UPMC - PUCE MUSE

Intégration des œuvres musicales composées et création de nouvelles œuvres pour les architectures logicielles développées à Puce Muse

---

Rapport de Stage

Alexandre Panizzoli

Master SDI Acoustique – Année 2010/2011

Stage encadré par Guillaume Bertrand guillaume.bertrand@pucemuse.com
Espace musical - PUCE MUSE 2, rue des Pyrénées, SILIC 520 - 94623 Rungis

Résumé du Stage

Depuis sa création en 1982, l'association Puce Muse oriente ses activités de Création, Recherche, Pédagogie et Diffusion vers la *M3V* : Musique Vivante, Visuelle et Virtuelle. Cette structure hybride (oscillant entre recherche pratique et création artistique) travaille en étroite collaboration avec plusieurs laboratoires renommés dans les domaines de l'acoustique et de l'informatique musicale (LAM, IRCAM, GRM,...).

Les thématiques de recherches ont amené au développement de nombreuses interfaces homme-machine innovantes, tant matérielle (Méta-Instrument) que logicielle (Méta-Malette), permettant le jeu collectif de musiques électro-acoustiques grâce à l'utilisation de contrôleurs de jeux vidéo.

C'est dans ce cadre que j'ai été amené à développer pour la plateforme multimédia *La Méta-Mallette* à l'aide du logiciel *Max/MSP/Jitter*, qui permet d'allier facilement son et image tout en gardant jouabilité et esprit de pédagogie comme ligne directrice de programmation. Mon travail consistait en l'élaboration de nouveaux instruments, la recherche de nouveaux algorithmes pour l'amélioration, l'analyse et la synthèse audio pour le *time-stretch* et le *pitch-shift*, le portage et versionnage d'instruments audio/vidéo/graphique3D existants.

Abstract

Since its creation in 1982, the voluntary association Puce Muse focuses its activities of Creation, Research, Pedagogy and Distribution on the LM2V: Live Visual and Virtual Music. This hybrid structure (half-way between practical studies and artistic creation) works in collaboration with various renowned laboratories which lead researches on acoustics and computer music (LAM, IRCAM, GRM,…).

These studies led to the development of many innovative Human-Computer Interfaces, such as the Meta-Instrument (complex mechanical-MIDI controller) or the Meta-Mallette (musical software), all allowing the cooperative playing of electro-acoustic music with the use or gamepads.

It is in this original framework that I was requested to develop for the multimedia platform *La Meta-Mallette* with the help of Max/MSP/Jitter software, which allows to link easily sound and pictures, with gameplay and pedagogy as a guideline. My internship work consisted in the design of new instruments, the research of audio enhancements algorithms for time-stretching and pitch-shifting as well as the software versioning of existing audio/video/CGI instruments.

Remerciement

Je remercie Serge de Laubier de m'avoir accueilli dans les Studios Puce Muse pour ce stage et de m'avoir guidé tout au long de celui-ci.

Merci à Guillaume Evrard pour mes premiers contacts à Puce Muse et à Guillaume Bertrand, mon tuteur, pour son accueil, nos échanges divers, ses conseils et son aide tout au long du stage.

Merci à Suguru Goto pour son aide, Emmanuel Soccodato, Kevine Hette, Amélie Piron, Thierry Jamet, Daphné Bakker et Raphaël Kurtag pour leur convivialité, leur ouverture d'esprit et leurs nombreux conseils.

Enfin, merci aux stagiaires Xavier et François-Xavier qui ont partagé ces moments très enrichissants au sein de « l'espace musical ».

Sommaire

Introduction

J'ai découvert PUCE MUSE par le biais d'un ami de ma promotion de Master Acoustique qui les a vu à de nombreuses reprises dans le cadre de leur résidence à l'UPMC, au cours de laquelle à été mis en place le Méta-Orchestre et pendant les *Nuits Blanches* à Paris 2010 (Eglise St Merry). J'étais déjà passionné par les arts visuels et la musique électronique moderne, mais mon intérêt pour leurs activités, plus portées sur la musique électro-acoustique contemporaine, m'a vraiment imprégné tout au long du stage. En effet, l'interactivité, la relation entre son et image, l'étude et la recherche de nouvelles interfaces pour le contrôle et le geste musical, sont des thèmes centraux des recherches et créations à Puce Muse qui me tiennent à cœur.

Mon sujet de stage « Intégration des œuvres musicales composées et création de nouvelles œuvres pour les architectures logicielles développées à PUCE MUSE » couvre volontairement un large domaine et reflète la diversité des tâches accomplies durant ces quinze semaines de stage.

Après un entretien dans leurs locaux de Rungis (zone industrielle SILIC), j'ai été accepté comme stagiaire en développement sur un logiciel qu'ils ont crée : la Méta-Mallette. Celle-ci permet le jeu interactif et collectif de musique électronique, la synthèse sonore et visuelle étant contrôlée par des interfaces classiques de jeu vidéo : joystick, Wiimote, gamepad, iPhone, Launchpad...

De manière générale, un stage à PUCE MUSE ne suit pas un sujet unique tout au long du stage, mais implique en premier lieu une familiarisation aux outils utilisés, notamment le logiciel Max, et à l'architecture logicielle de la Méta-Mallette. Les objectifs sont ensuite définis en fonction des besoins à court ou moyen terme de l'association, qui évoluent constamment avec les commandes et créations.

PUCE MUSE invite également chaque stagiaire à participer aux événements organisés, ce qui offre une ouverture aux créations artistiques et représentations qui sont l'envers du travail du développeur.

Ainsi, après avoir étudié la totalité des tutoriels Max/MSP/Jitter et m'être familiarisé à l'univers de la Méta-Mallette, qui requiert environ 3-4 semaines de formation en autonomie mais avec de nombreux conseils de Guillaume Bertrand, Suguru Goto et Serge de Laubier (que je remercie encore une fois ici) j'ai effectué le portage d'un premier instrument pour la version 4.0 de la MétaMallette: MM.Copland[1], un instrument qui utilise des marqueurs permettant de jouer des samples[2] à des index désirés et préenregistrés.

Mon stage s'est ensuite orienté vers le versionnage et l'amélioration d'un instrument (MM.Bach, utilisé pour des projections architecturales) basé sur la projection d'un plan sphérique ou planaire, texturé et en 3 Dimensions. Ce projet motivant m'a permis d'aborder le traitement d'image et notamment la programmation en OpenGL rendu intuitive, mais toujours complexe, par l'utilisation d'objets Jitter.

[1] Tout synthétiseur Méta-Mallette se doit d'avoir un nom commençant par « MM. », qui se prononce « MéMa », ou bien commençant par les initiales de son programmeur.

[2] Un **échantillon** (*sample* en anglais) est un extrait de musique ou un son réutilisé dans une nouvelle composition musicale, souvent joué en boucle. L'extrait original peut être une note, un motif musical ou sonore quelconque. Il peut être original ou réutilisé en dehors de son contexte d'origine. [Wikipedia]

Une fois les notions de paradigmes Max et d'architecture Méta-Malette maîtrisées, le second chantier proposé par Serge de Laubier était celui de l'implémentation d'algorithmes pour l'amélioration audio. Ce thème de recherche m'a permis d'approfondir mes connaissances sur la synthèse sonore, notamment la synthèse par l'utilisation de transformée de Fourier, mais plus généralement sur le travail d'analyse et de reconstruction d'un signal afin de préserver le maximum d'informations (amplitude, phase et notions de résolution spectrale et temporelle) et de minimiser le temps et le coût de calcul. Le but final était l'implémentation d'algorithmes de transposition fréquentielle (pitch-shifting) et d'étirement temporel (time-stretching) pour la plupart des instruments existant déjà au sein du logiciel Puce Muse.

Ce projet a abouti à la création d'un instrument MM.Freezer permettant également le contrôle et l'affichage d'une FFT en 3D et reprenant le mécanisme d'affichage des pointeurs de MM.Copland mais sur une implémentation totalement repensée et retravaillée pour l'interaction visuelle.

Enfin, mon dernier travail à Puce Muse s'est essentiellement concentré sur la préparation et le déroulement des spectacles dans le cadre du Festival Chalon dans la rue à Chalon-sur-Saône.

I. Présentation de la Structure : Puce Muse

1. Statut Légal

PUCE MUSE est une association loi 1901 basée à Rungis (94) dans la zone industrielle SILIC, dont les activités portent sur la création et la diffusion de la M3V : Musique Vivante, Visuelle et Virtuelle. Ses objectifs suivent trois axes : la création musicale et sa diffusion (concert, spectacles audiovisuels,...), l'éducation (travail avec des enseignants d'écoles de musique, collège,..), et enfin la recherche par le développement logiciel en synthèse sonore et visuelle contrôlée en temps réel.

L'association est financée principalement par la D.R.A.C. (Direction régionale des affaires culturelles) Île-de-France et le conseil régional d'Île-de-France. Elle se doit donc d'assurer un service public en matière de recherche et de sensibilisation à de nouvelles formes d'art multimédia. L'activité principale est la création d'œuvres associant musique, image et nouvelles technologies

2. Organigramme

La direction artistique est confiée à Serge de Laubier, qui est à la fois compositeur, interprète et chercheur. C'est le principal créateur des spectacles Puce Muse et il assure également l'orientation des sujets de recherche et développement.

L'organigramme en *figure 1* présente la répartition entre les trois pôles (Recherche et développement, Pédagogie, Création et diffusion) et les liens existants entre le personnel :

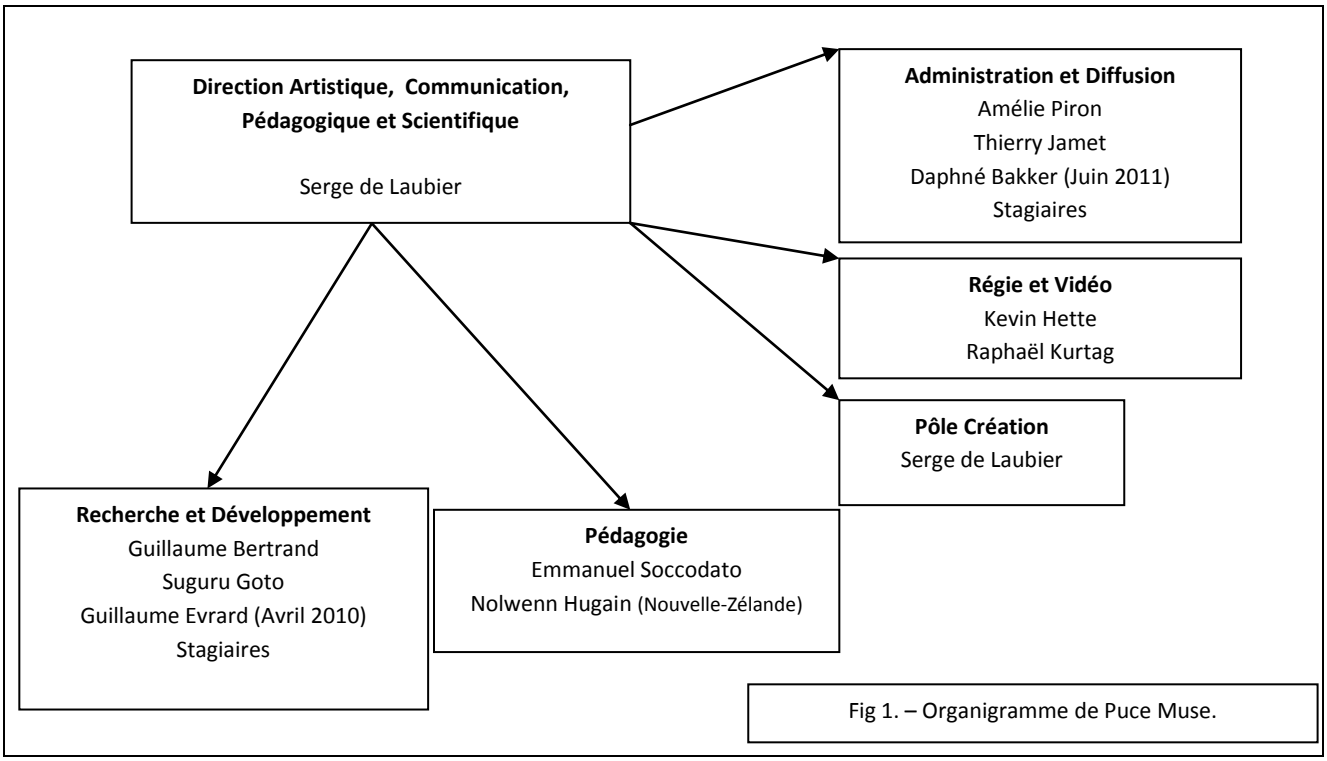


Fig 1. – Organigramme de Puce Muse.

En plus des permanents salariés de Puce Muse, il existe également un conseil d'administration dont les statuts sont les suivants :

- Création Musicale: François Bayle
- Arts de la Rue: Pierre Sauvageot
- Recherche : Hugues Genevois
- Pédagogie : Bernard Soules
- Arts Visuels : Catherine Hospite

Puce Muse tient également à rencontrer et travailler avec d'autres structures, artistes et pédagogues. Elle effectue donc de nombreuses commandes auprès d'artistes intéressés.

Ainsi, en plus d'accueillir des artistes en résidence, la compagnie collabore à la création de nombreuses œuvres, notamment de *La Grande Fabrique* (Haute-Normandie) ou bien garde des liens étroits avec son ancienne membre Nolwenn Hugain qui développe désormais son projet NOP en Nouvelle-Zélande.

Enfin, l'association participe à de nombreux événements culturels et artistiques, notamment au sein de l'UPMC, pour laquelle des UE ont été mis en place depuis 2008 pour divers cursus artistiques et optionnels.

