Projet MDMA - Rapport L2

Table des matières

Partie 1

Introduction

Dès le commencement, notre projet était de réaliser un logiciel permettant de faire de la musique sans rien toucher, en contrôlant chaque instrument, chaque paramètre par des mouvements. Sous la forme d'un contrôleur MIDI virtuel, nous mettons donc au point le logiciel MDMA, qui agit sur tout séquenceur ou synthétiseur en décryptant les gestes de nos mains.

Il existe de nombreux contrôleurs musicaux fonctionnant par signal MIDI. Certains sont de simples claviers de piano, d'autres proposent des potentiomètres, des pad, des matrices de clips, et tous sont configurables par l'utilisateur. Ce domaine est en plein essor, essentiellement grâce à la démocratisation de la MAO et de la popularité actuelle des musiques électroniques. Le point commun entre ces contrôleurs est leur utilisation des signaux MIDI pour coder les données musicales et communiquer avec un logiciel. Notre but est de fournir un contrôleur tout aussi ergonomique et polyvalent basé simplement sur de la reconnaissance de mouvements. Une caméra filme l'utilisateur en continu, et les mains de celui-ci, selon leur position, leur mouvement, et selon si elles sont ouvertes ou fermées, agissent en direct sur sa musique. On peut ainsi jouer des notes sur un synthétiseur virtuel, moduler la fréquence d'un filtre ou le tempo, ou même lancer des boucles audio d'un simple geste.

Nous avons pris le parti de demander à l'utilisateur un matériel minimal. Une simple webcam suffit à fournir l'image que notre logiciel analyse pour situer les mains. Dans un souci d'efficacité et de précision, nous envisageons également de proposer une version utilisant une *Kinect*. Ensuite, il s'agit de traduire ces mouvements en commandes MIDI et de pouvoir les exporter vers un autre logiciel.

Le travail a été divisé en plusieurs groupes, et chaque équipe a pu se confronter aux difficultés et examiner les différentes possibilités offertes. Voici un compte rendu de ce travail.

Partie 2

Spécifications

MDMA est multi-plateforme : des versions Linux et Windows sont développées, une version Mac OS est envisagée.

Le logiciel possède deux composants principaux : un analyseur de flux vidéo qui détecte certains types de mouvements et produit en sortie des signaux MIDI, et un éditeur de configurations.

Le flux vidéo, provenant soit d'une webcam soit d'un périphérique *Kinect*, est analysé en temps réel avec une latence non perceptible par l'être humain.

La position, la vitesse, l'accélération ainsi que l'état ouverte/fermée de chaque main sont déterminés.

Les événements sont définis à partir de zones et de segments. Ces derniers correspondent à différentes parties de l'image – une zone est un rectangle, un segment est un segment de droite ; notons qu'ils sont fixes - et c'est l'interaction des mains avec eux qui crée les événements. Ceux-ci sont de différents types :

- pour les segments : le franchissement
- pour les zones : l'ouverture (de la main), la fermeture (de la main), le déplacement pour chaque état (i.e. ouverte/fermée) et la frappe

Chacun de ces événements, suivant la configuration utilisée, peut déclencher l'émission d'un signal MIDI ce qui permet de simuler (voir illustration) :

- des fadders (potentiomètres rectilignes) et des pads XY, par une zone sensible au déplacement de la main dans l'état ouverte et/ou fermée
- de boutons, par un segment sensible au franchissement ou par une zone sensible à l'ouverture et/ou la fermeture (de la main) ou la frappe

Une configuration est une liste d'onglets, qui sont eux mêmes une liste de zones et de segments associés à un événement ainsi qu'à un signal MIDI ou un changement d'onglet. Cela correspond au fait que durant l'analyse vidéo, après avoir chargé une configuration, l'utilisateur peut naviguer entre les onglets et déclencher l'émission de signaux MIDI en provoquant les événements associés aux zones et segments de l'onglet en cours.

