omChroma : vers une formalisation compositionnelle des processus de synthèse sonore

Marco Stroppa‡, Serge Lemouton, Carlos Agon
Ircam, Centre Georges Pompidou
‡Hochschule für Musik und Darstellende Kunst - Stuttgart
{marco, lemouton, agon}@ircam.fr

Résumé

Chroma est l'environnement d'aide à la composition et de contrôle de la synthèse développé par Marco Stroppa depuis 1980 pour la réalisation de ses pièces.

Nous décrivons dans cet article l'intégration et les nouvelles extensions du système Chroma dans l'environnement OpenMusic. Nos principaux buts sont : la mise à disposition de l'environnement à d'autres compositeurs et la réalisation d'un système souple qui puisse servir de cadre de travail pour de futures recherches dans le domaine de la composition du son.

1. Introduction

L'un des apports les plus radicaux que l'informatique offre à la musique est la réunion sous un seul paradigme des univers macro- et micro-compositionnels. Dans cette optique, le traitement du son nous permet de penser le son comme le résultat d'un processus logique. Cependant, de nos jours les logiciels d'aide à la composition sont plutôt spécialisés en l'un de ces deux univers. En effet, la division entre CAO et traitement sonore reste une question d'actualité. En général, les travaux mettant en relation CAO et traitement du signal (DSP) se contentent de générer des paramètres de synthèse, tout en gardant bien définie la frontière entre les deux domaines, on parle donc de contrôle de la synthèse [1]. Il existe une autre famille d'environnements [2], [3], qui a pour but l'intégration sous un même programme des aspects symboliques et ceux liés à la synthèse.

Nous décrivons dans cet article l'intégration et l'implémentation du système *Chroma* dans l'environnement OpenMusic [4]. *Chroma* est le nom de l'environnement informatique que Marco Stroppa s'est forgé tout au long de son activité de compositeur et qu'il continue à utiliser et à enrichir. Système personnel en constante expansion, *Chroma* est exemplaire à au moins deux points de vue ; tout d'abord par sa longévité (plus de 20 ans) et ensuite par son utilité puisqu'il a servi lors de la composition de la plus grande partie des œuvres de son concepteur.

Les prémisses de *Chroma* datent de 1980 au Centro di Sonologia Computazionale, sous la forme de sous-programmes en Fortran pour *MusicV*, ensuite, une première généralisation a été faite en *LeLisp* au MIT et à l'IRCAM entre 1984 et 1990. Tout le système a été réécrit en *CLOS* par Serge Lemouton en 1996/97. Un historique plus complet se trouve en appendice de [5]. Nous travaillons actuellement à la mise à disposition de ces outils dans le cadre d'une librairie d'*OpenMusic*, ce qui pose des problèmes intéressants de généralisation et d'explicitation d'un système très riche mais idiosyncrasique. L'idée principale dans omChroma consiste à affaiblir la dichotomie entre un son vu comme le résultat d'un processus de synthèse ou comme une entité logique musicalement relevante. *Chroma* n'est pas un système réunissant CAO et DSP, nous sommes plutôt intéressés dans l'effacement ou le déplacement de la frontière séparant ces deux paradigmes. Chroma n'implémente pas des synthétiseurs ni ne prétend définir de nouvelles méthodes de synthèse. Notre principal objectif est la définition et la mise à disposition des diverses abstractions en relation avec des structures musicales de haut niveau (i.e. rythmes, accords, polyphonies, etc). Ces abstractions doivent aussi servir comme interface pour la génération des paramètres de synthèse.

