Open Music + Music Space = Open Space

Olivier Delerue
Sony CSL & Ircam
delerue@csl.sony.fr

Carlos Agon
Ircam
Carlos.Agon@ircam.fr

Résumé :

L'exploration en temps réel d'espaces de paramètres arbitrairement grands suppose 1/ un moyen ouvert pour définir ces espaces et 2/ une interface adaptée pour les exploiter. C'est à ce problème que nous tentons d'apporter une réponse en proposant un pont de communication entre le système de contrôle temps réel MusicSpace *et le langage visuel de programmation musicale* OpenMusic. *Grâce à cette connexion* MusicSpace *permet de contrôler dynamiquement et temps-réel un ensemble d'objets musicaux définis en* OpenMusic. *La description de cette connexion - baptisée* OpenSpace *- est illustrée par un exemple précis issu du contrôle de la synthèse.*

1.Introduction

Le système MusicSpace a été conçu initialement pour le contrôle en temps réel de la spatialisation de sources sonores sous contraintes : au sein d'une interface graphique, l'utilisateur a la possibilité de définir la position initiale de chacune des sources sonores à spatialiser, ainsi qu'un ensemble de contraintes sur ces sources sonores qui établissent des règles de comportement entre les différentes sources. L'interface représente graphiquement l'ensemble des sources sonores définies ainsi que, sous forme d'icône, un objet représentant l'utilisateur. Celui ci a alors la possibilité de déplacer chaque objet (dans un plan azimutal) et de percevoir en temps réel les modifications correspondantes de la spatialisation des sources [Delerue, 1998]. Les contraintes permettent alors d'établir des relations entre les sources sonores de manière à préserver une certaine qualité du mixage.

Cette interface présente des possibilités importantes puisqu'elle n'est pas liée à la nature des objets qu'elle contrôle : en particulier, elle n'est pas réduite au contrôle de la spatialisation qui n'en est qu'une application parmi d'autres. De même, l'ensemble des contraintes conçues pour l'application à la spatialisation du son appartient au type de représentation graphique et non à l'application elle-même, ce qui, dans un contexte différent, leur permet de conserver une interprétation valable.

Nous présentons ici une autre utilisation possible de MusicSpace : le contrôle de la synthèse. A la différence de la spatialisation, celle ci requiert un nombre plus important de paramètres (et donc d'objets graphiques) dont la génération manuelle est relativement fastidieuse. De plus, une caractéristique essentielle de ces espaces de paramètres est que leur définition (autant la définition même de ces paramètres que celle des relations qui les unissent) possède une forte dominante algorithmique : la conception d'un cas de figure particulier s'exprime le plus souvent par exemple, par l'intermédiaire de boucles. C'est donc très naturellement que s'adapte un environnement de programmation visuelle pour répondre à ce type de besoin.

OpenMusic [Agon, 1998] va répondre directement à ces deux problèmes en proposant la modélisation d'un tel espace sous la forme d'un patch : un nombre arbitrairement grand de paramètres peut alors être défini et organisé de manière algorithmique. Nous allons donc dans cet article proposer un moyen de faire communiquer ces deux systèmes. Cette communication repose sur l'utilisation systématique du système d'exploitation *MidiShare* [Fober, 1998], qui est utilisé pour l'implémentation des deux systèmes.

2.Contrôle de la synthèse

Le problème général que pose le contrôle de la synthèse revient d'une part à agir simultanément sur un ensemble arbitrairement grand de paramètres de contrôle, et d'autre part à faire varier ces paramètres de manière cohérente les uns par rapport aux autres : ceux-ci ne sont généralement pas totalement indépendants les uns des autres et donc la variation de l'un d'entre eux impose éventuellement celle de certains autres.

C'est à cette question que nous tentons de répondre en proposant, par l'association d'OpenMusic à MusicSpace, un environnement ouvert permettant de spécifier de manière algorithmique d'une part, et d'interpréter en temps-réel d'autre part de tels espaces de contrôle. De plus, cette utilisation combinée des deux logiciels tire directement parti des performances temporelles de chacun d'eux pour proposer un contrôle dynamique, c'est à dire pouvant varier au cours du temps : cette spécificité sera présentée en section 4.3.

