

Compte rendu d'avancement

Projet CONDUCT

Rappel du contexte

Ce projet s'inscrit dans les axes de recherche l'équipe Expression (<https://www-expression.irisa.fr/fr/>) qui concernent l'analyse et la synthèse des données expressives produites par l'humain (geste, son, image). Plus précisément, on s'intéresse ici au contrôle gestuel de processus de simulation sonore. Des travaux réalisés préalablement dans l'équipe [1], [2] et dans le cadre de projets de master 1 (SONIC 1 et SONIC 2) ont permis de d'analyser et de synthétiser des gestes instrumentaux (percussion, violon) et des gestes de chef d'orchestre.

[1] A. Bouénard, S. Gibet, M. Wanderley (2012). Hybrid Inverse Motion Control for Virtual Characters Interacting with Sound Synthesis - Application to percussion motion. The Visual Computer 28(4): 357-370, 2012 [2] Lei Chen, Sylvie Gibet. CONDUCT: an expressive conducting gesture corpus for sound control, LREC 2018, Japon, May 2018

Rappel de la problématique

L'objectif final du projet est de contrôler et synthétiser des sons au moyen de la reconnaissance et du tracking de gestes en temps réel. On s'intéresse à un ensemble de gestes d'interprétation d'une mélodie donnée qui permettent de moduler de manière expressive les sons constituants de cette mélodie (par exemple variation de l'intensité, de la radiosité, de l'attaque, arrêt, variation du tempo, etc.). Les gestes seront d'abord capturés au moyen du système Leap Motion. La reconnaissance de gestes statiques (postures clés prédéfinies) sera réalisée par un algorithme d'apprentissage simple (K-NN ou SVM). La détection des gestes dynamiques sera réalisée un tracking simple ou une reconnaissance temps réel du tracé du geste. La synthèse sera réalisée grâce au framework Faust et au système de synthèse Pure Data. Ce projet sera poursuivi en M2 dans le cadre d'une application de réalité virtuelle intégrant la visualisation interactive d'objets sonores (exemple Audica).

I) Prise en main des outils logiciels et du matériel d'interaction

1) Environnement de programmation Unity 3D

Dans la partie reconnaissance des gestes, nous avons utilisé l'environnement de programmation Unity 3D. Unity 3D est à la base un moteur de jeu multiplateforme, il n'est cependant plus utilisé seulement dans le domaine du jeux vidéo, comme en témoigne notre projet. Unity est un outil reconnu, et utilisé dans beaucoup de projets qu'ils soient indépendants ou non.

La programmation s'effectue en C#, langage objet proche de Java de par sa syntaxe et ses concepts. Il a été créé et commercialisé par Microsoft.

Un des atouts d'Unity est son système d'Asset. Les Assets peuvent être des textures, objets 3D, classes d'objets, etc... Que l'on va pouvoir utiliser dans notre programme en les téléchargeant depuis l'Assets Store.

La communication avec les dispositifs d'entrées (Leap Motion, Manette de jeu...) et de sorties (Casque de Réalité Virtuelle, Réalité Augmenté) se fait grâce à ces Assets, fournies par les entreprises qui développent ces dispositifs.

2) Synthèse de son grâce à l'environnement de programmation PureData

PureData est un logiciel de programmation graphique utilisé pour le traitement des signaux sonores. Il se base sur le principe de la programmation modulaire. On peut donc utiliser des objets PureData tel que des nombres, des messages, et des interrupteurs afin de modifier les signaux et informations reçues en entrée ou générées dans PureData.

Il est aussi possible de télécharger des "externals", qui sont des bibliothèques de programme PureData prêtes à l'emploi.

PureData étant utilisé dans le domaine du traitement de signaux sonores, il est possible de traiter et de générer des signaux continus. Nous pouvons donc additionner 2 signaux entre eux (pour générer un accord, par exemple), modifier l'amplitude ou la fréquence d'un signal.

Il est aussi possible de gérer des données numériques qui pourront être utilisés pour le traitement de signaux (comme pour changer le volume ou le tempo). Ces données peuvent provenir du programme en cours d'exécution, d'un autre objet PureData, ou même d'autres programmes (en utilisant par exemple un port UDP).

Une partie importante de notre projet est de prendre des informations provenant d'une entrée ou d'un fichier midi et de les faire jouer par des instruments synthétiques s'approchant de leur modèle réel. Ces instruments ont été générés avec Faust.

Faust est un langage de programmation fonctionnel axé sur la synthèse audio. Il permet notamment la création de synthétiseurs et d'instruments. Pour notre projet, nous avons décidé de prendre des modèles d'instruments générés avec Faust qui ont été par la suite exportés en tant que programme PureData. Cela nous permet de pouvoir manipuler les différents paramètres influant le son généré par ces instruments.