Concernant l'émission des signaux MIDI, MDMA peut au choix créer un port de sortie MIDI virtuel sur lequel peuvent écouter d'autres logiciels et en faire sortir ses messages, ou bien diriger ces derniers vers un port d'entrée MIDI déjà existant. Le logiciel MDMA se présente à l'utilisateur sous la forme d'une unique fenêtre affichant un menu général (proposant de charger et de sauvegarder une configuration), une boite affichant des informations sur la configuration utilisée et enfin un flux vidéo de contrôle représentant l'image acquise par la caméra sur laquelle se surimpression les différents zones et segments de l'onglet en cours

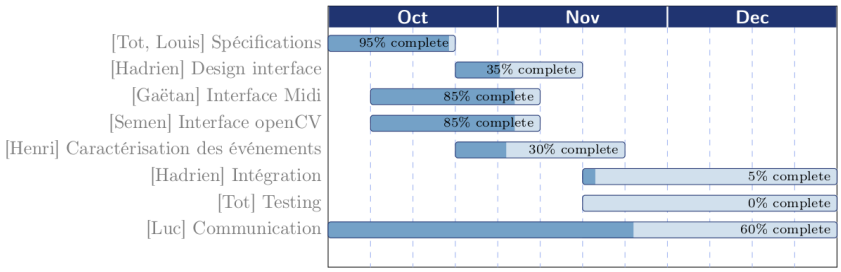

FIGURE 1 – État d'avancement du projet au 5 novembre 2012

dans la configuration utilisée. Un code de couleurs et d'opacités sur les zones et les segments permet de mettre en évidence la détection des événements.

Ce *workpackage* est en veille : le comportement du logiciel MDMA est clairement défini vis-à-vis de l'utilisateur, cependant il n'est pas impossible que l'évolution des différents autres *workpackage* nous amènes à repenser certaines fonctionnalités.

Partie 3

Design Interface

3.1 Participants

- Hadrien Croubois (responsable d'équipe) ;
- Luc Rocher

3.2 Objectifs

Le *workpackage* "Design Interface" a pour objectif de mettre au point et d'implémenter l'interface ainsi que de penser a l'interconnexion de l'interface avec l'ensemble des composants du logiciel. Il s'occupe en plus de la gestion des configurations décrites par l'utilisateur, qu'il s'agisse de leurs paramétrages via les différentes fenêtres ou de la partie chargement/enregistrement. La connaissance de l'état d'avancement des différents *workpackages* permet de définir efficacement les formats de configuration ainsi que de penser les flux de données nécessaires a l'affichage au cœur de la structure du programme.

Le *workpackage* "Design Interface" anticipe a ce titre une part de l'intégration.

3.3 Avancement

L'interface en elle même est écrite a l'aide du designer de fenêtre de *QtCreator*[1].

La fenêtre principale permet, en plus de la visualisation de l'image acquis par la webcam de charger et sauvegarder différentes configuration ainsi que d'avoir

1. QtCreator - `http://doc.qt.digia.com/qtcreator/index.html`

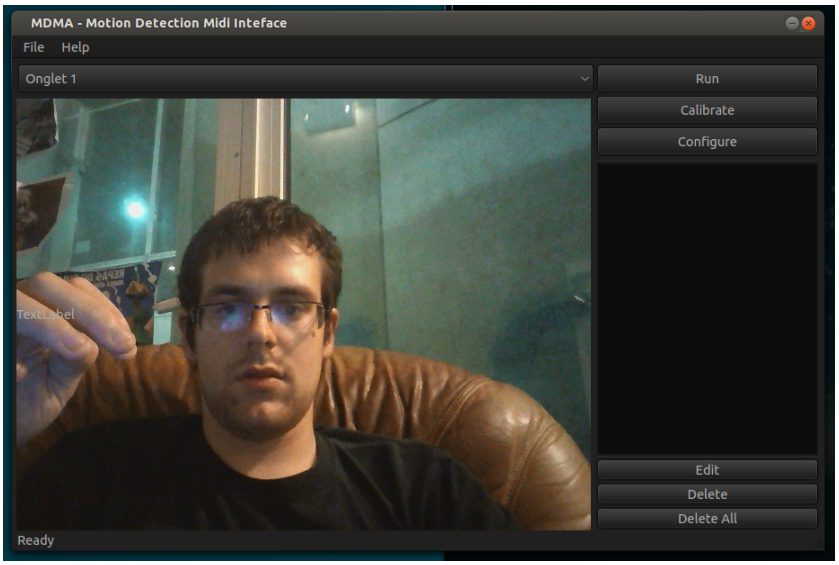

FIGURE 2 – La fenêtre principale de l'interface

un résumé des différentes zones définies. Ces zones peuvent être crées en cliquant sur la vidéo, modifiées et supprimées.