Dans l'exemple que nous présentons, celui de la synthèse additive, l'ensemble des paramètres est homogène et représente l'intensité d'un nombre arbitraire de partiels (i.e. oscillateurs sinusoïdaux simples). De tels objets sont crées dynamiquement à l'aide d'OpenMusic. La Figure 1 en donne une illustration, où trente objets partiels sont crées et répartis géographiquement en quatre groupes à partir des notes de quatre accords différents. Un patch permet donc, à partir d'un accord, de générer automatiquement les partiels correspondants aux fréquences harmoniques des notes de cet accord et de les regrouper dans une région particulière de la fenêtre.

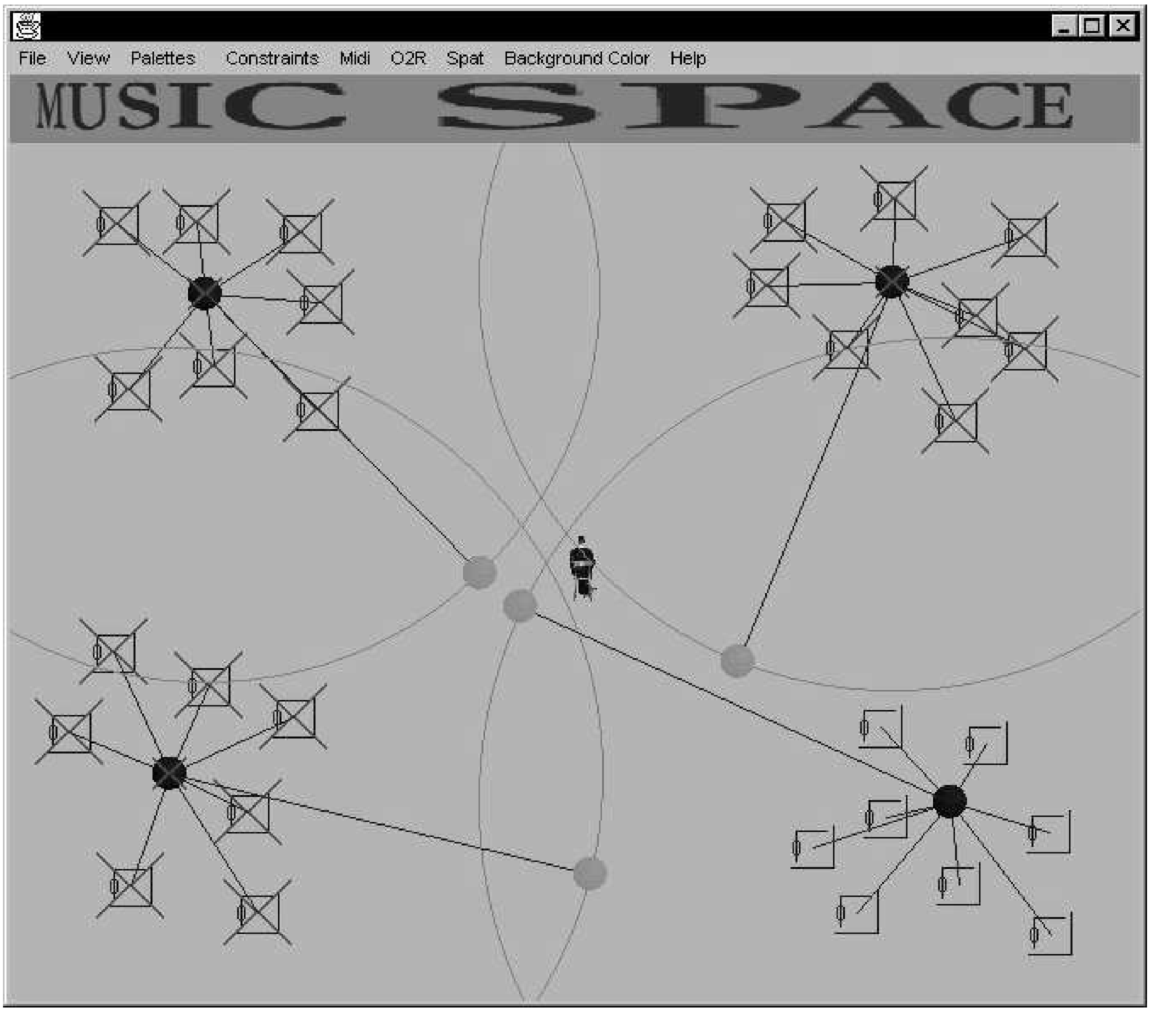

Figure 1 : l'intensité de chaque partiel est fonction de la distance du partiel à l'objet représentant l'utilisateur