3) Protocole de communication Open Sound Control

Open Sound Control est un protocole de communication utilisé dans la communication entre ordinateurs, synthétiseurs et périphériques d'entrées. C'est un protocole léger utilisant les technologies réseaux modernes afin de transmettre des informations en temps réel, ce dont ont vraiment besoin les applications de traitement du signal.

4) Le code MIDI

Le protocole MIDI est un protocole utilisé pour l'échange d'informations musicales entre instruments. En d'autres termes, il permet de traduire du son en texte (intensité, longueur, hauteur de la note). Cartes son, synthétiseurs, etc... sont capables d'interpréter et de jouer un fichier MIDI.

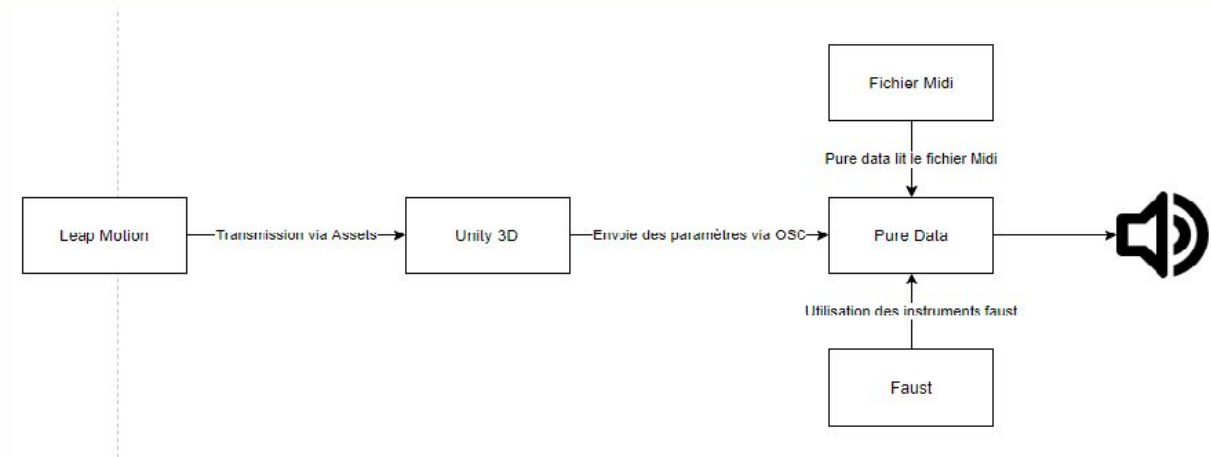
Ce protocole a l'avantage d'être très utilisé et facilement modulable. Un fichier MIDI est très souvent de petite taille (10 KB). Il est composé en 3 parties : le connecteur physique (qui fait le lien entre 2 appareils), le format des données (le coeur du fichier), et le format de stockage des données..

II) Travail réalisé

Dans ce projet, deux parties se distinguent assez rapidement, une partie concernant la reconnaissance de gestes en exploitant le système Leap Motion et l'environnement de programmation Unity 3D (Alexy Duhamel et Maxime Hamon ont travaillé sur cette partie) et

l'autre partie concerne la synthèse de son en utilisant Faust, Pure Data et un clavier d'entrée qui produit des signaux au format MIDI (Charles Cogoluègnes et Romann Yvinec ont travaillé sur cette partie).

Voici un schéma montrant les deux sous applications distinctes du projet :








1) Reconnaissance de configurations manuelles : Unity et Leap Motion

La reconnaissance de geste nécessite un dispositif de capture, nous avons choisi le Leap Motion. Ce périphérique est composé de deux caméras infrarouges ainsi que de trois LED infrarouges. En comparant les deux images fournies par les caméras, le Leap Motion peut déterminer la position des objets qu'il observe, ici les mains.

Dans cette application, les gestes statiques de la main gauche permettent de choisir différents modes, tandis que la main droite va servir à l'interprétation de ce mode. Par exemple, fermer la main gauche permet de choisir le mode 'volume' et bouger la main droite de haut en bas modifie le volume.

Actuellement, l'application de reconnaissance de configurations manuelles est capable de reconnaître 5 gestes différents : (compte rendu 07/02/20)

Main ouverte	Poing fermé	Un doigt	Deux doigts	Lettre J
				

Pour reconnaître les différentes configurations manuelles nous utilisons un algorithme des K plus proches voisins (que nous abrégeons par "k-NN"). Les données de l'algorithme k-NN sont composés de points en 5 dimensions, chaque coordonnée d'un point étant la distance entre le bout du doigt et la paume de la main. Quand on veut déterminer une configuration manuelle, on envoie à l'algorithme un point en 5 dimensions. L'algorithme va trouver les k points les plus proches dans l'espace 5D pour déterminer sa classe (la configuration manuelle).