3. Historique

En 1982, Serge de Laubier fonde l'Espace Musical - PUCE MUSE [1]. Son activité porte alors sur la recherche autour de la spatialisation du son. Cela débouchera en 1986 sur un brevet pour l'invention du Processeur Spatial Octophonique. En 1987 débutent les recherches sur le Méta-Instrument (fig. 2), dans le cadre du développement 2PIM (Plate-forme de Programmation Interactive Multimodale).

Fig 2. – Dernière version du Méta-Instrument.

Le Méta-Instrument est une sorte de gros Joystick qui permet de manipuler 54 variables simultanément et indépendamment les unes des autres (soit l'équivalent de 27 souris d'ordinateur). La première génération, plus modeste, a été utilisée en concert en 1989.

En quinze ans environ, 150 logiciels ont été développés pour lui (Serge de Laubier, Francis Faber, Mathieu Constant, Gyorgy Kurtag, Bernard Parmegiani, Pierre Sauvageot …), associant les gestes des musiciens à la manipulation de processus audio mais aussi lumineux, graphiques ou mécaniques.

Trois prototypes, compatibles ascendant, ont été développés en 1989, 1996 et 2004. Sa technologie fait appel à des développements électroniques, informatiques, mécaniques et musicaux spécifiques. La troisième génération du Méta-Instrument est un mesureur de gestes de grande précision. C'est une interface Homme-Machine mobile, de haut niveau, qui suscite un geste expert proche de celui du chef d'orchestre [1].

Ainsi, depuis la naissance de l'association en 1982, de nombreux projet ont vu le jour, parmi lesquels on peut citer :

- Le Méta-Instrument
- La Méta-Mallette
- La 2PIM
- Le logiciel Méta-Graphique
- Le Cabinet des Méta-Curiosités
- Le Guirlandophone
- Le Méta-Orchestre (Orchestre de 20 joueurs de joysticks auquel j'ai personnellement participé dans le cadre du festival Chalon dans la Rue)
- Le Concert de Concert

Depuis 2002, la Méta-Mallette et le Méta-Instrument sont en évolution ininterrompue et ont permis de créer des spectacles de plus en plus innovants, parmi lesquels la *Traverse de façade* (son et image contrôlés par le Méta-Instrument, l'image s'adaptant à l'architecture du bâtiment sur lequel elle est projetée) ou plus récemment le *Concert de concert* dans lequel jusqu'à 48 joueurs, bien souvent du public, peuvent jouer ensemble de la M3V (Musique Vivante Visuelle Virtuelle) à l'aide de manettes de jeu ou « gamepads », le *XXX360* (performance avec Méta-Instrument et orchestre de « joysticks », avec projection de l'image créée par les musiciens sur un espace à 360 °) et dernièrement *La Grande Pictophonie 3D* (projection architecturale 3D anaglyphe, concert de joystick Méta-Fanfare, Méta-Orchestre et Méta-Instrument).

4. Mission

Les objectifs de PUCE MUSE sont variés, nombreux et changeants : ils doivent tenir compte de l'évolution permanente des technologies tout en restant dans une démarche artistique. La dynamique de PUCE MUSE ne relève pas d'une logique d'entreprise fondée sur le profit mais sur la créativité artistique.

Comme on l'a vu précédemment, le Studio s'articule autour de 3 pôles principaux (Création, Recherche et Pédagogie), autour desquels diverses activités vont se greffer afin d'assurer la pérennité de l'association :

- Création : activité principale et artistique de laquelle découlent toutes les autres, mais qui tend à être remplacé (ou égalé) par l'activité de Développement.
- Recherche et Développement : principalement sur les plateformes Méta-Instrument et Méta-Mallette, mais aussi sur les contraintes technologiques imposées par chaque nouvelle création de concert (spectacle 3D Relief,etc...).
- Pédagogie : utilisation de la Méta-Mallette lors de nombreuses interventions en milieu scolaire ou public handicapé pour sensibiliser le public jeune et moins jeune à la musique électronique et contemporaine, en offrant la possibilité de jouer soi-même à l'aide d'interfaces comme le « joystick » ou le « gamepad » les œuvres composées par Puce Muse.
- Production et Diffusion : Direction Artistique, Communication, Recherche de financements,...
- Administration : incluant entre autres la gestion des contrats, droits d'auteurs...
- Régie Technique : Technique Audio, Vidéo et Lumière, pour les tests des configurations logicielles et matérielles pour les concerts.
- Vidéo : création d'objets, textures, images pour la Méta-Mallette, enregistrements et montages des prestations données par Puce Muse et des tutoriels d'apprentissage pour la Méta-Malette.

II. Plateforme et environnement de développement

La solution logicielle retenue par Puce Muse – que j'ai utilisé tout au long de ce stage - est un langage objet spécifique dont l'environnement de développement et de création répond au nom de *Max/MSP/Jitter*. Il s'agit d'un logiciel musical permettant de faire de la synthèse sonore, graphique (Video QuickTime et 3D OpenGL) et visuelle (module d'intégration Javscript, Java et Flash) de l'analyse, de l'enregistrement, ainsi que du contrôle d'instrument par protocole MIDI et réseau (intégration OSC, protocol TCP/UDP). Il a été développé par l'Ircam dans les années 1980, et est l'un des logiciels musicaux parmi les plus utilisés tant par les musiciens professionnels que par les amateurs[3]. La version 5.1.8 actuelle, proposée par Cycling74' permet également l'intégration logicielle pour le logiciel Ableton Live (MAX4LIVE). La version 6 est prévue pour la rentrée 2011, ce qui a notamment soulevé des interrogations à Puce Muse sur de futures évolutions et améliorations possibles pour la Méta-Mallette.

1. Max/MSP/Jitter

MAX est un environnement de programmation graphique permettant d'effectuer des opérations mathématiques et de contrôler des signaux MIDI en temps réel. MSP est la bibliothèque d'objets qui lui est associée, et qui permet le traitement du signal audionumérique de manière robuste et performante par l'utilisation de la bibliothèque de fonction DSP pour le traitement Audio.

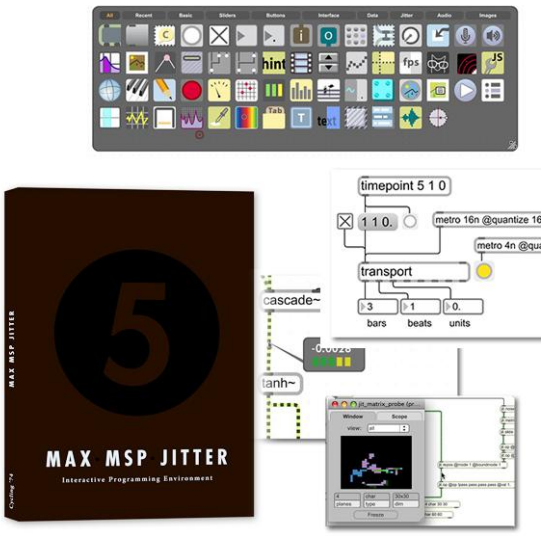


Fig 3. – Logiciel, Interface, et Objets Max

Enfin, Jitter est une bibliothèque supplémentaire de fonctions ajoutée à Max (comme l'est MSP), permettant de travailler sur des matrices. Son champ d'application est par conséquent multiple : traitement d'image en temps réel, le plus fréquemment relié à la pratique du Vjing[4], mais également synthèse audio matricielle (spectrographies FFT), matriçage mathématique ou encore modélisation matricielle 3D.

Le mode de fonctionnement et le paradigme de MAX/MSP/Jitter est tout à fait original puisqu'il consiste à interconnecter des boîtes de fonctions (objets, messages, attributs, nombres, etc.) avec des fils (représentant soit des valeurs numériques, soit une émulation de signal analogique). Ce mode de programmation temps-réel entraine donc des raisonnements algorithmiques qui peuvent différer fortement d'autres langages compilés tels que le langage C, Python, Java ou Matlab, ce qui rajoute l'avantage de permettre de prototyper rapidement et de voir instantanément l'effet d'un changement.

[3] Définition donnée par le site de Cycling74

[4] Acronyme de Vidéo-Jockey, qui est à l'origine d'une animation visuelle projetée, par affiliation au Disc-Jockey.

2. La Méta-Mallette

a. Objectifs

La Méta-Mallette est un dispositif de jeu musical collectif labellisé RIP (Reconnu d'Intérêt Pédagogique). Contrairement au Méta-Instrument qui explore les limites de la performance individuelle et multimédia, la Méta-Mallette est un dispositif permettant à tous - sans aucune connaissance préalable en musique, musique assistée par ordinateur, technologie ou informatique - de jouer, composer, tout seul ou à plusieurs, de la M3V. Elle est constituée d'une bibliothèque d'instruments audio-graphiques (produisant du son et de l'image en temps réel) qui peuvent être contrôlés par des interfaces telles que des « joysticks », des «gamepads», des tablettes graphiques ou des souris tout cela via une structure logicielle dédiée optimisée et développée sous Max/MSP [1].

On peut brancher autant de haut-parleurs et d'écrans que l'on souhaite en sortie de la Méta-Mallette, la configuration « classique » étant 12 haut-parleurs et 3 écrans. L'image produite en parallèle avec le son ne doit pas être perçue comme secondaire, au contraire elle amplifie l'écoute des auditeurs, des joueurs, voire des *spectacteurs* (néologisme soulignant l'abolition des frontières entre l'artiste se produisant face à des spectateurs inactifs).

Fig 4. Présentation du Méta-Orchestre dans une école

Fig 5. Répétition de la Méta-Fanfare aux Abattoirs de Chalon-sur-Saône

L'objectif de Puce Muse est d'utiliser cet environnement à tous les niveaux de gestion techniques durant les concerts et les ateliers pédagogiques. La prochaine version (Version 4.0 disponible en octobre 2011), devrait permettre une gestion totale des spectacles, en intégrant notamment le contrôle du Méta-Instrument, la gestion des diffusions en régie Lumière et Son, la projection Vidéo, le Mapping Architectural et bien entendu la production et le jeu en direct par des musiciens et orchestres (professionnels ou amateurs).

b. Description de l'interface

Voici ici une copie d'écran du menu principal de l'interface de la Méta-Malette et qui synthétise bien l'architecture et la conception du logiciel. Nous reviendrons plus en détail sur les instruments MéMa par la suite (cf. Partie III. Projet de Stage).

Fig 6.- Interface principale de la Méta-Malette

La Méta-Malette possède une interface très originale et intuitive par rapport à de nombreux logiciels de musique assistée par ordinateur. La logique de navigation est différente et intrigante au premier abord, mais elle à été pensé pour une utilisation du plus grand nombre. Elle se distingue notamment par sa partie centrale (cf. Fig 6. - Grande fenêtre de droite) où il est question de pupitres permettant de charger des instruments MéMa (16 sont visibles sur la photo, mais seulement les deux premiers contiennent des instruments).

Une fois l'instrument chargé, on peut configurer simplement le type de périphérique pour le jeu (manette, joystick, etc.), choisir les presets de réglages préenregistrés, l'affichage vidéo, les réglages audio, le métronome, et bien sur la gestion des projets existants. Une aide et des outils pour l'enregistrement et la configuration de la machine et des périphériques externes est également accessible.

La petite fenêtre en haut à gauche est destinée à accueillir le rendu vidéo généré par les différents instruments, qui sont codés à l'aide de Jitter.

Enfin, on peut accéder à un réglage plus fin des instruments en cliquant sur le pupitre correspondant et accéder ainsi à son interface de gestion des ***presets*** (par exemple réglage d'une texture vidéo, ou chargement d'un sample audio,...). Il est également possible de modifier le ***mapping*** (assignation des variables de contrôle de l'instrument à celui de l'interface de jeux type joystick ou autre) ainsi que le ***routing*** de chacun des instruments (contrôle des bus d'entrée/sortie audio et vidéo).

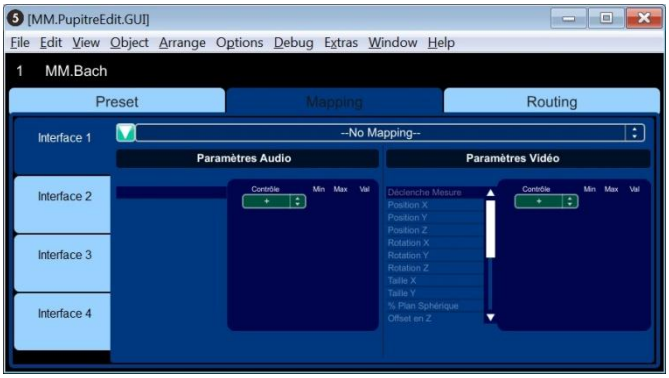


Fig 7. - Interface de gestion du Mapping

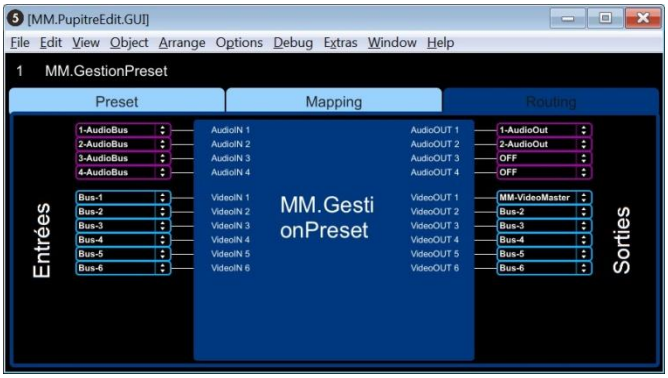


Fig 8. - Interface de gestion du Routing

Il faut noter que la gestion du traitement des données se fait de manière indépendante de la gestion de l'interface. Les développeurs ont donc mis en place de nombreux outils (abstractions Max pour le SDK), facilitant ainsi la sauvegarde et l'envoi des données entre l'interface de contrôle et le traitement des informations.

III. Projet de Stage

1. Intitulé

"Intégration des œuvres musicales composées et création de nouvelles œuvres pour les architectures logicielles développées à Puce Muse."