Un ensemble de contraintes a également été conçu relativement à notre exemple. La Figure 1 met en évidence deux exemples de contraintes (représentées par des boules sombres et claires, reliées par des segments aux objets contraints). Les contraintes «groupe de mute», représentées par des boules sombres, sont des contraintes n-aires, qui ne font que propager une information de «mute» aux objets contraints : lorsque cette contrainte est mutée (une croix rouge est dessinée dans ce cas), elle propage la commande de «mute» aux objets contraints ce qui, dans notre exemple, a pour effet de les rendre silencieux. Une contrainte de proximité, représentée par une boule claire, est utilisée pour chacun des groupes de mute : cette contrainte est unaire, et dessine une circonférence autour de son objet contraint, qui délimite une zone de validité. Ainsi, lorsque l'auditeur sort de cette zone, la contrainte mute son objet contraint. Inversement, lorsque l'auditeur entre dans cette zone, l'objet contraint est dé-muté.

En résumé, la figure 1 représente un ensemble de 30 objets «ostinato», divisé en quatre sous-groupes. Pour chacun de ces sous-groupes, deux contraintes (groupe de mute et proximité) ont été définies de manière à ce que l'espace soit divisé en régions. Lorsque l'auditeur passe d'une région à une autre, les sous-groupes correspondants sont activés ou désactivés. L'effet permet en l'occurrence de séparer géographiquement des partiels dissonants. Nous verrons dans la section 4.2 comment ces partiels sont générés en OpenMusic.

D'autres contraintes ont été réalisées comme, par exemple, des filtres passe-bande et harmonique. Une telle contrainte peut être appliquée sur la totalité des partiels et permet de n'en déplacer que ceux qui appartiennent à la zone d'influence du filtre. Puisque nos objets «ostinatos» sont également des objets rythmiques, il est possible également de concevoir des filtres rythmiques correspondants.

2.1 Schéma général

Le schéma général consiste à connecter la sortie d'OpenMusic à l'entrée de MusicSpace de manière à permettre la description et génération de l'espace de contrôle, puis la sortie de MusicSpace vers un moteur de synthèse temps-réel qui saura appliquer les modifications des paramètres de synthèse générées par MusicSpace

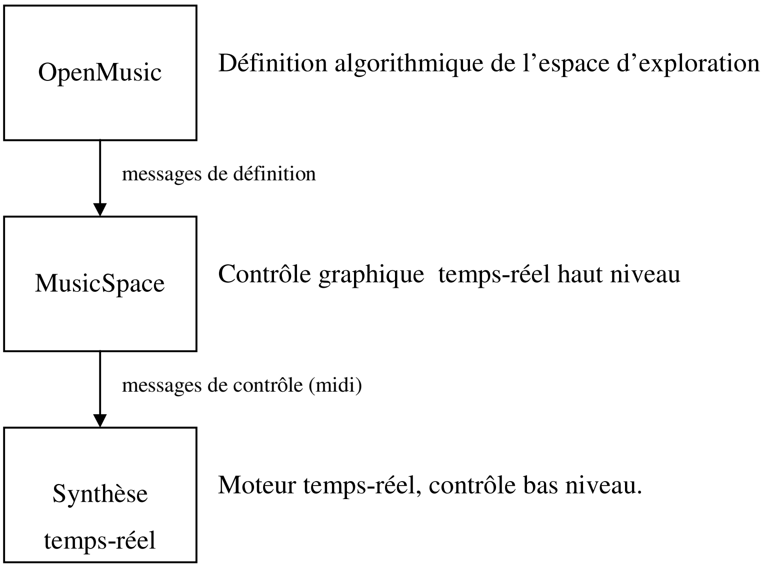

Figure 2 : schéma général

Dans notre exemple, des objets particuliers (objets «ostinato»), simples, ont été conçus de manière à pouvoir effectuer les tests préliminaires plus simplement : ces objets sont munis d'une hauteur Midi, d'un numéro de canal et d'une période et répètent inlassablement leur note toutes les unités de temps correspondant à la période. Ces objets remplacent donc la partie Synthèse de la Figure 2.