L'ouverture d'une nouvelle configuration ou la fermeture du programme alors que la configuration actuelle a été récemment modifiée demande a l'utilisateur de sauvegarder la configuration actuelle.

Un bouton "Calibrate" lancera une procédure de calibration, de même que le bouton "Configure" ouvre une fenêtre permettant de choisir différentes options (caméra si plusieurs sont disponible, port midi ...).

Un bouton "Run" lance le programme. Une fois lancé, la configuration n'est plus éditable (les différents boutons sont désactives). Seul le bouton "Run", renommé en "Stop", reste actif et permet d'arrêter le programme.

Partie 4

Interface MIDI

MDMA produit des messages MIDI qui ont pour but d'être interprétés par d'autres logiciels – typiquement des séquenceurs – comme des notes, des signaux lecture/stop ou tout autre modification de paramètre à la manière d'un contrôleur MIDI classique.

Pour gérer toutes les tâches concernant le protocole MIDI, nous avons décider d'utiliser la bibliothèque RtMidi. Toujours développée, cette bilbiothèque orientée programmation objet pour le C++ est multi-platforme (Linux, Windows et Mac OS). Nous exploitons essentiellement la classe RtMidiOut, qui gère l'envoie de messages MIDI en temps réel.

La première chose à faire est d'ouvrir un port MIDI virtuel pour y envoyer nos messages, ceux-ci pourront être récupérés par n'importe quel logiciel écoutant sur le port précédemment créé (figure 3). Malheureusement RtMidi ne permet pas une telle opération sous système Windows. Nous utilisons actuelle-

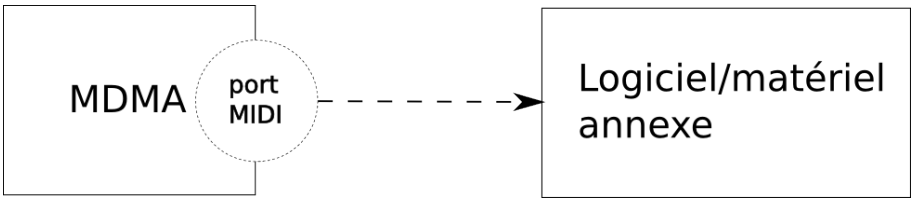

FIGURE 3 – Utilisation d'un port crée par MDMA

FIGURE 4 – Utilisation d'un port virtuel externe à MDMA

ment loopMIDI pour contourner ce problème : loopMIDI permet de créer deux ports MIDI, un en entrée et un en sortie, et fait transiter tout message reçu par le port d'entrée vers le port de sortie. MDMA n'a donc qu'à envoyer ses messages sur le port d'entrée, ils pourront alors être récupérés sur le port de sortie (figure 4).

Nous devrions à terme être capable de nous passer de loopMIDI : son développeur a créé un SDK permettant d'intégrer la création de ports MIDI virtuels directement dans un logiciel ; nous venons de le contacter et attendons sa réponse.

Une fois la manière de procéder déterminée (création ou non d'un port directement par MDMA), la gestion des messages est très simple. RtMidiOut possède une méthode openPort permettant de se connecter à un port de sortie, ainsi qu'une méthode sendMessage permettant d'envoyer un std : :vector<unsigned char> sur le port précédemment choisi.

L'envoie de messages est commandé par les mouvements de l'utilisateur, et s'effectue en bout de chaîne de la gestion des événements.

Le travail de ce *workpackages* est quasiment terminé : outre la création de port MIDI sous Windows, il ne s'agit essentiellement plus que d'intégrer l'envoie de message à la gestion des événements.

OpenCV Interface

Introduction

La tâche prise en charge initialement par les contributeurs du *workpackages* consistait en la création du module qui capturerait l'image de la camera, localiserait les mains sur cette image et transmettrait la totalité des informations : la position des mains, leur vitesse, leur accélération et la forme (fermée ou ouverte). La spécification initiale du système sous-entendait la possibilité que le système fonctionne sans utiliser de matériel onéreux.

