Lorenzo Pagliei – Septembre 2009

PERSPECTIVES

Dans cet article nous traiterons des recherches sur la synthèse modale en temps réel réalisées à l'Ircam dans la première partie du 2009. Ce texte concerne surtout la synthèse et les architectures de contrôle développées.
Pour une introduction générale au sujet et à l'approche utilisé, voir l'article « Synthèse par Modèles Physiques en temps réel et geste : Expériences » qui traite des recherches effectuées en 2008 pendant la réalisation de la pièce « L'Apparente » composée par l'auteur.
Pour accomplir la synthèse par modèles physiques, nous avons utilisé le logiciel Modalys dans l'environnement MaxMSP à travers l'objet modalys~ conçu à l'Ircam. Cet objet permet d'accéder aux ressources de synthèse du logiciel Modalys en temps réel avec toutes les possibilités de contrôle de MaxMSP.

Cadre technologique 2009

Il est important de souligner le cadre technologique des recherches actuelles, car entre 2008 et 2009 nous avons changé complètement l'instrumentation hardware et software utilisée dans nos recherches précédentes.
Nous sommes passé d'ordinateurs PPC à ordinateurs dotés de processeurs Intel ; le moteur de synthèse Modalys a été complètement réécrit pour les ordinateurs Intel et souvent il sonne différemment par rapport à la vieille version.
En outre, il y a eu le passage de MaxMSP 4 à MaxMSP 5 qui, au fait, est une application totalement réécrite. Une des nouveautés de MaxMSP 5, est la possibilité d'exploiter les ressources multi-thread d'un ordinateur multi-core à différence de MaxMSP 4. La synthèse modale est très gourmande en termes d'utilisation de CPU, par conséquent, si on veut l'employer en temps réel, il est fondamental pouvoir exploiter toutes ressources de calcul de l'ordinateur. Du point de vue musical, cela signifie avoir à disposition une polyphonie majeure et une meilleure flexibilité musicale.
Pendant le 2008, on a donné au logiciel Modalys la tache de gérer les ressources multi-thread car Max 4 n'avait pas cette capacité. Aujourd'hui, MaxMSP 5 gère les différents threads et on a développé des architectures de contrôle tout à fait différentes et plus dynamiques.

Améliorations

Après l'expérience de « L'Apparente », la priorité des recherches s'est dirigée vers la création de méthodes de contrôle permettant d'augmenter la flexibilité musicale des instruments. Pour flexibilité on entend la possibilité d'utiliser les instruments en toute liberté d'articulation musicale.
Ici les points fondamentaux sur lesquels on a travaillé :

1. **Articulation libre et flexible des fréquences fondamentales des instruments**
   Varier la fréquence fondamentale d'un instrument virtuel pendant le jeu est particulièrement critique car elle est un « paramètre physique » constitutif : si on excite l'instrument pendant cette variation on obtient des discontinuités dans l'audio écoutables en termes de clics ou de sons en *fortissimo* non désirés. Cela comporte que si on veut changer les paramètres constitutifs dans un passage musicale il faudrait envisager une pause entre deux « notes » ou alterner deux copies du même instrument. Ce problème d'initialisation réduit les possibilités d'articulation musicale et donne des ennuis dans l'utilisation concrète. L'écriture musicale de la partition de « L'Apparente » a été beaucoup conditionnée par ce problème.

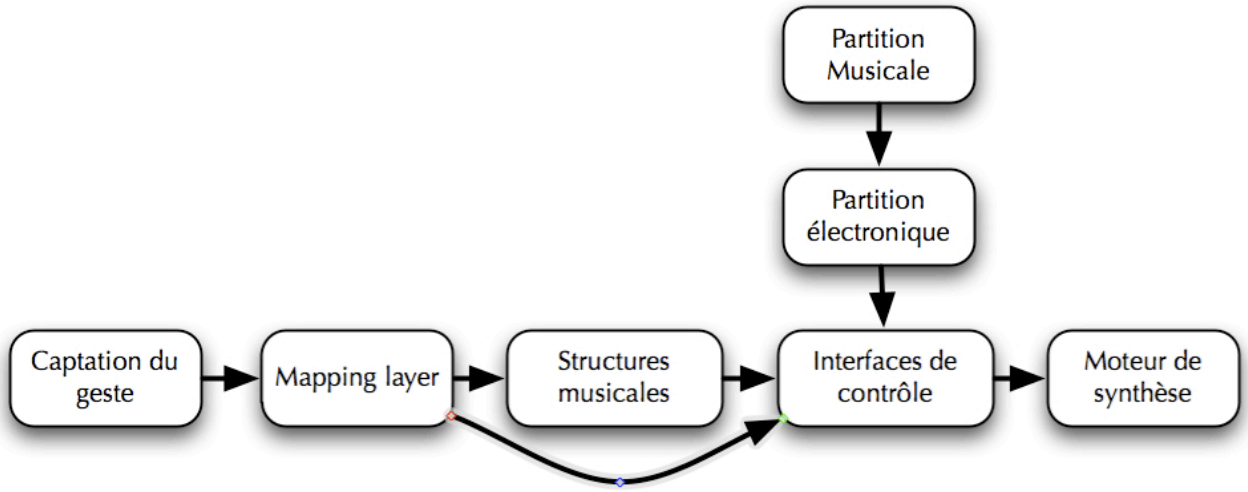
2. **Exploitation des ressources multi-thread des ordinateurs multi-core pour réaliser une majeure polyphonie**
   La disponibilité d'ordinateurs multi-core est relativement récente et les développements technologiques vont vers l'augmentation du nombre de cores même dans les ordinateurs personnels. Si les architectures de contrôle sont capables d'exploiter ces ressources de calcul on peut obtenir une grande polyphonie pour la synthèse modale en temps réel. C'est peut être le sujet principal des recherches dans ce domaine.

