonores » (Agon *et al.*, 2000). Des classes d'objets peuvent y être créées, étendues et personnalisées graphiquement avant d'être instanciées pour contrôler des processus de synthèse sonore.

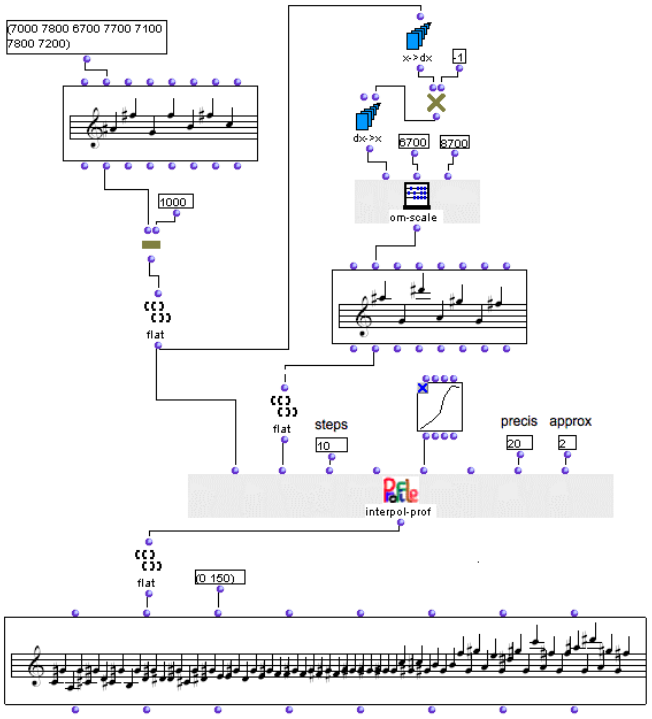


FIGURE 2.3: Christopher Melen : *Six Metal Fugue* (2006). Calcul d'une interpolation entre un profil mélodique et sa propre inversion (transformée par étirement dans le domaine des hauteurs). Extrait de *The OM Composer's Book 2* – (Melen, 2008).

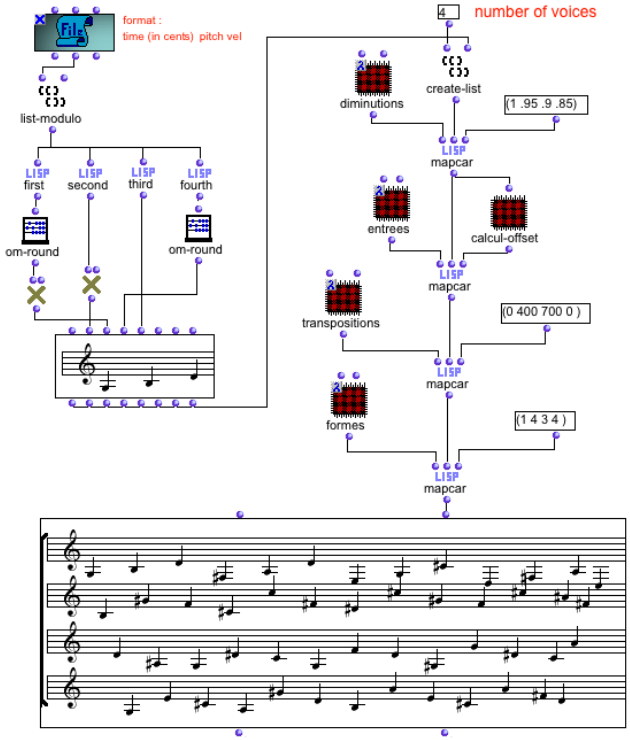


FIGURE 2.4: Asbjørn Schaathun : *Double Portrait* (2006). Transformation/élaboration de séquences. Les boîtes d'abstraction (sous-programmes, marqués par une icône λ) permettent de traiter des données d'entrée de façon modulaire par des appels à la fonction d'ordre supérieur *mapcar*. Extrait de *The OM Composer's Book 2* – (Lemouton et Schaathun, 2008).

2.5 DÉVELOPPEMENT DE L'ENVIRONNEMENT

Le développement et la mise à disposition d'un environnement stable et opérationnel pour les utilisateurs est primordial dans la conduite de mon projet de recherche sur le long terme, permettant d'insérer les résultats et avancées théoriques de manière efficace dans la pratique et l'activité régulière des compositeurs. Ce développement est également le terrain d'une recherche plus générale autour des modèles et architectures logicielles, de la programmation visuelle, ou encore de l'interaction homme-machine.

Suite à ma formation et à mon stage d'ingénieur réalisé dans l'équipe Représentations Musicales de l'IRCAM, ma première contribution au développement d'OpenMusic fut une mission de quelques mois pour laquelle j'ai été engagé en 2003, visant à développer un ensemble d'outils-support pour le format MIDI (Musical Instrument Digital Interface). L'idée, au delà d'un simple encodage des structures musicales destiné à leur restitution via le protocole MIDI, était de permettre la manipulation des données de « bas niveau » transitant dans ce format au sein des programmes visuels (Bresson et Agon, 2010) — je reviendrai également sur cette idée à propos d'autres types de données liées aux descriptions sonores dans le chapitre 4.

Je me suis ensuite inséré plus durablement dans l'équipe pour prendre part de manière significative aux tâches et responsabilités de développement, à travers un certain nombre de projets et de jalons.

MUSIQUE LAB 2. Commandé à l'IRCAM par le ministère de l'Éducation Nationale et le ministère de la Culture et de la Communication en 2004, le projet Musique Lab 2 visait à proposer une application pédagogique basée sur les concepts et technologies de composition assistée par ordinateur, destinée aux enseignants et aux élèves de classes de musique allant du collège jusqu'aux conservatoires, et permettant l'étude et la formation aux répertoires tant classique que contemporain (Puig *et al.*, 2005; Guédy *et al.*, 2007). Chargé du développement de l'application, j'ai élaboré à partir d'OpenMusic un protocole simplifié, basé sur les opérations de *drag-and-drop*, pour la création et la transformation de structures musicales suivant des procédures « classiques » de la CAO : traitement des structures harmoniques, rythmiques et mélodiques par la combinatoire, transpositions, analyse des signaux audio et extraction de données spectrales (Bresson, 2007), etc. L'application développée, [18] ajoute à ces opérations une notion de tonalité, permettant de couvrir le domaine de la musique tonale : arpégiation, marches d'harmonie, ou simples modulations — concepts peu considérés dans le contexte d'outils de création musicale contemporaine. Elle a été utilisée dans certains collèges et instituts de formation musicale, et distribuée sur la plateforme *Educ'Net* du ministère, ainsi que sur la plateforme *ForumNet* de l'IRCAM. [19] Ce projet a été documenté dans différents articles de journaux et conférences (Assayag et Bresson, 2006; Bresson *et al.*, 2006, 2013) (Bresson, 2010).

OM5. L'un des défis techniques dans l'implémentation du projet Musique Lab 2 est venu de la prescription initiale de produire une application exécutable sur le système d'exploitation Windows. OpenMusic était à l'époque exclusivement développé dans MCL, une implémentation de Common Lisp sur Macintosh intégrant un ensemble important

18. Cette application, initialement appelée *ML-Maquette* en référence à la *maquette* OpenMusic (voir chapitre 3, section 3.1), fut rebaptisée simplement *Musique Lab 2* suite à un resserrement du périmètre général du projet.

19. `http://forumnet.ircam.fr/product/musique-lab-2-en/`

de fonctions d'accès au système et à ses couches graphiques, et donc étroitement liée à cette plateforme.[20] Nous avons donc décidé de développer une version multi-plateforme de l'environnement de CAO, sur laquelle serait basée l'application Musique Lab 2. J'ai entrepris un travail de réorganisation du code en profondeur, qui a conduit à la création d'une API (Application Programming Interface) présentant les points d'accès principaux vers des fonctions systèmes (en particulier, la création de composants graphiques et les interactions utilisateur). OpenMusic pouvait dès lors être implémenté en ANSI Common Lisp (portable) avec des appels à cette API, elle-même implémentée parallèlement sur Mac et Windows. Ce travail m'a permis de mettre en 2005 à disposition des utilisateurs la version OpenMusic 5, première déclinaison multi-plateforme (Mac/Windows) du logiciel (Bresson *et al.*, 2005) 🖉.

Le passage et la maintenance multi-plateforme d'un environnement comme OpenMusic est un défi important, et plusieurs initiatives avaient été entreprises auparavant, visant à porter celui-ci (dont les sources sont libres et distribuées sous licence GNU GPL) sur Linux. L'une de celles-ci avait eu lieu peu avant mon arrivée à l'IRCAM avec OM4 (Sarria et Diago, 2003). Nous avons plus tard initié un autre portage Linux de l'API OM5 sur le compilateur LISP SBCL dans le cadre du stage d'ingénieur d'Ephrem Boudonnet (2006) [★]. Ces différentes initiatives n'ont cependant pas abouti à une distribution stable et pérenne.

OM6. La version OpenMusic 6 a été proposée aux utilisateurs en 2008, suite notamment à des évolutions des architectures Macintosh (abandon de l'architecture RISC/PowerPC et passage sur microprocesseurs Intel) et au portage qui s'en suivit de l'API OpenMusic sur LispWorks, une autre implémentation de Common Lisp cette fois dotée d'une couche d'abstraction graphique multi-plateforme.

J'ai pleinement endossé depuis le développement, la maintenance et la coordination des projets réalisés dans l'environnement, ainsi que l'animation de la communauté croissante de ses utilisateurs.

Quelques années plus tard je m'engageai dans une collaboration de développement avec Anders Vinjar [★], compositeur norvégien soutenu par le *Bergen senter for elektronisk kunst* (BEK), qui nous a permis de distribuer pour la première fois en 2013 une version stable et opérationnelle de OpenMusic sur Linux (Vinjar et Bresson, 2014) 🖉.[21]

OM#. A partir de 2013 j'ai initié également le développement d'une nouvelle implémentation du langage visuel OpenMusic, conçue dans le but de tester des nouvelles formes de calcul et d'interaction avec les programmes visuels (nous y reviendrons dans le chapitre 7). Ce prototype servira de base aux recherches menées entre 2013 et 2017 dans le cadre du projet ANR *EFFICACe* (Bresson *et al.*, 2015) 🖉 et fait l'objet de mes travaux actuels.[22]

20. Les ordinateurs Apple/Macintosh ont longtemps été les plus utilisés dans le domaine de la musique assistée par ordinateur.

21. Voir par exemple les articles parus dans *Linux Weekly News* : `http://lwn.net/Articles/574593/` ou sur *LinuxMAO.org* : `http://www.linuxmao.org/OpenMusic`.

22. **Note de révision, 7/03/2022 :** OM# était désigné par OM7 dans la version originale du manuscrit. Il s'agit à présent d'une branche de développement distincte : `https://github.com/cac-t-u-s/om-sharp`

PUBLICATIONS (CHAPITRE 2)

AGON, C., BRESSON, J. et ASSAYAG, G. (2008). OpenMusic : Design and Implementation Aspects of a Visual Programming Language. *In 1st European Lisp Symposium (ELS'08)*, Bordeaux, France. Work-in-progress paper.

ASSAYAG, G. et BRESSON, J. (2006). OpenMusic : de la composition à l'enseignement. *In L'inouï*, volume 2. Ircam - Léo Scheer.

BRESSON, J. (2007). Processus compositionnels et opérateurs musicaux dans ML-Maquette – Les outils de traitement du signal. *In Actes des Journées d'Informatique Musicale (JIM'07)*, Lyon, France.

BRESSON, J. (2010). ML-Maquette / Musique Lab 2. *In Proceedings of the International Computer Music Conference (ICMC'10)*, New York / Stony Brook, USA.

BRESSON, J. et AGON, C. (2010). Processing Sound and Music Description Data Using OpenMusic. *In Proceedings of the International Computer Music Conference*, New York / Stony Brook, USA.

BRESSON, J., AGON, C. et ASSAYAG, G. (2005). OpenMusic 5 : A Cross-Platform release of the Computer-Assisted Composition Environment. *In Proceedings of the 10th Brazilian Symposium on Computer Music (SBCM'05)*, Belo Horizonte, Brasil.

BRESSON, J., AGON, C. et ASSAYAG, G., éditeurs (2006/2008/2016). *The OM Composer's Book (3 volumes)*. Editions Delatour / Ircam-Centre Pompidou.

BRESSON, J., AGON, C. et ASSAYAG, G. (2009). Visual Lisp/CLOS Programming in OpenMusic. *Higher-Order and Symbolic Computation*, 22(1).

BRESSON, J., AGON, C. et ASSAYAG, G. (2011). OpenMusic. Visual Programming Environment for Music Composition, Analysis and Research. *In ACM MultiMedia 2011 (OpenSource Software Competition)*, Scottsdale, AZ, USA.

BRESSON, J., BOUCHE, D., GARCIA, J., CARPENTIER, T., JACQUEMARD, F., MACCALLUM, J. et SCHWARZ, D. (2015). Projet EFFICACE : Développements et perspectives en composition assistée par ordinateur. *In Actes des Journées d'Informatique Musicale (JIM'15)*, Montréal, Canada.

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, J., GUÉDY, F. et ASSAYAG, G. (2013). Musique Lab 2 : From Computer-Aided Composition to Music Education. *Journal of Music, Technology & Education*, 5(3).

GUÉDY, F., BRESSON, J. et ASSAYAG, G. (2007). Musique Lab 2 - Un environnement d'aide à la pédagogie musicale. *In Actes des Journées d'Informatique Musicale (JIM'07)*, Lyon, France.

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 (ICMC'05)*, Barcelona, Spain.

VINJAR, A. et BRESSON, J. (2014). OpenMusic on Linux. *In Linux Audio Conference (LAC'14)*, Karlsruhe, Germany.

3

PROJETS ET TRAVAUX DANS OPENMUSIC

Dans ce chapitre sont présentés un certain nombre de projets caractéristiques réalisés dans OpenMusic autour des questions des structures temporelles et rythmiques, de l'analyse musicale, de l'orchestration, ou encore dans les arts graphiques.

3.1 STRUCTURES TEMPORELLES : MAQUETTE/SHEET

Les structures temporelles sont fondamentales dans les processus compositionnels. Leurs représentations et modalités de création sont l'un des principaux défis des outils et recherches actuelles en composition assistée par ordinateur. [1]

Dans OpenMusic, des processus fonctionnels sont définis visuellement et permettent de générer des structures temporelles d'une complexité arbitraire. L'exécution des programmes n'a pas lieu dans un contexte temps-réel : elle est indépendante de la temporalité inhérente et du « rendu » des données musicales, et donc des contraintes de causalité temporelle (avant/après) ou de temps des calculs. [2] Cela permet à ces données d'être manipulées, générées et structurées dans le temps ; en d'autres termes, le temps ne dirige pas l'exécution des programmes, mais il est un paramètre, et l'une des données principales traitée et produite par ces programmes.

La représentation visuelle des programmes sous forme de *patch*, cependant, ne permet d'entrer dans la structure temporelle des processus que de façon limitée. Le concept de *maquette*, introduit par Carlos Agon et Gérard Assayag lors du passage de Patchwork à OpenMusic, constitue une avancée majeure à ce sujet (Assayag *et al.*, 1999). Il sera pour nous la première implémentation concrète d'un rapprochement entre les conceptions « extensionnelles » et procédurales des structures temporelles que nous avons mentionnées en introduction de ce mémoire.

MAQUETTE. La maquette est un programme visuel dont l'axe horizontal représente le temps, ce qui permet de considérer ce programme comme une structure temporelle intégrant les éléments dont il est constitué (voir figure 3.1). On peut ainsi voir celle-ci comme un séquenceur d'objets programmables dont le contenu et la structure seraient définis de manière fonctionnelle, permettant d'agir à la fois sur l'organisation temporelle et sur la définition des processus. Les liens fonctionnels entre sous-programmes au sein d'une maquette permettent d'exprimer des relations entre les objets musicaux, notamment par l'intermédiaire d'outils de « méta » programmation visuelle (Agon et Assayag, 2003) : comme on peut le voir par

1. Les travaux présentés ici sont à rattacher à une thématique générale de recherche de l'équipe Représentations Musicales de l'IRCAM autour du temps, par exemple avec des projets comme Antescofo (Cont, 2008), OMax (Assayag *et al.*, 2006), Partitions interactives (Allombert *et al.*, 2008), etc.

2. Nous reviendrons sur ces caractéristiques d'exécution dans le chapitre 7.

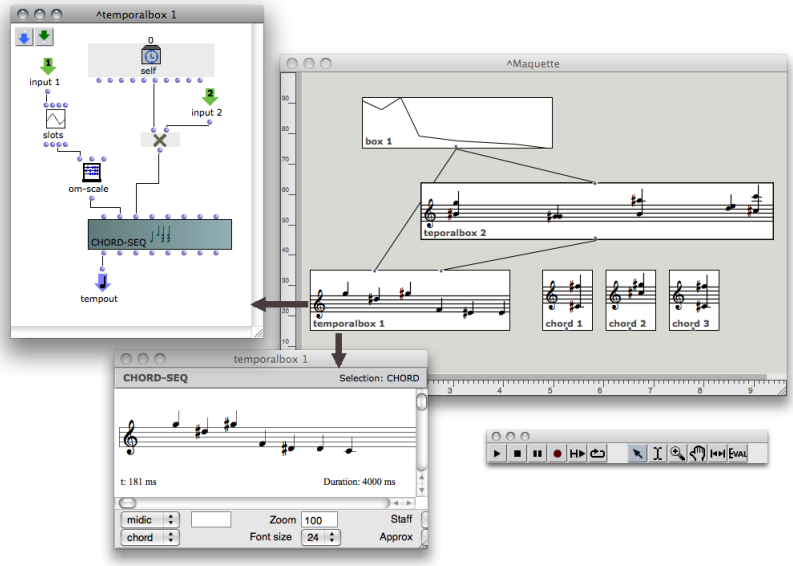


FIGURE 3.1: Une *maquette* OpenMusic : représentation temporelle et fonctionnelle d'une forme musicale. Le contenu de la boîte *temporalbox 1* est visible sous forme d'un processus (programme visuel) et d'une séquence musicale résultant de ce processus, intégrée dans la structure globale.

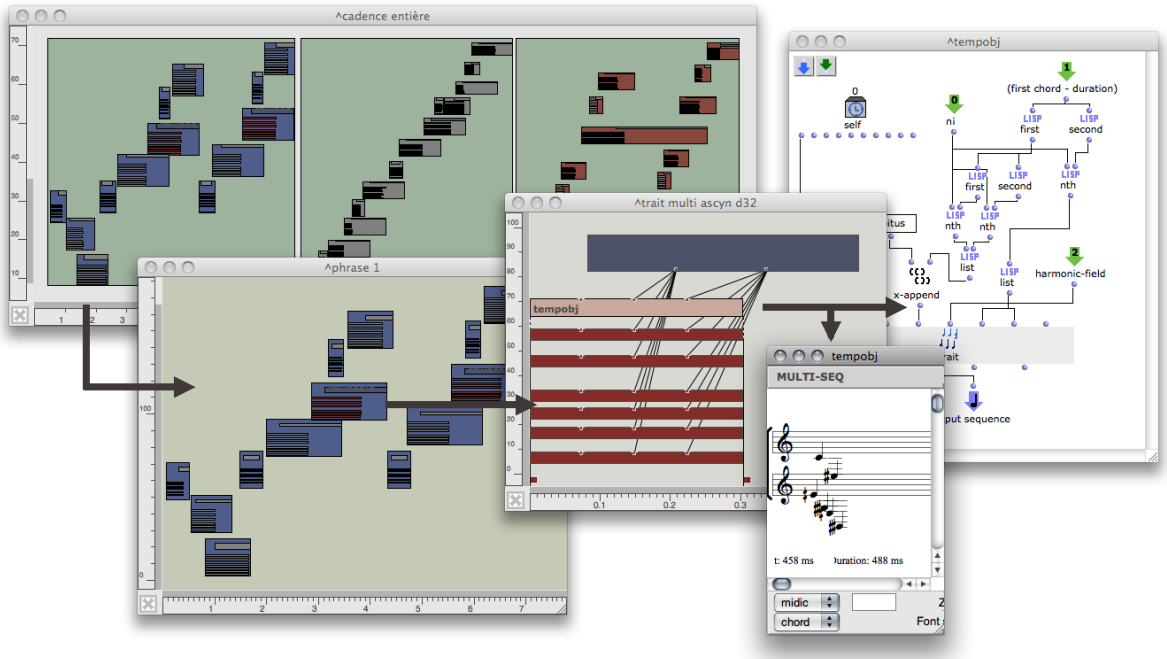


FIGURE 3.2: Reconstitution de la maquette de contrôle des pianos mécaniques dans *Encore* (2006) de Jean-Luc Hervé.

exemple sur la figure 3.1, les blocs fonctionnels (appelés *temporal boxes*) peuvent accéder en lecture ou en écriture à leurs propres caractéristiques graphiques et temporelles (boîte *self* sur la figure 3.1). Les relations programmables entre les positions des différents objets/processus dans le temps sont ainsi potentiellement plus complexes que les simples relations de la logique d'intervalles (Allen, 1983) ; par exemple, un bloc A se positionnant n secondes après que B ait terminé, n pouvant lui-même être une variable dans le calcul de la position de A, etc. Il est également possible d'utiliser des points d'ancrage fixes pour synchroniser débuts et fins des blocs temporels. Antoine Allombert, qui a réalisé sa thèse de l'Université de Bordeaux entre le LaBRI et l'équipe Représentations Musicales, a également implémenté dans la maquette un prototype permettant de spécifier et de résoudre des contraintes logiques entre blocs temporels au sein de « partitions interactives » (Desainte-Catherine et Allombert, 2005; Allombert *et al.*, 2008).

La possibilité d'imbriquer les maquettes les unes dans les autres permet par ailleurs de construire des structures temporelles hiérarchiques. Au niveau fonctionnel, ce modèle permet d'envisager le passage de paramètres d'un niveau hiérarchique à l'autre et de voir la maquette comme une abstraction, ou une fonction à part entière retournant une valeur globale dans un programme ou dans une autre maquette (Bresson et Agon, 2006) 🖉. Un exemple de cette utilisation est la maquette créée pour le contrôle de pianos mécaniques Disklavier dans la pièce *Encore* de Jean-Luc Hervé (2006), décrite dans (Hervé et Voisin, 2006). La figure 3.2 présente une reconstitution plus récente de cette même maquette. La figure A.5 (en annexe) montre une autre maquette réalisant un processus de montage audio à grande échelle pour la partie électronique de sa pièce *Germination* (2013) [🕮 14].

SHEET. L'objet *sheet* constitue une alternative à la maquette conçue pour intégrer des objets musicaux hétérogènes dans une structure polyphonique plus proche des standards de la notation musicale traditionnelle,[3] et permettant d'y intégrer différentes notions propres à la CAO. Une motivation de ce projet était également de représenter des « partitions électroacoustiques » réunissant parties instrumentales, structures et signaux de contrôle, et processus de synthèse sonore. Un *sheet* OpenMusic propose donc une représentation polyphonique d'objets hétérogènes (partitions, sons, courbes, graphiques, *maquettes*, etc.), un alignement et une synchronisation des différents modes de notation (rythmique, proportionnelle, continue, etc.), et une intégration de relations fonctionnelles entre les composants de la partition (sur le modèle de la maquette).

La simultanéité temporelle est matérialisée graphiquement par un alignement vertical. La coexistence des différents modes de notation nécessite donc de prendre en compte les rapports espace/temps dans chacun des composants de la partition, afin de déterminer une fonction de distorsion de l'espace à l'échelle de la partition (par exemple, une barre de mesure occupe de l'espace mais pas de temps, une note rapide occupera proportionnellement plus d'espace qu'une ronde par rapport à sa durée, etc.) Cette fonction appliquée à l'ensemble des objets permet ainsi d'aligner plusieurs voix, qu'elles soient représentées sous forme rythmique, proportionnelle, ou même constituée de sons ou de courbes de contrôle.

La version initiale du *sheet* a été décrite dans (Agon et Bresson, 2007) 🖉 et (Bresson et Agon, 2008) 🖉. Une seconde version fut implémentée avec OM6, et pour différence principale un mode de calcul interne séparé visuellement de la partition (voir figure 3.3), représenté

3. Le terme « *sheet music* » est utilisé en anglais pour désigner une partition écrite ou imprimée sur papier.

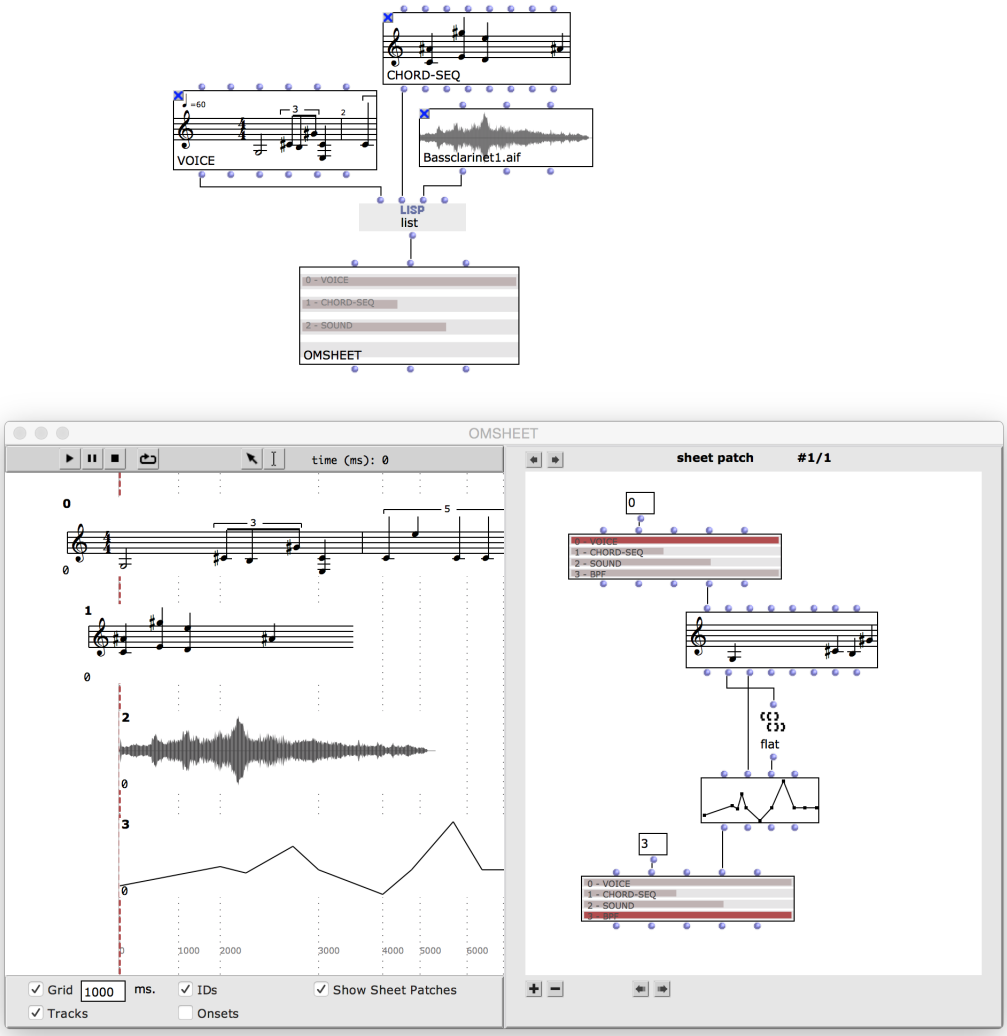


FIGURE 3.3: *OMSheet* : une proposition de représentation des structures et processus musicaux axée sur la notation musicale. En haut : construction des blocs dans un programme visuel. En bas : visualisation et édition de la partition. Sur la partie droite de l'éditeur, un programme intégré détermine les relations fonctionnelles entre éléments de la partition. La courbe (BPF) en bas de la partition est par exemple produite ici à partir des hauteurs des notes de la voix supérieure.

sous forme de *patch* « annexe » intégrant un accès au *sheet* « conteneur » (*sheet-access* sur la figure). Cette nouvelle version est présentée dans (Bresson et Agon, 2011) 🖉.

Ce travail restera parmi les « *proofs of concepts* » de l'environnement, n'ayant jamais réellement trouvé sa voie dans la pratique réelle des utilisateurs. Il aura cependant mis en avant des caractéristiques importantes qui inspireront les recherches à venir, par exemple concernant l'intégration d'objets hétérogènes dans la notation des processus de CAO.

3.2 RYTHME

OpenMusic propose un certain nombre de représentations permettant d'élaborer les constructions temporelles et rythmiques. Deux principales représentations coexistent dans l'environnement, l'une basée sur une mesure de temps « absolu » (c'est le cas du classique « piano-roll » associés aux partitions au format MIDI, où chaque note est décrite individuellement par un *onset* et une durée, généralement exprimés en millisecondes), l'autre sur une métrique et une pulsation, correspondant à la notation rythmique traditionnelle de la musique (Agon *et al.*, 2002). Ces représentations correspondent respectivement aux objets OpenMusic *chord-seq/multi-seq* et *voice/poly*. [4] Elles sont difficilement compatibles (le *sheet* présenté plus haut constitue à cet égard une proposition originale permettant de les insérer dans un même contexte graphique), et se rapportent à différentes approches compositionnelles (ou parfois à différents stades d'un même processus compositionnel).

Un problème classique de l'informatique musicale consiste à passer du temps physique, mesuré en millisecondes, à un temps musical exprimé sous forme de subdivisions d'une pulsation (donnée ou calculée), et noté sur une partition (Desain et Honing, 1992). Ce procédé de *quantification rythmique* intervient par exemple lors de la transcription d'une captation du jeu d'un musicien, ou simplement pour la conversion de structures temporelles produites en temps absolu vers le domaine de la notation rythmique (par exemple, pour permettre à un musicien de lire et jouer ces structures). Il a fait l'objet de nombreux travaux dans l'équipe Représentations Musicales, concrétisés notamment dans le quantificateur de l'environnement OpenMusic — *omquantify* (Agon *et al.*, 1994) — et dans son interface utilisateur, implémentée par la suite dans la bibliothèque OMKANT (Meudic, 2004).

Entre 2012 et 2016 j'ai co-encadré avec Florent Jacquemard (Inria / équipe-projet Mutant) plusieurs projets d'étudiants portant sur la conception de nouvelles approches pour la représentation des structures rythmiques et la quantification, qui sont présentés dans la suite de cette section.

ARBRES DE RYTHME ET RÉÉCRITURE. Dans le cadre de son mémoire de Licence de l'ENS Cachan-Bretagne, Pierre Donat-Bouillud (2013) [⋆] a développé un formalisme d'« arbres de rythme symboliques », inspiré des arbres rythmiques utilisés pour la représentation et la notation des rythmes dans l'environnement OpenMusic (Agon *et al.*, 2002). [5] Ce formalisme se base sur des arbres étiquetés par un petit ensemble de symboles (et non pas par des nombres comme dans les arbres OpenMusic), ce qui permet un traitement purement syntaxique (sans arithmétique) utilisant des techniques de transformation par réécriture de termes (Dershowitz et Jouannaud, 1990). Les systèmes de réécriture sont utilisés par exemple dans le domaine de la déduction automatique, dans la théorie des langages de programmation

4. Les objets *multi-seq* et *poly* sont les dérivés polyphoniques de *chord-seq* et *voice*.

5. Dans OpenMusic, un arbre rythmique (ou *rhythmic tree*) est une structure de donnée définie récursivement par un couple $RT = (d, s)$, où d est une durée et s est une liste de proportions d_i et/ou de sous-arbres (d_i, s_i).

fonctionnels, ou encore pour le traitement du langage naturel, dans les données web, etc. Ils procèdent à des transformations d'arbres établies selon des *règles* qui définissent des substitutions de motifs, correspondant dans le domaine rythmique à des modifications ou simplifications (conservatives ou pas, selon les cas) de la notation musicale (voir figure 3.4) (Jacquemard *et al.*, 2015) ✎. Les règles de réécriture peuvent ainsi être vues comme une axiomatisation de la notation rythmique, et appliquées au raisonnement sur les notations équivalentes dans les domaines de la composition ou de l'analyse musicale. Il est également possible d'utiliser différentes mesures de la complexité des arbres pour évaluer par exemple la qualité relative de notations équivalentes dans un processus de quantification.

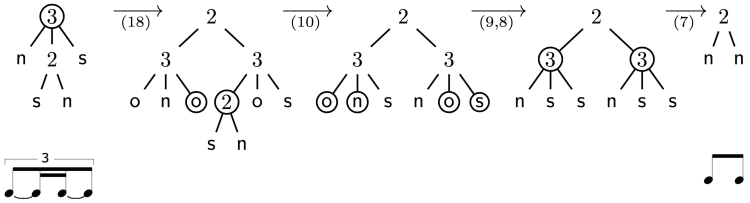


FIGURE 3.4: Transformations rythmiques par application de règles de réécriture.

Les automates d'arbres (Comon *et al.*, 2007) ont également été utilisés en complément des règles de réécriture pour contrôler les processus de modification/simplification des arbres. Un automate permet en effet de reconnaître ou caractériser différents types d'arbres et ainsi de filtrer des ensembles d'arbres générés par réécriture, favorisant certains motifs ou certaines propriétés structurelles (une approche similaire à l'utilisation de schémas pour le traitement de données XML).

SUPERVISION ET APPRENTISSAGE. Parallèlement, Adrien Maire (2013) [⋆] réalisait son projet de M1 de l'ENS autour de l'idée de pondération des arbres, et de l'utilisation d'*automates d'arbres pondérés* pour traiter ce problème de quantification rythmique. Il établit un formalisme de supervision basé sur une notion d'apprentissage de langage d'arbres par exemples positifs/négatifs, inspirée de (Niehren *et al.*, 2013). En effet, dans la plupart des cas (notamment dans ses applications « compositionnelles »), le processus de quantification rythmique n'admet pas une solution unique, et au contraire est l'objet de contraintes ou d'attentes spécifiques du compositeur, non nécessairement rationalisables. L'algorithme proposé permet ainsi, à partir des exemples positifs ou négatifs indiqués par l'utilisateur du système, de faire évoluer le langage de l'automate et converger par itérations successives le processus de simplification ou de quantification des arbres vers des solutions souhaitées.

ÉNUMÉRATION ET INTERACTION. Dans son stage de Master 2 ATIAM, Adrien Ycart (2015) [⋆] a prolongé ces précédents travaux en s'intéressant à l'énumération efficace de solutions. Un automate d'arbre pondéré est construit à partir d'une séquence temporelle et d'un *schéma* donné par l'utilisateur, défini de manière récursive comme divisions successives d'unités de temps. Ce schéma détermine un espace de possibilités rythmiques en termes de subdivisions des nœuds dans l'arbre (c'est à dire une « grille » sur laquelle les temps pourront être quantifiés), et guide la recherche de solutions candidates pour la transcription. L'automate d'arbre est alors inclus dans un algorithme d'énumération paresseuse adapté de (Huang et Chiang, 2005), permettant de fournir par paquets de taille constante les k meilleures solutions suivant des critères d'évaluation donnés (approche dite *k-best*). Cet algorithme permet ainsi de traiter la transcription comme un problème d'*énumération multi-critères* selon

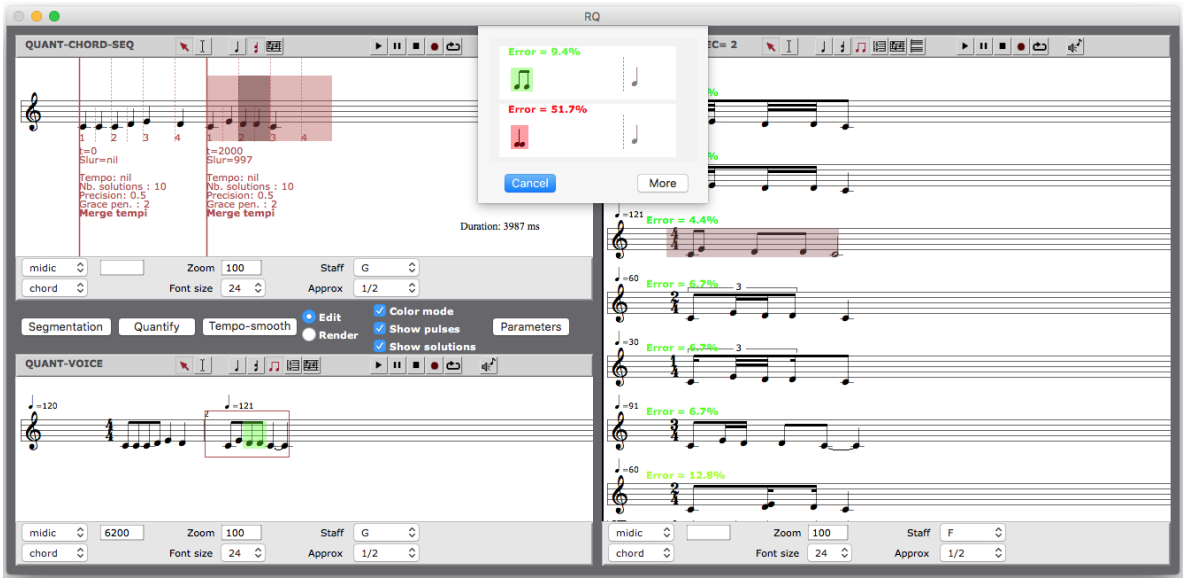


FIGURE 3.5: Interface graphique pour la supervision du processus d'énumération multicritère et de quantification rythmique dans la bibliothèque RQ.

des grilles non-uniformes (basées sur les schémas de subdivisions). Deux critères de qualité sont utilisés : une mesure de précision calculée comme somme des distances point-à-point entre l'entrée et la sortie, et un indicateur de la complexité de l'arbre de sortie calculé à partir de sa taille, des arités de ses nœuds, et du nombre de *grace notes* (appogiatures, ou notes « non quantifiées ») obtenues en sortie.

Ce projet s'est poursuivi en 2016 dans le cadre d'une Unité Projet Innovation (UPI) de l'IRCAM [★], afin de concrétiser ses résultats théoriques dans un outil opérationnel. Une bibliothèque (nommée RQ) a été développée, contenant l'algorithme *k-best* adapté et certaines améliorations, ainsi qu'une nouvelle méthode de couplage des tâches d'estimation de tempo et de quantification, reposant sur une segmentation préalable du flux d'entrée en régions à tempo constant. Sur le modèle de OMKANT, cette bibliothèque propose une interface pour la supervision du processus de quantification, permettant l'édition de la structure temporelle donnée en entrée, la segmentation manuelle ou automatique de cette séquence, la visualisation et la sélection de solutions proposées par l'algorithme *k-best*, ainsi que l'édition locale des solutions via des requêtes partielles à l'automate pour la substitution de sous-parties de l'arbre rythmique (voir figure 3.5) (Ycart *et al.*, 2016a,b) ✍. Cette bibliothèque a été utilisée notamment dans la pièce *East St. Louis Blues* de Alessandro Ratoci (2016).

MULTI-TEMPORALITÉ — COURBES DE TEMPO. Je me suis intéressé en 2013 lors d'une visite au Center for New Music and Audio Technologies (CNMAT) de l'Université de Berkeley à la question des poly-rythmes et des courbes de tempo (*tempo curving*) (Honing, 2001). Le travail de John MacCallum (compositeur/chercheur) et de ses collègues du CNMAT traitait de ces questions de « poly-temporalité » dans les constructions musicales, se rapprochant en certains points des problématiques de rythme et de quantification évoquées précédemment. Principalement considérées dans le contexte d'applications temps-réel, celles-ci ont abouti au développement d'un outil permettant de calculer des correspondances entre temps pulsé et temps réel étant donné une (ou des) courbe(s) de tempo (MacCallum et Schmeder, 2010). Nous avons ensemble esquissé une extension de la problématique dans

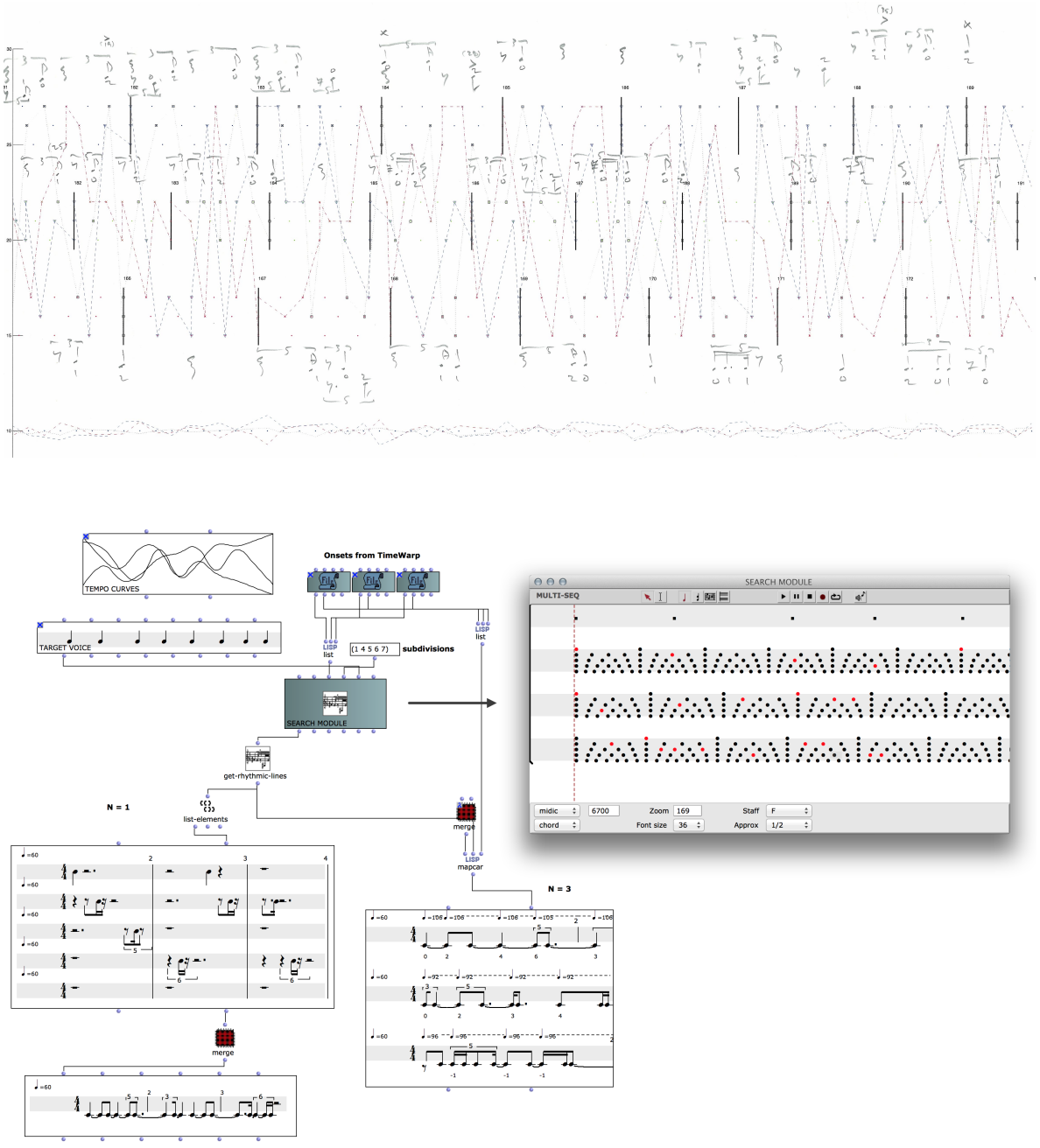


FIGURE 3.6: Haut : Esquisse de la pièce *aberration* (John MacCallum, 2010). Bas : Implémentation du processus de transcription poly-temporel dans OpenMusic.

le domaine de la CAO, afin de traiter efficacement dans OpenMusic une partie des tâches compositionnelles réalisées manuellement à partir de ces données poly-rythmiques, parfois de manière fastidieuse. Les séquences temporelles peuvent en effet être approximées et *quantifiées* sur des grilles non-uniformes définies par les courbes de tempo afin de les noter sur une partition, lisible par un interprète. L'environnement de CAO permet la génération rapide de ces grilles, ainsi qu'une approche efficace et, là encore, supervisée par un éditeur, pour guider les choix des grilles et des approximations suivant des décisions compositionnelles (Bresson et MacCallum, 2015) 🖉. La figure 3.6, illustre ce procédé réalisé sur papier par MacCallum pour l'esquisse de sa pièce *aberration* (2010) et le processus équivalent calculé et supervisé dans OpenMusic.

