Université de Valenciennes et du Hainaut-Cambrésis

Rapport de Projet ISIS

LAMMA : Motion Capture & Sound Design

Référent de Projet :
M. Viéville

dream AUDIOVISUEL&MULTIMÉDIA ©

SEGALEN Maël
EL FAKHOURI Amine

Ce compte rendu concerne le projet ISIS de Maël SEGALEN & Amine El Fakhouri, étudiants en Master 1 ISIS. Ce projet a été réalisé pendant l'année scolaire 2015-2016, sous la tutelle de M.Viéville.

Résumé

Ce projet aborde le développement d'un logiciel permettant de générer des bruitages en fonction d'une capture de mouvements. Le logiciel devrait permettre de produire des bandes son pour l'image (Cinéma, Jeux Vidéos, Publicité...) en direct. Une importante partie de ce Compte Rendu détaille nos recherches préliminaires sur la Vraisemblance du Bruitage.

Table des matières

I - Introduction

Background

L'idée de départ m'est venu au cours d'un stage. J'occupais le poste d'Assistant Son dans un studio de Post-Production à Paris. Le studio se nomme "Capitaine Plouf" et est spécialisé dans l'enregistrement, le montage son et le mixage de publicité. Le studio profite d'une certaine réputation et travaille avec des entreprises renommées telles que Peugeot ou L'Oréal. Ces publicités sont ensuite largement diffusées à la Télévision, à la Radio, sur Internet...

La majorité du travail des ingénieurs du son se restreint au studio et ils travaillent la plupart du temps à l'aide de banques de sons.

Cependant, grâce à leur renommée et à leur activité florissante, ils peuvent occasionnellement faire appel à des bruiteurs. J'ai donc eu l'opportunité d'entendre le travail d'un bruiteur, juxtaposé au travail d'un monteur Son.

Observation

Le test a été particulièrement marquant à l'écoute d'un court métrage pour la société AXA. De nombreux chocs d'objets et mouvements humains complexes se produisaient à l'image. J'ai pu écouter une première version, dont la bande son avait été réalisée par un monteur son, à l'aide de banques de son. Ensuite, une nouvelle écoute m'a permis d'entendre le travail d'un bruiteur, sur la même vidéo. La bande son réalisée par le bruiteur était criante de réalisme, le son paraissait beaucoup plus cohérent que lors de la précédente écoute.

Mais... Qu'est-ce qui change ? Ce questionnement est à l'origine du projet ISIS "Sound & Motion", que nous avons commencé en Septembre 2015, Amine El Fakhouri et moi-même (Maël Segalen).

Montage son & Bruitage

Dans l'Audiovisuel, la plupart des bruitages & effets sonores (i.e. les sons autres que la voix et la musique) sont ajoutés en postproduction. Deux techniques coexistent. Le montage son s'appuie sur des banques de sons préenregistrés, qui sont ensuite insérés dans la bande son. A contrario, le métier de bruiteur consiste à enregistrer les sons en studio, en fonction des besoins.

Deux bruiteurs en Studio

A mon sens, les résultats obtenus par bruitage sont bien plus agréables & cohérents (voire "humains"). Cependant, cette technique se perd car couteuse et difficile à mettre en œuvre. Le montage son prédomine dans beaucoup de secteurs comme la publicité, les métrages amateurs, une partie du cinéma...

Problématique

Cette différence de ressenti m'a poussé à me poser plusieurs questions.

Qu'est ce qui rend le travail d'un bruiteur plus "cohérent", "réaliste", "humain" ? Quels paramètres rendent une bande son réaliste ? Quels paramètres détériorent la sensation de réalisme ? Qu'en pensent les professionnels du Son ? Et enfin : est-il possible de proposer une solution à ce problème ?

Objectifs du projet

Au cours de ce projet, nous avons poursuivi ces questionnements, et aimerions proposer une alternative. Nous avons donc développé un outil permettant de produire rapidement des bruitages réalistes. Même si le secteur visé est avant tout le cinéma, des applications à la Réalité Virtuelle ou à la Motion Capture nous paraissent envisageables.

Pour des raisons pratiques, nous n'avons traité que les bruitages relatifs au corps humain (mouvements, pas, saisie d'objets...).

II - Recherche

Introduction

Cette partie du projet s'est déroulée du mois de Septembre jusqu'au mois de Janvier, durant tout le premier semestre de l'année. L'objectif était d'établir une problématique, qui nous permettrait de se poser les bonnes questions. Ces questionnements nous guideraient ensuite jusqu'à la réalisation d'un Cahier des Charges.

Problématique

La problématique présentée ici est similaire à celle énoncée plus haut.

Les questions "Quels sont les paramètres qui rendent un bruitage cohérent, réaliste ?", "Qu'est-ce qui différencie le travail d'un bruiteur et celui d'un monteur son ?", "Qu'en pensent les professionnels du Son ?". Et enfin : "Peut-on jouer sur les caractéristiques d'un Son afin d'en détourner le sens ? Si oui, lesquels ?".

Ces questions ont guidé notre raisonnement durant ce semestre de Recherche.

Les caractéristiques d'un Son

Nous avons commencé par nous intéresser à la question "Comment le bruitage devient-il cohérent ?". Nous nous sommes rapproché d'organismes tels que les bibliothèques, les écoles ou les laboratoires pouvant avoir étudié la question.

Le mémoire de fin d'année de Paul Sabin (étudiant à l'Ecole Nationale Supérieure Louis Lumière, promo 2014) porte très justement sur "La Vraisemblance du Bruitage", ce qui a constitué une très bonne introduction au sujet. Ses recherches ont visé à : **"*découvrir, à travers l'observation du métier* [de bruiteur, ndlr]*, la synthèse d'étude, et un test perceptif, si nous pouvons affecter cette sensation de vraisemblance chez le spectateur.*"** Ses recherches s'avèrent très poussées, allant jusqu'à questionner quelques grands noms du bruitage en France. Il en ressort quatre critères "critiques" ou "dominants" quand à la sensation de cohérence chez le spectateur.

Ces critères sont :

- le **Synchronisme**, défini par les différences de temps entre un geste à l'image et au son
- l'**Acoustique**, concerne la sensation de placement de la source sonore
- la **Matière**, conditionne la ressemblance entre la matière visible d'un objet et la matière audible de ce même objet
- le **Geste**, détermine si un geste à l'image est suivi par un son de même durée, présentant des évolutions similaires

P. Sabin note, après étude de ses résultats, que la matière joue le rôle le plus important dans la sensation de réalisme. Viennent ensuite le Geste, le Synchronisme puis l'Acoustique. Il ajoute que le

critère du Geste, conditionne une grand partie de l'intention des acteurs, au-delà de la sensation de cohérence audiovisuelle.

Nous nous sommes donc penché sur cette dernière idée, stipulant que les intentions d'un acteur sont fortement transmises par le geste, et avons cherché à confirmer ou infirmer ses dires. Nous n'avons retenu que ce critère, pensant que c'était celui qui induisait le plus de différences entre un bruiteur et un monteur son.

Et du coté de la Psychomotricité ?

La psychomotricité, c'est : "*accompagner des personnes ayant des difficultés psychiques exprimées par le corps et notamment la motricité (« psychomotricité ») ou inversement des troubles moteurs perturbant leurs psychismes.*" Les psychomotriciens sont constamment confrontés au lien entre psychique et corporel. Nous avons ainsi voulu savoir s'ils voyaient un lien entre geste et intention, comme l'avait observé Paul Sabin.

E. Moulin est psychomotricienne, en fonction à l'APHP (Hôpitaux de Paris). Après entretien avec elle, et consultation de documents de Recherche, nous avons pu confirmer notre hypothèse. Pour les psychomotriciens, le geste : "*intègre pour sa part l'intentionnalité humaine et l'aspect signifiant dans la communication* [...]". Il fait partie intègre de la Communication Non-Verbale, qui correspond : "*à des gestes, à des postures, à des orientations du corps, à des singularités somatiques, naturelles ou artificielles, voire à des organisations d'objets, à des rapports de distance entre les individus, grâce auxquels une information est émise.*"

Nous validons ainsi notre hypothèse, stipulant que l'intention d'une personne est grandement transmise via ses gestes.

Entretiens avec M.Viéville

Parallèlement à nos recherches, nous avons commencé à consulter M. Viéville, encadrant de notre projet. Son avis nous a permis d'avancer efficacement et de prendre du recul quand nous étions plongés dans notre travail.

Malgré les résultats précédemment énoncés, nous avions tendance à nous éparpiller, particulièrement au moment d'aborder les entretiens avec les professionnels.

Nous avons donc mis en place un protocole de questions, de concert avec M. Viéville, afin de disposer d'une base pour nos entretiens.

Les questions portaient sur deux sujets :

- Quelles différences entre bruitage & montage son ?
- Peut-on réduire ces différences ?

Afin d'avancer efficacement sur ces deux sujets, il nous paraissait important de ne pas trop guider les réponses. Nous évitions ainsi de souffler à l'interlocuteur ce que nous voulions entendre, et pouvions disposer de son avis, de la façon la plus neutre possible. De même, nous voulions interroger un échantillon représentatif. Les entretiens ont donc concerné à la fois des Monteurs Son, des bruiteurs et de profils mixtes.

Nous commencions nos entretiens par des questions larges, en précisant progressivement, et en rebondissant sur les réponses de nos interlocuteurs. Le questionnaire nous ayant servi de base est consultable en Annexe 1.

Entretien avec P. Sabin, O. Chatron & G. le Guen

Armés de notre questionnaire, nous avons posé nos questions à une série de personnes.

Du coté des bruiteurs se trouvaient Sophie Bissantz, Bertrand Amiel (tous deux bruiteurs à Radio France) et Gadou Naudin (bruiteur à M Studio).

Du coté des Monteurs Son se trouvait Guillaume le Guen (Réalisateur Son chez Capitaine Plouf).

Venaient compléter cette liste : Paul Sabin (auteur du mémoire précédemment cité, qui nous a éclairé de son point de vue de chercheur) et Olivier Chatron (présent sur tous les fronts, à la fois Monteur Son et Bruiteur freelance).

Nous avons pu, encore une fois, vérifier le lien entre intention et geste. Paul Sabin précise que le bruiteur "**s'investit corporellement dans le personnage, Il incarne le personnage**".

Cette dernier point nous as permis de mieux comprendre quelles étaient les différence entre le travail d'un bruiteur et celui d'un monteur son. "**L'intention ne peut être que juste car il revit la scène**", ajoute-t-il.

Olivier Chatron étend cette idée : "Le travail du bruiteur est réfléchi en terme de psycho acoustique. **Comment le spectateur va percevoir la chose ?**". Il évoque ensuite le montage son : "**La banque de son est froide et elle ne pense pas**".