3. **Expansion de l'ambitus dynamique des instruments virtuels**
   La gestion des dynamique des instruments virtuels en temps réel a été un des problèmes de L'Apparente, les variations dynamique étaient contrôlées pour la plupart grâce à la partition électronique. On voudrait donner aux instrumentistes le contrôle complet des dynamiques des instruments virtuels.

A l'issu des recherches de ces mois en collaboration avec Nicholas Ellis, nous avons donné une réponse à tous ces problèmes et maintenant on propose une architecture de contrôle plus stable et flexible par rapport au 2008.

ARCHITECTURE ET APPROCHE

Voici un schéma de l'architecture utilisée dans notre approche à la synthèse modale en temps réel. L'excitation des instruments est réalisée par un musicien soit avec les gestes des mains, soit à partir d'un signal audio. Les systèmes de captation du geste, de l'audio et les techniques de Mapping sont expliqués dans l'article « Synthèse par Modèles Physiques en temps réel : Expériences ». Ici nous traiterons en concret seulement des nouvelles interfaces de contrôle et des méthodes de gestion du moteur de synthèse Modalys.



(Fig. 1 – Architecture globale)

Les données musicales sont placées dans les Structures musicales. Les Structures musicales sont nécessaires parce que aujourd'hui on n'a pas à disposition un contrôleur qui peut communiquer directement avec l'interface de contrôle. Un clavier midi serait l'idéal pour piloter le système mais nous n'avons pas l'intérêt de gérer la synthèse avec ce type d'instrument. Les Structures musicales sont comme un réservoir de données dont la lecture est provoquée par le musicien en temps réel.
Notre idée est de créer un système de synthèse qui peut être contrôlé avec plusieurs méthodes et instruments, par conséquent on veut garder une approche la plus générale possible. En fait, au début de la chaine de synthèse (Fig. 1) on peut avoir : un capteur du geste, un senseur de mouvement, une caméra, de la captation audio ou un contrôleur quelconque (midi, touch screen, etc). Ensuite le Mapping Layer adapte les signaux provenant du musicien à la lecture des Structures musicales lesquelles envoient les « demandes de note » à l'interface de contrôle qui les traduit en messages lisibles par le moteur de synthèse.

Dans « L'Apparente » on avait l'intention de construire un riche orchestre de matériaux, pour cette raison on a conçu une collection de patches MaxMSP qui géraient différents types d'instruments modaux. Chaque instrument était, en réalité, constitué par plusieurs instruments modaux égaux utilisant des scripts Modalys ainsi dit « Multi-Instrument ». En fait, seulement avec des scripts « Multi-instrument » Modalys était capable de partager le calcul modal entre les différents processeurs d'un ordinateur et on pouvait obtenir une polyphonie majeure.

Cette architecture impliquait qu'à l'activation d'un Multi-instrument constitué par exemple de 4 plaques, toutes les plaques étaient calculées même si l'on excitait une seule.

Aujourd'hui, nous avons modifié l'architecture en donnant à MaxMSP la gestion des différents thread et les instruments actives ou inactives. MaxMSP 5 est capable de gérer les ressources multi-thread grâce à l'objet poly~ et au système concerné. MaxMSP 4 n'était pas capable d'exploiter correctement ces ressources.
Si on fait tourner un patch MaxMSP 4 sur un ordinateur multi-processeur et on observe l'utilisation de CPU, on peut remarquer très clairement que l'audio est calculé par un seul processeur et que un deuxième processeur se charge de la partie graphique et de l'affichage. Les autres processeurs ne sont pas du tout utilisés par l'application !!!
Au contraire, dans MaxMSP 5 en utilisant l'objet poly~ avec les bons paramètres d'initialisation, on peut observer que tous les processeurs sont utilisés en quantité égale. Malheureusement, cette capacité s'applique seulement au calcul de l'audio et à l'intérieur d'une « architecture poly~ » ; par conséquent, si on utilise un patch complexe qui doit gérer un flux énorme de signaux de contrôle, on est face à la même situation de MaxMSP 4 : un seul processeur sera particulièrement chargé par les données de contrôle et les ressources ne seront pas proprement exploitées.
L'objet poly~ de MaxMSP est expressément conçu pour gérer la polyphonie, il permets la gestion de copies multiples du même instrument : il allume ou éteint les différentes instances et permet ainsi d'économiser les ressources de CPU car les instances inactives ne sont pas calculées. Cet objet accepte des paramètres d'initialisation qui lui indiquent si travailler en mode multi-thread et combien de cores sont à disposition dans l'ordinateur.