3. Protocole de communication

La communication entre applications est effectuée via Midishare [Fober 95], une architecture logicielle dédiée aux applications musicales et permettant la collaboration entre applications : Midishare permet en effet des communications entre applications exécutées sur des plates-formes différentes (via les câbles et interfaces Midi) mais aussi au sein d'une même plate-forme, en utilisant la même norme et les mêmes messages [Fober 98].

Deux implémentations différentes de ces communications peuvent être envisagées. Tout d'abord un type d'événement particulier et propre à Midishare (puisqu'il n'est pas prévu par la norme Midi) a été conçu spécialement pour la collaboration entre application : le type «typePrivate». Ce type d'événement permet à l'utilisateur de véhiculer des données entre deux applications. L'avantage d'utiliser ce message est d'assurer des temps de communication optimum, puisque le message (signé par l'application réceptrice) ira directement à son destinataire sans encombrer les ports Midi. En revanche, cette communication ne pourra s'opérer qu'entre des applications exécutées sur un seul ordinateur.

Une deuxième possibilité consiste à utiliser des messages de type «système exclusif ». Ces messages ne sont pas limités en taille mais risquent, en revanche, d'encombrer les ports Midi.

Nous envisageons donc une double implémentation qui permettra de faire face aux différentes conditions d'utilisation : les messages de type «typePrivate» seront utilisés lorsque OpenMusic et MusicSpace sont exécutés sur la même plate-forme, les messages de type «système exclusif» seront utilisés lorsqu'un câble MIDI sépare les deux applications.

3.1 Description des messages

Les messages à transmettre depuis OpenMusic doivent permettre la création, la modification ainsi que la suppression d'un ensemble arbitraire d'objets. A chaque objet crée, un numéro unique, l'identifiant, est attribué par OpenMusic. C'est par ce numéro qu'OpenMusic pourra y faire référence ultérieurement pour une éventuelle modification ou suppression. Chaque message commencera donc par l'ouverture d'une session qui déclarera un numéro d'identifiant. A l'intérieur de cette session viennent les informations proprement dites (pour la création d'un objet par exemple, la déclaration du type de cet objet, l'ajustement de ses divers paramètres tels que sa position sur chaque axe, et la demande d'instanciation). Pour terminer, la session est refermée.

D'autre part, nous tenons à ce que les messages définis dans ce protocole soient compatibles tant avec une communication par système exclusif qu'avec une communication utilisant les événements «typePrivate». Ainsi, de manière générale, un message sera défini par une série de commandes constituées de quatre octets, respectant le format défini dans le tableau de la section 3.2.

Un message exclusif adopte le format générique suivant : F0 xx {nn = données} F7 où F0 correspond au code «début exclusif», F7 au code «fin exclusif». L'indice xx correspond au numéro de code du constructeur. (exemple : Sony = 4C, Kurzweil = 07,…). Un numéro code particulier est retenu : 7D = usage non commercial, destiné aux université et à la recherche. Les messages exclusifs utilisés auront donc la forme : F0 7D {0nnnnnn 0nnnnnn 0nnnnnn 0nnnnnn }+ F7 où {0nnnnnn 0nnnnnn 0nnnnnn 0nnnnnn } représente un message sur 4 octets équivalent à un message de type «typePrivate».

La section 3.3 donne deux exemples de messages.