Le design d'un tel environnement de composition implique une réflexion sur un certain nombre de difficultés. Tout d'abord, la plupart des méthodes de synthèse numérique nécessite la gestion de très nombreux paramètres. D'où le besoin d'outils automatiques de génération de ces données, qu'il serait trop fastidieux de spécifier "à la main". Une autre difficulté inhérente à l'utilisation musicale des synthétiseurs est la difficulté d'obtenir un résultat "vivant", auditivement aussi intéressant que le son produit par un instrument de musique ou une voix humaine. On cherche à éviter des sons qui révèlent trop rapidement leur nature "synthétique", et pour cela il faut utiliser soit des synthétiseurs très élaborés, soit un contrôle très fins des paramètres. Le matériau premier de la musique étant le temps, nous devons disposer de fonctionnalités permettant de le manipuler. C'est ce qu'attend un compositeur de musique qui par sa pratique de la composition est habitué à ce travail sur le temps, et à le penser selon différents modes (cf. *en-temps* et *hors-*

temps chez Xenakis [6]). On souhaite également trouver une façon de s'abstraire du synthétiseur : chaque méthode de synthèse implique des paramètres différents, alors que l'on souhaiterait faire jouer la même "partition" par plusieurs instruments ou instrumentistes (de synthèse) différents. De plus, les synthétiseurs eux-mêmes ont une durée de vie limitée, bien plus limitée que celle des instruments de musique réels.
Dans cet article, nous décrivons divers outils et techniques adaptées aux problématiques ici mentionnées. La section 2 fait une description globale de l'architecture de Chroma et la manière dont nous intégrons cet environnement dans OpenMusic. Nous décrivons dans la section 3 la principale abstraction de synthèse. La section 4 expose notre idée de synthétiseur virtuel. Nous montrerons dans la section 5 les possibilités de manipulation dans le temps des abstractions de synthèses. Nous finissons cet article avec quelques conclusions, ainsi qu'avec les travaux à venir.

2. Description globale.

La Figure 1 donne une vue globale de l'architecture de l'environnement. Deux notions sont à la base d'omChroma : la bibliothèque d'objets de synthèse (section 2) et le synthétiseur virtuel (section 3).

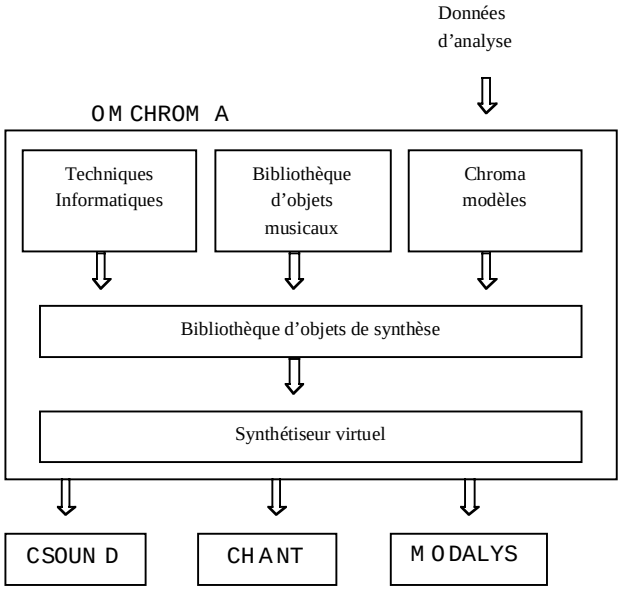

Figure 1. Chroma Architecture.

Les objets de synthèse peuvent être générés soit par le compositeur, soit à partir de structures plus ou moins constituées (par exemple, des algorithmes ou des bases de données). Etant donné que *Chroma* est écrit en Common Lisp, ces objets peuvent être construits par n'importe quelle technique informatique (moyennant l'écriture ou l'utilisation de la fonction correspondante).
Différentes classes d'objets exprimant des caractéristiques du son ou de la musique se sont sédimentées dans *Chroma*, au cours de son évolution. Ces classes sont un reflet de l'expertise du compositeur ainsi que des catégories qu'il utilise dans sa pratique de la composition musicale. Parmi ces classes, on trouve par exemple, des fonctions temporelles ("FUN") de type *Break Point Functions*, avec des méthodes permettant par exemple de faire de la réduction de données. Il existe également une base de données d'enveloppes (VE : "Enveloppes Virtuelles"), ces enveloppes sont des formes de bases utilisées en tant que répertoire d'évolution pour les paramètres de contrôle. Parmi les objets plus spécifiquement musicaux, il existe de nombreuses classes destinées à la représentation de "Structures Verticales de Hauteurs" (VPS), classes permettant de traiter de façon homogène aussi bien des accords musicaux que des spectres sonores.