Le montage son, n'étant pas joué "en direct" par une personne cherchant à donner une **intention** et à provoquer une **sensation**, souffrirait d'une **intention inexacte**. Cela correspondra au point de départ de notre Cahier des Charges.

Conclusion

Ces quelques mois de Recherche nous ont permis de mieux cerner les tenants et les aboutissants du projet. Les objectifs étaient mieux définis.

Nous avons terminé le premier semestre avec une vue d'ensemble de la profession. Après avoir déterminé ce qui faisait défaut aux bandes son réalisées par montage son, nous avons pu écrire un Cahier des Charges.

Cela nous a aussi permis d'en apprendre plus sur les différentes façons de composer une bande son. Les entretiens avec les bruiteurs nous ont permis d'appréhender leur technique et leurs habitudes, leur "artisanat". Il en a été de même avec les monteurs son.

Nous avons enfin appris qu'il serait impossible (ou bien extrêmement complexe) de "modifier l'intention" d'un son après enregistrement. Nous avons donc abandonné cette idée au profit d'une autre : choisir le son dont l'intention est la plus appropriée.

III - Le Cahier des Charges

Il nous a paru important d'établir un court Cahier des Charges. Nous gardions ainsi la même optique que précédemment : organiser le travail, et ne pas perdre de vue les objectifs.

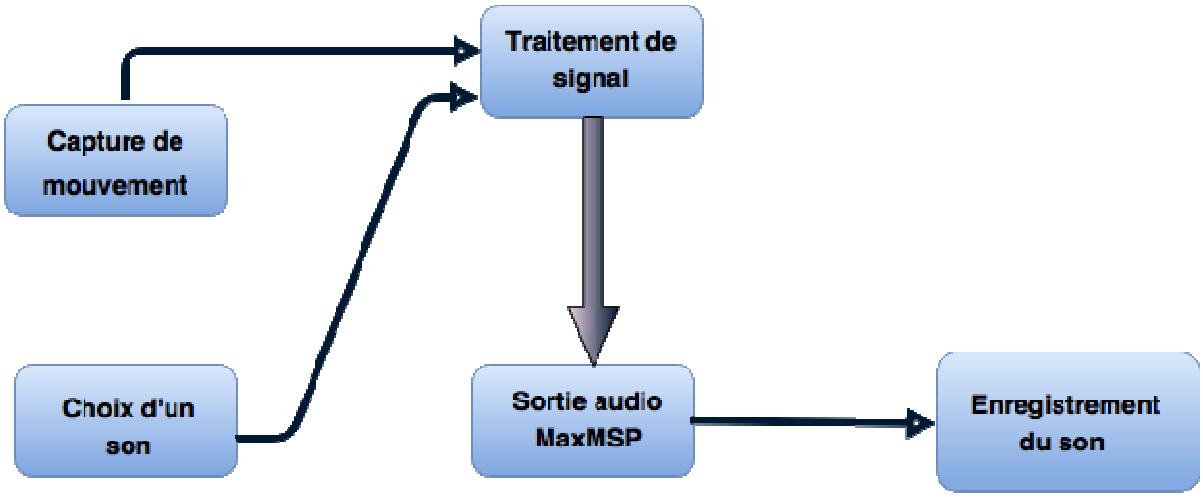
Grâce à nos recherches, nous avions toutes les cartes en main. Nous voulions développer un logiciel destiné aux Monteurs Son, qui s'approche de la logique du bruiteur. Il devait permettre de produire des bruitages dont l'intention était choisie, et soit cohérente avec l'image.

Nous avons listé les besoins du logiciel comme suit :

- En **Entrée** : les **gestes** d'un opérateur. Nous serions ainsi en mesure d'identifier l'**intention** voulue
- Le **Traitement du Signal** : le logiciel devra permettre de scanner une banque de sons et de les classer, en fonction d'un critère choisi, en relation avec l'intention
- En **Sortie** : un flux **audio**

"Identifier l'intention" est un terme que nous trouvons particulièrement flou, dans une démarche de Développement Technique. Nous nous sommes donc focalisé sur un cas particulier. Les bruits de pas sont des sons faciles à manipuler, car ils sont constitués d'une enveloppe sonore. En terme d'intention, nous avons établi une échelle de vitesse, qui nous permettait de déterminer si le pas est calme (lent) ou brutal (rapide).

Ces informations sont regroupées sur le diagramme ci-dessous.



Fonctionnement Global du Projet

IV - Recherche Technique

Introduction

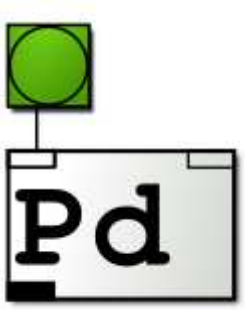
Le fonctionnement global du logiciel défini, nous avons effectué un travail de Recherche Technologique. Nous avons ainsi comparé les différentes options qui s'offraient à nous afin de choisir celle qui nous correspondrait le mieux. Sont regroupées ci-dessous les principaux logiciels que nous avons passé en revue, ainsi qu'une description.

La connaissance de tous ces logiciels n'est pas primordiale pour la compréhension de notre code, mais il nous a paru intéressant de condenser ce travail, qui s'apparente à une Veille Technologique. Nous désirons ainsi fournir à d'autres étudiants intéressés par la Motion Capture et/ou le Sound Design une base de travail utilisable.

Recherche Technologique

Pure Data

https://puredata.info/



PureData, souvent abrégé Pd, est un logiciel de création multimédia interactive en temps réel, permettant la gestion de l'image, du son ou des données.

Pure Data tire son origine de l'éditeur Patcher, écrit par Miller Puckette en 1988, Patcher a ensuite été réécrit par David Zicarelli, sous le nouveau nom de MAX/MSP.

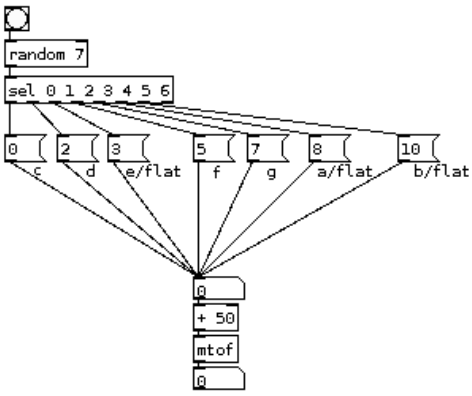
Miller Puckette a cependant décidé de reprendre la conception de Patcher pour en faire un nouveau logiciel, libre et transportable, et il le maintien encore aujourd'hui sous le nom de PureData.

Couramment utilisé dans les domaines artistiques, scientifiques et pédagogiques, sa popularité réside dans sa facilité d'utilisation.

Plutôt qu'un langage de programmation textuel, Pure Data propose un environnement de programmation graphique dans lequel l'utilisateur est invité à manipuler des icônes représentant des fonctionnalités et à les brancher ensemble.

Il permet donc de manipuler des flux de données en temps réel : analyse, transformation, synthétisation d'audio, de vidéo, de texte ou des données issues d'un ou plusieurs capteurs.

Un programme PureData se présente sous la forme suivante :



Exemple de Patch PureData

On appelle ces fichiers d'extension .pd des patchs (ou moins couramment « canevas » en français). Dans sa version textuelle (fichier .txt), on peut voir que le mot « canvas » est suivi de l'appel des créations d'objets et de leurs connectons.

Cette économie d'informations rend ces fichiers très légers et permet les échanges de patchs sur internet très faciles : en copiant le texte et en l'enregistrant sous l'extension .pd, on obtient un fichier PureData valide.

Max MSP

https://cycling74.com/

Max/MSP comme son ancêtre PureData est un environnement graphique pour la programmation d'applications sonores, visuel et interactives.

Il résulte de l'association de deux logiciels:

- MAX pour le contrôle en temps-réel d'applications musicales et multimédia interactives par des événements MIDI.
- MSP (Max Signal Processing) : la bibliothèque d'objets pour l'analyse, la synthèse et le traitement de l'audio en temps réel.

Conçu par l'IRCAM et Cycling'74, MAX/MSP est destiné aux artistes, aux musiciens, aux designers sonores, aux enseignants ou encore aux chercheurs.

Cet outil part d'un principe assez singulier: dans un espace de travail similaire à une page blanche, vous allez glisser, puis connecter entre-elles, des boîtes auxquelles vous attribuerez un rôle spécifique: les entrées (inlets) et sorties (outlets) permettront à ces boîtes (opérateurs mathématiques, contrôleurs, filtres, visualiseurs, etc.) de générer, recevoir ou diffuser des

informations et des valeurs, constituant ainsi des outils de traitement de différentes natures (signaux MIDI, audio, vidéos, données…).

Ce principe de travail peut rappeler les bases de conception d'un synthétiseur analogique.

Nous parlons donc ici de création sonore et de programmation. Même si l'absence de connaissance des langages de programmation ne constitue pas un handicap, il faudra cependant plonger dans l'indispensable documentation.

Kinect

Conçu par *Microsoft,* la *Kinect* est un périphérique destiné à la base à la console Xbox 360 permettant de réaliser de la capture d'image 3D. Elle est constitué d'une base motorisée, une caméra couleur RGB, un capteur de profondeur, une rangée de microphones, et enfin un ensemble logiciel permettant la reconnaissance de mouvement.

Ce périphérique est dotée d'un capteur de profondeur qui permet la mise en relief de l'image saisie par la *Kinect* est en fait constitué d'un projecteur laser infrarouge et d'une mini caméra CMOS monochrome. Ce capteur de profondeur permet la capture d'image en 3D, quelles que soient les conditions lumineuses.

La pile logicielle embarquée sur la Kinect, permet de réaliser de la reconnaissance de gestes du corps humain, la reconnaissance faciale et la reconnaissance vocale, en utilisant des techniques d'interaction par commande vocale, reconnaissance de mouvement et d'image développées *PrimeSense.*

La Kinect est capable de tracker jusqu'à 6 personnes en même temps. Pour la personne activement reconnue (l'équivalent du joueur), 20 positions d'articulation sont interprétées.

