ENSEIRB-MATMECA

Filière informatique, 2ème année

PFA - GUITAR TUTOR

Auteurs :
David BOUST
Sébastien DUBOUCHEZ
Taha ELHATIMI
Fabien FLEUREY
Emmanuel JULIEN
Alexia MENAND
Nicolas RETRAIN
Charaf-Eddine SHITIT

Encadrant :
David JANIN

9 mai 2012

Table des matières

I Introduction

Ce projet proposé par les chercheurs du Labri Mr. Pierre Hanna ainsi que Mr. Matthias Robine, consiste à réaliser un logiciel qui sert comme un support pédagogique à l'école de musique Rock et Chanson afin d'aider les élèves à jouer de la guitare. En effet, il s'agit d'un jeu interactif permettant aux élèves de jouer des morceaux de musiques édités préalablement par les professeurs grâce à une interface faisant défiler les accords à jouer et d'analyser par la suite leur performance (élèves). Nous avons commencé l'implémentation en se basant sur un projet du Labri nommé **GuitarTutor**.

I.1 Rappel des fonctionnalités

Rappelons les fonctionnalités à développer pour la réalisation d'un logiciel qui correspond pleinement aux attentes des clients.

Player
- Installation facile et multi-platforme
- Interface intuitive et facile à utiliser
- Choix de la musique à jouer
- Retour sur la performance de l'utilisateur

Editeur
- Création des tablatures
- Edition des tablatures
- Numérisation d'un morceau

II Installation facile

II.1 Rappel

Lors de l'obtention des sources, la compilation du projet devait se faire en ligne de commande et ce sur chaque plateforme différente. Nous disposions néanmoins d'une base de projet pour **QT**. Nous avons donc du apprendre à utiliser les fonctionnalités de cette librairie et de ses différents modules pour parvenir à prendre en main le code et commencer à y intégrer de nouvelles fonctionnalités.

Objectifs

Cette façon de procéder étant tout de même lourde, il nous avait été donné comme objectif de réaliser un installeur pour le logiciel afin de ne pas rebuter ses utilisateurs potentiels. Le but étant donc d'avoir un livrable sur un périphérique qui soit multiplateforme et simple d'utilisation. En effet, il avait été convenu avec les clients que la principale plateforme d'utilisation serait MAC OS X. Celà a posé toutefois d'assez nombreux problèmes car nous n'avions ni les connaissances, ni le matériel nécessaire à la mise en oeuvre d'un installeur sous MAC.
De plus deux installeurs différents ont été demandé par plateforme, selon que la version à installer soit destinée au professeur (version contenant l'Éditeur en plus) ou à l'élève. Ci-dessous un schéma de l'arborescence sur le périphérique.

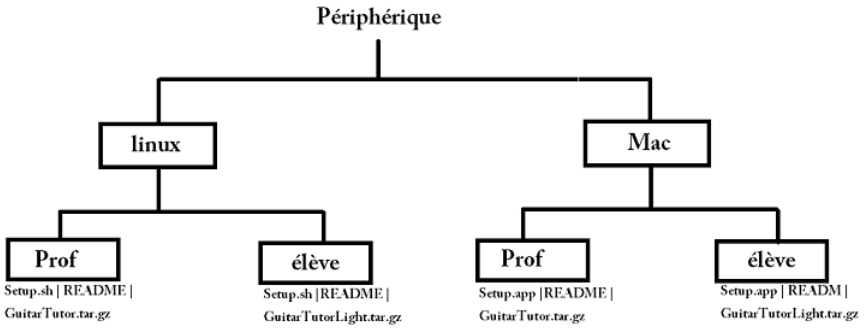

FIGURE II.1 – Schéma d'arborescence de l'installation

II.1.1 Rendu

L'installation se fait simplement par un double-clic sur une icône correspondant à l'installeur et communique avec le client par l'intermédiaire de boîtes de dialogue. Lors de cette installation le client doit pouvoir bénéficier des privilèges d'administrateur. Ce point nous a posé de nombreux problèmes pour l'installation sous MAC étant donné qu'il nous est impossible de privilégier de ces droits sur les machines de l'ENSEIRB.
Le premier point de l'installation est l'installation des librairies recquises, en voici la liste.
Pour Linux :