La troisième façon de remplir des objets de synthèse utilise le concept de modélisation. L'emploi que nous faisons ici du mot "modèle" est à mi-chemin entre le sens scientifique et le sens artistique du terme, puisque le modèle procède à la fois d'un choix et d'une analyse. "L'abstraction est pour le compositeur une étape précédant la reconstitution d'un monde de phénomènes à la fois analogue et différent."[7]. Un modèle au sens *Chroma* est une structure de données extraites d'analyses d'un fichier son (un extrait musical) destinée à être utilisée en tant que réservoir de formes et de paramètres applicables à un synthétiseur sonore quelconque. À la limite, le modèle doit être capable de resynthétiser une version très proche du son original, ou bien des versions très éloignées voire même totalement méconnaissables. Dans ce cas, nous ne sommes plus dans le

cadre d'une activité de modélisation scientifique, puisque l'activité créatrice cherche à dépasser la simple reproduction pour arriver non à une explication mais à une générativité du modèle. Il en est donc une "représentation potentielle", plus ductile que le son lui-même.

Concrètement, la première étape de la modélisation consiste en une segmentation de l'extrait sonore choisi. Les structures temporelles résultant de cette segmentation sont constituées de régions (définies par leurs frontières : un début et une fin) et sont hiérarchisées. La segmentation se fait à plusieurs niveaux, le niveau le plus bas étant celui où sont définis des événements élémentaires (Figure 2). Cela permet une analyse de l'extrait musical où sont préservés les différents niveaux de structuration tels que (de bas en haut) : période fondamentale, ornementation, note, mélisme, motif, phrase. Les modèles permettent d'avoir différentes "vues", différentes focalisations sur un même fragment musical.

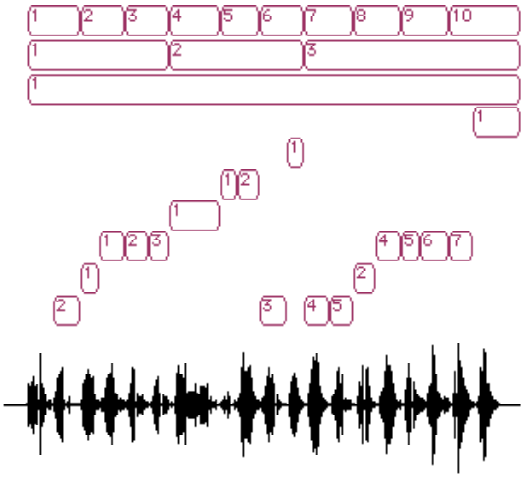

Figure 2. Exemple de segmentation d'un son

Concrètement, un modèle est constitué d'une liste de structures verticales (VPS, voir plus haut) accompagné de fonctions temporelles (FUN, id) représentant les variations en fréquence et en amplitude de chacune de ces composantes. Il intègre également une structure rythmique hiérarchisée.

Un "modèle" partant de l'analyse d'un son réel permet de disposer de données intéressantes par ce qu'elles contiennent d'aléatoire, d'*organicité*. Ce qui permet d'obtenir facilement des sons d'apparence moins synthétique et d'éviter ainsi une des difficultés majeures de la synthèse sonore numérique.
Le modèle sonore est indépendant à la fois de la méthode et du système de synthèse utilisé. Les données qu'il contient (musicales et de contrôle) sont suffisamment abstraites pour pouvoir être appliqués aussi bien à un synthétiseur FM en CSound qu'à un synthétiseur Chant en MSP, ou synthèse par modèles physiques en Modalys.

