MidiSpace, un spatialisateur Midi interactif. Delerue & Pachet, Journées d'informatique Musicale, JIM 98, Agelonde, D. Arfib Ed.

MidiSpace, un spatialisateur Midi interactif

Olivier Delerue, François Pachet
Sony CSL Paris, 6, rue Amyot, 75005 Paris, France
Email : delerue {pachet} @csl.sony.fr

Résumé :
Nous présentons dans ce papier un spatialisateur Midi interactif, qui permet à un utilisateur non expert de modifier la localisation de sources sonores de manière interactive. Nous décrivons les principaux problèmes techniques rencontrés et les solutions adoptées. Nous présentons brièvement quelques extensions de MidiSpace en cours permettant de mettre en oeuvre le système dans le cadre du projet d'*Ecoute Active* de Sony CSL.

1. Ecoute active

L'écoute active désigne une des activités de recherche menées dans l'équipe musique du laboratoire Sony CSL de Paris. Elle consiste a explorer divers paramètres de contrôle agissant sur de la musique préenregistrée, et afin d'enrichir la pratique de l'écoute musicale. Ces paramètres de contrôle définissent des espaces virtuels dans lesquels des pièces de musique peuvent être jouées par des auditeurs non experts tout en subissant certaines variations. L'objectif est à la fois d'augmenter le confort d'écoute des auditeurs, mais aussi de proposer des "chemins" à des musiques mal connues ou difficiles d'accès. Proche de l'idée de forme ouverte, l'écoute active s'en différencie en ce que nous nous intéressons à l'exploitation de répertoires musicaux existants, plus qu'à l'exploration libre ou la création de nouveaux matériaux musicaux.

Parmi les différents paramètres qu'il est possible de mettre entre les mains d'utilisateurs non experts, l'un d'entre eux particulièrement naturel: le mixage des différentes sources sonores, ou plus généralement, la spatialisation du son. Ce papier décrit quelques expérimentations et résultats obtenus en ce sens autour d'un système interactif de spatialisation temps réel appelé MidiSpace. La section 2 rappelle certains principes de la spatialisation. La section 3 décrit MidiSpace, notre système de spatialisation Midi. La section 4 décrit quelques extensions en cours au système MidiSpace.

2. La spatialisation

La spatialisation du son est un domaine qui a fait l'objet de recherches intensives en informatique musicale. La plupart de ces études ont abouti à des systèmes qui permettent de simuler des espaces acoustiques en filtrant des signaux sonores. Ces travaux sont fondés sur des études psychoacoustiques qui permettent de modéliser la perception auditive de l'espace par un nombre limité de paramètres perceptifs (Chowning, 1971). Ces modèles perceptifs ont abouti à un ensemble de techniques permettant de recréer la sensation de localisation sonore tridimensionnelle en utilisant un nombre limité de haut-parleurs par exemple. Ces techniques exploitent typiquement trois paramètres: la différence d'amplitude entre les canaux, les délais entre les canaux, et le contenu spectral des signaux de chaque canal. Combinés ensembles, ces trois paramètres permettent d'obtenir des impressions de direction et de distance tout à fait réalistes (Chowning, 1971).
Par exemple, le *Spatialisateur Ircam* (Jot & Warusfel, 1995) est un processeur d'acoustique virtuelle qui réalise la synthèse de la localisation des sources sonores et de l'effet de salle

MidiSpace, un spatialisateur Midi interactif. Delerue & Pachet, Journées d'informatique Musicale, JIM 98, Agelonde, D. Arfib Ed.

(réverbération artificielle). L'une des originalités de ce processeur est d'offrir un paramétrage de la scène sonore synthétisée sous la forme d'un jeu de facteurs perceptifs qui comprennent les angles d'azimut et d'élévation et d'orientation des sources sonores par rapport à l'auditeur et des descripteurs de la qualité acoustique (effet de salle) associée à chaque source. Le Spatialisateur s'adapte automatiquement au mode de reproduction et à la configuration du système électroacoustique : reproduction tridimensionnelle sur casque ou couple de haut-parleurs, systèmes multi-canaux incluant les configurations stéréo-3/2 et mode de reproduction Ambisonics.
Le marché a récemment vu apparaître d'autres système commerciaux, tels que le système RSS de Roland, le *Spatializer* (Spatializer Audio Labs) qui permet de produire un signal stéréo tridimensionnel à partir de huit pistes monophoniques contrôlées à partir de joysticks, ou encore le système *Q-sound* (Q-sound Labs) qui construit un signal stéréophonique élargi en mettant en oeuvre des techniques similaires.