3.3 MODÉLISATION ET ANALYSE MUSICALE

L'analyse musicale assistée par ordinateur consiste à utiliser des modèles et processus informatiques pour extraire des connaissances à partir d'extraits musicaux donnés sous forme de partition numérique ou sous forme sonore. Ses déclinaisons et applications s'étendent du domaine du MIR (*Music Information Retrieval*) à l'analyse musicale proprement dite — on parlera aussi de « musicologie computationnelle » (Meredith, 2015).

L'environnement OpenMusic permet la modélisation et l'expérimentation rapide de tous types de processus musicaux, et se prête particulièrement bien à ces différents domaines d'application — voir par exemple (Andreatta et Agon, 2003). Un certain nombre d'outils que j'ai développés pour le traitement de données de description musicales et sonores (Bresson et Agon, 2010) 🖉 facilitent le prototypage des processus d'analyse : des applications dans ce domaine furent par exemple l'implémentation du concept d'« analyse spectrale différentielle » avec le compositeur et musicologue Jean-Marc Chouvel (Chouvel *et al.*, 2007) 🖉, ou encore l'utilisation des outils de quantification rythmique dans le cadre de la thèse en musicologie de Olivier Migliore à l'Université de Montpellier, pour l'analyse prosodique d'enregistrements de rap français (Migliore *et al.*, 2014) 🖉.

J'ai également eu l'occasion à différentes reprises de collaborer à la création de bibliothèques OpenMusic dédiées à l'analyse, comme par exemple la bibliothèque SOAL (Sound Object Analysis Library) développée par le groupe Mus3 (*Musicologia, Sonologia e Computação*) de l'Université de João Pessoa au Brésil (Rolim *et al.*, 2005), ou encore à la bibliothèque PARETO, créée à partir de l'ensemble de *patches* OpenMusic du même nom produits par le compositeur Fabien Lévy dans ses précédents travaux en ethno-musicologie.

SEGMENTATION. En 2010 lors d'un séjour invité au sein du groupe Pattern Recognition and Artificial Intelligence du Département *Lenguajes y Sistemas Informaticos* de l'Université d'Alicante (Espagne) nous avons implémenté dans OpenMusic le modèle d'analyse harmonique développé par ce groupe de recherche, distribué depuis sous forme d'une bibliothèque, HARMONIC-ANALYSIS [6]. À cette occasion nous avons mis en place une architecture pour la segmentation des objets musicaux, une notion fondamentale dans la plupart des processus et approches analytiques (Bresson et Pérez-Sancho, 2012) 🖉. Une segmentation peut être pensée comme un découpage dans le temps, ou comme une partition des données (au sens ensembliste) conçue de manière fonctionnelle (groupements d'accords ou de notes suivant des caractéristiques diverses), et interprétée de manières très diverses suivant les objectifs du processus d'analyse. Différents cas de figure et applications ont donc été envisagés : analyse

6. `http://grfia.dlsi.ua.es/cm/projects/drims/software.php`

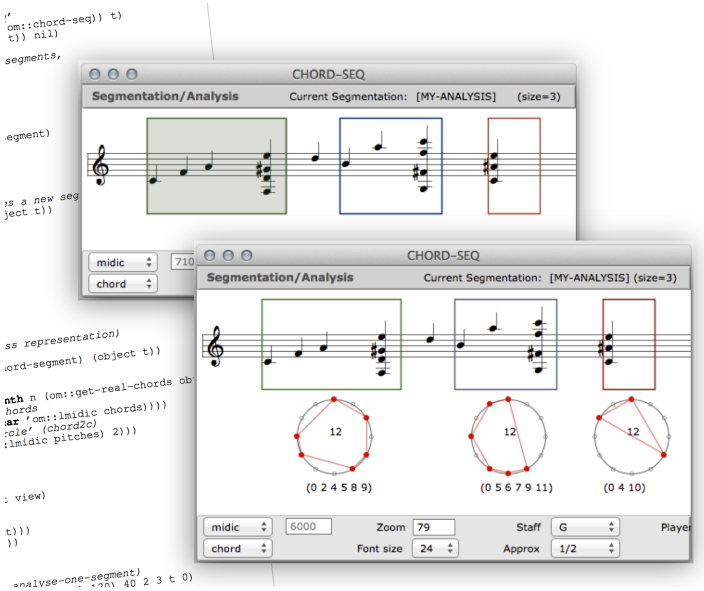


FIGURE 3.7: Segmentation d'une séquence musicale et application d'un processus d'analyse *pitch-class sets* sur les segments.

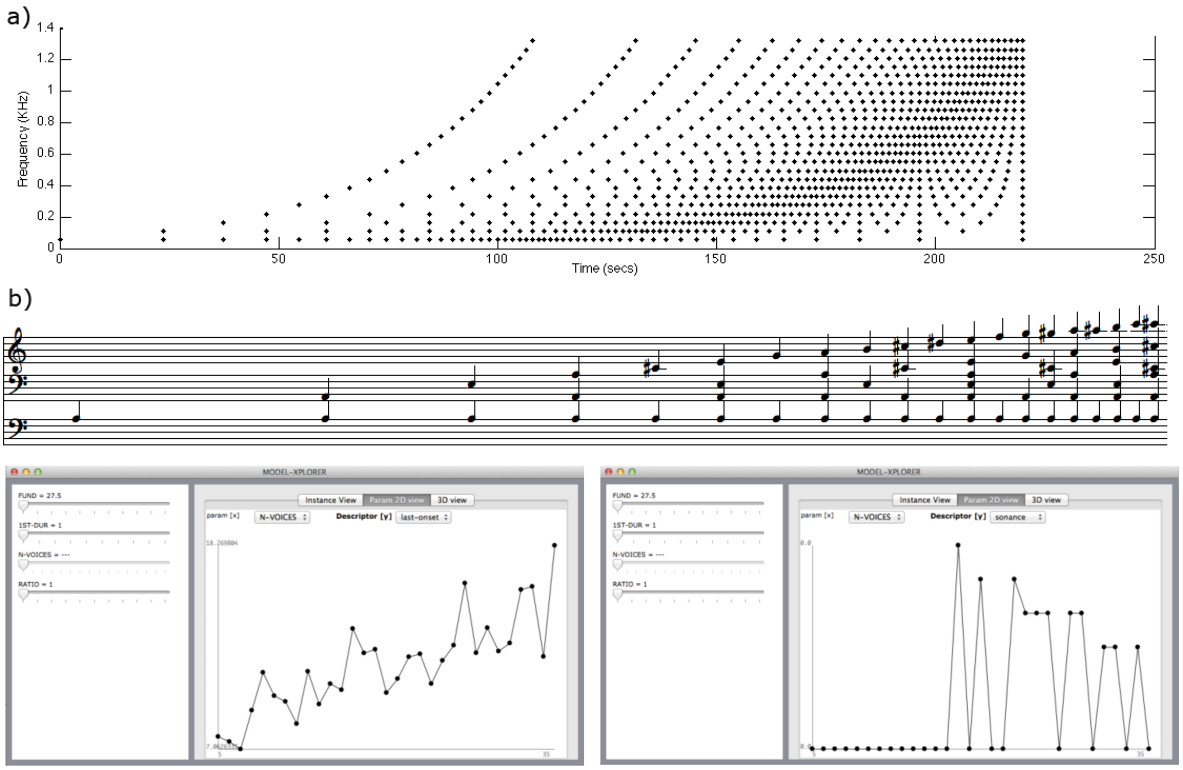


FIGURE 3.8: Modélisation et simulation : application à l'analyse musicale (thèse de Charles de Paiva Santana, UPMC / Université de Campinas).

par accords ou groupes de notes (par exemple en classes de hauteur — voir figure 3.7), segmentation et analyse rythmique (quantification), etc. [7]

MODÉLISATION ET SIMULATION. Dans le cadre du Programme Doctoral International « Modélisation des Systèmes Complexes » de l'Unité Mixte Internationale UMMISCO (Université Pierre et Marie Curie — Institut de Recheche pour le Développement), j'ai encadré la thèse de doctorat de Charles de Paiva (2012-2016) [⋆] en co-tutelle avec l'Université de Campinas (Brésil) et le NICS (*Núcleo Interdisciplinar de Comunicação Sonora*). Celle-ci pose l'hypothèse d'une approche musicologique basée sur les notions de modélisation et de simulation informatiques. La considération d'une œuvre musicale donnée comme instance, ou configuration particulière des paramètres d'un modèle, ouvre des perspectives expérimentales via des démarches de simulation, produisant par exemple des variantes et permettant de tirer des conclusions sur les processus mis en œuvre, ainsi que sur l'équilibre et le caractère « optimal » (ou stable) du cas particulier que constitue une pièce du répertoire, figée sur une partition. Des modèles ont ainsi été produits, autour de pièces ou de processus compositionnels se prêtant à la modélisation informatique, comme ceux réalisés par les compositeurs James Tenney (De Paiva Santana *et al.*, 2013) 🖉, Pierre Boulez ou György Ligeti. À partir de ces modèles, des instances et variations sont produites et analysées à l'aide de critères quantitatifs (par exemple statistiques, sur l'harmonicité, les rapports de durées, etc.) ou d'autres critères plus spécifiques (par exemple ceux calculés par des outils comme la bibliothèque SOAL que nous avons mentionnée plus haut). De ces données analytiques peuvent être dérivés de nouveaux espaces de représentation permettant de situer la pièce dans le contexte de ses possibles variations (De Paiva Santana *et al.*, 2015) 🖉 — cf. figure 3.8.

MATH/MUSIQUE. Entre 2012 et 2014 j'ai coordonné à l'IRCAM le séminaire *MaMuX* (Mathématiques, Musique et relations à d'autres disciplines) initié par Moreno Andreatta une dizaine d'années plus tôt. [8] J'ai créé et animé dans ce contexte et sur cette même période un réseau soutenu par le Réseau National des Systèmes Complexes (RNSC). [9] Celui-ci (entre autres choses) a poussé au premier plan certaines problématiques liées à la modélisation et à des approches mathématiques plus avancées. [10]

En collaboration avec Jean-Louis Giavitto, je me suis intéressé par exemple à l'application du calcul et des représentations spatiales (DeHon *et al.*, 2007) pour la programmation et la modélisation des structures musicales. Suite à la thèse de Louis Bigo réalisée dans l'équipe en co-tutelle avec l'Université de Paris-Est (Bigo, 2013) nous avons ensemble encadré en 2014 le projet de fin d'études de Diego Diverio [⋆] dans le cadre de son parcours à l'ENSIMAG et dans le Master Art, Sciences et Technologies de l'INP Grenoble. Ce stage portait sur l'application des outils théoriques utilisés par Louis Bigo, comme les complexes simpliciaux et les représentations des structures harmoniques sous forme de réseaux hexagonaux (*Tonnetz*), et leur implémentation dans OpenMusic.

7. Cette architecture est aussi à la base du système de segmentation de la bibliothèque de quantification RQ.
8. Voir `http://repmus.ircam.fr/mamux/`
9. La collaboration et le soutien du RNSC s'est poursuivie à partir de 2015 à travers le nouveau réseau *MusICAL* (Interaction, Calcul, Algorithmique et Languages appliqués à la Musique), cette fois à l'échelle nationale sous forme d'un partenariat avec le groupe *Algomus* des Universités de Lille et Picardie Jules Verne (Amiens), le LaBRI (Université de Bordeaux), et Grame (Lyon).
10. Le séminaire MaMuX fut dans les années 2000 une des prémisses de ce qui deviendra la *Society for Mathematics and Computation in Music*, dont nous avons organisé la troisième conférence bi-annuelle en 2011 (Agon *et al.*, 2011) 🖉.

3.4 ORCHESTRATION

Dans le cadre de la thèse de Grégoire Carpentier (Carpentier, 2008), et du projet ANR *SampleOrchestrator*, nous nous sommes intéressés à la question du contrôle du logiciel d'orchestration Orchidée (Carpentier *et al.*, 2010) avec l'aide des outils de CAO (Carpentier et Bresson, 2010) ✎. Orchidée est un serveur dédié à l'orchestration musicale assistée par ordinateur, effectuant des recherches de combinaisons de sons d'instruments d'orchestre se rapprochant le plus possible d'un timbre « cible » donné. J'ai conçu et développé la bibliothèque OpenMusic OM-ORCHIDÉE (voir figure 3.9), proposant une interface graphique pour l'édition de configuration d'orchestres (choix d'instruments et d'instrumentistes), l'édition de cibles à partir de spécifications « symboliques » (hauteurs) et « signal » (propriétés timbrales), la spécification de contraintes et la communication bi-directionnelle avec le serveur de calcul. La bibliothèque proposait également une navigation sommaire dans la liste des solutions proposées par le logiciel d'orchestration, l'écoute et la conversion de ces solutions en partitions, utilisables dans OpenMusic ou exportables vers d'autres logiciels de notation.

Ce projet constitue un premier exemple de système mixte mêlant interactions, environnements externes hétérogènes, et processus de CAO, précurseur des travaux que nous développerons ensuite et décrirons plus spécifiquement dans le chapitre 7.

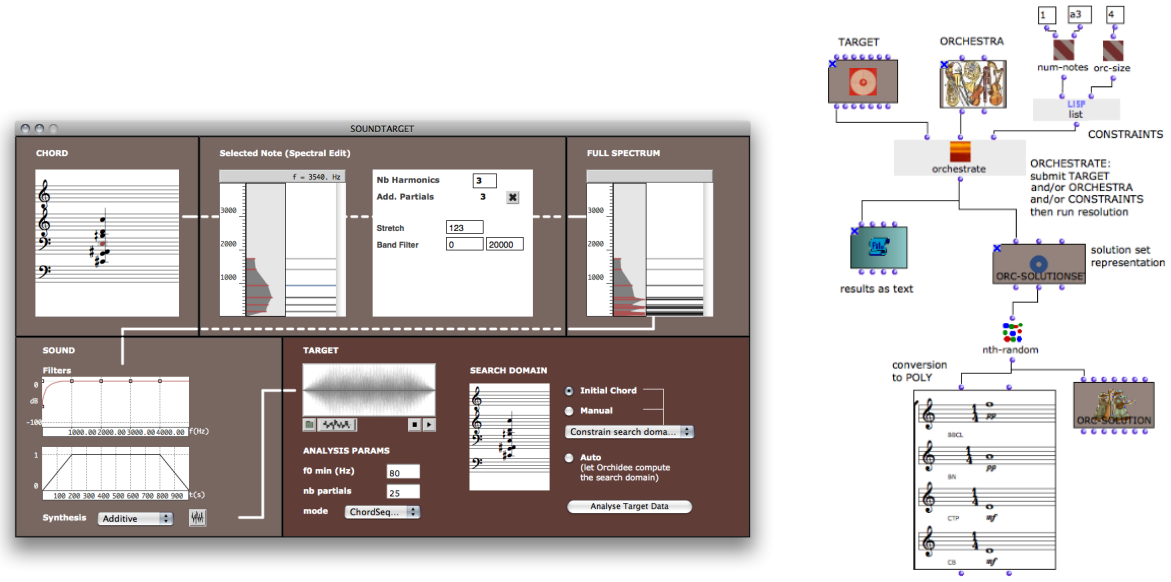


FIGURE 3.9: OM-ORCHIDÉE : interfaces de contrôle des processus d'orchestration dans OpenMusic et communication avec le serveur Orchidée.

3.5 APPLICATIONS DANS LES ARTS GRAPHIQUES

Nous concluons ce chapitre avec un projet réalisé en collaboration avec l'artiste plasticienne Shanta Rao. *Skyline* est une installation vidéo présentée dans le cadre de la Biennale de Belleville et de la manifestation Nuit Blanche 2010. Shanta Rao s'est intéressée dans ce projet à la question de l'apparition des formes et à l'*exploration des possibles du devenir d'un objet* : un objet « source » qui s'émancipe de son champ d'origine pour devenir un objet nouveau. Elle utilise l'environnement de CAO pour générer non pas des structures musicales, mais des tableaux de données en deux-dimensions représentant des images. La bibliothèque PIXELS créée à cet effet propose des fonctionnalités dédiées au traitement de tableaux de pixels, où

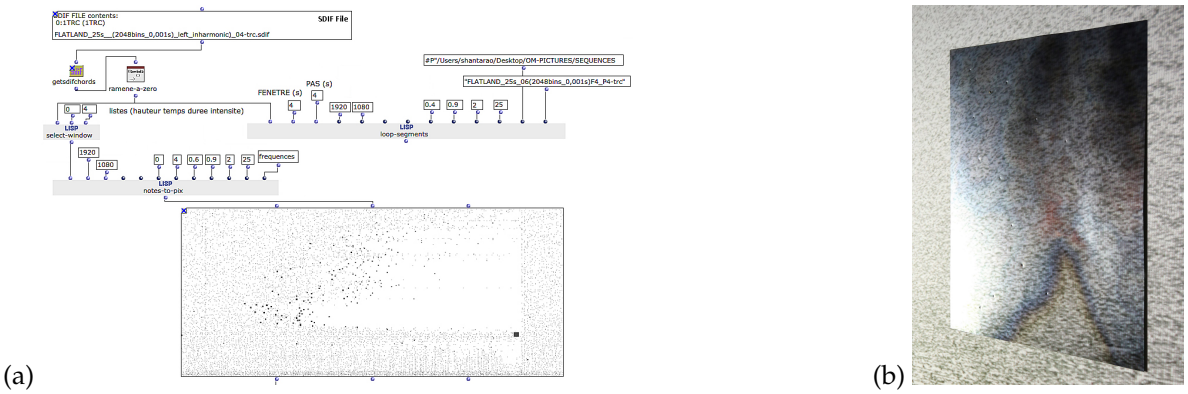
chaque point (pixel) est déterminé par une valeur entre 0 et 1 (pour des images en niveau de gris), ou un triplet (rouge, vert, bleu) pour des images en couleur.

Pour cette installation l'artiste a généré un ensemble d'images à partir de lignes d'horizon tracées sur des photographies d'archives de territoires dévastés par les bombardements de la seconde guerre mondiale. Les courbes, tracées et enregistrées dans un éditeur, sont transformées en séries de valeurs et niveaux de gris, et différents traitements permettent d'en faire varier les seuils et contrastes. La figure 3.10-a montre un exemple simplifié de ce procédé. [11] Un ensemble d'images est produit de cette manière, à partir de différentes sources, assemblées en séquence pour former l'animation projetée sur l'installation (figure 3.10-b).

En 2016, Shanta Rao a également exposé plusieurs œuvres utilisant des projections de motifs générés dans OpenMusic à partir de données issues d'analyses spectrales d'enregistrements sonores (voir figure 3.11). Nous aborderons ce volet de mes travaux sur l'utilisation des données d'analyse et de description sonore dans le chapitre suivant.

(a) (b)

FIGURE 3.10: *Skyline* (2010) installation, dimensions variables, Biennale de Belleville & Nuit Blanche, École Nationale Supérieure d'Architecture de Paris-Belleville. (a) *Patch* OpenMusic générant des images à partir de lignes d'horizon tracées dans un éditeur. (b) Vues de l'installation.



(a) (b)

FIGURE 3.11: *How the Stranger vainly endeavoured to reveal to me in words the mysteries of Flatland* (2016), installation, dimensions variables, Karlin Studios/Centre d'art contemporain Futura, Prague, République Tchèque. (a) *Patch* OpenMusic générant les images à partir de données d'analyse et de description sonore (cf. chapitre 4, section 4.1). (b) Détail de la projection.

11. Dans *The OM Composer's Book 3* [5], le compositeur Luiz Castelões décrit des procédés comparables d'analyse d'images et de contours pour générer des structures musicales — cf. figure A.2 en annexe.

PUBLICATIONS (CHAPITRE 3)

AGON, C., AMIOT, E., ANDREATTA, M., ASSAYAG, G., BRESSON, J. et MANDEREAU, J., éditeurs (2011). *Mathematics and Computation in Music (Proceedings of the 3rd International Conference)*, volume 6726 de *Lecture Notes in Artificial Intelligence*, Paris, France. Springer.

AGON, C. et BRESSON, J. (2007). Mixing Time Representations in a Programmable Score Editor. *In Proceedings of the Sound and Music Computing conference (SMC'07)*, Lefkada, Greece.

BRESSON, J. et AGON, C. (2006). Temporal Control over Sound Synthesis Processes. *In Proceedings of the Sound and Music Computing conference (SMC'06)*, Marseille, France.

BRESSON, J. et AGON, C. (2008). Scores, Programs and Time Representations : The Sheet Object in OpenMusic. *Computer Music Journal*, 32(4).

BRESSON, J. et AGON, C. (2010). Processing Sound and Music Description Data Using OpenMusic. *In Proceedings of the International Computer Music Conference*, New York / Stony Brook, USA.

BRESSON, J. et AGON, C. (2011). Visual Programming and Music Score Generation with OpenMusic. *In IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Pittsburg, USA.

BRESSON, J. et MACCALLUM, J. (2015). Tempo Curving as a Framework for Interactive Computer-Aided Composition. *In Proc. Sound and Music Computing (SMC'15)*, Maynooth, Ireland.

BRESSON, J. et PÉREZ-SANCHO, C. (2012). New Framework for Score Segmentation and Analysis in OpenMusic. *In Proc. Sound and Music Computing conference (SMC'12)*, Copenhagen, Denmark.

CARPENTIER, G. et BRESSON, J. (2010). Interacting with Symbol, Sound and Feature Spaces in Orchidée, a Computer-Aided Orchestration Environment. *Computer Music Journal*, 34(1).

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 Electroacoustic Music Studies Network Conference (EMS'07)*, Leicester, UK.

DE PAIVA SANTANA, C., BRESSON, J. et ANDREATTA, M. (2013). Modeling and Simulation : The Spectral CANON for CONLON Nancarrow by James Tenney. *In Proceedings of the Sound and Music Computing conference (SMC'13)*, Stockholm, Sweden.

DE PAIVA SANTANA, C., MANZOLLI, J., BRESSON, J. et ANDREATTA, M. (2015). Towards a Borgean Musical Space : An Experimental Interface for Exploring Musical Models. *In Electronic Visualisation and the Arts (EVA'15)*, London, United Kingdom.

JACQUEMARD, F., DONAT-BOUILLUD, P. et BRESSON, J. (2015). A Structural Theory of Rhythm Notation based on Tree Representations and Term Rewriting. *In Proceedings of Mathematics and Computation in Music*, Springer *Lecture Notes in Artificial Intelligence*, 9110, London, UK.

MIGLIORE, O., OBIN, N. et BRESSON, J. (2014). Rap, ragga et punk français (1977-1992) : une analyse computationnelle. *Journées d'Analyse Musicale de la Sfam (JAM)*. Poster.

YCART, A., BRESSON, J., JACQUEMARD, F. et STAWORKO, S. (2016a). Une approche interactive pour la transcription rythmique dans OpenMusic. *In Journées d'Informatique Musicale*, Albi, France.

YCART, A., JACQUEMARD, F., BRESSON, J. et STAWORKO, S. (2016b). A Supervised Approach for Rhythm Transcription Based on Tree Series Enumeration. *In Proceedings of the International Computer Music Conference (ICMC'16)*, Utrecht, The Netherlands.

4

TRAITEMENT ET SYNTHÈSE SONORE EN CAO

Depuis l'apparition des synthétiseurs analogiques, numériques, puis des langages de programmations dédiés à la synthèse sonore, et à la suite des travaux pionniers de compositeurs comme Jean-Claude Risset, Karlheinz Stockhausen, John Chowning et bien d'autres depuis, synthèse et traitements du son font désormais partie intégrante des démarches musicales et des processus compositionnels contemporains (Risset, 1993). Il existe de multiples façons de concevoir et modéliser les signaux sonores, et le traitement ou la production de ces signaux dans un *contexte de langage musical* (Eckel, 1993) reste aujourd'hui encore problématique. En ce sens, la CAO est susceptible de favoriser une approche « compositionnelle », permettant de formaliser les structures, données et processus de description et de génération du son (Bresson et Agon, 2009) 🖉.[1]

L'application des principes de la CAO au domaine de la création sonore a été l'objet de ma thèse de doctorat, intitulée *La synthèse sonore en composition musicale assistée par ordinateur* (Bresson, 2007) 🖉.[2] Cette idée s'est ensuite prolongée dans un ensemble de directions et projets. Dans le contexte d'une utilisation de plus en plus marquée des techniques de traitement et synthèse sonore en composition musicale, la proposition principale qui y était développée est donc celle d'un environnement de CAO permettant l'*écriture* du son (au sens de l'« écriture musicale ») (Bresson et Agon, 2006) 🖉 et pour cela, d'une nouvelle conception du son comme *objet d'un modèle compositionnel* représenté sous forme de programmes visuels (Bresson et Agon, 2007) 🖉.

Un certain nombre de questions posées alors sont devenues des thématiques récurrentes dans les projets que j'ai menés par la suite. Dans ce chapitre, nous revenons sur certains aspects de ce travail afin de mieux appréhender leur implication et leur impact, et de rendre compte de l'application effective des outils développés depuis leur mise à disposition pour les utilisateurs. Nous abordons également différents prolongements de ces travaux à travers la supervision de travaux universitaires.

4.1 DESCRIPTION ET TRAITEMENT DES SIGNAUX

L'assimilation du matériau sonore dans le domaine de la composition musicale nécessite le recours à des représentations spécifiques, à la fois adaptées aux contraintes de performance liées au traitement numérique du signal, et porteuses de sens pour le compositeur (ou l'utilisa-

1. Risset (1977) mettait déjà sur un même plan techniques d'écriture et recherche de timbres sonores, et se positionnait en faveur d'une approche calculatoire faisant intervenir l'humain pour une utilisation efficace, expressive et personnelle de la machine. De ce point de vue, le compositeur s'inscrivait parmi les précurseurs des principes de la CAO — qui n'avait pas encore de nom à l'époque — appliqués au domaine de la recherche sonore.

2. Université Pierre et Marie Curie, Paris 6. Sous la direction de Carlos Agon. Soutenue le 30/11/2007.

teur d'un système informatique en général). [3] Programmeurs et musiciens se retrouvent ainsi fréquemment confrontés au traitement de données temporelles complexes (nous aborderons la question du temps dans le chapitre suivant), intégrées dans différents formalismes de programmation et de calcul. Parmi ces données, nous incluons par exemple les représentations sonores produites par les procédés d'analyse (analyse « spectrale », « descripteurs de haut niveaux », etc.), ou celles destinées à contrôler les processus de traitement et de synthèse sonore. À mi-chemin entre les domaines du signal et de la formalisation musicale (on parlera aussi d'une dualité « signal/symbole »), elles peuvent être mono- ou multidimensionnelles, et échantillonnées à des taux variables, du taux dit de « contrôle » (de l'ordre de quelques millisecondes), au taux d'échantillonnage des signaux sonores numériques (entre 44 et 96 kHz), ou encore au niveau « macro-temporel » des évènements musicaux.

Le transfert et la représentation de telles données entre le domaine de l'écriture et celui du traitement du signal, ou du traitement informatisé de l'information musicale en général, sont présents de façon plus ou moins explicite dans la plupart des projets de recherche en informatique musicale. Certains travaux tels que (Camurri, 1990) ou (Leman, 1993) ont proposé des bases pertinentes pour une étude de cette problématique, par exemple avec la mise en avant de la distinction *symbolique/subsymbolique*, reprise dans (Hanappe et Assayag, 1998) ou dans (Bresson et Agon, 2007) ✎. Ces questions furent en réalité l'angle initial par lequel j'ai initié mon approche de l'informatique musicale lors de mon projet de fin d'études d'ingénieur, qui portait sur la représentation des données de descriptions sonores au format SDIF (Bresson, 2003) ✎.

SDIF (Sound Description Interchange Format) [4] (Schwarz et Wright, 2000) est un format de description utilisé par différentes applications réalisant des analyses ou traitements sonores. Les descriptions du son y sont stockées dans des trames (*frames*) étiquetées par une date, un ensemble ouvert et extensible de *types* et par un identifiant de flux (*stream ID*) permettant d'organiser les données sous forme de flux temporels parallèles, à taux d'échantillonnage variables. Les données dans chaque trame sont des matrices décrivant un certain nombre de paramètres, ou « champs de description », pour un certain nombre de composants. SDIF propose donc un support générique pour la représentation de descriptions sous une forme relativement flexible et ouverte de flux multidimensionnels.

J'ai développé l'application SDIF-EDIT (voir figure 4.1) [5] utilisant OpenGL pour la visualisation 3D de données de description sonore au format SDIF — typiquement, les analyses sonores (suivi de fondamentale, spectrogramme, suivi de partiels, etc.) exportées par des outils d'analyse comme le logiciel AudioSculpt de l'IRCAM, et par la suite générées directement dans OpenMusic.

Dans un article présentant ce travail (Bresson et Agon, 2004) ✎, nous avons également proposé de concevoir les données au format SDIF de façon plus abstraite, comme support générique pour les descriptions sonores dans l'environnement OpenMusic. Ainsi un ensemble d'outils pour la manipulation des primitives du format ont été définis, constituant

3. *"[...] two domains with very different requirements will coexist in one sole environment : The Signal-Processing Domain – with its strong demand for computational power – and the Symbolic Processing Domain (constituted of as many logical layers with their abstraction and representational tools as needed for the composition of a piece of music), urging for multiple perspectives and flexibility in expression"* (Eckel et Gonzalez-Arroyo, 1994).

4. `http://sdif.sourceforge.net/`

5. SDIF-EDIT a reçu le *Prix spécial du jury* au concours de logiciels musicaux *LoMus 2006* de l'Association Française d'Informatique Musicale (AFIM). → `http://repmus.ircam.fr/bresson/projects/sdifedit`

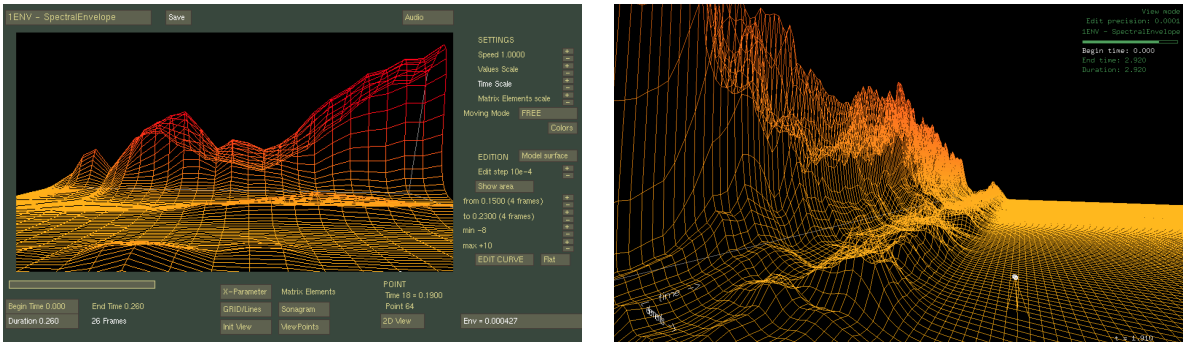


FIGURE 4.1: L'application SDIF-Edit : visualisation 3D de données de description sonore.

un pendant « signal » de fonctionnalités développées précédemment pour le traitement des données musicales au format MIDI (Bresson et Agon, 2010) ✎. Associés aux facilités de programmation proposées dans OpenMusic (itérations, fonctions d'ordre supérieur, etc.), ces outils ouvrent la voie à de nouvelles approches pour le traitement des données musicales de bas niveau, mais également à des applications dans le domaine du MIR (*Music Information Retrieval*) ou de l'analyse musicale. [6] Un exemple de manipulation de données SDIF est donné dans [🕮 10] par le compositeur Aaron Einbond, avec un procédé de mosaïque audio réalisé à partir de données de description obtenues par la segmentation et l'analyse d'extraits sonores — voir figure A.4 en annexe. Nous mentionnons aussi ces outils dans (Einbond *et al.*, 2009) ✎.

GESTE. La notion de « geste musical » est souvent abordée par les compositeurs, de manière plus ou moins abstraite [7] ou technique (Godøy et Leman, 2010), avec en particulier le développement récent de nombreux outils et technologies liées à l'apprentissage, au *mapping* (Wanderley, 2002), au traitement ou à la re-synthèse des mouvements et des gestes musicaux (Bevilacqua *et al.*, 2011; Françoise, 2013; Caramiaux *et al.*, 2014).

Marlon Schumacher s'est intéressé à cette notion de geste dans son doctorat de l'Université McGill (2016) [★], dans le but d'intégrer les signaux issus de gestes « physiques » dans les processus de composition assistés par ordinateur (Schumacher et Wanderley, 2017). [8] La bibliothèque OM-GESTE qu'il a créée propose des outils pour la représentation des données gestuelles comme signaux multidimensionnels basés sur des descriptions encodées dans le format SDIF (voir figure 4.2). Le *mapping* de ces données dans des objets musicaux à différentes échelles et résolutions temporelles permet d'insérer ces descriptions dans des processus compositionnels pour la production de structures musicales symboliques (partitions) ou la synthèse sonore.

6. Nous avons cité dans le chapitre précédent une application avec l'exemple de l'analyse musicale « spectrale différentielle » (Chouvel *et al.*, 2007) ✎.

7. Voir par exemple le texte de Jean-Luc Hervé dans *The OM Composer's Book 1* (Hervé et Voisin, 2006).

8. La thèse de Marlon Schumacher portait plus généralement sur la composition (assistée par ordinateur) « de l'espace, du geste et du son ». L'aspect spatialisation est un point fondamental, et probablement l'une des principales contributions de ce travail. Nous y reviendrons dans le chapitre 6. Nous reviendrons également sur d'autres aspects liés au traitement et à la synthèse sonore dans la section 4.3.

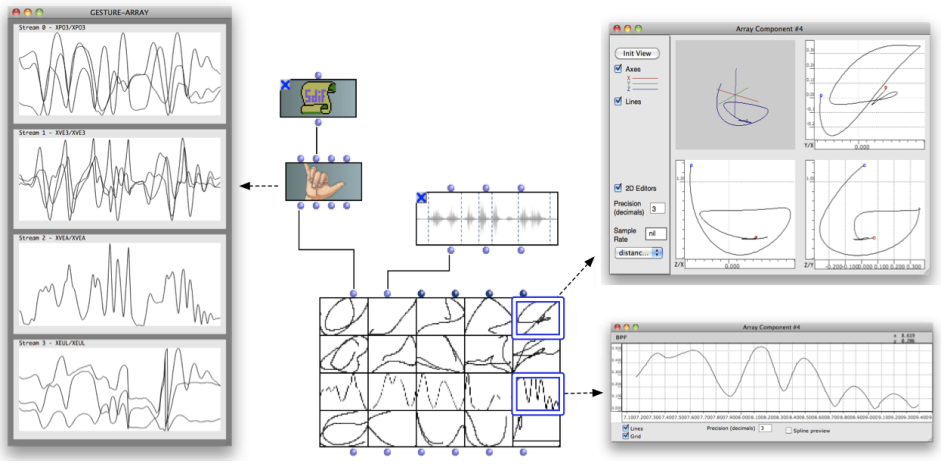


FIGURE 4.2: OM-GESTE : représentation et traitement des données gestuelles dans OpenMusic.

4.2 BIBLIOTHÈQUES POUR L'ANALYSE ET LA SYNTHÈSE SONORE

J'ai développé ou contribué au développement de plusieurs bibliothèques dédiées au traitement et à la synthèse sonore (Bresson, 2006) 🖉. La plupart de ces bibliothèques (dont celles présentées ci-après) font appel à des outils et programmes dédiés pour effectuer les calculs et traitements dans le domaine du signal sonore (notamment, le langage Csound et d'autres outils développés par l'équipe Analyse/Synthèse de l'IRCAM). Les processus développés dans l'environnement de CAO deviennent ainsi les contrôleurs d'un ensemble de programmes externes permettant d'étendre le champ compositionnel au domaine sonore (Bresson *et al.*, 2005) 🖉.

CSOUND est un langage de référence pour la synthèse sonore.[9] Un programme Csound est constitué d'un ensemble d'« instruments » de synthèse composés de modules élémentaires de génération ou traitement des signaux numériques, activés par des instructions et paramètres spécifiés dans une « partition » (*score*). La bibliothèque OM2CSOUND est issue de travaux originaux de Laurent Pottier dans Patchwork (CSOUND-EDITSCO), qui furent portés sur OpenMusic puis adaptés/homogénéisés aux outils d'analyse/synthèse dans les versions récentes de l'environnement. Elle permet la création d'instruments sous forme de graphe (ou écrits en texte), le paramétrage de ces instruments via le formatage des *scores*, et la production des sons par des appels externes au moteur de synthèse. Les chapitres de Laurent Pottier dans *The OM Composer's Book 2* (Pottier, 2008) et de Takeshi Tsuchiya dans *The OM Composer's Book 3* [🕮 15] sont des exemples d'utilisation de cette bibliothèque.

SUPERVP est un moteur d'analyse et de traitement des sons basé sur l'analyse spectrale, développé par l'équipe Analyse/Synthèse de l'IRCAM. Il permet de réaliser des analyses temps/fréquence de type STFT (*short-term Fourier transform*), estimation de fréquence fondamentale, détection de transitoires, ou analyse de formants, et des filtrages et transformations comme le *time-stretching* (étirement/compression temporelle), le *pitch-* ou le *frequency-shifting* (transpositions sans modification temporelle), etc. La bibliotèque OM-SUPERVP, que j'ai créée en collaboration avec Jean Lochard du département de la pédagogie de l'IRCAM, permet de contrôler ces différentes analyses et traitements depuis l'environnement de CAO.

9. `http://www.csounds.com/`

Le chapitre de Hans Tutschku dans *The OM Composer's Book 2* est un exemple d'utilisation de cette bibliothèque (Tutschku, 2008). Dans *The OM Composer's Book 3* une proportion importante d'auteurs en mentionne l'utilisation plus ou moins systématique, dénotant une nette popularisation parmi la nouvelle génération de compositeurs : Fabio de Sanctis de Benedictis [🕮 17] (cf. figure A.6 en annexe), Aaron Einbond [🕮 10], Jean-Luc Hervé [🕮 14], Matthew Schumaker [🕮 8], Takeshi Tsuchiya [🕮 15], Alessandro Ratoci [🕮 20]...

PM2 est un autre moteur d'analyse/synthèse sonore, également développé à l'IRCAM. Basé sur la représentation « additive » des signaux, soit une modélisation du son par « partiels » — composants sinusoïdaux pouvant être reproduits par des oscillateurs (McAulay et Quatieri, 1986), il permet de réaliser les analyses dites de *partial tracking* (harmonique/inharmonique), l'analyse *chord-sequence*, ainsi que le processus inverse de synthèse additive. La bibliothèque OM-PM2 permet de contrôler dans OpenMusic ces processus, fondamentaux dans les démarches compositionnelles dérivées du spectralisme. Matthew Schumaker et Alessandro Ratoci utilisent cette bibliothèque dans [🕮 8] et [🕮 20] (cf. figure A.3 en annexe).

4.3 MONTAGE ET MANIPULATIONS AUDIO-NUMÉRIQUES

Le montage audio fut la base des premières expérimentations en musique électroacoustique, notamment à travers le mouvement de musique « concrète » (Schaeffer, 1952). Curieusement, ses ramifications dans le domaine de la CAO ne se sont développées que bien après des techniques plus avancées, comme celles d'analyse et de synthèse sonore vues dans la section précédente.

TRAITEMENT ALGORITHMIQUE DES RESSOURCES AUDIO. En 2005 j'ai mis en place dans OpenMusic une nouvelle architecture audio multi-plateforme basée sur la bibliothèque LibAudioStream [10] développée à Grame. [11] Cette bibliothèque est basée sur le concept de *stream*, abstraction d'un signal audio pouvant subir des transformations via des opérateurs composés suivant un modèle fonctionnel. Elle permet de réaliser un ensemble d'opérations sur les ressources sonores (coupes, mixage, séquences, etc.) lors du rendu audio temps réel (via le *player* de l'environnement) ou au sein de programmes visuels, pour la génération *offline* de fichiers audio. Une première génération d'outils de traitement algorithmique des sons a alors été développée et intégrée dans OpenMusic (Bresson, 2006) 🖋. Quelques années plus tard j'encadrais le stage d'ingénieur de Dimitri Bouche (2013) [★], au cours duquel nous avons réalisé une refonte de cette architecture audio, et un portage de ces outils algorithmiques de montage audio en LISP, via une gestion optimisée des tableaux (*arrays*) et des accès mémoire. Alessandro Ratoci [🕮 20] donne un exemple d'utilisation de ces outils pour une application de découpage/remontage algorithmique d'enregistrements audio (voir figure A.7 en annexe).