- libqt4-dev
- portaudio19
- libsndfile1-dev
- libfmodex4

Et pour MAC :

- qt4-base-mac-4.7.3-3
- portaudio18.1-3
- libsndfile1-dev1.0.25-1
- libfmodex4

Nous avions au départ opté pour une installation des librairies directement à partir des dépôts grâce à une connexion internet. Cela permettait d'avoir une compilation et un linkage des librairies propre et automatique. Il nous a été opposé que les dépôts n'étaient pas forcément mis à jour régulièrement et que le client ne bénéficiait pas forcément d'une connexion internet et nous avons alors dû revoir notre approche de l'installation. Nous avons donc dû archiver les sources des librairies et prévoir à la fois leur compilation et leur linkage dans notre installeur. Encore une fois, nous ne pouvions pas nous atteler à cette tâche des machines MAC de l'ENSEIRB étant donné que nous n'avions pas les droits.

Ensuite on installe le logiciel en soit (selon l'installation choisie c'est à dire la version professeur ou élève). On a un retour visuel et absolument aucune ligne de commande à taper comme on peut le voir sur les exemples ci-dessous :

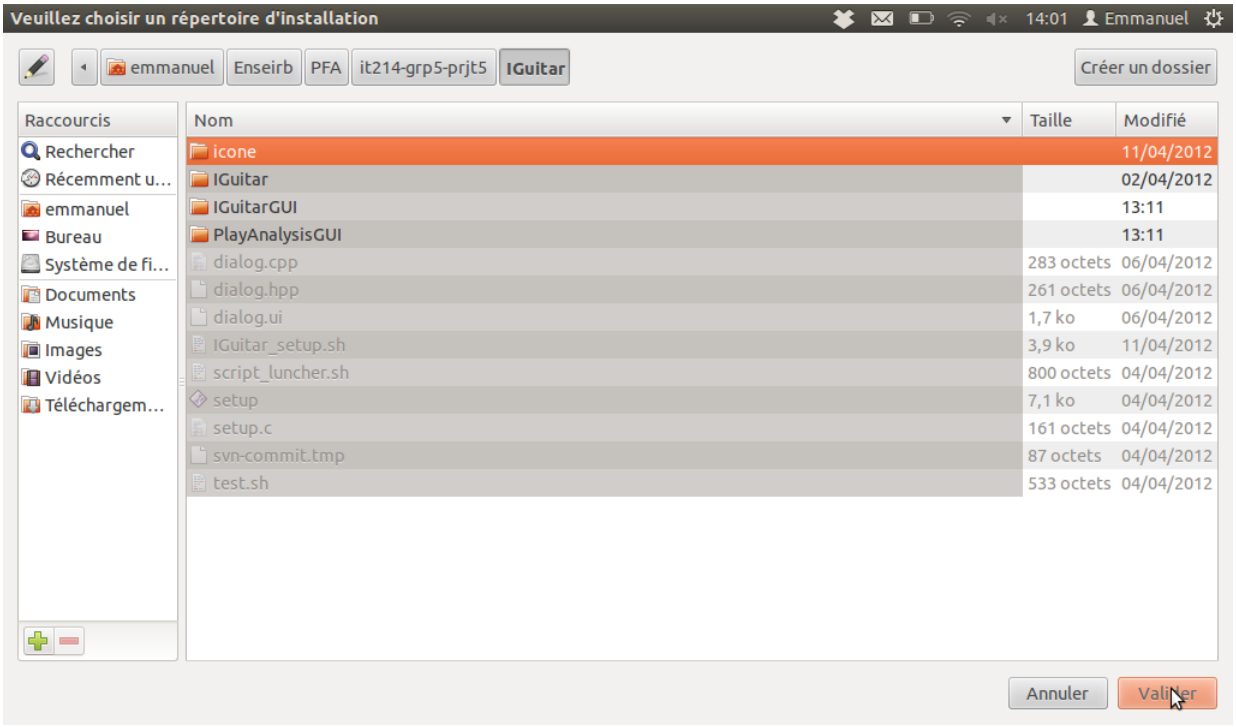

FIGURE II.2 – Capture d'écran 1

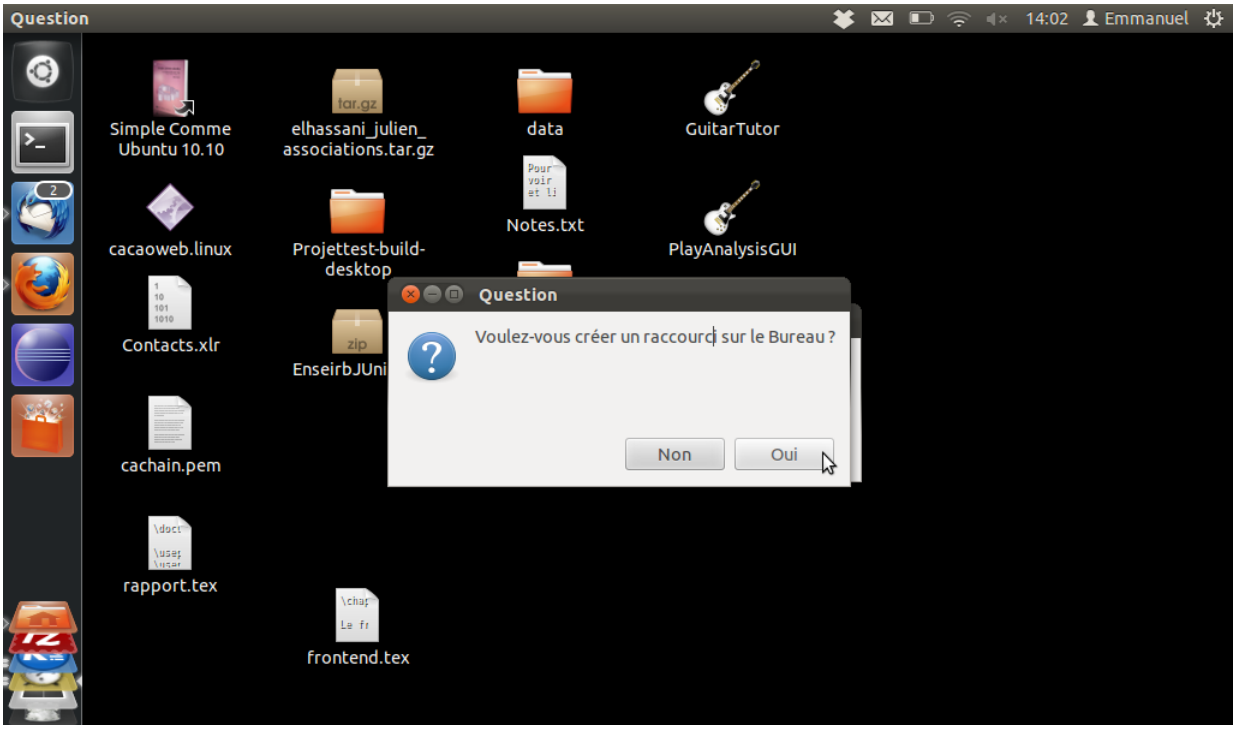

FIGURE II.3 – Capture d'écran 2

III Player

Pour la partie Player du logiciel, celle qui permet à un étudiant de joueur un morceau de guitar, le travail a été centré sur l'ajout de deux nouvelles fonctionnalités qui étaient vu comme les plus importantes :

- Une installation facile et multi plateformes (cf partie installation).
- Une interface permettant le choix des chansons.