Pour mettre en oeuvre cette algorithme, nous avons besoin d'avoir accès aux positions des différentes parties de la main (les différentes phalanges, la paume...) nous utilisons le SDK fournies par Leap Motion pour cela.



Grâce à cet asset, nous avons accès à chaque os de chaque doigt mais aussi, pour chaque partie de la main, aux caractéristiques qui la définissent, tel que sa position, sa rotation... etc.

Nous n'avons pour l'instant pas fait de tests pour mesurer la fiabilité de l'algorithme, on peut cependant empiriquement déduire que l'application est assez fiable. Il faut également prendre en compte les limitations techniques du Leap Motion. Par exemple nous ne pouvons pas prendre des distances trop grandes ou trop faibles avec le dispositif, sous peine de perdre la reconnaissance des mains.

Nous avons également développé une interface rudimentaire permettant à l'utilisateur d'avoir un feedback lors de la calibration des distances au démarrage de l'application et aussi lors de l'exécution des gestes.

2) Synthèse sonore sous PureData

Notre environnement PureData s'organise de la manière suivante :

- Un fichier Main.pd correspondant au fichier principale de notre partie du projet. Pour commencer, nous lisons en boucle un fichier MIDI et nous extrayons les informations utiles à nos instruments virtuels (la fréquence, la vitesse, etc). Ces paramètres sont ensuite directement envoyés à nos instruments, qui sont des objets PureData que nous détaillerons par la suite. Ces derniers se chargent de jouer la mélodie.

Dans un même temps, nous écoutons sur le port 5000, les paramètres à modifier envoyés par l'équipe Unity sous le protocole Open Sound Control. Nous parons les données avec l'objet OSCParser et nous envoyons les paramètres directement aux instruments. Nous arrivons donc à modifier en temps réel le volume, le tempo (possible grâce à TempoFixer), l'attaque et le vibrato (compte-rendu du 27/01/2020).

Bien entendu, nous pouvons modifier autant de paramètres que nécessaire.

- Le dossier MIDI_files contient toutes les mélodies sous format MIDI.
- Le dossier instruments contient d'une part les instruments sous forme d'objets PureData et d'autre part les instruments sous forme de fichier Faust (.dsp).

Les instruments Faust sont donc convertis en objet PureData. Grâce à la librairie PureData faustgen, cette conversion est transparente. Il suffit simplement d'envoyer les paramètres sous forme de message avec le nom de ce dernier (ex : pression 0.8). Les instruments disponibles sont le violon, la guitare, la clarinette et le vocal. Prenons par exemple le violon : une fois l'objet construit, il suffit d'envoyer les paramètres comme la

fréquence (qui correspond à la note), le sustain (qui correspond à la longueur), etc. L'objet envoie en sortie un signal que l'on envoie directement à un DAC (Digital Analog Converter).

- Le dossier util contient les objets PureData cités précédemment (TempoFixer et OSCParser), ainsi que DataSender qui permet de tester l'envoi des paramètres sur le port 5000 via le protocole Open Sound Control.
- Le dossier MIDI_misc contient l'objet MIDIFileWriter qui permet de jouer une mélodie et de l'enregistrer sous format MIDI. Enfin, il contient le fichier MIDIHaken qui permet de jouer sur le clavier Haken dans PureData.

3) Mise en commun de l'ensemble

L'application de reconnaissance de gestes va envoyer selon les gestes (dynamiques ou statiques) reconnus des informations à l'application de synthèse sonore via le protocole de transmission de données Open Sound Control. L'ensemble fonctionne plutôt bien pour l'instant, même si, forcément, il y a toujours des fonctionnalités à améliorer et à ajouter. (compte rendu 27/01/20)

III. Difficultés rencontrées

La plus grande difficulté que nous avons rencontrée dans ce projet est que toutes les technologies, langages de programmations, environnements de programmations et algorithmes sont nouveaux pour nous. Nous avons donc dû apprendre à utiliser Unity 3D, Pure data et Faust. Pour l'application de reconnaissance des gestes, il a fallu apprendre les algorithmes de reconnaissances. Pour l'application de synthèse sonore, nous avons dû acquérir des connaissances en traitement du signal et modélisation physique d'instruments. Les dispositifs de captures comme le Leap Motion et le clavier Haken sont également nouveaux pour nous.

1) Reconnaissance de configurations manuelles : Unity et Leap Motion

Pour la reconnaissance de gestes, la première difficulté que nous avons rencontrée était la connexion et l'utilisation de Leap Motion dans Unity 3D. Nous devons nous aider du projet de l'année précédente (SONIC 2) pour cette partie, cependant, le projet manquait de documentation. Pour résoudre ce problème, nous avons donc fait appel à l'aide de Sylvie Gibet

(tutrice du projet), Axel Jacomme (Ingénieur SSI Vannes) et Mansour Tchehegnon (Étudiant en Master 2 ayant participé au projet SONIC 2).