OM-FAUST. Le principal objectif et la contribution majeure du sujet de stage proposé à Dimitri Bouche consistait en l'établissement d'une connexion entre l'environnement OpenMusic et le langage de synthèse et de traitement sonore Faust (Bouche *et al.*, 2014) 🖋. Faust (Orlarey *et al.*, 2004) est un langage fonctionnel pour le traitement audio temps réel. [12] Ses

10. `http://libaudiostream.sourceforge.net/`
11. Grame, Centre National de Création Musicale (Lyon). `http://www.grame.fr`
12. `http://faust.grame.fr/`

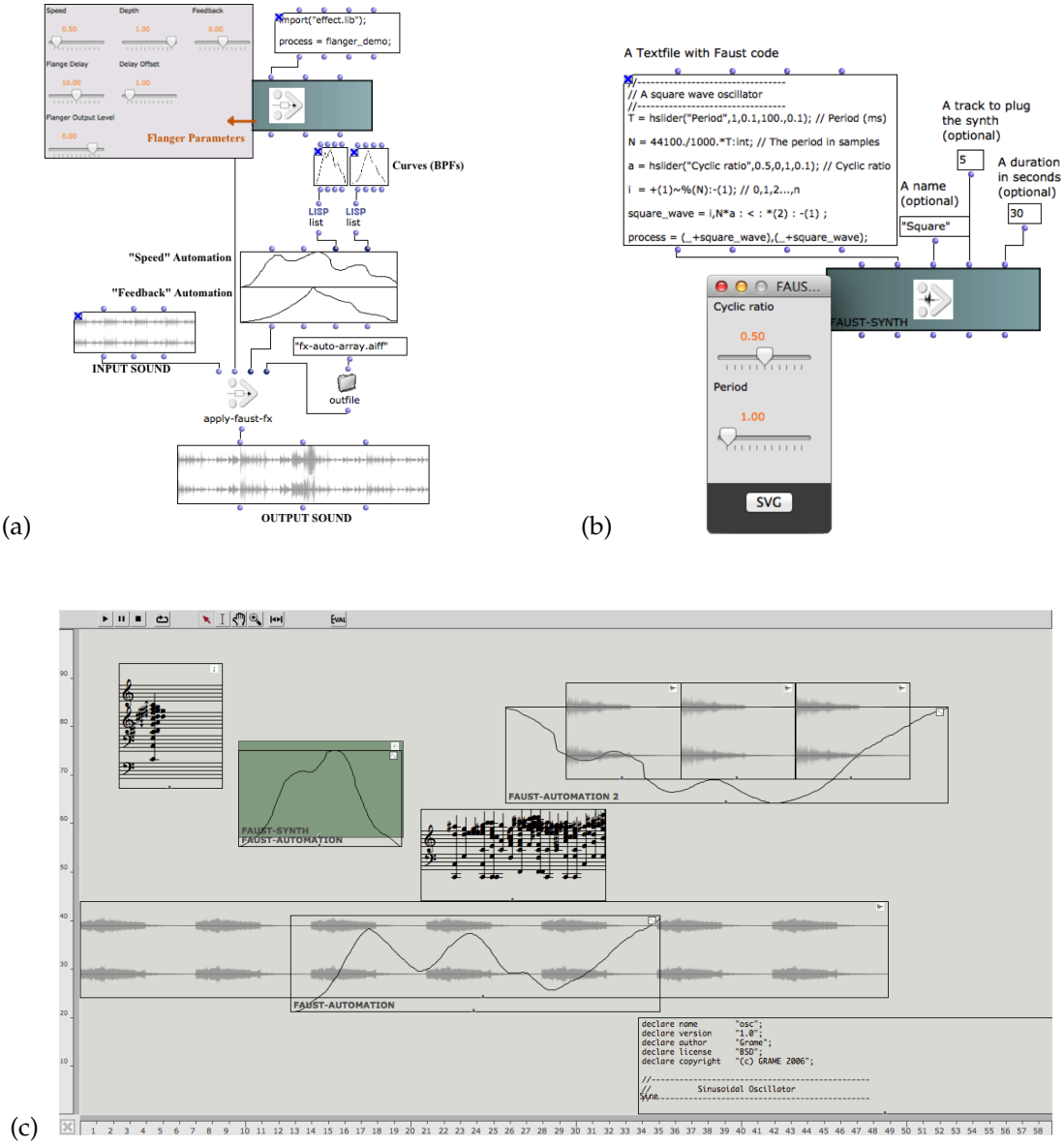


FIGURE 4.3: OM-FAUST : traitement et synthèse sonore temps réel/temps différé. (a) Traitement et contrôle audio temps-différé. (b) Génération automatique d'un panneau de contrôle pour un effet Faust. (c) Organisation de sons, d'effets temps-réel et d'automations dans une structure temporelle (*maquette*).

programmes peuvent être compilés à la volée (Orlarey *et al.*, 2014) et ainsi entrer dans le flux de traitement, synthèse ou de rendu audio de la bibliothèque LibAudioStream. OM-FAUST, réalisée à partir de ces technologies, permet donc de compiler des effets ou synthétiseurs écrits en Faust dans OpenMusic, et de les insérer dans les chaînes de traitement des ressources audio. Ceux-ci peuvent opérer en temps différé, au sein des programmes visuels exécutés dans l'environnement (voir figure 4.3-a), ou en temps réel sur les différentes pistes de la sortie audio. Les programmes Faust, par ailleurs, contiennent une description de leur couche de contrôle, c'est à dire une documentation structurée de leurs différents paramètres incluant une plage de variation, voire une description d'objet graphique (*widget*) destiné à opérer ce contrôle (potentiomètre, *slider*, etc.). OM-FAUST peut ainsi construire automatiquement une interface graphique sur les effets et synthétiseurs Faust compilés, permettant de régler, voire de contrôler en temps réel leurs paramètres (figure 4.3-b). Enfin, des objets d'automation ont été créés, permettant de programmer de tels contrôleurs sous forme de courbes, également activées en temps différé sur les programmes ou en temps réel lors de la restitution audio (figure 4.3-c).

Ce travail constitua une base importante pour une exploration des relations entre calculs en temps différé et en temps réel, particulièrement dans les processus liant les domaines de la composition et du traitement du signal audio. Nous reviendrons sur ces idées dans les dernières sections de ce mémoire.

OM-SOX. Toujours dans le domaine de la manipulation des signaux audio, Marlon Schumacher a également développé durant sa thèse [★] la bibliothèque OM-SOX, qui fournit une interface dans OpenMusic vers les fonctionnalités de SoX, un outil open-source en ligne de commande proposant un nombre important d'opérations pour la manipulation des fichiers audio, la conversion entre formats, ou l'application d'effets et de transformations sonores. [13] OM-SOX comprend un ensemble de classes (représentant les données) et de fonctions (représentant les processus de traitement) pouvant être connectés pour former un graphe de traitement audio multi-canal. Ici encore, ces traitements peuvent être appliqués en mémoire, et leur sortie restituée dynamiquement sur un périphérique audio temps-réel ; un autre pas en direction de systèmes hybrides entre composition et interaction musicale.

Dans (Garcia *et al.*, 2015) 🖉, une application de cette bibliothèque est présentée, réalisant un processus de spatialisation sonore (voir figure 4.4). Marlon Schumacher l'utilise également dans [🕮 21] pour réaliser automatiquement une banque de sons à partir d'enregistrements de notes jouées par un piano mécanique.

OM-PURSUIT. Un autre volet de la thèse de Marlon Schumacher portait enfin sur la « synthèse par décomposition atomique », ou *corpus-based atomic decomposition*, (CBAD) un principe visant à s'approcher d'un son cible par l'accumulation de grains sonores sélectionnés dans un corpus. La bibliothèque OM-PURSUIT, réalisée dans ce contexte, propose un système d'analyse et de recomposition basé sur un algorithme glouton de *matching pursuit* (Mallat et Zhang, 1993), dont la sous-optimalité est utilisée à des fins compositionnelles : un nombre d'itérations plus ou moins élevé permettra de plus ou moins rapprocher la recomposition atomique du son cible, et le jeu sur les bases d'échantillons permet d'explorer des variantes des modèles sonores utilisés (voir figure 4.5). La pièce *Ab-Tasten* (2011) [🕮 21] est une application de cette approche et de la bibliothèque OM-PURSUIT.

13. `http://sox.sourceforge.net/`

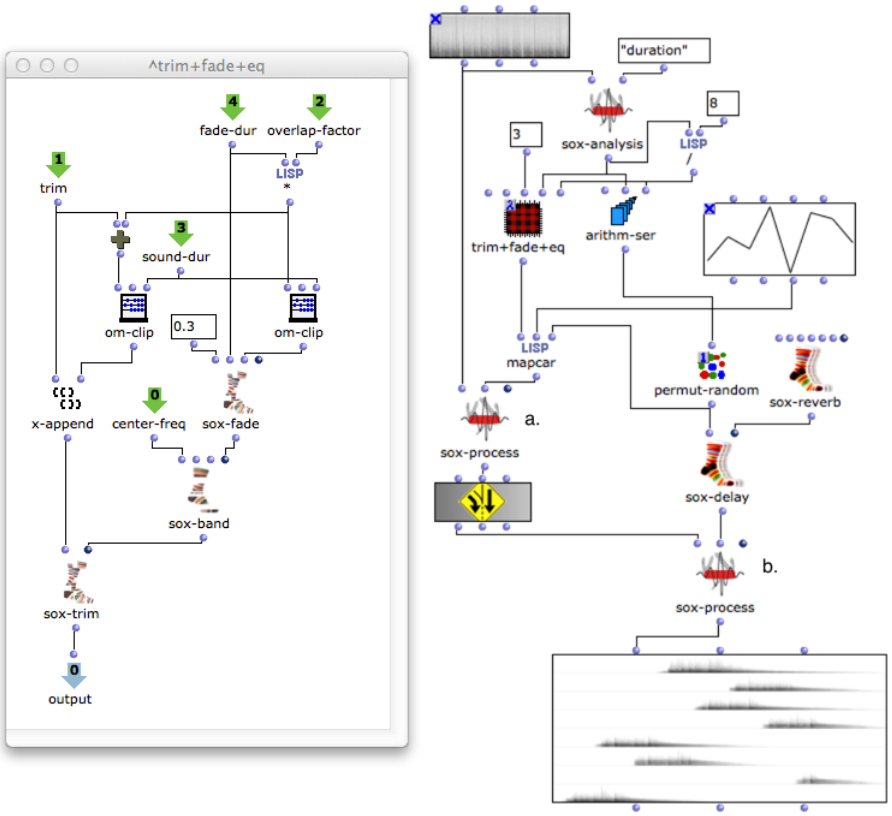


FIGURE 4.4: Traitement fonctionnel de données pour la réalisation d'un fichier audio multicanal avec OM-SOX.

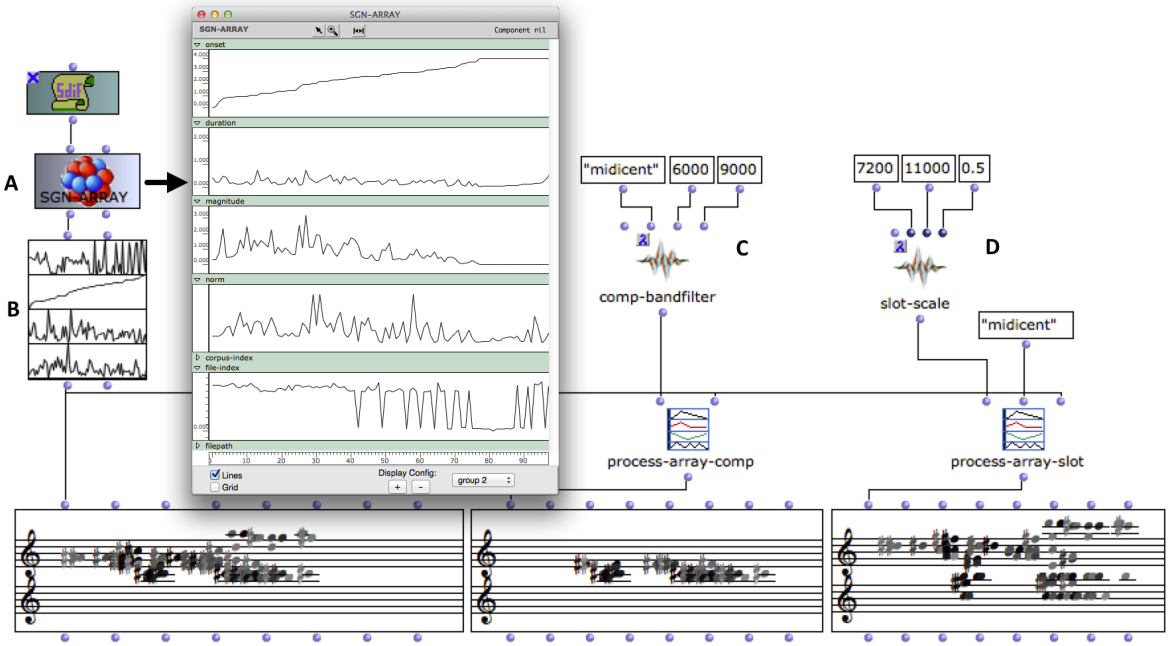


FIGURE 4.5: OM-PURSUIT : *Corpus-Based Atomic Decomposition*.

PUBLICATIONS (CHAPITRE 4)

BOUCHE, D., BRESSON, J. et LETZ, S. (2014). Programmation and Control of Faust Sound Processing in OpenMusic. *In Joint International Computer Music / Sound and Music Computing Conferences (ICMC-SMC'14)*, Athens, Greece.

BRESSON, J. (2003). Représentation et manipulation de données d'analyse sonore pour la composition musicale. Rapport de stage. ESSI – Polytech'Nice-Sophia Antipolis.

BRESSON, J. (2006). Sound Processing in OpenMusic. *In Proceedings of the International Conference on Digital Audio Effects (DAFx'06)*, Montreal, Canada.

BRESSON, J. (2007). *La synthèse sonore en composition musicale assistée par ordinateur : modélisation et écriture du son*. Thèse de doctorat, Université Pierre et Marie Curie, Paris, France.

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 (ICMC'04)*, Miami, USA.

BRESSON, J. et AGON, C. (2006). 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, J. et AGON, C. (2007). Musical Representation of Sound in Computer-aided Composition : A Visual Programming Framework. *Journal of New Music Research*, 36(4).

BRESSON, J. et AGON, C. (2009). Synthèse sonore et composition musicale : Problématiques et outils d'écriture électroacoustique. *In Electroacoustic Music Studies Network Conference (EMS'09)*, Buenos Aires, Argentina.

BRESSON, J. et AGON, C. (2010). Processing Sound and Music Description Data Using OpenMusic. *In Proceedings of the International Computer Music Conference*, New York / Stony Brook, USA.

BRESSON, J., STROPPA, M. et AGON, C. (2005). Symbolic Control of Sound Synthesis in Computer-Assisted Composition. *In Proceedings of the International Computer Music Conference (ICMC'05)*, Barcelona, Spain.

BRESSON, J., STROPPA, M. et AGON, C. (2007). Generation and Representation of Data and Events for the Control of Sound Synthesis. *In Proceedings of the Sound and Music Computing conference (SMC'07)*, Lefkada, Greece.

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 Electroacoustic Music Studies Network Conference (EMS'07)*, Leicester, UK.

EINBOND, A., SCHWARZ, D. et BRESSON, J. (2009). Corpus-Based Transcription as an Approach to the Compositional Control of Timbre. *In Proceedings of the International Computer Music Conference (ICMC'09)*, Montreal, Canada.

GARCIA, J., BRESSON, J., SCHUMACHER, M., CARPENTIER, T. et FAVORY, X. (2015). Tools and Applications for Interactive-Algorithmic Control of Sound Spatialization in OpenMusic. *In inSonic2015, Aesthetics of Spatial Audio in Sound, Music and Sound Art*, Karlsruhe, Germany.

5

STRUCTURE TEMPORELLE DES PROCESSUS DE SYNTHÈSE

Nous nous intéressons ici à l'organisation temporelle des données et des comportements fonctionnels mis en jeu dans le contrôle de la synthèse sonore — ou en d'autres termes, à une possible approche musicale et compositionelle de la synthèse. Je présenterai successivement des travaux sur le contrôle des processus de synthèse avec la bibliothèque OMCHROMA, des outils et représentations temporelles conçus pour structurer ce contrôle, puis des projets plus récents autour du synthétiseur Chant avec l'idée d'un contrôle « continu » appliqué à ces processus. Un exemple de production (l'opéra *Re Orso* de Marco Stroppa) constituera l'illustration d'une application de la CAO articulée principalement autour de ces différents aspects.

5.1 CONTRÔLE DE LA SYNTHÈSE SONORE AVEC OMCHROMA

OMCHROMA (Agon *et al.*, 2011) est une bibliothèque de contrôle de la synthèse sonore implémentée dans OpenMusic, basée sur le système Chroma du compositeur Marco Stroppa (Stroppa, 2000).[1] Cette bibliothèque s'articule autour de structures de données abstraites représentant les paramètres d'algorithmes de synthèse sonore. Elle propose un ensemble de modules prédéfinis, présentés à l'utilisateur sous forme d'une bibliothèque de *classes* (au sens de la programmation par objets), chacune associée à un *instrument* Csound. Ces classes sont des types particuliers de matrices bi-dimensionnelles, dont l'une des deux dimensions représente les différents paramètres de l'instrument, et la deuxième représente différents « composants » de synthèse, ou instantiations virtuelles de cet instrument (Agon *et al.*, 2000). On se représentera typiquement comme paramètres les valeurs de fréquence, d'amplitude, de durée pour l'activation d'un oscillateur numérique dans un processus de synthèse additive, par exemple, et comme composants les différentes instances de cet oscillateur comme autant de partiels, ou composantes sinusoïdales élémentaires du son (voir figure 5.1-a). Cette structure est en réalité très générique et a été utilisée dans d'autres volets de nos travaux, par exemple pour contrôler la synthèse vocale (section 5.3 ci-dessous), le traitement de données gestuelles (section 4.1) ou encore les processus de spatialisation sonore (chapitre 6). Elle offre des possibilités originales pour la paramétrisation des processus, notamment via des mécanismes d'« évaluation paresseuse » (*lazy evaluation*) et l'utilisation de fonctions d'ordre supérieur comme paramètres d'initialisation, ou encore via le paramétrage croisé entre des attributs dépendants les uns des autres (cf. figure 5.1-b).

Une matrice est par ailleurs considérée comme un « évènement » dans le système de contrôle. En complément du contrôle micro-temporel opéré en interne dans cette structure

1. La bibliothèque OMCHROMA a été créée initialement par Carlos Agon, Serge Lemouton et Marco Stroppa (Stroppa *et al.*, 2002) et nous avons continué son développement jusqu'à ce jour.

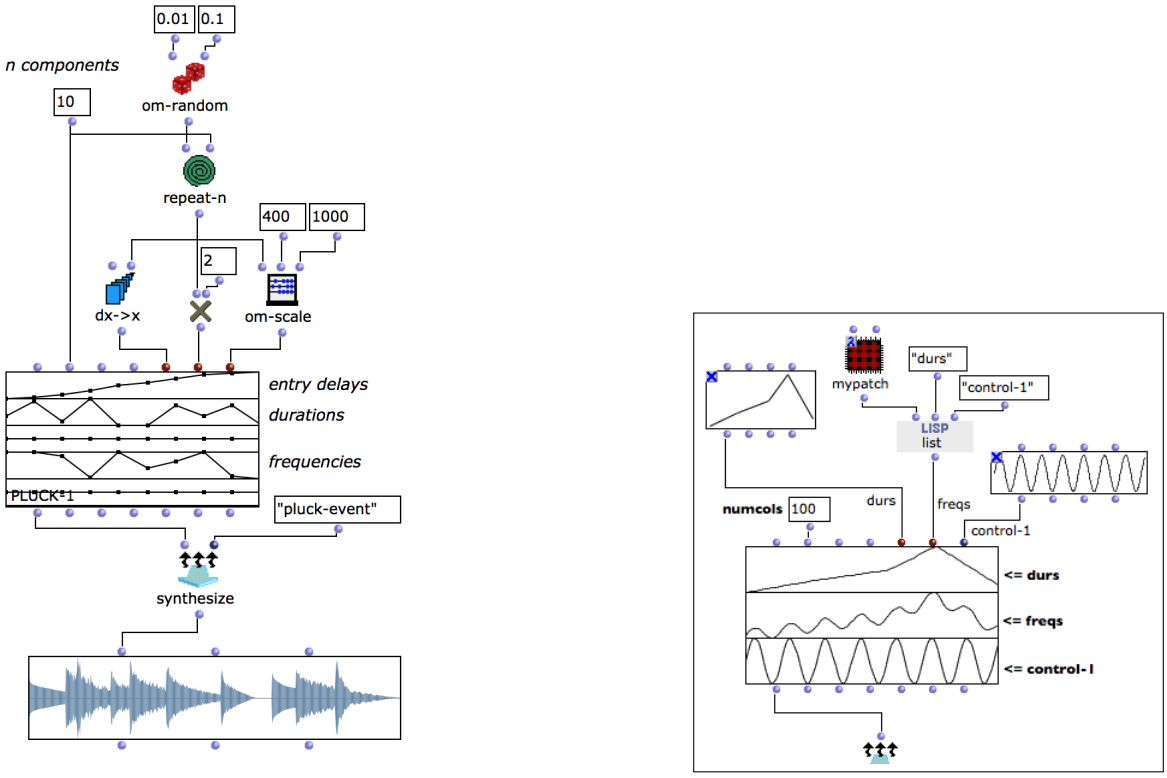
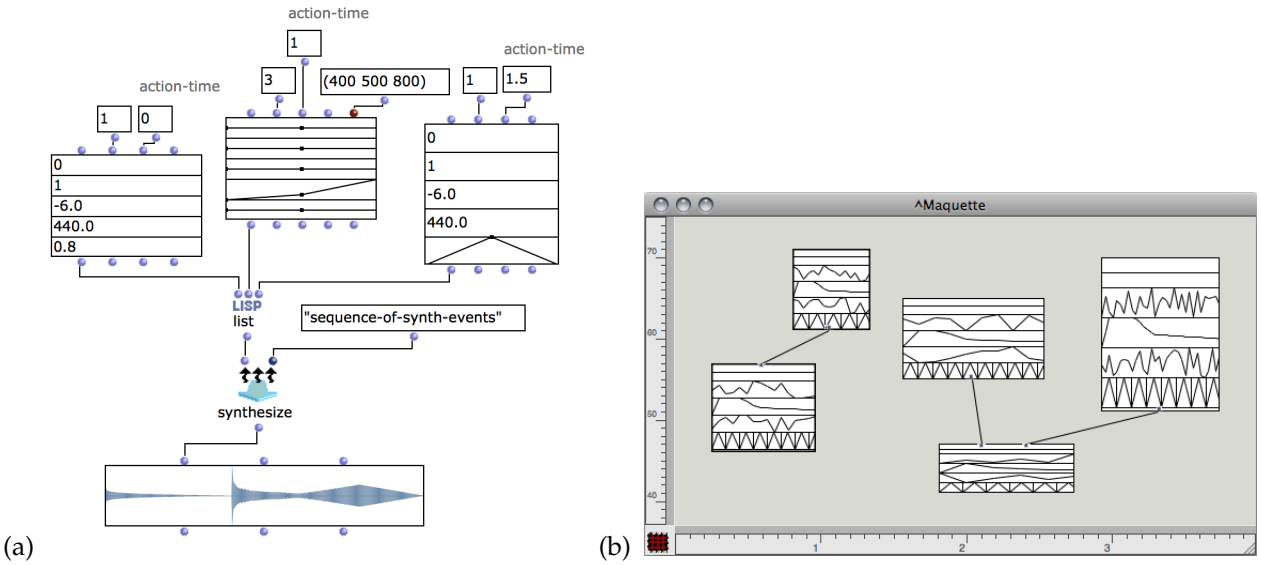


FIGURE 5.1: OMCHROMA : bibliothèque pour le contrôle de la synthèse sonore. A droite : Exemple d'initialisation et dépendances des paramètres de contrôle (détermination du paramètre *freqs* comme modulation des valeurs de *durs* par un contrôleur sinusoïdal *control-1*).



(a) (b)

FIGURE 5.2: Organisation temporelle avec des structures/« évènements » OMCHROMA (a) dans un *patch* OpenMusic et (b) dans une *maquette*.

(paramètres de *durées* ou d'*offsets* attribués aux différents composants), elle est elle-même assortie d'un *onset*, ou date absolue dans un référentiel temporel donné, qui permet donc d'organiser les structures de contrôle relativement les unes aux autres dans une forme : une vision multi-échelle, s'étendant du « micro » jusqu'au « macro-sonore » (voir figure 5.2).

Détachée autant que possible de biais esthétiques (et ce malgré l'implication fondamentale d'un compositeur dans sa conception même), OMCHROMA compte aujourd'hui un nombre conséquent d'utilisateurs. Elle est l'objet d'un groupe dédié sur le site IRCAM *ForumNet* (ainsi que sur *Facebook* !). Les chapitres de Fabio De Sanctis De Benedictis [🕮 17], de Alireza Farhang [🕮 18], ou encore de Alessandro Ratoci [🕮 20] rendent compte de son utilisation.

5.2 REPRÉSENTATION DES STRUCTURES TEMPORELLES

MODÈLES SONORES. Les *modèles sonores* sont des représentations spécifiquement orientées vers la synthèse sonore et le déploiement de ses structures de contrôle dans le temps, développées dans OMCHROMA. Je me suis intéressé à cette notion de « modèle » telle que formulée par Marco Stroppa : « *[...] une structure de données extraites d'analyses [...] destinée à être utilisée en tant que réservoir de formes et de paramètres applicables à un synthétiseur sonore* » (Stroppa *et al.*, 2002), que j'ai implémentée sous la forme relativement générique d'un conteneur de hauteurs organisées suivant une segmentation temporelle — ces deux composantes pouvant être issues de données d'analyse ou programmées par le compositeur. Ce conteneur est par ailleurs doté d'un éditeur permettant de visualiser la structure du modèle (voir figure 5.3). Une fonction de *mapping* (également programmable visuellement) permet ensuite de produire des données de contrôle pour la synthèse à partir de la structure du modèle (Bresson *et al.*, 2007) 🖉. Nous en verrons des utilisations dans la suite de ce chapitre (section 5.5).

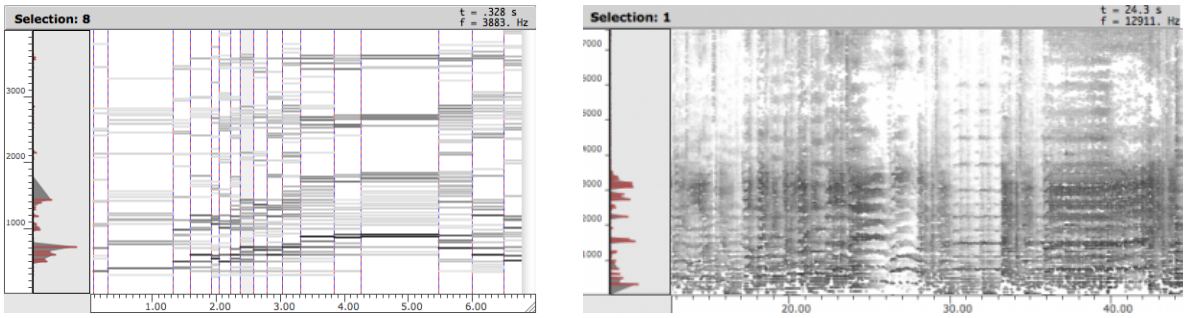


FIGURE 5.3: Descriptions sonores pour le contrôle de la synthèse : l'éditeur de CR-MODEL.

MAQUETTES. L'utilisation de la *maquette* (voir section 3.1) est une autre direction proposée dans mes précédents travaux pour structurer les processus de synthèse sonore dans le temps (Bresson et Agon, 2006) 🖉. Les processus développés dans une maquette comprennent et distinguent plusieurs niveaux. Les composants *discrets* de la structure temporelle, que l'on peut comparer à des « évènements », sont représentés par des « boîtes » et organisés dans cette structure. Chacun est un processus partiel générant des données. Il est possible alors de définir une sémantique pour l'intégration et l'interprétation de ces composants dans ce que nous appelons la « valeur » d'une maquette, et ainsi déployer ces données en un résultat sonore transmis à un niveau d'abstraction supérieur. [2] La maquette peut donc représenter un

2. On retrouvera également cette idée de structuration du sonore par des éléments discrets chez certains auteurs : "*From a constitutional point of view, a sound object, will in general be a complex object, decomposable into simpler*

arrangement temporel de données plus ou moins abstraites destinées à être intégrées dans la production d'une structure globale — par exemple, des paramètres de synthèse intégrés *a posteriori* pour produire un son (voir l'exemple de la figure 5.2 avec les matrices OMCHROMA).

Ici encore, cette vision des processus temporels permet de concevoir des formes structurées non seulement au niveau « macro » des processus compositionnels, mais également au niveau « micro » de la synthèse sonore. Un exemple intéressant est décrit dans *The OM Composer's Book 2* par Tolga Tüzün au sujet de la pièce *Metathesis* (2006) (Tüzün, 2008).

5.3 OM-CHANT : VERS UN PARADIGME DE CONTRÔLE CONTINU

CHANT. Le synthétiseur Chant (Rodet *et al.*, 1984) a été développé à l'IRCAM dans les années 80. Il permet de simuler un modèle de voix chantée à partir de la technique de synthèse par Fonctions d'Ondes Formantiques (FOF) (Rodet, 1984). Un générateur de FOF produit un train périodique de sinusoïdes amorties, ayant pour effet l'équivalent d'un « formant » dans le spectre sonore, c'est à dire un maximum d'énergie déterminé essentiellement par une amplitude, une fréquence centrale et une largeur de bande, caractéristique de l'observation des sons voisés. Le synthétiseur est composé d'un banc de générateurs de FOFs connectés en parallèle,[3] chacun produisant donc la contribution d'un formant, fonctionnant à une même période et de manière synchrone. Cette période commune détermine la fréquence fondamentale (hauteur « perçue ») du son. Le contrôle de la synthèse est opéré par des processus indépendants actualisant périodiquement les paramètres des différents modules et générateurs. La figure 5.4 montre un sonagramme de son synthétisé par Chant, sur lequel apparaissent clairement les évolutions indépendantes de la fréquence fondamentale et de la structure formantique du son.

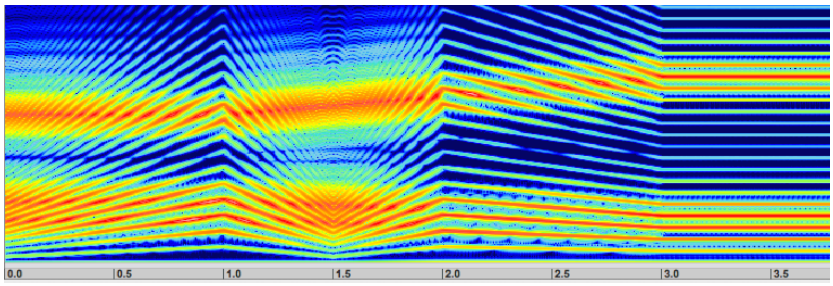


FIGURE 5.4: Sonagramme (représentation temps/fréquence, amplitude en niveau de couleur) d'un son produit par le synthétiseur Chant. La fréquence fondamentale suit une courbe brisée, dont les multiples forment l'ensemble des partiels (composantes harmoniques). Deux *formants* produisent une modulation de l'amplitude du spectre, évoluant sur la durée du son.

Les paramètres contrôlant les modules de Chant peuvent être considérés comme des signaux définis par des fonctions $f : T \to \mathbb{R}$ du temps vers des valeurs scalaires (où T est un intervalle de $\mathbb{R}^+$ définissant la durée totale du processus de synthèse — en réalité cet intervalle est discrétisé avec une période d'échantillonnage donnée). L'état du synthétiseur (soit un ensemble des valeurs $f_p(t)$, où f_p est la fonction associée au paramètre p) n'est pas nécessairement spécifié à chaque instant : Chant peut réaliser un rendu *offline*, prenant

elements, which we shall call components, organized under a logical structure. Thus, if undoubtedly the composition of sound is related to the signal processing algorithms, it is just as much, if not more, the description of the structure of relationships and behaviours affecting the different components which constitute it." (Eckel et Gonzalez-Arroyo, 1994).

3. Ces générateurs peuvent être associés à d'autres générateurs de signaux ou à des bancs de filtres résonnants.

en compte les valeurs données passées et futures de chaque paramètre pour interpoler systématiquement un état et calculer $f_p(t)$ pour chaque paramètre p et pour tout $t \in T$. Cette approche du contrôle propose ainsi une vision « continue » plutôt que basée sur des « évènements » discrets. Elle contraint le processus de synthèse à un mode monophonique (un seul état par instant), mais offre une certaine expressivité et une prise subtile sur des effets de phrasé (nous y reviendrons dans les pages qui suivent). [4]

Un système de règles permettait dans les versions originales du synthétiseur de lier les valeurs de certains paramètres à des modulations telles que le *vibrato* (modulation sinusoïdale) ou le *jitter* (variations pseudo-aléatoires autour d'une valeur donnée), ou encore à l'évolution d'autres paramètres (par exemple, faire évoluer les fréquences des formants en fonction de la fréquence fondamentale, etc.). De tels procédés furent mis en œuvre dans des environnements de contrôle comme Formes (Rodet et Cointe, 1984) (voir chapitre 2), qui permettait de définir des processus temporels intégrant ces règles de contrôle. Le logiciel Diphone (Rodet et Lefevre, 1997), quelques années plus tard, permettait de combiner graphiquement des unités statiques (segments de signaux de contrôle) pour générer des phrases continues avec Chant via une interpolation généralisée des paramètres — *Generalised Diphone control* (Depalle *et al.*, 1993), mais abandonnait le contrôle procédural par règles.

OM-CHANT. La bibliothèque OM-CHANT de OpenMusic permet de générer des données de contrôle et de réaliser des appels externes au synthétiseur Chant depuis notre environnement de CAO (Bresson et Michon, 2011; Bresson et Stroppa, 2011) 🖉. [5] Elle contient par ailleurs un ensemble d'utilitaires pour la spécification des paramètres du synthétiseur : bases de données de formants, règles de contrôle, application de modulations sur les signaux de contrôle, etc. [6]

Bien qu'héritant de l'approche « continue » de Chant, OM-CHANT se structure autour d'objets, appelés « évènements » de contrôle, permettant de spécifier les valeurs et l'évolution des différents modules du synthétiseur (générateurs de FOFs, filtres etc.) à une date donnée et sur une certaine durée. Ces objets sont en réalité des matrices, similaires à celles utilisées dans la bibliothèque OMCHROMA vue précédemment (voir figure 5.5). Une matrice de contrôle de FOFs, par exemple, spécifiera les différents paramètres (fréquence, amplitude, largeur de bande, etc.) des différents générateurs de FOF en activité sur un intervalle de temps donné. La fréquence fondamentale est contrôlée par un signal de valeur scalaire, également représentée par des évènements (et donc également articulés dans le temps et la durée). Sur le modèle des évènements de contrôle dans OMCHROMA, les évènements de contrôle OM-CHANT peuvent ainsi être localisés dans le temps et combinés dans des structures temporelles à plus grande échelle — par exemple dans une *maquette* (voir figure 5.6). [7]

4. Bien que tombé en relative désuétude aujourd'hui, Chant a été utilisé pour la création de plusieurs œuvres importantes des années 80 telles que *Mortuos Plango, Vivos Voco* de Jonathan Harvey (1980), *Chréode* de Jean-Baptiste Barrière (1983), ou encore *Les chants de l'amour* de Gérard Grisey (1985).

5. L'article (Bresson et Michon, 2011) 🖉 met en regard OM-CHANT avec la bibliothèque CHANT-LIB réalisée par Romain Michon dans le cadre de son M1 à l'Université de Saint-Etienne (2010), qui implémente des fonctionnalités similaires en utilisant OpenMusic et le synthétiseur Csound.

6. Une grande partie de ces outils a été implémentée à partir des sources originales des modules de contrôle, retrouvées à l'IRCAM et interprétées avec l'aide précieuse de Xavier Rodet. Certains étaient également disponibles dans la bibliothèque PW-CHANT de l'environnement Patchwork, créée par Laurent Pottier.

7. Une documentation complète de la bibliothèque OM-CHANT est disponible à l'adresse `http://support.ircam.fr/docs/om-libraries/om-chant/`. La bibliothèque est également distribuée avec une quarantaine de *patch*-tutoriels.

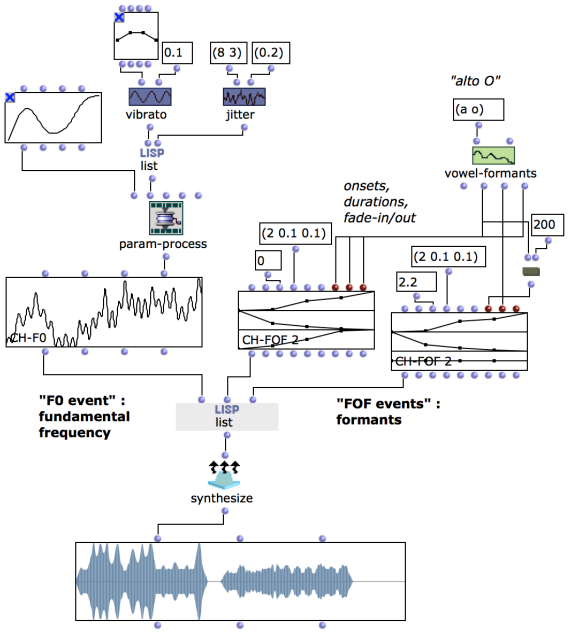


FIGURE 5.5: Évènements de contrôle dans OM-CHANT : CH-F0 (fréquence fondamentale) et CH-FOF (paramètres des FOFs).

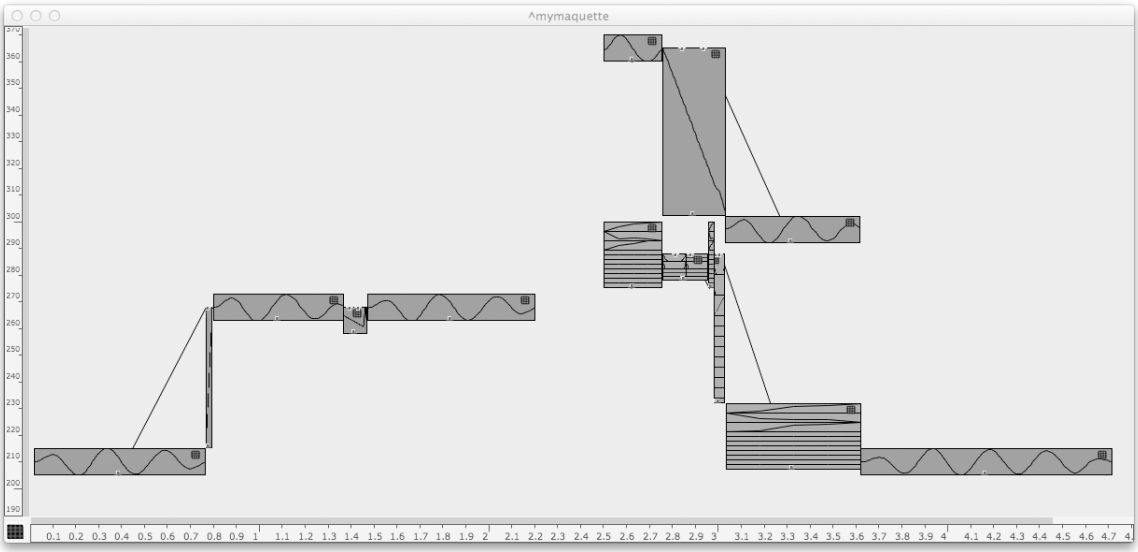


FIGURE 5.6: Evènements de contrôle dans OM-CHANT : structuration temporelle discrète des processus dans une *maquette*.

Pour produire un son, la fonction *synthesize* de OM-CHANT rassemble un ensemble d'évènements et fusionne leur contenu pour produire une séquence de trames de contrôle décrivant les différentes fonctions f_p par une séquence ordonnée de valeurs. Le synthétiseur, appelé en ligne de commande, interprète ces trames pour calculer $f_p(t)$ par interpolation pour tout $t \in T$ (produisant ainsi des transitions continues entre les valeurs de contrôle).

Cette considération pour la notion d'évènement (entités symboliques formant une structure composée) en parallèle de l'approche « continue » proposée par le synthétiseur, constitue un système original combinant la construction musicale et les spécifications de phrasé dans une expression globale de la forme temporelle des sons (Bresson et Stroppa, 2011) 🖉. Dans ce système les aspects continus peuvent-être appréhendés à différents niveaux :

— *À l'intérieur des évènements* : des évolutions et modulations de complexité arbitraire, programmées dans l'environnement de CAO ou issues d'analyses, peuvent être spécifiées pour tout ou partie des paramètres de contrôle (voir par exemple l'évènement « F0 » contrôlant l'évolution de la fréquence fondamentale sur la figure 5.5).

— *Entre les évènements* : dans la gestion des intervalles inter-évènements ou des superpositions temporelles des évènements.

5.4 PHRASÉ ET TRANSITIONS

Le processus de conversion d'évènements discrets composés en structures temporelles vers un flux de contrôle continu est l'un des enjeux principaux de l'implémentation du système OM-CHANT. Comme on peut le voir sur la figure 5.7, les zones temporelles « sous-spécifiées » de $f_p(t)$ (à l'intérieur ou entre les évènements) peuvent être gérées par interpolation et ré-échantillonnage des données de contrôle (par le synthétiseur). Elles peuvent aussi faire l'objet d'un traitement explicite afin de contrôler les transitions entre évènements (par exemple, générer un *fade-in/fade-out* d'amplitude, soit un passage par zero de la fonction de paramètre correspondante, pour un effet de *staccato*). Les zones « sur-spécifiées » (par exemple résultant de la superposition de plusieurs évènements — voir figure 5.7-b) représentent par contre des inconsistances ou des ambigüités dans la spécification de ce signal, et doivent subir un traitement spécifique afin d'assurer la cohérence du signal de contrôle généré. Ce traitement sera également l'opportunité de nouvelles possibilités compositionnelles.

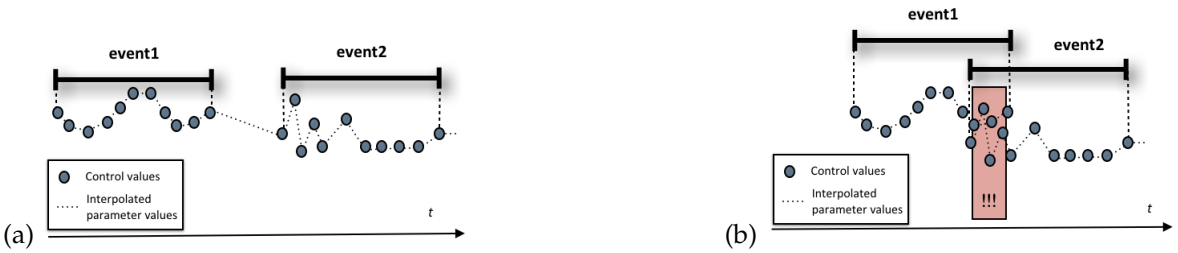
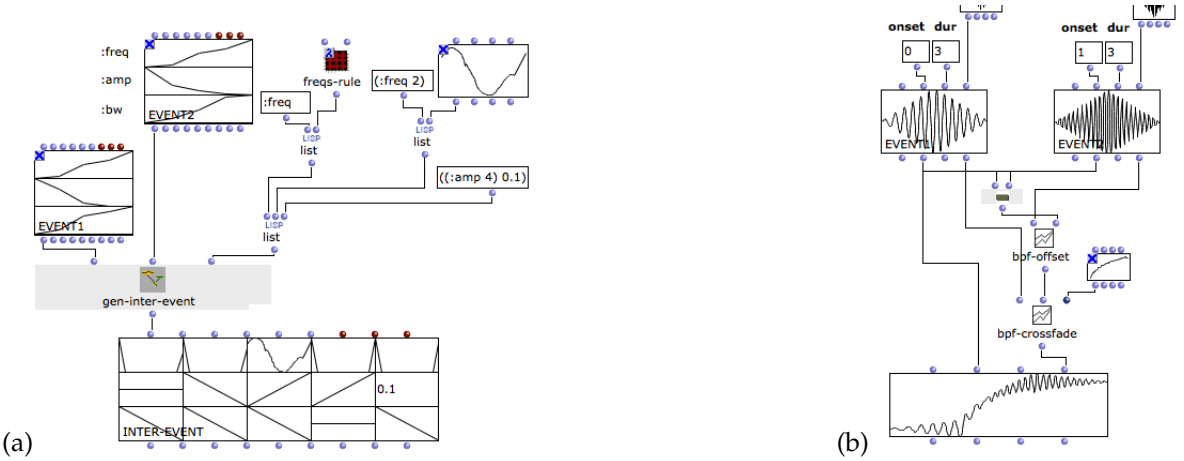


FIGURE 5.7: Représentation schématique de deux évènements OM-CHANT :valeurs de contrôle et fonctions $f_p(t)$ ré-échantillonnées (a) sans superposition et (b) avec superposition.