Microsoft fournit aussi le kit de développement logiciel Kinect pour *Windows 7*, Ce *SDK* a été conçu pour permettre aux développeurs d'écrire des applications *Kinect* en *C ++ / CLI, C # ou Visual Basic .NET.*

Capteur :
Lentilles détectant la couleur et la profondeur
Micro à reconnaissance vocale
Capteur motorisé pour suivre les déplacements
Champ de vision :
Champ de vision horizontal : 57 degrés
Champ de vision vertical : 43 degrés
Marge de déplacement du capteur : ± 27 degrés
Portée du capteur : 1,2 m – 3,5 m (à partir de 50 cm)
Flux de données :
320 × 240 en couleur 16 bits à 30 images par seconde
640 × 480 en couleur 32 bits à 30 images par seconde
Audio 16 bits à 16 kHz

Système de reconnaissance physique :
Jusqu'à 6 personnes et 2 joueurs actifs (4 joueurs actifs avec le SDK 1.0)
20 articulations par squelette
Application des mouvements des joueurs sur leurs avatars
Audio :
Suppression de l'écho
Reconnaissance vocale multilingue

Processing

Processing est un environnement de développement libre (sous licence GNU GPL) basé sur la plate-forme Java, qui fonctionne sur les plates-formes Windows, Linux, Mac (et sur toute autre plate-forme pouvant faire fonctionner des logiciels conçus en Java).

Il a été conçu au laboratoire Aesthetics + Computation Group (ACG) du MIT Media Lab par Ben Fry et Casey Reas en 2001.

Conçu par des artistes, et pour des artistes, Processing est un des principaux environnements de création utilisant le code informatique pour générer des œuvres multimédias.

L'attrait de ce logiciel réside dans sa simplicité d'utilisation et dans la diversité de ses applications : image, son, applications pour le web et mobiles, conception d'objets électroniques interactifs.

Processing possède la particularité d'utiliser des instructions informatiques pour dessiner, réaliser des animations en 2 ou 3 dimensions, créer des œuvres sonores et visuelles, concevoir des objets communicants qui interagissent avec leur environnement. Ce mode d'expression artistique par le code utilise les caractéristiques propres à l'informatique (rapidité d'exécution, automatisation des actions et des répétitions, interaction, etc.)

Processing permet également de programmer des circuits électroniques qui interagissent avec le milieu qui les entoure.

Connectés à des capteurs sonores, de mouvement, ou autres, des microcontrôleurs peuvent en retour générer des images, actionner un bras articulé, envoyer des messages sur Internet.

Processing fédère une forte communauté d'utilisateurs professionnels et amateurs : artistes, graphistes, vidéastes, typographes, architectes, web designers et designers en général.

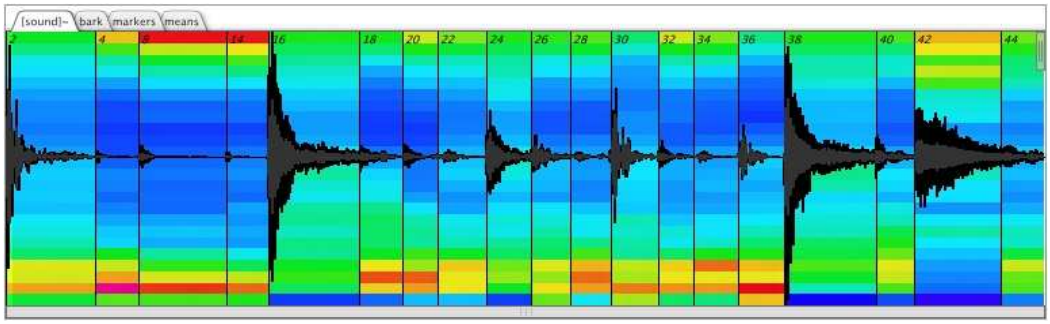
MuBu

Containeur de données sonores et de mouvement sous forme d'objets Max.

MuBu multi-buffer est un containeur de données sonores et de mouvement développé par l'IRCAM. Le rôle de cet outil est de fournir de la mémoire structurée pour le son et le mouvement enregistrés à travers des interfaces et opérateurs en temps réel en tant qu'objets externes complémentaires pour *Max.*

Les buffers d'un containeur *MuBu* associent des pistes multiples de données alignées à des structures de données complexes comme : données audio segmentées avec descripteurs et annotation, ou alignées, ainsi que des données de mouvement de captation annotées.

Chaque piste d'un *MuBu buffer* peut représenter un flux de données échantillonné ou une séquence d'événements temporels étiquetés, comme par exemple : des échantillons audio, des descripteurs audio, des données de mouvement de captation, des marqueurs, des segments, et des événements musicaux.



Exemple de segmentation par attaque

Un conteneur *MuBu* peut importer des données à partir de divers formats : AIFF, RIFF, Ogg/Vorbis, SDIF (son format natif), texte, fichiers MIDI standard.

On trouve aussi *MuBu pour Max* qui est distribué comme un ensemble d'externes fournissant des fonctionnalités pour la visualisation, la manipulation et l'enregistrement des structures de données MuBu :

Externes	Fonction
MuBu	conteneur multi-buffer
imubu	conteneur avec interface graphique
mubu.track	accès optimisé à une piste
mubu.record, mubu.record~	enregistrement de flux de données et de séquences
mubu.process	traitement de flux de données

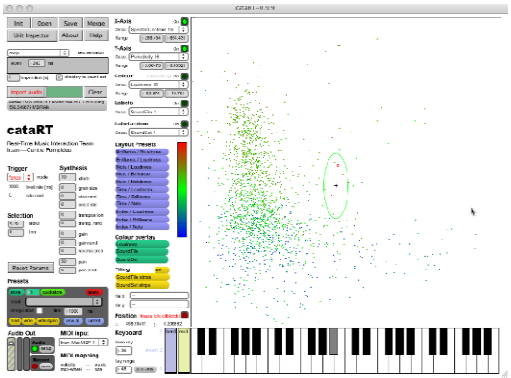
CataRT

Une approche innovante pour la synthèse sonore basée sur des représentations multicritères des échantillons contenus dans une base de données

Développé par l'IRCAM *CataRT* est un système de synthèse concaténative temps réel qui permet de jouer des « grains » sonores (à partir d'un grand corpus de sons segmentés et analysés par descripteur) selon la proximité avec une cible dans l'espace descripteur, au moyen d'une souris ou d'un contrôleur externe.

Il se présente comme une application standalone sur Mac OS X, il est aussi implémenté dans MaxMSP à travers l'extension FTM&Co.

CataRT peut être vue comme une extension de la synthèse granulaire en donnant un accès à des caractéristiques sonores spécifiques. L'interaction repose sur une interface simple consistant en l'affichage d'une projection 2D de l'espace de descripteurs, et une navigation avec la souris, où les grains sont sélectionnés et joués par proximité géométrique.

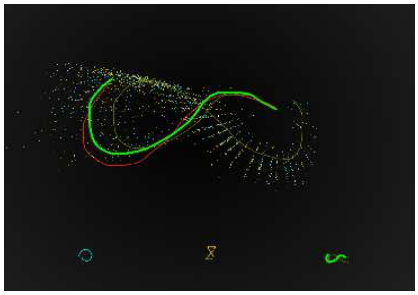


Exemple de synthèse granulaire avec CataRT

Domaines	applications
général	Analyse et visualisation et jeu interactif avec des sons et exportation des données d'analyse
Cinéma, vidéo, postproduction	re-synthèse et modification d'ambiances et de textures sonores environnementales, recherche d'événements ou singularités dans de longs enregistrements et séparation de bruits parasites
Design sonore	contrôle par descripteurs perceptifs de haut niveau, brassages et juxtapositions d'un nombre massif de sons et flexibilité des traitements granulaires
Composition	tous les paramètres de description et transformation sonore accessible, composition par navigation interactive dans un espace de descripteurs sonores et communication avec séquenceurs MIDI

Gesture follower

http://imtr.ircam.fr/imtr/Gesture_Follower



Gesture follower est un ensemble de modules *Max / MSP* qui permet d'effectuer la reconnaissance de gestes en temps réel. Il est intégré dans la boîte à outils MnM de la bibliothèque FTM.

Cet outil se base sur l'idée d'obtenir des paramètres de comparaison entre une performance devant un capteur et un ensemble d'exemples préenregistrés en utilisant des techniques d'apprentissage automatique.

Dans *Max / MSP* les données alimentant *Gesture follower* peuvent parvenir des paramètres sonores (hauteur, amplitude, etc.), d'une souris, joystick, coordonnées, des paramètres de suivi vidéo (EyesWeb, Jitter, etc.), Capteur (Wiimote, Kinect), MIDI, ou toute combinaison de ce qui précède (système multimodal).

Exemple d'utilisation de Gesture Follower

Gesture follower répond aux deux questions suivantes :

- La similitude du geste effectué à des gestes de vraisemblance : De quel geste s'agit-il ? (Réponse directe : noir ou blanc. Réponses intermédiaire "niveaux de gris" : approximation par rapport aux gestes enregistrés), ce qui permet la sélection du geste le plus probable à tout moment.

- La progression temporelle du geste effectué :Où sommes-nous ? (Début, milieu ou fin du geste), ce qui permet l'estimation de l'indice temporel à l'intérieur du geste en cours.

Ces données de sortie sont particulièrement bien adaptés (mise à jour en continue) à la fois pour la sélection et la synchronisation des différents processus visuels ou sonores continus avec des gestes.

FAAST

Flexible Action and Articulated Skeleton Toolkit

http://projects.ict.usc.edu/mxr/faast/

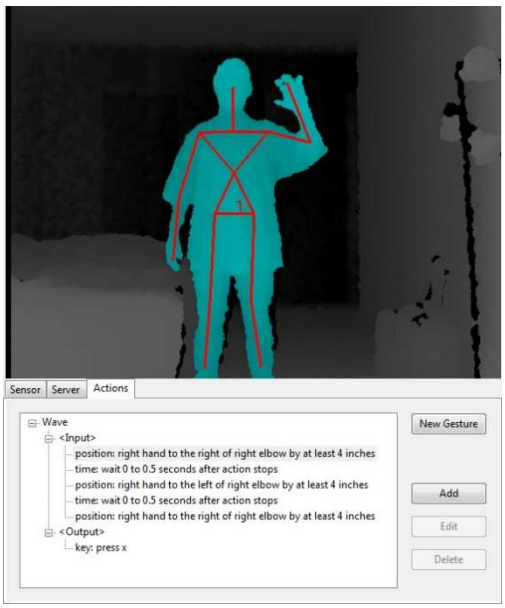
FAAST est un *middleware* qui permet de faciliter l'intégration du contrôle complet du corps avec des jeux et des applications de réalité virtuelle. Il utilise soit *OpenNI*, soit le logiciel *Microsoft Kinect* de tracking de squelette.

FAAST utilise un serveur personnalisé *VRPN (Virtual Reality Peripheral Network)* pour diffuser jusqu'à quatre squelettes d'utilisateurs sur un réseau, permettant aux applications de réalité virtuelle de lire les articulations du squelette.

Cette boîte à outils peut également émuler l'entrée clavier par la posture du corps ou des gestes spécifiques. Cela permet à l'utilisateur d'ajouter des mécanismes personnalisés de contrôle aux celles déjà existante dans des jeux qui ne fournissent pas de support officiel pour capteurs de profondeur.