Revenons à l'architecture de contrôle de la synthèse ; le système utilisé en 2008 était constitué d'un petit nombre d'instruments multiples (Multi-instrument), alors que maintenant, nous proposons plutôt un grand nombre d'instruments simples.
Les avantages de cette architecture du point de vue technique sont : la majeure facilité et flexibilité de gestion, la facilité de transférer le système entre ordinateurs de puissances différentes, l'économie de CPU et la majeure stabilité. Les avantages du point de vue musical sont : la majeure flexibilité d'articulation, la majeure polyphonie et un ambitus dynamique élargi.
Il y a une majeure facilité de gestion parce que c'est le système et non plus l'utilisateur qui gère quelle instance particulière de l'instrument allumer ou éteindre. Toutefois dans notre système, l'utilisateur peut gérer lui même l'activation des instances grâce à des paramètres locaux directs.
Il y a une majeure flexibilité parce que on peut très facilement ajouter ou éliminer instances de l'instrument sans être obligé de changer les interfaces de contrôle, l'architecture, le nom des paramètres ou le script Modalys chargé par l'objet modalys~ : il suffit un message à l'objet poly~ pour augmenter ou réduire le nombre d'instances et chacune lit le même script Modalys. Cela rend très facile transférer le système d'un ordinateur puissant à un autre moins puissant et vice-versa ; il faut simplement régler les paramètres multi-thread et choisir le nombre d'instances selon les ressources de l'ordinateur, pas besoin de changer l'interface.

Ensuite, il y a une meilleure économie de CPU parce que il n'y a jamais des instances actives inutilement grâce à l'architecture poly~.
Enfin, la stabilité est majeure parce que si une instance a un problème ou crashe, c'est difficile que l'entier système puisse se planter : le problème survenu reste local et n'empêche pas à l'application Modalys de tourner. Dans l'architecture 2008 si un Multi-instrument crashait il rendait impossible son utilisation en entière.

La stabilité de Modalys en temps réel est aussi améliorée parce que pendant le travail sur « L'Apparente » on a poussé aux limites les possibilités de cette approche particulière et Nicholas Ellis a pu améliorer, corriger et optimiser le logiciel en ayant pour cible l'utilisation efficace en temps réel.

EN CONCRET

L'architecture de contrôle de la synthèse est constituée de patches contenants des objets poly~ ; chaque poly~ charge un patch à l'intérieur du quel il y a un objet modalys~. Les objets modalys~ lisent le même script Modalys, par conséquent, au chargement, toutes les instances sont initialisées avec les paramètres initiaux du script et prêtes à être utilisées.
L'utilisateur communique avec le système à travers une « demande de note » qui vient d'une partition ou d'un message. Ce message peut être envoyé à partir d'un capteur du geste, un instrument acoustique, un clavier midi, une partition électronique pilotée par une pédale ou à la main, etc.
La demande de note doit contenir aussi la valeur d'amplitude qui est interprétée de façon différente selon la méthode d'amplitude choisie par l'utilisateur et le type d'excitation utilisée.
Dans cet article nous montreront cette architecture appliquée à un seul type d'instrument : une plaque. L'architecture est ensuite exportable à un instrument quelconque à 1 ou 2 dimensions.

On veut avoir à disposition une collection de plaques modulables en tous les paramètres constitutifs et secondaires et totalement gérable du point de vu de l'articulation musicale. Il faut aussi pouvoir modifier les paramètres des instruments soit localement soit globalement. Cela permet à l'utilisateur de diviser le nombre de plaques en plusieurs couches par exemple, selon ses exigences musicales.
Nous avons choisi la plaque parce que avec ce type d'instrument on peut obtenir une énorme variété de sons proches à nombreux instruments réels et matériaux : cloches, gongs, peaux, céramiques, bois, métaux, verres, plastiques, pierres, etc. Ces instruments peuvent être harmoniques ou inharmoniques.

Les problèmes à résoudre dans la création d'une architecture de contrôle pour instruments modaux sont :

1. l'initialisation des instruments
2. la gestion des dynamiques
3. les paramètres locaux et globaux
4. l'élimination des paramètres redondants
5. les queues de données inutiles
6. la gestion de la polyphonie.

Avant d'examiner ces problèmes, voyons le type d'instrument qu'on veut créer. Nous avons créé un instrument générique et simple : une plaque qu'on peut mettre en vibration dans plusieurs façons.
Le deux excitations fondamentales pour une plaque sont le frappement et la force.
La mise en vibration par force équivaut à secouer l'instrument en lui envoyant du son.
Le frappement en Modalys peut être réalisé avec deux méthodes (dits connexion ou interaction en langage Modalys : le Strike ou le Felt. Dans les deux cas on frappe la plaque avec une baguette simulée (un système avec deux masses et un ressort) ; toutefois le Felt est une modélisation de la percussion d'un marteau de piano et les paramètres de contrôle sont beaucoup plus fins. Par contre, elle est légèrement plus gourmande en termes d'utilisation de ressources de CPU.
Dans notre expérience l'interaction Strike est un peu moins riche en termes d'« épaisseur » du timbre, elle est plus figée et moins modulable au niveau du son. Pour varier le son issu d'une connexion Strike il faut changer la pente de descente du mouvement de la baguette, l'hauteur du mouvement ou le « poids » de l'interaction (nous expliquerons dans un moment le significat du « poids » dans le langage Modalys).
Nous avons essayé de modifier les paramètres de la baguette même (la souplesse du ressort, la dureté, la perte, etc.) mais ils ne changent pas trop le résultat sonore. Dans tous les essais avec le Strike, nous retrouvons un transitoire d'attaque trop figé et similaire même avec des dynamiques différentes, pour cette raison aujourd'hui on s'est adressés vers l'utilisation du Felt pour réaliser les frappements.