L'utilisation des ces techniques pour la réalisation de documents sonore pose certains problèmes en particulier lors de la diffusion radiophonique : en effet, la prise en compte des différences temporelles entre les canaux produit des signaux dont les différences de phase interdisent toute réduction monophonique. En revanche, le monde du multimédia s'est largement développé dans cette direction : l'API *DirectX* de la société Microsoft (DirectX, 1998) précise cette tendance en intégrant un ensemble de routines permettant de gérer un espace sonore tridimensionnel.

Ces techniques de spatialisation sont exploitées principalement pour la constructions d'univers virtuels tels que *the Cave* ou le *Cyberstage* (Dai et al 97), (Eckel, 97). Récemment, la spatialisation du son a également été intégrée, de manière partielle, dans certains environnements tridimensionnels tels que VRML dans son implémentation par Community place (Lea et al., 1996) ou ET++ (Ackermann, 1996).

Sur la base de ces nombreux travaux, sous souhaitons exploiter les techniques de spatialisation du son pour construire des environnements d'écoute plus riches. Un des aspects qui nous semblent important est de maintenir une certaine *cohérence* musicale ou sonore, tout en permettant à l'utilisateur d'effectuer librement un certain nombre de modifications et d'explorations. La section 3 décrit le système de base que nous avons conçu, puis dans la section 4 nous verrons comment certaines informations sémantiques peuvent être ajoutées afin de maintenir cette cohérence.

3. MidiSpace

MidiSpace est un logiciel dont l'objectif est, dans un premier temps, de mettre en application nos idées sur l'écoute active dans le cadre de la spatialisation du son. Il s'agit d'un programme permettant d'écouter de la musique, dans lequel l'utilisateur a la possibilité d'agir en temps réel afin de modifier, d'adapter à sa convenance, la répartition spatiale des différentes sources sonores.

3.1 Description générale

La version actuelle de MidiSpace est destinée à l'écoute de fichiers Midi et permet à l'utilisateur de contrôler en temps réel la localisation des sources sonores à travers une interface en deux dimensions présentée Figure 1. L'idée de base est de représenter dans un éditeur d'une part les différentes sources sonores qui composent le fichier Midi, et d'autre part une icône qui symbolise l'auditeur : son *avatar*.

MidiSpace, un spatialisateur Midi interactif. Delerue & Pachet, Journées d'informatique Musicale, JIM 98, Agelonde, D. Arfib Ed.

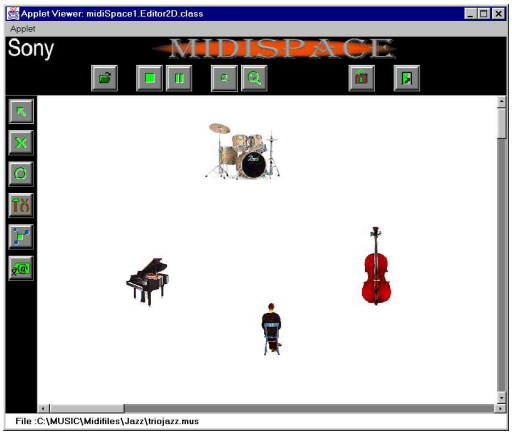

Figure 1 : L'interface en deux dimensions de MidiSpace.

La convention *General Midi* associe un type de son précis à chacun des 128 numéros de programmes. Par exemple, le numéro 1 correspond à un piano, le 2 est également un piano, mais avec un son plus brillant, le numéro 13 est un marimba, etc. Cette spécification nous est nécessaire afin que le logiciel puisse interpréter correctement les fichiers Midi, c'est à dire en utilisant des sons qui correspondent en théorie aux sons originaux. De plus, en nous fondant sur cette spécification, il est possible de représenter les sources sonores par une icône qui corresponde à l'instrument utilisé. Dans l'exemple présenté sur la Figure 1, par exemple, l'utilisateur est informé implicitement que le fichier Midi qu'il vient d'ouvrir contient trois instruments, un piano, une batterie et une contrebasse. Cette fonctionnalité rend l'interface de MidiSpace intuitive et facile à utiliser.

L'architecture générale de MidiSpace, présentée Figure 2, s'organise autour d'un séquenceur spécialisé. MidiSpace reçoit deux types différents de données : d'une part un fichier midi, préalablement filtré et réorganisé par un Parser, pour en construire une représentation interne. Ces données seront exploitées par le séquenceur lors de la lecture. D'autre part, un ensemble de données proviennent, elles, en temps réel, des différentes actions que l'utilisateur effectue via l'interface. Celles-ci sont interprétées par le spatialisateur avant d'être adressées au séquenceur.