Ce sont ces deux points que nous allons développer maintenant.

III.1 Choix des chansons

Dans la première version fournie par les chercheurs du LabRI, une seule chanson était disponible. Il s'agissait d'une version de démonstration du logiciel plus que d'une version purement dédiée à l'apprentissage de la guitare.

Avec la création de la partie éditeur, la bibliothèque des musiques jouables avec le logiciel a été étendue. Il existe désormais un répertoire comprenant pour chaque morceau jouable un fichier .txt et un .wav, ces deux fichiers contiennent toutes les informations nécessaire pour que le morceau soit jouable dans le Player de Guitar tutor.

Pour que le morceau soit joué, il faut le sélectionner au préalable. Pour ce faire, nous avons introduit une ComboBox, afin de lister facilement les choix possibles. Il suffit alors d'indiquer à partir de quel répertoire nous pouvons choisir une chanson, et le Player présente la liste des musiques disponibles dans ce répertoire.

Le joueur n'a plus qu'à selectionner un titre, une fois celui-ci sélectionné, la grille d'accords correspondant à la chanson est affichée, et sa lecture peut commencer.

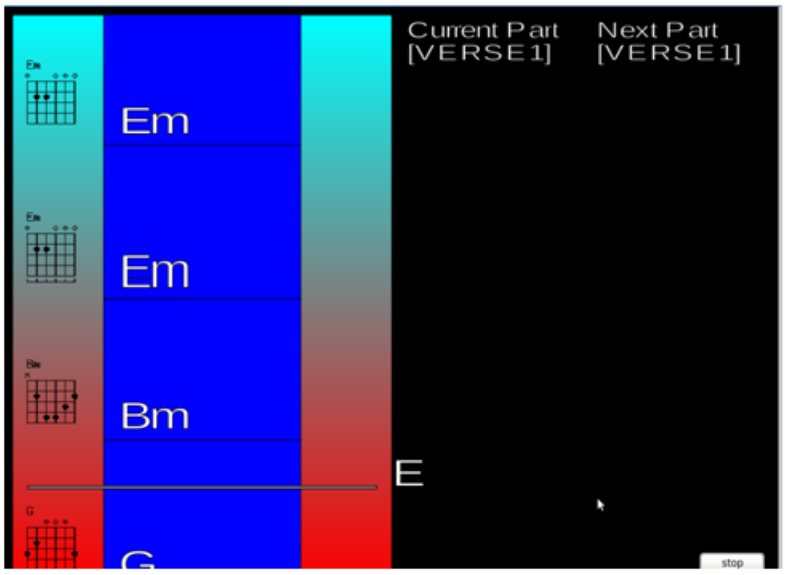

FIGURE III.1 – L'interface du Player

IV Editeur

Pour pouvoir utiliser le logiciel de jeu existant sur d'autres morceaux, il fallait être capable de créer un document compris par le logiciel de jeu mais correspondant à d'autres musiques. Pour ce faire, nous avons créé un logiciel d'édition de morceaux, qui pouvait créer un fichier correspondant au format lu par le logiciel de jeu. Le format d'échange des données, qui est décrit plus bas, nous a donc imposé de diviser la partie éditeur en deux. Chronologiquement, une première partie où l'utilisateur saisit les accords à jouer et une deuxième où il indique les moments auxquels il faut les jouer.

IV.1 Format d'échange

Le format d'échange entre le logiciel de jeu et le logiciel d'édition est le même que celui qu'utilisait auparavant le logiciel de jeu. En voici un exemple :

```
[INTRO]
1.914761 G
1.455219 Bm
3.149589 Em
6.328478 C
[VERSE1]
1.577916 G
1.497687 Bm
1.567347 Em
```

Comme nous pouvons le voir, deux ou trois éléments le compose :
- des annotations entre crochets
- les notes à jouer
- la durée de la note, qui correspond au temps entre deux notes

Pour répondre à ce format, nous avons séparés la saisi des notes et des annotations dans la première partie et la saisi des intervalles de temps dans la seconde.