Comme pour « L'Apparente » les interfaces de contrôle ont été conçues avec trois buts : l'expérimentation, la mémorisation et l'exécution.
Les paramètres de contrôle d'un instrument virtuel sont innombrables et pas tous ont le même poids perceptif ; l'avantage du temps réel est que l'utilisateur peut jouer l'instrument modal comme un vrai instrument, écouter directement les résultats provoqués par les variations des paramètres, en comprendre le poids perceptif, trouver des situations musicalement intéressantes et les mémoriser pour l'utilisation future.
A différence de l'architecture de « L'Apparente » dans ces nouvelles interfaces, l'utilisateur peut mémoriser le paramètres avec des *presets* grâce au système du *pattrstorage*. Toutefois, c'est toujours possible utiliser des boîtes de message pour communiquer avec les instruments car tous les paramètres sont prêts à recevoir des valeurs à travers des send/receive.

Architecture poly~

Un instrument poly~ dans MaxMSP peut être conçu dans plusieurs façons. Après avoir essayé différentes méthodes de gestion des instances, nous avons enfin choisi la technique de gestion polyphonique qui utilise le message *midinote*.

Si on envoie à un poly~ le message *midinote* suivi par deux valeurs, représentant la note midi et la vélocité, le système allume les instances par lui même et l'utilisateur doit seulement « jouer » sans se préoccuper de quelle instance appeler. C'est une technique pensée pour jouer le poly~ avec un clavier Midi ou un fichier Midi. Le système « mute » les instances allumées quand il reçoit des « note-off » (un message *midinote* avec le numéro de note qu'on veut muter et la vélocité à zéro).

Pour utiliser ce système il faut employer une « partition » constituée par des note-on et note-off : par exemple une partition midi ou des classes musicales comme dans OpenMusic. MaxMSP est un logiciel qui peut lire des fichiers midi mais il manque totalement de classes musicales. Le problème peut être résolu en lisant des fichiers midi ou avec un suivi de partition en utilisant des objets FTM qui gèrent une partition midi ou en écrivant des fichiers coll à partir d'OpenMusic. Les données d'un fichier coll sont lues en utilisant un index, par conséquent il faut envisager des index avec les note-off pour les notes qu'il faut muter.

Initialisation et gestion des instances

Comme on a déjà signalé, un des problèmes majeures pendant le travail sur « L'Apparente » a été l'initialisation des paramètres de synthèse principaux des instruments. Les instruments Modalys ne peuvent pas être mis en vibration pendant l'initialisation, sauf les cordes qui sont un model physique moins gourmand.

En utilisant le message *midinote*, MaxMSP appelle une instance inactive à chaque « demande de note », l'instance « précédente » ainsi est libre de résonner selon ses paramètres de résonnance. Par conséquent, il n'y a plus le problème de changement de paramètre physique et de lien entre notes car elles sont réalisées avec deux instruments différents.

Avec cette architecture d'initialisation, par contre, se présentent deux problèmes : la gestion des instances lorsqu'on demande des notes répétées (*ribattuto*) et la gestion des note-offs par rapport aux paramètres de résonance de l'instrument.

Si on utilise le message *midinote* et on demande deux notes successives avec la même hauteur et en vitesse, le poly~ appelle pour la deuxième note une nouvelle instance car il n'a pas reçu clairement un note-off ou il est en encore en train de réaliser son enveloppe d'amplitude. Si on veut une séquence de 10 notes répétées vite, le poly~ peut allumer jusqu'à 10 instances selon le temps de résonance de l'instrument et la réception des note-off.

Il ne s'agit pas seulement d'un problème d'utilisation de CPU mais aussi d'une situation peu réaliste ; en fait, dans la réalité un instrument frappé plusieurs fois résonne de façon particulière, parce que chaque frappe excite un corps déjà en vibration, donc son timbre sera différent, modulé. Au contraire, dans un poly~ géré avec la technique du *midinote*, chaque instance à sa propre vie et sonne séparément des autres. En outre, si à chaque

note répétée on allume une nouvelle instance, la dynamique devient très forte car le niveau de chaque instance s'ajoute aux précédant, alors que une vraie plaque frappée plusieurs fois ne générera pas un doublement du niveau sonore mais un crescendo très particulier.
Pour résoudre ce problème, on a conçu un système de gestion des instances qui se rappelle des notes allumées et qui envoie la requête de note à l'instance correspondante déjà en résonance lorsqu'on demande une note de la même hauteur. Pour réaliser ce système on a crée un objet javascript similaire au midinote mais doté d'une mémoire d'un certain nombre de notes allumées. Le son aussi est plus similaire à la réalité physique car si on demande un ribattuto c'est le même objet physique qui vibre avec les mêmes effets de timbre et dynamique de la réalité physique.