Une autre difficulté a été de choisir les bons paramètres à prendre en compte pour avoir une reconnaissance de gestes, fiable. Nous avons choisi de prendre la distance entre la paume de la main et chaque bout de doigt. Si un seul doigt bouge dans une position par rapport à un autre, les points dans l'espace 5 dimensions ont tous une position radicalement différente, ce qui rend les gestes bien dissociables.

2) Synthèse sonore sous Pure data

La première difficulté rencontrée était liée à l'utilisation de PureData. Ceci est dû à deux facteurs. Le premier est que ce programme se base sur un paradigme de programmation qui nous était inconnu : La programmation graphique. Étant habitué à la programmation textuelle, il a fallu nous familiariser rapidement avec ce nouveau paradigme. Le second facteur est que nous n'avions aucune expérience dans le domaine de la synthèse sonore. C'est un domaine complexe nécessitant beaucoup de temps pour être maîtrisé.

Le second obstacle majeur a été la lecture de fichiers midi. PureData ne permettant pas directement de lire de tels fichiers, nous devions trouver une librairie nous permettant de faire une telle chose. C'est dans cette optique que nous avons décidé d'utiliser la librairie Cyclone. Elle contient plusieurs objets utiles pour notre projet. Le premier de ces objets est "seq" permettant à la fois la lecture et l'écriture de fichier midi. Le second est "midiparse", qui permet de séparer les différents composants d'une note midi. Bien que ces objets soient faciles à utiliser (si on sait s'en servir), il fût compliqué de les découvrir et de comprendre leurs fonctionnements respectifs, étant donné que Cyclone ne dispose que de peu de documentation technique sur internet. Heureusement, cette librairie inclut des programmes d'exemples. Cela nous a permis de comprendre le fonctionnement de "seq" et de "midiparse".

La troisième difficulté concerne la gestion du tempo en temps réel. L'un des avantages de l'objet "seq" est qu'il permet de choisir le tempo au lancement de la lecture du fichier midi. Cependant, pour le modifier, il était nécessaire de relancer la lecture du fichier depuis le début, ce qui allait à l'encontre de notre objectif. Ce problème a facilement été réglé en envoyant un message différent à "seq".

La dernière difficulté rencontrée a été l'utilisation du clavier Haken (compte-rendu du 05/03/2020). Son installation et sa calibration s'est révélée beaucoup trop complexe. Nous avons

décidé de ne pas l'utiliser étant donné qu'il existe d'autres alternatives pour pouvoir envoyer des notes à PureData, comme la lecture d'un fichier midi, ou l'utilisation d'un clavier midi physique ou virtuelle.

IV. Travail restant à faire

1) Reconnaissance des gestes

Pour cette partie du projet, il nous reste la reconnaissance de gestes dynamiques à mettre en place.. En effet, quand pour reconnaître un geste statique nous devons "simplement" détecter un geste à une frame donnée, pour un geste dynamique, nous devons reconnaître le geste dans un ensemble de frames, dont on ne connaît pas le nombre d'avance. Selon la vitesse d'exécution du geste, le nombre de frames sur lesquelles nous allons travailler sera différent. L'autre difficulté est qu'il faut détecter le début et la fin du geste. C'est pour ces deux raisons que nous allons utiliser des algorithmes à fenêtre de temps glissante tel que l'algorithme DTW (Dynamic time warping). L'algorithme DTW va prendre une fenêtre de temps "glissante" plutôt que bloc de temps par bloc de temps.

Nous aimerions aussi avoir une interface graphique plus complète, avec un feedback plus élaboré, pour pouvoir mieux visualiser ce qu'il se passe lors de détection de gestes dynamiques ou statiques.

2) Synthèse sonore

Comme dit précédemment, nous faisons jouer notre mélodie par des instruments synthétiques (compte-rendu du 05/03/2020). Ces instruments se basant sur leur homologues réels, ils disposent de nombreux paramètres influant sur le son émis. Ces paramètres peuvent drastiquement varier entre les différents instruments. Par exemple, le violon prend en compte la position de l'archet sur les cordes alors que le son d'une clarinette varie en fonction de la rigidité de son roseau. De plus, les différents paramètres peuvent être liés les uns les autres. Un violoniste placera son archet et ses doigts différemment selon la note qu'il souhaite jouer. Pour produire un son harmonieux, il nous sera donc nécessaire de trouver les liens entre ces différents paramètres physiques (compte-rendu du 19/03/2020).

V. Planning