3. Bibliothèque d'objets de synthèse.

OpenMusic fournit une bibliothèque d'objets musicaux, c'est-à-dire un ensemble de classes et de méthodes pour la représentation et le traitement de l'écriture musicale (i .e. notes, accords, voix, polyphonies). Au même titre, nous décrirons dans dette section la bibliothèque d'objets de synthèse, qui a à voir avec des entités de haut niveau pour le processus de synthèse de son. À la base des objets de synthèse, nous avons un représentation matricielle, où lignes et colonnes de la matrice sont associés aux paramètres de synthèse. Nous avons implémenté une classe abstraite appelée *array* avec un *slot* particulier *numcol* spécifiant le nombre de colonnes. La classe *array* n'est pas utilisable directement, elle doit être sous-classée en ajoutant des *slots* qui correspondront aux paramètres de synthèse et qui ajouteront des lignes à la matrice. Dans la Figure 3, nous voyons la création d'une matrice à 20 colonnes et 5 lignes (*numcol* = 20 et 5 *slots* ajoutés). Si les valeurs des paramètres sont des nombres les lignes pourront être représentées sous forme de courbe, comme il est montré dans la Figure 3. Une enveloppe sera clairement une matrice avec une seule ligne.

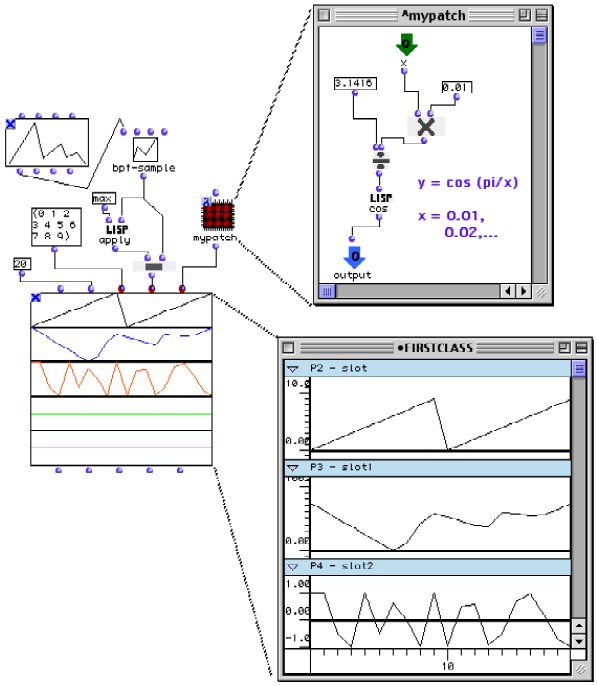

Figure 3. Une instance de la classe *array*.

Il n'est pas nécessaire de remplir toutes les lignes, dans notre exemple, seulement 3 des 5 *slots* ont été fournis. Pour les deux autres, on utilise les valeurs par défaut données lors de la définition de la classe, les courbes en question seront des constantes. Une particularité de la classe *array* est que les valeurs des *slots* peuvent être données sous diverses formes. Dans la Figure 3, les trois entrées de paramètres se font de gauche à droite. Ce sont successivement :
-une liste d'entiers, laquelle est répétée cycliquement jusqu'à atteindre le nombre de colonnes
-le résultat d'un algorithme visuel (patch *OpenMusic*)
-une lambda expression visuelle (montrée dans la fenêtre *mypatch*, y = cos(Pi/x)).
Quand les valeurs des *slots* sont des fonctions, la matrice ne garde pas les valeurs de l'application de la fonction, mais elles seront calculés postérieurement lorsque cela sera nécessaire. Cette technique est appelée évaluation paresseuse. Les *Arrays* peuvent être construits à partir d'objets musicaux. Pour cela il faut définir un algorithme d'interprétation, comme montré dans la Figure 4.