Domaines:
Informatique, traitement du signal, synthèse et représentation audio/vidéo/graphique sous Max/MSP.

Travail à effectuer:
« Lors de son stage, le stagiaire sera amené à effectuer divers développements autour des architectures 2PIM et Méta-Mallette dans l'environnement Max/MSP/Jitter. Il pourra s'agir de mettre en place de nouveaux instruments audio/graphiques, ou de travailler dans la continuité de développements antérieurs. Ces développements feront appels à des connaissances variées, concernant aussi bien l'informatique générale (réseaux, programmation Max/MSP/Jitter/java/javascript/C, php/html...), les connaissances musicales (instruments, partitions, MIDI, OSC...). »

« La participation aux manifestations données par PuceMuse est également à envisager. »[5]

Plus concrètement, mon travail au cours de ces 15 semaines de stage s'est essentiellement déroulé selon 2 parties distinctes entre versionnage et recherche, reliées tout de même par la création et l'insertion de mes recherches et divers travaux de développement au sein d'instruments Méta-Mallette.

Une mention prévoyait également ma participation aux évènements Puce Muse et j'ai ainsi pu effectuer plusieurs mises en place de spectacles en tant que technicien ou assistant régisseur, mais également comme joueur de Joystick au sein de la Méta-Fanfare (la description technique d'un chariot de secours, que j'ai moi-même monté en amont du spectacle, est consultable en Annexe B).

Ainsi, après m'être familiarisé avec l'environnement Max et Méta-Mallette, j'ai mené une longue recherche de fond sur les algorithmes et techniques existantes pour l'amélioration audio. Il m'a fallu environ un mois pour passer en revue les nombreux tutoriels Max-Msp-Jitter et appréhender l'architecture et la *surcouche* Max développée spécifiquement pour la Méta-Mallette.

2. Portage et Intégration d'instruments

Comme je l'ai spécifié dans la partie description de la Méta-Mallette 4.0 (cf. II.2), son lancement commercial est prévu pour la rentrée 2011. Serge de Laubier et l'équipe de développeur (G. Bertrand et G. Evrard) ont ainsi comme projet de fournir des instruments MéMa dans un pack lors de l'achat du logiciel et/ou

[5] Intitulé et descriptif fournit par Puce Muse au nombreux stagiaires dans le domaine d'ingénierie.

son kit de développement (SDK[6] – que j'ai utilisé pour développer mes patch[7]s Max). Cet aspect du stage prévoyait donc de porter les instruments déjà existants de la version 3.0 vers sa mise à jour.

a. MM.Copland

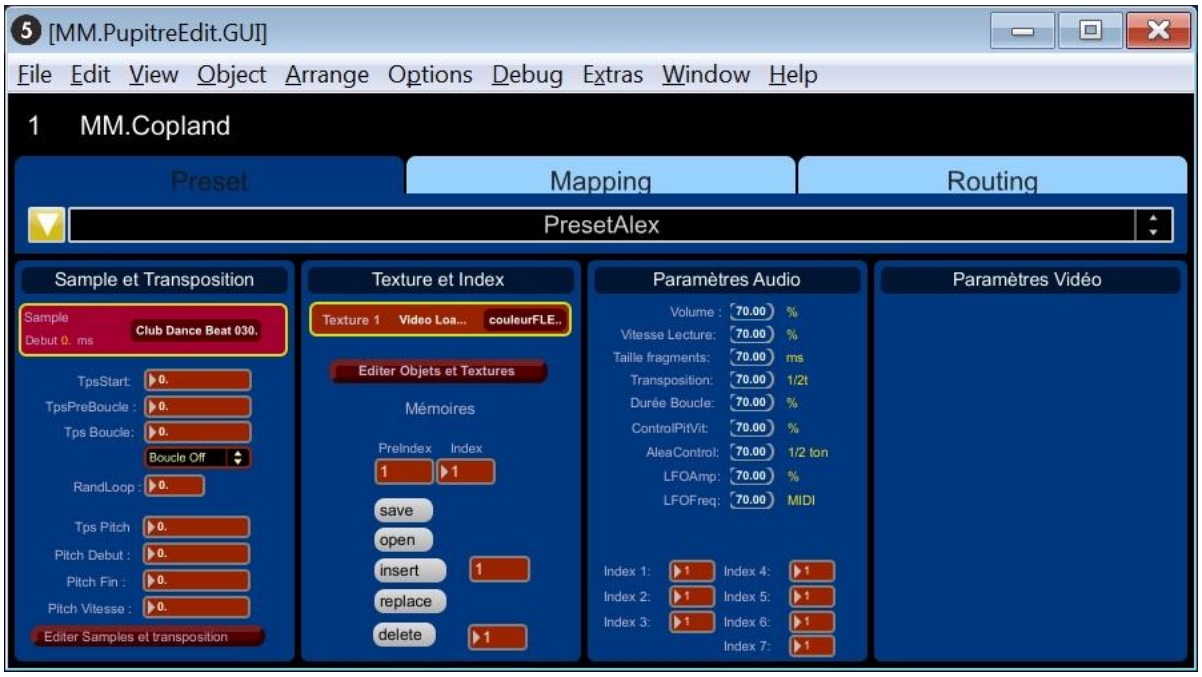


Fig 9. - Interface graphique des réglages de Presets de l'instrument Copland

Mon premier travail au sein de Puce Muse à donc été le portage (ou versionnage) de l'instrument MM.Copland de la version 3.0 vers la v.4.0.

Le rôle de cet instrument est principalement de jouer des œuvres musicales à l'aide de samples (initialement et principalement d'Aaron Copland, compositeur de musique contemporaine américain).

La plus grosse partie de l'algorithme consiste en effet à jouer sur des temps de boucles et de pré-boucles, les enregistrer et de revenir sur ces boucles temporelles à l'aide de navigation dans des index par l'intermédiaire d'une manette de jeu (dont chaque bouton aura une fonction assignée dans l'interface de Mapping ; par exemple charger un nouveau sample, ou lancer l'index 3 de la boucle temporelle).

[6] Software Development Kit – Kit de développement fournit par Puce Muse pour développer des instruments plus facilement et optimisés pour l'architecture Méta-Malette.

[7] Un ensemble de boîtes connectées, remplissant une ou plusieurs fonctions particulières et définies, qu'on appellerait classiquement un « programme » dans d'autres langages de programmation, se nomme un patch ou patcher chez les utilisateurs de Max. Etant donné la complexité que peut prendre un programme avec une architecture logicielle comme la Méta-Mallette, on en vient rapidement à imbriquer ces boîtes de fonctions entres elles. Ainsi, on nomme « sous-patch » (ou « subpatch ») un patch à l'intérieur d'un autre patch. Il se présente alors sous la forme d'une boîte dont le nom commence par « p » ou « patcher » et on peut lui ajouter un nom correspondant à sa fonction (comme on le ferait avec un nom de variable ou de fonction dans un langage de programmation classique). On nomme « abstraction » un patch utilisé comme un objet dans d'autres patches. L'abstraction se différencie du sous-patch en tant qu'elle existe indépendamment du patch dans lequel elle est utilisée. Ces abstractions sont utilisés notamment pour le SDK de la Meta-Mallette et permet de réutiliser des fonctions courantes n'existant pas dans Max.

Le contrôle de la vitesse de lecture et de la transposition est également implémenté à l'aide d'une synthèse granulaire, ce qui m'a permit d'aborder la future partie de recherche sur ces algorithmes de transformation audio.

En pratique, il m'a fallut comprendre en détail le parcours des données au travers des patchs existants afin de debugger, sélectionner et agir sur les principaux contrôles visibles dans l'interface.

J'ai également du supprimer ou modifier de nombreuses parties du programme rendues obsolètes par l'implémentation de la nouvelle architecture Méta-Mallette et l'existence de nouveaux objets Max (optimisés pour la synthèse granulaire par exemple). Cela m'a également permit de maîtriser la gestion et l'enregistrement des presets de réglages de l'instrument sur l'interface ainsi que dans les divers patchs de traitement, en utilisant le SDK mentionné plus haut.

b. MM.Bach

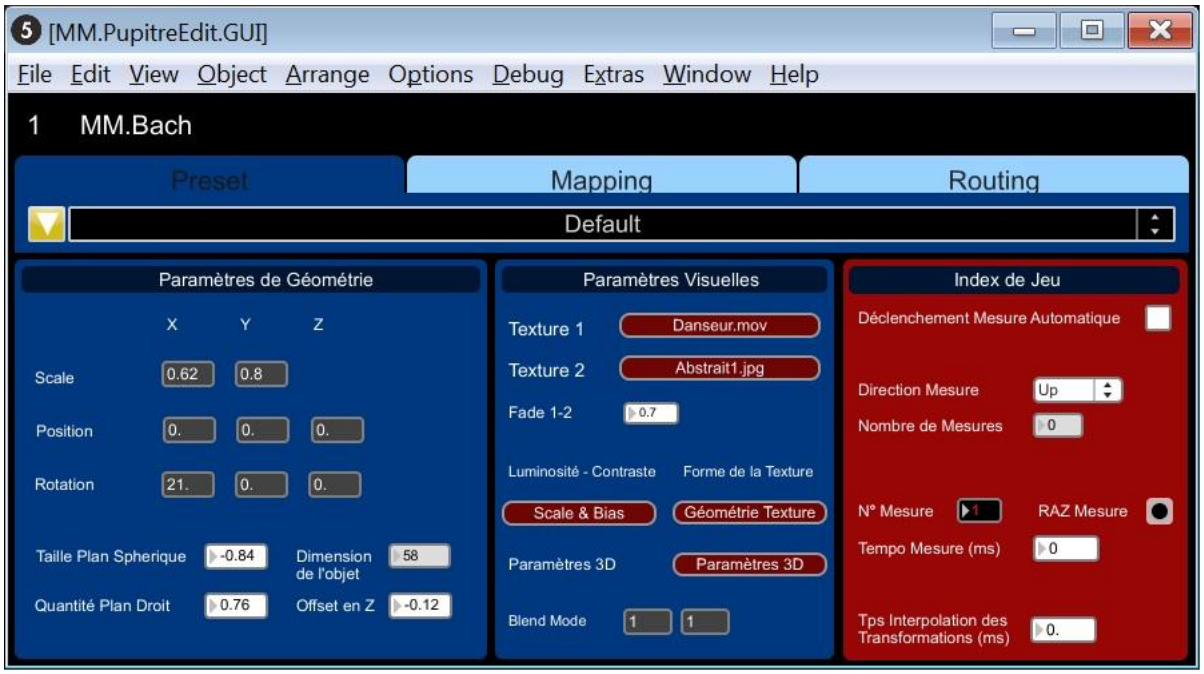


Fig 10. - Interface graphique des réglages de Presets de l'instrument MM.Bach

Le deuxième versionnage que j'ai effectué m'a permit d'aborder la partie Jitter et tous les aspects de création et modification d'une scène 3D OpenGl[8].

[8] **Open G**raphics **L**ibrary est une spécification qui définit une API (Application Programming Interface) multiplate-forme pour la conception d'applications générant des images 3D (mais également 2D). Elle utilise en interne les représentations de la géométrie projective pour éviter toute situation faisant intervenir des infinis. L'interface regroupe environ 250 fonctions différentes qui peuvent être utilisées pour afficher des scènes tridimensionnelles complexes à partir de simples primitives géométriques. *[Traduction de http://www.opengl.org/pipeline/article/]*

Il faut donc distinguer les 2 types d'utilisation de jitter qui sont de :

- Traiter la vidéo comme un éditeur vidéo classique permettant le coupage, collage ou des effets visuels comme le feedback, mais on travaillera ici directement sur des matrices contenants les différents plans de la vidéo (Rouge, Vert, Bleu et canal Alpha de transparence). Tous les effets QuickTime sont également accessibles par l'utilisation d'attributs ou messages envoyés vers les objets et fonctions générant la vidéo.

- Générer ou importer une scène en 3 ou 2 dimensions et la modifier en temps réel en la reliant par exemple avec des objets MSP, utilisés principalement pour le traitement sonore, en agissant sur les matrices stockant les scènes et plans 3D.

L'instrument Bach utilise principalement les fonctions 3D de jitter, mais ceux-ci se mélangent assez facilement avec la partie vidéo, puisque un outil ou une opération jitter plus spécifique à la vidéo peut tout à fait être utilisé pour traiter un objet jit.gl (il s'agit avant tout d'opération sur des matrices contenant des nombres) ce qui complexifie et fait de jitter un outil très puissant. Même si les rendus visibles dans l'instrument Bach (ou d'autres utilisant les jit.gl) peuvent parfois paraître faibles face à certains logiciels actuels (pour le jeu vidéo ou le cinéma), la grosse différence réside dans son optimisation pour le temps-réel, l'interaction et la liberté de création, puisqu'il s'agit bien d'un langage de programmation et non d'une interface de création graphique.

Les objets jit.gl (contraction de Jitter-OpenGL) sont nombreux et permettent également l'utilisation de fonctions optimisées pour la gestion de la lumière et ses composantes de couleurs ou bien de contraste et de luminosité. Mais ces fonctions permettent également d'agir sur les paramètres de géométrie d'un objet 3D (perspective, etc..). Nombreux de ces paramètres sont utilisés dans MM.Bach et les tutoriels Jitter m'ont permis de comprendre plus facilement les divers opérations effectuées dans le patch de traitement. Le but principal de cet instrument est de créer un objet plan contenant une texture (image ou vidéo) et prenant la forme d'une sphère ou d'un plan cartésien afin de jouer sur le fondu entre ces 2 géométries.