Pour la conception et le mapping des gestes, *FAAST* utilise une interface utilisateur graphique. Plusieurs événements d'entrée et de sortie peuvent être spécifiés simultanément ou en séquence.

Tout événement répertorié immédiatement après un événement précédent est considéré simultané (par exemple bouger deux mains vers l'avant à la fois). Deux ou plusieurs événements séparés par un temps seront traités comme des séquences, permettant la réalisation des mouvements plus complexes sur une période de temps (dessin d'un symbole spécifique dans l'air).



Exemple d'utilisation de FAAST

FAAST est libre d'utiliser et de distribuer, et disponible uniquement sous *Windows*. Il suffit d'installer *Microsoft Kinect for Windows 1.8 runtime* et de vérifier si on dispose du *Microsoft Visual C ++ 2012 Redistributable Package* pour le faire fonctionner avec le capteur *Kinect*.

OpenNI/NiTE

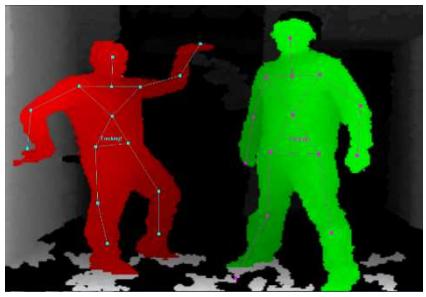
Cette interface de programmation *API* donne accès à tous les capteur de profondeur compatibles *PrimeSense* (*Microsoft Kinect*). Il permet à une application (*middleware*) d'initialiser un capteur et recevoir la profondeur, RGB et flux vidéo IR de l'appareil. Il propose une interface unique et unifiée à des capteurs et fournit des enregistrements *.ONI* créés avec des capteurs de profondeur.

Le PrimeSense Nite est un middleware basé sur la norme *OpenNI* et reconnu pour sa performance en terme de ressources (optimisé pour des architectures *x86 et ARM*) et compatible avec plusieurs plateformes (*Windows, Linux, Mac OS and Android*).

Il permet de récupérer des données très élaborées en plus de la profondeur, la couleur, IR et des informations audio, ce qui lui permet d'exécuter des fonctions telles que la localisation de la main et le tracking. Il dispose aussi d'un analyseur de scène (séparation des utilisateurs de fond), et précise le tracking des articulations du squelette.

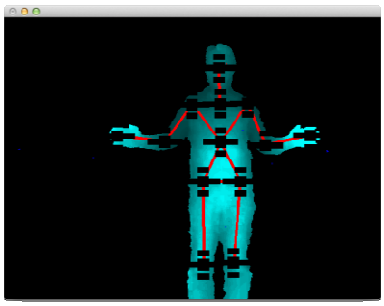
dp.kinect

dp.kinect est une extension de l'environnement de développement Max/MSP basé sur la plate-forme officielle de *Kinect for Windows SDK* qui permet de contrôler et recevoir des données à partir d'un capteur *Microsoft Kinect.*



Il prend en considération : la profondeur, l'image de couleur, le tracking de squelette et du visage, le tracking sonore, la reconnaissance vocale, les nuages de points, les accéléromètres…

OSCeleton



Il s'agit d'un serveur *OSC* (*Open Sound Control*) qui se connecte au serveur *OpenNI* pour transférer les coordonnées des articulations du squelette par l'intermédiaire de messages *OSC* à d'autres programmes (*Pure Data, MaxMSP, Processing*).

Rewire

Rewire est un protocole d'échange d'informations entre plusieurs logiciels d'édition musicale (Ableton Live, Cubase, Pro Tools…). Développé par Propellerhead, il permet la synchronisation de la lecture et des évènements au sein du projet musical en temps réel entre deux applications compatibles (client/serveur ou mixer/synth).

Principe: l'application maître doit être ouverte en premier avant le lancement de l'application cliente qui reconnait la présence d'un hôte Rewire et se configure automatiquement. Il ne reste plus qu'à créer une piste MIDI dans le logiciel hôte, et choisir le(s) Son(s) offert(s) par le logiciel esclave.

Les pistes MIDI maître du logiciel envoient des informations par le protocole ReWire aux pistes des logiciels esclaves. Ceux-ci renvoient du son aux pistes du logiciel maître.

Les canaux Rewire peuvent être utilisés pour gérer des effets send-return.

Conclusion

Nous voulions trouver un système rapide à mettre en œuvre, afin de pouvoir proposer un prototype avant la fin de l'année.

En ce qui concerne la capture de mouvements, la kinect nous paraissait être la solution la plus simple.

Max/MSP & PureData nous paraissaient être de bonnes solutions car nous pouvions réaliser rapidement des patchs, en écrivant des codes très simplement. De plus, ces logiciels ont rassemblé de nombreux utilisateurs. Les communautés sont donc très actives, et partagent beaucoup de connaissances, librairies, patchs...

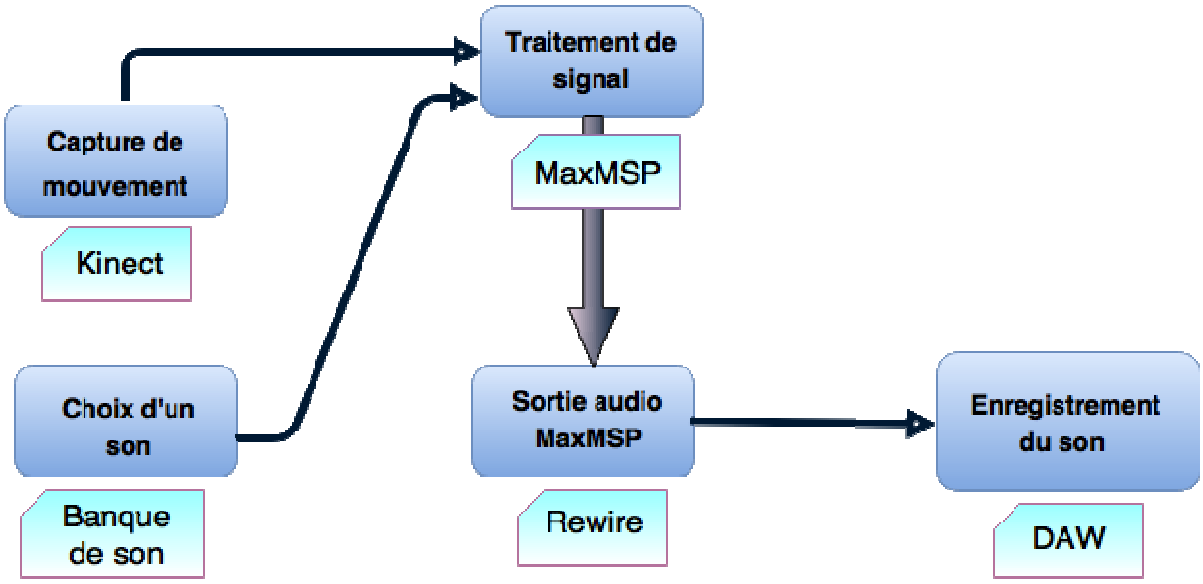
Nous avons plus tard pris connaissance des ressources publiées par l'IRCAM pour MaxMSP. Utiliser des librairies telles que *Gesture Follower* ou *MuBu* nous permettait de développer le logiciel plus rapidement, nous avons donc choisi MaxMSP pour le Traitement du Signal.

Passer en revue ces logiciels nous a permis de mettre le doigt sur un problème : nous ne savions pas comment relier la captation de mouvements (via la kinect) au logiciel censé traiter le signal.

Nous avons commencé le développement par la mise en série d'un code Processing et d'un patch Max/MSP. Le code Processing nous permettait de récupérer les coordonnées de l'exosquelette et les envoyer sur le réseau via le protocole OSC (Open Sound Control, une surcouche du protocole UDP). Le patch Max/MSP nous permettait de traiter les données comme bon nous le semblait. Nous avons cependant rencontré des problèmes de latence du à l'utilisation de Processing et de l'OSC. Ce problème est décrit en détails plus loin.

Nous avons plus tard trouvé une solution, en utilisant une librairie de Max/MSP nommée dp.kinect. Cela nous a forcé à reprendre une partie du code mais nous profitons depuis d'une solution plus simple et plus performante.

C'est cette configuration que nous avons conservé ces derniers mois. Vous trouverez ci-dessous un nouveau diagramme, amélioré par des précisions concernant les technologies.



Fonctionnement Global du Projet, et Technologies utilisées

Cours de Pure Data

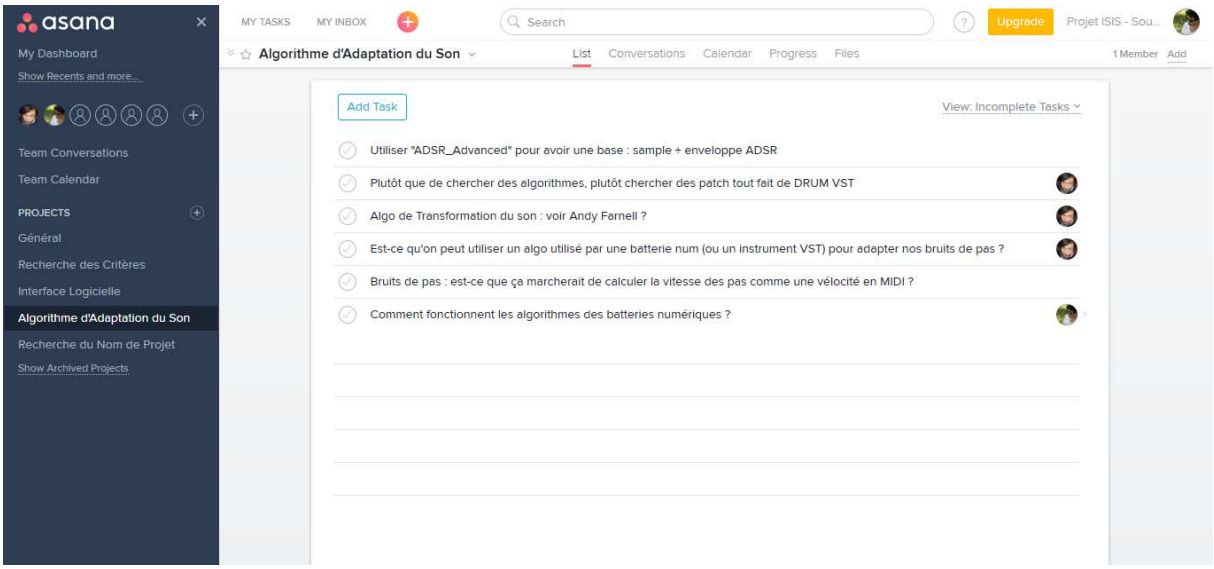
En début d'année, M. Coudoux nous a offert l'opportunité d'aller étudier les "Technologies Créatives" à l'Université de Mons. Nous avons saisi cette chance pour nous initier à PureData, une fois par semaine, huit semaines durant. Nous ne savions pas si nous allions utiliser PureData ou Max/MSP à cette période, mais comme ces deux logiciels partagent le même fonctionnement, il nous a paru judicieux d'y aller.