Le deuxième problème est qu'il faut envoyer les notes-off seulement après que l'instrument ait fini de résonner, autrement le poly~ est muté et on peut entendre un clic et la résonance est coupé. Par conséquent, si on utilise un instrument avec une longue résonance et une partition Midi comme structure musicale, on peut avoir des problèmes, parce que les notes sont écrites avec des « durées de notation » (qui déjà ne correspondent pas trop aux durées de jeu) et pas avec les durées réelles de résonance de l'instrument.
Cela semble nous amener à changer les durées de la Structure musicale, ce qui peut être très ennuyant et surtout on aurait une Structure musicale qui dépendrait de l'instrument. Nous voulons conserver les Structures musicales avec les « durées de notation » qui restent les plus valables du point de vue de la partition symbolique. Ainsi, dans notre système l'instance contient une information de durée d'envoi du note-off qui doit être indiquée par l'utilisateur selon la résonance de l'instrument utilisée et qui retarde l'envoi du note-off. Une amélioration à réaliser serait de rendre le système capable de calculer tout seul cette durée selon le temps de résonance globale et la fréquence de l'instrument (en fait les instruments plus graves normalement résonnent plus longuement).

Dynamiques

Pendant le travail sur « L'Apparente » nous avons obtenu un ambitus dynamique des instruments de synthèse restreint par rapport a celui d'un instrument acoustique. Pour réaliser les dynamiques voulues en partition, nous avons varié dans la partition électronique les valeurs d'entrée des instruments virtuels en les prédisposant aux actions des musiciens. Pourtant, le contrôle des dynamiques était géré par eux seulement en partie.
L'ambitus dynamique était difficilement gérable dans le cas de l'interaction Strike ; en fait pour réaliser des dynamiques différentes dans un Strike il faut profiler de façon très fine le mouvement de la baguette en relation aussi à la taille de l'instrument.
Dans le cas de l'interaction Force les choses sont plus faciles mais il reste le problème de considérer la taille et le matériel de l'instrument.
On ne pouvait pas prendre avantage de toutes les variations fines du geste qu'un instrumentiste réalise pour obtenir une dynamique particulière ; en fait, un musicien modifie son geste en rapport à l'objet résonnant qu'il a en face et de l'objet excitateur. C'est une sensibilité développée avec des années d'expérience (pas seulement

instrumentale) et qui concerne un rapport tactile direct avec l'objet excitateur et le corps résonnant. Cette sensibilité est perdue lorsqu'on bouge les mains dans le vide pour produire des sons de synthèse (comme dans le cas du percussionniste – mime de « L'Apparente ») et aussi lorsqu'on frappe la même table en bois pour exciter des instruments virtuels qui simulent objets de différents tailles.
Pour s'approcher à ce type de sensibilité ou pour en développer d'autres encore plus fines (pourquoi pas ?), il faudrait concevoir des capteurs spécifiques où la grande sensibilité des mains puisse s'exprimer au plus haut degré. L'exemple du Theremin est pertinent : avec cet instrument, pour la première fois dans l'histoire de la musique, on peut produire du son sans le contact direct avec un objet mais en utilisant les mains dans l'air. Le principe est simple et les paramètres limités à deux. Une des premières et plus grandes solistes de Theremin a été, non par hasard, une violoniste : Clara Rockmore. Elle quitta le violon parce que fascinée par la possibilité d'exprimer toute sa sensibilité musicale directement avec ses mains sans la contrainte mécanique d'un objet intermédiaire entre le geste et le son. Dans ce cas, le nouvel instrument technologique propose une sensibilité plus fine dans les variations que celle d'un violon, car on est libéré de la contrainte de tenir dans la main l'archet
Si on voulait réaliser un travail approfondi à ce propos, il faudrait projeter des instruments « hardware » spécifiques qui puissent prendre plein avantage de la sensibilité des mains d'un musicien. Un instrumentiste est habitué à étudier et s'adapter mais il faut le mettre dans la condition d'entrevoir la richesse d'un nouvel instrument et le laisser pratiquer longuement.
Cela n'a rien à voir avec la « facilité » d'utilisation de l'instrument : le Theremin par exemple est un instrument terriblement difficile à maîtriser mais un musicien est habitué à faire un travail d'étude lorsqu'il voit le but de ses actions. C'est très important, donc, lui donner le plus de contrôle et du feedback possible par rapport à un nouvel instrument.

Dans « L'Apparente », nous avons choisi de moduler dans la partition électronique les niveaux d'entrée des instruments virtuels. En réalité dans Modalys il y a une technique plus fine pour varier la dynamique des sons mais nous n'avons pas voulu charger trop le moteur Modalys, déjà poussé au maximum en temps réel.
Pour utiliser cette technique il faut introduire le concept de « poids » d'une interaction. Le « poids » est un paramètre qui « adoucit » l'interaction et est applicable à toutes interactions Modalys. Ce paramètre varie entre 0 et 1 : s'il est à 1 l'interaction se comporte normalement ; en allant vers le 0 toutes les caractéristiques de l'interaction vont s'adoucir.
Dans le 2008 nous n'avons pas trop varié le « poids » alors que aujourd'hui il est devenu le paramètre fondamental pour moduler les dynamiques.
Une conséquence intéressante est que si on applique la variation de poids à un Strike ou un Felt, ce n'est pas seulement la dynamique qui varie mais aussi le timbre ; cela nous donne des résultats plus similaire à la réalité physique.
En mettant en relation le paramètre de vélocité Midi avec le poids (de façon linéaire ou exponentielle), on peut obtenir même avec l'interaction Strike une gradation dynamique crédible. Nous avons appliqué une relation similaire pour toutes les interactions de mise en vibration des instruments.