Les principaux paramètres que j'ai rendus réglables et assignables pour la géométrie sont :

- La taille, la position et le mouvement (en translation et en rotation) de l'objet;
- La taille et la position dynamique d'une forme, d'un plan (droit ou sphérique), sa définition (taille de la matrice définissant l'objet), l'extrusion de l'objet selon le plan Z (également relié au canal rouge de la vidéo, donnant un effet vivant à la forme créée) ;

La partie centrale de l'instrument (cf. Fig.10 Interface des Presets) autorise également le jeu avec divers caractéristiques de texture (image appliquée à l'objet 3D). Une des fonctions jitter permet par exemple d'effectuer un fondu entre deux textures, je l'ai donc rendu accessible sur l'interface, ce qui permet d'effectuer une transition entre une image fixe et une vidéo. Ces textures sont toutes disponibles et modifiables à l'aide d'une abstraction qui autorise l'ouverture de l'explorateur dans un dossier racine contenant divers fichiers reconnus par Max (.jpeg, .mov, .jxf [matrice jitter], .obj [objet 3D], ...). Un sous-patch d'interface permet le réglage plus fin de la géométrie de ces textures :

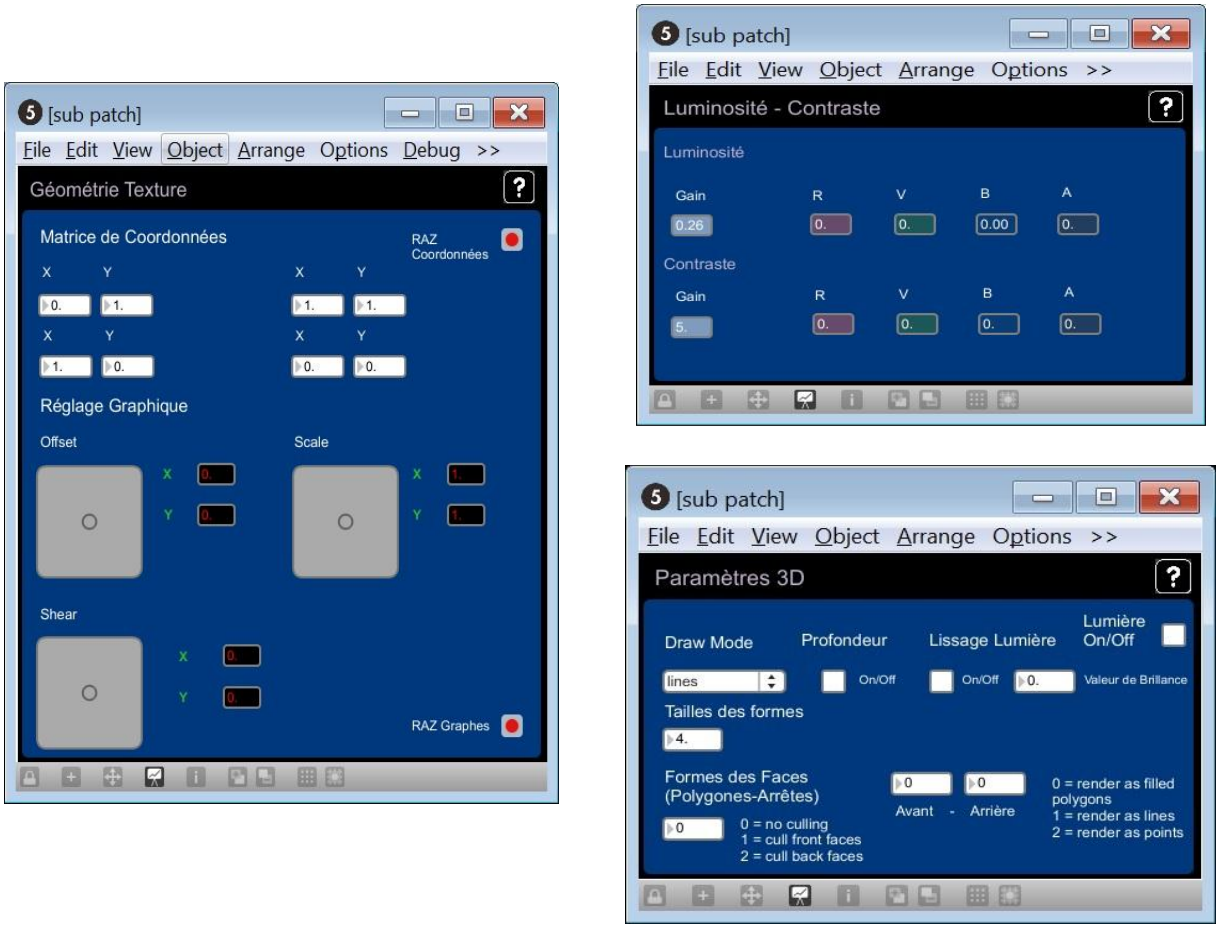


Fig 11. – Sous patches de gestion de Géométrie Texture, Luminosité-Contraste et Paramètres 3D

La Méta-Mallette permet également la gestion des paramètres de caméra, à savoir la direction de visée, la position de la caméra, l'ouverture focale et l'activation du feedback vidéo (effet de traînée vidéo similaire à une réverbération audio et rendu possible grâce à un instrument MM.FeedbackDessin déjà existant).

L'objet jit.gl.sketch qui crée le rendu 3D est certainement un des plus complet à utiliser dans jitter et il permet la gestion de tous ces paramètres en même temps. Combiné à la fonction jit.gl.shader, gérant tous les paramètres d'éclairage de la scène (réflexion spéculaire, diffuse, ambiante, etc...), il permet d'afficher un rendu très réaliste.

Les principaux paramètres de gestion des ces 2 objets (sketch et shader) sont donc disponible dans une sous-fenêtre de l'interface (Fig. 11.). Sans rentrer en détail sur les modes de fonctionnements et toutes les opérations possibles, le sous-patch « Paramètres 3D » permet de gérer la formes des faces des objets (taille et mode de dessin en lignes, points, triangles, arrêtes, faces pleines, etc...) ainsi que les paramètres de transparence et d'éclairage (profondeur, lissage, brillance, calque sur la source et la destination de chacun des pixels constituant l'objet,..).

Pour finir, la partie de droite (Fig. 10. - « Index de jeu », cadre rouge) reprenait l'idée initiale de l'instrument qui se basait sur un jeu définit par une *timeline* structurée pareillement à une composition

musicale. Ce mode d'indexation facilite donc le jeu en parallèle d'une portée musicale avec sa notion de Tempo, de nombres et numéros de Mesures, de Direction (avant, arrière) et de temps d'interpolation entre chaque forme (assimilable à une longueur de note, ronde, noire, croche, ...). Ce type de séquençage étant très courant dans la Méta-Mallette ainsi que dans la plupart des logiciels musicaux, il nous a parut intéressant de reprendre l'algorithme et de l'étendre à la gestion de tous les presets de l'interface des pupitres.

c. MM.GestionPreset

En prolongeant le déroulement de mon stage de manière chronologique, j'ai donc du étendre le concept de « Séquenceur de Presets » à un instrument complet. Il reprend exactement l'explication ci-dessus, mais étendu à chacun des réglages de préset pour chacun des instruments chargés et séquencés en temps-réel. La forme de l'interface récupère donc l'idée de la fenêtre principale Méta-Mallette, avec 30 lignes représentants le nombre de pupitres (et donc d'instruments) possibles et 8 mesures (nombre de colonnes) pour lesquelles on viendra charger le preset sélectionné dans un menu déroulant. Un indicateur, ou pointeur lumineux rouge nous indiquera à quel numéro de mesure on se situe (*timeline*). Le mapping permet d'assigner le déclenchement de la mesure automatiquement ou bien de venir charger un preset spécifique correspondant à un numéro de mesure. Le déplacement dans les mesures peut s'effectuer de gauche à droite ou de droite à gauche. Le Tempo de la mesure représente en faite le temps d'interpolation entre un réglage de preset et celui de la mesure suivante ; ceci permet des réglages plus fins, par exemple pour l'instrument MM.Bach qui affiche des formes complexes et nécessite des transitions douces entre un réglage en forme de boule texturé en vidéo vers une texture fixe affiché sur un plan morcelé en points.

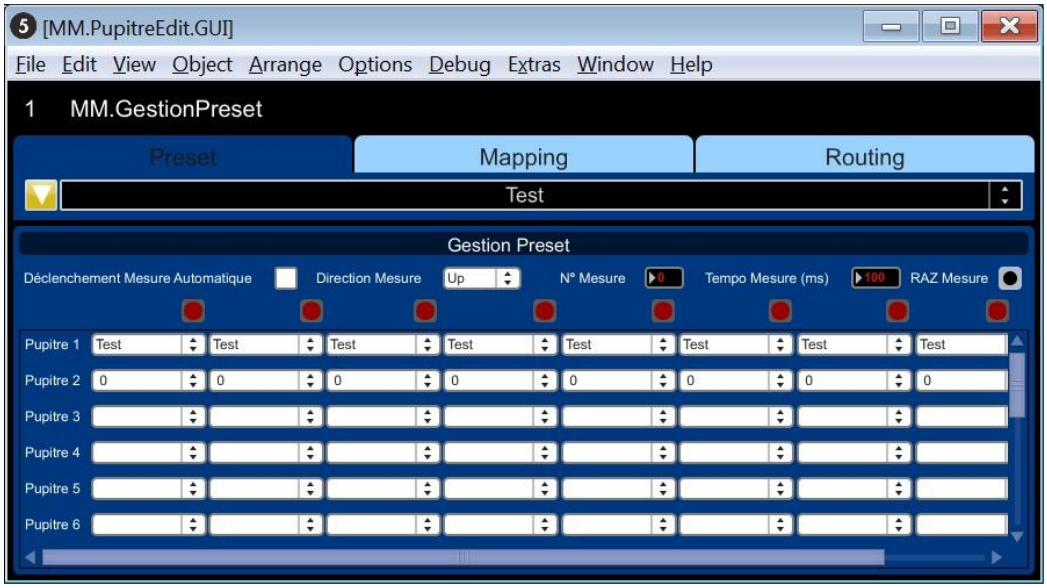


Fig 12. - Interface graphique des réglages de Presets de l'instrument MM.GestionPreset

Cet instrument a été assez rapide à mettre en place mais il est un outil très pratique et pourrait s'avérer très utile pendant les concerts PuceMuse, ou de nombreux paramètres sont à gérer en même temps et peuvent donc entrainer un retard sur le temps de mesure, ce qui ne peut pas arriver avec un tel instrument. Comme amélioration à apporter il manque justement le lien entre le tempo du séquenceur et le métronome inclus dans la Méta-Mallette.

3. Amélioration Audio pour le Time-Stretching et le Pitch-Shifting

C'est essentiellement par un échange avec de nombreux chercheurs ayant collaboré (ou été en relation) avec Serge de Laubier que j'ai ainsi pu synthétiser les divers techniques et état-de-l'art des méthodes actuels pour la transformation audio. C'est notamment Mr Emmanuel Favreau qui m'a fournit les réponses les plus pertinentes. Mr Favreau est ingénieur et responsable des développements des GRM Tools (plug-ins audio développés par le fameux Groupe de Recherche Musical qui se basent sur ces mêmes algorithmes d'analyse et de synthèse pour la transformation vocale et instrumentale).

a. Problème posé

La demande principale de Serge de Laubier était d'utiliser un algorithme de transformation utilisant le minimum de ressources CPU mais qui permettait tout de même d'améliorer les capacités actuelles des programmes utilisés dans le logiciel. En effet, l'utilisation « multi-joueur » de la Méta-Mallette engendre déjà de nombreux accès au DSP et au GPU pour le *render* d'affichage graphique. Il fallait donc trouver un programme qui préservait au maximum la qualité du son original, tout en permettant des transformations extrêmes : vitesse de lecture quasi nulle et transposition de ± 1 octave. Le compromis à trouver étant assez difficile, il a fallut un temps de recherche assez long (environ 3 semaines) pour réussir à analyser, synthétiser et tester les solutions existantes et accessibles avec Max.

b. Solutions existantes

En regardant en détail la majorité des instruments effectuant une transposition ou un time-stretch, j'ai pu constater qu'il s'agissait d'une lecture par synthèse granulaire ou bien de l'utilisation de l'objet *gizmo~* et *freqshift~.* Ses objets fournis dans la bibliothèque MSP sont déjà très robustes et optimisés, mais entraînent une dégradation du signal (distorsion harmonique et effet de dispersion de phase prononcé) empêchant l'intelligibilité et la reconnaissance du son original.

De ce fait, j'ai pu constater qu'il n'existe pas de solution universelle car le type de signaux à traiter conditionne beaucoup le choix de l'algorithme :

Signal monodique harmonique : techniques du type PSOLA
Signal complexe, polyphonique, etc... : techniques fréquentielles, phase-vocoder.

Il convient également de préciser que le terme « Frequency Shifting » est différent de celui de « Pitch Shifting », puisque le premier effectue seulement un déplacement des fréquences équivalent pour toute la largeur du spectre, tandis que le second (dont nous recherchons les qualités) dilatera le spectre de manière proportionnel en gardant les relations harmoniques du son original. En général le Pitch Shifting est obtenu en utilisant une des techniques de Time Stretching (étirement temporel) décrites ci-dessous et en effectuant un changement de fréquence d'échantillonnage à la reconstruction du signal.

Même si mon choix final s'est porté sur une utilisation intelligente du Phase Vocoder et de l'Interpolation Spectrale, je présenterai subséquemment les diverses méthodes testées et leur principe de traitement du signal.

c. Modèle Sinusoïdal

Le principe de cette technique repose, comme nous le verrons plus tard pour le phase-vocoder, sur une reconstruction de l'amplitude et de la phase du signal [2] et [3]. Mais contrairement au Vocodeur de phase, la technique PSOLA repose sur une analyse dans le domaine temporelle.