3.2 Tableau récapitulatif des commandes

1	2	3	4
00 = Début Session	ObjectId MSB	ObjectId LSB	non utilisé
01 = Définir objet	01 = objet Partiel	Non utilisé	non utilisé
	02 = objet contrainte	01 = niveaux ensembles	non utilisé
		02 = niveaux contraires	non utilisé
		12 = groupe de mute	non utilisé
		13 = contrainte de proximité	non utilisé
	03 = objet control change	nn = (0-127) numéro ctrl	non utilisé
	04 = objet position	Non utilisé	non utilisé
	05 = objet ostinato	Non utilisé	non utilisé
02 = ajuster paramètre	01 = position angulaire	Angle MSB	angle LSB
	02 = position radiale	Rayon MSB	rayon LSB
	03 = position X	Position MSB	position LSB
	04 = position Y	Position MSB	position LSB
	05 = fréquence partiel	Fréquence MSB	fréquence LSB
	07 = Amplitude partiel	Amplitude MSB	amplitude LSB
	08 = Canal Midi	nn = (0-15) numéro canal	non utilisé
	09 = Pitch	nn= (0-127) numéro note	non utilisé
	10 = Duration (ms)	Duration MSB	Duration LSB
	11 = Rayon	Rayon MSB	rayon LSB
03 = Contraindre objet	ObjectId MSB	ObjectId LSB	non utilisé
04 = Relaxer objet	ObjectId MSB	ObjectId LSB	non utilisé
05 = Créer instance objet	non utilisé	Non utilisé	non utilisé
06 = Supprimer objet	non utilisé	Non utilisé	non utilisé
07 = Déplacer objet	PositionObjectId MSB	PositionObjectLSB	nn = duration
127 = Fermer session	ObjectId MSB	ObjectId LSB	non utilisé

3.3 Exemples

L'exemple suivant montre la création d'un objet ostinato d'identifiant 375, sur la note Do4, de canal midi 1, dans le quart supérieur gauche de la fenêtre, et qui se répète toutes les 1200 ms :

F7h				Début exclusif
7Dh				Constructeur = universités
0	2	119	x	Ouverture de session, 375 = **2** * 128 + **119**
1	5	x	x	Définition type (5 = ostinato)
2	3	31	32	Ajustement position x, 4000 = **31** * 128 + **32**
2	3	31	32	Ajustement position y, 4000 = **31** * 128 + **32**
2	9	60	x	Ajustement note (DO4 = numéro midi 60).
2	8	0	x	Ajustement canal midi (0 = canal midi 1).
2	10	9	48	Ajustement période : 1200 = **9** * 128 + **48**
5	x	x	x	Instancier l'objet
126	2	119	x	Fermeture de la session
F0h				Fin exclusif

Les 'x' correspondent aux paramètres non utilisés. On pourra, par convention, leur donner la valeur 0.

Si maintenant nous souhaitons contraindre cet objet à l'aide d'une contrainte de proximité, nous construisons une contrainte du type correspondant en ouvrant une nouvelle session (avec l'identifiant 376, par exemple), puis nous déclarons le type de l'objet (contrainte de proximité) et les objets à contraindre. Pour finir, l'objet est instancié et la session refermée. Le message est le suivant :

F7h				Début Exclusif
7Dh				Constructeur = universités
0	2	120	x	Ouverture de session, 376 = **2** * 128 + **120**
1	2	13	x	Définition type (2 = contrainte, 13 = sous type proximité)
3	2	119	x	Contraindre l'objet d'identificateur **2** * 128 + **119** = **375** (c'est à dire l'objet ostinato que l'on vient de définir).
…	…	…	x	Cette ligne peut être répétée si l'on a d'autres objets à contraindre simultanément.
2	11	7	104	Ajustement du rayon 1000 = 7 * 128 + 104
5	x	x	x	Instancier l'objet
126	2	120	x	Fermeture de la session
F0h				Fin exclusif

4. Implémentation en OpenMusic

Un des attraits conséquents de ce projet est la facilité de mettre en œuvre et de concevoir l'espace de contrôle à l'aide d'OpenMusic : seule une petite partie de l'implémentation est propre à MusicSpace et est décrite dans la section 4.1. Le reste de cette implémentation est réalisé naturellement avec les objets et classes déjà existant en OpenMusic : un exemple sera décrit dans la section 4.2. Finalement, nous présenterons dans la section 4.3 l'organisation temporelle des messages de commande à l'aide de l'éditeur de maquettes.

4.1 Hiérarchie de classe MusicSpace en OpenMusic

Cette section décrit la partie propre à MusicSpace, réalisée en OpenMusic : il s'agit d'une hiérarchie de classes adaptée au protocole précédemment défini : chacune des classes étend la classe de base «eventmidi». Toute instance de ces classes construit donc un objet qui pourra être interprété comme un message midi, c'est à dire répondre positivement à la méthode «play» existante.