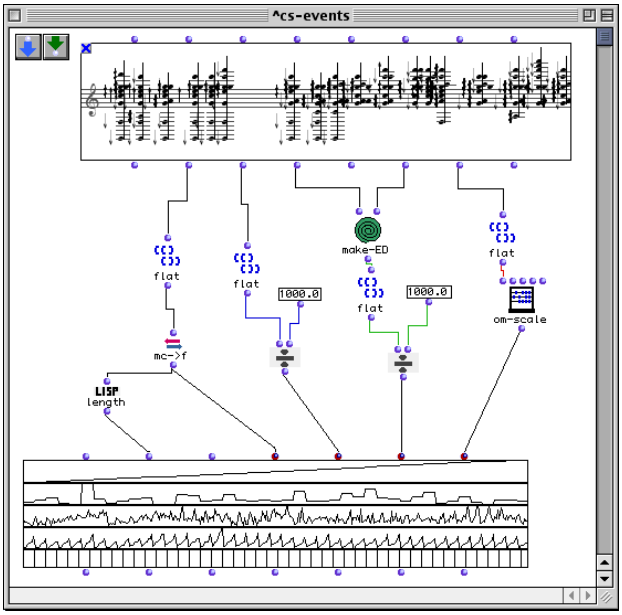

Figure 4. Création d'un *array* à partir d'un objet musical.

4. Synthétiseur virtuel.

Tout modèle de génération sonore a besoin de données de contrôle, qui dépendent du synthétiseur utilisé. Cependant plusieurs paramètres de contrôle peuvent être conceptuellement les mêmes, indépendamment de leur implémentation. Un vibrato, par exemple, sera toujours utilisé comme un vibrato peu importe la façon dont il est codé.

Les *arrays* peuvent être groupés en sous-classes abstraites qui, grâce au polymorphisme, peuvent être interprétées par un synthétiseur donné. Ainsi des données similaires n'auront pas besoin de structures différentes pour être traités par différents synthétiseurs. Les algorithmes de production de données sont ainsi isolés du moteur de synthèse. Cette étape d'interprétation relève de ce que nous appelons un " synthétiseur virtuel ". Le polymorphisme est concentré dans la fonction générique *synthesize*. Cette fonction est spécialisée par le nom de synthétiseur désiré, elle prend une liste d'*arrays* en filtrant toutes les instances qui ne sont pas des sous-classes de la classe abstraite associée au synthétiseur. Cette fonction traduit les *arrays* dans un format approprié : *orc* et *score* fichiers pour *Csound* , un fichier *SDIF* pour *Chant* , paquets *OSC* pour *Max/msp* ou *jMax*, etc. L'adjonction de nouveaux synthétiseur au système sera réduite à la définition de nouvelles méthodes pour *synthesize*. La Figure 5 montre un exemple de la notion de synthétiseur virtuel. Le matériau de base vient d'une analyse d'un son de cymbale. Le côté gauche de la figure montre une synthèse utilisant *Chant*. Le côté droit renvoie les mêmes données d'analyse à *Csound*. Les résultats des deux synthèses sont identiques.

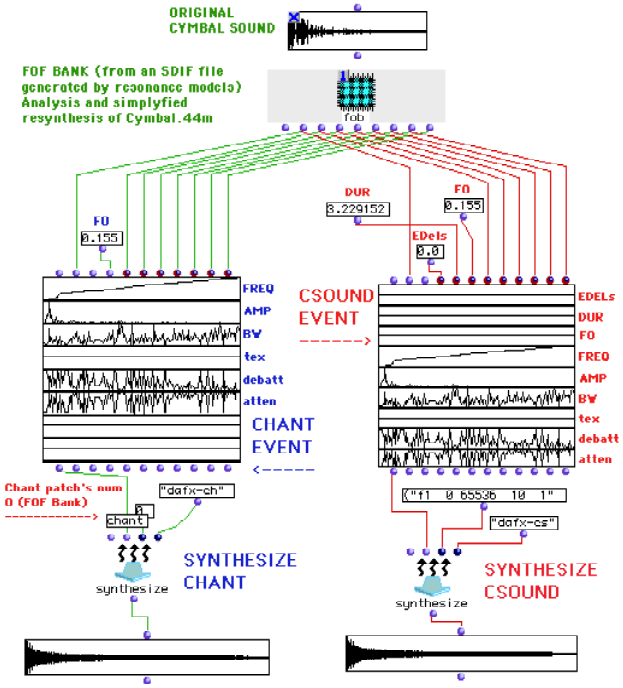

Figure 5. Création d'un *array* à partir d'un objet musical.