Il s'agit en effet de trouver la période (ou de manière équivalente la fondamentale) et les partiels (ou harmoniques) d'une section donnée de l'enregistrement en effectuant une auto-corrélation du signal original. Le déphasage entre les différents harmoniques sera représenté comme une différence relative entre la fondamentale du signal et les harmoniques suivants. La reconstruction et les transformations (pitch et time-stretch) sont ensuite rendues possibles par une interpolation linéaire entre les différentes fenêtres de corrélations au cours du temps. Ainsi, ce sont bien des modèles de synthèse additive qui permettent la reconstruction du signal, tels des oscillateurs à fréquences uniques (mais évoluant au cours du temps) et placés en parallèles, donnant en sortie une reconstruction simplifiée du signal.

Le problème engendré est bien entendu lié à la complexité du signal de base, qui ne permet pas d'effectuer une corrélation « propre » lorsque le signal possède de nombreux harmoniques (musique polyphonique,…).

Une des solutions existantes actuellement est celle du Pitch Synchronous Overlap Add Method ou PSOLA [5]. Cette technique est notamment développée par l'IRCAM au sein de l'extension (nommé *external*) FTM.Gabor pour le logiciel Max/Msp, ce qui aurait pu faciliter une adaptation pour des instruments Méta-Mallette.

Au fur et à mesure de mes recherches et de mes tests j'ai donc pu me rendre compte que ce modèle était beaucoup plus limité que le phase-vocoder même s'il demandait moins de ressources processeur, ce qui était bien entendu un avantage indéniable pour une application temps-réel. L'intérêt de cette technique est donc plus porté sur des signaux monophoniques comme des voix ou des instruments solistes.

d. Phase Vocoder[9]

Au bout d'une longue période de recherche, c'est finalement sur le site de Cycling74 (société qui développe le logiciel Max-Msp-Jitter) que j'ai découvert un patch Max avec des explications claires et d'une qualité bien supérieure aux autres algorithmes trouvés auparavant. Ce patch à été écrit par Mr Richard Dudas et Mr Cort Lippe et inspiré par Miller Puckette [6].

[9] Il existe souvent une confusion entre un Vocoder classique et le Vocoder de phase. Le Vocoder classique utilise deux signaux d'entrées (un signal porteur et un modulant) pour produire un seul son en sortie. Il permet par exemple de parler dans un microphone et transformer sa voix avec un piano synthétiseur, donnant un effet de voix métallique ou de robot. Le signal modulant est envoyé vers une banque de filtre dont les amplitudes serviront à moduler les bandes du signal porteur. Les deux techniques se rapprochent par le seul fait qu'elles utilisent des banques de filtres (la STFT peut-être vue comme une banque de filtre constitué de largeur de bande constante les unes à la suite des autres et/ou se chevauchant [overlap]). Dans le cas du Vocoder de phase, la résolution spectrale nécessite un minimum de 512 bandes tandis que le Vocoder classique n'en utilise qu'une dizaine. Le terme de « Voice Coder » se réfère à leurs applications originales de deux procédés de codage de voix dans des applications militaires.

[traduction de l'article de Stephane Bernsee du site web dspdimension.com]

Le principe de traitement du phase-vocoder repose sur le calcul de la Transformée de Fourier à Court Terme (*Short Time Fourier Transform* en anglais). Il s'agit en fait d'une FFT discrète classique mais qui découpe le signal d'entrée en plusieurs morceaux, ce qui permet d'effectuer le traitement en temps-réel et de manière très rapide. Une valeur d'overlap est également requise pour le recouvrement des fenêtres et assurer une meilleure analyse, chaque FFT sera ensuite multiplié par une fenêtre de Hanning pour éviter les discontinuités. Un overlap (équivalent au hop size), égal à 2 correspond à un recouvrement de FFT sur la moitié des échantillons de toute la fenêtre de FFT (*cf. fig. 13 – Principe de l'analyse STFT).* Après quelques tests (écoute subjective mais avec plusieurs personnes et analyse pourcentage d'accès CPU), un overlap de 4 semble être optimum. Une taille de 1024 bandes discrètes de fréquence pour une fréquence d'échantillonnage de 44.1 Khz à également été choisit et correspondait parfaitement aux recommandations faites par Richard Dudas.

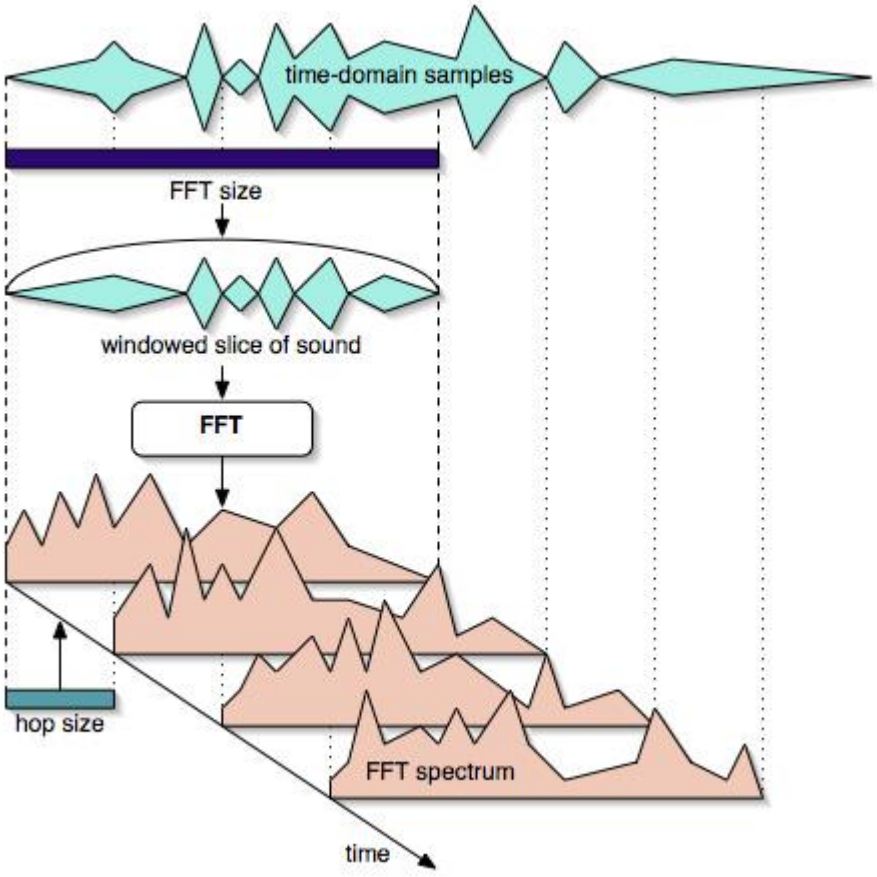


Fig. 13 – Principe de l'analyse STFT

Ainsi, pour chaque bande de fréquence, nous obtenons les informations d'amplitude et de phase, mais l'effet de fenêtrage entraîne évidemment des pertes d'informations spectrales. En effet, du fait de la résolution temporelle et/ou fréquentielle choisie pour le calcul de la fft, certaines composantes sinusoïdales constituant le signal peuvent se retrouver dans une bande adjacente. C'est donc grâce au recouvrement des fenêtres (overlap) successives et du calcul de la différence entre deux valeurs de phases que l'on réussit à reconstruire le signal avec une cohérence temporelle (horizontale) et fréquentielle (verticale) qui soit la plus proche du signal original.

Le but du phase-vocoder étant de faire du time-stretch et de la transposition, ces techniques seront rendus possibles grâce au contrôle de la vitesse de lecture de l'échantillon sonore original.

Pour le time-stretch, il s'agit simplement de lire une fenêtre temporelle plus longue ou plus courte pour une même vitesse de lecture de l'échantillon. Au niveau du patch Max, il s'agit de donner la valeur d'index à laquelle lire le son (stocké dans un *buffer~*) et sur une largeur proportionnelle à la vitesse de lecture désirée (comme on vient de le voir, plus la FFT sera calculé sur une fenêtre temporelle large, plus on mettra de temps à lire le son dans as totalité).

Pour la transposition on effectue le même traitement, mais en venant interférer en plus sur la vitesse de lecture. Ainsi, si le facteur de transposition est plus grand que 1, nous lirons une fenêtre plus large dans le *buffer~* et a une plus grande vitesse ; à l'inverse si le facteur de transposition est plus petit que 1, nous prendrons des tailles de fenêtres temporelles plus courtes et les lirons à une vitesse plus lente.

Ayant discuté de ces techniques avec des personnes expertes du GRM ou du LIMSI, j'ai pu constater que problèmes liés au phase-vocoder sont bien connus. En résumé, pour avoir un bon calcul des fréquences il faut un grand recouvrement des fenêtres (augmentation des ressources CPU) et s'assurer de n'avoir qu'un partiel par bande (augmentation de la taille de FFT et perte de résolution temporelle). Ces problèmes de résolution spectrale et temporelle, inhérents au calcul de FFT et que Jean-François Charles (cf. partie IV.3.e sur les techniques fréquentielles) compare au principe d'incertitude de Heisenberg[10], m'ont permis d'aborder les limites des techniques actuelles en traitement du signal.

e. Autres solutions et techniques d'analyse

La première solution envisagée et discutée avec Serge de Laubier était l'utilisation des patches proposés par Jean-François Charles [7], qui reprenaient également le principe du phase-vocoder mais en utilisant la particularité de Jitter et de travailler sur des matrices stockant les différents sonogrammes calculés (en temps-réel ou non). Sa proposition était également d'analyser les transitoires d'attaques du son afin de les relire en vitesse normale (sur des temps très court et donc avec une résolution temporelle très fine) et de varier cette vitesse lors du passage au temps de suspend et de relâchement. Le problème de ces programmes résidait justement au niveau de l'analyse et du stockage des données et qui entrainaient de nombreuses distorsions du signal et des effets de dispersions (*smearing effect*).

Nous verrons dans la partie création d'instrument (§ IV.3) que j'ai tout de même utilisé un patch particulier de Mr Charles et qui permettait de gérer le Freeze en effectuant une interpolation linéaire entre deux fenêtres de FFT consécutives. L'effet sonore rendu était au moins aussi intéressant que pour le programme de Mr Dudas utilisée pour une vitesse de lecture quasi-nulle (équivalente au Freeze).

Les *externals* du CNMAT (Université de Berkeley, Calofornie) sont d'autres solutions que j'ai découvertes en parcourant le net, qui regorgent d'informations pas toujours pertinentes, notamment sur les forums où il est parfois difficile de distinguer les experts des gens qui « pensent savoir ». Ces objets fournis avec des aides (ou

[10] De manière simplifiée, ce principe d'indétermination énonce que pour une particule massive donnée, on ne peut pas connaître simultanément sa position et sa vitesse. Soit on peut connaître précisément sa position (par ex: à ± 1 mm) contre une grande incertitude sur la valeur de sa vitesse (par ex: à ± 100 m/s), soit on peut connaître précisément sa vitesse (par ex: à ± 0,0001 m/s) contre une grande incertitude sur la valeur de sa position (par ex: à ± 1 km).

tutoriaux) utilisent aussi la représentation par modèle sinusoïdal, mais reprennent en plus l'idée de filtres résonnants appliquée lors de la synthèse du signal et qui donne une richesse et une profondeur au son.

Cependant, malgré la qualité et le peu de ressources demandées pour le temps de calcul, le gros problème de ces objets résidaient dans le fait qu'ils utilisaient des modèles fixes de sons tels des fonctions de transfert d'instruments (comme des sons de cloches ou d'instruments solistes). Ces modèles nécessitaient une représentation spécifique des données, compilées uniquement par certains logiciels traitant les modèles sinusoïdaux. J'ai donc utilisé le logiciel gratuit SPEAR (pour Sinusoidal Partial Editing Analysis and Resynthesis) qui calcul des FFT (en précisant la résolution spectrale ou temporelle désirée) et permet de créer des modèles sinusoïdaux afin de tester les limites du dispositif.

J'ai donc vite fait la constatation que pour des modèles complexes et évoluant rapidement dans le temps, le modèle du filtre résonnant entrainait des artefacts sonores similaires à une synthèse FM. Ceci était certainement du au fait que lors de la resynthèse, les différents harmoniques étaient reconstruit avec une incertitude au niveau des différentes composantes du modèle.

Il convient toutefois de préciser que les logiciels de traitement audio de haute-qualité combinent différentes techniques d'analyse et de synthèse pour différentes parties du signal, en séparant par exemple les transitoires (ou attaque du son) et les signaux périodiques représentant le temps de suspend de la forme d'onde sonore. Ainsi, en poussant mes recherches sur les différents types d'analyse existants (en plus des classiques FFT, auto-corrélation et Densité Spectrale de Puissance), j'ai découvert que de nombreux travaux étaient menés sur l'analyse par transformée en ondelette ou bien par modèle adaptatif de Transformée de Fourier afin de s'affranchir au maximum du problème de la résolution spectrale et/ou temporelle.

Au niveau de la resynthèse, je n'expliquerais pas non plus en détail le principe de la synthèse granulaire, sur lequel j'avais effectué un projet au cours de mon année universitaire et qui était utilisé en majorité dans la plupart des instruments Méta-Mallette traitant des samples audio et effectuant des transformations sonores. Ce type de synthèse, en plus d'être utile pour le time-stretch ou la transposition, permet de créer des sons inattendus en prenant des *'micro-parties'* d'un son et en '*collant les différents grains'* composant le sample les uns à la suite des autres.

Cependant, le temps impartit pour le stage et la complexité de ce type d'analyse (n'ayant pas d'application actuel au sein du logiciel Max/MSP) m'ont amené à laisser mes recherches en suspend et commencer la création d'un instrument MéMa.

4. Création d'un Instrument Méta-Mallette : MM.Freezer

A la suite de mes recherches, j'ai du commencer à utiliser ces algorithmes existants pour les utiliser dans un instrument Méta-Mallette. Comme je l'ai dit en introduction, mon idée finale était d'utiliser de façon combinée les meilleurs résultats obtenus pour le Freeze avec l'algorithme de JF Charles et d'utiliser le Phase Vocoder de Richard Dudas pour le jeu normal, ralenti et accéléré.