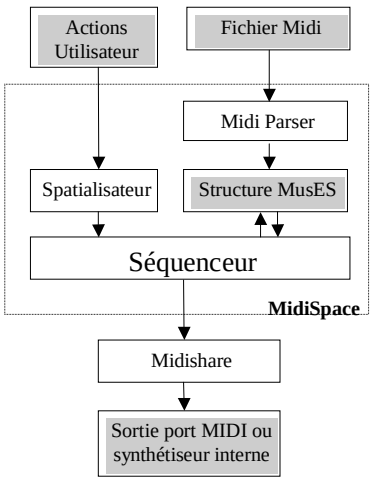

Figure 2 : Architecture de MidiSpace

MidiSpace, un spatialisateur Midi interactif. Delerue & Pachet, Journées d'informatique Musicale, JIM 98, Agelonde, D. Arfib Ed.

En résumé, MidiSpace est conçu pour spatialiser de manière interactive, en temps réel, les différentes sources sonores qui composent des fichiers Midi quelconques. Si le principe de MidiSpace est simple, sa mise en oeuvre soulève plusieurs problèmes techniques non triviaux. Les sections suivantes de cet article détaillent ces points. La section 3.2 décrit le module de spatialisation Midi et met en évidence le problème du filtrage et de la réorganisation des fichiers Midi. La solution à ce problème passe par un traitement initial des fichiers midi, détaillé dans la section 3.3. La partie temps réel de MidiSpace, l'implémentation du séquenceur, est décrite dans la section 3.4. La section 3.5 regroupe les résultats obtenus ainsi que les limitations de notre système. Enfin, quelques extensions en cours de MidiSpace sont présentées section 4.

3.2 Le module de spatialisation Midi

Dans l'éditeur de MidiSpace, l'utilisateur peut déplacer à sa guise les sources sonores ou son avatar avec la souris : les positions relatives des différentes sources par rapport à celle de l'avatar déterminent alors l'ensemble du mixage sonore, suivant des règles géométriques simples représentées Figure 3. Ainsi, le niveau sonore d'une source est calculé comme une fonction de la distance ρ de cette source à l'avatar. Le résultat est ensuite converti en une valeur de niveau Midi, c'est à dire un entier compris entre 0 (le niveau le plus faible) et 127 (le niveau nominal).

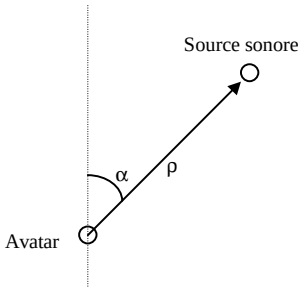

Figure 3 : position d'une source par rapport à l'avatar, exprimée en coordonnées polaires par un angle α et un rayon ρ

De la même manière, la position stéréophonique de la source sera évaluée à partir de l'angle α que fait cette source avec l'axe vertical passant par l'avatar. La valeur obtenue est également convertie en valeur de panoramique Midi, c'est à dire un entier compris entre 0,où lorsque le son provient de la gauche, et 127, lorsque le son provient de la droite.

Ainsi, le mixage en temps réel des différentes sources sonores est réalisé par l'envoi de messages Midi *control change* de type *volume* (numéro de contrôleur égal à 7) et *panoramique* (numéro de contrôleur égal à 10). Chaque fois qu'un instrument est déplacé, ses valeurs de niveau et de panoramique sont réévaluées puis envoyées sur le canal Midi correspondant. Lorsque c'est l'avatar qui est déplacé, cette opération est effectuée pour l'ensemble des sources sonores. La combinaison de ces contrôles de volume et de panoramique résulte physiquement en des variations sur le volume global du son, ainsi qu'un dosage entre le niveau des canaux gauche et droit. Cette limitation liée directement au système Midi signifie que nous n'utilisons qu'un seul des trois paramètres propres au techniques binaurales ou transaurales de simulation de localisation des sources, les différences temporelles et spectrales entre les canaux ne pouvant pas prises en compte de manière générique. Cependant l'expérimentation montre que la seule utilisation de ce volumes suffit pour produire des résultats convaincants.

MidiSpace, un spatialisateur Midi interactif. Delerue & Pachet, Journées d'informatique Musicale, JIM 98, Agelonde, D. Arfib Ed.

Cependant, cette spatialisation Midi ainsi décrite requiert que les données musicales soient organisées de manière cohérente. En particulier, il est impératif que chaque instrument représenté graphiquement représente de manière bijective un instrument au niveau sonore. C'est le rôle du traitement initial des fichiers Midi, détaillé dans la section suivante.