A la racine des classes définies, une classe mère appelée «musicspaceevent» regroupe tout ce qu'il y a de commun aux messages définis dans notre protocole. En particulier, si la communication a lieu à l'aide de messages exclusifs, celle-ci enrobera les données des commandes «début exclusif », «constructeur 126», et «fin exclusif». De cette classe dérive un ensemble de sous-classes correspondant à chacune des commandes de notre protocole. La Figure 3 présente la hiérarchie de classes qui a été définie.

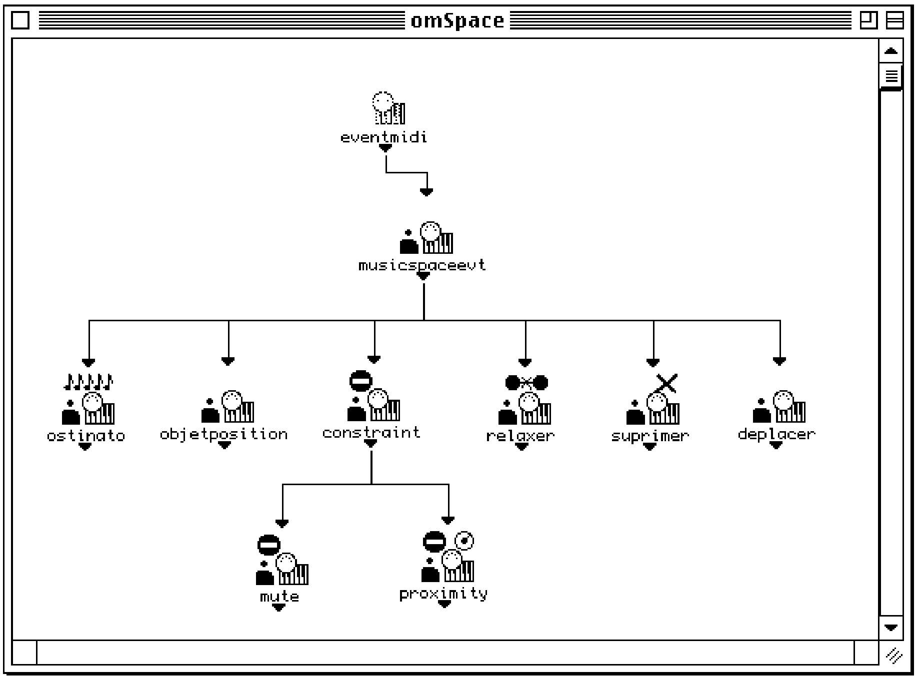

Figure 3 : la hiérarchie des classes MusicSpace en OpenMusic

Parmi les sous-classes de la classe «musicspaceevent», nous distinguons les classes «ostinato», «mute» et «proximity» qui seront utilisées l'exemple de la section suivante.

4.2 Génération de partiels à partir d'un accord

L'exemple que nous proposons à titre d'illustration est de générer un ensemble d'objets «ostinato» à partir d'un accord. Chaque note de l'accord constitue une fréquence fondamentale à partir de laquelle va être générée une série harmonique (un ensemble d'objets ostinato dont les valeurs de fréquence sont un multiple entier de la fondamentale). La Figure 4 représente le patch principal qui génère ces objets : il est organisé autour d'une boucle principale (omloop) dont le contenu est représenté en Figure 5 et qui prend en entrée un accord.