L'idée de traiter les transitions et les états intermédiaires comme des éléments spécifiques du système de contrôle fut l'objet du stage de M2 de **Raphaël Foulon** (2012) [★], dont des résultats sont rapportés dans (Foulon et Bresson, 2013) 🖉. Lors de ce stage Raphaël Foulon a dans un premier temps étudié les phénomènes de production vocale à partir d'analyses temps-fréquence d'extraits audio (bases de données de phonèmes, enregistrements de chant lyrique, etc.), en discriminant les états stables des états transitoires selon le comportement

observé des formants. Alors que les états stables peuvent être représentés par des valeurs contrôlées des paramètres de FOF, l'objet principal de ce travail était la modélisation des parties transitoires en tant que *profils formantiques*, définissant des morphologies générales de passages des valeurs d'un état stable au suivant. Un certain nombre de tels profils ont été élaborés empiriquement à partir d'observations des analyses d'extraits sonores. L'article (Foulon et Bresson, 2013) décrit la modélisation de profils formantiques simulant des consonnes comme le « m » ou le « b », ainsi que des phénomènes, tels que le *r* « roulé », ou encore le *morphing* contrôlé entre plusieurs états « stables » de la synthèse par FOF. Un ensemble d'outils a été réalisé, permettant la génération de données de contrôle liant les évènements d'un processus de synthèse. Cette génération peut donc faire appel à des modèles de transition prédéfinis, mais est également librement programmable (voir figure 5.8). [8]



(a) (b)

FIGURE 5.8: Exemples de calcul de transitions entre deux évènements OM-CHANT. (a) La fonction *gen-inter-event* : $Event \times Event \times (\mathbb{R} \times \mathbb{R} \rightarrow Time \rightarrow \mathbb{R})^* \rightarrow Event$. Trois règles sont appliquées (de gauche à droite) : l'amplitude du 5^e formant (*:amp 4*) durant la transition est fixée à 0.1 ; la transition de la fréquence fréquence du 3^e formant (*:freq 2*) suit une courbe dessinée à la main ; et la transition des autres fréquences (*:freq*) est définie dans le programme *freqs-rule*. (b) Utilisation du *crossfade* : fusion des deux évènements.

Dans l'article (Bresson *et al.*, 2013), nous proposons différentes stratégies pour gérer les transitions, et généralisons leur application sur de longues séquences d'évènements à l'aide d'un mécanisme inspiré de *reduce*, un outil classique de la programmation fonctionnelle. [9] Nous appelons « fonction de transition » une fonction de deux évènements successifs, considérée comme un combinateur temporel définissant une stratégie pour la gestion des transitions, et produisant un ou plusieurs évènements en sortie. Cette fonction peut être définie graphiquement par l'utilisateur sous forme de *patch* ; une fonction d'ordre supérieur (appelée *ch-transitions* : $Event^* \times (Event \times Event \rightarrow Event^*) \rightarrow Event^*$) opère sur la liste initiale d'évènements par application récursive de cette fonction, produisant une nouvelle séquence d'évènements combinés.

8. En règle générale, les fonctions de génération proposées sont de type $Event \times Event \rightarrow Event^*$. Elle produisent, à partir de deux évènements successifs ou partiellement superposés, soit un évènement transitoire, soit un nouvel ensemble d'évènements (par exemple si les évènements originaux sont modifiés), soit encore un unique évènement réalisant la fusion des évènements de départ.

9. La fonction *reduce* (également appelée *fold*) opère sur une structure de données récursive (par exemple une liste ou un arbre), et construit une valeur par combinaison des résultats de l'application récursive d'une fonction sur les éléments de cette structure (Hutton, 1999). Dans notre cas, la combinaison se fait par concaténation.

5.5 UNE APPLICATION EN PRODUCTION : *re orso*

Nous nous sommes attardés dans les chapitres et sections précédents sur la question du contrôle de la synthèse sonore dans OpenMusic, en essayant de mettre en avant la formalisation des processus temporels et les articulations possibles entre temps discret et temps continu. Cette formalisation souligne également la spécificité d'un contrôle en « temps-différé » des processus de synthèse.

Une partie importante de ce travail a été effectuée en collaboration avec le compositeur Marco Stroppa : l'exemple de production et de recherche musicale développé dans cette section avec son opéra *Re Orso* (2012) est une réalisation majeure qui nous a permis d'appliquer et d'étendre cette réflexion (Stroppa et Bresson, 2016) [19]. Ce détour au cœur d'un projet compositionnel nous montrera, avec un exemple concret, différentes implications possibles de la CAO dans la création de structures musicales et de sons de synthèse.

FIGURE 5.9: *Re Orso* (2012). Photo © E. Carecchio.

Re Orso (*Légende pour 4 chanteurs, 4 acteurs, ensemble, électronique, spatialisation et totem acoustique*) est un opéra composé par Marco Stroppa en 2012. [10] Voix et électronique sont deux éléments essentiels de ce projet, écrites et organisées à l'aide d'outils de composition assistée par ordinateur — notamment ceux que nous avons présentés dans ce chapitre. Le contrôle de la synthèse dans OpenMusic a permis d'établir une complémentarité entre la structure formelle et les considérations de phrasé ou d'évolution du rendu sonore, déterminants dans la qualité, l'expressivité et l'unité esthétique des sons produits par la synthèse.

MODÈLES SONORES. La notion de *modèle sonore* présentée en section 5.2 est essentielle dans *Re Orso* (comme dans la plupart des œuvres de Stroppa). Chaque instant musical de cet opéra est inspiré des prescriptions du libretto et de la structure de la dramaturgie : un objectif important, du point de vue compositionnel, était d'attribuer un rôle précis (au sens dramaturgique) au matériau électronique, de sorte qu'il soit perçu comme un personnage en soi (ou parfois, comme plusieurs personnages) dont la personnalité apparaît et se développe au cours de l'action. Les modèles permettent ainsi au compositeur d'instancier des *Leitgestalten*, c'est-à-dire des groupes de formes plus ou moins abstraites possédant des propriétés communes, ou pouvant être reconnues comme dérivés d'une même identité. L'un des principaux modèles (ou *Leitgestalten*/idées récurrentes de l'opéra) est celui d'une cloche

10. Commande de Françoise et Jean-Philippe Billarant pour l'IRCAM, l'Ensemble Intercontemporain, l'Opéra Comique, La Monnaie de Bruxelles, et de l'État Français. Électronique réalisée à l'IRCAM avec l'assistance de Carlo Laurenzi.

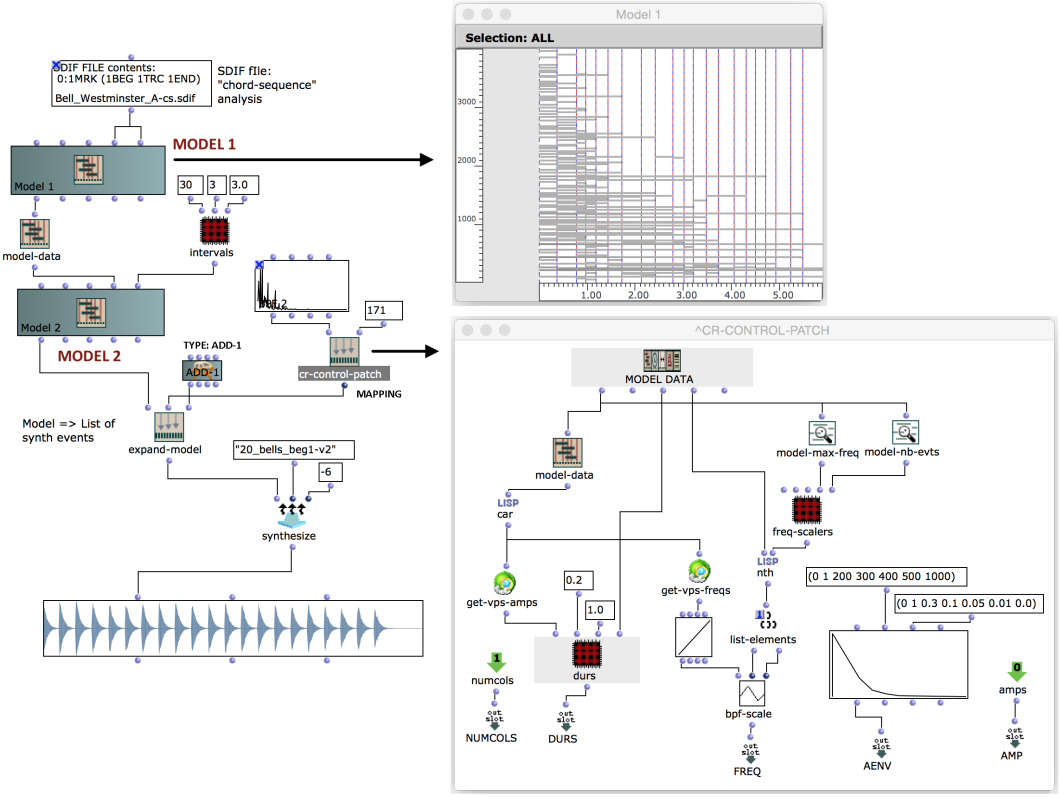


FIGURE 5.10: *Patch* OpenMusic synthétisant les sons de cloches dans *Re Orso*. En haut, les données d'analyses utilisées pour instancier le modèle sont importées à partir d'un fichier SDIF. La fonction *expand-model* convertit les données de ce modèle en structures de contrôle pour la synthèse sonore à partir des spécifications (*mapping* des paramètres) exprimées dans le sous-*patch* : *cr-control-patch*.

« fantôme ». Il est utilisé notamment pour synthétiser une série sons que l'on entend vers le milieu de l'opéra lors de son *intermezzo storico*. L'analyse d'un son de cloche réelle a servi à construire ce modèle, qui est ensuite décliné dans ses différentes instanciations (dans les différents coups de cloche virtuelle synthétisés), pour rendre le son de plus en plus « sombre ». Le *patch* OpenMusic de la figure 5.10 montre l'implémentation de ce processus. On peut y voir le matériau initial : le résultat de l'analyse spectrale du son de cloche (sous forme de CR-MODEL), contenant des informations fréquentielles segmentées temporellement, ainsi que les procédés mis en œuvre pour la transformation de ce matériau et la production d'une série de sons de synthèse. Le modèle reflète la structure et l'évolution interne du son. Ses différents « segments » sont traités successivement par une fonction de *mapping* produisant des instances de matrices d'une classe OMCHROMA associée à un module de synthèse additive.

La même notion de modèle est utilisée pour la composition de parties instrumentales dans l'opéra. Les parties interprétées par un Disklavier (piano mécanique contrôlé via MIDI) sont notamment réalisées à partir de données issues de modèles sonores, et de processus similaires de filtrage et de *mapping* vers les paramètres d'une partition MIDI.

SYNTHÈSE DE VOIX. Les voix chantées générées par ordinateur occupent une place importante dans le matériel dramaturgique de *Re Orso*, et constituèrent l'un des principaux défis dans la création des parties électroniques de l'opéra. Plusieurs familles de sons ont été créées, parmi lesquelles des phonèmes produits par une succession de voyelle-consonne-voyelle, des *messa di voce* (*crescendo* suivi par un *diminuendo*), des voix « éthérées » dont les paramètres formantiques évoluent librement dans le spectre, et des voix humoristiques dans le registre *coloratura*. Ces sons générés à l'aide de la bibliothèque OM-CHANT apparaissent dans différents passages de l'opéra. La figure 5.11 montre le programme OpenMusic produisant ceux diffusés lorsque le Roi entend pour la première fois la voix intérieure du Vers (un personnage imaginaire reflétant sa propre conscience), dans la deuxième scène. On y observe deux objets principaux (CH-FOF et CH-F0) utilisés respectivement pour spécifier l'évolution des paramètres des générateurs de FOFs et de la fréquence fondamentale. Sur la gauche, les paramètres des FOFs (fréquences, amplitudes, largeurs de bande, etc.) sont déterminés par la fonction *vowel-formants* à partir d'une voyelle donnée (soprano « o », alto « e », etc.). Un ensemble de règles leurs sont appliquées, telles que *autobend* (qui translate les fréquences centrales des premiers formants en fonction de la fréquence fondamentale), *comp-formants* (qui ajoute une résonnance dans la région basse du spectre), ou *autoamp* (qui ajuste automatiquement les amplitudes relatives des formants). D'autres paramètres de synthèse sont issus des entrées du programme, représentées par des flèches numérotées de 0 à 11, et contrôlent par exemple la forme de l'enveloppe des sinusoïdes amorties (*win*, *wdur*, *wout*) ou des formants (*bw-scale*). Sur la droite de la figure, une courbe de fréquence fondamentale est produite par modulation d'une hauteur de base à l'aide d'un vibrato (contrôlé par des enveloppes de fréquence et d'amplitude) et d'un *jitter* (également contrôlé par des valeurs de fréquence et d'amplitude). La fonction *synthesize* en bas collecte les deux évènements, en traite les différents paramètres, et réalise un appel externe au synthétiseur pour produire un fichier audio.

Ce programme (ici appelé *one-voice*) est relativement générique et peut être contrôlé librement par de nombreux paramètres (les entrées 0-11 sur la figure 5.11). Sur la figure 5.12, il est utilisé itérativement dans un autre programme pour réaliser une série de neuf sons avec différents jeux de paramètres. Cette série de sons réalise une progression partant de voix réalistes (paramètres de synthèse dans une plage de valeurs proches de celle des voix humaines, courbes de vibrato naturelles, etc.) vers des voix imaginaires aux caractéristiques improbables. Les derniers sons et jeux de paramètres de cet exemple sont particulièrement intéressants : une fréquence fondamentale très basse, de l'ordre de quelques Hertz, est en effet perçue, non plus comme une hauteur, mais comme un train d'impulsions. Par ailleurs, des attaques brèves et des temps de résonance croissants attribués aux générateurs de FOFs produisent l'effet de percussions, perceptivement proches de sons de cloches. [11]

La possibilité de contrôle du phrasé et des évolutions continues est également mise à contribution, par exemple lors de la production de sons transformant progressivement une sonnerie de téléphone en une mélodie de voix lyrique chantée, qui sont entendus au tout début de l'opéra. Les processus de synthèse réalisant ces sons produisent également, de manière itérative, des évènements de type CH-FOF et CH-F0, cette fois combinés en une unique séquence

11. Le contrôle des largeurs de bandes (et de la forme des formants en général) est un aspect important dans la synthèse par FOF : dans un cas limite, une largeur de bande infiniment faible (0Hz) correspond à une distribution de Dirac dans le spectre, et donc à un simple partiel (signal sinusoïdal élémentaire) dans le domaine temporel. La synthèse par FOF peut alors produire des sons similaires à ceux produits par synthèse additive.

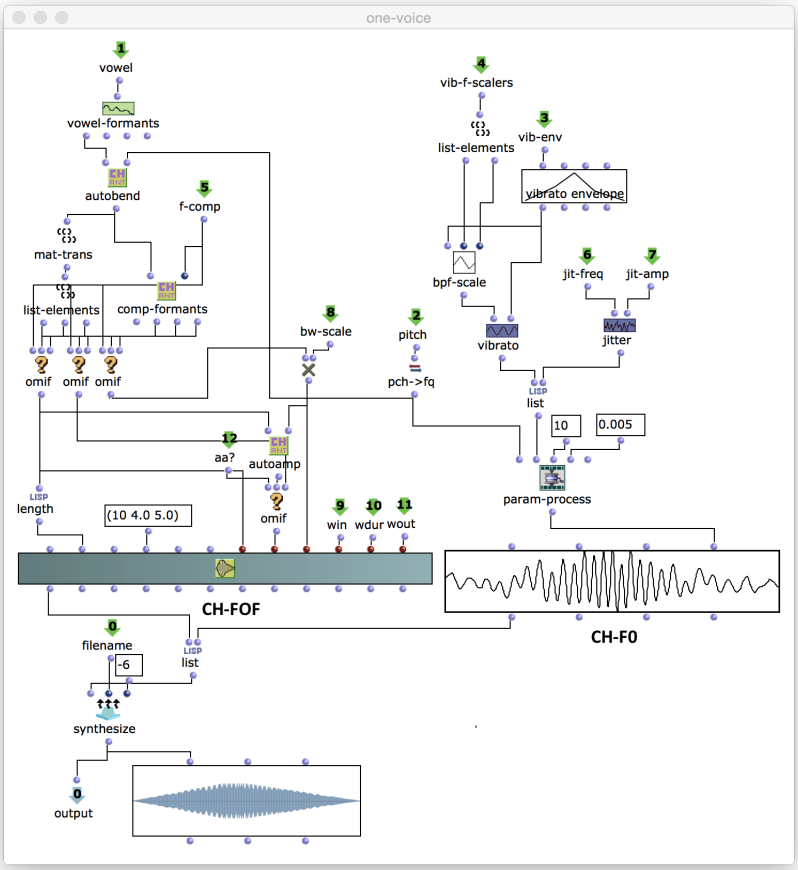


FIGURE 5.11: Synthèse de *messa di voce* avec OM-CHANT.

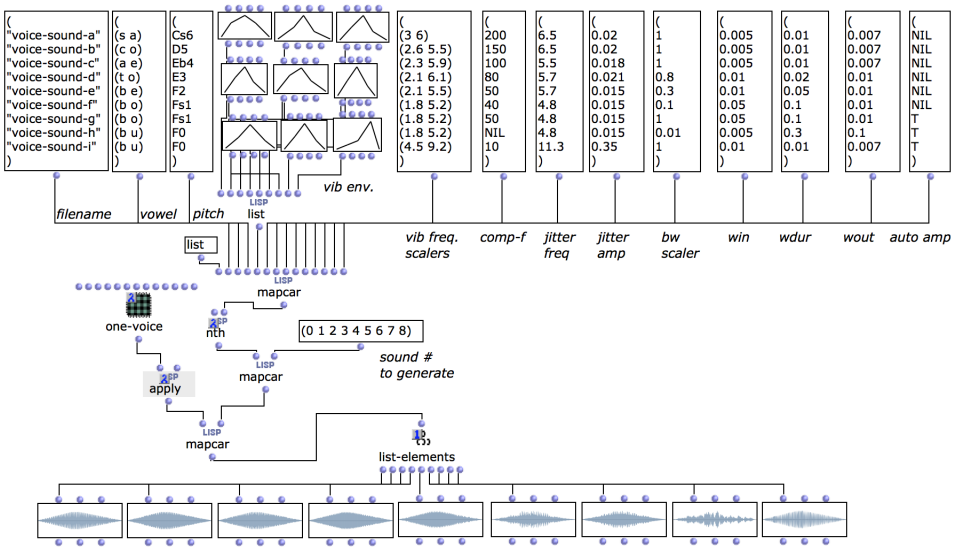


FIGURE 5.12: Série d'appels au programme de la figure 5.11 avec différents jeux de paramètres. Voix imaginaires dans *Re Orso* (Partie I, Scène 2).

traitée et synthétisée par OM-CHANT. Les paramètres sont donc interpolés progressivement entre des valeurs initiales correspondant au timbre synthétique des notes courtes émises par une sonnerie de téléphone, vers des valeurs reproduisant des voyelles chantées (voir le résultat sur la figure 5.13). Une interpolation se produit également sur les articulations, en passant progressivement d'un *staccato* pur à un *legato*, sur la durée des transitions, de plus en plus longues, et sur le vibrato, inexistant au début et de plus en plus marqué vers la fin.

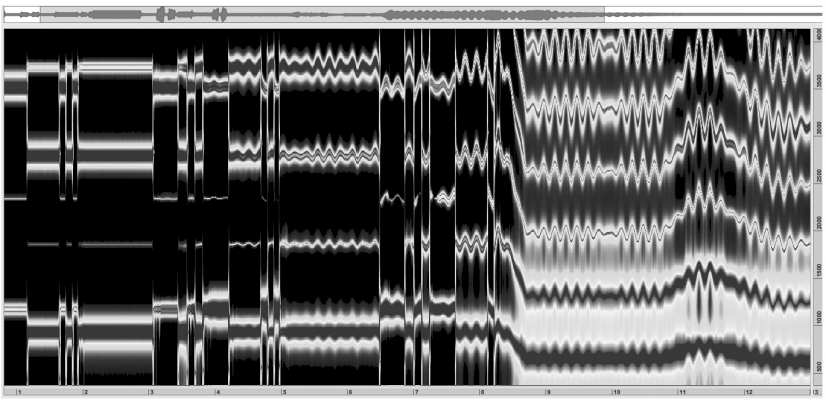


FIGURE 5.13: Sonagramme d'une « phrase » synthétisée par OM-CHANT à partir d'une séquence d'évènements : transitions continues d'une sonnerie de téléphone portable vers une mélodie chantée.

MORT DU ROI. Re Orso meurt à la fin, ainsi que la propre voix du personnage. Le passage de la mort du Roi a été créé à l'aide d'un processus de synthèse par FOFs (voir figure 5.14), d'une durée d'environ 90 secondes. Le sonagramme du son résultant de ce programme (visible sur la figure 5.14) en révèle un certain nombre de détails intéressants. La fréquence fondamentale des générateurs de FOF part d'une hauteur relativement élevée (Ré4, 585 Hz) et plonge jusqu'à des valeurs de moins de 0.3 Hz, perçues comme des impulsions. La descente est divisée en octaves (Ré5-Ré4, Ré4-Ré3, etc.) : dans certaines de ces octaves, un vibrato est appliqué et vient moduler le spectre, conférant au son une connotation vocale plus forte. Parallèlement, les cinq formants vocaux, identifiables dans la première moitié de la séquence, semblent s'effondrer pour se transformer en séquences d'impulsions. Les largeurs de bandes des formants diminuent progressivement, et les impulsions deviennent de plus en plus longues, laissant entendre les résonances produites par les modifications du spectre. Vers la fin de la séquence, le spectre est similaire à celui d'un son issu d'une synthèse additive (accumulation de sinusoïdes), dont les caractéristiques sont dérivées du modèle de son de cloche décrit précédemment.

Le son de cloche final est formé de 25 partiels qui apparaissent vers le milieu de la séquence, dérivés des 5 formants initiaux, donc organisés en 5 groupes de formants qui s'écartent progressivement de leur fréquence centrale à mesure que les largeurs de bande diminuent. Cette structure est concrétisée sous forme d'un autre *modèle*, visible sur la figure 5.15-b et produit par le programme *gen-fql*, détaillé sur la figure 5.15-a. La séquence commence avec des valeurs de formants extraites d'une base de données de voyelles (Phase 1, partie gauche de la figure 5.15-a). Durant cette phase, le synthétiseur produira des transitions continues entre les valeurs d'un « u » tel que chanté par une voix de soprano (*su*), d'alto (*au*), de ténor (*tu*), et de basse (*bu*). La fin de la séquence (Phase 3) correspond aux valeurs de partiels du son de cloche, déterminées par une analyse spectrale sur le modèle du procédé vu

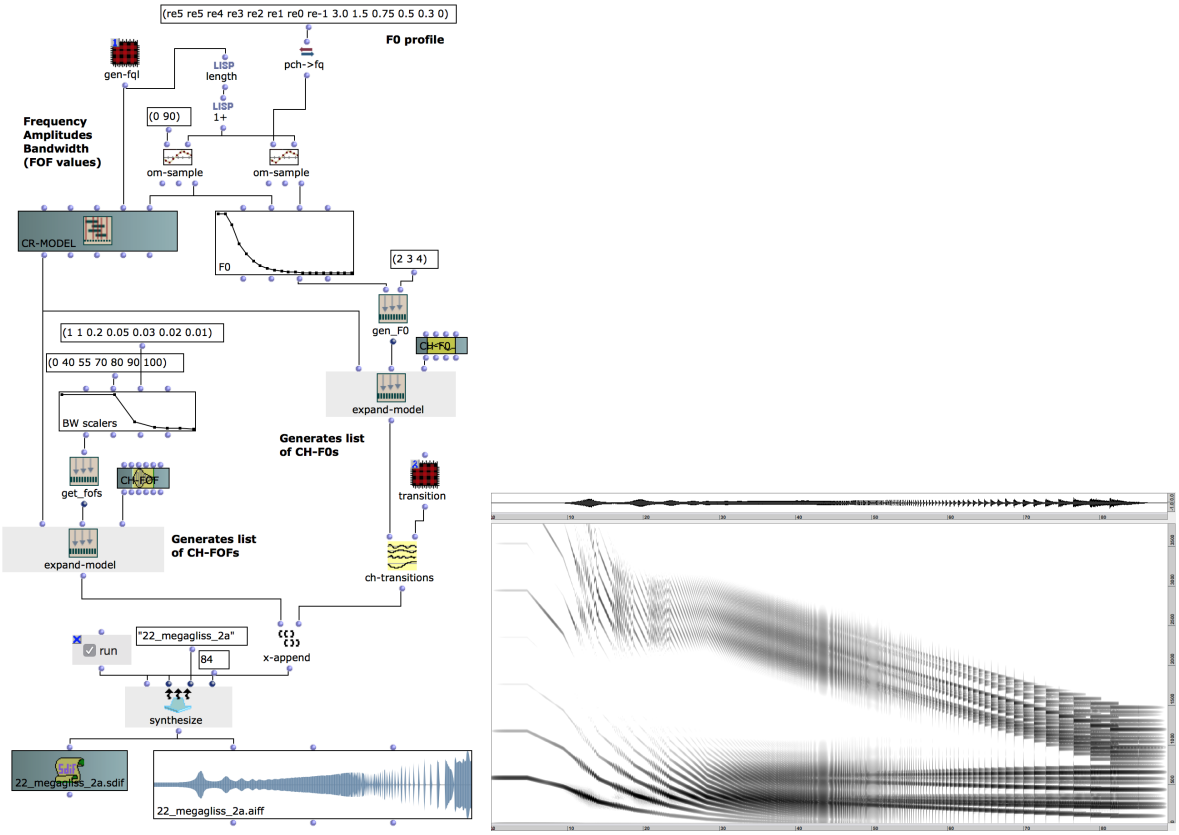


FIGURE 5.14: *Patch* OpenMusic (version simplifiée) réalisant la synthèse de la mort de la voix du Roi. À droite : analyse spectrale du son synthétisé

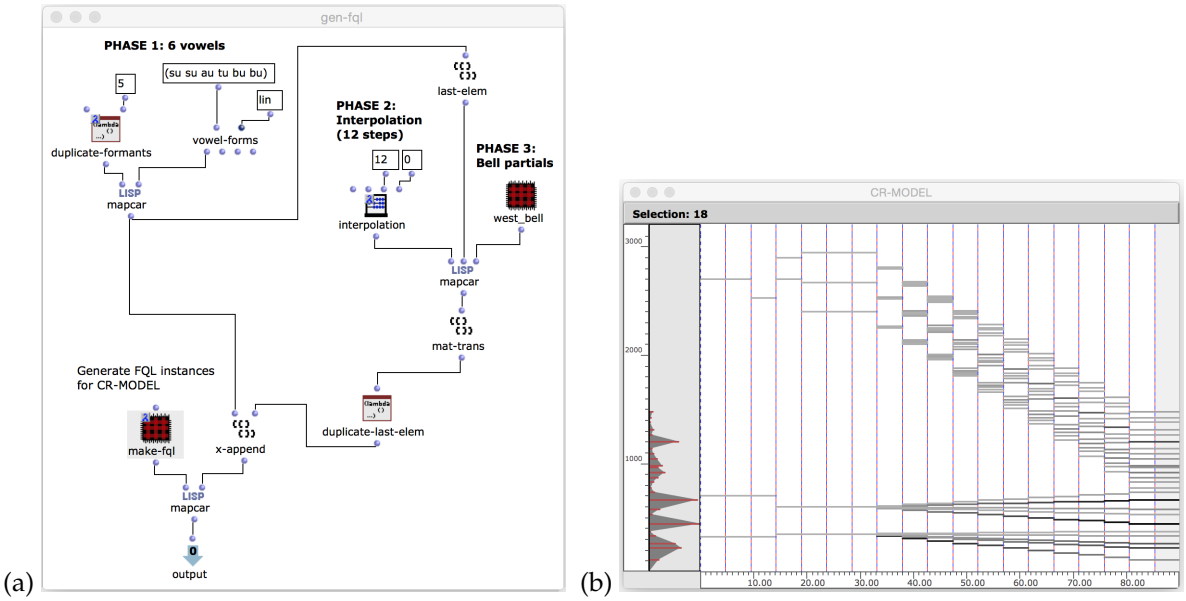


FIGURE 5.15: Génération du modèle pour la synthèse de la mort de la voix du Roi : structuration des données temporelles et fréquentielles. (a) Détail du sous-*patch gen-fql* de la Figure 5.14. (b) contenu de l'objet CR-MODEL.

précédemment (figure 5.10). Entre ces deux phases (Phase 2) la fonction *interpolation* génère 12 états intermédiaires. Sur le modèle des précédents exemples, les contrôleurs des générateurs de FOF à proprement parler seront produits dans une itération, prenant en compte des fonctions de *mapping* permettant de convertir les données abstraites du modèle en paramètres de contrôle, qui seront ensuite interpolés et échantillonnés par le synthétiseur. L'ensemble de la séquence est ainsi pensé et réalisé comme une entité sonore unique, structurée par des évènements mais évoluant continuellement entre ces évènements sur la durée du processus suivant une logique et un certain nombre de règles de développement programmées.

PUBLICATIONS (CHAPITRE 5)

AGON, C., BRESSON, J. et STROPPA, M. (2011). OMChroma : Compositional Control of Sound Synthesis. *Computer Music Journal*, 35(2).

BRESSON, J. et AGON, C. (2006). Temporal Control over Sound Synthesis Processes. *In Proceedings of the Sound and Music Computing conference (SMC'06)*, Marseille, France.

BRESSON, J., FOULON, R. et STROPPA, M. (2013). Reduction as a Transition Controller for Sound Synthesis Events. *In ACM SIGPLAN Workshop on Functional Art, Music, Modeling & Design (FARM'13) – Proceedings of the International Conference on Functional Programmaing (ICFP'13)*, Boston, United States.

BRESSON, J. et MICHON, R. (2011). Implémentations et contrôle du synthétiseur CHANT dans OpenMusic. *In Actes des Journées d'Informatique Musicale (JIM'11)*, Saint Etienne, France.

BRESSON, J. et STROPPA, M. (2011). The Control of the Chant Synthesizer in OpenMusic : Modelling Continuous Aspects in Sound Synthesis. *In Proceedings of the International Computer Music Conference (ICMC'11)*, Huddersfield, UK.

FOULON, R. et BRESSON, J. (2013). Un modèle de contrôle pour la synthèse par fonctions d'ondes formantiques avec OM-Chant. *In Actes des Journées d'Informatique Musicale (JIM'13)*, Saint-Denis, France.

STROPPA, M. et BRESSON, J. (2016). Electronic dramaturgy and computer-aided composition in Re Orso. *In* BRESSON, J., AGON, C. et ASSAYAG, G., éditeurs : *The OM Composer's Book 3*. Editions Delatour / Ircam-Centre Pompidou.

6

SPATIALISATION

La spatialisation sonore consiste à localiser ou déplacer des sources sonores dans un espace de projection réel ou virtuel, en principe équipé d'un dispositif de reproduction sonore multicanal : stéréo, quadriphonie, 5.1, octophonie, jusqu'à des des configurations plus ambitieuses (réseaux linéaires, sphériques ou hémisphériques) de plusieurs dizaines ou centaines de haut-parleurs. Par extension, on englobe également sous ce terme la directivité, le rayonnement des sources ou les autres effets de salle simulés par traitements audionumériques. La liste des techniques et outils de spatialisation développés dans les dernières décennies dénote l'activité d'un des domaines les plus innovants dans la recherche actuelle autour des technologies de la musique et du son. Celle-ci va des techniques les plus élémentaires d'*amplitude panning* distribuant les sources sur les différents haut parleurs — *vector base amplitude panning* (VBAP) (Pulkki, 1997), *distance-based amplitude panning* (DBAP) (Lossius *et al.*, 2009), etc. — qui peuvent être contrôlées à haute résolution et pour un nombre important de sources sonores et de canaux de sortie, à des modèles théoriques plus élaborés tels que la *wave field synthesis* (Berkhout *et al.*, 1993) et l'*ambisonic* 2D/3D (Daniel, 2000) [1] qui réalisent la synthèse de champs acoustiques (« holoponie »), ou encore les techniques binaurales pour la reproduction au casque.

À la suite d'œuvres de compositeurs pionniers comme Edgar Varèse (*Hyperprsim*, 1922 ; *Poème électronique*, 1958), Karlheinz Stockhausen (*Gruppen*, 1957), Boulez (*Repons*, 1981), l'espace et la spatialisation ont été de plus en plus mis en avant dans les processus de composition musicale contemporains (Harley, 1994), et particulièrement popularisés avec le développement de ces nouvelles technologies (Baalman, 2010). Des études telles que (Otondo, 2008) ou (Peters *et al.*, 2011) dressent un paysage éclairant de leurs usages et applications actuelles.

Deux principales approches sont généralement distinguées, impliquant des conceptions musicales de l'espace et des outils de traitement audio spatialisé relativement différentes. L'une est centrée sur la distribution des sources sonores entre un certain nombre de haut-parleurs, utilisés comme relais des processus électroacoustiques (approche parfois qualifiée d'« acousmatique ») ; l'autre est plus axée sur la perception et la localisation virtuelle des sources dans l'espace, utilisant les processeurs audio pour simuler leur position à partir d'une configuration donnée de transducteurs. Dans la suite de cette section, nous nous intéresseront principalement à cette deuxième approche, c'est-à-dire à la localisation et aux

1. La technique de spatialisation *ambisonic* (également appelée HOA, pour *higher-order ambisonics*) consiste à représenter une scène sonore tridimensionnelle comme une combinaison linéaire de fonctions spatiales de base. Cette représentation constitue un encodage modal du champ sonore, indépendant du dispositif de restitution, et restreint à un ordre n caractérisant la résolution spatio-fréquentielle du champ. Le flux audio encodé selon ce formalisme peut ensuite être décodé (avec différents degrés de qualité) sur n'importe quelle configuration de haut parleurs (en principe disposés régulièrement autour de l'auditeur).

déplacements virtuels de sources sonores dans l'espace. Pour autant, les deux approches ne sont pas strictement exclusives : nous avons montré par exemple dans (Garcia *et al.*, 2015) 🖉 que les outils de CAO présentés précédemment pouvaient être utilisés pour l'implémentation de processus de spatialisation sonore inspirés d'une approche acousmatique, centrée sur les canaux de diffusion plus que sur les positions spatiales de sources virtuelles (voir figure 4.4). [2]

L'application au domaine de la spatialisation sonore des principes de contrôle de la synthèse en CAO — et plus spécifiquement, l'intégration de l'espace et du temps dans les modèles compositionnels — était l'une des perspectives les plus immédiates à la suite de mon doctorat. Je l'ai explorée sur plusieurs fronts en parallèle, qui seront dévelopés dans ce chapitre, et notamment via l'encadrement de Marlon Schumacher [⋆] entre 2009 et 2016 dans une collaboration avec le Centre for Interdisciplinary Research in Music Media and Technology (CIRMMT) de l'Université McGill à Montréal.

6.1 DESCRIPTION ET CONTRÔLE

Contrôler un processus de spatialisation sonore peut nécessiter la spécification explicite de nombreux paramètres, liés au positionnement, à la directivité des sources, ou aux caractéristiques de l'espace, dont les valeurs sont par ailleurs susceptibles d'évoluer dans le temps. La plupart des outils disponibles actuellement pour spatialiser les sources sonores sont cependant déployés comme processeurs audio temps-réel sous forme de *plug-ins* dans des stations audio-numériques, ou de modules dans des environnements comme Max ou PureData (Jot et Warusfel, 1995; Schacher et Kocher, 2006), c'est-à-dire, sans prise en compte explicite d'une dimension ou d'une structure temporelle liée à la construction musicale. Des projets voient le jour visant à abstraire les couches de contrôle des moteurs de rendu spatialisé — voir par exemple (Ahrens *et al.*, 2008). Plus rares, en revanche, sont les projets et outils réellement ancrés dans une orientation compositionnelle. On pourra citer par exemple dans ce cas des travaux comme ceux de (Todoroff *et al.*, 1997), (Pottier, 1998) ou (Lopez-Lezcano, 2008).

Jusqu'à la fin des années 2000, deux projets avaient été réalisés dans OpenMusic en lien avec la spatialisation sonore. OpenSpace (Delerue et Agon, 1999) était un outil intégrant le logiciel MusicSpace (Delerue, 2004), qui contrôlait les positions de sources sonores dans l'espace à l'aide d'un système de contraintes, dans OpenMusic. Cette intégration dans l'environnement de CAO permettait de contrôler le système en créant et modifiant ces contraintes dynamiquement dans un déroulement temporel. Un autre projet important fut la bibliothèque OMSPAT (Nouno et Agon, 2002), qui permettait d'associer trajectoires et paramètres spatiaux générés dans OpenMusic à des sources sonores, par l'intermédiaire d'une structure matricielle formatée et transférée à une interface de contrôle pour le Spatialisateur de l'IRCAM — également appelé SPAT (Jot, 1999). [3] Les travaux présentés dans ce chapitre se situent donc dans la continuité de ces projets (Bresson *et al.*, 2010; Bresson, 2012) 🖉.

TRAJECTOIRES. Les trajectoires des sources sonores dans l'espace constituent généralement le principal aspect du contrôle de la spatialisation entrant en compte dans les structures

2. Le chapitre de Hans Tutschku dans *The OM Composer's Book 2* (Tutschku, 2008) est un autre exemple de processus de spatialisation sonore basé sur la distribution de sources entre différents canaux de diffusion, réalisée grâce à la bibliothèque OM-SUPERVP.

3. Cette bibliothèque fut utilisée par exemple par Gilbert Nouno avec le compositeur Brian Ferneyhough pour la création de la pièce *Stellæ for failed times* (2001) — voir (Nouno, 2008).

et processus compositonnels. Nous utilisons pour cela des représentations de courbes 2D ou 3D dans OpenMusic, auxquelles est attachée une dimension temporelle. Les trajectoires peuvent ainsi être générées algorithmiquement à partir de spécifications mathématiques, de fonctions de ré-échantillonnage ou de courbes paramétriques, mais également à l'aide d'opérations manuelles dans des éditeurs graphiques (figure 6.1).

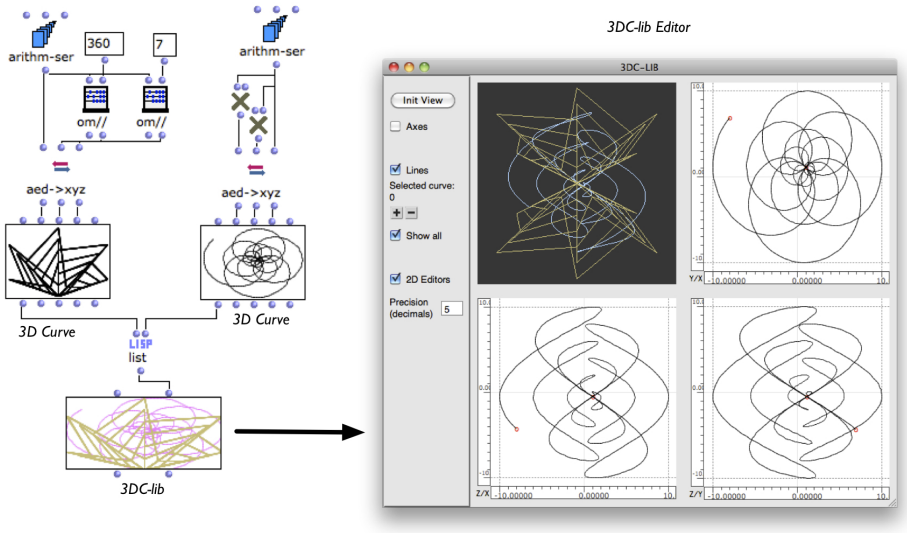


FIGURE 6.1: Génération de trajectoires dans OpenMusic. Processus algorithmiques et éditeurs graphiques.

Comme nous l'avons vu précédemment, les paramètres et signaux de description sonore mis en jeu dans les processus de CAO doivent répondre à des besoins parfois contradictoires d'abstraction (pour le contexte symbolique du processus compositionnel) et de précision (données échantillonnées, parfois à haute définition, pour le rendu sonore). C'est particulièrement vrai pour les représentations spatiales et leurs aspects temporels : au niveau compositionnel, on souhaitera par exemple déterminer un début et une fin pour une trajectoire de source sonore dans l'espace ; éventuellement certains points d'intérêt ou d'articulation, synchronisés avec une date ou un autre élément musical. Les autres points peuvent être positionnés et/ou temporisés implicitement ; cependant, leurs valeurs précises (spatiales et temporelles) devront finalement être calculées (par exemple par interpolation) afin de produire des signaux de contrôle. Les objets proposés dans OpenMusic pour la définition de trajectoires permettent ainsi de définir des points de contrôle, des stratégies d'échantillonnages (à vitesse constante, à temps constant entre points de contrôle, etc.) et de traiter ces paramètres spatiaux à différents niveaux d'abstraction. [4]

REPRÉSENTATION DES SCÈNES SONORES. Sur le modèle des précédents travaux présentés autour des systèmes de contrôle de la synthèse sonore, nous utilisons des structures matricielles bi-dimensionnelles pour la représentation des processus et scènes sonores spatialisés. Ici, chaque ligne d'une matrice représente une source sonore à spatialiser (une, dans les cas les plus simples ; plusieurs centaines dans des cas plus complexes), et chaque colonne représente un champ de description (principalement, des positions ou trajectoires

4. Dans la suite de ce mémoire (section 7.3), nous présenterons une généralisation de cette idée avec un modèle abstrait des structures temporelles dans les objets musicaux (Garcia *et al.*, 2017).

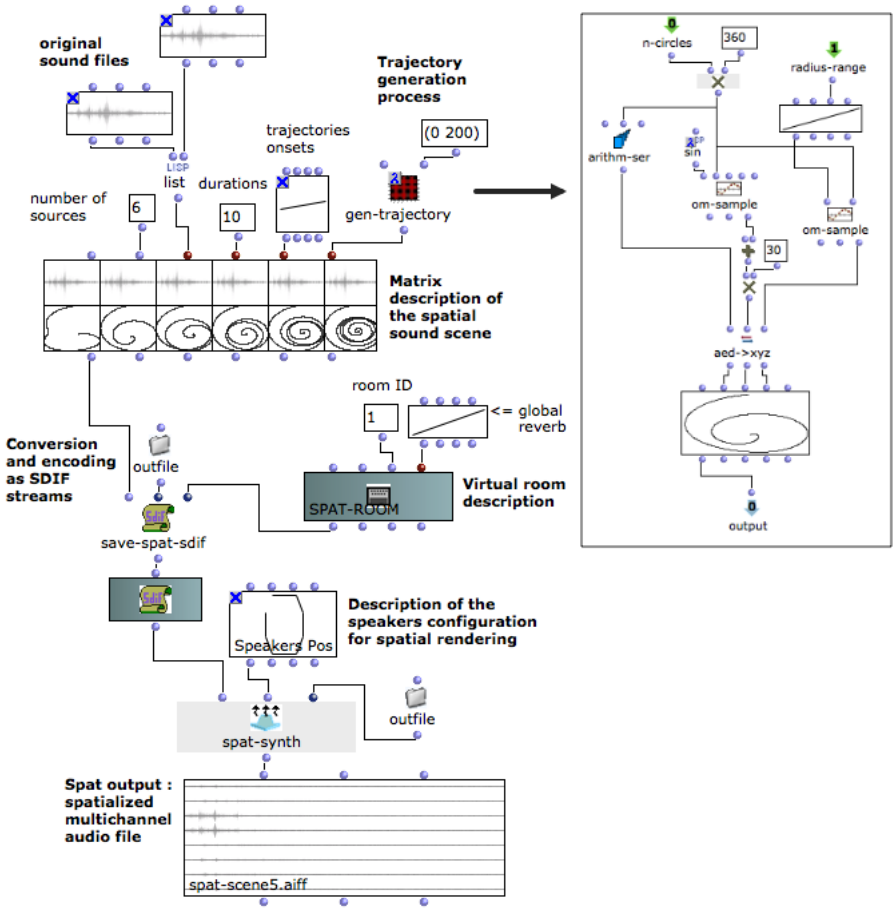


FIGURE 6.2: Représentation et synthèse d'une scène sonore spatialisée avec OM-SPAT 2. La matrice de description contient six sources sonores et des paramètres de spatialisation, spécifiés littéralement (*durations*, *onsets*...) ou sous forme « fonctionnelle » (génération de trajectoire sous forme de fonction *lambda*). La description de la scène, complétée par une description des caractéristiques de la salle virtuelle, est encodée sous forme d'un fichier SDIF et finalement reproduite sous forme d'un fichier audio multi-canal selon un nombre et une configuration donnés des haut-parleurs.

bi-/tri-dimensionnelles, mais également des éléments de description de la directivité, du rayonnement, de la réverbération des sources, etc.). Cette description est ainsi homogène aux autres outils de contrôle (nous verrons comment ceux-ci peuvent être combinés avec la bibliothèque OMPRISMA, section 6.3) et stable quelle que soit la taille ou la complexité de la scène sonore. En regroupant l'ensemble des descriptions dans une même structure, elle permet, comme nous l'avons vu par exemple dans la section 5.1 pour les composants de synthèse sonore, d'associer aux différents composants d'une scène spatialisée une même trajectoire, d'établir des rapports entre leurs trajectoires respectives, ou encore de définir un processus générant dynamiquement l'ensemble de ces trajectoires. La figure 6.2 montre un exemple de construction d'une « matrice de spatialisation », associant sources sonores et paramètres de spatialisation.