Cependant, il est peu probable qu'un compositeur pense un processus de synthèse seulement en fonction d'un son unique. D'après notre expérience, un compositeur verra ce processus plutôt comme un modèle qui génère une famille de sons pouvant aussi être étendue progressivement. De ce point de vue, la notion de son est remplacée par celle de potentiel sonore [8]. Ce dernier a à voir avec l'idée d'une classe d'équivalence de tous les résultats possibles d'un modèle de synthèse. Ainsi, les deux sons de la Figure 5 ne sont pas équivalents bien qu'ils produisent le même résultat. Nous devons voir alors le son pas uniquement comme un résultat, mais aussi en prenant en compte la façon (l'algorithme) dont il a été produit et ses possibles extensions. Notre but sera alors de fournir une représentation (informatique et graphique) qui rende compte en même temps du contrôle de la synthèse et de sa pertinence pour la composition.

5. Maquettes : événements en temps.

Les compositeurs habitués aux techniques informatiques ont accès à des outils leur permettant de décrire, de visualiser, d'écouter et de manipuler des structures musicales qui reflètent leur manière de penser. Cependant, au point de vue temporel ces efforts n'ont pas été poussés aussi loin que pour d'autres dimensions telles que la hauteur, par exemple. Au niveau de la synthèse de son, il existe peu d'outils qui

permettent d'avoir un contrôle du temps autre que le temps chronologique dans lequel le résultat doit se déployer. La notion de maquette [4] dans OpenMusic a été conçue afin de donner un outil pour la manipulation d'objets musicaux en temps et hors temps. Une maquette est une sorte de surface dont une dimension est le temps. Les objets musicaux, tels que notes, accords, polyphonies, peuvent être placés dans une maquette, ainsi sont créées, par superposition, concaténation et relations fonctionnelles, différents niveaux d'articulation entre objets élémentaires. Nous avons étendu la class *array* de façon à pouvoir placer aussi ses instances dans une maquette. L'exemple de la Figure 6 montre les nouvelles possibilités de cette approche.

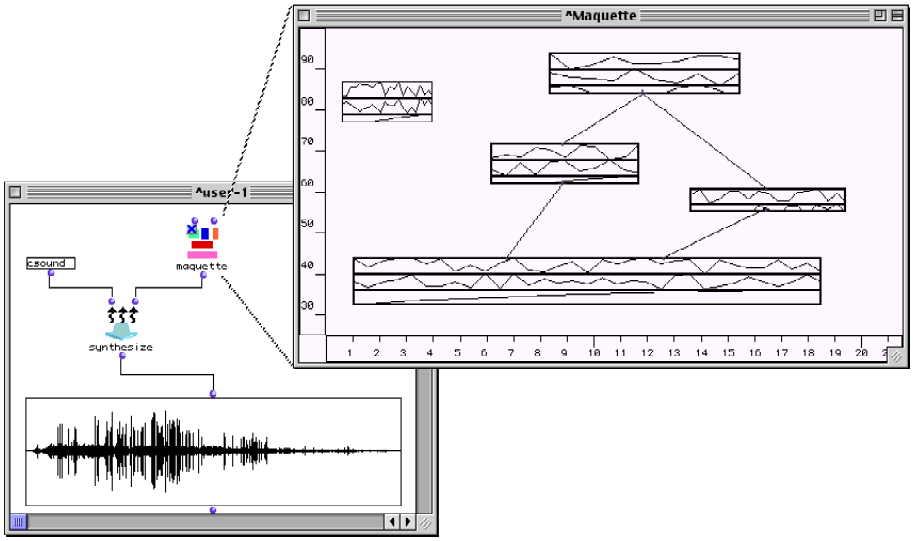

Figure 6. Synthèse d'*arrays* dans une maquette.