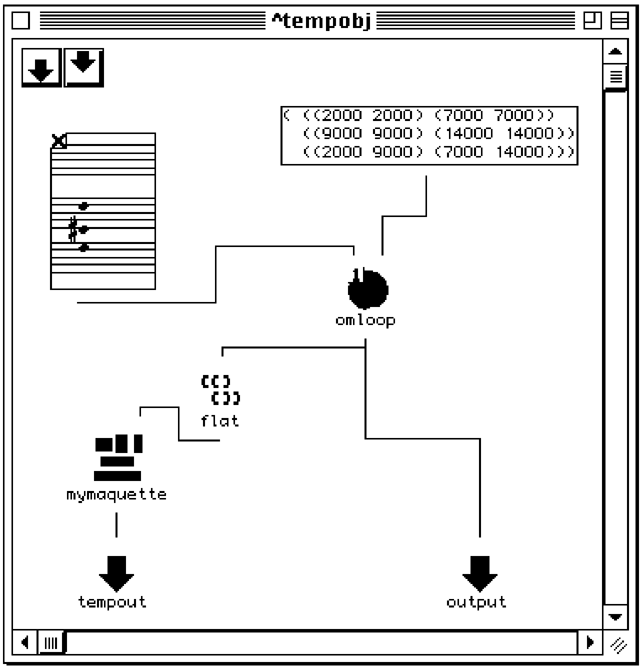

Figure 4 : un patch utilisant un accord pour construire des objets «ostinato » dans MusicSpace

Pour finir, le résultat de cette boucle est remis à un constructeur de maquette qui va distribuer proportionnellement les différents messages générés au niveau temporel.

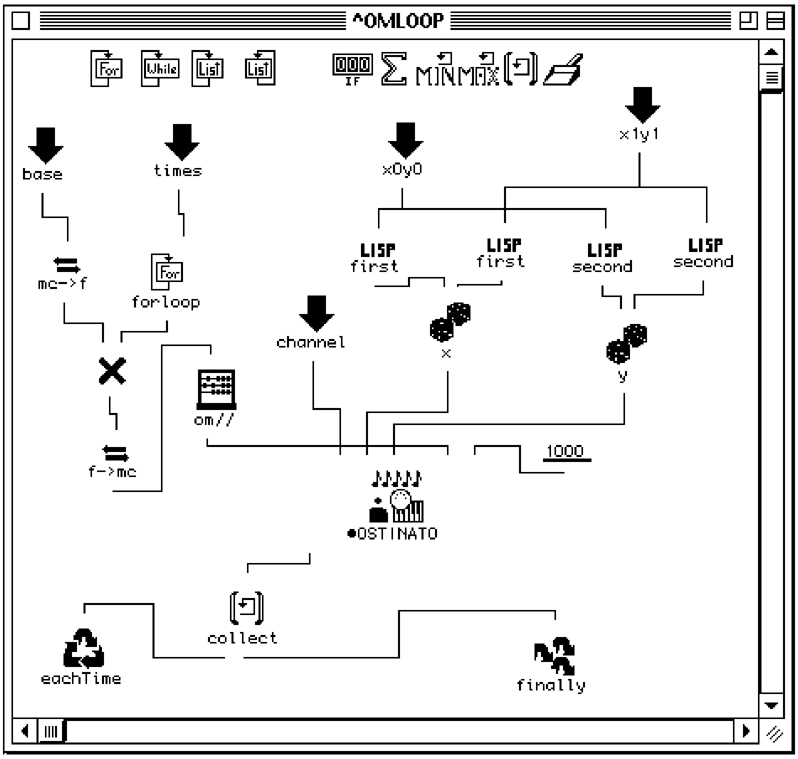

Figure 5 : détail de l'implémentation générant principale du patch de la figure 3

La Figure 5 détaille la création des objets «ostinato» à l'intérieur de la boucle : pour chaque note fondamentale donnée (paramètre «base»), nous générons «times» hauteurs obtenues par muliplications entières de «base». Une conversion «mc->f» (hauteur Midi vers fréquence) permet d'effectuer ce calcul dans le domaine fréquentiel. La hauteur résultante est ensuite reconvertie en hauteur Midi pour servir de paramètre au constructeur d'ostinato.

4.3 Gestion temporelle des commandes MusicSpace

Pour terminer, nous montrons comment, à l'aide de l'éditeur de maquette d'OpenMusic, il est possible d'organiser dans le temps, la gestion des messages de commande.

La Figure 6 représente une maquette dans laquelle ont été plongés trois patchs différents. Le premier de ces patchs correspond à celui décrit dans la Figure 4 : celui-ci va donc créer l'ensemble des objets ostinato correspondant aux différents spectres harmoniques, de manière progressive et pendant les cinq premières secondes. Puis, dans un deuxième temps, des contraintes de « mute » et de proximité sont crées : cette opération dure elle même 5 secondes. Finalement, un troisième patch va progressivement supprimer tous les objets qui ont été crées.