STOCKAGE ET TRANSFERT DE DONNÉES. La standardisation des formats d'échange pour la spatialisation est une question importante, dans la mesure où les tâches de contrôle, de calcul et de reproduction sonore spatialisée sont le plus souvent réalisées dans des environnements et contextes différents. [5] Dans (Bresson et Schumacher, 2011) nous avons proposé une solution de stockage et de transfert des descriptions scènes sonores utilisant le format SDIF, dont nous avions besoin afin de transmettre ces descriptions à un moteur de spatialisation. [6] Cette proposition se voulait tournée vers l'aspect « dénotationnel » des descriptions, c'est-à-dire décrivant des trajectoires et signaux de contrôle échantillonnés dans leur résolution finale, ne nécessitant pas d'interprétation supplémentaire par le processus de spatialisation (qui pourrait ainsi être opéré par n'importe quel outil compatible avec le format). La temporisation libre (non nécessairement échantillonnée) des flux de données SDIF et la flexibilité du format se présentaient comme des caractéristiques adaptées pour ce cas de figure. Différents types de *frames* et matrices SDIF ont été proposés, permettant une description exhaustive de nos scènes sonores. La flexibilité du format permet également de structurer les scènes en flux indépendants par source sonore (figure 6.3-a), ou de rassembler les sources dans des descriptions synchrones au sein d'un flux unique (figure 6.3-b). Divers paramètres tels que les effets de salles peuvent enfin être inclus, soit dans les flux existants (par exemple attachés à une source ou un ensemble de sources), soit dans des flux indépendants.

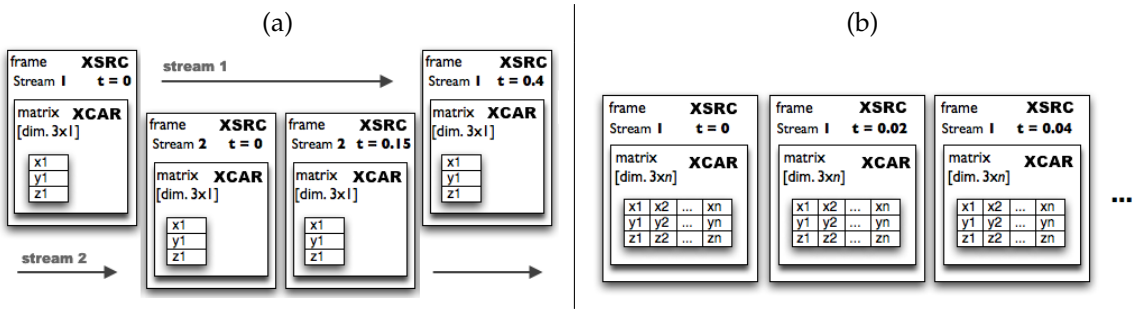


FIGURE 6.3: Descriptions de scènes spatiales au format SDIF. (a) Flux indépendants par source. (b) Flux synchrone pour un groupe de sources.

5. Voir par exemple le panel dédié à ce sujet lors de la conférence ICMC 2008 (Kendall *et al.*, 2008).
6. Différentes propositions de formats et initiatives de standardisation, comme SpatDIF (Peters *et al.*, 2013), ont été développées depuis.

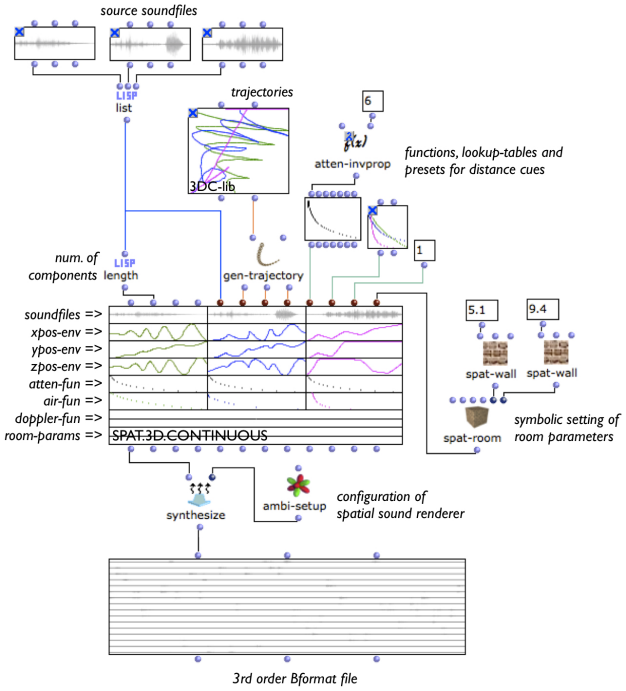


FIGURE 6.4: OMPRISMA : spatialisation de trois sources sonores utilisant la technique HOA (*ambisonic* 3e ordre).

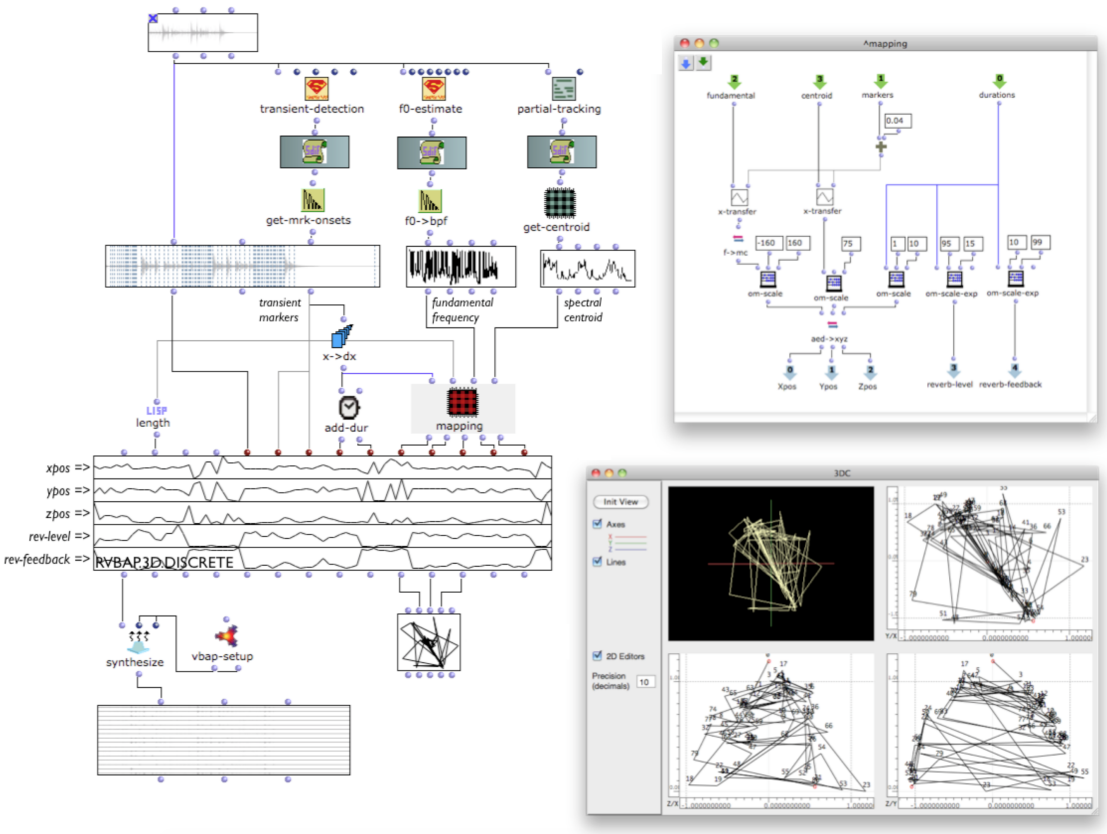


FIGURE 6.5: OMPRISMA : spatialisation à partir de données d'analyse du signal source.

6.2 BIBLIOTHÈQUES SPÉCIALISÉES DANS OPENMUSIC

OM-SPAT 2. Les outils présentés jusqu'ici nous ont permis de mettre sur pied une nouvelle version de la bibliothèque OM-SPAT, désormais distribuée parmi les logiciels du Forum IRCAM. Celle-ci englobe les descriptions de sources et de scènes spatiales dans un déroulé temporel, permet leur échantillonnage puis leur encodage sous forme de fichier SDIF, et un rendu spatialisé via une version *offline* du Spatialisateur de l'IRCAM (Carpentier *et al.*, 2015). L'ensemble des paramètres du Spatialisateur peut être contrôlé via cette bibliothèque. Différents modes de rendu peuvent être appliqués (y compris *vbap*, binaural, HOA/*ambisonics*, etc.), sur un nombre arbitraire et une configuration libre des hauts-parleurs.

La figure 6.2 est un processus complet réalisé avec OM-SPAT. Le compositeur Fabio De Sanctis De Benedictis utilise cette bibliothèque dans [🕮 17].

OMPRISMA. Le travail de Marlon Schumacher [★] étend les concepts de la bibliothèque OMCHROMA au domaine de la spatialisation sonore. La bibliothèque OMPRISMA développée dans sa thèse comprend un nombre important de *classes* qui, sur le modèle des classes correspondant aux instruments de synthèse dans OMCHROMA, correspondent ici à des processeurs audio spatialisés, implémentant (toujours dans Csound) un ensemble de techniques et algorithmes de spatialisation : stéréo, quadriphonie, VBAP, DBAP, *ambisonic*, ainsi que des modules optionnels de simulation d'acoustique des salles et de dynamique des sources sonores (effet Doppler, etc.). Les principes hérités du système OMCHROMA (voir section 5.1) permettent une polyphonie et une complexité virtuellement illimitées pour ces processus de spatialisation, appliqués à des sources sonores données comme paramètres d'entrée et connectés à des processus compositionnels dans OpenMusic (Schumacher et Bresson, 2010a) 🖉. L'interchangeabilité des différentes classes permet de concevoir les scènes sonores de façon relativement indépendante de la méthode de spatialisation choisie.

Les figures 6.4 et 6.5 montrent des exemples de processus de spatialisation réalisés avec OMPRISMA. Sur la figure 6.4 la classe (et donc, l'algorithme Csound) *spat.3D* est utilisée pour spatialiser trois sources sonores. Les sources, des trajectoires en trois dimensions et un certain nombre d'autres paramètres acoustiques sont rassemblés dans une matrice, transformée en un fichier audio encodé au format *ambisonic* de troisième ordre. Sur la figure 6.5 une seule source sonore est spatialisée, cette fois à l'aide de l'algorithme *rvbap.3D* dont les paramètres son déterminés à partir d'analyses de cette même source (transitoires d'attaque, fréquence fondamentale et centroïde spectral). D'autres procédures de paramétrage peuvent être implémentées, par exemple pour produire des sources composites complexes dynamiquement à partir de spécifications de haut-niveau dans la matrice de contrôle — des exemples sont donnés dans (Schumacher et Bresson, 2010a) 🖉 et (Schumacher et Bresson, 2010b) 🖉.

Le compositeur Christopher Trapani présente dans [🕮 13] une application de cette bibliothèque.

6.3 SYNTHÈSE SONORE SPATIALISÉE

Le développement de la bibliothèque OMPRISMA sur le modèle, et en tant qu'extension de OMCHROMA nous a conduit à concevoir un système original de combinaison des processus de synthèse et de spatialisation sonore appelé *synthèse sonore spatialisée* (Schumacher et Bresson, 2010b) 🖉. Généralisant des principes existants tels que la spatialisation par décomposition spectrale (Kim-Boyle, 2008; Topper *et al.*, 2002), ou encore les modèles spatiaux « granulaires » (McLeran *et al.*, 2008; Wilson, 2008), ce système traite la spatialisation au niveau de la

production des sons, afin d'intégrer les aspects de contrôle et de configuration spatiale dans les algorithmes de synthèse (on ne spatialise donc plus nécessairement des sources sonores pré-existantes, mais on construit des éléments sonores dans l'espace). D'un point de vue pragmatique dans notre contexte, il s'agit de permettre une libre combinaison des classes de synthèse (pour la plupart contenues dans OMCHROMA) et des classes de spatialisation (dans OMPRISMA), de sorte que chaque composant d'un processus de synthèse (partiel ou oscillateur, grain sonore, FOF, etc.) soit enrichi de paramètres et de traitements spatiaux.

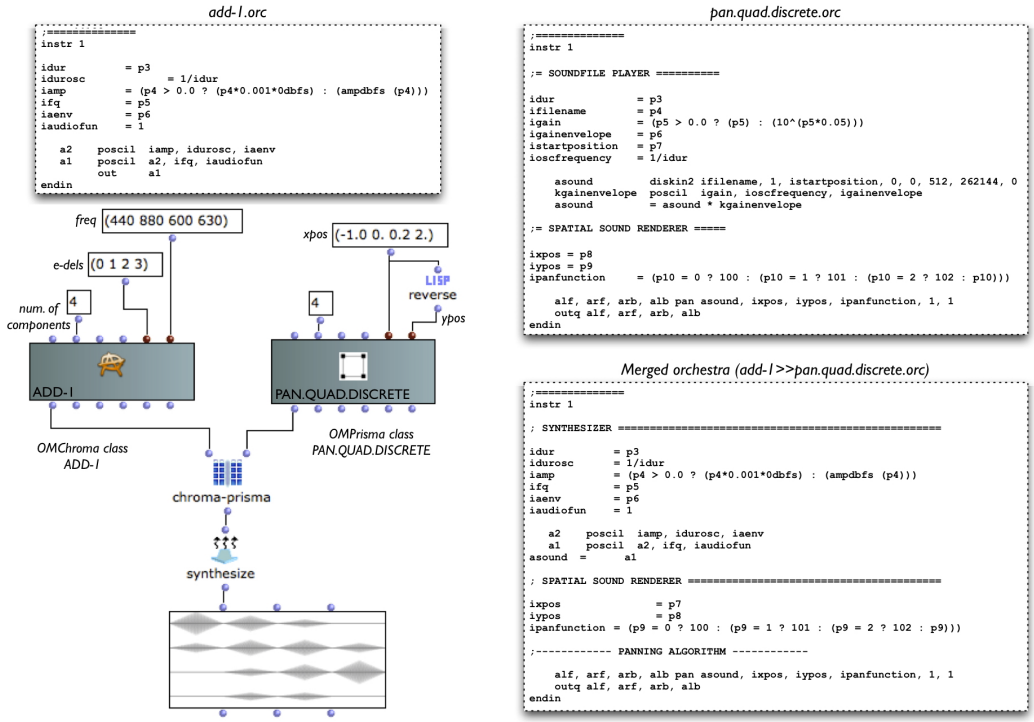


FIGURE 6.6: OMPRISMA : fusion des instruments Csound pour la synthèse et la spatialisation de sources/composants sonores.

Ce principe est mis en œuvre par une hybridation de classes, opérée dans un *patch* OpenMusic de manière transparente du point de vue de l'utilisateur grâce aux outils de programmation par meta-objets (définition dynamique de classes) et à un algorithme de fusion des programmes (*instruments*) Csound. Le code Csound de deux classes (synthèse et spatialisation) est parcouru par un *parser* afin de déterminer et connecter via des variables les points de sortie des algorithmes de synthèse et les points d'entrée des algorithmes de spatialisation. Un nouveau code est généré, puis utilisé pour définir une nouvelle classe, instanciée par rassemblement des attributs des deux instances à l'origine de la fusion (voir figure 6.6).

Ce procédé permet donc d'opérer n'importe quelle combinaison de classes. Les processus de spatialisation sonore sont appliqués au niveau microscopique des composants sonores, quels que soient le nombre et la complexité de ces composants, au sein de morphologies spatiales complexes et structurées. Un exemple combinant la synthèse additive et la spatialisation *ambisonic* 3D est présenté sur la figure 6.7.

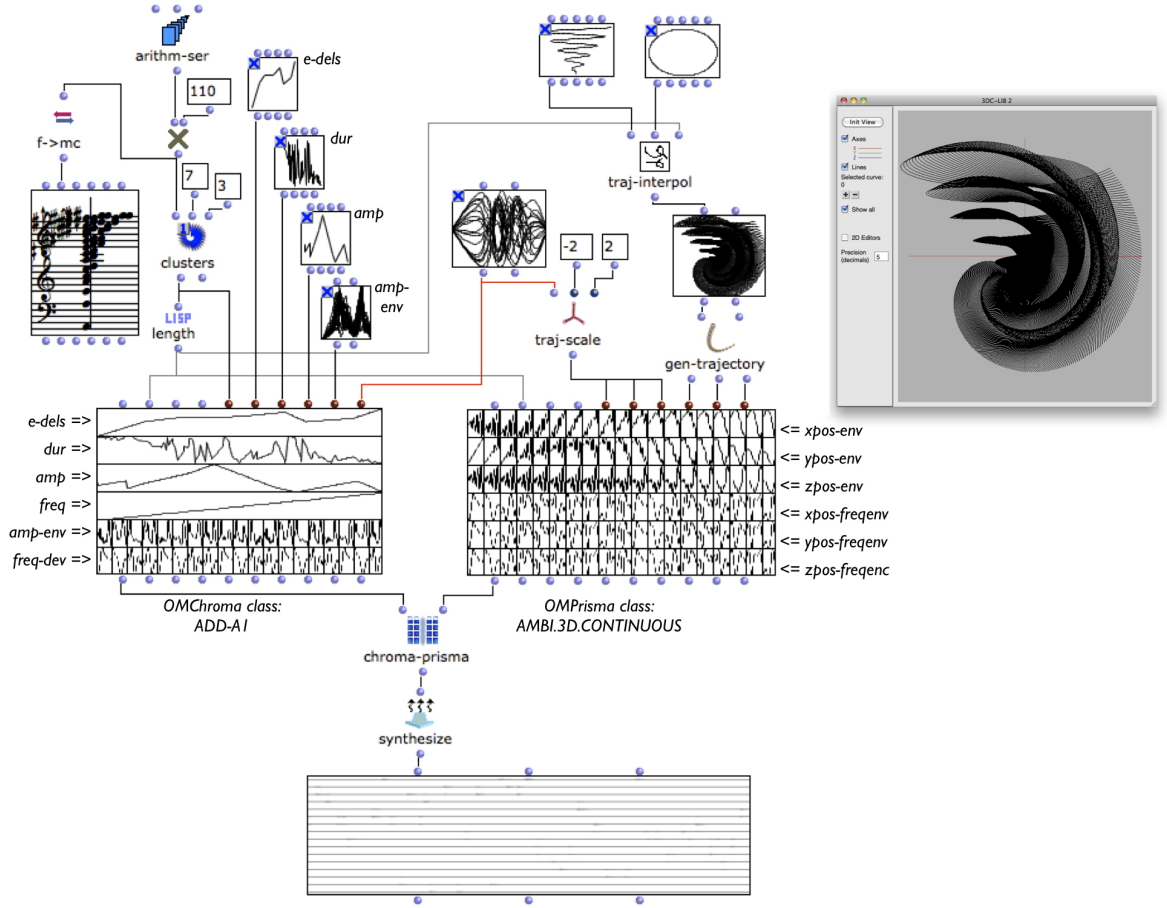


FIGURE 6.7: Synthèse sonore spatialisée avec OMPRISMA : spécification de trajectoires et spatialisation individuelle des partiels d'un processus de synthèse additive à l'aide de la technique *ambisonic* 3D.

6.4 VERS LE TEMPS-RÉEL

Nous présentons dans cette section des travaux préliminaires visant à connecter les outils de composition (où des processus musicaux formalisés produisent des structures temporelles) aux systèmes temps-réel (où se produisent généralement les calculs audio et le rendu sonore, notamment lors de concerts). Cette connexion sera généralisée dans le chapitre suivant, proposée comme réponse aux problèmes émergeant avec le déploiement de systèmes interactifs temps-réel en relation aux structures musicales, toujours plus complexes, produites par les compositeurs dans les environnements de CAO.

RESTITUTION DES DESCRIPTIONS DE SCÈNES SONORES. L'encodage intermédiaire de données de spatialisation au format SDIF dans OM-SPAT permet d'envisager une connexion modulaire à des environnements de spatialisation audio. Les données de spatialisation, préparées dans l'environnement de CAO et organisées en flux temporisés, peuvent être lues comme une structure musicale et envoyées à une application externe pour la réalisation du rendu sonore (sur le modèle des évènements MIDI envoyés à un synthétiseur pour le rendu sonore d'une partition).

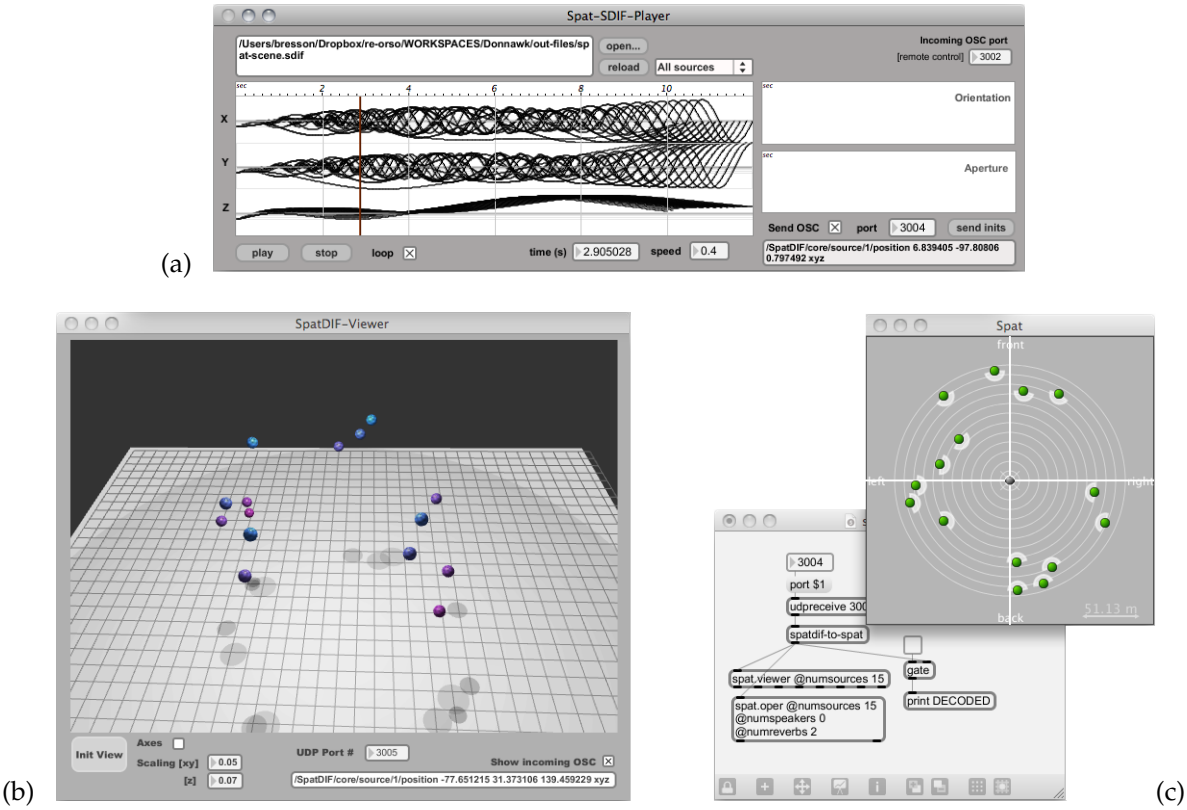


FIGURE 6.8: (a) Lecture/*streaming* des signaux de contrôle produits dans OpenMusic pour une scène sonore spatialisée (SPAT-SDIF-PLAYER). Réception dans des environnements externes temps-réel : (b) visualisation des trajectoires en 3D avec SPATDIF-VIEWER, et (c) et visualisation et rendu spatialisé avec la bibliothèque SPAT dans Max.

Suivant ce principe l'application SPAT-SDIF-PLAYER visible sur la figure 6.8-a permet de charger un fichier de description au format SDIF créé par OM-SPAT, et de l'« exécuter », non pas via un rendu audio spatialisé, mais par l'envoi (*streaming*) via UDP de messages de contrôle au format OSC — OpenSoundControl (Wright, 2005). Cette application propose les commandes classiques des *players* musicaux : *play/stop/pause/loop* etc., et communique avec OpenMusic (également via OSC) pour un contrôle distant. [7] Sur la figure 6.8 on peut voir deux logiciels différent recevant et interprétant ces données : (b) un programme de visualisation 3D, et (c) un *patch* Max convertissant ces messages en contrôleurs pour le module *spat.oper* de la bibliothèque SPAT (Carpentier *et al.*, 2015). [8]

7. Les systèmes d'ordonnancement et de rendu des structures musicales que nous avons développés plus tard, qui seront présentés dans le chapitre 7 (section 7.3), nous permettront de réaliser cette lecture des données de contrôle sous forme de messages OSC, ou même de rendu audio spatialisé, directement dans OpenMusic.

8. Malgré la flexibilité de ce protocole de communication par *streaming* des données, des limitations en termes de synchronisation ou de délai peuvent apparaître lorsque la densité du flux de messages augmente (par exemple, avec le nombre de sources sonores, de paramètres de spatialisation, la fréquence d'échantillonnage, etc.). Nous proposerons dans le chapitre 7 (section 7.2) des solutions pour une communication plus compacte et structurée dans ce type de situation.

FIGURE 6.9: TRAJECTOIRES : une application mobile pour la saisie, l'édition et le rendu de trajectoires de contrôle pour la spatialisation sonore.

OUTILS INTERACTIFS POUR LA SAISIE ET L'ÉDITION DES DONNÉES. J'ai encadré dans le cadre d'un projet ANR Jeunes Chercheuses et Jeunes Chercheurs [9] le post-doctorat de Jérémie Garcia (2014-2015) [⋆], qui s'est intéressé aux interfaces de contrôle pour la spatialisation sonore, notamment par la connexion des outils de représentation des trajectoires tri-dimensionnelles existant dans OpenMusic à des modalités interactives de saisie des données spatio-temporelles. Dans (Garcia *et al.*, 2015a) des premiers prototypes sont présentés, utilisant par exemple un serveur OSC dans les éditeurs graphiques pour capter des données saisies avec des contrôleurs gestuels.

Le stage M2 de Xavier Favory (2015) [⋆] s'est inscrit également dans cette dynamique avec le développement d'un application mobile (TRAJECTOIRES) pour tablette et smartphone, dédiée à la saisie et à la transformation de trajectoires pour le contrôle de la spatialisation (voire figure 6.9) (Garcia *et al.*, 2016). La conception de cette application a été réalisée suivant un mode de design participatif à partir d'une série d'entretiens et de collaborations suivies avec des compositeurs (Favory *et al.*, 2015). Elle a été utilisée par plusieurs d'entre eux, étudiants au cursus de composition ou résidents en production à l'IRCAM, pour la création de projets présentés au public en 2015.

UNE INTERFACE ENTRE CAO ET SPATIALISATION TEMPS RÉEL. Dans le même contexte et en collaboration avec Thibaut Carpentier de l'équipe Espaces Acoustiques et Cognitifs (EAC) de l'IRCAM, nous avons souhaité aborder le contrôle de la spatialisation suivant une nouvelle approche combinant la structuration temporelle permise par les outils « hors-temps » de la CAO et les fonctionnalités de représentation et de rendu audio spatialisé offertes par les systèmes temps-réel.

SPAT-SCENE (Garcia *et al.*, 2017b) est un projet implémenté grâce une nouvelle intégration dynamique entre notre environnement de CAO et la bibliothèque SPAT. [10] Il s'agit essentielle-

9. EFFICACe ANR-13-JS02-0004 (voir chapitre 7).

10. Cette intégration concerne également les interfaces graphiques, qui sont empruntées à la bibliothèque et offrent à l'utilisateur une vue en tous points identique à celle proposée par SPAT dans les environnements temps-réel (Max, PureData, etc.) — voir à ce sujet la présentation de Thibaut Carpentier au Juce Summit 2015 : *Integrating JUCE-based GUI in Max/MSP and OpenMusic*.

ment d'un conteneur/éditeur musical construit sur le modèle des matrices OM-SPAT à partir d'un ensemble de sources sonores et de trajectoires, générées algorithmiquement ou à l'aide d'interfaces graphiques. L'interface utilisateur propose une représentation orthogonale de ces trajectoires : dans l'espace, grâce aux vues de SPAT ou à des représentations en 3D, et dans le temps, grâce à des « pistes » ou *time-lines* éditables pour les différentes sources (voir figure 6.10). Les mouvements individuels, mais aussi les éventuelles synchronisations des sources dans l'espace et dans le temps (ainsi que les différentes transformations d'étirement ou de compression qui peuvent en découler) peuvent ainsi être contrôlés dans une interface relativement simple, connectée à des fonctions de rendu paramétrables (audio spatialisé, messages OSC, etc.).

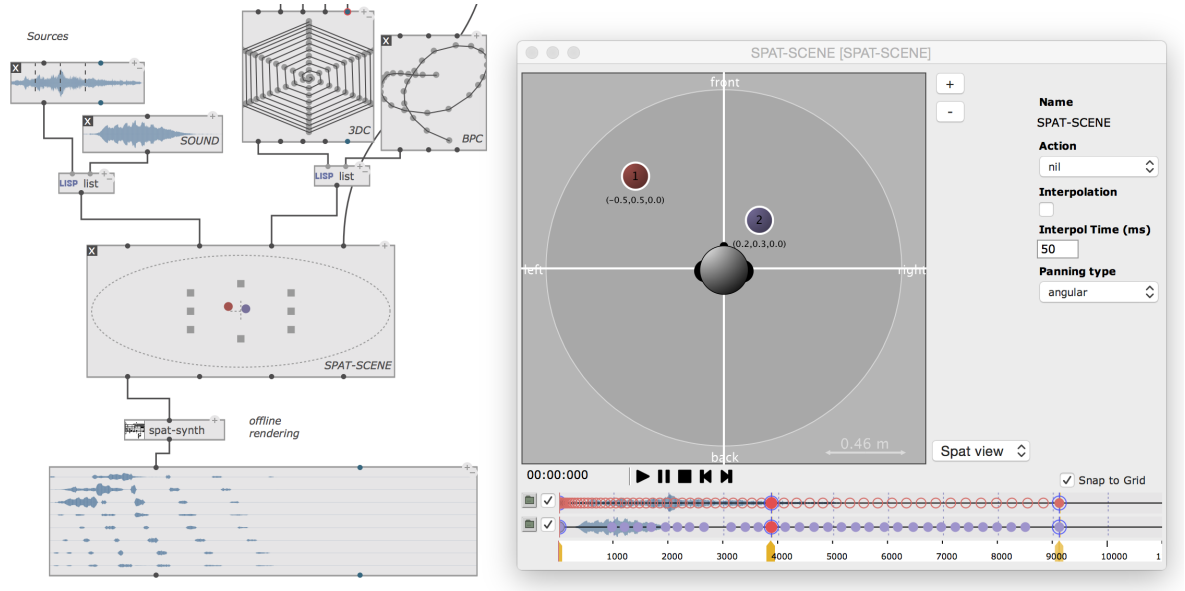


FIGURE 6.10: SPAT-SCENE : un éditeur OpenMusic connecté à la bibliothèque SPAT pour un contrôle compositionnel et interactif des scènes sonores spatialisées.

L'un des objectifs de ce projet était l'expérimentation de modalités de description et de contrôle des processus audio spatialisés permettant à la fois une approche musicale structurée (à l'aide notamment de la programmation et de l'édition de constructions temporelles), et une approche gestuelle et intuitive permettant un retour (audio, graphique...) dans les phases d'idéation, d'expérimentation et de production musicale.[11] Il s'appuie pour cela sur un certain nombre de technologies que nous décrirons dans le chapitre suivant ; notamment, une description générique des états et messages de contrôles via le format OSC, et un rendu graphique et audio spatialisé en temps réel grâce à une architecture d'ordonnancement adaptée.[12]

11. Lors d'une précédente collaboration avec l'équipe EAC nous avions déjà abordé cette idée de lier les processus temporels de la CAO au rendu auditif temps-réel, pour une meilleure rétroaction dans expériences perception auditive spatiale (Viaud-Delmon *et al.*, 2007).

12. A l'heure de la rédaction de ce mémoire, la compositrice Savannah Agger réalise dans le cadre d'une résidence « recherche musicale » à l'IRCAM une pièce basée (entre autres) sur cet outil, et sur ses principes étendus à d'autres traitements numériques des signaux sonores.

PUBLICATIONS (CHAPITRE 6)

Bresson, J. (2012). Spatial Structures Programming for Music. *In Spatial Computing Workshop (SCW)*, Valencia, Spain. Colocated with AAMAS'2012 (International Conference on Autonomous Agents and Multiagent Systems).

Bresson, J., Agon, C. et Schumacher, M. (2010). Représentation des données de contrôle pour la spatialisation dans OpenMusic. *In Actes des Journées d'Informatique Musicale (JIM'10)*, Rennes, France.

Bresson, J. et Schumacher, M. (2011). Representation and Interchange of Sound Spatialization Data for Compositional Applications. *In Proceedings of the International Computer Music Conference (ICMC'11)*, Huddersfield, UK.

Favory, X., Garcia, J. et Bresson, J. (2015). Trajectoires : une application mobile pour le contrôle et l'écriture de la spatialisation sonore. *In 27ème conférence francophone sur l'Interaction Homme-Machine (IHM'15)*, Toulouse, France. ACM.

Garcia, J., Bouche, D. et Bresson, J. (2017a). Timed Sequences : A Framework for Computer-Aided Composition with Temporal Structures. *In Proceedings of the International Conference on Technologies for Music Notation and Representation (TENOR'17)*, A Coruña, Spain.

Garcia, J., Bresson, J. et Carpentier, T. (2015a). Towards Interactive Authoring Tools for Composing Spatialization. *In IEEE 10th Symposium on 3D User Interfaces*, Arles, France.

Garcia, J., Bresson, J., Schumacher, M., Carpentier, T. et Favory, X. (2015b). Tools and Applications for Interactive-Algorithmic Control of Sound Spatialization in OpenMusic. *In inSonic2015, Aesthetics of Spatial Audio in Sound, Music and Sound Art*, Karlsruhe, Germany.

Garcia, J., Carpentier, T. et Bresson, J. (2017b). Interactive-Compositional Authoring of Sound Spatialization. *Journal of New Music Research*, 46(1).

Garcia, J., Favory, X. et Bresson, J. (2016). Trajectoires : a Mobile Application for Controlling Sound Spatialization. *In CHI EA'16 : ACM Extended Abstracts on Human Factors in Computing Systems*, San Jose, United States.

Schumacher, M. et Bresson, J. (2010a). Compositional Control of Periphonic Sound Spatialization. *In 2nd International Symposium on Ambisonics and Spherical Acoustics*, Paris, France.

Schumacher, M. et Bresson, J. (2010b). Spatial Sound Synthesis in Computer-Aided Composition. *Organised Sound*, 15(3).

Viaud-Delmon, I., Bresson, J., Pachet, F., Bevilacqua, F., Roy, P. et Warusfel, O. (2007). Eartoy : interactions ludiques par l'audition. *In Actes des Journées d'Informatique Musicale (JIM'07)*, Lyon, France.

7

PROCESSUS COMPOSITIONNELS INTERACTIFS

Ce dernier chapitre traite des aspects les plus récents de mes travaux autour des systèmes réactifs pour la composition assistée par ordinateur, réalisés principalement dans le cadre du projet ANR EFFICACe [1] que j'ai conduit entre 2013 et 2017 (Bresson *et al.*, 2015) ✎. Ce projet visait à réunir les systèmes de composition assistée par ordinateur et les systèmes interactifs et/ou de traitement audio temps-réel, [2] en intégrant notre logiciel de CAO dans un réseau d'interactions structurées avec son environnement, envisagées aussi bien dans une optique compositionnelle (lorsque les processus de CAO mettent en jeu des dispositifs d'entrée/sortie ou communiquent de manière asynchrone avec d'autres applications), que dans un contexte « performatif » (lorsque la musique est exécutée et qu'elle fait appel à des processus génératifs formalisés, à la manipulation de données structurées, ou à l'exécution de calculs complexes). [3] Dans ce contexte les compositeurs nécessitent des outils de programmation et de calcul permettant de répondre à des évènements par des effets immédiats, mais également par des processus plus complexes déployés dans le temps.

Nous avons noté dans le chapitre 2 comment un pas significatif dans le développement des environnements de CAO fut franchi (notamment grâce à la programmation visuelle) en permettant à l'utilisateur d'intervenir et d'interagir à différents stades du calcul des données musicales, et nous plaçons donc ici également dans la continuité de cette évolution.

Nous avons abordé aussi dans le chapitre 2 la question de l'exécution des programmes dans OpenMusic : en contraste avec la majorité des environnements musicaux interactifs, où les programmes s'exécutent périodiquement ou de manière réactive sous l'impulsion de flux d'évènements, de métronomes ou de signaux (exécution *event-*, ou *time-driven*), les calculs dans OpenMusic sont déclenchés par des requêtes de calculs ou de mises à jour locales des données, émanant de l'utilisateur (exécution *demand-driven*) — voir figure 7.1. L'exécution des programmes et le rendu musical (jeu/écoute) des données produites prennent donc place dans des phases de temps distinctes, alors qu'elles sont pratiquement confondues dans les systèmes que nous avons qualifiés de « performatifs » — c'est la principale raison pour

1. *Extended Framework for "In-time" Computer-Aided Composition*. ANR-13-JS02-0004, Programme Jeunes Chercheuses et Jeunes Chercheurs (JCJC), édition 2013 — `http://repmus.ircam.fr/efficace/`

2. Voir le numéro spécial de *Journal of New Music Research* dédié à ce sujet (Bresson et Chadabe, 2017) ✎.

3. Des précédents exemples d'applications mixtes entremêlant composition et performance sont par exemple les systèmes de *live-coding*, où la programmation des structures musicales a lieu sur scène devant le public (Wang et Cook, 2004), ou encore les systèmes génératifs d'accompagnement ou d'improvisation, où des parties musicales structurées sont produites à la volée via un apprentissage basé sur le jeu d'un musicien (Assayag, 2016).

laquelle ceux-ci ne permettent pas, ou difficilement, la construction de structures temporelles complexes sur le long terme. [4]

Xenakis (1971) suggérait une distinction entre les structures musicales « en-temps » et « hors-temps », que l'on peut retrouver dans cette problématique. Les structures « hors-temps » ont des règles de compositions indépendantes du temps « réel » et de leur intégration dans une séquence temporelle. Les structures « en temps » constituent une application de ces structures « hors-temps » sur une structure temporelle. Le matériau musical est donc pensé et construit suivant des formalismes spécifiques, et déployé dans le temps suivant d'autres formalismes. De ce point de vue, les processus de composition assistée par ordinateur sont généralement considérés comme des systèmes « hors-temps », et ceux produisant les structures musicales dans le temps réel (par exemple via des instruments ou des synthétiseurs audio) peuvent être vus comme les processus déployant la musique « en-temps ».

En décloisonnant la CAO du domaine hors-temps, nous esquissons donc ici un cadre paradigmatique original pour le calcul et l'exécution des structures musicales. Nous aborderons celui-ci sous plusieurs angles, présentés dans les différentes sections de ce chapitre, tous regroupés dans l'implémentation d'une nouvelle génération d'environnement de CAO. Successivement, nous présenterons des travaux liés à l'extension réactive des processus compositionnels (section 7.1), à la communication entre applications (section 7.2), aux systèmes d'ordonnancement et de rendu des structures musicales (section 7.3), puis nous concluerons provisoirement en revenant sur les questions du traitement du signal audio (section 7.4).

7.1 EXTENSION RÉACTIVE D'OPENMUSIC

Nous avons présenté dans (Bresson et Giavitto, 2014) un modèle réactif pour l'exécution des programmes visuels dans OpenMusic, combinant les paradigmes de calcul et de programmation *demand-driven* et *event-driven*. Ce travail propose deux principales contributions : une sémantique dénotationnelle du langage visuel et un formalisme simple permettant d'étendre celle-ci pour intégrer un comportement réactif.

La sémantique proposée prend en compte dans un même modèle la construction et l'exécution des programmes visuels. Nous considérons la programmation dans OpenMusic comme une construction incrémentale de graphes fonctionnels, entremêlée à des manipulations de ces graphes pour la saisie ou l'édition de données, ou encore pour la commande d'évaluations partielles. Ces graphes sont donc des ensembles de boîtes (associées à des fonctions dont la sémantique est définie par ailleurs), d'arêtes, et de fonctions totales des arêtes vers les entrées/sorties des boîtes définissant leur connectivité. L'évaluation à partir d'une boîte correspond à un parcours *en amont* du graphe, se propageant d'une boîte à l'autre et conduisant à la mise à jour de portions spécifiques de ce graphe (voir figure 7.1-a). Le résultat de l'évaluation d'une boîte est enregistré temporairement comme valeur de cette boîte. [5]

L'extension réactive de ce mécanisme prend en compte certains *évènements* modifiant le contenu ou la structure du programme, et entraîne une mise à jour automatique des valeurs

4. Il est à noter cependant qu'un certain nombre de projets tels que FTM/Mubu (Schnell *et al.*, 2005, 2009) ou *bach* (Agostini et Ghisi, 2015) permettent aujourd'hui un contrôle des structures musicales, et un couplage plus riche des phases de composition et d'interaction dans les systèmes temps-réel.

5. La notion d'*état* attribué aux différentes boîtes (bloquée, lambda, etc.) permet de leur assigner un comportement spécifique lors des évaluations (par exemple pour gérer les appels multiples à une même boîte). Elle rentre ainsi dans cette description formelle et dans la détermination des dépendances fonctionnelles.