Conditions de fonctionnement

Pendant le développement, il était entendu que la reconnaissance des mains se ferait sans avoir recours à des marqueurs et sans l'utilisation de matériel spécial (comme par exemple l'outil *Microsoft Kinect*). Après avoir considéré certaines contraintes pour assurer la reconnaissance des mains, l'équipe de ce *workpackage* a décidé d'utiliser des gants noirs. Les tests ont montré que la reconnaissance des mains avec les gants est stable sous les quelques conditions suivantes :

- Le fond ne contient pas d'objets foncés.
- Le vêtement d'utilisateur est notablement plus clair que les gants, qui doivent être très foncés (noirs).
- L'éclairage doit être suffisant.

Du point de vue logiciel, le module nécessite que la librairie OpenCV soit installée sur le système.

Comme extension du projet, nous envisageons de développer une version de notre logiciel fonctionnant avec une *Kinect*. La version ne sera qu'optionnelle tant qu'elle demandera des dépenses supplémentaires de part de l'utilisateur. Malgré ce défaut le développement de la version est raisonnable tant qu'il assure la reconnaissance grâce à l'utilisation de matériels prévus pour le tracking des objets.

Algorithme développé

Pour l'instant le résultat est l'algorithme développé et implémenté avec l'utilisation de la librairie OpenCV. Une implémentation développée pour fournir une plate-forme-indépendante.

Structure de l'algorithme

- Calibration du système. L'utilisateur place les mains dans les zones spéciales. Après cela le système fixe L' égal à l'éclairage du point le moins éclairé dehors de ces zones. $L = L' - \Delta$ devient le seuil critique de luminosité. Après cette étape la boucle principale du fonctionnement se déroule.
- Commencement de la boucle. Capture de l'image.
- Conversion RGB vers Niveau de Gris.
- Création du masque. Les 1 prennent la place de tous les pixels dont la luminosité est inférieur à L.
- Masquage des pixels de la zone couvrant le corps de l'utilisateur. Les pixels de cette zone sont fixés à 0.
- Érosion de l'image.
  Définition de l'érosion. L'érosion est l'une des deux opérations fondamentales du traitement d'image morphologique. Soit A une image binaire et B un élément de structure. L'érosion de A par B est donnée par : $A \bigodot B = \{z|(B)_z \subset A\}$.
  Pour l'élément structurant dans l'implémentation nous utilisons une ellipse 3×3.
  L'érosion est utilisée pour le filtrage du bruit, c'est à dire les petites particules blanches qui peuvent apparaître sur le masque par hasard. L'érosion est appliquée au masque deux fois.

0	1	0
1	1	1
0	1	0

TABLE 1 – Élément structurant : ellipse 3×3

- Dilatation de l'image.
  La dilatation a pour but de compenser l'utilisation de l'érosion et de simplifier les frontières des objets. La dilatation est définie comme $A \bigoplus B = \cup_{b \in B} A_b$. Pour la dilatation le même élément structurant que pour l'érosion dans l'étape précédente.
- La recherche des contours. La recherche des contours est réalisée en utilisant l'algorithme de Teh-Chin [2].
- Construction de l'enveloppe convexe. Les points des contours sont divisés en deux ensembles selon la partie de l'image (gauche ou droite) à laquelle ils appartiennent. Pour chaque ensemble on construit une enveloppe convexe avec l'algorithme de Sklansky [3].
- La moyenne des coordonnées de chaque enveloppe est considérée comme la coordonnée de la main. En gardant l'historique des positions des mains on peut calculer la vitesse et l'accélération.
- On recommence la boucle.

Roadmap

L'algorithme étant développé et testé, il ne reste que l'intégration avec le *workpackages* "Gestion des événements", c'est-à-dire qu'il reste à créer l'objet actif qui servira d'interface au module de gestion des événements.

Partie 5

Caractérisation des événements

5.1

Introduction

Le *workpackages* de caractérisation des événements se charge du déclenchement des signaux MIDI fournis par l'interface MIDI lorsque des événements sont détectés dans des zones décrites par l'utilisateur et de l'affichage en surimpression des zones activées. Les spécifications nous donnent les types d'événements qu'il faut pouvoir détecter et quels types de zones on doit disposer et l'interface MIDI nous fournit ce qu'on doit envoyer. Il s'agit donc d'implémenter la détection d'événement, le déclenchement des signaux et le dessins des zones.