IV.2 Saisie des notes

Tout d'abord, l'utilisateur commence par entrer les accords dans une grille dédiée. Par défaut il n'y a que 4 colonnes sur cette grille, plus celle d'annotation, mais il est possible de faire varier le

nombre de lignes et de colonnes de la grille. Le choix du nombre d'accords se fera en cliquant sur un bouton **créer un morceau**.

L'utilisateur peut alors commencer à remplir la grille en remplissant les cases avec des notes mises à sa disposition. Pour ce faire, il peut sélectionner une ou plusieures cases, et choisir ensuite dans la liste à gauche l'accord souhaité en double cliquant dessus.

Plusieurs manipulations de la grille d'accords sont possibles. L'utilisateur peut supprimer une ligne en cliquant sur le bouton **delete row** après l'avoir sélectionnée en cliquant sur son chiffre. La suppression est aussi bien possible pour la dernière ligne que pour une autre. Vous pouvez supprimer plusieurs lignes en en sélectionnant plusieurs en restant appuyer sur **ctrl** puis en cliquant sur les chiffres des lignes à supprimer.

Les cases d'annotations sont des cases de texte libre permettant de renseigner les informations souhaités par l'utilisateur.

Une copie est églament possible pour faciliter la saisie des notes. En effet, l'utilisateur peut copier sur la ligne en dessous un bloc de plusieurs lignes qui se répète avec le bouton **Copy down**.

Les boutons **Open** et **Save** permettent respectivement d'ouvrir la dernière grille créé, et de sauvegarder la grille en cours de création. Voici une capture d'écran illustrant la fenêtre intégrant cette grille d'accords et les boutons en question :

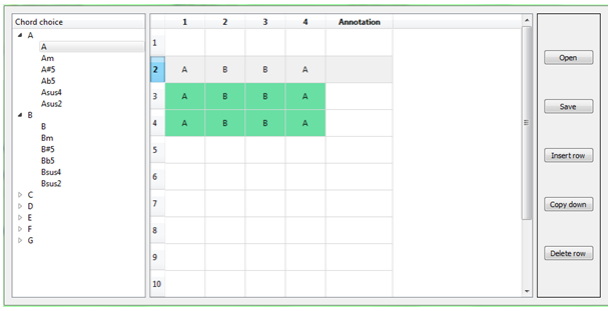

FIGURE IV.1 – Interface de saisie des notes

IV.3 Saisie des temps

Une fois la saisie des notes réalisée, l'utilisateur accède à une seconde vue pour réaliser la saisie des temps. Voici l'interface qu'aperçoit l'utilisateur : Nous pouvons observer, de haut en bas :

- nom de la piste en lecture
- accords précédemment saisis pour lesquels il faut saisir un temps
- la position dans la musique
- un icone représentant l'état actuel de la musique
- les trois boutons pour gérer la lecture de la musique ainsi que le bouton pour passer en mode visualisation