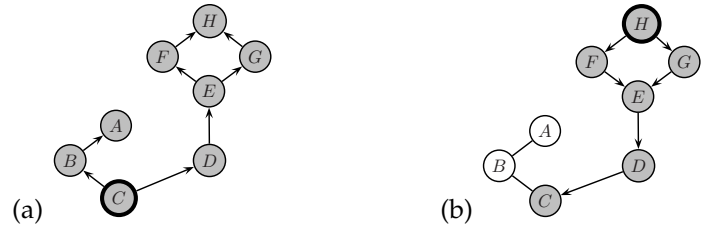


FIGURE 7.1: Exécution d'un graphe fonctionnel (a) *demand-driven* et (b) *event-driven*. *Demand-driven* est le mode d'évaluation par défaut d'OpenMusic : un nœud du graphe (ici, C) entraîne l'évaluation d'autres nœuds en amont afin d'obtenir ses valeurs d'entrée. Dans le mode *event-driven*, c'est le nœud H qui déclenche l'exécution du sous-graphe en aval.

de parties *aval* du graphe fonctionnel (voir figure 7.1-b). Elle permet ainsi de maintenir la cohérence des relations entre entrées et sorties des composants d'un programme visuel, sans requête d'évaluation explicite. Les boîtes ont pour cela une caractéristique mutable déterminant leur réactivité : seules les boîtes actives pourront être à l'origine d'un évènement ou participer aux réactions de mises à jour des valeurs dans le graphe. [6]

IMPLÉMENTATION. L'implémentation de ce modèle concerne principalement les conditions d'apparition des évènements (actions/éditions de l'utilisateur sur une donnée, message entrant via UDP ou MIDI, etc.), et la détermination des dépendances d'une boîte dans le graphe nécessitant une mise à jour si celle-ci engendre un évènement, ou participe à une réaction. Ces dépendances sont déterminées par une simple descente du graphe en profondeur (*depth-first*) : chaque descendant actif propage la réaction, jusqu'à ce qu'une boîte « terminale » (sans descendant actif) soit atteinte. Les boîtes terminales sont alors évaluées suivant le modèle existant (*demand-driven*) du langage.

L'ordonnancement réactif des calculs n'est donc pas centralisé dans un processus dédié, comme c'est le cas dans la plupart des systèmes « fonctionnels réactifs » (Elliott et Hudak, 1997), mais distribué au sein même des programmes dans leur mécanisme d'édition et d'évaluation. Le temps du système réactif est déterminé implicitement par une horloge logique abstraite correspondant aux états successifs du graphe, et donc déclenchée essentiellement par des évènements — modèle *event-driven* (Wan *et al.*, 2001). Dans chaque instant logique, le temps écoulé (même s'il est long) n'est pas pris en compte dans ce modèle, qui privilégie toujours les aspects compositionnels « hors-temps » sur les aspects performatifs.

APPLICATIONS. Ce nouveau mode d'exécution des programmes permet de lier les approches formelles et déclaratives *offline* des environnements de CAO aux approches dynamiques des systèmes réactifs, et de les étendre au contrôle temps-réel des traitements et interactions musicales (Bresson, 2014) ✎.

Du point de vue de l'utilisateur, un *patch* réactif offre également une meilleure vision et un retour immédiat sur les dépendances mutuelles des jeux de données édités dans les programmes visuels. [7] On en trouve les premières utilisations dans les travaux de compositeurs tels que Geof Holbrook [🕮 7] ou Alessandro Ratoci [🕮 20] (voir figure A.7 en annexe).

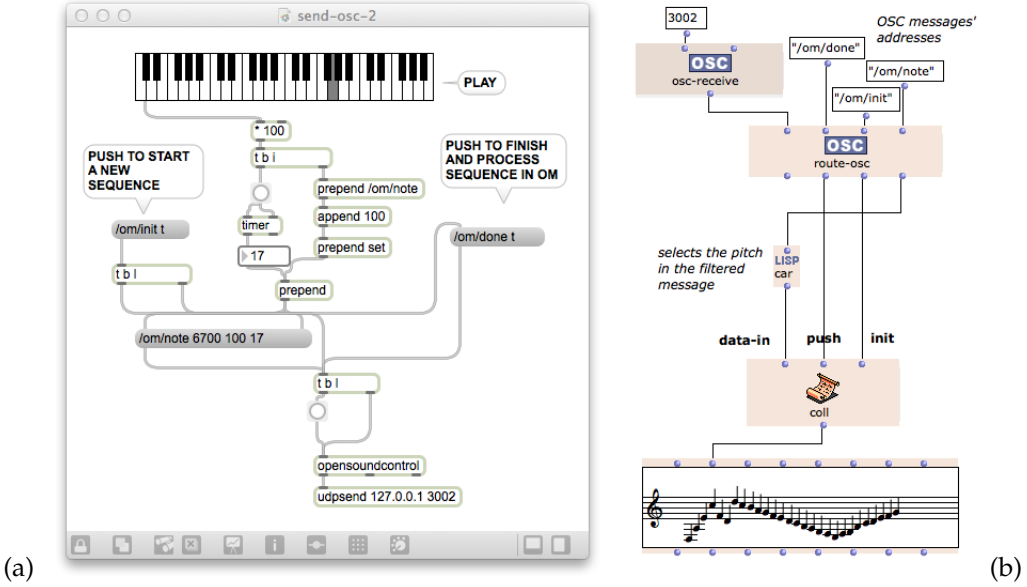
6. Le caractère mutable (optionnel) de l'état actif rend le modèle réactif *local* (applicable sur des sous-parties du graphe) et *conservatif* (les programmes existants et leur sémantique ne sont pas affectés).

7. Il peut être utile cependant de rappeler l'importance du caractère optionnel de ce fonctionnement. En effet dans de nombreux cas la complexité des processus de CAO ne permettent pas un déclenchement excessif de mises à jour du programme. On préfèrera ainsi, selon le cas, une mise à jour sur requête explicite de l'utilisateur.

7.2 COMMUNICATION EXTERNE ET INTER-APPLICATIONS

La fonctionnalité réactive de l'environnement de CAO facilite la mise en œuvre de communication et d'interactions avec d'autres logiciels ou avec des dispositifs d'entrée/sortie. Les processus compositionnels peuvent ainsi être construits et exécutés par l'utilisateur, mais également être contrôlés par des processus externes. La caractéristique *offline* et les possibilités de calcul des processus de CAO sont alors entremêlés à des contextes logiciels et/ou temporels de plus vaste échelle. Ces processus peuvent écouter, analyser, enregistrer voire apprendre à partir de données provenant de leur environnement, et transformer ou générer des données structurées en réponse à ces entrées.

RÉCEPTION ET TRAITEMENT DE MESSAGES. La communication inter-application est principalement supportée à travers des échanges de messages MIDI (Loy, 1985) ou OSC (Wright, 2005). [8] Les boîtes OpenMusic *osc-revceive* ou *midi-in*, lorsqu'elles sont actives (au sens exposé plus haut), exécutent un processus « serveur » à l'écoute de messages entrant sur un port donné. Sur réception d'un message, leur valeur est mise à jour et une réaction en chaîne est déclenchée. La figure 7.2 montre par exemple un *patch* OpenMusic réactif à la réception de messages OSC.



(a) (b)

FIGURE 7.2: Réception, aiguillage, collecte et traitement de données reçues via OSC. (a) Un programme Max envoie des messages de contrôle et des données via UDP suivant les actions de l'utilisateur sur un clavier graphique. (b) Le programme réactif OpenMusic collecte les données des messages et construit une séquence musicale.

La notion de message, reçu et traité dans un évènement, n'est pas temporisée. Notre système n'intègre pas la notion de flux, et chaque évènement est traité indépendamment. Pour construire une structure temporelle à partir d'un flux de messages, des outils spécifiques pour la collecte, la temporisation et la structuration de données sont donc nécessaires. Sur la figure 7.2 par exemple, la boîte *coll* collecte et mémorise les messages, et *route* permet de rediriger la propagation vers un traitement spécifique suivant le contenu du message. Ces

8. Les messages OSC sont généralement échangés via le protocole UDP.

outils réalisent une phase spécifique dans les mécanismes réactifs décrits plus haut, avec des opérations de filtrage ou d'enregistrement effectuées sur les données avant de continuer la propagation des réactions.

APPLICATION (1) : DONNÉES GESTUELLES / PAPIER INTERACTIF. Une première application importante du modèle réactif d'OpenMusic en communication avec un système externe fut mise en œuvre par Jérémie Garcia dans une collaboration avec le compositeur Philippe Leroux (Garcia *et al.*, 2014).

Lors de sa thèse réalisée en co-tutelle entre l'IRCAM et l'équipe *insitu* du LRI (Inria-CNRS-Université Paris Sud), Jérémie Garcia a développé des interfaces de « papier interactif » dédiées à la création musicale (Garcia, 2014). Le système de papier interactif permet de recevoir et traiter des informations écrites ou dessinées sur une feuille de papier à l'aide d'un stylo (Garcia *et al.*, 2012). [9] Dans la pièce *Quid Sit Musicus ?* de Philippe Leroux (2014) [10] cette technologie est utilisée pour transférer des gestes d'écriture, réalisés par le compositeur sur des copies d'anciens manuscrits enluminés, vers des programmes (processus compositionnels) OpenMusic (voir figure 7.3). [11] Une catégorisation des gestes permet d'aiguiller les messages OSC provenant du stylo vers des procédés de génération ou de transformation de données musicales : notes produites sur le plan temps/fréquence suivant la géométrie des tracés, transformation d'accords à partir des profils de vitesse, etc.

FIGURE 7.3: Interfaces tangibles (papier interactif) pour la composition assistée par ordinateur : *Quid Sit Musicus ?* de Philippe Leroux (2014). Photo H. Raguet © Inria.

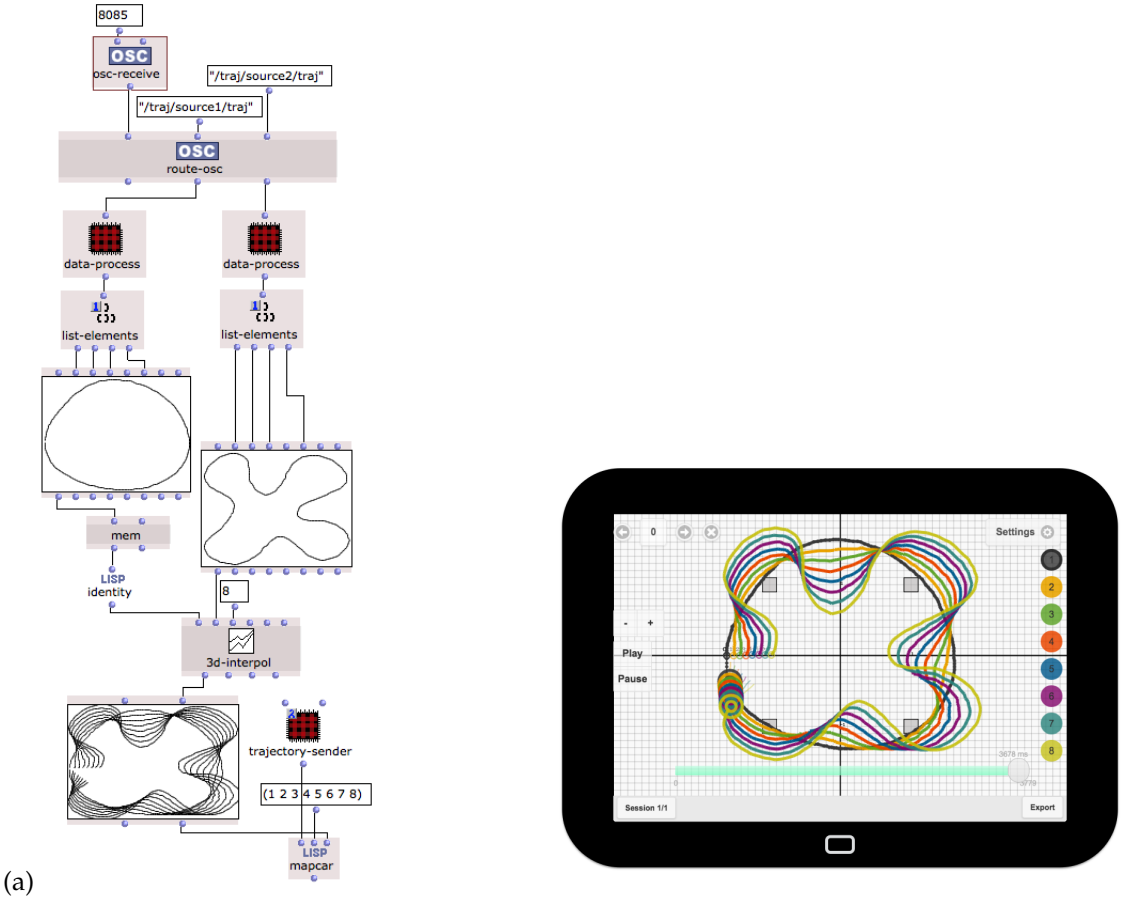
APPLICATION (2) : DISPOSITIFS MOBILES / SPATIALISATION. L'application TRAJECTOIRES décrite dans le chapitre précédent (section 6.4) a été également utilisée pour la démonstration de scénarios originaux coordonnant les processus algorithmiques et le rendu spatialisé temps-réel (Garcia *et al.*, 2015). Plusieurs cas de figure ont été implémentés, impliquant une communication réactive bi-directionnelle entre l'application mobile et l'environnement réactif de CAO : génération algorithmique de trajectoires dans OpenMusic, transférées à l'application mobile pour la constitution d'un « dictionnaire » de

9. Ce système est basé sur la technologie Anoto (`http://www.anoto.com`). Le stylo est équipé d'une caméra, et le papier est imprimé avec une trame microscopique permettant de localiser les gestes.

10. Création Festival Manifeste 2014 de l'Ircam, reprises au Festival MANCA (théâtre de Grasse) en 2014 et au Théâtre du Capitole de Toulouse en 2016.

11. Voir également la vidéo *Images d'une œuvre : Quid sit musicus ?*. → `http://medias.ircam.fr/x159d48`

formes spatiales, ou transfert de formes dessinées sur l'application mobile vers l'environnement de CAO (sur le modèle du précédent exemple). En combinant ces différentes modalités (réception de données, transformations algorithmiques, retour des données transformées), il est possible par exemple de traiter dynamiquement des gestes produits sur la tablette ou le dispositif mobile pour les transformer, voire les faire proliférer en un ensemble de gestes dérivés. L'exemple présenté sur la figure 7.4 suit ce principe : un mécanisme d'aiguillage (*route-osc*) permet d'isoler et de stocker deux trajectoires entrées sur la tablette, afin de calculer, sur réception de la seconde, une série de trajectoires interpolées renvoyées immédiatement vers l'application mobile, et pouvant ainsi être jouées dans l'instant sur le dispositif audio multi-canal contrôlé par l'application.



(a) (b)

FIGURE 7.4: (a) Processus réactif d'interpolation de trajectoires dans OpenMusic. Ce *patch* mémorise une première trajectoire reçue via OSC et calcule une série de trajectoires interpolées sur réception d'une seconde trajectoire. (b) Tracé, envoi, réception, visualisation et rendu des courbes dans l'application mobile TRAJECTOIRES.

O.OM : COMMUNICATION STRUCTURÉE AVEC OSC. La structuration des processus de communication et des interactions entre applications peut également être envisagée dans la conception même des protocoles de transfert. J'ai étudié cette idée lors de mon séjour au *Center for New Music and Audio Technologies* (CNMAT) de l'Université de Berkeley en 2016. [12]

12. Séjour dans le cadre d'une bourse de recherche Fulbright, février-août 2016.

La bibliothèque *o.* (*odot*) développée au CNMAT propose un ensemble de fonctionnalités et d'extensions du format de messages OSC (Freed *et al.*, 2011). La syntaxe des messages OSC est relativement simple ; elle consiste en une liste de valeurs typées de longueur variable, précédée d'une « adresse » du type URL (`a/b/...`) décrivant généralement une cible pour l'application des valeurs du message. Les messages s'agrègent dans une structure appelée *bundle*. Odot a pour objectif de tirer parti de ces éléments structurants et du potentiel de la programmation fonctionnelle pour améliorer et structurer le transfert de données interne et externe aux applications. [13] La bibliothèque embarque l'interprète d'un langage d'expressions permettant l'écriture et l'exécution de procédures opérant directement sur les structures de données (*bundles*) OSC. [14] Les expressions peuvent être écrites dans les messages de ces mêmes *bundles*, qui deviennent donc l'unique contexte pour l'agrégation de données, l'écriture, et l'exécution des programmes. [15]

Nous avons vu précédemment différentes utilisations d'OSC dans OpenMusic pour le transfert des données à des applications externes. J'ai intégré dans les récentes implémentations de l'environnement un support natif du format via une interface avec la bibliothèque *libo* du CNMAT (MacCallum *et al.*, 2015) ✎. L'objet OSC-BUNDLE est utilisé comme structure-support pour la communication, mais aussi comme élément temporisé d'une séquence musicale (voir section 7.3). Le support du langage *odot* embarqué comprend des outils pour la construction et le formattage des expressions à l'aide de primitives graphiques, et une connexion à l'interprète permettant d'évaluer ces expressions transformant les *bundles* OSC. Il est possible alors par exemple de construire des expressions *odot* dans OpenMusic, et d'envoyer ou exécuter ces expressions dynamiquement à différents points des échanges inter-applications. L'utilisation d'un langage dédié et de structures de données communes entre plusieurs environnements communicants permet ainsi de délocaliser arbitrairement la conception ou l'écriture des programmes, mais également leur exécution, dans le réseau d'applications (Bresson *et al.*, 2016) ✎.

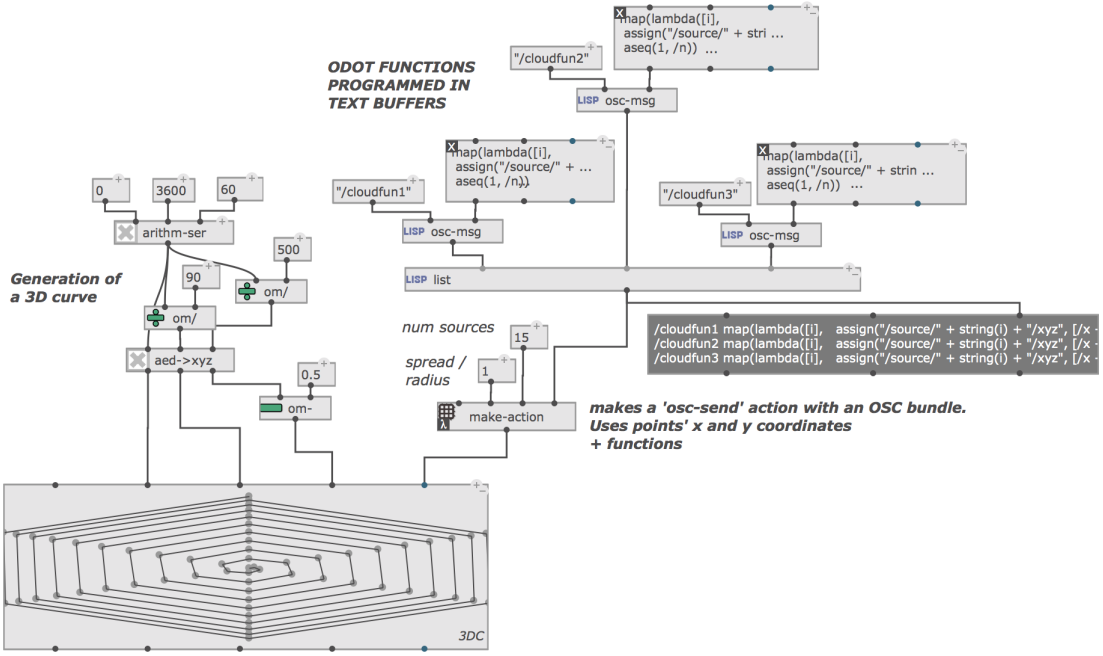
L'exemple de la figure 7.5 a été proposé dans l'article cité ci-dessus et concerne le contrôle temps-réel d'un processus de spatialisation incluant de multiples sources virtuelles (voir chapitre 6). Le contrôle d'un processus de spatialisation peut nécessiter l'encodage et la transmission simultanée d'un ensemble important de données ; or dans le cas où la distribution spatiale des sources sonores suit une règle connue à chaque instant t, une scène sonore complexe peut être décrite de manière fonctionnelle par un petit ensemble de messages (et de taille constante quelque soit le nombre de sources). Le *patch* de la figure 7.5-a implémente ce cas de figure. Il fait appel à des séquences de données spatiales (trajectoires), exécutant des actions programmées via l'ordonnanceur du système (nous développerons ce point dans la section 7.3). Des *bundles* OSC, créés et envoyés par OpenMusic pour chaque point d'une trajectoire, contiennent plusieurs fonctions générant la position d'un nombre variable de

13. La principale implémentation de la bibliothèque *odot* à l'heure actuelle est utilisée dans l'environnement Max pour le traitement structuré des données transitant en temps-réel dans les programmes *data-flow*.

14. La syntaxe et la sémantique des expressions sont relativement simples et sont inspirées des langages de *scripting* fonctionnels classiques. Elles incluent des opérateurs arithmétiques s'appliquant sur les valeurs scalaires ou vectorielles, des opérateurs d'assignation, des mécanismes d'abstraction et des fonctions d'ordre supérieur (*map*, *fold*, ...), ainsi que des opérations s'appliquant sur les messages contenus dans un *bundle* OSC telles que *assign*, *delete*, etc.

15. Lors de l'évaluation d'une expression sur un *bundle* (donné comme argument), les fonctions sont évaluées en utilisant ou créant les valeurs référencées par les adresses OSC dans ce *bundle*. L'expression *"/y = /x + 1"* signifie par exemple qu'un message dont l'adresse est */y* sera créé (s'il n'existe pas déjà) et prendra une valeur calculée à partir de la valeur courante d'un autre message du *bundle* dont l'adresse est */x*.

(a)



(b)

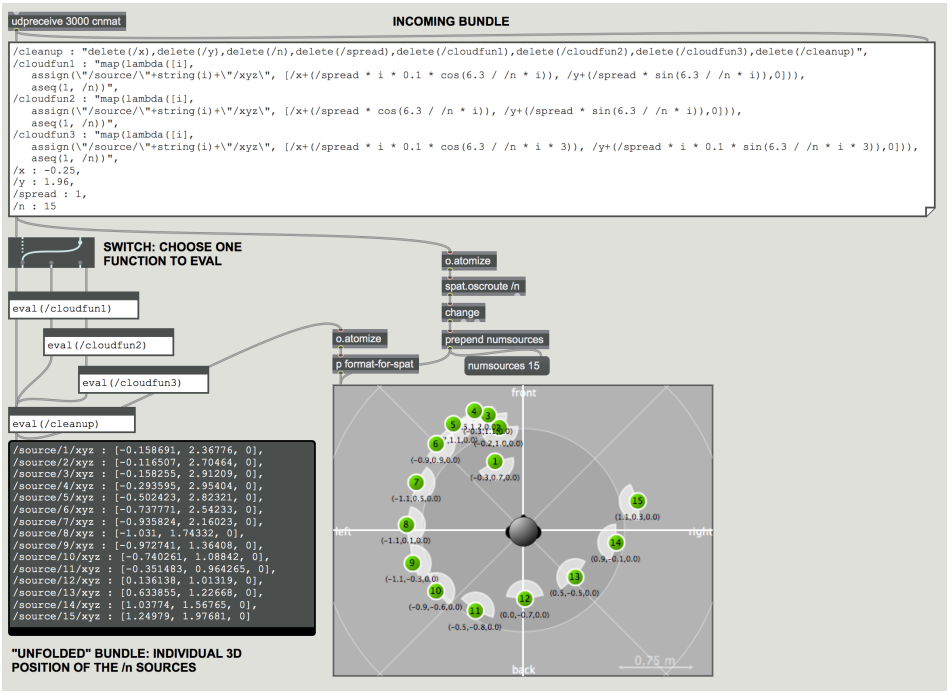


FIGURE 7.5: (a) Production d'expressions *odot* et de *bundles* OSC pour le rendu d'une courbe 3D dans OM#. La boîte *make-action* produit une lambda-expression contenant un appel *osc-send* avec un *bundle* OSC correspondant à chaque point de la trajectoire. Le *bundle* et l'action sont produits à la volée au moment de l'exécution. Sur la droite, un *bundle* est présenté à titre d'illustration ; on y voit les expressions correspondant à trois fonctions d'interprétation possibles des points de la trajectoire pour le calcul des positions des sources. (b) Réception des *bundles* dans Max, sélection d'une fonction (ici de type « spirale »), évaluation et production des données de contrôle en temps réel pour l'ensemble des sources sonores à l'instant t.

sources sonores relativement à celle de ce point, de sorte que l'application cible recevant les messages puisse choisir à la volée l'une de ces fonctions afin de déterminer un mode de distribution, et calculer les positions de l'ensemble des sources (figure 7.5-b).

7.3 STRUCTURES TEMPORELLES DYNAMIQUES

La thèse de Dimitri Bouche (2013-2016) [⋆] a porté essentiellement sur la thématique des structures temporelles et de l'ordonnancement musical dans le cadre mixte *offline*/interactif présenté dans les sections précédentes. Dans ce contexte en effet, les structures musicales ne sont plus simplement calculées *puis* jouées, mais sont susceptibles d'être calculées en phase de jeu, ou d'être jouées/exécutées dynamiquement à la suite de calculs.

Les temporalités du calcul et du jeu (ou « rendu ») des structures musicales, traditionnellement distinctes dans les outils de CAO, reflètent en termes plus techniques les notions de *planning* et d'*exécution* des processus. Leur convergence autorise le calcul et la planification de données au sein d'exécutions musicales, et vice-versa, l'exécution d'actions au sein des processus compositionnels responsables du planning. Elle ouvre ainsi le champ à des perspectives d'expressivité nouvelles pour la programmation de scénarios temporels interactifs : des boucles réactives mettant en jeu des processus compositionnels — où le temps réel laisse place à des calculs musicaux avancés — dont les résultats peuvent être réinjectés dans le flux de temps « réel ».

CAO et interactivité s'articulent ici autour du concept de *méta-composition* (Bouche *et al.*, 2017) ✎, entendu comme composition de structures musicales constituées de données et de processus, pouvant être modifiées et ré-ordonnancées dynamiquement par des interactions ou par leur propre restitution. Tout objet musical est associé à un *plan*, ou séquence d'actions datées, lues et traitées par un processus de « répartition ». La figure 7.6 en donne un exemple simple, avec une séquence de quelques notes sur une partition. Son plan est principalement constitué d'une suite d'envois de messages MIDI, qui seront éventuellement interprétés par un synthétiseur. Cet objet/méta-partition contient également une action spéciale (indiquée par la lettre A dans le plan) déclenchant un calcul modifiant le contenu de la fin de la séquence. Dans un tel cas de figure, la caractéristique dynamique du système d'exécution dépend des modalités de mise à jour du plan (suite à des calculs ou à des éditions sur les objets), et de leur prise en compte effective par le processus de répartition des actions.

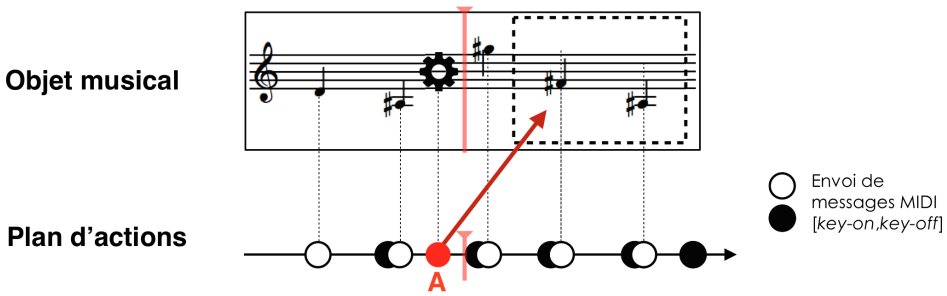


FIGURE 7.6: Exemple de méta-partition : une structure musicale constituée de données et de processus produisant ou modifiant ces données. Le rendu, matérialisé par la position du curseur sur la partition, correspond à l'exécution du plan (représenté en bas) par un processus de répartition des actions.

ARCHITECTURE ET STRATÉGIES D'ORDONNANCEMENT. Le formalisme proposé par Dimitri Bouche rapproche donc la notion d'exécution dynamique des structures musicales et celles de re-ordonnancement ou de re-planification (desJardins *et al.*, 1999), largement étudiées dans des domaines tels que la robotique, les jeux videos, etc. Plusieurs stratégies peuvent être adoptées pour ordonnancer (c'est à dire, créer un plan d'actions) et exécuter une structure musicale dans un système informatique (Graham *et al.*, 1979). Un plan total peut généralement être réalisé dans une première phase, puis exécuté (approche « statique »). Dans une approche plus dynamique, un ordonnanceur peut être sollicité de manière réactive ou périodique pour produire des plans partiels (c'est à dire déterminer uniquement la prochaine, ou les n prochaines actions à exécuter). Dans ce cas, l'ordonnancement est entremêlé à l'exécution du plan.

Les stratégies intermédiaires proposées dans cette thèse permettent d'adapter une approche statique (présumant d'une certaine complexité des structures musicales composées) à des systèmes dynamiques (Bouche et Bresson, 2015b,c) ✎. Grâce à la combinaison de différentes heuristiques et optimisations, le système d'ordonnancement développé intègre de manière transparente les tâches de calculs, les modifications des plans d'actions, et le déclenchement de processus réactifs au sein des exécutions (c'est-à-dire pendant la répartition des actions). Ces heuristiques comprennent par exemple la prise en compte des différents objets d'une partition comme producteurs de « sous-plans » indépendants, l'ordonnancement borné adaptatif, consistant à fragmenter les opérations d'ordonnancement par horizons temporels successifs de durée variable (Bouche et Bresson, 2015a) ✎, l'anticipation des actions susceptibles de déclencher des tâches de calculs étendues dans le temps, ou encore le ré-ordonnancement conditionnel, stratégie de décision pour la révision des plans à la suite de modification des objets en cours d'exécution.

Ce système est implémenté à l'aide de trois processus opérant en parallèle (voir figure 7.7) :

— Un **calculateur** en charge de produire des structures musicales en exécutant des *tâches* provenant de requêtes d'utilisateur, ou de l'exécution de ces mêmes structures (si les actions des plans engendrent des calculs).
— Un **ordonnanceur** en charge de fournir des plans d'actions à partir d'un ensemble d'objets musicaux à exécuter.
— Un **répartiteur** en charge de l'exécution des plans, effectuant des requêtes à l'ordonnanceur pour le calcul de plans partiels, et au calculateur pour l'exécution éventuelle de tâches émanant du plan d'actions.

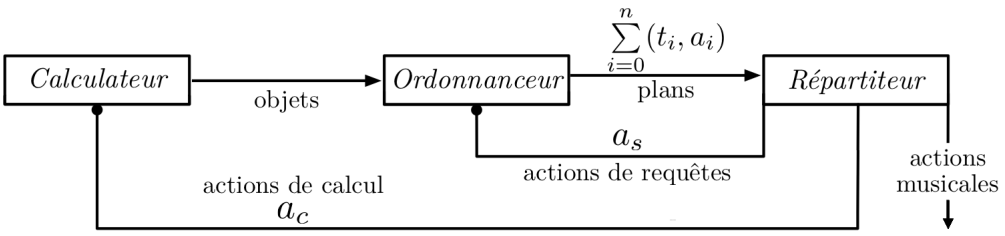


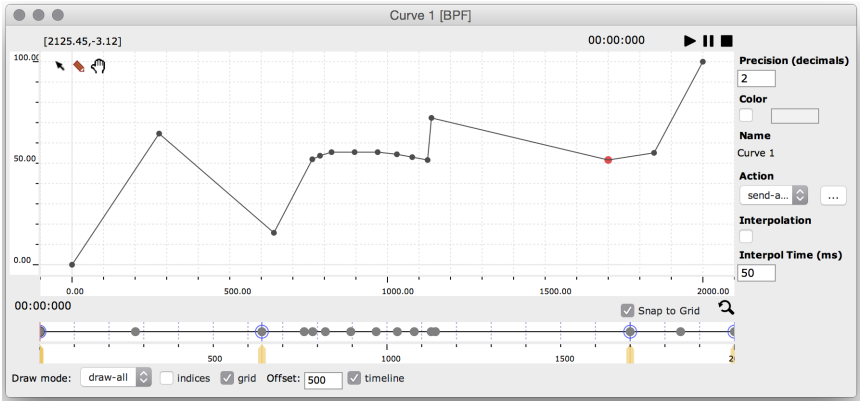
FIGURE 7.7: Architecture du système d'ordonnancement dynamique pour les méta-partitions.

IMPLÉMENTATION ET INTERFACE (1) : OBJETS MUSICAUX. Suivant le système présenté plus haut, l'exécution d'une structure musicale sur un intervalle de temps donné consiste en la mise en œuvre, en temps voulu, des actions d'un plan déterminées

sur cet intervalle. Une interface de programmation (API) simple permet le développement et l'intégration rapide d'objets et de structures musicales temporisées dans ce système. Parallèlement, dans le cadre du travail post-doctoral de Jérémie Garcia sur les interfaces utilisateurs le contrôle de la spatialisation [⋆] (voir chapitre 6, figure 6.10), un composant d'interface graphique (appelé *timeline*) a été développé pour représenter et manipuler la structure temporelle correspondant au plan d'exécution d'un objet (Garcia *et al.*, 2017) (dans le cas de mouvements de source sonores, il s'agissait essentiellement des dates et des marqueurs de synchronisation assignés aux points constituant les trajectoires).

Tous les objets « exécutables » de l'environnement de CAO (séquences « *piano-roll* » MIDI, trajectoires de contrôle, automations, etc.) implémentent cette interface, et leur structure temporelle peut ainsi être manipulée à l'aide de ce même composant graphique (voir figure 7.8). Nous l'avons également utilisée dans (Bresson *et al.*, 2016) pour construire des flux de *bundles* OSC à l'intérieur d'un conteneur générique de données (*data-stream*, voir figure 7.9), permettant de représenter, manipuler et exécuter des séquences d'objets de types arbitraires (la seule contrainte étant que ces objets soient datés et/ou se reflètent sous forme d'actions datées dans un plan d'exécution — dans le cas de *bundles* OSC, ces actions sont de simples envois du *bundle* via UDP).

(a)



(b)

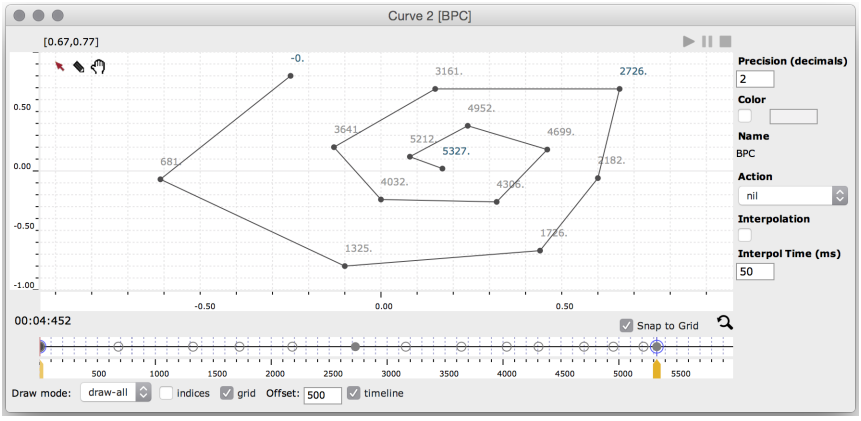


FIGURE 7.8: Contrôleurs intégrés dans l'API d'ordonnancement et de représentation temporelle des structures musicales. (a) Automation (temps, valeur de contrôle). (b) Trajectoire bidimentionnelle (x, y) — le temps est indiqué textuellement sur chaque point. Chaque éditeur est complété (en bas de la fenêtre) par un composant graphique *timeline* pour l'édition de la structure temporelle. Dans ce composant d'interface (et dans les structures sous-jacentes), certains éléments spécifiques (représentés par des points cerclés ⊙) permettent d'étirer ou de compresser les objets, ou encore de se synchroniser avec les composants d'autres *timelines*.

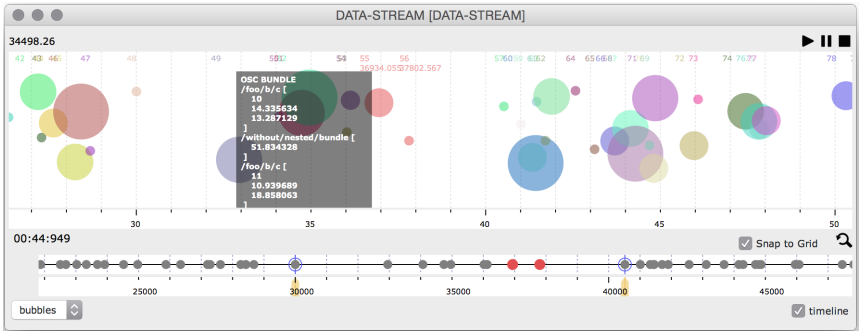


FIGURE 7.9: DATA-STREAM : un conteneur générique pour l'édition et le rendu de données temporisées. Ici, utilisé pour visualiser et « exécuter » une séquence de *bundles* OSC. Le composant *timeline* en bas de la fenêtre permet de visualiser et modifier la séquence temporelle.

IMPLÉMENTATION ET INTERFACE (2) : MÉTA-PARTITIONS. L'architecture présentée précédemment opère au cœur de l'environnement OM# (en cours de développement à l'heure d'écriture de ce mémoire) comme moteur de calcul et d'exécution des structures musicales. Elle a donné lieu à l'implémentation d'une nouvelle génération d'éditeur musical sur le modèle de la *maquette* (voir chapitre 3, section 3.1) permettant la création de processus compositionnels dynamiques. Les objets placés dans la maquette/méta-partition peuvent être des structures de données statiques, des programmes visuels (considérés comme « tâches » à exécuter), ou encore des objets statiques déclenchant indirectement des tâches de calcul par le biais de leurs actions (voir figure 7.10).

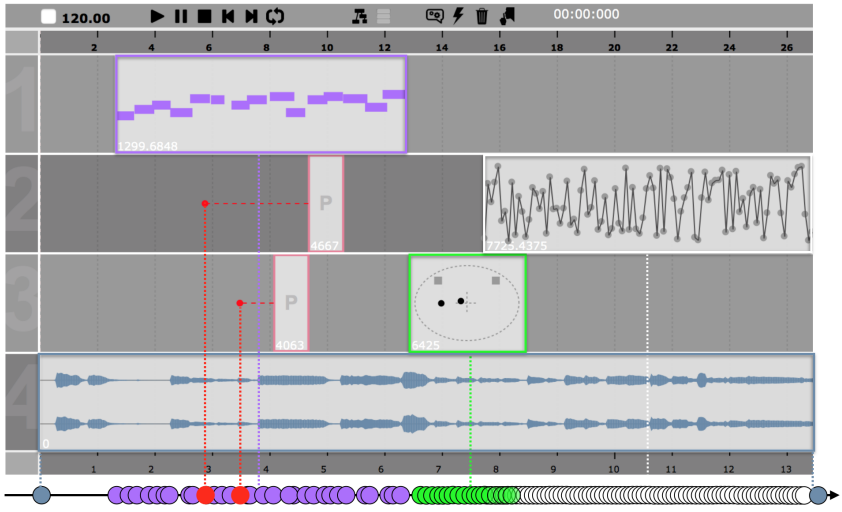


FIGURE 7.10: Méta-partition (*maquette*) : environnement d'exécution pour les objets et processus musicaux. Les « processus » sont indiqués par la lettre P et renferment des programmes visuels produisant des données ou modifiant le futur de la séquence. Le segment horizontal qui les précède indique un pré-délai d'anticipation fixé pour leur déclenchement. En bas : représentation schématique du plan d'actions correspondant.

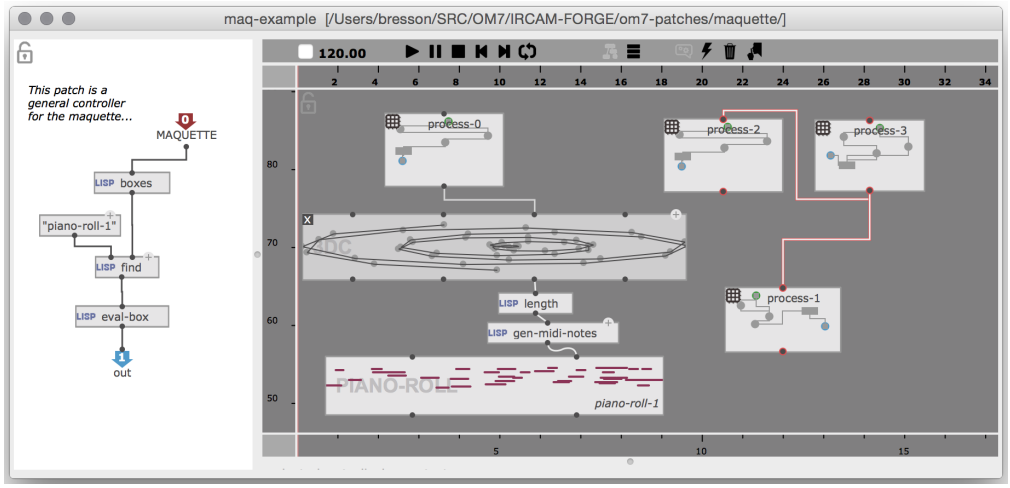
L'interface principale se présente donc comme un séquenceur, pouvant cette fois être visualisé alternativement sous forme de programme visuel étendu (sur le modèle de la *maquette*) ou sous forme de pistes parallèles disposant les objets de manière structurée pour l'exécution (voir figure 7.11).

Les programmes disposés sur cette interface peuvent produire de nouvelles structures intégrables dynamiquement par l'ordonnanceur dans l'exécution musicale en cours. Ils sont exécuté, selon le choix de l'utilisateur/programmeur de la méta-partition :

— Avant le démarrage de la lecture ;
— En cours, et comme partie intégrante de la lecture (c'est-à-dire au moment déterminé par leur position sur l'axe temporel) ;
— En cours de lecture avec anticipation, comme l'indiquent par exemple les lignes pointillées sur la gauche du processus se trouvant sur la piste 4 de la figure 7.11-b, ou sur ceux de la figure 7.10.

Sur le modèle du conteneur de *sheet* (chapitre 3, figure 3.3), un *patch*/programme « contrôle » contient également des actions exécutées sur le contenu de la méta-partition, de manière anticipée/statique ou dynamique (en cours d'exécution), avec une vue globale sur sa structure (cf. figure 7.11-a).

(a)



(b)

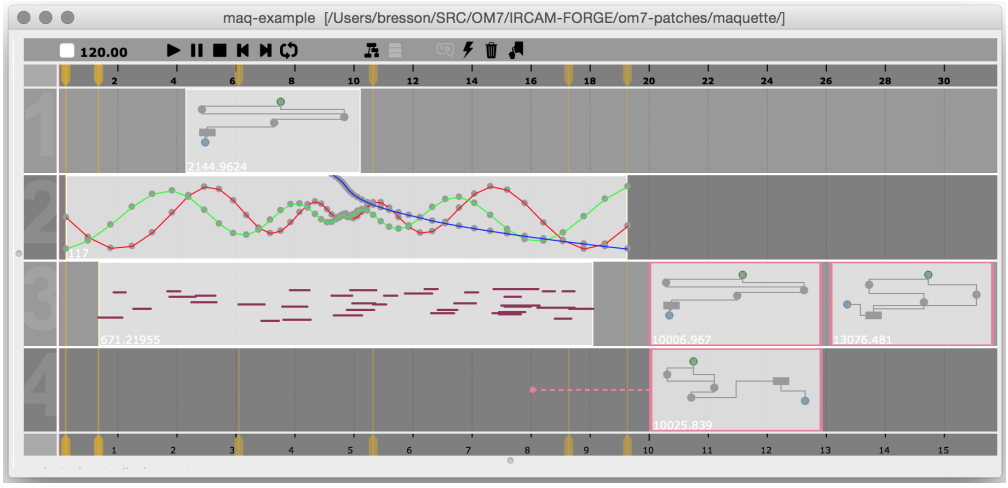


FIGURE 7.11: Interface OM# pour une méta-partition (*maquette*). (a) Vue « classique » incluant les connections et relations fonctionnelles entre objets et processus. (b) Vue structurée en pistes (*tracks*). Les points d'ancrage temporels (marqueurs verticaux) sur les objets visibles en (b) sont issus de leur structure temporelle interne et utilisés pour déplacer, étirer/compresser ou synchroniser les objets.