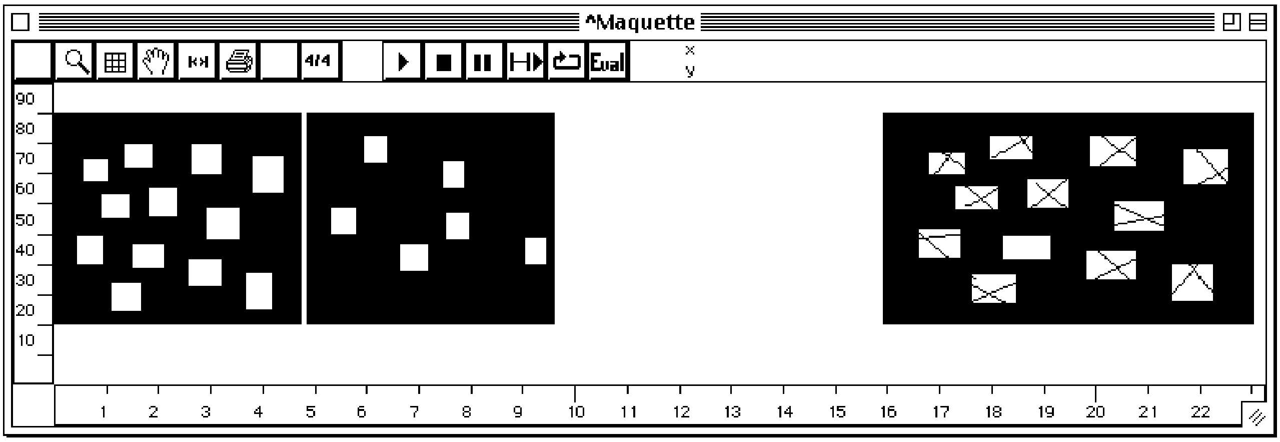

Figure 6 : organisation temporelle des messages de commandes à l'aide de l'éditeur de maquettes

Ces patchs temporels peuvent alors facilement être déplacés (dans le temps) ou étirés.

5. Perspectives

Notre exemple met en évidence la création, modification ou suppression dynamique d'objets sonores dans MusicSpace à l'aide d'OpenMusic et de son éditeur temporel. Une perspective possible de recherche consisterait à rendre également dynamique une partie de l'interface même de MusicSpace. En plus du déplacement libre des objets à l'aide de la souris, l'utilisateur aurait la possibilité d'agir via des boutons, des commandes en menus ou autres qui seraient crées, modifiés ou supprimés dynamiquement depuis OpenMusic.

6. Références

[**Agon98**] Agon, C., Assayag, G., Delerue, O., Rueda, C. *« Objects, Time and Constraints in OpenMu0sic »*, proceedings of the International Computer Music Conference, Ann Arbor 1998.

[**Braut95**] Braut, Christian, *«Le livre d'or de la norme MIDI»*, tome 2, éditions Sybex, Paris 1995.

[**Delerue 98**] Delerue, O., Pachet, F. *«MidiSpace, un spatialisateur Midi interactif.»* Actes des Journée d'informatique Musicale 1998.

[**Fober 98**] Fober, D., Letz, S., Orlarey, Y. Midishare, *«Un système d'exploitation musical pour la communication et la collaboration»* Recherches et Applications en Informatique Musicale, Hermes, 1998.

[**Fober 98**] Fober, D., Letz, S., Orlarey, Y, Carron, T. *«Cristallisation d'applications musicales par collaboration»* Actes des Journées d'Informatique Musicale 1998b.

[**Furia87**] de Furia, Steve, Scacciaferro, Joe *«The MIDI ressource book»*, Third Earth Publishing INc., Pompton Lakes,N.J., 1987

[**Furia86**] de Furia, Steve, Scacciaferro, Joe *«The MIDI implementation book»*, Third Earth Publishing INc., Pompton Lakes,N.J., 1986

[**Midi89**] Midi 1.0 Detailed specification Document version 4.1 - Janvier 1989 - Copyright c 1989, midi Manufacturers association, Japan Midi Standards Committee. Published and distributed exclusively by : the International Midi Association 5316 W. 57th St. Los Angeles, CA 90056 USA 213 / 649 - 6434