3.3 Parsing des fichiers Midi

La nature même des fichiers Midi rend nécessaire cette opération de Parsing. En effet, le module de spatialisation nécessite que chaque instrument représenté graphiquement à l'écran représente de manière non ambiguë un instrument sonore. Or, la structure organisée en *tracks* des fichiers Midi ne correspond pas forcément à ce type d'organisation : elle reflète principalement la manière dont le fichier a été créé, puisqu'elle est généralement générée par le séquenceur qui a servi à enregistrer la musique et n'est donc pas à l'image d'une organisation intuitive par instruments musicaux. Par exemple, une partie de piano peut avoir été enregistrée en plusieurs fois, et l'on en trouvera des fragments sur plusieurs *tracks* différentes. A l'inverse, une seule *track* du fichier midi peut contenir plusieurs instruments puisque chaque message midi à l'intérieur d'une *track* possède son propre numéro de canal Midi. C'est le cas en particulier dans le format MidiFile 0, pour lequel l'ensemble des données musicales sont regroupées sur une seule piste !

Le parsing des fichiers Midi a donc pour rôle de transcrire les informations du fichier Midi en construisant une représentation interne adaptée à la représentation graphique que l'on souhaite proposer à l'utilisateur, ainsi qu'aux différents traitement qui seront effectués. Le résultat de cette réorganisation des données musicales est une structure objet compatible avec le système MusES, décrit dans (Pachet, 1996).

Ce filtre intervient donc au chargement du fichier Midi : les informations contenues dans les fichiers Midi doivent être restructurées par un algorithme de tri, en deux passes successives. Tout d'abord, lors d'une lecture séquentielle (linéaire) du fichier, les messages *program-change* sont identifiés de manière à construire un ensemble de pistes qui représente la structure globale des données musicales. Lors de cette même lecture, le reste des messages Midi va être mis de coté pour la deuxième passe.

En effet, l'envoi d'un message *program-change*, comprenant un canal Midi et un numéro de programme vers le port Midi initialise le canal Midi correspondant au numéro de programme. Ce sont donc les message *program-change* qui vont permettre de délimiter clairement les différentes voix du fichier : chaque fois qu'un message Midi *program-change* est rencontré, une nouvelle mélodie, vide, sera créée. Cependant, si une mélodie a déjà été créée pour le canal midi et le numéro de programme correspondant, le message *program-change* est alors inséré dans cette mélodie : un canal midi peut être partagé entre plusieurs instruments différents, à la condition qu'ils ne jouent pas simultanément. Dans ce cas, les messages *program-change* s'interprètent comme le basculement d'un instrument à un autre. La Figure 4 décrit en pseudo-code la manière dont les messages *program-change* sont gérés dans le parser.

```
ParseProgramChanges(program-change : new_program_change )
found = null
For each melody in multimelody
  if ( melody.channel = new_program_change.channel
     and melody.program = new_program_change.program )
          found = melody
  end if
end for

if found ≠ null /* une mélodie correspond au canal et programme Midi */
  found.add( new_program_change ) /* on ajoute simplement le nouveau message */
else
  new_melody = new melody() /* sinon, on crée une nouvelle mélodie */
```

MidiSpace, un spatialisateur Midi interactif. Delerue & Pachet, Journées d'informatique Musicale, JIM 98, Agelonde, D. Arfib Ed.

```
    new_melody.program = new_program_change.program
    new_melody.channel = new_program_change.channel
    new_melody.add( new_program_change)
  end if
```

Figure 4 : gestion des messages program-change

Dans un deuxième temps, les éléments restant sont distribués dans l'ensemble des différentes mélodies créées pendant la première passe. Naturellement, chaque message Midi va être attribuée à la mélodie qui correspond à son canal Midi. Cependant, lorsque il existe plusieurs mélodies utilisant un même canal (dans le cas où ce canal Midi est partagé entre plusieurs instruments), le message sera attribué à la mélodie possédant le message *program-change* antérieur le plus proche du message à classer.