Une application de cette méta-partition a été proposée par exemple en collaboration avec Jérôme Nika dans le cadre de sa thèse portant sur l'improvisation guidée par scénario (Nika, 2016). Sur la figure 7.12 la méta-partition exécutée contient initialement un accompagnement statique et deux instances d'un programme (« agent improvisateur »). Chacune des instances de ce programme se réfère à une grille harmonique prédéfinie, à une mémoire musicale du passé (par exemple basée sur un corpus), et aux données produites par son co-improvisateur pour (1) produire à son tour une séquence de jeu (exécution du processus), et (2) initialiser et planifier à la suite une prochaine exécution du processus co-improvisateur (construction du processus et requête au système d'ordonnancement) (Nika *et al.*, 2015) ✎.

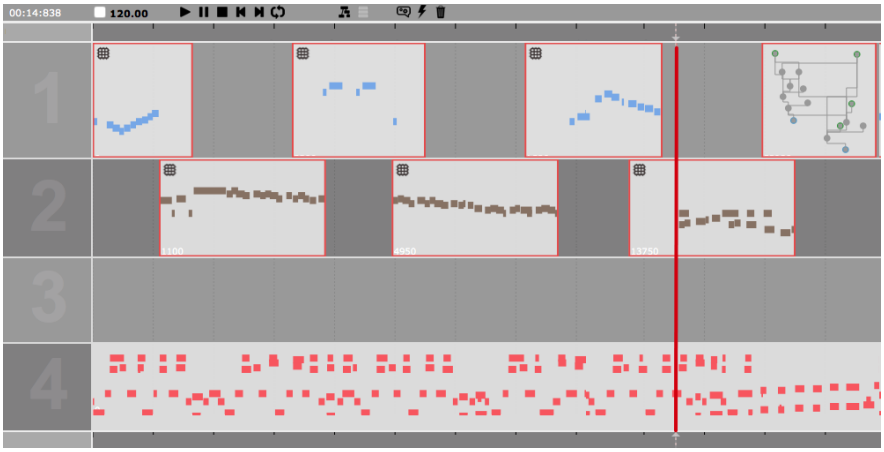


FIGURE 7.12: Scénario réactif d'improvisation dans un séquenceur OM#. Les pistes 1 et 2 contiennent des instances d'un programme « improvisateur ». Noter la dernière instance sur la piste 1, planifiée non encore calculée à l'instant de la prise de vue. Piste du bas : accompagnement statique (pré-calculé).

7.4 RETOUR VERS LES PROBLÉMATIQUES SIGNAL

Les environnements musicaux mettent généralement en œuvre une couche logicielle et un processus dédiée au signal audio, connectés aux dispositifs de sortie du système. L'architecture présentée dans la section précédente supporte de manière transparente aussi bien les calculs sur les données symboliques, que les traitements et la synthèse de signaux. Dès lors, la production de sons (effectuée par le processus calculateur) peut être opérée soit « hors-temps », soit en temps réel (si elle est suffisamment rapide), soit encore par anticipation, via le calcul asynchrone de séquences d'échantillons (ou *buffers* audio) planifiés dans un futur plus ou moins rapproché par le système d'ordonnancement, et transférés sur un *buffer* de sortie parcouru en continu par un processus de lecture audio.

Le cas du SPAT-SCENE, présenté dans la section 6.4, est un premier exemple de cette application du système d'exécution et d'ordonnancement dans le domaine du signal : SPAT-SCENE peut en effet être « synthétisé » (*offline*) en un fichier audio multi-canal, mais également exécuté et reproduit dynamiquement sous forme sonore via des requêtes périodiques de calcul d'échantillons sonores (spatialisés sur n canaux) émanant du processus répartiteur lors de la lecture.

Un autre exemple est présenté sur la figure 7.13, où un conteneur de type *data-stream* (cf. section 7.3/figure 7.9) est cette fois peuplé d'objets appelés *IAE-request*, connectés à un moteur de synthèse granulaire.[16] Chacun de ces objets effectue (par l'intermédiaire d'une *action* associée) une requête de recherche dans une base de données pour la production et le rendu dynamique d'un fragment sonore suivant des valeurs de descripteurs données.

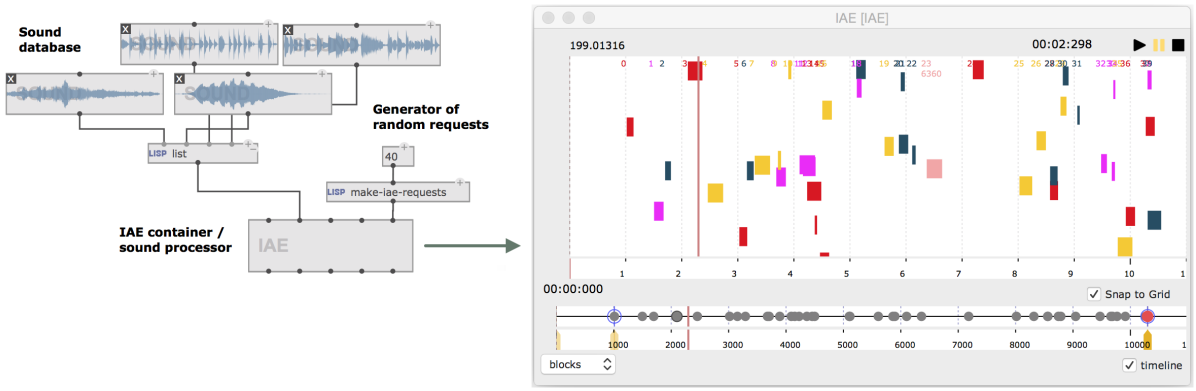


FIGURE 7.13: OM-IAE : bibliothèque OM# pour le contrôle interactif et musical de la bibliothèque IAE : calcul de et requête de descripteurs audio et synthèse granulaire.

Les principaux logiciels musicaux proposés actuellement dans le commerce traitent essentiellement la question du contrôle dans le temps des traitements audionumériques par l'intermédiaire de pistes d'automations. Des exemples de couplage élémentaire de structures ordonnées dans le temps avec la programmation de processus et de traitements audio temps-réel se développent, par exemple avec l'intégration du système temps-réel Max dans la station audio-numérique *Live* de Ableton, aujourd'hui largement répandue dans les productions musicales contemporaines. Dans ce contexte, l'approche proposée avec cette extension réactive de la CAO associée aux capacités de traitement et de restitution sonore nous permet de mettre en avant l'idée de programmation de structures musicales et sonores, à la fois formalisées dans le temps long et englobées dans un même modèle de calcul et de représentation.

16. Bibliothèque IAE, développée par l'équipe Interactions Son Musique Mouvement (ISMM) de l'IRCAM.

PUBLICATIONS (CHAPITRE 7)

Bouche, D. et Bresson, J. (2015a). Adaptive lookahead planning for performing music composition. *In International Conference on Automated Planning and Scheduling (ICAPS'15)*, Jerusalem, Israel. Poster.

Bouche, D. et Bresson, J. (2015b). Articulation dynamique de structures temporelles pour l'informatique musicale. *In* Merz, S. et Pétin, J.-F., éditeurs : *Modélisation des Systèmes Réactifs (MSR'15)*, Nancy, France.

Bouche, D. et Bresson, J. (2015c). Planning and Scheduling Actions in a Computer-Aided Music Composition System. *In Proceedings of the ICAPS'15 Scheduling and Planning Applications woRKshop (SPARK)*, Jerusalem, Israel.

Bouche, D., Nika, J., Chechile, A. et Bresson, J. (2017). Computer-aided Composition of Musical Processes. *Journal of New Music Research*, 46(1).

Bresson, J. (2014). Reactive Visual Programs for Computer-Aided Music Composition. *In IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Melbourne, Australia.

Bresson, J., Bouche, D., Garcia, J., Carpentier, T., Jacquemard, F., Maccallum, J. et Schwarz, D. (2015). Projet EFFICACE : Développements et perspectives en composition assistée par ordinateur. *In Actes des Journées d'Informatique Musicale (JIM'15)*, Montréal, Canada.

Bresson, J. et Chadabe, J., éditeurs (2017). *Special Issue on Interactive Composition*. Journal of New Music Research, 46(1). Taylor & Francis (Routledge).

Bresson, J. et Giavitto, J.-L. (2014). A Reactive Extension of the OpenMusic Visual Programming Language. *Journal of Visual Languages and Computing*, 25(4).

Bresson, J., MacCallum, J. et Freed, A. (2016). o.OM : Structured-Functional Communication between Computer Music Systems using OSC and Odot. *In ACM SIGPLAN Workshop on Functional Art, Music, Modeling & Design (FARM'16) – Proceedings of the International Conference on Functional Programmaing (ICFP'16)*, Nara, Japan.

Garcia, J., Bouche, D. et Bresson, J. (2017). Timed Sequences : A Framework for Computer-Aided Composition with Temporal Structures. *In Proceedings of the International Conference on Technologies for Music Notation and Representation (TENOR'17)*, A Coruña, Spain.

Garcia, J., Bresson, J., Schumacher, M., Carpentier, T. et Favory, X. (2015). Tools and Applications for Interactive-Algorithmic Control of Sound Spatialization in OpenMusic. *In inSonic2015, Aesthetics of Spatial Audio in Sound, Music and Sound Art*, Karlsruhe, Germany.

Garcia, J., Leroux, P. et Bresson, J. (2014). pOM : Linking Pen Gestures to Computer-Aided Composition Processes. *In Joint International Computer Music / Sound and Music Computing Conferences (ICMC-SMC'14)*, Athenes, Greece.

MacCallum, J., Gottfried, R., Rostovtsev, I., Bresson, J. et Freed, A. (2015). Dynamic Message-Oriented Middleware with Open Sound Control and Odot. *In Proceedings of the International Computer Music Conference (ICMC'15)*, Denton, USA.

Nika, J., Bouche, D., Bresson, J., Chemillier, M. et Assayag, G. (2015). Guided improvisation as dynamic calls to an offline model. *In Proceeddings of the Sound and Music Computing conference (SMC'15)*, Maynooth, Ireland.

8

CONCLUSION ET PERSPECTIVES

J'ai parcouru dans ce mémoire différents volets de mon activité de recherche, en essayant d'esquisser à travers ceux-ci une évolution de la composition musicale assistée par ordinateur et de son champ d'action. Ce domaine de recherche met en avant l'association des formalismes musicaux et calculatoires, et propose des outils de programmation et une approche interactive pour la modélisation des processus compositionnels. Considérant le compositeur comme un *end-user programmer*, je me suis intéressé à la fois aux outils de programmation (notamment visuelle), à l'ingénierie et à la conception logicielle, aux interfaces homme-machine, aux pratiques musicales et aux diverses représentations de la musique, vue comme une structuration du son dans le temps ; l'objectif étant d'optimiser l'exploitation des capacités expressives et calculatoires des outils informatiques utilisés à des fins créatives. Mes travaux se sont démarqués en particulier par une ouverture à divers champs disciplinaires tels que le traitement du signal, l'acoustique et le son spatialisé, la musicologie, ou encore la programmation réactive, dans une démarche visant à unifier des concepts clivant traditionnellement la recherche et les développements en informatique musicale.

Les résultats de ces recherches ont été diffusés à travers les différentes publications, collaborations, et travaux universitaires supervisés au cours des douze années passées. Dans ce mémoire, j'ai essayé autant que possible d'en relever les applications concrètes à travers l'expérience directe des compositeurs. Mon activité dans le domaine la CAO s'est aussi manifestée à travers diverses initiatives d'animation communautaire autour de séminaires et de groupes de travail, de conférences et de projets éditoriaux (Bresson et Fober, 2014; Bresson et Chadabe, 2017; Bresson *et al.*, 2016) ✎.

Elle s'ouvre désormais sur plusieurs axes complémentaires.

SIGNAL/SYMBOLE

Mes premiers travaux en informatique musicale préfiguraient déjà une démarche d'intégration des domaines *signal* et *symbolique* dans le traitement et la représentation de l'information musicale. Cette démarche est restée ensuite relativement constante dans mon activité, que ce soit dans le domaine de l'analyse/synthèse sonore, de la spatialisation, ou d'autres descriptions physiques du réel. Mes projets les plus récents, présentés notamment dans le chapitre 7, ont permis de revenir sur cette question sous un angle nouveau, à travers le rapprochement entre systèmes temps-réel et temps-différé.

En effet la distinction signal/symbole en informatique musicale a eu tendance à s'ancrer historiquement dans ce clivage : la programmation et le calcul appliqués aux signaux sonores ayant lieu dans la grande majorité des cas dans des environnements temps-réel

(en interaction continue avec des signaux ou des flux entrants d'évènements), alors que les outils symboliques de composition ont gardé le monopole des environnements de calcul « temps-différé ». Cette perspective sur le problème ne concerne donc plus vraiment le type d'information traitée par tel ou tel système, mais la façon dont cette information est calculée et déployée dans le temps. [1]

J'ai à quelques reprises mentionné des outils existant dans le monde des systèmes temps-réel se rapprochant du domaine de la composition. En face, et à l'image par exemple de Jean-Claude Risset, de nombreux compositeurs se sont également positionnés pour une approche compositionnelle « temps-différée » des processus de synthèse sonore ; [2] la plupart des outils présentés dans les chapitres 4 et 5 constituant une implémentation directe de cette proposition dans notre environnement de CAO.

Le système d'ordonnancement réactif intégré dans OM# et présenté dans les sections 7.3 et 7.4 inclut une de gestion dynamique des calculs et du signal sonore, qui nous a permis de proposer de nouvelles modalités de création de structures temporelles intégrant des traitements audio avancés (spatialisation, synthèse granulaire, etc.). Nous terminons donc provisoirement ce parcours en envisageant une nouvelle perspective de recherche, revisitant les processus de traitement et de synthèse sonore dans un contexte dynamique et interactif développé autour des processus de composition assistée par ordinateur.

NOTATION

La notation a été le principal support de la formalisation et de la transmission de la musique occidentale depuis des siècles. De la composition à l'interprétation, en passant par l'analyse, elle est à la base de tout le spectre des pratiques musicales, et le principal moyen pour les compositeurs de *penser* la musique. Le contexte musical actuel (esthétique, technologique...) questionne cependant l'adéquation de la notation traditionnelle avec la pensée et les nouvelles formes d'expression contemporaines.

Cette problématique n'est pas uniquement liée à la création, mais s'étend également à d'autres domaines de recherche liés à la musique : musicologie, ethnomusicologie, archivage et préservation... qui appellent également des solutions nouvelles pour la notation et la représentation des structures musicales. Ainsi un champ de recherche spécialisé a émergé récemment, auquel j'ai contribué par la création d'un groupe de travail à l'AFIM (Association Francophone d'Informatique Musicale) : *Les nouveaux espaces de la notation musicale* (Fober *et al.*, 2015) ✎, puis le lancement de la première conférence internationale sur les Technologies pour la Notation et la Représentation de la musique (TENOR) qui a eu lieu à Paris en 2015 (Battier *et al.*, 2015) ✎. [3]

Les relations entre les espaces formel et graphique des structures musicales sont en effet fondamentales dans les démarches compositionnelles, et constituent un aspect encore traité de façon relativement superficielle par les systèmes informatiques actuels. Les outils les plus avancés portent principalement sur la gravure musicale au sens traditionnel, et

1. *"Whilst the one-directional time flow of the signal processing domain advances in small units which we want to minimize for qualitative reasons ; time in the symbolic domain is a main subject of its processing"* (Eckel et Gonzalez-Arroyo, 1994).

2. *"C'est le temps différé qui permet d'étendre à la microstructure sonore des possibilités de notation et de travail proprement compositionnel"* (Risset, 1993).

3. TENOR : *International conference on Technologies for music Notation and Representation* a été reconduite en 2016 à l'Université Anglia Ruskin de Cambridge (Royaume Uni), en 2017 à l'Université de La Corogne (Espagne), et aura lieu en 2018 à Montréal (Canada). → `http://tenor-conference.org`]

peu de solutions existent prenant en compte les nouvelles dimensions (sonores, spatiales, conceptuelles) présentes dans les processus compositionnels contemporains.

Que représente la notion même de notation dans les contextes technologiques et esthétiques actuels ? Quel pourrait-être aujourd'hui un support comparable à la notation traditionnelle pour la pensée musicale contemporaine ? Pour conserver leur pertinence en support à la création, les systèmes de composition assistée par ordinateur devront aborder ces questions et proposer des outils de notation adaptés aux nouvelles formes et aux nouvelles modalités d'exécution des processus musicaux.

Cette thématique occupera donc vraisemblablement une place importante dans nos projets à venir : sur le versant théorique, à travers la recherche de modes de représentations à la fois symboliques, quantifiables et exécutables pour la notation ; et sur un plan plus pratique, à travers la conception de systèmes « supervisés » où les interactions de l'utilisateur peuvent être déterminantes dans l'exécution des processus. Notre intérêt se tourne donc désormais vers des outils intégrant à la fois les modes traditionnels de notation, des représentations graphiques personnalisables, et les aspects calculatoires permis par les systèmes d'aide informatique à la composition. Ceux-ci devront permettre aux utilisateurs de traiter de multiples dimensions musicales dans un contexte cohérent et expressif, susceptible de supporter des démarches novatrices dans les domaines de la composition comme de l'analyse musicale.

Dans la section 3.1, nous avons déjà proposé des pistes en direction de représentations musicales hétérogènes intégrant à la fois la notation traditionnelle et le traitement ou la génération algorithmique des données dans une vision nouvelle du support informatique pour la partition. Ici encore, les nouveaux paradigmes de calcul et de représentation développés depuis sont susceptibles d'apporter une dimension interactive et dynamique, et une expressivité renouvelée à ce type d'outil (voir par exemple les représentations programmables proposées dans les éditeurs de la section 7.3, comme celui visible sur les figures 7.9 et 7.13). Différentes directions concrètes peuvent dès lors être envisagées pour aller plus loin : intégration des aspects liés à la synthèse, aux traitements et aux descriptions du signal dans l'espace graphique de la notation ; interfaces pour la définition et la personnalisation des représentations graphiques ; interaction « spatiale » (2D, 3D) avec ces représentations ; intégration d'outils de programmation et de calcul liés aux structures de la notation graphique.

En liant les aspects graphiques, interactifs et calculatoires des représentations musicales dans le contexte de la composition assistée par ordinateur, se dessine ainsi une nouvelle approche de la notation, en tant que langage prescriptif et descriptif de la musique, mais également *exécutable* et donc, d'une certaine manière, se rapprochant également de la définition d'un langage informatique.

POUR FINIR... ET POUR CONTINUER

Afin de favoriser la progression d'une discipline et d'une communauté artistique et scientifique associée à la CAO, il est important d'en questionner et renouveler régulièrement les paradigmes et les outils. Il semble que cette discipline ait atteint un certain stade de maturité et de popularité qui, s'il peut constituer un frein au changement, doit aussi nous servir de base pour une réflexion sur ses évolutions à venir.

Les derniers développements de nos recherches esquissent ainsi le projet d'une nouvelle génération d'environnement de CAO. Nous avons, dans les précédents chapitres ou plus haut dans celui-ci, ébauché certaines des caractéristiques visées de cet environnement. Parmi celles-ci, nous avons relevé notamment la pertinence d'une ouverture sur différents domaines et

compétences technologiques (traitement du signal, audio spatialisé, analyse, visualisation...), ou encore la mise en avant des interactions de l'utilisateur dans la spécification et dans l'exécution des calculs.

Nos récents travaux nous poussent aussi dans le sens d'une plus grande modularité, d'une compartimentation des architectures logicielles, et renforcent notre volonté politique de développement sous forme de logiciel libre, afin de favoriser l'extensibilité, la diffusion, mais aussi l'implication de contributeurs dans les évolutions du projet. De même, le développement de stratégies de communication robustes et expressives pour le partage d'information entre environnements multimédia spécialisés permettra une meilleure distribution des tâches, et la constitution d'environnements informatiques polyvalents et efficaces en support aux activités de création contemporaines.

Enfin, il restera primordial d'impliquer les compositeurs dans les différents stades de conception, de test et de validation de ces outils : la recherche en composition assistée par ordinateur ne saurait être pertinente sans une compréhension profonde des évolutions des méthodes et approches compositionnelles de ses principaux utilisateurs.

PUBLICATIONS (CHAPITRE 8)

Battier, M., Bresson, J., Couprie, P., Davy-Rigaux, C., Fober, D., Geslin, Y., Genevois, H., Picard, F. et Tacaille, A., éditeurs (2015). *Proceedings of the First International Conference on Technologies for Music Notation and Representation (TENOR'15)*, Paris, France.

Bresson, J., Agon, C. et Assayag, G., éditeurs (2006/2008/2016). *The OM Composer's Book (3 volumes)*. Editions Delatour / Ircam-Centre Pompidou.

Bresson, J. et Chadabe, J., éditeurs (2017). *Special Issue on Interactive Composition*. Journal of New Music Research, 46(1). Taylor & Francis (Routledge).

Bresson, J. et Fober, D., éditeurs (2014). *Informatique Musicale*, volume 33 de *Technique et Science Informatiques (TSI)*. Lavoisier.

Fober, D., Bresson, J., Couprie, P. et Geslin, Y. (2015). Les nouveaux espaces de la notation musicale. *In Actes des Journées d'Informatique Musicale (JIM'15)*, Montréal, Canada.

[⋆] SUPERVISION DE TRAVAUX UNIVERSITAIRES

Thèses de doctorat

MARLON SCHUMACHER :
A Framework for Computer-Aided Composition of Space, Gesture, and Sound. Theory, Tools and Applications.
McGill University, Montréal, Canada
Direction : Sean Ferguson, Marcelo Wanderley, Jean Bresson
Soutenue le 14/06/2016.

CHARLES DE PAIVA SANTANA :
The Musical Score as an Instance : Essays in Computer-Assisted Analysis.
Université Pierre et Marie Curie, Paris / **Université de Campinas**, Brésil
Programme Doctoral International *Modélisation des Systèmes Complexes* (IRD/UMMISCO)
Direction : Jean Bresson, Moreno Andreatta , Jônatas Manzolli
Soutenue le 6/12/2106.

DIMITRI BOUCHE :
Processus compositionnels interactifs : Une architecture pour la programmation et l'exécution des structures musicales.
Université Pierre et Marie Curie, Paris
Direction : Jean Bresson, Gérard Assayag
Soutenue le 12/12/2016.

Masters, Licences, diplômes d'ingénieur

2016 Geoffroy Zoetardt
Interface and application of constraint programming for musical composition
Master's degree – Université Catholique de Louvain, Belgique
(co-encadré par Peter Van Roy)

2015 Xavier Favory
Applications mobiles pour le contrôle de la spatialisation sonore
Master 2 ATIAM – IRCAM/Université Pierre et Marie Curie
(co-encadré par Jérémie Garcia)

2015 Adrien Ycart
Quantification rythmique dans OpenMusic
Master 2 ATIAM – IRCAM/Université Pierre et Marie Curie
(co-encadré par Florent Jacquemard)

2014 Diego Diverio
Représentations spatiales pour la composition assistée par ordinateur
Master 2 AST (Arts, Science et Technologie) – PHELMA / INP Grenoble / ENSIMAG
(co-encadré par Jean-Louis Giavitto)

2013 Dimitri Bouche
Architecture audio dans OpenMusic
Diplôme d'ingénieur ENSEA, Cergy

2013 Pierre Donat-Bouillud
Transcription rythmique dans OpenMusic
Licence 3 ENS Cachan Bretagne
(co-encadré par Florent Jacquemard)

2013 Adrien Maire
Quantification musicale avec apprentissage sur des exemples
Master 1 ENS Cachan
(co-encadré par Florent Jacquemard)

2012 Raphaël Foulon
Procédures de contrôle continu pour la synthèse par fonctions d'ondes formantiques
Master 2 ATIAM – IRCAM/Université Pierre et Marie Curie

2009 Marlon Schumacher
Sound spatialization in a computer-aided composition environment
CIRMMT student internship, McGill University

2006 Ephrem Boudonnet
Portage de l'environnement OpenMusic sur Linux
Diplôme d'ingénieur EPITA, Paris
(co-encadré par Carlos Agon)

Autres (supervision de travaux)

2016-2017 Anders Vinjar (*développement*)
Projet ANR JCJC EFFICACe

2015-2016 Adrien Ycart (*ingénieur de recherche*)
Unité Projet Innovation IRCAM *Quantification rythmique*
(co-encadré par Florent Jacquemard)

2014-2015 Jérémie Garcia (*post-doctorat*)
Projet ANR JCJC EFFICACe

Résidences / Recherche musicale IRCAM

2017 Savannah Agger
Using 3D models to expose, control and process sound objects on a micro-level
(Projet ANR EFFICACe)

2015 Geof Holbrook
Toward a GA-oriented composition environment
(Résidences en Recherche Artistique IRCAM)

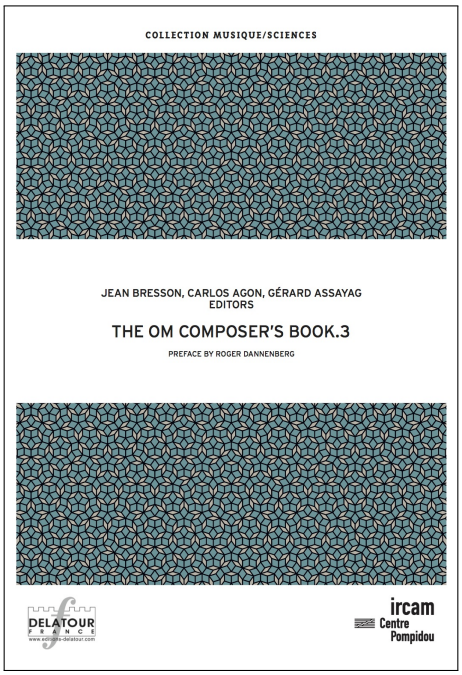
2014 John MacCallum et Teoma Naccaratto
Heart rate from contemporary dancers
(Résidence en Recherche Artistique IRCAM / Projet ANR EFFICACe)

2012 Marco Stroppa
Re Orso
(Conseiller scientifique)

2011 Roque Rivas
Ecriture et contrôle de la synthèse dans OMChroma
(Résidence en Recherche Artistique IRCAM)

THE OM COMPOSER'S BOOK – VOLUME 3

Après deux précédents volumes publiés en 2006 et 2008, *The OM Composer's Book 3* (2016) [1] propose 21 nouveaux témoignages de compositeurs utilisant OpenMusic et la composition assistée par ordinateur dans leurs travaux. Nous donnons ici un rapide aperçu du contenu des différents chapitres.



1. ISBN 978-2-7521-0283-6. Editions Delatour France / Ircam-Centre Pompidou.

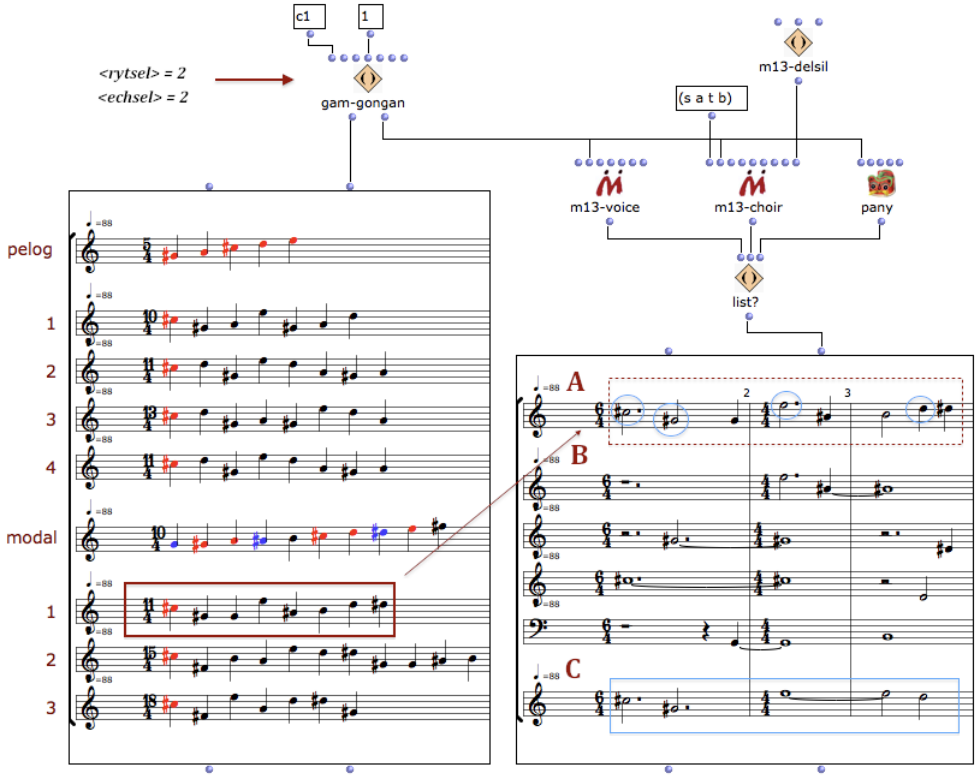


FIGURE A.1: Figure extraite du chapitre [🕮 3] « *Tri Bhuwana* for 12 voices and Balinese gamelan: Three worlds/tripartite world ». Modélisation et prolifération de motifs de musique traditionnelle balinaise.

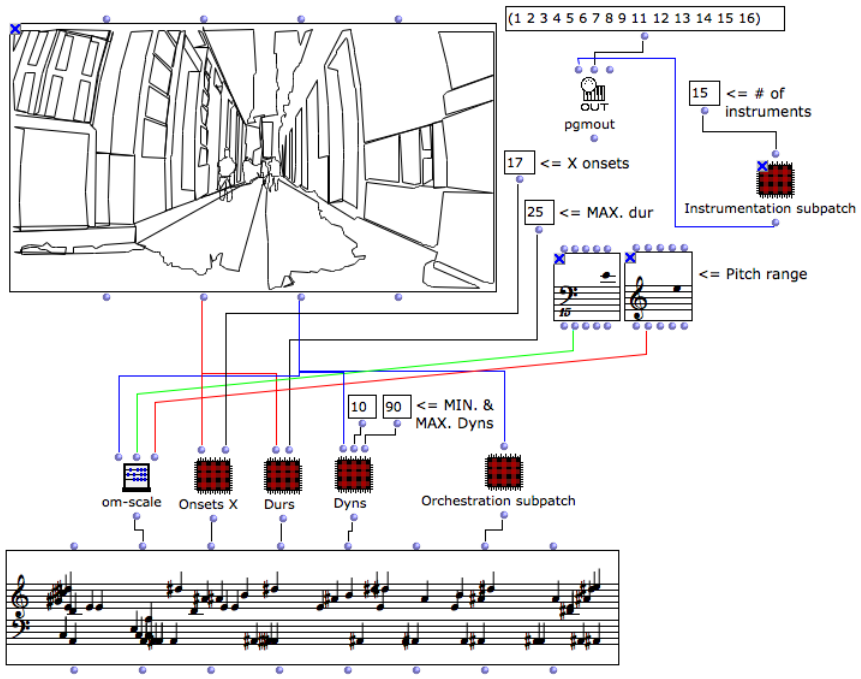


FIGURE A.2: Figure extraite du chapitre [🕮 5] « Musicalising sonification: Image-to-music conversions using OpenMusic ». Création de partitions MIDI à partir de contours graphiques.

1. *Variazioni su AlDo ClEmenti*

Dans ce chapitre Michele Zaccagnini (compositeur italien, basé aux Etats Unis) décrit les algorithmes mis en œuvre pour la composition de deux mouvements de sa pièce *Variazioni su AlDo ClEmenti* (2013) pour orchestre de chambre. Ceux-ci implémentent différents processus comme la simulation d'un phénomène de rebonds pour la génération de structures rythmiques, des effets de canon, et de manière plus générale, une exploration des rapports entre les états de chaos et d'équilibre dans les structures musicales.

2. Folk material transformations and elaborations in *A Vida é Nossa*

Gonçalo Gato (Portugal) décrit les procédures utilisées pour générer le matériau musical de sa pièce *A Vida é Nossa* (2013) pour orchestre d'instruments à vent. Ce matériau est dérivé d'extraits de musique populaire, et traité par différents algorithmes (concentration rythmique, mélodique, prolifération, ...) Gato développe notamment une réflexion entre production algorithmique et interventions « manuelles » dans ces processus.

3. *Tri Bhuwana* for 12 voices and Balinese gamelan: Three worlds/tripartite world

Dans ce chapitre, Philippe Boivin (France) modélise grâce à OpenMusic des processus et textures musicales balinaises (voir figure A.1). Les modèles créés permettent de produire du nouveau matériel par variation. Les structures musicales générées sont utilisées pour la composition de *Tri Bhuwana* (2013) pour Gamelan et voix.

4. Programming modular progressions in OpenMusic

Matthew Lane (Canada) décrit dans ce chapitre un système modulaire de combinaisons de processus qu'il a conçu et implémenté dans OpenMusic. Son application, ainsi que les opérations d'interprétation « post-calculatoires » effectuées sur le matériau musical produit, sont mis à exécution dans les pièces *Sliding Apart* (2013) et *Short Pieces on Falling* (2015) pour flûte, clarinette, piano, violon, et violoncelle.

5. Musicalising sonification: Image-to-music conversions using OpenMusic

Luiz Castelões (Brésil) utilise l'environnement de CAO pour implémenter des processus de conversion de données graphiques (images, contours, couleurs) vers des structures musicales. Ces processus sont utilisés dans la bande sonore du projet *VIA* (2013) (figure A.2 ci-contre) et dans la pièce *3 Transcrições* (2011).

6. On "slow" computer-aided composition

Julien Vincenot (France) présente des outils dédiés à l'interpolation, à la transformation de séquences symboliques ou encore à la programmation par contraintes, développés pour la plupart avec le logiciel PWGL. Il décrit leur application dans les pièces *Ascidiacea* (2010) pour flûte basse, *silent_data_corrupt* (2014) pour saxophone alto et électronique temps-réel, et *Mémoire de l'eau* (2015) pour accordéon et électronique temps-réel, et propose une réflexion sur l'utilisation des outils de CAO « temps différé » comparée aux pratiques « temps réel ».

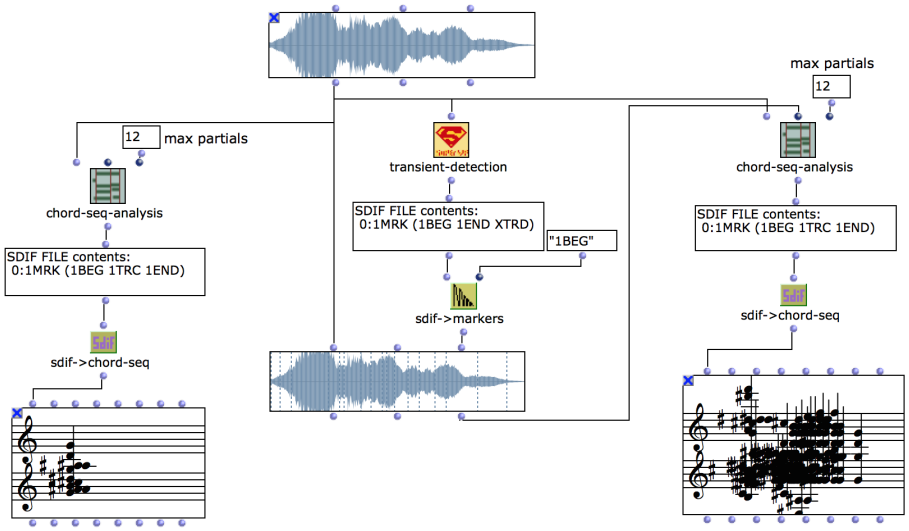


FIGURE A.3: Figure extraite du chapitre [8] « Computer-aided composition in the creation of *As I ride the late night freeways* ». Extraction de structures harmoniques à partir d'enregistrements sonores.

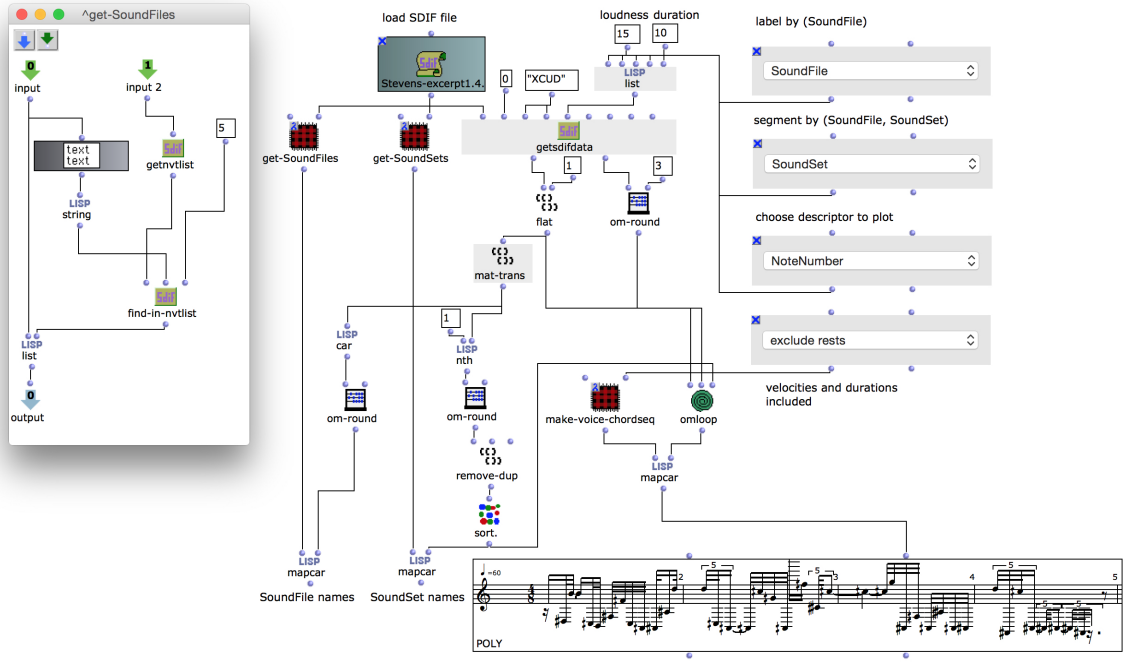


FIGURE A.4: Figure extraite du chapitre [10] « Musique instrumentale concrète: Timbral transcription in *What the Blind See* and *Without Words* ». Conversion de séquences de descripteurs audio en notation musicale symbolique.

7. OM-Darwin: Generative and descriptive aspects of genetic algorithms

Geof Holbrook (Canada) présente la bibliothèque OM-DARWIN et son utilisation dans la pièce pour ensemble *Future Perfect* (2010). OM-DARWIN est une implémentation musicale du principe d'algorithme génétique, visant à optimiser une « population » d'individus caractérisés par des gènes (dans ce cas, un encodage symbolique de séquences musicales) à partir de critères déterminant leur survie ou leur propagation au fil des générations.

8. Computer-aided composition in the creation of *As I ride the late night freeways*

Matt Schumaker (Etats Unis) présente plusieurs procédés convoqués pour la création de sa pièce *As I ride the late night freeways* pour soprano et orchestre, écrite en 2014-2015. Inspirés d'un poême de Cathy Park Hong, des concepts de formes « paramétriques », et plus généralement de l'activité de conduite automobile, les procédés décrits dans ce chapitre consistent principalement en la production de textures sonores et harmoniques à partir d'analyses spectrales d'enregistrements d'automobiles (cf. figure A.3), et en la création de courbes évoquant l'idée d'aérodynamisme, appliquées à la formation de lignes mélodiques.

9. Materials and techniques in *D'improvviso da immobile s'illumina* for bass clarinet, two orchestras, piano, and percussion

Le compositeur et guitariste Federico Bonacossa (Italie, basé aux Etats Unis) décrit dans ce chapitre divers aspects de la composition de sa pièce *D'improvviso da immobile s'illumina* (2013), un concerto pour clarinette basse, deux orchestres, piano, et percussion. Dans le domaine des hauteurs, il met à contribution conjointement des procédés d'analyse spectrale et les outils de la *set theory* proposés par le package MATHTOOLS de OpenMusic. Du point de vue rythmique, il s'intéresse à l'idée de *rhythmicon* (une machine poly-rythmique créée par H. Cowell dans les années 30) pour créer des structures polyphoniques mettant en relation les caractéristiques rythmiques (durées) et spectrales (hauteurs) des événements musicaux.

10. Musique instrumentale concrète: Timbral transcription in *What the Blind See* and *Without Words*

Aaron Einbond (Etats Unis, basé au Royaume Uni) s'intéresse à l'utilisation de bases de données indexées par des descripteurs de signaux pour la transcription symbolique d'enregistrements sonores suivant un principe de « mosaïque ». Le logiciel de synthèse concaténative par corpus CataRT est utilisé pour la recherche de grains sonores suivant des critères exprimés par valeurs de descripteurs audio. La transcription symbolique permet de transférer les résultats sous forme de partition dédiée à l'interprétation instrumentale (voir figure A.4). Ce procédé est décliné dans la composition de *What the Blind See* (2009) pour alto, clarinette basse, harpe, piano, percussion, et électronique, et dans *Without Words* (2012) pour soprano, onze instruments, et électronique.

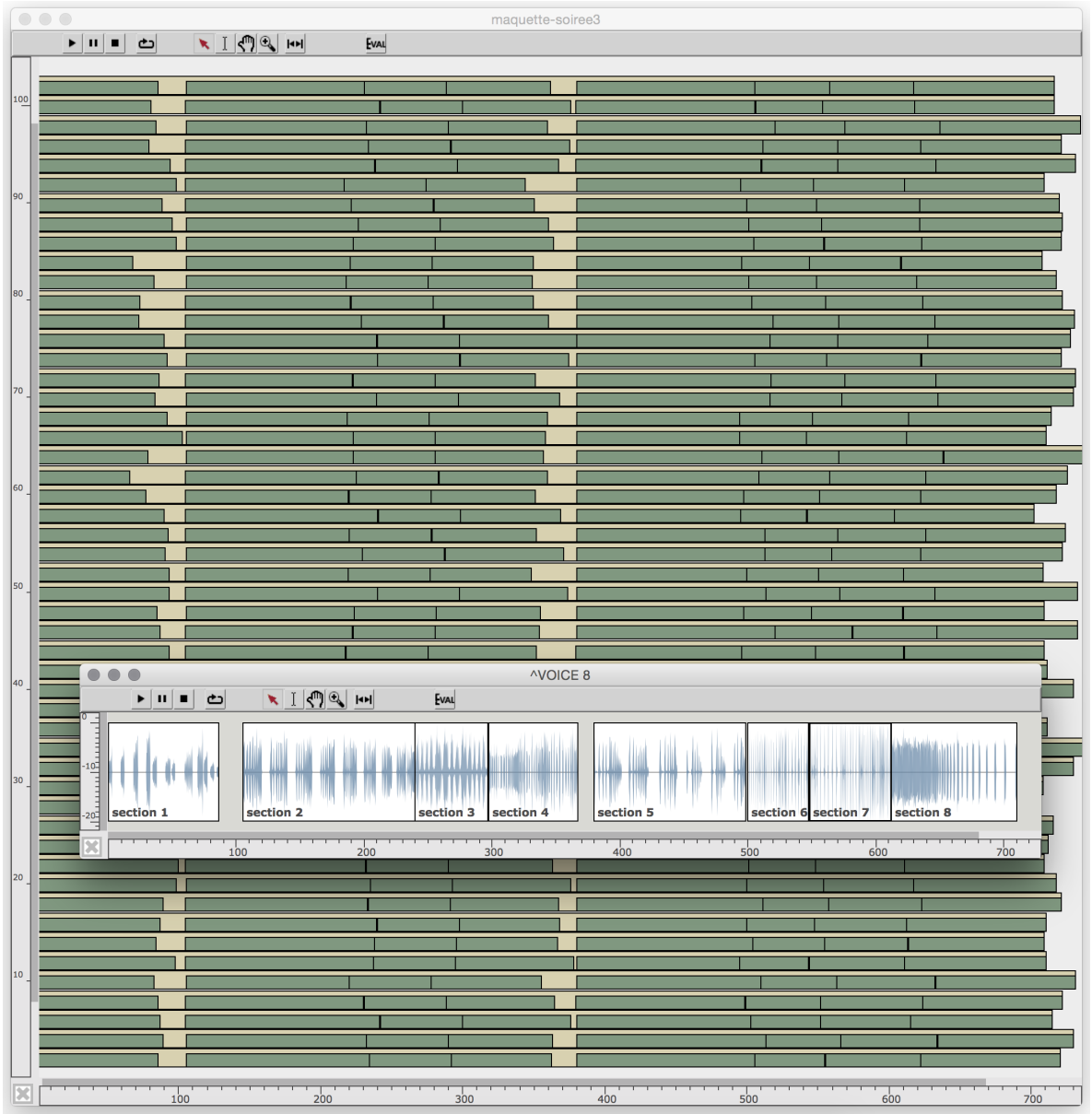


FIGURE A.5: Figure extraite du chapitre [14] « *Germination* ». Maquette générale d'une représentation. Chacune des 50 voix contient une variation sur une séquence de 8 sections sonores.