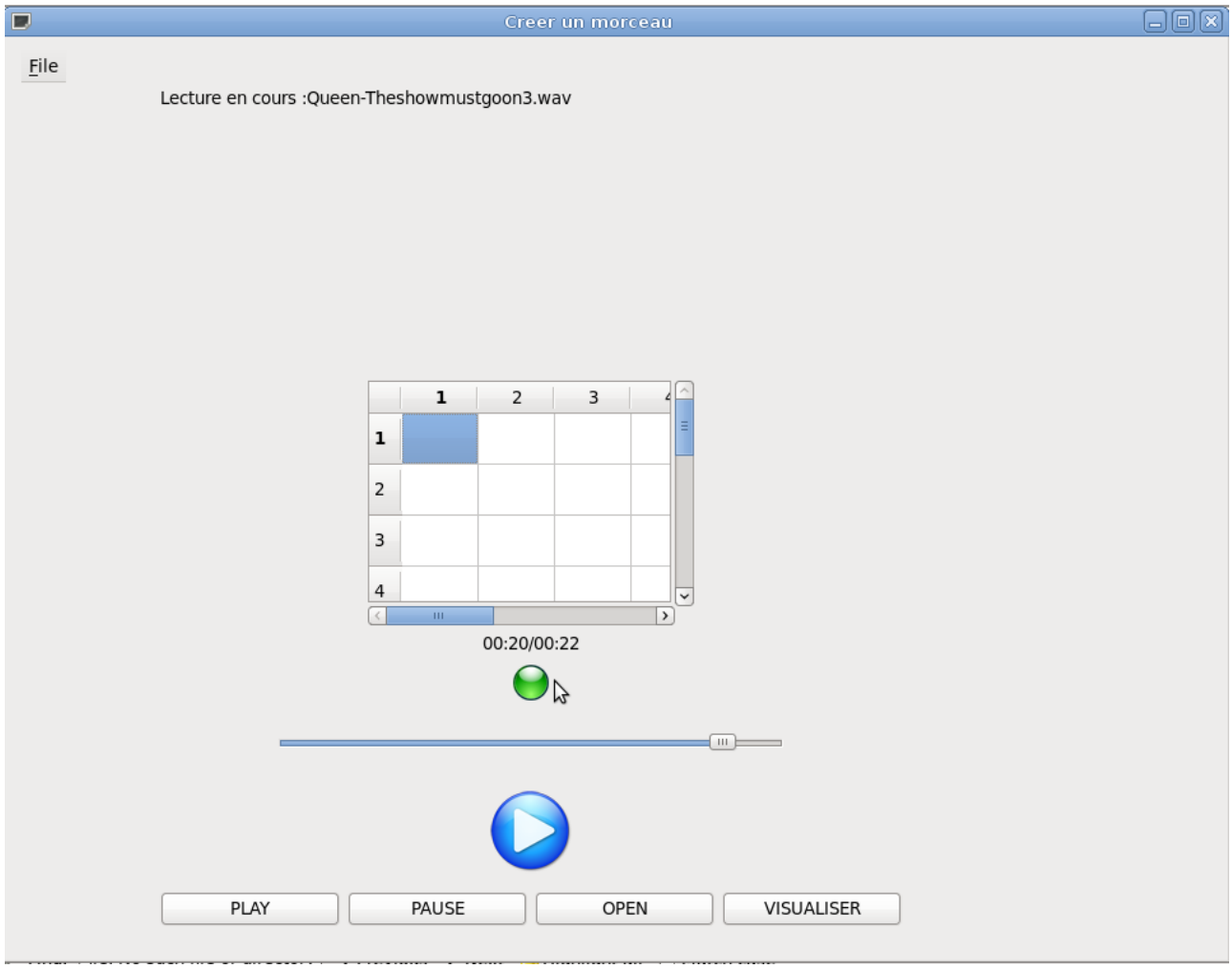

FIGURE IV.2 – Interface de saisie des temps

IV.3.1 fonctionnement

Afin de réaliser le fichier texte correspondant à la musique que l'on veut créer, l'utilisateur commence par sélectionner le fichier wav lui correspondant. Nous avons implémenté trois manières d'ouvrir un fichier :

- à partir de la barre de menu file -> ouvrir une musique
- en utilisant le raccourci clavier Ctrl + o
- en appuyant sur le bouton **ouvrir**

Une fois la musique chargée, l'utilisateur dispose des fonctionnalités d'un lecteur classique, c'est-à-dire la lecture, la pause ainsi que le déplacement dans la musique en déplacant le curseur de la

barre d'avancement. Il peut, bien entendu, également changer de musique en réutilisant la barre des menus ou le raccourci Ctrl + o.