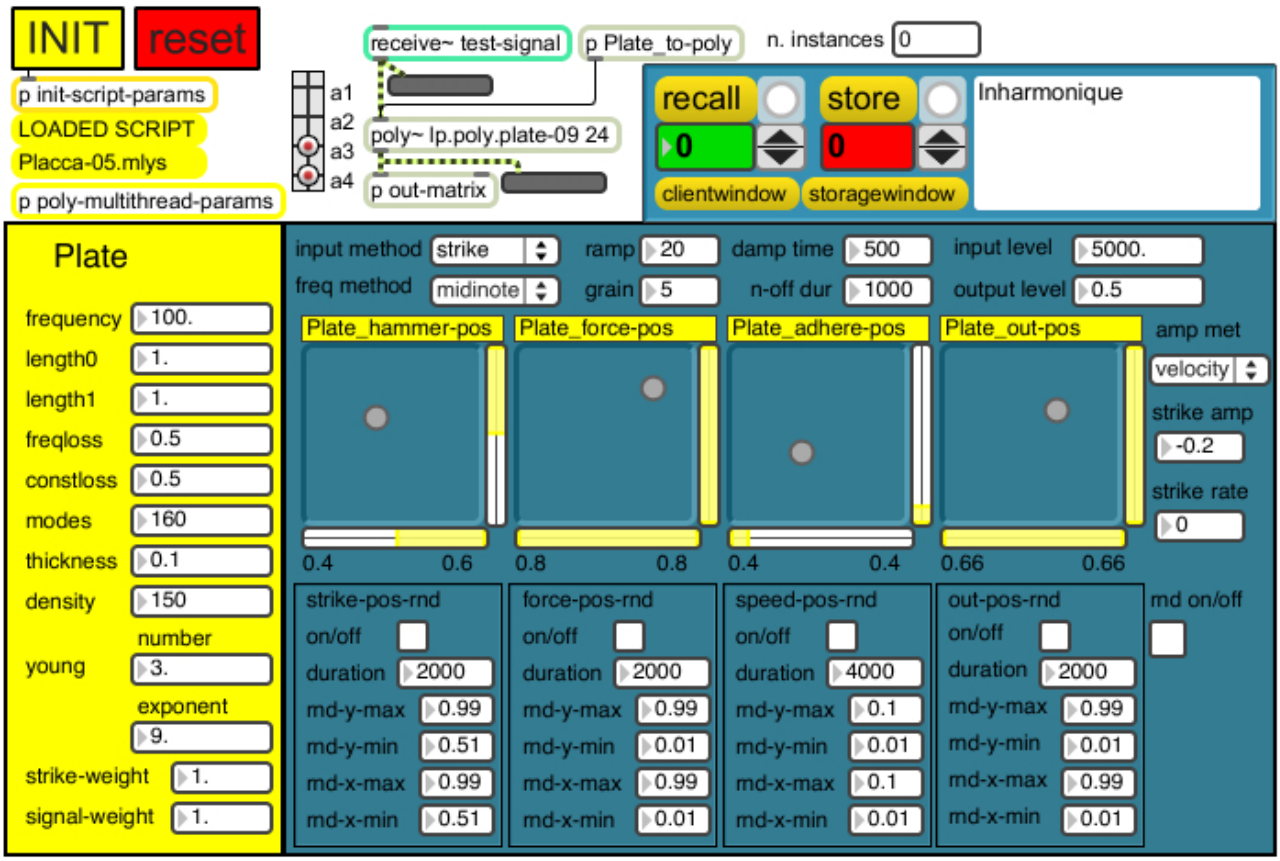
De plus, dans notre système c'est le geste de l'instrumentiste qui produit la vélocité midi selon la vitesse du geste, par conséquent le musicien devient totalement responsable

La dynamique de l'action de l'instrumentiste peut être utilisée pour moduler plusieurs paramètres ; par exemple dans le cas du Felt, qui est une interaction plus raffinée, nous avons module non seulement le poids de l'interaction mais aussi le paramètre F0 qui conditionne la force du feutre du marteau.

Le système donne aussi la possibilité à l'utilisateur de « peser » selon ses exigences la relation entre la vélocité et le poids d'une interaction en utilisant un minimum, un maximum et une courbure.

Interfaces

Dans la fig. 2 est montrée l'interface de contrôle globale de l'instrument Plate.



(Fig. 2 Interface de contrôle pour la plaque)

A Gauche sur fond jaune il y a tous les paramètres physiques constitutifs de l'instrument. Sur fond bleu on trouve les paramètres secondaires comme le point d'application d'une interaction (contrôlable aussi de façon random entre deux limites). Les interactions dans le cas de cet instrument sont : le point de frappement du marteau (Plate_hammer-pos) , la position d'excitation avec une force (Plate_force-pos), le point ou l'étouffoir

après la percussion (Plate_adhere-pos) utile pour l'interaction Felt et le point d'écoute sur la plaque (Plate_out-pos).
On trouve aussi des paramètres de contrôle de la méthode d'entrée (strike, felt ou force), de la fréquence (fréquence ou midinote, le temps de rampe et le grain pour toutes les interpolation internes de l'instrument (réalisées avec des *line*), le temps employé par l'étouffoir pour se mettre en place (damp time), le délai d'envoi du note-off par rapport au note-off de la Structure musicale, le niveau d'entrée et de sortie, la méthode pour l'amplitude (vélocité ou linéaire), l'amplitude du mouvement et la vélocité de descente de la baguette pour le Strike (ces dernières deux paramètres peuvent être mis en relation avec l'ampleur et la vitesse de mouvement du musicien pour lui donner le contrôle totale de la dynamique avec une connexion Strike).
En haut il y a le bouton d'initialisation et de reset de l'instrument, le subpatch qui gère les paramètres multithread du poly~ (on détaillera cet aspect plus tard), le poly~ contenant l'instrument (ici avec 24 instances), le subpatch Plate_to-poly qui contient l'objet javascript qui gère les instances et les notes répétées, enfin la partie de mémorisation réalisée avec le système du pattrsorage.