2. Teh, C.H. and Chin, R.T., On the Detection of Dominant Points on Digital Curve. PAMI 11 8, pp 859-872 (1989)
3. Sklansky, J., Finding the Convex Hull of a Simple Polygon. pp 79-83 (1982)

5.2

Description des zones

Une zones est décrites initialement par sa position, sa taille, son type (fadder, pad ou segment) et, pour chaque événement, un ensemble de signaux MIDI (éventuellement vide) qui doivent être déclenché lorsque cette événement est détecté. Les événements que l'on souhaite repéré dans une zone sont l'ouverture et fermeture de main, le déplacement de main, le franchissement pour un segment et l'entrée et la sortie pour un pad. (les pads XY seront gérés comme des fadders à axes doubles)

Pour détecter un événement, on dispose de deux positions de la main, l'actuelle et la précédente et de l'état de la main, ouverte ou fermée. De ces informations on détecte directement l'entrée/sortie d'une zone, le franchissement d'un segment. Pour les autres événements, il nous faut également l'état de la zone :

- active ou non pour lancer ou arrêter un clip
- l'ouverture de la main à l'entrée dans la zone pour les fadders, une main qui rentre fermée dans un fadder ne doit pas changer sa valeur.

Une fois un événement détecté, on envoie les signaux correspondant à l'interface midi.

5.3

Implémentation

La détection ET le déclenchement de signaux sont réalisés par les zones. La gestion des événements consiste à envoyer à chaque zone/segment les informations suffisantes relatives à la main. Il s'agit donc d'une boucle disposant de la liste des zones (fournis par la configuration) qui la parcourt en appelant pour chaque zone la détection.

5.4

Affichage des zones

Cette partie n'a pas encore été codé car est le résultat de notre dernière réunion. Les zones et les segments seront dessinés sur un Label Qt puis sera superposé à l'image de la caméra avec de la transparence. Il s'agit de montrer quelles zones sont actives et donc d'offrir un support visuel à l'utilisateur.

Partie 6

Workpackage Communication

6.1

Introduction

Le *workpackage* « communication » est composé de trois personnes : Luc Rocher (responsable d'équipe), Timothée Bernard & Hadrien Croubois.

Il s'occupe de la communication du projet sur deux échelles, en externe et en interne. La communication externe concerne l'ensemble des liens entre *MDMA* et les acteurs extérieurs (que ce soit par voie numérique – site Web, courriels –

ou par voie orale). La communication interne est chargée de gérer les discussions dans l'équipe, ainsi qu'avec les responsables du projet et du département.

6.2 Définition d'une identité graphique

Pour débuter une bonne communication, il faut d'abord réfléchir à l'image que l'on veut donner au projet. MDMA a pour but de développer un logiciel innovant : l'interface homme-machine se veut délibérément moderne, sans clavier ni souris l'utilisateur peut commander son ordinateur pour produire des signaux musicaux. Mais il s'inscrit dans une tendance alternative, celle de la musique électronique et des synthétiseurs sonores. Nous avons cherché à construire une identité assez *rétro* mais véhiculant la nouveauté du projet.

Nous sommes parti d'un ensemble de couleurs très vivantes – du jaune au rouge – assez agressives :

FIGURE 5 – Gamme de couleurs utilisées pour la charte graphique

Avec ces couleurs, nous avons pu réfléchir à un logotype. Plusieurs pistes étant possible, celle d'un texte uniquement (« MDMA ») assez minimaliste a été privilégié, avec une police très *pixelisée*. Trois propositions ont été réalisées pour affiner le logo :

FIGURE 6 – Trois propositions de logotype

Le premier graphique a été sélectionné par l'équipe, et rapidement intégré au site Internet.

6.3 Communication numérique

Site Internet

Afin d'obtenir un minimum de visibilité, le projet avait besoin d'une présence sur Internet. Nous avons donc d'abord travaillé à mettre en place un site Web. Nous avons travaillé sur une interface assez minimaliste, fondée sur les couleurs de la gamme représentée à la figure 5.

Le site Internet (`http://graal.ens-lyon.fr/mdma/`) contient différentes informations, dont :

- une présentation du projet, minimale sur la première page, puis détaillée par la suite ;
- une description complète de chaque *workpackage* ainsi que leurs avancées et découvertes au fil du temps ;
- une page personnelle pour chaque membre, avec la possibilité de nous contacter.