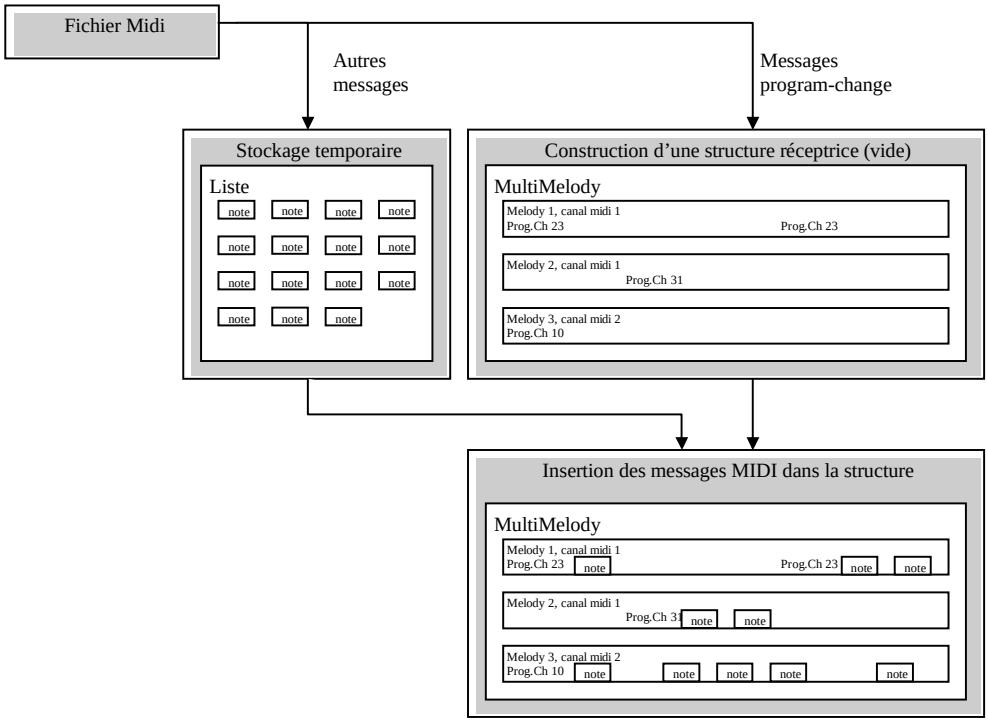

Figure 5 : Traitement initial des fichiers Midi

Pour finir, une opération de nettoyage va analyser l'ensemble des mélodies créées et supprimer celles qui ne contiennent pas de notes. Ce cas peut se produire lorsqu'un message *program-change* est isolé dans un fichier midi, c'est à dire qu'il n'est pas suivi par des notes sur ce canal, ou qu'il est directement suivi d'un autre message *program-change* : il provoque alors la création d'une mélodie qui reste vide lors de la deuxième passe du parser. De même, cette phase est également l'occasion de supprimer d'éventuels doublons, des messages *program-change*, redondants et générés par certain séquenceurs lors de l'exportation au format midifile.

En conclusion, cette opération d'interprétation permet de résoudre un certain nombre d'ambiguïtés rencontrées dans les fichiers Midi : une fois cette phase accomplie, la structure interne des données musicales reflète une organisation fondée sur les instruments (et non plus sur les *tracks*).
Cette solution a deux inconvénients. D'une part, les fichiers Midi qui ne sont pas munis de messages programmes ne peuvent pas être interprétés correctement. Cette limitation est rhédibitoire.

MidiSpace, un spatialisateur Midi interactif. Delerue & Pachet, Journées d'informatique Musicale, JIM 98, Agelonde, D. Arfib Ed.

D'autre part, il n'est pas envisageable de déterminer si deux pistes différentes, munies de canaux Midi différents, mais utilisant le même numéro de programme, décrivent des données appartenant à un ou deux instruments différents. Par exemple, deux pistes sur des canaux Midi différents utilisant le même numéro de programme, un piano par exemple, peuvent aussi bien décrire une pièce pour deux pianos, que les mains gauche et droite d'un morceau pour un seul piano. Il n'est pas possible de réaliser cette distinction de manière automatique et systématique. Par défaut, nous affectons des canaux Midi différents à des instruments différents. Ainsi, dans l'exemple précédent, deux objets graphiques distincts apparaîtraient à l'écran. Bien sûr ce choix est paramétrable.

3.4 Le séquenceur Midi

La partie Midi temps réel de MidiSpace repose sur le système d'exploitation Midi *MidiShare* développé au studio Grame (Fober et al., 1998). Midishare intègre un séquenceur Midi dans sa version de base. Cela signifie qu'il est donc possible, sans effort particulier de programmation, de lire un fichier Midi, de le convertir en structure Midishare et d'en exécuter la lecture. De plus, étant donné que ce séquenceur est écrit en code natif, il agit à un niveau très bas, et garantit que la lecture ne sera pas interrompue de manière intempestive par le système d'exploitation ou les applications en cours. Ceci est particulièrement appréciable lorsque le langage de programmation, de haut niveau, ne permet pas de garantir des temps d'exécution.
La conséquence est que ce séquenceur n'offre pas de contrôle sur les événements envoyés au port Midi, au moment où ils sont programmés. Ceci est donc incompatible avec la possibilité de réaliser du filtrage à la volée. C'est la raison pour laquelle nous avons réalisé notre propre séquenceur, dont cette section en décrit l'implémentation, à partir des fonctionnalités de base de Midishare.

Pour chaque Melody du système MusES, une tâche Midi (MidiTask) est créée. Celles ci sont régulièrement programmées à la date du prochain événement de la mélodie, et invoquent une méthode playSlice du séquenceur Java dont la fonction est d'envoyer au périphérique Midi (ou à une autre application Midishare) tous les messages qui doivent être joués à l'instant du réveil.