Tous les paramètres de cette interface sont globaux, par conséquent si l'utilisateur change une valeur elle sera appliquée à toutes les instances de l'instrument. Par contre, les variations random des interactions sur l'instrument sont calculées individuellement pour chaque instance pour ne pas avoir un timbre égal pour toutes.
Les paramètres sont gérables soit avec le système du pattrstorage soit avec des messages de partition et ils peuvent être locaux ou globaux. Ainsi, l'utilisateur peut toujours envoyer une valeur de paramètre à une instance particulière et diviser par exemple les instances en couches musicales selon les exigences de la partition.

Redondances

Dans l'utilisation de Modalys en temps réel c'est très important éliminer l'envoi des paramètres redondants ou inutiles. Pendant le travail sur « L'Apparente », on a vérifié que même si un objet modalys~ est désactivé les messages de contrôle sont conservés dans une queue de données en attente d'être exécutés. Lorsque on allume l'objet, l'entière queue des données lui est envoyée et cela crée une congestion dans l'exécution des instructions qui peut causer des sons non voulus.
Dans notre architecture les objets modalys~ à l'intérieur du poly~ ne reçoivent aucune donnée s'ils ne doivent pas jouer, car les messages de contrôle sont totalement filtrés si l'instrument est inactive.

Articulations

Avec ce type d'architecture on n'a plus de problèmes d'articulation musicale et d'initialisation ; en fait chaque note est réalisée par une nouvelle instance. Sont donc réalisables des gestes musicaux impossibles seulement l'année dernière et il reste toujours possible l'exploration fine du timbre à l'intérieur d'une seule instance.

Types d'instruments

Cette architecture est un exemple élémentaire exportable à une série d'instruments différents. On peut l'appliquer à presque tous les corps vibrant Modalys : membranes, plaques circulaires, objet 3D, cordes, etc.
Pour d'autres modèles, comme les instruments à vent, l'approche serait totalement différente et il y a des recherches déjà abouties au sein de l'Ircam. Pour cette raison on s'est dédié plutôt aux modèles qui impliquent une utilisation polyphonique ou pour lesquels l'utilisation polyphonique en temps réel était un problème a résoudre.

Paramètres multi-thread et utilisation de CPU

Comme on a dit, MaxMSP 5 est capable d'exploiter les ressources d'un ordinateur multi-core lorsque du calcul audio si on utilise le système du poly~ et l'architecture concernée. Pour fonctionner proprement, le poly~ a besoin que l'utilisateur indique le numéro exact de cores de l'ordinateur utilisé et s'il doit travailler en mode parallèle.
Ce n'est pas seulement l'utilisation de CPU qui peut donner des problèmes à Modalys mais surtout les pics improvises de demande de calcul ; ils peuvent provoquer des brèves coupures dans l'audio.
Nous avons travaillé et effectué des testes sur deux modèles d'ordinateurs Apple : un MacBook Pro avec un processeur Intel Core 2 Duo 2.5 GHz (Penryn) et un Mac Pro avec un processeur Intel Xeon 8-core 2.8 GHz (Penryn). On a utilisé premièrement le système opératif OSX 10.5.8 et ensuite OSX 10.6.2 Snow Leopard.
Soit avec le Core 2 Duo soit avec le Xeon 8-core, nous avons vérifié qu'en mode parallèle tous les processeurs sont employés en quantité égale. Cependant, on a remarqué que le système n'arrive pas à utiliser plus que le 82-85% des singles cores et si on demande plus d'instruments ou de variations lorsqu'on est au 85% se présentent des clics audio typiques des pics d'utilisation de CPU.

Pour donner une idée de la quantité d'instances simultanées possibles, voici quelques données. Avec un ordinateur portable Apple doté d'un Core 2 Duo 2.5 GHz, nous avons excité avec une Force jusqu'à 20-21 plaques dans le registre medium aigu sans bouger les points d'interaction. Le signal vector size était à 512 échantillons, ce qu'il donne une latence acceptable.
La même situation instrumentale portée sur un ordinateur Apple doté d'un 8-core 2.8 GHz, rend possible de synthétiser jusqu'à 80-85 instances : quatre fois plus par rapport à l'ordinateur portable, en proportion exacte avec le numéro de cores disponibles.
Bien évidemment, le numéro d'instances possibles dépend des paramètres des plaques synthétisées : le numéro de modes, le registre et les paramètres de résonnance. Toutefois les possibilités polyphoniques par rapport au 2008 sont dramatiquement augmentées, ce qui permet une utilisation plus agile des instruments.
Nous n'avons pas encore testé l'architecture sur des ordinateurs multi-core Nehalem qui ont des benchmarks multi-thread incroyablement supérieurs aux Penryn utilisés jusqu'à maintenant. Les benchmarks de ce type de machines montrent que la puissance d'un 4-core Nehalem est légèrement inferieure à celle d'un 8-core

Penryn : presque le même résultât avec la moitié de cores. En fait, dans l'architecture Nehalem chaque core gère deux threads virtuels et il serait intéressant pour nous tester l'efficacité de cette double voie de calcul.