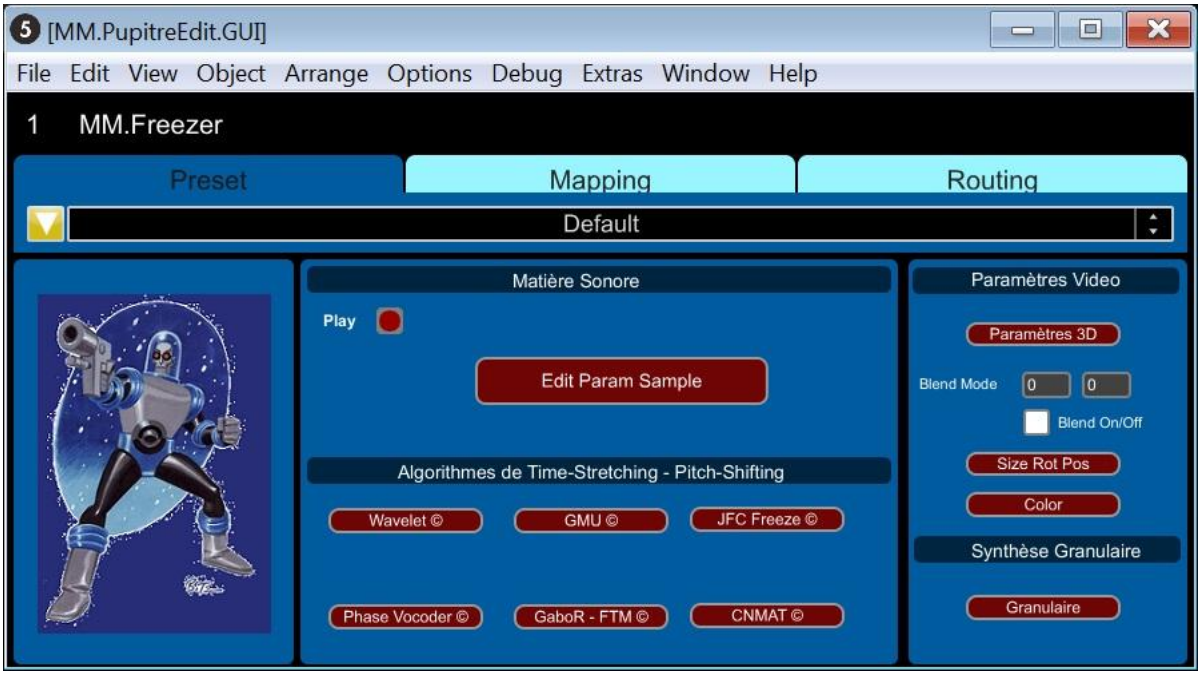


Fig 14 – Interface des Presets de l'instrument MM.Freezer

Cet instrument est donc une synthèse de mon travail abordé à Puce Muse, puisqu'il s'agit d'un assemblage des nombreux patches et algorithmes utilisés au cours de mon stage. La gestion des périphériques de jeu (mapping) reprend de manière classique la *'philosophie'* et les méthodes utilisées dans la plupart des instruments Puce Muse et l'algorithme mis en place pour la lecture d'échantillon audio est très ressemblant à celui décrit pour l'instrument MM.Copland. Il s'agit en effet de jouer sur des indices de temps de pré-boucle et de boucle que l'on aura précédemment enregistrés.

Le temps de boucle sera contraint par une croissance et/ou décroissance de vitesse de jeu et de pitch (hauteur de la note), ce qui permet notamment de rester *freezé* (ou bloqué) sur une note ou phrase musicale particulière que l'utilisateur trouvera à son goût.

Cependant, bien que l'idée de base était assez similaire à MM.Copland, le patch *phase_vocoder.maxpat (cf Annexe partie A)* n'utilise pas des valeurs numériques, mais principalement des émulations de signaux audio et leurs FFT correspondantes comme on l'a vu partie IV.3.b. Cette distinction de traitement des informations à donc entrainé de nombreuses difficultés pour jouer sur la gestion des index de boucle.

En effet, la vitesse de changement et la conversion d'état d'un signal vers une donnée numérique est assez difficile à implémenter de façon stable dans Max et cela a amené des complications et des bugs lors du saut vers les différentes temps de boucle préalablement stockés dans les listes de sauvegarde.

Il faut de plus ajouter que la gestion et le traitement de ces données doit rester invisible pour l'utilisateur. Il a donc fallu rendre agréable et intuitif la façon de stocker ces valeurs de boucles. Heureusement, Max possède de nombreux objets qui facilitent la visualisation d'informations, notamment l'objet *waveform~* qui permet directement l'affichage d'un buffer contenant un fichier audio et sa représentation temporelle. Je l'ai donc rendu accessible dans un sous-patch d'édition des samples (cf Fig. ???). On peut ainsi zoomer, éditer, enregistrer et lire sa sélection à volonté comme dans un logiciel classique de séquençage musical (à la différence qu'il faut déclencher le son à l'aide de la gâchette du joystick pour écouter sa création).

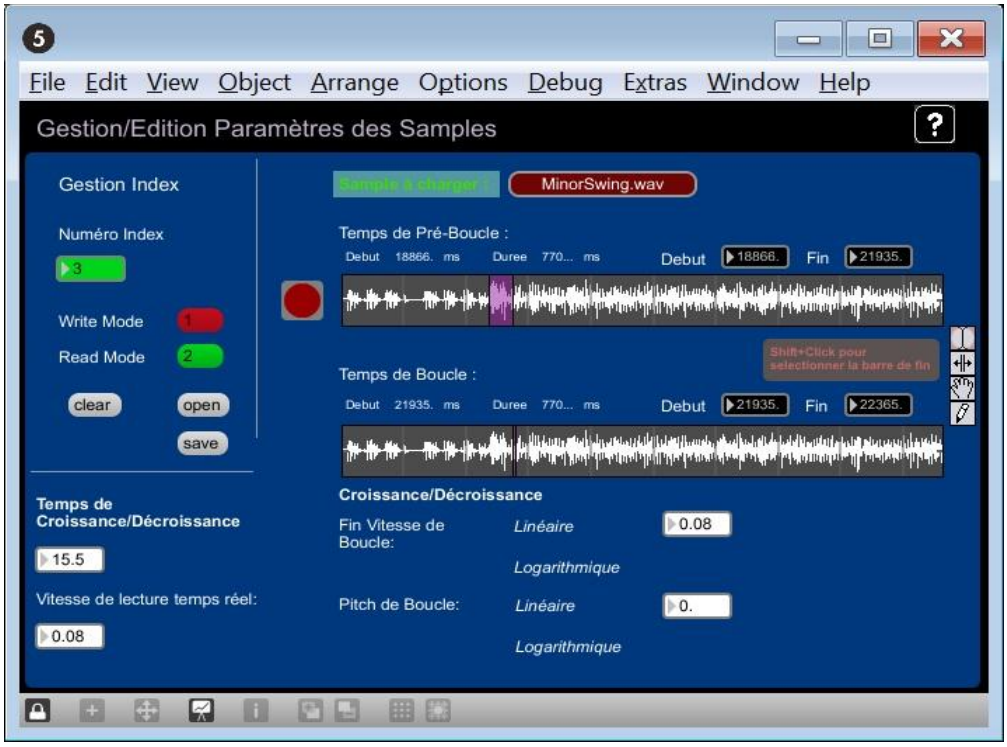


Fig. 15 – Sous-patch de Gestion/Edition des Samples

L'interface de contrôle du phase-vocoder et du freeze est assez simple puisqu'il affiche principalement la vitesse de jeu et la hauteur de la note pour la transposition désirée. Pour celui de JF-Charles, on peut également gérer le lissage d'interpolation entre deux fenêtres de FFT (cross-fade entre deux matrices contenant les valeurs d'amplitude et de phase) ainsi que le niveau de bruit ajouté.

La partie graphique reprend quant à elle les programmes d'un instrument existant : MM.SampleBayleGraph, développé par Puce Muse pour le compositeur de musique acousmatique François Bayle. L'idée est de représenter en 3 dimensions la FFT (type *waterfall*, amplitude seulement) d'un sample, selon les dimensions et la durée choisie.

Ma contribution à ce programme à été d'envoyer les bonnes valeurs d'index et de pointeur de lecture sur la figure générée en OpenGL afin de visualiser en direct les possibilités de déclenchement de saut d'index ou de variation de vitesse. J'ai également reporté les patches d'interfaces pour la taille, la position et la rotation de l'objet 3D ainsi que les gestions de la lumière et de la couleur de l'objet que j'avais créé pour l'instrument MM.Bach.

L'instrument final contient cependant quelques bugs puisque l'affichage 3D ne fonctionne pas sur les samples dépassant une certaine taille ou longueur (au-delà de 30 secondes). Je n'ai pas pu résoudre ce problème à temps car il s'agissait de modifier le programme Jitter complexe jouant sur les matrices et il aurait fallu modifier l'amont du programme pour découper les fichiers audio qui posent problèmes.

Une des requêtes initiale était également la possibilité de jouer sur des fichiers stéréo, mais ayant constaté une forte demande CPU pour des fichiers mono, j'ai décidé de ne pas le mettre en place immédiatement et proposer d'attendre le débogage et l'amélioration complète de l'instrument. Il conviendra en effet de supprimer de nombreuses parties du patch de calibrage et de routage du Joystick puisque ces programmes provenaient d'une ancienne version de la Méta-Mallette et ne trouve donc plus de raison d'être au sein de la nouvelle architecture logicielle.

Conclusion

J'ai eu la chance de trouver un stage dans un champ d'application qui m'intéresse particulièrement : l'acoustique musicale. La double approche scientifique et musicale encouragée et largement mise en œuvre par toute l'équipe de PUCE MUSE m'a beaucoup apporté, et j'ai pu constater que ces deux domaines, loin d'être opposés et hermétiques l'un à l'autre, s'alimentent mutuellement : le scientifique et l'artiste s'enrichissent de leurs échanges. Ayant comme projet à long terme de travailler dans le domaine du multimédia, cette expérience m'a avant tout permis d'acquérir de solides bases - tant théoriques que pratiques - dans ce domaine.

J'ai en premier lieu pu apprendre à utiliser le logiciel Max/MSP/Jitter très largement utilisé dans le traitement audio et graphique pour la création. C'est un logiciel dont les raisonnements algorithmiques sont assez originaux, où l'utilisation de l'expression « programmation orienté objet » prend tout son sens puisqu'il s'agit de connecter des boîtes de fonctions entres elles afin de créer un programme complexe et optimisé pour le temps-réel. Une immersion suffisamment longue avec cet outil est donc nécessaire pour en maîtriser l'utilisation et les subtilités. Ce stage a donc été, en premier lieu, une excellente formation en informatique musicale et m'a également permis de comprendre en détail la conception et l'architecture sous-jacente d'un logiciel de Musique Assistée par Ordinateur. L'étude des traitements audio pour le time-stretching et le pitch-shifting réalisée au cours de la programmation de l'instrument MM.Freezer a également été très enrichissante. J'ai pu y appliquer des notions enseignées en Master SDI, notamment de Traitement du Signal. Toute la période de recherche m'a permis de partir sur de bonnes bases et de découvrir des notions qui sont à la pointe de la recherche actuelle en traitement du signal pour la musique.

D'autre part, l'environnement professionnel dans lequel j'ai évolué m'a beaucoup apporté sur la gestion d'un projet destiné à être pris en main et poursuivi par une autre personne. Posant les toutes premières bases de cet instrument, j'ai dû apprendre à envisager de nombreuses solutions logicielles différentes pour une même fonction, à les justifier, les modifier ou les éliminer pour en discuter par la suite avec les programmeurs ou mon maître de stage. Sans oublier la partie versionnage et portage d'instrument qui a été très formatrice et m'a permis de faire avancer les divers chantiers et besoins requis pour l'avancement et la commercialisation de la Méta-Malette.

Je suis très satisfait de l'ensemble des aspects de ce premier stage. Faute de temps je n'ai pu résoudre tous les problèmes posés ni approfondir suffisamment certains points restés en suspens (réglages de presets, de mapping et débug des instruments), mais les nombreux stagiaires et développeurs collaborant avec Puce Muse auront la possibilité de développer mon travail qui a été documenté et rendu compréhensible avec les nombreux commentaires présents dans les patches et leurs interfaces rendues les plus intuitives possibles.

Enfin ce stage dans une structure de type association loi de 1901, développant des produits artistiques et techniques, m'a dévoilé de nouvelles voies pour réaliser des projets musicaux personnels et d'autres professionnels, alliant originalement la pratique instrumentale et des outils informatiques évolués. Ayant une pratique régulière et une formation dans le mixage audio et le multimédia, la découverte de nombreux travaux d'artistes et chercheurs au cours de discussions ou divers recherches m'ont ouvert de nouveaux horizons créatifs, notamment sur tous les aspects de transformations et détournements technologiques.

Références

Bibliographie

[1] DE LAUBIER Serge, GOUDARD Vincent, *PUCE MUSE - La MétaMallette*, 2007.

[2] Di Federico Riccardo, *Waveform Preserving Time Stretching and Pitch Shifting for Sinusoidal Models of Sound*, Università degli Studi di Padova.

[3] Wright M., Dudas R., Khoury S., Wang R., Zicarelly D., *Supporting the Sound Description Interchange Format in the Max/MSP Environment*, CNMAT, UC Berkeley.

[4] RIOUX Vincent, POLETTI Manuel, *An experimental SDIF-sampler in Max/MSP,* International Computer Music Conference, September 2002.

[5] MARCHAND Sylvain, RASPAUD Martin, *Enhanced Time-Stretching using order-2 sinusoidal modeling*, SCRIME – LaBRI,Université Bordeaux, DAFX, October 2004.