11. Recomposing Beethoven with Music Neurotechnology

Eduardo Miranda (Brésil, basé au Royaume Uni) et Anders Vinjar (Norvège) présentent dans ce chapitre des techniques de composition basées sur la *neuro-technologie* et l'utilisation d'imagerie cérébrale par résonance magnétique fonctionnelle, appliquées dans les pièces pour orchestres de Miranda *Symphony of Minds Listening* (2013), *Corpus Callosum* (2015) et *Shockwaves* (2015). Les données d'imagerie cérébrale de sujets écoutant de la musique sont indexées, classifiées et transformées pour recomposer des structures musicales.

12. Composing for the resonance: Finding new relationships between architecture and musical composition

Ambrose Field (Royaume Uni) utilise les caractéristiques de résonance (réponses acoustiques) captées dans des espaces de concert ou de projection sonore afin de créer le matériau instrumental (structures rythmiques et harmoniques) destiné à être exécuté dans ces mêmes espaces. Ce procédé est mis en œuvre dans *Quantaform Series* (2013) pour « flûte et résonances » et dans *Architexture Series* (2012-2015), un ensemble de pièces pour ensemble vocal.

13. Sketching, synthesis, spatialisation: The integrated workflow of *Cognitive Consonance*

Dans ce chapitre Christopher Trapani (Etats Unis) décrit le rôle de la CAO dans la composition d'un mouvement de sa pièce *Cognitive Consonance* (2011). Ce mouvement est joué par un qanûn, instrument traditionnel oriental, et comprend une partie électronique. Une première phase d'expérimentation avec les micro-intervalles a permis l'élaboration de données harmoniques et d'échelles pour l'écriture instrumentale. Une deuxième phase, axée sur la production de sons de synthèse, associe la bibliothèques MODALYS (synthèse par modèles physiques), le montage d'échantillons sonores et la synthèse sonore spatialisée via la bibliothèque OMPRISMA.

14. *Germination*

Dans ce chapitre Jean-Luc Hervé et Serge Lemouton (France) décrivent un procédé algorithmique de montage audio réalisé pour une pièce/installation sonore (*Germination*, 2013) diffusée en plein air sur 50 canaux indépendants. Des banques de sons sont traitées et transformées (notamment via la bibliothèque OM-SUPERVP) afin de produire une cohérence de forme, spatiale et sonore (cf. figure A.5). Certains aspects paramétrables (ou aléatoires) du processus permettent de générer une bande sonore différente pour chaque représentation.

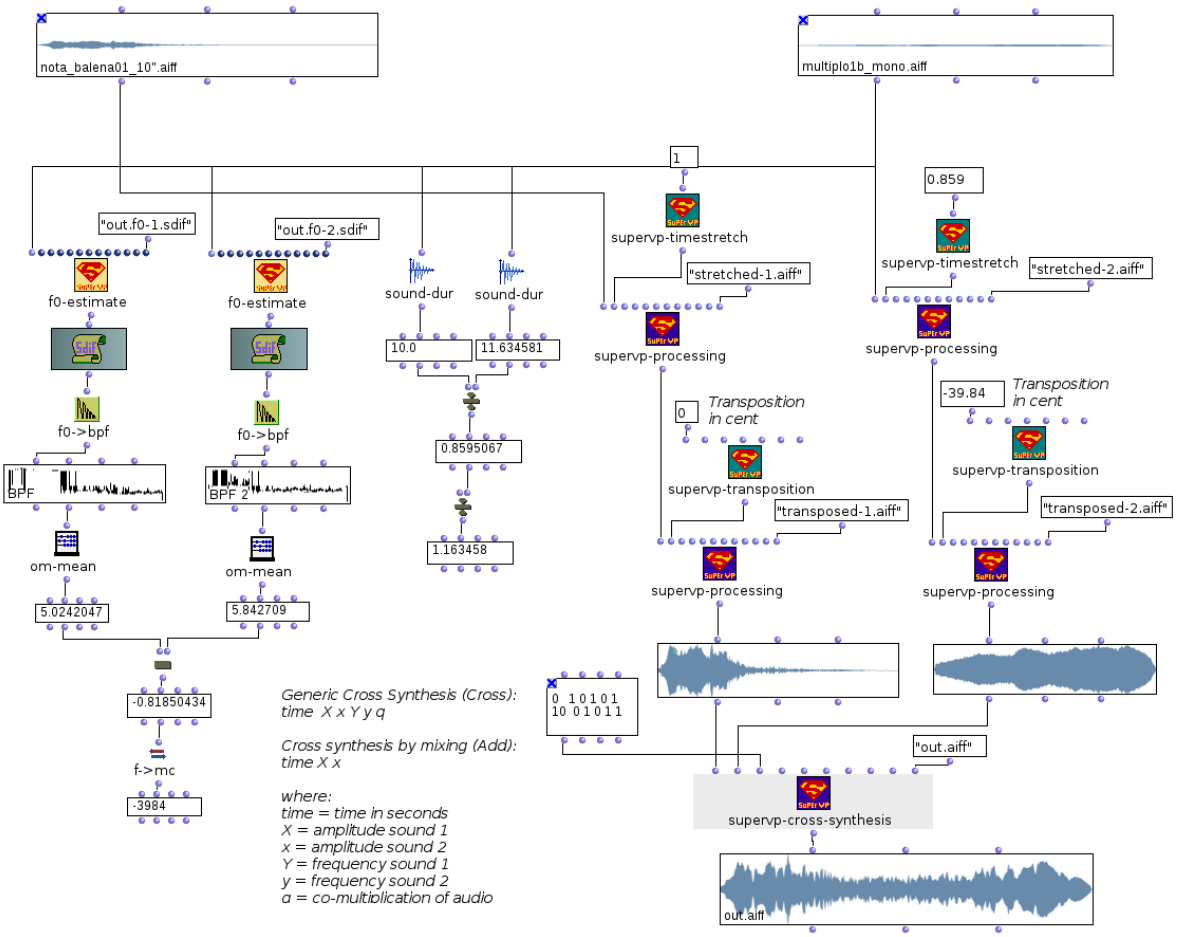


FIGURE A.6: Figure extraite du chapitre [17] « Electronic sound creation in *Balænoptera* ». Transformations de matériau sonore avec OM-SUPERVP.

15. Dialogue with OpenMusic in the process of composing *Nothing that is not there and the Nothing that is*

Takeshi Tsuchiya (Japon) s'inspire de la tradition japonaise pour concevoir la structure globale de son travail et son lien aux détails de réalisation. Dans ce chapitre il décrit des procédés mis en œuvre pour la composition de *Nothing that is not there and the Nothing that is* (2013) pour violoncelle et électronique, notamment la construction de la forme (utilisant la *maquette* d'OpenMusic), les processus de synthèse et de traitement sonore pour la partie électronique, ou encore la création de lignes mélodiques.

16. *Koch's Space*

Dans *Koch's Space* (2011), pour saxophone et électronique temps réel, Julián Ávila (Espagne) génère dans OpenMusic des formes fractales utilisées pour produire la partition instrumentale, mais aussi pour paramétrer un « filtre topologique » appliquant différentes atténuations de fréquences selon la position du son spatialisé en temps réel et projeté dans la salle via un dispositif multicanal.

17. Electronic sound creation in *Balænoptera* for bass clarinet, electronic sounds, and live electronics

Fabio De Sanctis De Benedictis (Italie) décrit les processus réalisés dans OpenMusic pour la création des parties électroniques de sa pièce *Balænoptera* (2015) pour clarinette basse, bande et électronique temps-réel. De nombreux logiciels sont mis à contribution, ainsi que plusieurs bibliothèques OpenMusic pour l'analyse, le traitement, la synthèse et la spatialisation sonore (voir figure A.6). Des structures rythmiques sont également programmées et utilisées pour l'organisation temporelle et le montage des fichiers audio synthétisés.

18. Modelling a gesture: *Tak-Sīm* for string quartet and live electronics

Pour composer la pièce *Tak-Sīm* (2012) pour quatuor à cordes, Alireza Farhang (compositeur franco-iranien) s'est inspiré de la tradition musicale persane et de sa confrontation à l'univers de la musique savante occidentale. L'analyse des harmonies, puis celle des gestes virtuoses pratiqués sur le setār, permettent au compositeur d'extraire des structures musicales utilisées pour la composition des parties instrumentales, et pour la synthèse des sons composant la partie électronique de l'œuvre.

19. Electronic dramaturgy and computer-aided composition in *Re Orso*

Dans ce chapitre nous décrivons une partie des procédés de CAO mis en œuvre lors de la création de l'opéra *Re Orso* de Marco Stroppa (2012). Les processus mis en avant dans ce texte sont ceux liant les modèles d'analyse à la production de sons de synthèse (par exemple pour la synthèse des sons de cloche), ou encore ceux utilisant ces modèles pour le contrôle d'un piano robotisé. Nous décrivons également l'utilisation du synthétiseur Chant et de la bibliothèque OM-CHANT pour la production de voix virtuelles (un aspect également traité dans la section 5.3, chapitre 4 de ce mémoire).

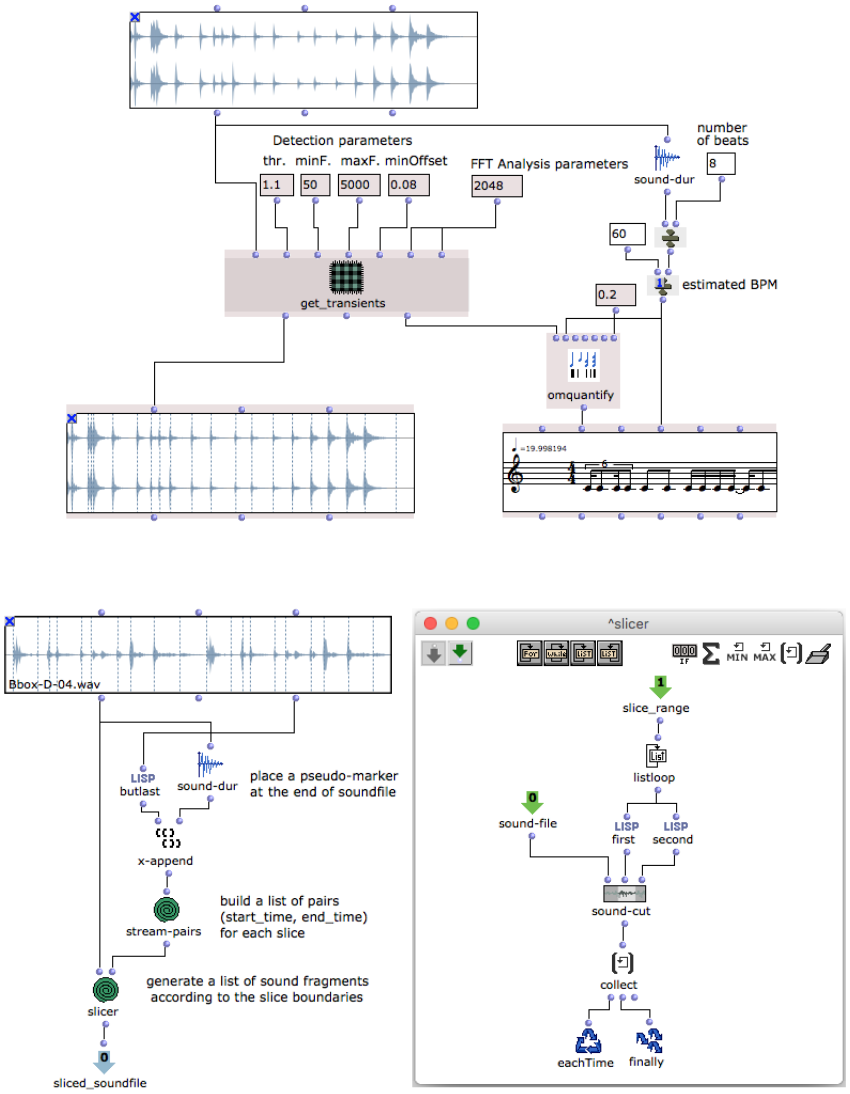


FIGURE A.7: Figures extraites du chapitre [20] « *Rima Flow*: Oral tradition and composition ». Découpage et montage algorithmique de segments audio. Note : la modification de paramètres dans les portions *réactives* (orangées) du programme visuel en haut de la figure entraine une mise à jour automatique de la segmentation et du rythme quantifié obtenu (voir chapitre 7, section 7.1).

20. *Rima Flow*: Oral tradition and composition

Alessandro Ratoci (Italie) décrit différents procédés compositionnels mis en œuvre avec OpenMusic dans la pièce *Rima Flow* (2015) pour tuba et électronique. Cette pièce explore un face-à-face entre une tradition orale italienne appelée *ottava rima* et les pratiques actuelles de *beat-boxing*. Les procédés développés comprennent l'analyse audio et symbolique de corpus musicaux, la transcription (notamment rythmique), la transformation algorithmique de matériau musical (également audio ou symbolique), ou encore la synthèse sonore et le montage audio (figure A.7 ci-contre).

21. *Ab-Tasten*: Atomic sound modelling with a computer-controlled grand piano

Dans ce dernier chapitre Marlon Schumacher (Allemagne) décrit un procédé de micro-montage sonore appelé *Corpus-based Atomic Decomposition*. À l'aide de sa bibliothèque OM-PURSUIT, Schumacher réalise des recompositions d'extraits sonores « cibles » à partir de superposition de grains sonores issus d'un corpus donné. Dans *Ab-Tasten* (2011) pour piano robotisé et électronique, une base d'échantillon de sons de piano est créée dans OpenMusic en commandant automatiquement le piano mécanique via MIDI, et enregistrant les notes jouées successivement à l'aide de la bibliothèque OM-SOX, également créée par l'auteur. Enfin, Schumacher développe dans ce travail un effet perceptif de synthèse spatialisée réalisé en alliant procédés compositionnels, synthèse sonore et configuration spatiale de la pièce.

BIBLIOGRAPHIE

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, C. et Assayag, G. (2003). OM: A Graphical Extension of CLOS using the MOP. *In Proceedings of the International Lisp Conference (ILC'03)*, New York, USA.

Agon, C., Assayag, G., Fineberg, J. et Rueda, C. (1994). Kant: A critique of pure quantification. *In Proceedings of the International Computer Music Conference (ICMC'94)*, Aarhus, Denmark.

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, C., Stroppa, M. et Assayag, G. (2000). High Level Control of Sound Synthesis in OpenMusic. *In Proceedings of the International Computer Music Conference (ICMC'00)*, Berlin, Germany.

Agostini, A. et Ghisi, D. (2015). A Max Library for Musical Notation and Computer-Aided Composition. *Computer Music Journal*, 39(2).

Ahrens, J., Geier, M. et Spors, S. (2008). The SoundScape Renderer: A Unified Spatial Audio Reproduction Framework for Arbitrary Rendering Methods. *In Audio Engineering Society Convention 124*, Amsterdam, The Netherlands.

Allen, J. F. (1983). Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26.

Allombert, A., Desainte-Catherine, M., Laralde, J. et Assayag, G. (2008). A system of interactive scores based on qualitative and quantitative temporal constraints. *In ARTECH 2008. Proceedings of the 4th International Conference on Digital Arts*, Porto, Portugal.

Anders, T. et Miranda, E. (2011). A Survey of Constraint Programming Systems for Modeling Music Theories and Composition. *ACM Computing Surveys*, 43(4).

Andreatta, M. et Agon, C. (2003). Implementing Algebraic Methods in OpenMusic. *In Proceedings of the International Computer Music Conference (ICMC'03)*, Singapore.

Assayag, G. (2009). Algorithmes, langages et modèles pour la recherche musicale : de la composition à l'interaction. Habilitation à Diriger des Recherches, Université de Bordeaux 1.

Assayag, G. (2016). Improvising in Creative Symbolic Interaction. *In* Smith, J. B. L., Chew, E. et Assayag, G., éditeurs : *Mathematical Conversations: Mathematics and Computation in Music Performance and Composition*, numéro 32 de Lecture Notes Series. World Scientific.

Assayag, G., Agon, C., Fineberg, J. et Hanappe, P. (1997). An Object Oriented Visual Environment for Musical Composition. *In Proceedings of the International Computer Music Conference (ICMC'97)*, Thessaloniki, Greece.

Assayag, G., Bloch, G., Chemillier, M., Cont, A. et Dubnov, S. (2006). Omax Brothers: A Dynamic Topology of Agents for Improvization Learning. *In Workshop on Audio and Music Computing for Multimedia, ACM MultiMedia*, Santa Barbara, CA, USA.

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 (ICMC'85)*, Vancouver, Canada.

Assayag, G. et Rueda, C. (1993). The Music Representation Project at Ircam. *In Proceedings of the International Computer Music Conference (ICMC'93)*, Tokyo, Japan.

Assayag, G., Rueda, C., Laurson, M., Agon, C. et Delerue, O. (1999). Computer Assisted Composition at IRCAM: From PatchWork to OpenMusic. *Computer Music Journal*, 23(3).

Assayag, G. et Timis, D. (1986). A toolbox for music notation. *In Proceedings of the International Computer Music Conference*, Den Haagen, The Netherlands.

Baalman, M. A. (2010). Spatial Composition Techniques and Sound Spatialisation Technologies. *Organised Sound*, 15(3).

Barbaud, P. (1968). *La musique, discipline scientifique*. Dunod.

Berkhout, A. J., de Vries, D. et Vogel, P. (1993). Acoustic Control by Wave Field Synthesis. *The Journal of the Acoustical Society of America*, 93(5).

Bevilacqua, F., Schnell, N., Rasamimanana, N., Zamborlin, B. et Guédy, F. (2011). Online Gesture Analysis and Control of Audio Processing. *In* Solis, J. et Ng, K., éditeurs : *Musical Robots and Interactive Multimodal Systems*. Springer.

Bigo, L. (2013). *Représentations symboliques musicales et calcul spatial*. Thèse de doctorat, Université Paris-Est LACL/IRCAM, Paris, France.

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.

Boshernitsan, M. et Downes, M. (1997). Visual Programming Languages: A Survey. Rapport technique CSD-04-1368, University of California, Berkeley, USA.

Boulanger, R., éditeur (2000). *The Csound Book: Perspectives in Software Synthesis, Sound Design, Signal Processing, and Programming*. MIT Press.

Boynton, L., Duthen, J., Potard, Y. et Rodet, X. (1986). Adding a Graphical User Interface to FORMES. *In Proceedings of the International Computer Music Conference (ICMC'86)*, Den Haag, Netherlands.

Burnett, M. (1999). Visual programming. *In Wiley Encyclopedia of Electrical and Electronics Engineering*. John Wiley & Sons.

Burnett, M., Atwood, J., Walpole Djang, R., Reichwein, J., Gottfried, H. et Yang, S. (2001). Forms/3: A first-order visual language to explore the boundaries of the spreadsheet paradigm. *Journal of Functional Programming*, 11(2).

Burnett, M. et Scaffidi, C. (2014). End-user development. *In* Soegaard, M. et Dam, R. F., éditeurs : *The Encyclopedia of Human-Computer Interaction*. Interaction Design Foundation.

Burnett, M. M. et Baker, M. J. (1994). A Classification System for Visual Programming Languages. *Journal of Visual Languages and Computing*, 5(3).

Camurri, A. (1990). On the Role of Artificial Intelligence in Music Research. *Interface, Journal of New Music Research*, 19(2-3).

Caramiaux, B., Montecchio, N., Tanaka, A. et Bevilacqua, F. (2014). Adaptive Gesture Recognition with Variation Estimation for Interactive Systems. *ACM Transactions on Interactive Intelligent Systems*, 4(4).

Carpentier, G. (2008). *Approche computationnelle de l'orchestration musicale – Optimisation multicritère sous contraintes de combinaisons instrumentales dans de grandes banques de sons*. Thèse de doctorat, Université Pierre et Marie Curie, Paris, France.

Carpentier, G., Assayag, G. et Saint-James, E. (2010). Solving the musical orchestration problem using multiobjective constrained optimization with a genetic local search approach. *Journal of Heuristics*, 16(5).

Carpentier, T., Noisternig, M. et Warusfel, O. (2015). Twenty Years of Ircam Spat: Looking Back, Looking Forward. *In Proceedings of the International Computer Music Conference*, Denton, TX, USA.

Cointe, P. et Rodet, X. (1984). Formes: An Object and Time Oriented System for Music Composition and Synthesis. *In Proceedings of the 1984 ACM Symposium on LISP and Functional Programming*, Austin, USA.

Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Löding, C., Lugiez, D., Tison, S. et Tommasi, M. (2007). Tree Automata Techniques and Applications. `http://tata.gforge.inria.fr`.

Cont, A. (2008). ANTESCOFO: Anticipatory Synchronization and Control of Interactive Parameters in Computer Music. *In Proceedings of the International Computer Music Conference (ICMC'08)*, Belfast, Ireland.

Cope, D. (1996). *Experiments in Musical Intelligence*. A-R Editions, Madison.

Cox, P. T., Giles, F. R. et Pietrzykowski, T. (1989). Prograph: A step towards liberating programming from textual conditioning. *In IEEE Workshop on Visual Languages*, Rome, Italy.

Daniel, J. (2000). *Représentation de champs acoustiques, applications à la transmission et à la reproduction de scènes sonores complexes dans un contexte multimedia*. Thèse de doctorat, Université Pierre et Marie Curie, Paris, France.

Dannenberg, R. B. (1997). Machine Tongues XIX: Nyquist, a Language for Composition and Sound Synthesis. *Computer Music Journal*, 21(3).

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, R. B., McAvinney, P. et Rubine, D. (1986). Arctic: A Functional Language for Real-Time Systems. *Computer Music Journal*, 10(4).

DeHon, A., Giavitto, J.-L. et Gruau, F., éditeurs (2007). *Computing Media Languages for Space-Oriented Computation*, Dagstuhl Seminar Proceedings – 06361 Executive Report, Dagstuhl, Germany. Internationales Begegnungs und Forschungszentrum für Informatik (IBFI).

Delerue, O. (2004). *Spatialisation du son et programmation par contraintes : le système MusicSpace*. Thèse de doctorat, Université Pierre et Marie Curie, Paris, France.

Delerue, O. et Agon, C. (1999). Openmusic + musicspace = openspace. *In Actes de Journées d'Informatique Musicale (JIM'99)*, Issy-les-Moulineaux, France.

Depalle, P., Rodet, X., Galas, T. et Eckel, G. (1993). Generalized Diphone Control. *In Proceedings of the International Computer Music Conference (ICMC'93)*, Tokyo, Japan.

Dershowitz, N. et Jouannaud, J.-P. (1990). Rewrite systems. *In* van Leeuwen, J., éditeur : *Handbook of Theoretical Computer Science*, volume Volume B: Formal Models and Semantics, chapitre 6, pages 243–320. North-Holland, Amsterdam.

Desain, P. (1986). Graphical Programming in Computer Music – A Proposal. *In Proceedings of the International Computer Music Conference (ICMC'86)*, Den Haag, Netherlands.

Desain, P. et Honing, H. (1992). The Quantization Problem: Traditional and Connectionist Approaches. *In* Balaban, M., Ebcioglu, K. et Laske, O., éditeurs : *Understanding Music with AI: Perspectives on Music Cognition*. MIT Press.

Desainte-Catherine, M. et Allombert, A. (2005). Interactive scores: A model for specifying temporal relations between interactive and static events. *Journal of New Music Research*, 34(4).

desJardins, M. E., Durfee Jr., E. H., Ortiz, C. L. et Wolverton, M. J. (1999). A Survey of Research in Distributed, Continual Planning. *AI Magazine*, 20(4).

Echeveste, J., Cont, A., Giavitto, J.-L. et Jacquemard, F. (2013). Operational semantics of a domain specific language for real time musician-computer interaction. *Discrete Event Dynamic Systems*, 23(4).

Eckel, G. (1993). La maîtrise de la syntèse sonore. *In Les cahiers de l'Ircam (2) – La synthèse sonore*. IRCAM - Centre Pompidou.

Eckel, G. et Gonzalez-Arroyo, R. (1994). Musically Salient Control Abstractions for Sound Synthesis. *In Proceedings of the International Computer Music Conference (ICMC'94)*, Aarhus, Denmark.

Eckel, G., González-Arroyo, R. et Rumori, M. (2004). Once again text & parenthesis – sound synthesis with Foo. *In Proceedings of the Linux Audio Conference (LAC'04)*, Karlsruhe, Germany.

Elliott, C. et Hudak, P. (1997). Functional Reactive Animation. *In International Conference on Functional Programming (ICFP'97)*, Amsterdam, The Netherlands.

Erwig, M. et Meyer, B. (1995). Heterogeneous Visual Languages – Integrating Visual and Text Programming. *In 11th IEEE Symposium on Visual Languages*, Darmstadt, Germany.

Françoise, J. (2013). Gesture-Sound Mapping by Demonstration in Interactive Music Systems. *In ACM international conference on Multimedia (MM'13)*, Barcelona, Spain.

Freed, A., MacCallum, J. et Schmeder, A. (2011). Dynamic, Instance-Based, Object-Oriented Programming in Max/MSP using Open Sound Control Message Delegation. *In Proceedings of the International Computer Music Conference (ICMC'11)*, Huddersfield, UK.

Gabriel, R. P., White, J. L. et Bobrow, D. G. (1991). CLOS: Integration Object-oriented and Functional Programming. *Communications of the ACM*, 34(9).

Garcia, J. (2014). *Supporting Music Composition with Interactive Paper*. Thèse de doctorat, Université Paris-Sud.

Garcia, J., Tsandilas, T., Agon, C. et Mackay, W. (2012). Interactive Paper Substrates to support Musical Creation. *In CHI'12: Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, Austin, USA.

Godøy, R. I. et Leman, M., éditeurs (2010). *Musical Gestures: Sound, Movement, and Meaning*. Routledge.

Graham, R. L., Lawler, E. L., Lenstra, J. K. et Kan, A. H. G. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5.

Green, T. R. et Petre, M. (1992). When visual programs are harder to read than textual programs. *In Proceedings of the Sixth European Conference on Cognitive Ergonomics (ECCE 6)*.

Haddad, K. (2008). *Livre Premier de Motets*: The Time-Block Concept in OpenMusic. *In* Bresson, J., Agon, C. et Assayag, G., éditeurs : *The OM Composer's Book. 2*. Editions Delatour / Ircam-Centre Pompidou.

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.

Harley, M. A. (1994). *Space and Spatialization in Contemporary Music : History and Analysis, Ideas and Implementations*. Thèse de doctorat, McGill University, School of Music, Montreal, Canada.

Helsel, R. (1998). *Visual programming with HP VEE (3rd Edition)*. Prentice-Hall, Upper Saddle River, USA.

Hervé, J.-L. et Voisin, F. (2006). Composing the Qualitative, on *Encore* Composition. *In* Agon, C., Assayag, G. et Bresson, J., éditeurs : *The OM Composer's Book. 1*. Editions Delatour / Ircam-Centre Pompidou.

Hiller, L. A. et Isaacson, L. M. (1959). *Experimental Music: Composition With an Electronic Computer*. McGraw-Hill, New York.

Hils, D. D. (1991). DataVis: A Visual Programming Language for Scientific Visualization. *In Proceedings of the ACM Computer Science Conference*, San Antonio, USA.

Hirs, R. et Gilmore, B., éditeurs (2009). *Contemporary Compositional Techniques and OpenMusic*. Editions Delatour France.

Honing, H. (2001). From Time to Time: The Representation of Timing and Tempo. *Computer Music Journal*, 25(3).

Huang, L. et Chiang, D. (2005). Better *k*-best parsing. *In Proceedings of the International Workshop on Parsing Technology*, Vancouver, Canada.

Hudak, P., Makucevich, T., Gadde, S. et Whong, B. (1996). Haskore Music Notation – An Algebra of Music. *Journal of Functional Programming*, 6(3).

Hutton, G. (1999). A Tutorial on the Universality and Expressiveness of Fold. *Journal of Functional Programming*, 9(4).

Jot, J.-M. (1999). Real-time spatial processing of sounds for music, multimedia and interactive human-computer interfaces. *ACM Multimedia Systems Journal (Special issue on Audio and Multimedia)*, 7(1):55–69.

Jot, J.-M. et Warusfel, O. (1995). A Real-Time Spatial Sound Processor for Music and Virtual Reality Applications. *In Proceedings of the International Computer Music Conference (ICMC'95)*, Banff, Canada.

Kendall, G., Peters, N. et Geier, M. (2008). Towards an Interchange Format for Spatial Audio Scenes. *In Proceedings of the International Computer Music Conference (ICMC'08)*, Belfast, Ireland.

Kiczales, G., des Rivieres, J. et Bobrow, D. G. (1991). *The Art of the Metaobject Protocol*. MIT Press.

Kim-Boyle, D. (2008). Spectral Spatialization – An Overview. *In Proceedings of the International Computer Music Conference (ICMC'08)*, Belfast, Ireland.

Kopetz, H. (1991). Event-triggered versus time-triggered real-time systems. *In* Karshmer, A. et Nehmer, J., éditeurs : *Operating Systems of the 90s and Beyond*, numéro 563 de Lecture Notes in Computer Science. Springer Berlin Heidelberg.

Larkin, J. H. et Simon, H. A. (1987). Why a Diagram is (Sometimes) Worth Ten Thousand Words. *Cognitive Science*, 11.

Lau-Kee, D., Billyard, A., Faichney, R., Kozato, Y., Otto, P., Smith, M. et Wilkinson, I. (1991). VPL: An Active, Declarative Visual Programming System. *In Proceedings of the IEEE Workshop on Visual Languages*, Kobe, Japan.

Laurson, M. et Duthen, J. (1989). Patchwork, a Graphic Language in PreForm. *In Proceedings of the International Computer Music Conference (ICMC'89)*, Ohio State University, USA.

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 (ICMC'02)*, Gothenburg, Sweden.

Leman, M. (1993). Symbolic and subsymbolic description of music. *In* Haus, G., éditeur : *Music Processing*. Oxford University Press.

Lemouton, S. et Schaathun, A. (2008). Knitting and Weaving: Using OpenMusic to Generate Canonic Musical Material. *In* Bresson, J., Agon, C. et Assayag, G., éditeurs : *The OM Composer's Book. 2*. Editions Delatour / Ircam-Centre Pompidou.

Lopez-Lezcano, F. (2008). A Dynamic Spatial Locator ugen for CLM. *In Proceedings of the Sound and Music Computing Conference (ICMC'08)*, Berlin, Germany.

Lossius, T., Baltazar, P. et de la Hogue, T. (2009). DBAP – Distance-Based Amplitude Panning. *In Proceedings of the International Computer Music Conference (ICMC'09)*, Montreal, Canada.

Loy, G. (1985). Musicians make a standard: the MIDI phenomenon. *Computer Music Journal*, 9(4).

MacCallum, J. et Schmeder, A. (2010). Timewarp: A Graphical Tool for the Control of Polyphonic Smoothly Varying Tempos. *In Proceedings of the International Computer Music Conference (ICMC'10)*, New York / Stony Brook, USA.

Malherbe, C. (2008). *Locus*: Rien n'aura eu lieu que le lieu. *In* Bresson, J., Agon, C. et Assayag, G., éditeurs : *The OM Composer's Book. 2*. Editions Delatour / Ircam-Centre Pompidou.

Mallat, S. et Zhang, Z. (1993). Matching Pursuits With Time-Frequency Dictionaries. *IEEE Transactions on Signal Processing*, 41(12).

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.

Mathews, M. (1963). The digital computer as a musical instrument. *Science*, 142(3592).

Mathews, M., éditeur (1969). *The Technology of Computer Music*. MIT Press.

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).

McCartney, J. (2006). Rethinking the computer music language: Supercollider. *Computer Music Journal*, 26(4).

McLeran, A., Roads, C., Sturm, B. L. et Shynk, J. J. (2008). Granular Sound Spatialization Using Dictionary-Based Methods. *In Proceedings of the Sound and Music Computing Conference (SMC'08)*, Berlin, Germany.

Melen, C. (2008). *Six Metal Fugue*. *In* Bresson, J., Agon, C. et Assayag, G., éditeurs : *The OM Composer's Book. 2*. Editions Delatour / Ircam-Centre Pompidou.

Meredith, D., éditeur (2015). *Computational Music Analysis*. Springer.

Meudic, B. (2004). *Détermination automatique de la pulsation, de la métrique et des motifs musicaux dans des interprétations à tempo variable d'oeuvres polyphoniques*. Thèse de doctorat, Université Pierre et Marie Curie, Paris.

Monden, N., Yoshimoto, I., Hirakawa, M., Tanaka, M. et Ichikawa, T. (1984). HI-VISUAL: A Language Supporting Visual Interaction in Programming. *In Proceedings of the IEEE Workshop on Visual Languages*, Hiroshima, Japan.

Niehren, J., Champavère, J., Gilleron, R. et Lemay, A. (2013). Query Induction with Schema-Guided Pruning Strategies. *Journal of Machine Learning Research*, 14.

Nierhaus, G. (2008). *Algorithmic Composition: Paradigms of Automated Music Generation*. Springer Verlag.

Nika, J. (2016). *Guiding human-computer music improvisation: introducing authoring and control with temporal scenarios*. Thèse de doctorat, Université Pierre et Marie Curie, Paris.

Nouno, G. (2008). Some Considerations on Brian Ferneyhough's Musical Language through His Use of CAC. Part II — Spatialization as a Musical Parameter. *In* Bresson, J., Agon, C. et Assayag, G., éditeurs : *The OM Composer's Book. 2*. Editions Delatour / Ircam-Centre Pompidou.

Nouno, G. et Agon, C. (2002). Contrôle de la spatialisation comme paramètre musical. *In Actes des Journées d'Informatique Musicale (JIM'02)*, Marseille, France.

Orlarey, Y., Fober, D. et Letz, S. (1997). Elody: a Java+Midishare based Music Composition Environment. *In Proceedings of the International Computer Music Conference (ICMC'97)*, Thessaloniki, Greece.

Orlarey, Y., Fober, D. et Letz, S. (2004). Syntactical and Semantical Aspects of Faust. *Soft Computing*, 8(9).

Orlarey, Y., Letz, S. et Fober, D. (2014). Version librairie du compilateur Faust. Inedit – ANR-12-CORD-0009, Grame.

Otondo, F. (2008). Contemporary trends in the use of space in electroacoustic music. *Organised Sound*, 13(1).

Peters, N., Lossius, T. et Schacher, J. C. (2013). The Spatial Sound Description Interchange For- mat: Principles, Specification, and Examples. *Computer Music Journal*, 37(1).

Peters, N., Marentakis, G. et McAdams, S. (2011). Current Technologies and Compositional Practices for Spatialization: A Qualitative and Quantitative Analysis. *Computer Music Journal*, 35(1).

Pope, S. T., éditeur (1991). *The Well Tempered Object (Musical Applications of Object-Oriented Software Technology)*. MIT Press.

Pottier, L. (1998). Dynamical spatialization of sound. HOLOPHON: a graphic and algorithmic editor for Σ1. *In Proceedings of the International Conference on Digital Audio Effects (DAFx'98)*, Barcelona, Spain.

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-Centre Pompidou.

Puckette, M. (1988). The Patcher. *In Proceedings of the International Computer Music Conference (ICMC'88)*, Cologne, Germany.

Puckette, M. (1991). Combining Event and Signal Processing in the MAX Graphical Programming Environment. *Computer Music Journal*, 15(3):68–77.

Puckette, M. (1996). Pure Data: Another Integrated Computer Music Environment. *In Proceedings of the Second Intercollege Computer Music Concerts*, Tachikawa, Japan.

Pulkki, V. (1997). Virtual Sound Source Positioning Using Vector Base Amplitude Panning. *Journal of the Audio Engineering Society*, 45(6).

Rasure, J. R. et Williams, C. S. (1991). An Integrated Data Flow Visual Language and software Development Environment. *Journal of Visual Languages and Computing*, 2(3).

Riotte, A. et Mesnage, M. (2006). *Formalismes et modèles musicaux. 2 Volumes.* Editions Delatour / Ircam.

Risset, J.-C. (1977). Musique, un calcul secret ? *Critique*, 359.

Risset, J.-C. (1993). Synthèse et materiau musical. *In Les cahiers de l'Ircam (2) – La synthèse sonore*. Ircam – Centre Georges Pompidou.

Rodet, X. (1984). Time-domain Formant-wave Function Synthesis. *Computer Music Journal*, 8(3):32–48.

Rodet, X., Barrière, J.-B., Cointe, P. et Potard, Y. (1982). The CHANT Project: Modelization and Production, an Environment for Composers Including the FORMES Language for Describing and Controlling Sound and Musical Processes. *In Proceedings of the International Computer Music Conference (ICMC'82)*, Venice, Italy.

Rodet, X. et Cointe, P. (1984). Formes: Compostion and Scheduling of Processes. *Computer Music Journal*, 8(3).

Rodet, X. et Lefevre, A. (1997). Diphone. *In Actes des Journées d'Informatique Musicale (JIM'97)*, Lyon, France.

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).

Rolim, A. L., Jefferson, B. et Guigue, D. (2005). SOAL: Ferramenta para análise musical no ambiente Open Music. *In Proceedings of the 10th Brazilian Symposium on Computer Music (SBCM'05)*, Belo Horizonte, Brasil.

Sandred, O. (2000). OMRC 1.1: A library for controling rhythm by constraints. Rapport technique, IRCAM, Paris, France.

Sarria, G. et Diago, J. (2003). OpenMusic for Linux and MacOS X. Rapport technique, IRCAM.

Schacher, J. C. et Kocher, P. (2006). Ambisonics Spatialization Tools for Max/MSP. *In Proceedings of the International Computer Music Conference (ICMC'06)*, New Orleans, USA.

Schaeffer, P. (1952). *A la recherche de la musique concrète*. Seuil, Paris.

Schnell, N., Borghesi, R., Schwarz, D., Bevilacqua, F. et Muller, R. (2005). FTM – Complex Data Structures for Max. *In Proceedings of the International Computer Music Conference (ICMC'05)*, Barcelona, Spain.

Schnell, N., Röbel, A., Schwarz, D., Peeters, G. et Borghesi, R. (2009). MuBu & Friends – Assembling Tools for Content Based Real-Time Interactive Audio Processing in Max/MSP. *In Proceedings of the International Computer Music Conference (ICMC'09)*, Montreal, Canada.

Schumacher, M. et Wanderley, M. (2017). Integrating gesture data in computer-aided composition: A framework for representation, processing and mapping. *Journal of New Music Research*, 46(1).

Schwarz, D. et Wright, M. (2000). Extensions and Applications of the SDIF Sound Description Interchange Format. *In Proceedings of the International Computer Music Conference (ICMC'00)*, Berlin, Germany.

Sorensen, A. (2005). Impromptu: An Interactive Programming Environment for Composition and Performance. *In Proceedings of the Australasian Computer Music Conference*, Brisbane, Australia.

Steele, G. L. (1990). *Common Lisp, the Language. Second edition*. Digital Press.

Stroppa, M. (2000). Paradigms for the high level musical control of digital signal processing. *In Proceedings of the International conference on Digital Audio Effects (DAFx'00)*, Verona, Italia.

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 (JIM'02)*, Marseille, France.

Taube, H. (1991). Common Music: A Music Composition Language in Common Lisp and CLOS. *Computer Music Journal*, 15(2).

Todoroff, T., Traube, C. et Ledent, J.-M. (1997). NeXT-STEP Graphical Interfaces to Control Sound Processing and Spatialization Instruments. *In Proceedings of the International Computer Music Conference (ICMC'97)*, Thessaloniki, Greece.

Topper, D., Burtner, M. et Serafin, S. (2002). Spatio-Operational Spectral (S.O.S) Synthesis. *In Proceedings of the International Conference on Digital Audio Effects (DAFx'02)*, Hamburg, Germany.

Toro, M., Rueda, C., Agon, C. et Assayag, G. (2016). Gelisp: A framework to represent musical constraint satisfaction problems and search strategies. *Journal of Theoretical & Applied Information Technology*, 86(2).

Truchet, C. et Assayag, G., éditeurs (2011). *Constraint Programming in Music*. Wiley-ISTE.

Truchet, C., Assayag, G. et Codognet, P. (2003). OMClouds, petits nuages de contraintes dans OpenMusic. *In Actes des Journées d'Informatique Musicale (JIM'03)*, Montbeliard, France.

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-Centre Pompidou.

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-Centre Pompidou.

Van Cauwelaert, S., Gutiérrez Sabogal, G. A. et Van Roy, P. (2012). A new approach for constraint programming in music using relation domains. *In Proceedings of the International Computer Music Conference (ICMC'12)*, Ljubljana, Slovenia.

Vose, M. G. et Williams, G. (1986). Labview: Laboratory virtual instrument engineering workbench. *Byte*, 11.

Wan, Z., Taha, W. et Hudak, P. (2001). Event-Driven FRP. *In International Conference on Functional Programming (ICFP'01)*, Florence, Italy.

Wanderley, M., éditeur (2002). *Mapping Strategies in Real-time Computer Music*. Special Issue of *Organised Sound* 7(2). Cambridge University Press.

Wang, G. et Cook, P. R. (2004). On-the-fly Programming: Using Code as an Expressive Musical Instrument. *In Proceedings of the International Conference on New Interfaces for Musical Expression (NIME'04)*, Hamamatsu, Japan.

Wang, G., Cook, P. R. et Salazar, S. (2015). Chuck: A strongly timed computer music language. *Computer Music Journal*, 39(4).

Whitley, K. N. (1997). Visual Programming Languages and the Empirical Evidence For and Against. *Journal of Visual Languages and Computing*, 8(1).

Whitley, K. N. et Blackwell, A. F. (1997). Visual Programming: The Outlook from Academia and Industry. *In Proceedings of the 7th Workshop on Empirical Studies of Programmers*, Alexandria, USA.

Wilson, S. (2008). Spatial Swarm Granulation. *In Proceedings of the International Computer Music Conference (ICMC'08)*, Belfast, Ireland.

Wright, M. (2005). Open Sound Control: an enabling technology for musical networking. *Organised Sound*, 10(3).

Xenakis, I. (1971). *Formalized Music: Thought and Mathematics in Composition*. Indiana University Press.

Young, M., Argiro, D. et Kubica, S. (1995). Cantata: Visual programming environment for the khoros system. *Computer Graphics*, 29.