Nous avons aussi vérifié que le nouveau système opératif récemment introduit par Apple (*Snow Leopard* OSX 10.6) gère mieux les ressources multi-core. Avec *Snow Leopard* notre architecture instrumentale utilise entre 5 et 10% en moins de ressources CPU(!). Il s'agit d'une amélioration remarquable parce que due exclusivement d'une mise à jour du système opératif.

Autres économies de ressources CPU

Une autre technique pour réduire l'utilisation de CPU est de neutraliser les interactions contenues dans le script chargé par l'objet modalys~ pendant les passages musicaux où elles ne sont pas utilisées. En fait, même si pour exciter l'instrument on utilise seulement une seule interaction, le moteur est toujours prêt à synthétiser toutes les autres. Dans Modalys il n'y a pas une fonction « freeze » pour les interactions ; la méthode plus simple est de préparer des Scripts contenant seulement une interaction pour les passages particulièrement chargées de demande de CPU.
Le chargement d'un script est aujourd'hui beaucoup plus stable par rapport à autre fois et on peut imaginer de faire lire des scripts différents mais tous compatibles pendant le déroulement d'une pièce.

Classes musicales

Pour mieux utiliser le système il faut penser à une technique pour stocker les Classes musicales de notre architecture. On imagine des collections de séquences de note-on et note-off HORS du temps et accessibles à la lecture par le musicien. Il faut rappeler que toutes les vélocités midi doivent être générées à partir du geste de l'instrumentiste en temps réel.
Nous avons déjà détaillé des méthodes simples de lecture de données musicales dans l'article « Synthèse par Modèles Physiques en temps réel : Expériences ». Pour une partition plus complexe et un suivi plus sûr de la partition, on pourrait utiliser par exemple un système de suivi de partition appliqué aux gestes des instrumentistes associé avec une partition de données musicales. Ce système doit permettre d'accepter les vélocités instantanées provoquées par l'instrumentiste.

Contrôle instrumental de la synthèse

Maintenant, les recherches sont face au problème de l'interface physique à utiliser pour contrôler le moteur de synthèse car il est assez stable. L'instrument plus facile à employer est un clavier Midi mais ce n'est pas notre intérêt d'utiliser ce type d'interface.
Nous avons réalisé des expériences de contrôle d'instruments virtuels en utilisant des microphones à contact fixés sur des plaques en bois. Cela donne un contrôle assez efficace et direct de l'instrument, ce qu'un musicien souhaite le plus.
Nous avons utilisé aussi des capteurs sans fils capables de capter 6 paramètres d'accélération ; avec ces capteurs l'instrumentiste était visiblement engagé et pouvait jouer des instruments virtuels avec les mouvements des mains dans l'espace. C'est cette dernière voie qu'on voudrait entreprendre : on voudrait projeter une interface de contrôle qui permet à un musicien de jouer ces instruments librement avec ses gestes dans l'air. Aujourd'hui la technologie nous permet de développer des interfaces de contrôle pareilles et il y a déjà beaucoup de chemins ouverts : capteurs sans fils, accéléromètres, cameras, cameras infrarouges, etc.

CONCLUSIONS

Nous avons donné une solution aux principaux problèmes de gestion des instruments virtuels pour lesquels l'utilisation polyphonique est essentielle. L'architecture conçue est beaucoup plus stable par rapport à « L'Apparente » qui a été une première phase importante de recherche. Aujourd'hui, elle peut être sereinement employée en concert.
Maintenant on peut créer une grande polyphonie, réaliser librement des articulations musicales et faire contrôler l'amplitude des sons directement par les interprètes de façon plus réaliste et efficace. En un mot le système est devenu plus instrumental et moins « piloté » de l'extérieur par des partitions automatiques.
Pour accomplir pleinement cette transformation et aboutir ces recherches, maintenant on est face à une urgence : il faut mettre en relation cette architecture avec les nouvelles technologies et les méthodes de captation du geste. En fait, si on arrivait à donner au musicien le contrôle gestuel des instruments virtuels on réaliserait un instrument riche de possibilités musicales et d'implications visuelles qui peut être utilisé et expérimenté concert.
Après avoir accompli un système de synthèse totalement gérable par les gestes, on pourrait passer aux recherches strictement musicales et instrumentales avec un musicien qui devrait s'impliquer à créer une (ou sa propre) pratique instrumentale. Les gestes pourraient avoir aussi des implications visuelles très fortes car le musicien ne toucherait aucun objet mais il ne faut pas oublier que la motivation principale des gestes doit être instrumentale et musicale.
Selon la captation du geste employée, les mouvements pourraient être plus ou moins « scéniques » ou fonctionnels. Dans le cas du Theremin, les gestes sont totalement fonctionnels car il y a seulement deux paramètres et l'interprète ne doit pas bouger le corps pour ne pas modifier le son avec ce mouvement. Par contre, si dans notre système on utilisait des capteurs sans fils, le corps du musicien pourrait bouger assez librement sur la scène et ouvrir toute une série de possibilités scéniques - visuelles intéressantes à explorer.