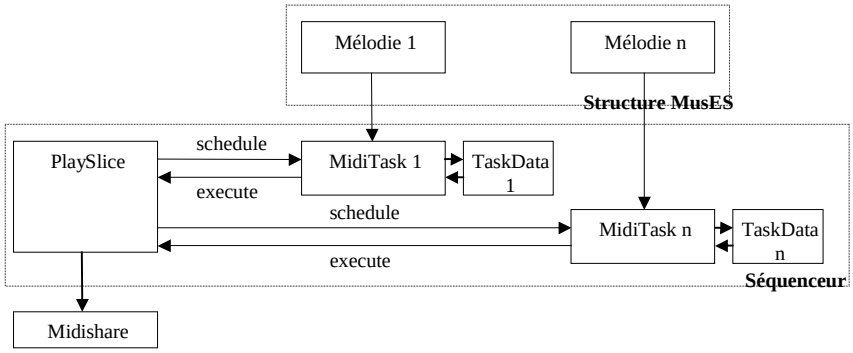

Figure 6 : architecture du séquenceur de MidiSpace

La Figure 7 décrit en pseudo-code la méthode playSlice du séquenceur dont la fonction est, pour une mélodie et à une date données, d'envoyer au port Midi tous les événements de la mélodie qui correspondent à cette date. A la fin de cette opération, s'il reste des événements à jouer dans la mélodie, la tache est reprogrammée à la date de l'événement suivant.

```
playSlice(date : thisDate, task : musicalTask )
  index = musicalTask.Index
  eventToPlay = musicalTask.melody.event(index)
```

MidiSpace, un spatialisateur Midi interactif. Delerue & Pachet, Journées d'informatique Musicale, JIM 98, Agelonde, D. Arfib Ed.

```
// joue tous les événements de melody dont la date coïncide avec thisDate.
while (index < musicalTask.melody.size and eventToPlay.date = thisDate )
        eventToPlay.playAt( thisDate + advance )
        index++
        eventToPlay = musicalTask.melody.event(index)
endWhile
if  (index = musicalTask.melody.size) // la fin de la melodie est atteinte
        return
else
        musicalTask.index = index
        musicalTask.scheduleAt(eventToPlay.date)
        // la tache est reprogrammée à la prochaine date
endIf
```

Figure 7 : Algorithme de la fonction principale du séquenceur

On remarquera dans l'algorithme que chaque événement est joué à la date `thisDate + advance` : ce paramètre supplémentaire est une variable globale du séquenceur, fixé arbitrairement à 200 millisecondes. Ce décalage global du temps permet au programme de faire face à un éventuel blocage du système au moment où l'événement doit être envoyé à Midishare et donc d'assurer que, globalement, l'ensemble des événements sera parfaitement synchronisé.

3.5 Résultats

MidiSpace se révèle à l'usage tout a fait satisfaisant, d'une part vis à vis d'un corpus de fichiers Midi récoltés sur Internet (par exemple MidiFiles, 1998), et d'autre part vis à vis de l'expérimentation avec des utilisateurs.

MidiSpace possède quand même ses propres limitations, dues en partie à son implémentation Midi. En effet, un fichier Midi peut ne pas être compatible avec les spécifications *General Midi*. Il en résulte une écoute avec des sons qui ne correspondent pas à ceux prévus par le compositeur. Or le terme *General Midi*, qui correspond davantage à une convention qu'à un format, apporte des spécifications au niveau des *appareils* Midi et non des *fichiers*. Ainsi, rien n'est inscrit à l'intérieur du fichier qui permettrait de prévoir si celui-ci sera lu avec les instruments qui lui correspondent ou non. Il n'est donc pas possible de filtrer ces fichiers, pour éliminer ceux qui ne seraient pas interprétés correctement par notre système. Plusieurs efforts ont été porté récemment pour étendre le format Midi afin de pallier ces inconvénients (Zipi, MOD, etc.). Même si certaines de ces propositions sont techniquement convaincantes, elles ne se sont pas imposées comme standard.

Une autre limitation constatée concerne les sons percussifs (de type boite à rythme) que la spécification *General Midi* limite au canal 10. Cela signifie entre autre qu'il ne sera pas possible d'avoir plusieurs objets graphiques correspondant à des parties percussives différentes et simultanées : en effet, deux objets graphiques ne peuvent représenter en même temps le même canal Midi.

4. Extensions de MidiSpace

Cette version précédemment décrite de MidiSpace propose à l'utilisateur un premier type de contrôle sur l'aspect qualitatif du son : il lui est possible de mettre en jeu son appréciation sur le mixage des différentes voix et éventuellement de l'adapter à ses préférences, par l'intermédiaire du module de spatialisation. Ce système est alors un point de départ vers d'autres recherches qui visent notamment à intégrer un ensemble de connaissances supplémentaires aux systèmes multimédia de manière à ce que, aussi ouverts qu'ils soient, le résultat restera toujours cohérent avec le contenu musical initial.