La deuxième fonctionnalité majeure est la saisi des temps où il faut jouer les accords. Pour cela, l'utilisateur doit, tout en écoutant la musique, appuyer sur la barre espace pour saisir un temps. Si l'utilisateur se trompe, il peut revenir en arrière avec le curseur de la barre d'avancement qui aura pour effet de supprimer toutes les saisies postérieures à la nouvelle postion du curseur dans la musique.

Enfin il exsite un mode de visualisation pour voir les temps saisis. Pour l'activer, il suffit de cliquer sur le bouton Visualisation, la saisie des temps sera toujours possible, mais le déplacement dans la musique ne supprime plus les temps saisies. Pour le quitter il suffit simplement de cliquer sur le bouton Stop Visualisation.

Enfin lorsque l'utilisateur est satisfait des temps et des accords saisis, il lui suffit d'exporter les données dans le format voulu en utilisant la barre des menus ou le raccourci Ctrl + e.

IV.3.2 implémentation

Pour réaliser cette partie, le code a été divisé en deux. Une première classe se chargeant de la partie lecture audio et saisi des temps et une deuxième qui s'occupe de la phase de visualisation.

Lors de la création d'un nouveau morceaux, la classe createwindow se charge d'instancier un objet player, qui correspont à la première partie cité ci-dessus, et de connecter les signaux aux méthodes. Ensuite c'est l'instance du player qui se charge de créer un visualisationthread dès lors que l'utilisateur souhaite visualiser ce qu'il a déjà saisi.
Enfin pour aider l'utilisateur une barre des menus QMenuBar reste à disposition.

IV.3.3 correspondance avec les demandes du client

La partie lecture audio n'ayant pas été abordé avec le client, nous nous sommes contentés des fonctionnalités de base, contrairement à la saisi des temps par la barre espace qui avait été discuté et dont nous avions trouvé accord. Un retour visuel et une visualisation possible de ce qui a été fait a été ajouté pour permettre une meilleur utilisation.

IV.4 Evaluation cognitive

Après la mise en place de l'éditeur, nous avons essayé d'évaluer qualitativement l'interface de celui-ci mettant en oeuvre des techniques de cognitique utiles à l'évaluation ergonomique d'une interface. Nous avons centré la validation de notre interface sur un nombre limité de critères ergonomiques dont l'évaluation se fera d'une part avec une évaluation analytique : la Balade cognitive, et d'autre part avec une évaluation expérimentale : les tests utilisateurs.

- La balade cognitive : Nous avons essayé d'imaginer tous les scénarii possibles pour la création d'un morçeau. Par la suite, nous avons effectué des modifications sur l'interface en prenant en considération les problèmes rencontrés.
- Les tests utilisateurs : Nous avons créé un recette de test utilisateurs. Ensuite, nous avons déroulé le scénario test "création d'un morceau" en invitant un élève de l'ENSEIRB-MATMECA à utiliser le logiciel éditeur.

Enfin, nous nous sommes intéressés à l'évaluation de cette expérimentation et finir par effectuer des modifications sur l'interface.

V Partie Génie Logiciel

V.1 Méthodologie

Pour la reprise et le développement du logiciel Guitar Tutor, la méthodologie adoptée a été de réaliser la reprise et le développement de celui-ci selon une méthode AGILE. Nous avions opté pour la méthode SCRUM, qui consiste à faire des sprints de développement, et des réunions régulièrement de mise au point après l'intégration des modules en fin de sprint. Le but étant que le logiciel puisse tourner à la fin de chacun des sprints et que chaque développeur soit occupé pour chacun de ces sprints. Lors de ces réunions, nous vérifions que le code intégré est bien fonctionnel et que nous n'avons pas régressé.Nous avions planifié 5 sprints pour répartir les taches entre plusieurs binômes de travail, tous le long du temps de développement. Au fur et à mesure nous avons du réduire le nombre de fonctionnalités, en privilégiant les plus essentielles, en accord avec nos clients. Car le développement du programme s'est avéré bien plus difficile que prévu, de plus notre cahier des charges était par trop ambitieux. Nous avons donc développé en priorité les fonctionnalités prévu pour l'éditeur, la possibilité de choisir un morceau dans le Player, et enfin une installation facile et rapide.