V - Le Développement

L'organisation du travail

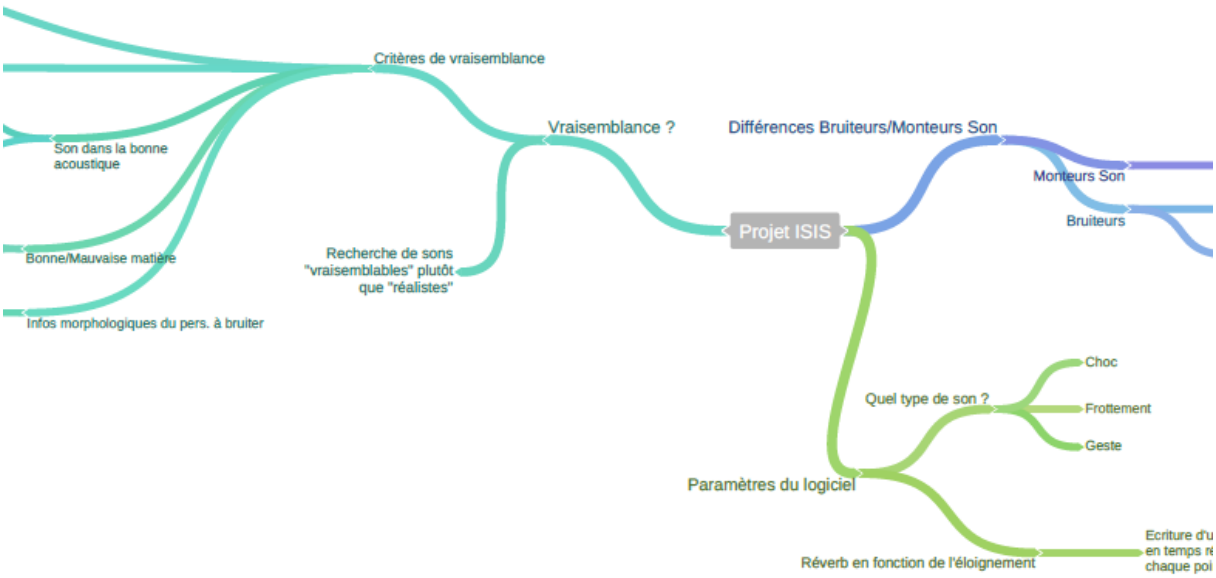
Tout d'abord, un rapide point sur notre organisation générale.

Le Cahier des Charges nous a permis de définir une suite de petites tâches, réalisables plus facilement. Plusieurs services de gestion des tâches collaboratives, nous ont permis de travailler en équipe.



Gestion des tâches au sein du binôme

Parallèlement à ces services, M. Viéville nous a proposé d'utiliser la méthode du "*Mind-Mapping*" (ou Carte Heuristique) pour organiser nos pensées.



Une partie de la Carte Heuristique du Projet

Grâce à cette organisation, nous avons pu établir un planning et ne pas perdre de vue nos objectifs.

L'Algorithme

Nous allons ici détailler l'Algorithme de notre patch Max. Il doit répondre aux besoins suivants :

- Détecter un mouvement (ici : "baisser un pied")
- Calculer à quelle vitesse le mouvement est réalisé
- Détecter si le pied est posé au sol
- Jouer un sample, choisi en fonction de la vitesse du pied

Nous proposons tout d'abord un diagramme général présentant notre façon d'appréhender le problème.

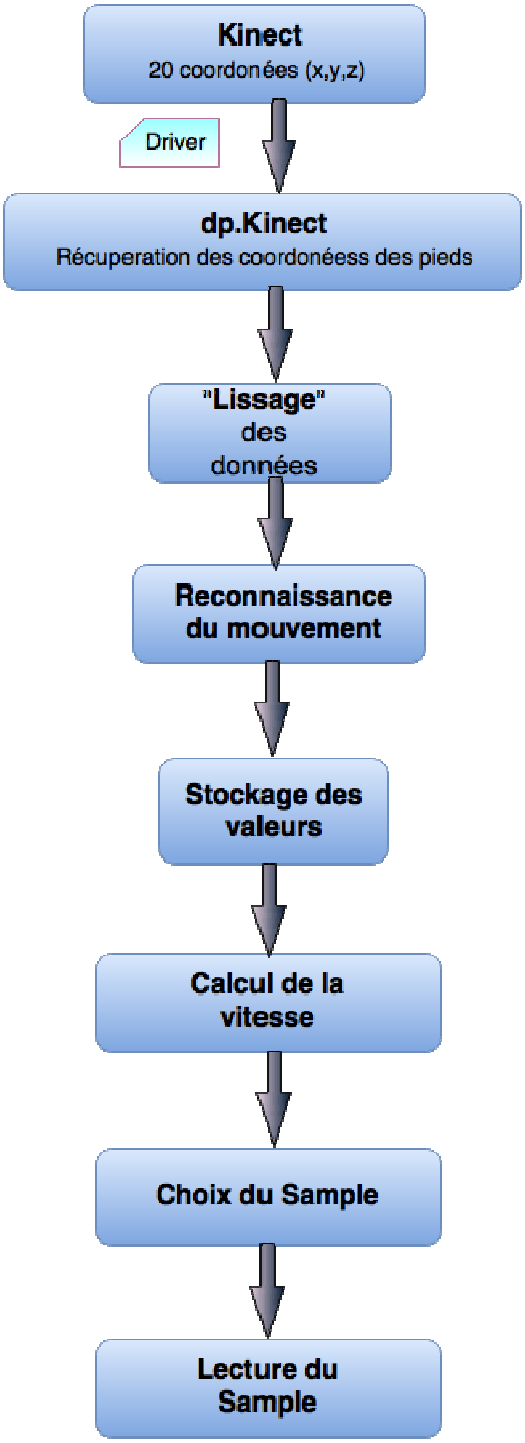
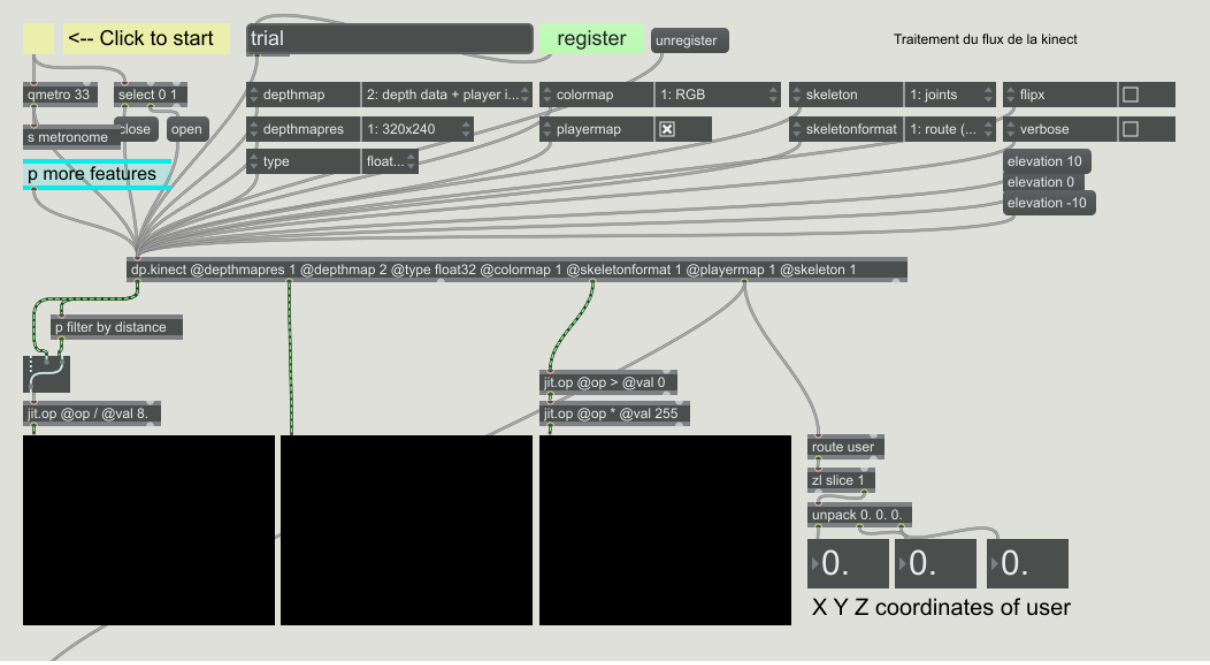


Diagramme du fonctionnement de l'Algorithme

Nous allons détailler ci-dessous chaque partie du code indépendamment.

Lien avec la Kinect et Capture des Données



Capture d'écran de la récupération des données de la kinect

Ce patch fait partie de la librairie "dp.kinect", que nous avons implémenté dans notre patch.

En haut à gauche de ce patch se trouve tout d'abord une boite cliquable, qui permet d'initialiser la connexion avec la kinect.

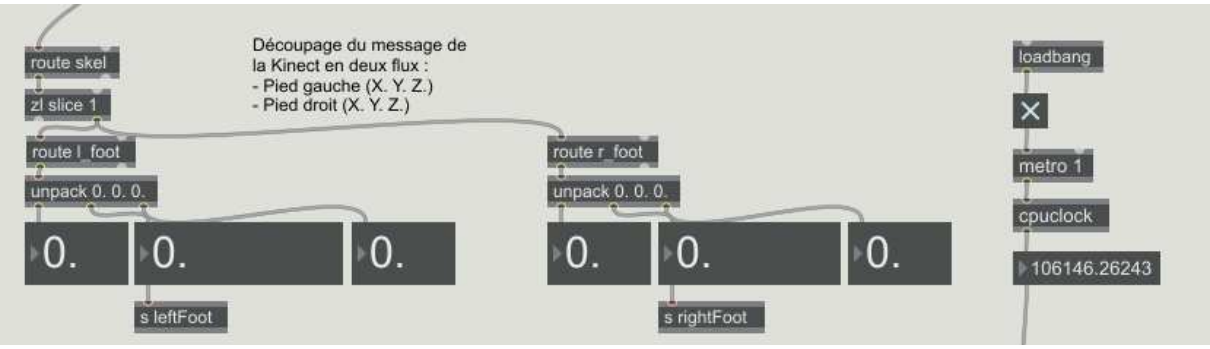
La fonction "qmetro" permet d'envoyer au reste du patch les coordonnées (X,Y,Z) des 20 joints de l'exosquelette, ici chaque 33 millisecondes. Nous disposons donc d'environ 30 informations par joint chaque seconde.