Ainsi, à la version de base de MidiSpace s'intègrent trois extensions qui représentent des directions de recherches différentes dans l'optique de l'écoute active (Figure 8).

MidiSpace, un spatialisateur Midi interactif. Delerue & Pachet, Journées d'informatique Musicale, JIM 98, Agelonde, D. Arfib Ed.

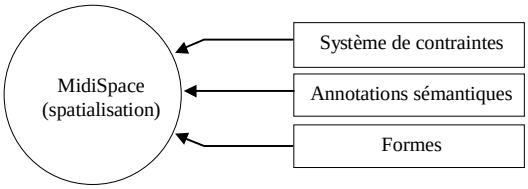

Figure 8 : MidiSpace et ses différentes extensions

4.1 Contraintes

Tout d'abord un système de propagation de contraintes en cours de réalisation (Pachet & Delerue, 1998) permet d'exprimer et de formaliser un ensemble de connaissances, propres au domaine du mixage sonore, et ainsi assurer que les modifications effectuées par l'utilisateur restent dans des limites cohérentes avec la nature du morceau de musique écouté. Ce système permet essentiellement à un compositeur de spécifier des contraintes graphiquement en liant plusieurs objets graphiques par des relations, à partir d'une palette prédéfine de contraintes de base. Ainsi, on peut spécifier que les instruments rythmiques (basse, batterie) doivent rester ensemble, autant que possibles, ou bien que les instruments "solistes" (piano, saxophone, etc.) doivent être, au contraire, en continuelle balance avec la rythmique, etc. Lorsque l'utilisateur déplace un objet, le système de contrainte déplace de lui-même les autres objets, de manière à satisfaire les contraintes.

4.2 Annotations

D'autre part, un système d'annotation des fichiers musicaux permet de spécifier des connaissances diverses sur le "contenu musical" du morceau telles que sa structure globale (découpage en segments comme introduction, chorus, coda, etc.), sa structure harmonique ou sa tonalité par exemple. Ces diverses annotations temporelles permettent d'enrichir le système en donnant des informations qui peuvent être exploitées par MidiSpace. Typiquement, si le système sait que tel instrument est en train de jouer un chorus, il peut alors le rapprocher de l'auditeur, éventuellement en satisfiant les contraintes préalablement spécifiées, etc.

4.3 Formes

Enfin, le système de formes a pour but d'élargir les possibilités de contrôle de l'utilisateur à un plus haut niveau que l'aspect purement "instrumental" de la spatialisation. L'idée ici est de construire des objets graphiques qui ne correspondent pas simplement à des instruments, mais à des "formes musicales" abstraites. Par exemple, on peut spatialiser les notes aiguës, les notes graves, les notes formant des intervalles consonants, ou bien encore les éléments musicaux (notes, percussions) faisant partie d'un thème, pour les oeuvres thématiques comme les opéras.

5. Conclusion

L'étude et la réalisation de MidiSpace a permis de dégager un certain nombre d'idées intéressantes et originales sur l'exploitation de la spatialisation à des fins d'écoute par des non experts.

Les travaux en cours concernent d'une part les diverses extensions décrites dans la section précédente. D'autre part, la réalisation d'une version audio de MidiSpace est prévue, en faisant appel par exemple à la librairie *DirectX*. Elle permettra d'obtenir des résultats plus

MidiSpace, un spatialisateur Midi interactif. Delerue & Pachet, Journées d'informatique Musicale, JIM 98, Agelonde, D. Arfib Ed.

convainquants sur le plan sonore puisque cette librairie intègre, pour gérer la localisation spatiale des sources, non seulement les différences de niveau entre les canaux gauche et droit, mais aussi les différences temporelles et les différences spectrales (prise en compte par un filtre *Muffling*). Le gain sera également important du point de vue de la source sonore, a priori plus naturelle, et autorisant des source vocales, non disponibles en Midi.

6. Références

Ackermann P., *Developing object-oriented multimedia software*, Dpunkt, Heidelberg, 1996.

Assayag G., Agon C., Fineberg, J., Hanappe P., "An Object Oriented Visual Environment For Musical Composition", *Proceedings of the International Computer Music Conference*, pp. 364-367, Thessaloniki, 1997.

Borning A., Anderson R., Freeman-Benson B., "Indigo: A Local Propagation Algorithm for Inequality Constraints", *Proceedings of the ACM Symposium on User Interface Software and Technology*, pp. 129-136, 1996.

Borning A., Freeman-Benson, B. "The OTI Constraint Solver : a Constraint Library for Constructing Interactive Graphical User Interfaces", *Proceedings of the First International Conference on Principles and Practice of Constraint Programming*, pp. 624-628, 1995.