[6] DUDAS Richard, LIPPE Cort, *The Phase Vocoder Parts I & II*, On-line article published on www.cycling74.com, October, 2006.

[7] CHARLES Jean-François, *A tutorial on spectral sound processing using Max/MSP and Jitter*, Computer Music Journal, 2008.

Webographie

Objet externes Max :
http://imtr.ircam.fr/imtr/Max/MSP_externals

Introduction aux objets FTM et Gabor :
http://ftm.ircam.fr/index.php/A_brief_introduction_to_FTM

Site internet et forum de Cycling74 :

- alternative to elastic~

http://cycling74.com/forums/topic.php?id=28746

- The "phase" part of the phase vocoder
http://cycling74.com/forums/topic.php?id=28215

- The phase Vocoder by Richard Dudas & Cort Lippe – part I & II

http://cycling74.com/2006/11/02/the-phase-vocoder-%E2%80%93-part-i/

- FFT~ Wavelet~

http://www.cycling74.com/forums/topic.php?id=23021

- extremely precise sonogram

http://cycling74.com/forums/topic.php?id=25439

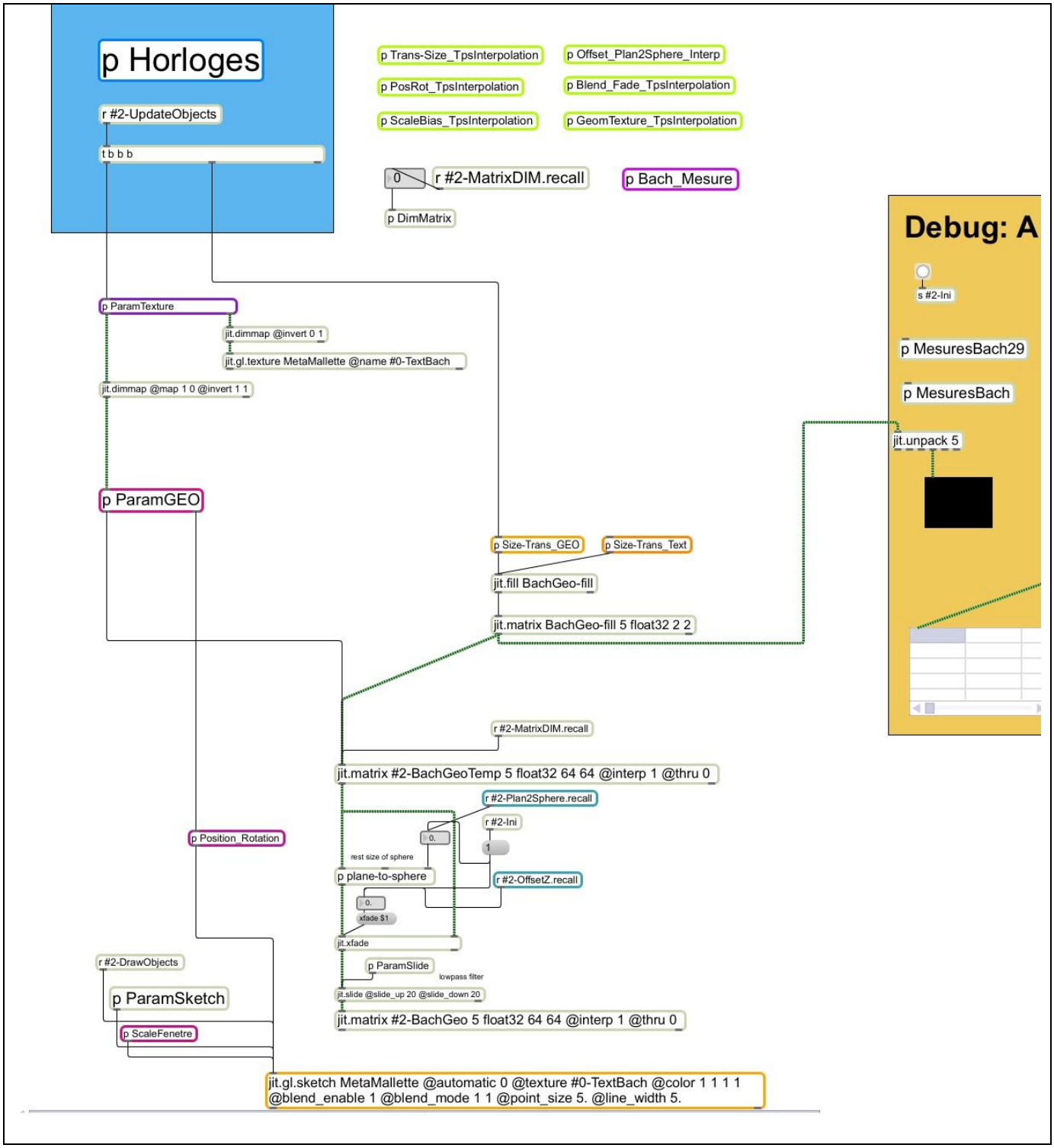
- Time Stretch & Pitch Shift

http://www.dspdimension.com/admin/time-pitch-overview/
http://www.dspdimension.com/admin/pitch-shifting-using-the-ft/

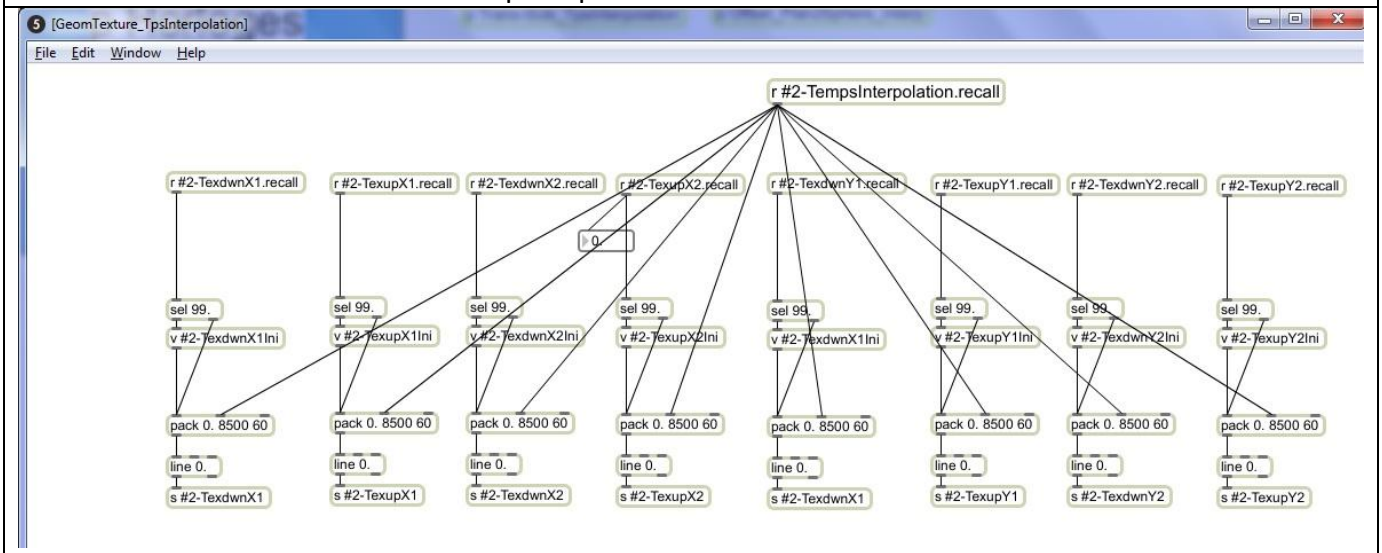
Table des figures

Annexes

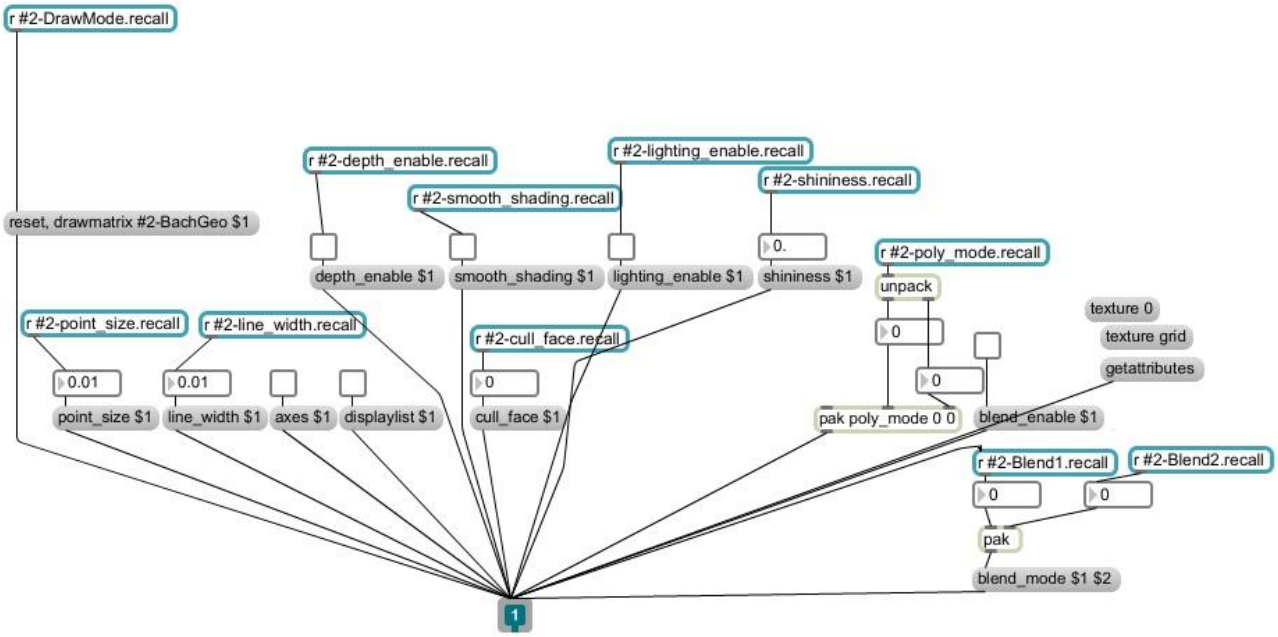
A. Patches MAX



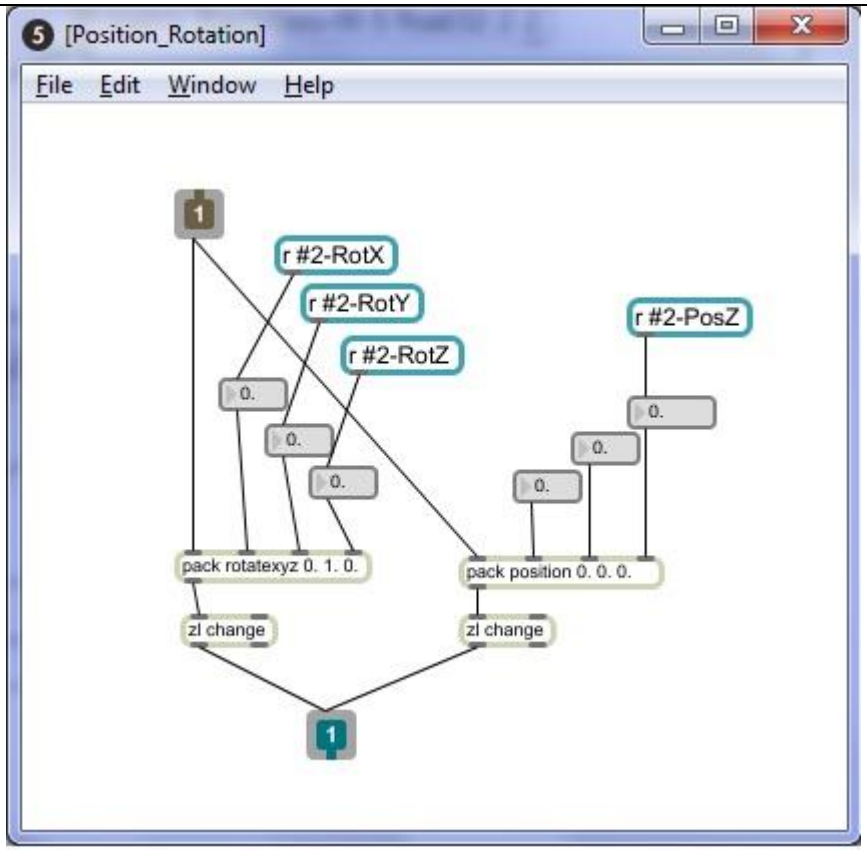
Patch principal de l'instrument MM.Bach



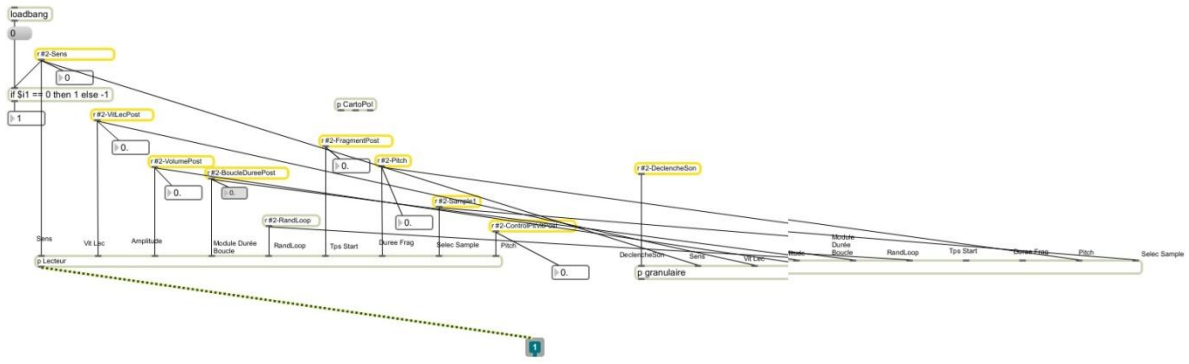
Sous-patch de géométrie de texture – MM.Bach