S'affichent enfin trois flux vidéo :

- La Profondeur
- La Couleur
- L'exosquelette

Les coordonnées (X,Y,Z) de tous les joints sont transférés vers la suite du patch.

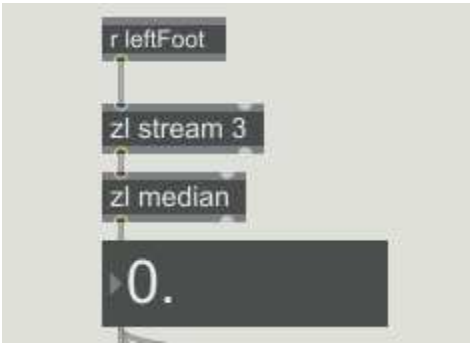
Tri et Affichage des valeurs



Capture d'écran du patch - Tri des valeurs

Ici est effectué le tri des valeurs. Le début du patch (les fonctions "route" et "zl.slice") nous permet de ne conserver que les messages concernant les pieds (les messages commençant par "l_foot" et "r_foot", respectivement pied gauche et pied droit). Via la fonction "unpack" et les boites d'affichage de chiffres en dessous, nous pouvons visualiser les coordonnées (X,Y,Z) en temps réel. L'afficheur de gauche nous fournit les coordonnées du pied gauche, celui de droite nous fournit les coordonnées du pied droit.

Sur l'extrême droite de la capture d'écran, une suite de fonctions est responsable de l'envoi de l'heure du système (fonction "cpuclock") au reste du patch. L'heure est récupérée et envoyée chaque milliseconde.

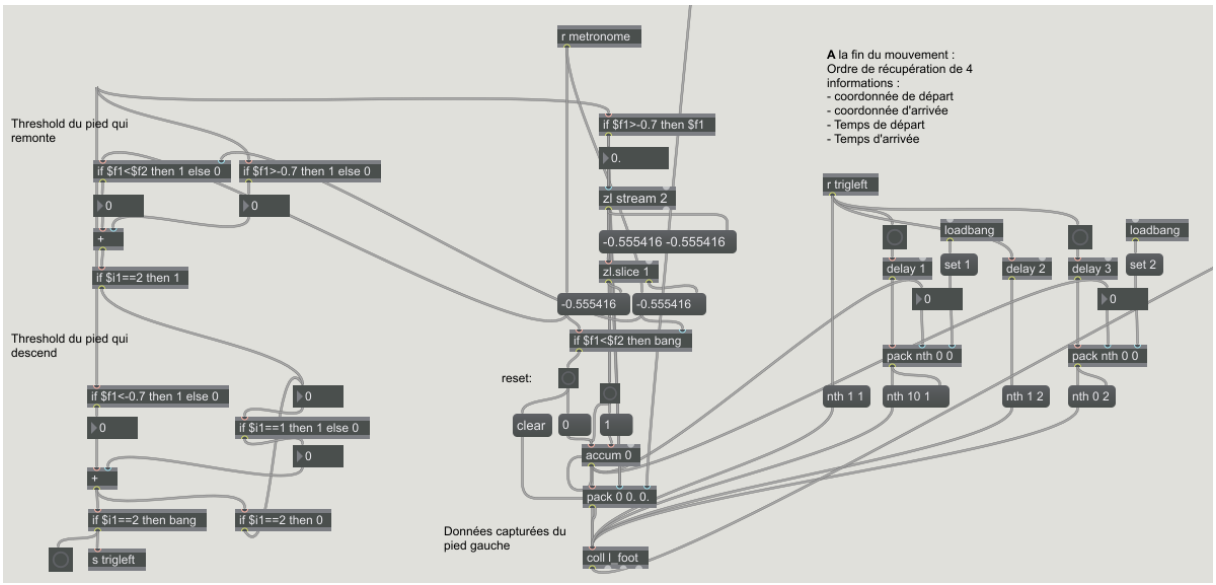


Capture d'écran du patch - Lissage des Valeurs

Ci-dessus, une courte partie du patch, servant à "lisser" les valeurs de la kinect. Nous nous sommes aperçus que les valeurs pouvaient parfois varier brutalement. Pour pallier à ce problème, nous avons intégré ces quelques fonctions, dont voici le fonctionnement :

- récupération des valeurs (fonction "receive", abréviée "r")
- stockage en série de trois valeurs consécutives
- calculs de la moyenne des trois valeurs à chaque coup d'horloge (nous sommes toujours synchronisés à la fréquence de 30 coups par seconde)

Détection et Stockage des valeurs



Capture d'écran du patch - Détection & Stockage des valeurs

Cette portion du patch est, avouons-le, très peu claire. Cependant, il ne nous paraît pas judicieux d'étudier son fonctionnement bloc par bloc et allons donc présenter le fonctionnement global de ceux-ci.

Sur la partie gauche de l'image, deux "*thresholds*" (en français "seuils") sont définis. Chaque partie fonctionne de façon simple : quand les valeurs d'entrée passent en-dessous du seuil, une valeur est passée de 0 (booléen *faux*) à 1 (booléen *vrai*).

Ainsi, quand les valeurs en ordonnée d'un pied montent, le système est "armé". Si le système est "armé" et que le pied passe en-dessous de la valeur -0.7, un message est envoyé à la suite du patch. Passer en-dessous de cette valeur "désarme" aussi le système. Si l'utilisateur veut envoyer un nouveau message, il est forcé de remonter le pied pour "réarmer.

Ce système nous permet de détecter quand le pied est posé au sol, et de le détecter une fois seulement.

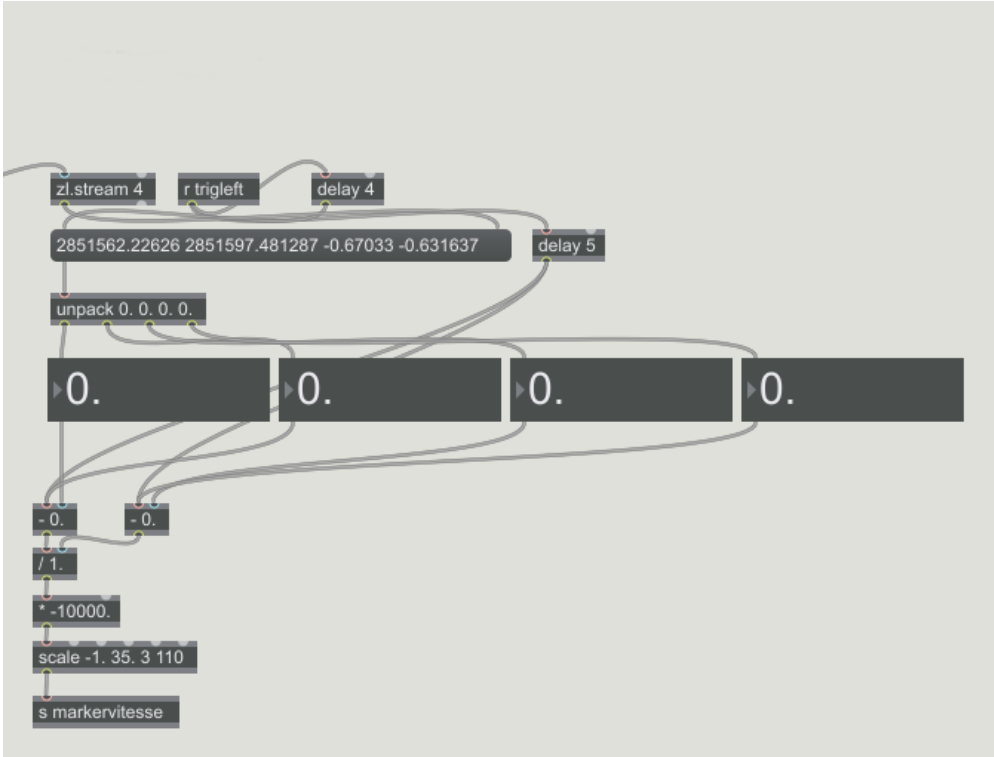
Dans un second temps, un stockage des données du mouvement est effectué. Les données en ordonnée du pied ne sont stockées que si les valeurs décroissent. Nous avons ainsi accès à l'intégralité du mouvement, du haut en bas. Si les valeurs commencent à remonter, toutes les valeurs stockées précédemment sont effacées, car cette partie du mouvement ne nous intéresse pas.

Revenons un peu en arrière. Nous avons précédemment parlé d'un message envoyé au reste du système. Ce message est envoyé quand le pied est détecté comme étant "posé au sol". Il consiste en un déclencheur, qui demande quatre valeurs :

- la valeur en ordonnées du début du mouvement
- la valeur en ordonnées de la fin du mouvement
- l'instant auquel le mouvement à débuté
- l'instant auquel le mouvement s'est arrêté

Ces données récupérées, nous arrivons naturellement au patch du calcul de la vitesse.

Calcul de la Vitesse



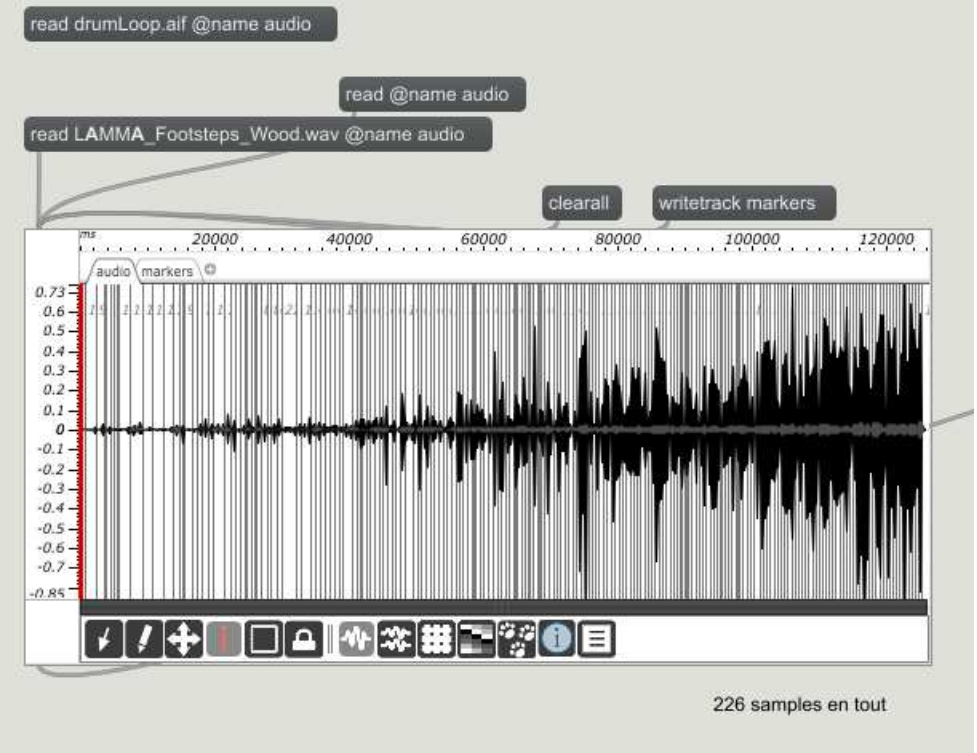
Capture d'écran du patch - Fonctions de calcul de la vitesse

Les données récupérées précédemment nous permettent de calculer la vitesse du mouvement. La formule ci-dessous est utilisée :

$$v = \frac{y_2 - y_1}{t_2 - t_1}$$

Le calcul nous renvoie des valeurs comprises entre 0 et 35. L'objet "scale" nous permet de changer d'échelle. Ici, nous passons à des valeurs comprises entre 3 et 110. Cette valeur sera utilisée plus tard pour choisir le sample à utiliser.