Dans la Figure 6 nous avons placé dans une maquette diverses instances d'*arrays* représentant des partiels pour une synthèse additive. La durée de chaque partiel est exprimée par la taille en abscisse de chaque boîte. La taille en ordonnée détermine un facteur multiplicatif pour l'enveloppe d'amplitude. La position en *x* détermine un temps d'attaque tandis qu'en *y* elle donne une valeur relative pour la fréquence. Les différents paramètres de chaque boîte sont visualisés sous formes d'enveloppes. Nous pouvons remarquer aussi qu'il existe des connexions entre certaines boîtes. Un lien entre deux boîtes exprime une relation fonctionnelle permettant de créer une dépendance causale entre elles. Ainsi, l'amplitude d'une boîte peut être calculée en prenant comme entrée l'amplitude d'une autre et en appliquant une certaine opération (une transposition par exemple). La maquette est connectée directement à la fonction *synthesize* spécialisée par le synthétiseur *Csound*. Enfin, *synthesize* est relié à un objet de type *sound* permettant de visualiser et d'écouter le résultat de la synthèse.

La maquette devient ainsi un éditeur pour la visualisation d'un processus de synthèse à différents niveaux d'information :

- Le niveau statique de l'architecture du son : donné par le nombre de boîtes, leur position, et les possibles imbrications.
- Le niveau dynamique de l'architecture du son : explicité par les différentes relations fonctionnelles entre les boîtes.
- Le niveau syntaxique : ce qui concerne les différents calculs, à l'intérieur de chaque boîte, qui génèrent les *arrays* visualisés comme résultat final.
- Le niveau du matériau : consistant en les données de base proposées par le compositeur.

Ces niveaux d'information sont interconnectés entre eux. La maquette permet aussi le contrôle des différentes niveaux mentionnés. Cela a pour conséquence une source d'expérimentation pour la synthèse :

- Les boîtes peuvent être déplacées ou étirées.
- Les relations fonctionnelles peuvent changer sans modifier le niveau statique.
- Les algorithmes de production des *arrays* peuvent être modifiés (i.e . on peut changer l'intervalle de transposition pour une enveloppe qui dépend de l'enveloppe d'une autre boîte)
- Les matériaux peuvent être modifiés. Un changement uniquement à ce niveau peut être vu comme une sorte de modélisation du son où le matériau devient un paramètre abstrait.

6. Conclusion.

Dans cet article, nous avons présenté un système d'aide à la composition qui intègre la synthèse dans le processus de composition musicale.
La mise à disposition de *Chroma* pour d'autres compositeurs nous permettra d'évaluer la pertinence des concepts sous-jacents.

7. Références

[1] H. Taube. *Common Music Reference Manual*. University of Illinois, Illinois 2002.
[2] Eckel, G., González-Arroyo, R. *Musically Salient Control Abstractions for Sound Synthesis.* Proceedings of the 1994 International Computer Music Conference Aarhus,1994.
[3] Rodet X. et Cointe P. *FORMES Composition et ordonnancement de processus*. Rapports de recherche No 36, Ircam, Paris, 1985.
[4] Assayag G., Agon C. *OpenMusic Architecture*. Proceedings of the ICMC 96, Hong Kong, 1996.
[5] Stroppa M. *OmChroma 1.0 Manual & Tutorial*, Ircam, Paris, 2000
[6] Xenakis I. *Musiques formelles*. Stock Musique, Paris, 1981.
[7] Mâche François-Bernard : *Musique, mythe, nature*, Klincksieck, Paris, 1983.
[8] Stroppa M. *Paradigms for the high-level musical control of digital signal processing*. Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00), Verona, Italy, 2000.