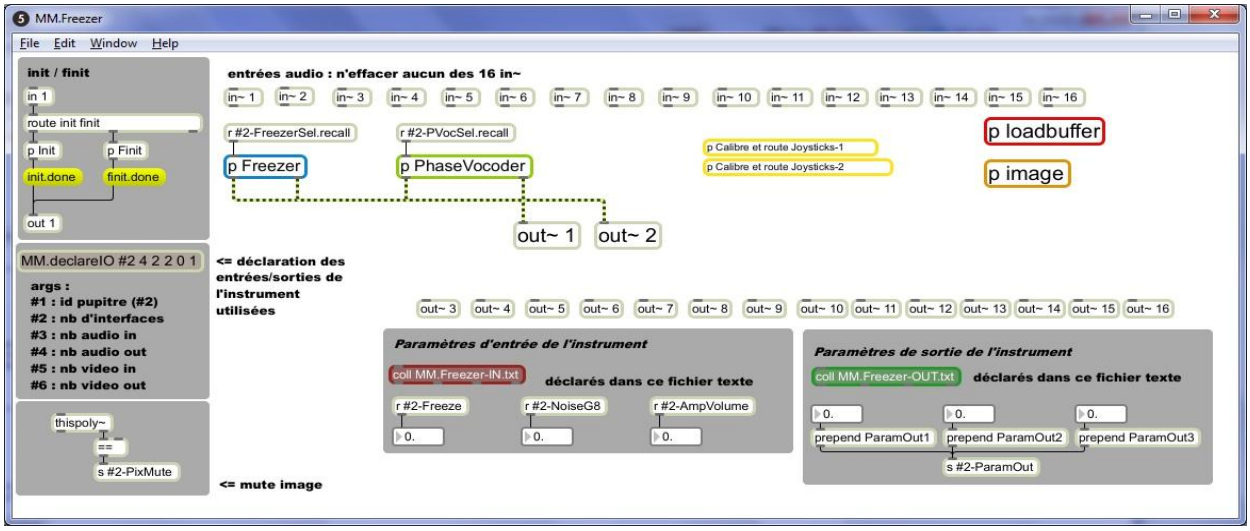
Sous-patch de gestion de l’objet Sketch – MM.Bach



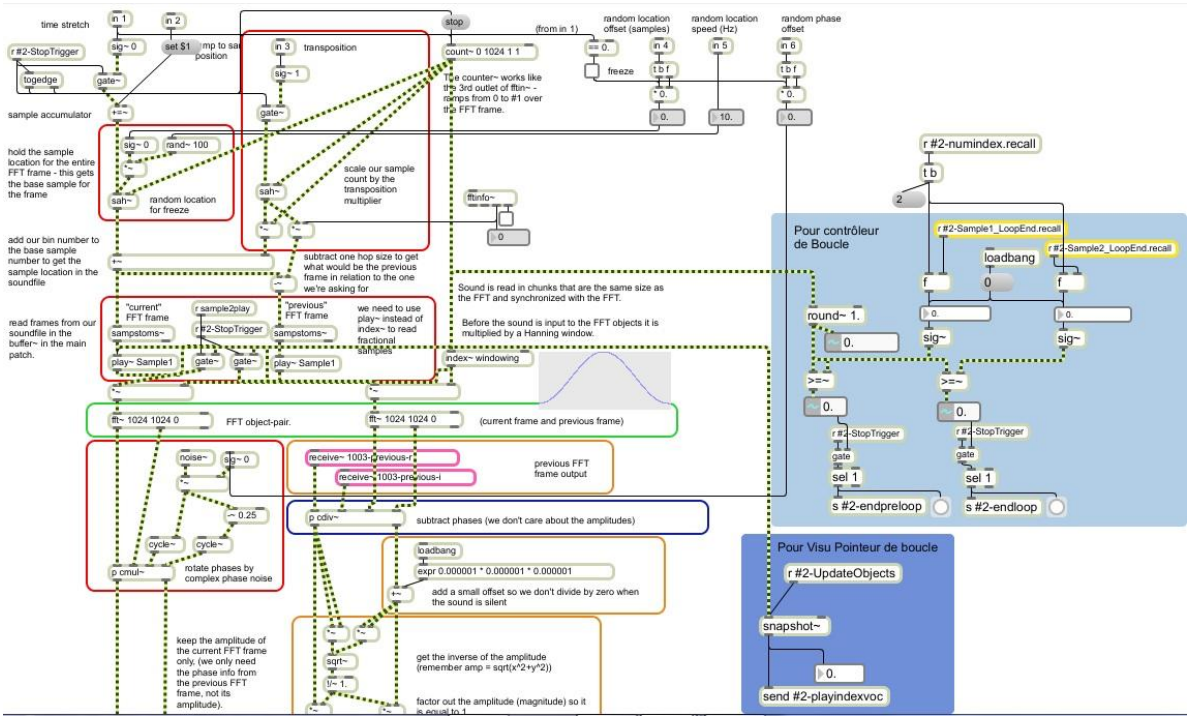
Sous-patch de gestion de la rotation et de la position – MM.Bach



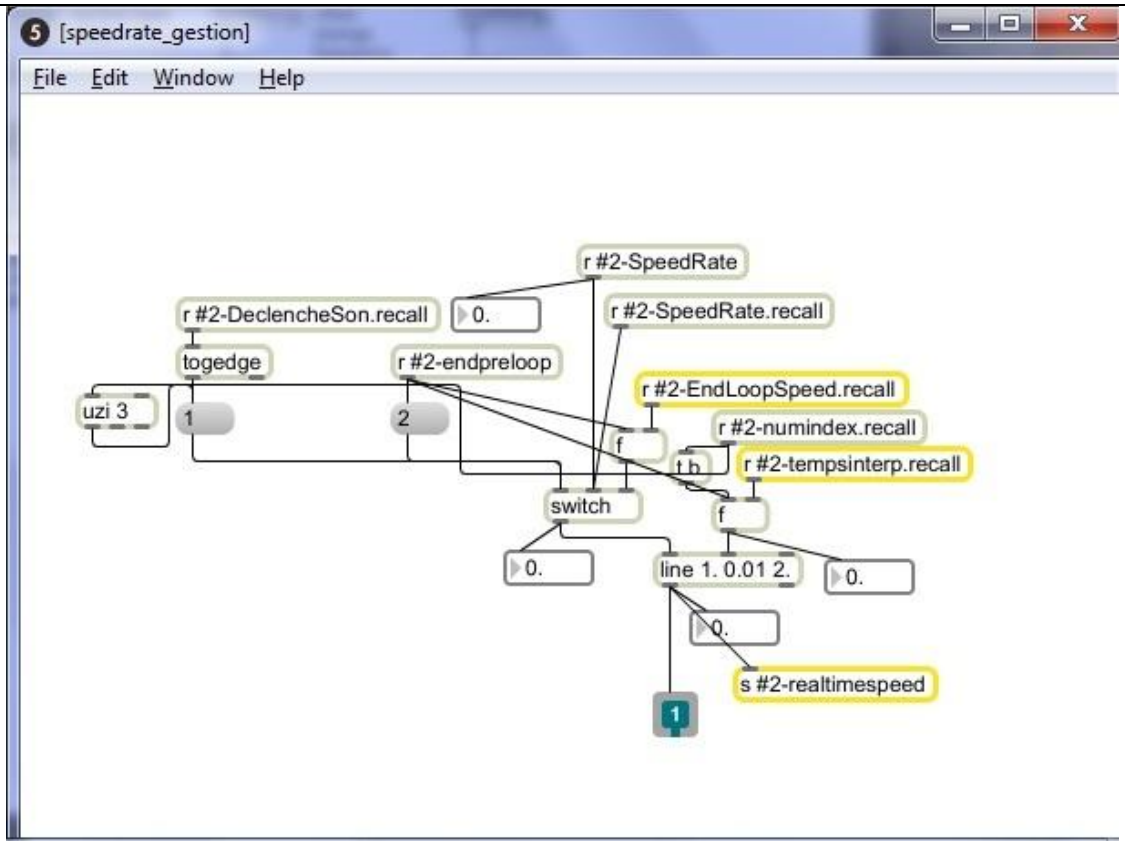
Sous-patch de gestion de la lecture – MM.Copland



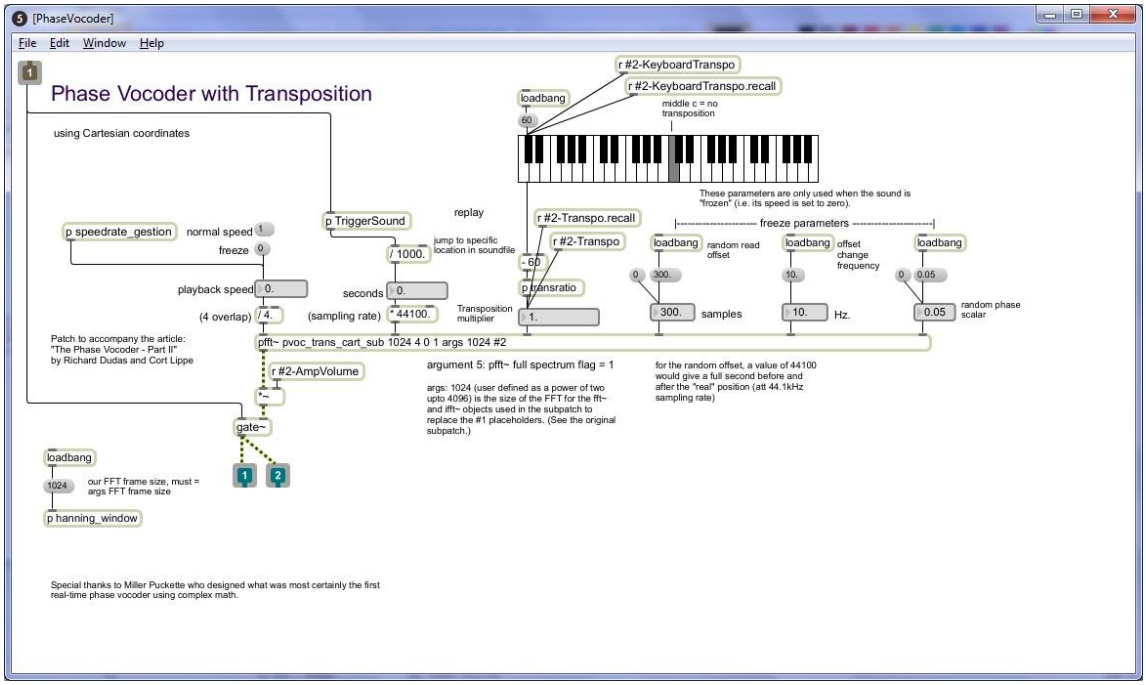
Patch principal de l'instrument MM.Freezer



Patch pfft de lecture et d'analyse de l'échantillon – MM.Freezer



Sous-patch de gestion de vitesse du pointeur de boucle – MM-Freezer



Patch du phase-vocoder – MM.Freezer

B. Montage d'un Chariot MétaFanfare

Dans le cadre du stage à Puce Muse, j'ai été amené à construire un chariot de secours pour la MétaFanfare :

« *La Méta-Fanfare propose d'inventer le SON d'un grand orchestre électronique ambulant. Elle fait appel au Conservatoire du Grand Chalon pour présenter une musique d'espace où les musiciens mobiles défilent à travers le public, encerclent le site, se diluent dans ses méandres, se retrouvent en un point pour concerter avec un orchestre constitué de Spect/Acteurs.*

Symphonie visuelle monumentale en 3D pour une Méta-Fanfare de 16 joystickeurs ambulants, un Méta-Orchestre de 24 gamepadistes, deux chefs d'orchestre, et un Méta-Instrumentiste. La Grande Pictophonie s'écoute avec des lunettes. Composée en 3D, elle prolonge cette aventure des musiques visuelles depuis Scriabine et Xénakis jusqu'au VJing. La Grande Pictophonie est participative. Elle fait appel aux élèves du conservatoire pour constituer une Méta-Fanfare de joysticks et au public pour constituer un Méta-Orchestre de gamepads. La Grande Pictophonie circule entre réel et virtuel. Haut-parleurs embarqués sur le dos des musiciens, ou sons circulants virtuellement d'un haut parleur à l'autre, images réelles détournant l'architecture des bâtiments ou images virtuelles en relief traversant les façades. La Grande Pictophonie est classique et s'écrit en trois mouvements : un appel mobile faisant sonner le site, un concerto pour Méta-Instrument et orchestre de joysticks, et un final en tutti pour deux orchestres, deux chefs et un soliste.»

Ces chariots étaient utilisés pour des déambulations urbaines sur lequel reposait un système comprenant la gestion de la diffusion et de la création de la synthèse sonore à partir d'un Joystick relié vers un ordinateur tournant avec la Meta-Mallette (version 3.0). Les déambulateurs étaient constitués de 6 binômes dont une des deux personnes tirait le chariot, et où la totalité des joueurs portaient sur leur dos leur propre système de sonorisation amplifié (haut-parleur).

Le montage comprenait donc le découpage et l'assemblage des différents matériaux constituant le chariot (plastique, mousse de protection, balais et peinture pour la traction des chariots), ainsi que la partie électrique, permettant de jouer et d'illuminer le chariot à l'aide de LED bleues.

J'ai donc du effectuer tout le raccordement comprenant le soudage des câbles audio (RCA et câble Jack) entre le PC et l'amplificateur de puissance (constitué d'une batterie de voiture et d'un ampli de voiture, raccordé par des câbles XLR habituellement utilisé pour des signaux de type microphone), ainsi qu'entre l'ampli et les haut-parleurs (câbles Speakon pour des niveaux de puissance plus importants).