Le site est fondé sur *Wordpress* – un moteur dynamique – qui nous permet de gérer des commentaires pour chaque page, une première page dynamique avec la liste des derniers articles, des flux RSS… Par ailleurs, un **travail de traduction** a été réalisé sur presque l'intégralité du site, pour obtenir un site entièrement bilingue, en français et en anglais. Un menu en haut du site contient des boutons avec icônes pour sélectionner la langue ; par ailleurs, selon la langue du navigateur, la langue optimale est automatiquement choisie.

FIGURE 7 – Capture d'écran du site (version mobile)

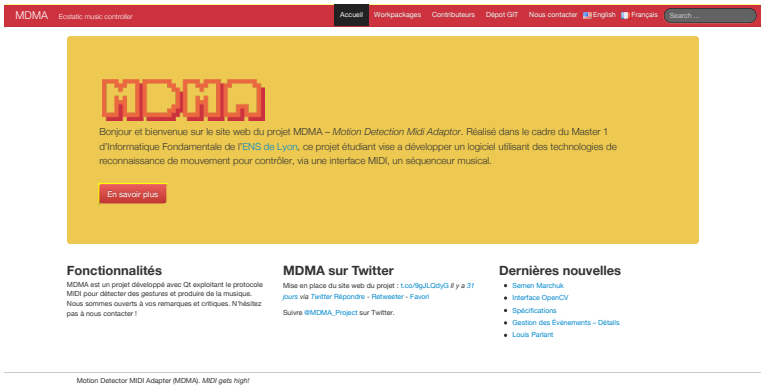

FIGURE 8 – Capture d'écran du site du projet

Autre présence

Le projet dispose aussi d'un page sur *Facebook* (`http://www.facebook.com/mdma.project`) et d'un compte *Twitter*. Ces espaces ne sont pas très actifs, mais pourront le devenir dès que le projet disposera d'un logiciel fonctionnel et pouvant être diffusé.

6.4

Diffusion numérique

L'équipe « communication » se charge du lien entre le projet et l'équipe pédagogique, par exemple pour des questions de budget (le financement récent d'un *Microsoft Kinect* par exemple) ; il a aussi pour vocation de diffuser notre projet à des musiciens et des personnes intéressées par la musique électronique. Nous chercherons ainsi des utilisateurs capable de tester MDMA et de produire des retours en situation réelle. Cette axe là pourra se développer dans les semaines qui viennent, avec la finalisation du logiciel.

6.5

Bilan

À l'orée de la moitié du projet, la communication a réalisé une grande partie du travail nécessaire. Il reste encore à rajouter du contenu sur le site, à favoriser les échanges avec les professionnels de la musique, et pourquoi pas à étudier un volet *presse*.

Partie 7

Conclusion

Au point où nous en sommes, nous avons déjà pu nous confronter aux difficultés réelles posées par chacune de nos exigences. Nous avons cependant contourné la plupart de ces problèmes en trouvant des solutions compatibles avec notre but premier, fournir un contrôleur reconnaissant nos mouvements avec un matériel minimal. Nous n'avons aucun retard par rapport au planning prévu, et l'avancée du projet s'annonce encourageante. Il nous reste à examiner des *workpackages*

Partie 8

Intégration et tests

Ce *workpackages* débutera mi-novembre, il s'agira d'assembler les différents modules afin de réaliser le logiciel final.

Partie 9

Conclusion

Au point où nous en sommes, nous avons déjà pu nous confronter aux difficultés réelles posées par chacune de nos exigences. Nous avons cependant contourné la plupart de ces problèmes en trouvant des solutions compatibles avec notre but premier, fournir un contrôleur reconnaissant nos mouvements avec un matériel minimal. Nous n'avons aucun retard par rapport au planning prévu, et l'avancée du projet s'annonce encourageante. Il nous reste à examiner des *workpackages* délicats comme celui s'attachant à intégrer et tester le logiciel final, mais il reste suffisamment de temps pour mener à bien cette partie du travail, et songer aux améliorations possibles. Nous envisageons notamment de développer entièrement une version utilisant la *Kinect*, et également de rendre MDMA totalement multi-plateforme.