Lecture et Découpage du sample



Capture d'écran du patch - Troncature du fichier audio

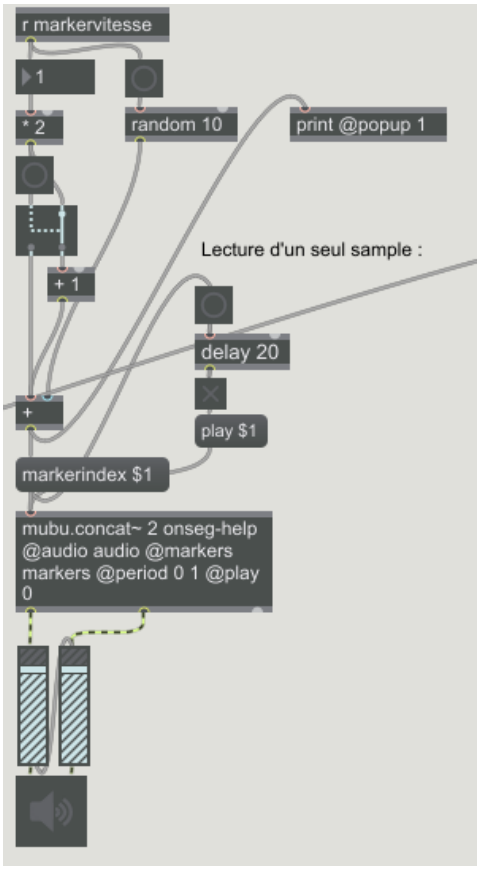
Comme nous l'avons précisé plus haut, le programme réalise le choix d'un sample parmi toute une collection. Cette partie du patch utilise la librairie MuBu (pour *multi-buffer*, qui est détaillé dans la partie "Recherche Technologique"). Cette dernière offre des objets Max dédiés au découpage, au traitement et à la manipulation de sons, dont *imubu*, que nous utilisons ici. Cet objet prend en entrée un fichier son (ici, une prise de sons de multiples bruits de pas) et le découpe selon des paramètres prédéfinis. Ces paramètres sont brièvement décris ensuite.

Paramètres du Découpage du sample

Capture d'écran du patch - Paramètres de l'objet imubu

Nous ne rentrerons pas dans les détails de ces objets, car une grande partie de ceux-ci ne nous ont pas servis. Nous attirons cependant l'attention sur les boites "onseg.threshold" et "onseg.offthresh", qui conditionnent le découpage des samples. Ils ont été réglés dans le but de séparer au mieux chaque bruit de pas, malgré les différences d'intensité et de dynamique de chacun d'entre eux.

Aléatoire et sortie Audio



Capture d'écran du patch - Valeurs aléatoires et Sortie Audio

Ici, nous joignons deux parties du patch : le fichier audio découpé (samplé), et la valeur de la vitesse. Nous ajoutons tout d'abord un peu d'aléatoire à la variable "vitesse". Elle se voit ajouter une valeur comprise entre -5 et +5. De plus, la valeur est forcée à être successivement paire, puis impaire, puis paire, etc. Nous choisissons ainsi successivement forcément un bruit de pas de pied gauche, puis de pied droit dans le fichier audio. Rajouter de l'aléatoire dans ces valeurs nous permet de choisir des samples toujours différents, et de rendre la démarche plus "humaine".

La valeur finale nous sert à choisir parmi les samples découpés dans le fichier audio. Nous savons que la valeur finale de la variable est au maximum de 226. Nous avons calibré nous même cette valeur, car nous savons qu'elle correspond au nombre total de bruits de pas dans le fichier.

Enfin, nous choisissons le sample possédant le même numéro que notre variable de vitesse. Le sample est joué et le son est dirigé vers la sortie audio.

Enregistrement d'une piste de bruits de pas personnalisés

Nous avons vu que le fichier audio était découpé en 226 samples, et que nous avions calibré notre valeur finale de vitesse au maximum à 226. Enfin, nous avons vu que plus la vitesse était élevée, plus le sample choisi était situé loin dans le fichier. Les samples étaient donc déjà classés du plus faible au plus fort.

Malgré de nombreuses recherches, nous n'avons pas trouvé de prise de son de ce type, présentant des sons du plus faible au plus fort. Il nous a donc fallu enregistrer ce fichier, dans le plateau son. Nous avons effectué une prise de sons de bruits de pas sur du bois, du plus calme au plus brutal.

Un problème particulier : la latence

Nous avons peu évoqué notre utilisation de Processing dans ce compte rendu. La première version du projet se basait sur une partie d'acquisition des données (via Processing et le code "KinOSC"), et sur une partie traitement du signal (via Max/MSP, qui comprenait les mêmes modules, sans le tout premier, "dp.kinect").

A la fin d'une présentation pour M. Viéville, ce dernier nous a suggéré d'étudier la latence du programme, depuis le mouvement jusqu'à la sortie audio. Les résultats, assez mauvais, nous ont convaincus de chercher une autre solution. C'est pourquoi nous utilisons désormais *dp.kinect*.

Nous avons récemment reproduis les tests de latence avec ce nouveau système. Les résultats sont meilleurs, quoique toujours un peu limite. Nous obtenons ainsi 8 images de latence, en 50 images par seconde. Convertit en temps, nous arrivons à environ 0.16 secondes de décalage.

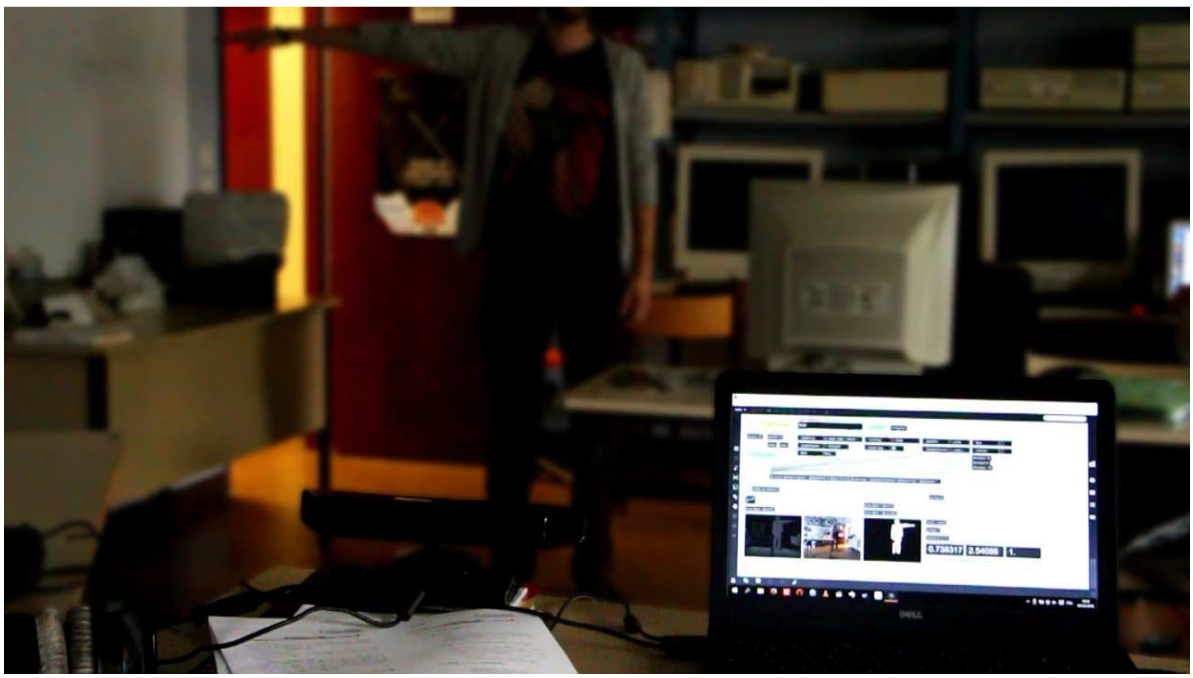


Image issu du test de latence

VI - Conclusion

Résumé

Au cours de ce Rapport, nous avons tout d'abord mené des Recherches. Nous avons appris que les bruiteurs insufflaient une intention dans leur bandes son, en rejouant la scène. Nous avons également appris que les bandes son produites par des Monteurs son manquaient de cette même intention. Nous nous sommes donc lancé dans la réalisation d'un outil permettant aux monteurs son de réaliser des bandes son enrichies d'une intention.

Après avoir passé en revue plusieurs technologies, nous avons choisi MaxMSP comme environnement de développement, nous permettant de profiter des ressources libres partagées par l'IRCAM.

Nous avons enfin mis au point un prototype de logiciel, proposant au Monteur Son de rejouer une scène (à la manière d'un bruiteur) devant un capteur (une kinect). Le logiciel sélectionne automatiquement le bruitage dont l'intention est la plus cohérente avec le jeu.

Pour des raisons de manque de temps, nous nous sommes volontairement restreints à la génération de bruits de pas. De plus, nous avons choisi comme seule variable mesurée la vitesse du mouvement. Nous pouvions ainsi proposer une échelle d'intentions allant de "calme" jusqu'à "brutal".

Ressenti

Nous sommes très contents du rendu de ce projet. Nous n'avons fait qu'écorcher la surface de la thématique mais ce travail nous a beaucoup plus, tant dans la Recherche que dans le Développement. En terme de compétences pures, nous sommes désormais beaucoup plus à l'aise avec la gestion de projet, et avec le développement informatique.