Borning A., Freeman-Benson, B., "Ultraviolet: A Constraint Satisfaction Algorithm for Interactive Graphics", *Constraints*, Special Issue on Constraints, Graphics, and Visualization, Vol. 3 No. 1, pp. 9-32, April 1998.

Borning A., "The Programming Aspects of ThingLab, a Constraint-Oriented Simulation Laboratory", *ACM Transactions on Programming Languages and Systems*, 3 (1981), pp. 353-387.

Borning, A. Lin, R., Marriott, K. "Constraints for the web", *Proceedings of ACM Multimedia Conference*, Seattle, pp. 173-181, 1997.

Chowing, J. (1971), "The simulation of moving sound sources", *JAES*, vol. 19, n. 1, p. 2-6.

Dai P., Eckel G., Göbel M., Hasenbrink F., Lalioti V., Lechner U., Strassner J., Tramberend H., Wesche G., "Virtual Spaces: VR Projection System Technologies and Applications", Tutorial Notes, Eurographics '97, Budapest 1997, 75 pages.

DirectX 1998 : Référence en ligne sur le site microsoft. http://www.microsoft.com/msdownload/directx/dxf/sdk5.0/default.htm

Eckel G., "Exploring Musical Space by Means of Virtual Architecture", *Proceedings of the 8th International Symposium on Electronic Art*, School of the Art Institute of Chicago, 1997.

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.

Hosobe H., Matsuoka S. Yonezawa A., "Generalized local propagation: a framework for solving constraint hierarchies", *Proceedings of CP' 96*, Boston, August 1996.

Lopez G., Freeman-Benson B., Borning A., "Kaleidoscope: A Constraint Imperative Programming Language", In *Constraint Programming*, B. Mayoh, E. Tougu, J. Penjam (Eds.), NATO Advanced Science Institute Series, Series F: Computer and System Sciences, Vol 131, Springer-Verlag, 1994, pages 313-329.

Hower W., Graf, W. H. "a Bibliographical Survey of Constraint-Based Approaches to CAD, Graphics, Layout, Visualization, and related topics", *Knowledge-Based Systems*, Elsevier, vol. 9, n. 7, pp. 449-464, 1996.

IMA, "Midi musical instrument digital interface specification 1.0", Los Angeles, International Midi Association, 1983.

Jot J.-M., Warusfel O. "A Real-Time Spatial Sound Processor for Music and Virtual Reality Applications", *Proceedings of International Computer Music Conference*, September 1995.

MidiSpace, un spatialisateur Midi interactif. Delerue & Pachet, Journées d'informatique Musicale, JIM 98, Agelonde, D. Arfib Ed.

Laurson M., Duthen J., "PatchWork, a graphical language in PreForm", *Proceedings of the International Computer Music Conference*, San Francisco,172-175, 1989.

Lea R., Matsuda K., Myashita K., *Java for 3D and VRML worlds*, New Riders Publishing, 1996.

Meyer L., *Emotions and meaning in music*, University of Chicago Press, 1956.

MidiFiles, 1998. MidiFile repository sur internet, à l'adresse: (Midi Farm Internet) http://www.midifarm.com/files/midifiles/General_Midi/

Orlarey Y., Lequay H. "MidiShare: a real time multi-tasks software module for Midi applications", *Proceedings of the ICMC*, 1989, ICMA, San Francisco.

Pachet F., Ramalho G., Carrive J., "Representing temporal musical objects and reasoning in the MusES system", *Journal of New Music Research*, vol. 25, n° 3, pp. 252-275, 1996.

Pachet F., Delerue O. "A Constraint-Based Temporal Spatializer", *ACM Multimedia Conference*, submitted, 1998.

Pachet F., Ramalho G., Carrive J. "Representing temporal musical objects and reasoning in the MusES system", *Journal of New Music Research*, vol. 25, n. 3, pp. 252-275, 1996.

Sanella M., Maloney J., Freeman-Benson B., Borning A., "Multi-way versus one-way constraints in user interfaces: experiences with the DeltaBlue algorithm", *Software Practice and Experience*, 23(5):529-566, 1993.

Sloan Donald, "Aspects of Music Representation in Hytime/SMDL", *Computer Music Journal*, Cambridge, MA, MIT Press, 17:4, Winter 1993.

SMDL, Draft International Standard, ISO/IEC CD 10743, 1995.

Taube H., "Common Music: A Music Composition Language in Common Lisp and CLOS", *Computer Music Journal*, vol. 15, n° 2, 21-32, 1991.

Vander Zanden Brad, "An incremental algorithm for satisfying hierarchies of multi-way dataflow constraints", *ACM Transactions on Programming Languages and Systems*, 18(1):30-72, 1996.