Par rapport au Cahier des Charges

Avant de nous lancer dans la réalisation du Cahier des Charges, nous avions beaucoup réduit nos objectifs. Nous nous étions fixé un but raisonnable, compte tenu du temps qui nous était imparti. C'est grâce à cette logique que nous avons pu mener notre projet à bien. Cependant, nous sommes forcé de reconnaître que le Cahier des Charges n'était pas suffisamment précis quand nous l'avons rédigé. Nous avons donc pris conscience de l'ampleur de certaines tâches bien plus tard, ce qui nous forçait à revoir régulièrement notre planning.

Points négatifs ?

Nous nous devons cependant de rester critiques envers le projet. Notre solution actuelle (un patch Max/MSP relié en ReWire à une station de travail audionumérique) nous paraît un peu trop lourde. De plus, Max/MSP est un logiciel payant et nos résultats en terme de latence ne sont pas exempts de défauts. Nous pensons donc faire évoluer le projet vers une autre plateforme.

Et après ?

Au long terme, nous voudrions au minimum passer l'intégralité du patch sous PureData, afin de pouvoir utiliser un logiciel Open Source. Dans le meilleur des cas, nous aimerions développer un logiciel en C++. Cela nous offrirait de bien meilleures performances, et un contrôle total sur notre projet, jusqu'aux moindres détails de chaque fonction.

En ce qui concerne le cœur du logiciel, nous voudrions continuer de l'améliorer. Nous pourrions travailler à une meilleure détection des mouvements, à rajouter de nouvelles caractéristiques de détection,... ou encore rajouter de nouveaux mouvements (des bruits de présences de vêtements générés en fonction des mouvements du corps tout entier ? Des ouvertures de portes ?).

Ce projet serait complet s'il permettait de piocher dans n'importe quelle banque de son, ce qui n'est pas encore le cas à l'heure actuelle. Nous aimerions développer un logiciel capable de scanner une banque de sons et de classer cette grande banque de samples selon des critères précis, qui seraient accessibles par la capture de mouvement.

Enfin, ce projet pourrait être utile en Réalité Virtuelle (afin de générer des bruitages en fonction des mouvements du joueur) ou en Motion Capture pour le Cinéma (afin d'immerger l'acteur dans son rôle en lui faisant entendre des sons relatifs au personnage qu'il incarne).

Nous avons dernièrement renommé le projet "LAMMA". LAMMA est un acronyme pour "Listening Adapter for Motion-controlled Mindful Audio".

Bibliographies

Andy Farnell, *Designing Sound*, MIT Press, ISBN: 9780262014410, 2010

Sabin Paul, *La Vraisemblance du Bruitage*, Ecole National Supérieur Louis Lumière, France, 2014.

Sicard Arnaud, *Etude perceptive des interrelations morpho dynamiques gestes-images-sons*, LAM, France, 2006.

Frisson Christan, *Designing interaction for browsing media collections*, Université de Mons, Belgique, 2015.

Stefano Trento et de Götzen Amalia, *Foley Sounds vs Real Sounds*, Aalborg University, Denmark, 2011

Curtis Roads, *L'audionumérique Musique et informatique*, Dunod, France, 2007

Moulin Elodie, *Un regard pour exister, un corps pour s'exprimer, la psychomotricité pour tout allier*, Les Hôpitaux Universitaires Pitié Salpêtrière, France, 2014.

Du mouvement au geste, Les Hôpitaux Universitaires Pitié Salpêtrière, France, 2014.

Annexes

Annexe 1 - Protocole de questions

- Comment vous appelez vous ?
- Quel est votre métier ? Dans quelle entreprise ?
- Je fais une étude sur la vraisemblance des bruitages au cinéma. Qu'est ce qui, pour vous, sont les caractéristiques qui rendent une bande son réaliste ? Cohérente ? Ou au contraire, quels sont les défauts à éviter ?
- Je travaille plus particulièrement sur les différences entre une bande son produite par un bruiteur et une bande son produite par un monteur son, uniquement à l'aide de banques de son. Selon vous, quelles sont ces différences ?
- Un monteur son est "techniquement parfait" alors qu'un bruiteur peut produire des bruitages un peu moins carrés à l'image. Cependant, le résultat paraît plus adapté. Qu'en pensez vous ?
    - ⇨ vous avez un exemple de film qui aurait été réalisé une fois par un bruiteur et une fois par un monteur son ? Pour pouvoir comparer les deux
    - ⇨ ou bien, d'expériences qui ont été faites à ce sujet, d'études....
- Au cours de mes recherches, j'ai cru comprendre qu'un bruiteur arrivait à donner une intention au bruitages qu'il produisait, cela peut il expliquer en partie cette différence ?
- Si on devait étudier le travail du bruiteur, qu'est ce qui, selon vous, est la caractéristique qui porte l'intention ?
- En terme de vraisemblance des bruitages, nous avons déterminé après recherches que l'on pouvait jouer sur 4 critères : la matière, l'acoustique, le synchronisme, le geste. Nous pensons que c'est surtout le critère de geste qui porte l'intention et qui fait vraiment la différence entre des bruitages réalisés par un bruiteur et des bruitages tirés d'une banque de son.
- Admettons ceci : un bruiteur veut créer des bruitages à l'image, il réalise une série de mouvements (volontaires et involontaires) qui traduisent d'une intention, d'un sentiment. Ce serait cette suite de gestes qui ferait la différence.

Annexe 2 - Entretien avec Paul Sabin

Prise de Notes du 09/12/15

le plus important c'est l'intention

un monteur son va faire attention au poids du personnage, mais en plus de ça, un bruiteur va prendre en compte la fatigue du personnage. Il s'investit corporellement dans le personnage, Il incarne le personnage.

L'intention ne peut être que juste car il revit la scène. Ce qui est difficile chez le bruiteur, c'est que c'est spontané, on arrive pas trop à l'expliquer.

De plus, les bruiteurs ont une homogénéité absente du montage son => ils utilisent les mêmes objets et ont une intention du début à la fin

Un logiciel qui peut modifier des paramètres du son pour le faire coller à une autre intention peut être intéressant. Par rapport à mon mémoire, il faudrait refaire des tests perceptifs et trouver plus de critères.

ATTENTION car ton sujet est un sujet qui parle autant de l'image que du son. L'image guide tellement l'écoute qu'elle permet de rendre cohérent l'écoute jusqu'à une certaine limite. Pour ta recherche, il faudrait chercher des critères PAR RAPPORT A l'image (au plan) et non pas juste entre sons seuls. Prendre des exemples qui sont entre le bruitage et le FX (genre un train qui passe).

Pour moi ce qui serait intéressant, ce serait de poser des noms sur des intentions, et de faire des traitements en fonction de ça. Ce serait intéressant de développer un plugin qui est capable de texturiser des sons => pour moi le geste => concerne la texture qui rentre en jeu

Pour les psychomotriciens, le geste porte l'intention mais pour nous bruiteurs le geste est dédié à la création d'un son, c'est pas la même chose que dans la vraie vie => c'est pas forcément de faire un pas plus fort pour donner l'impression du poids, peut être poser le pieds plus longtemps etc.

Faire une étude sur le corporel => qu'est ce que le mouvement et le corps changent dans le son

Ta recherche permet de faire abstraction de l'image, c'est malin !

c'est aussi une somme de sons qui donnent une intention => ça aussi qui donne une homogénéité

Paramètre hyper important que je n'ai pas trop pris en compte : la variabilité

Le bruiteur est capable d'avoir une espèce d'aléatoire qui rend le geste plus fort par son dynamisme

Tu fais trois bruits de pas sur place puis trois sur un seul pied ça crée pas la même chose

Annexe 3 - Entretien avec Guillaume le Guen

Prise de Notes du 29/10/15

La principale différence, c'est que le bruiteur fait une espèce "d'analyse émotionnelle", "d'analyse d'intention" de l'image, avant de produire du son.

Ci on compare le travail d'un bruiteur et celui d'un monteur son, le travail du monteur son va être techniquement "parfait", c'est à dire calé parfaitement à l'image près dans le temps, alors que celui du bruiteur ne le sera pas tout à fait, ce sera beaucoup plus "flottant". Cependant, le travail du bruiteur nous paraîtra beaucoup plus naturel que l'autre, car il recréera l'intention.

Le cerveau du spectateur n'est pas gêné par le fait que le travail du bruiteur ne soit pas calé "parfaitement", on s'habitue à un certain "delta" de temps et on a une sensation de synchro alors que pas du tout

Intérêt du travail du bruiteur, c'est aussi la diversité des sons car on peut claquer une porte de différentes façons, l'enregistrer de différentes façons...

Il y a un rapport très particulier à la matière (i.e. la matière utilisée, le verra, les graviers, les chaussures...) et au microphone (utilisation de tel ou tel défaut des micros, de l'effet de proximité pour produire un "wooof" à l'ouverture d'une porte, etc.)

Pour prolonger l'idée d'utilisation des micros et de leurs défauts : ils se servent des défauts pour recréer **une sensation** avant de recréer un son. Pour reprendre l'exemple de la porte qui fait un appel d'air, ils vont se servir de l'effet de proximité pour donner l'impression que la porte déplace beaucoup d'air => sensation de mouvement, sensation d'y être...

Annexe 4 - Entretien avec Olivier Chatron

Prise de Notes du 09/12/15

Le bruiteur est toujours nouveau, c'est toujours unique

Les banques de son c'est toujours les mêmes. Les banques il faut le travailler pour qu'il devienne unique

Le travail du bruiteur est réfléchi en terme de psycho acoustique. Comment le spectateur va percevoir la chose ?

La banque de son est froide et elle ne pense pas (est ce qu'elle a les petits détails qui peuvent manquer)

créativité intrinsèque => il peut jouer sur le microphone, l'éloigner, etc.

bande son réaliste => bande son qui te donne les sensations espérées

Voit le son en 3 ou 4 dimensions => dimension du niveau (ou énergie) (énergie du son par rapport à lui même ET AUSSI l'énergie du son par rapport aux autres)

=> dimension de la couleur (plus grave ou plus aigu) (idem, par rapport à soit aux autres)

=> dimension du temps (son rythmé ou non) (idem + placement du son)

=> point d'ouïe

Pour lui, ce n'est pas possible de modifier l'intention d'un son. Il prend pour exemple un violoniste qui joue un morceau fortissimo. Si on enregistre ce morceau et qu'on le repasse, même à faible volume, on entendra quand même que l'intention était de jouer fortissimo. Certaines informations sont présentes dans le son et on ne peut pas les enlever/modifier.