V.2 Outil de suivi

Pour le suivie notre projet, nous avons décidé d'utiliser la forge de l'ENSEIRB-MATMECA. Elle dispose en plus du dépôt subversion de différents outils de gestion. Nous avons notamment utilisé un diagramme de Gantt sur lequel nous avons créé les taches et les sous-taches a réaliser. Un système de filtre permet d'y voir rapidement les taches qui sont en cours de développement, terminées, et celles qui sont attribuées ou non attribuées.

V.3 Analyse

L'expérience de la méthode agile n'a pas été concluante pour notre groupe de travail. Si elle est efficace dans le monde professionnel, nous avons éprouvé quelques difficultés à les mettre en œuvre, surtout dans notre cadre scolaire. En effet notre groupe de PFA est composé d'élèves qui suivent différentes options, et n'ont pas cours en même temps. Et si au début du semestre des créneaux étaient réservés au PFA, ils ont souvent été utilisé par l'administration comme créneau de rattrapage de cours. Et sur la fin, nous n'avions plus du tout de créneau réservé à la réalisation du PFA, les réunions elle même étaient donc difficiles à mettre en place. Nous pensions faire des

sprints de 2 semaines, mais nous nous sommes rendu compte que cette durée était trop courte, nous avons donc repoussé d'une semaine la fin des sprints. De plus, le temps passé à mettre à jour le diagramme de Gantt d'un sprint à un autre était considérable (report des taches non accomplies, attribution de toutes les taches à un binôme). Autant de temps qui finalement n'était pas utilisé pour le développement.

V.4 Reprise de code

Nous avons sous estimé la charge de travail inhérente à la reprise du code originel du logiciel, et au temps de formation sur les librairies (FMODEX, Qt) et sur le développement sous mac. L'absence de documentation, de **raedme**, ou d'autres aides à la compréhension du code nous a conduit à perdre du temps dans la maîtrise de celui-ci et à tâtonner avant de pouvoir commencer le développement. De plus, nous avons du nous familiariser avec le modèle MVC, utilisé dans le code fourni, avant de pouvoir interragir proprement.

V.5 Documentation et tests

Dans un but de pérennisation du code du projet, nous avons particulièrement fait attention à la rédaction de manuels. Nous avons rédigé un manuel de maintenance qui explique l'architecture du code ainsi que les points délicats de celui-ci, et un manuel d'utilisation à destination des clients qui explique comment installer et utiliser toutes les fonctionnalités des produits.(Vous trouverez ces manuels en annexes.) Nous avions prévu de faire des tests avec l'école de musique afin prendre en compte leur dernière remarque sur les livrables. Malheureusement nous n'avons pas eu le temps de mettre en place cette réunion car nous n'avions pas fini le développement à temps. Nous avons donc exécuté ces tests nous même en essayant d'être le plus critique possible.

VI Conclusion

Pour conclure, ce projet fut très enrichissant puisqu'il s'agit de l'une de nos premières expériences dans un projet concret qui réunit plusieurs compétences. En effet, ce projet nous a permis de mettre en pratique différentes compétences souvent théoriques pour aboutir à cette application notamment le développement logiciel, la programmation C++ et d'autres aspects de génie logiciel.

Ce projet nous a également permis d'acquérir une expérience dans le volet de conduite de projet (réunions client, réunions avec l'encadrant pédagogique, réunions au sein de l'équipe, respect des deadlines des sprints ...). Nous avons pu mettre en oeuvre des pratiques nécessaires à la satisfaction des clients et respecter les consignes de nos encadrants pédagogiques.

Le découpage du projet en deux parties **Editeur** et **player**, ainsi que la répartition des tâches au sein du groupe nous a permis de tous nous focaliser sur la partie assignée.
Nous estimons que ce projet reste intéressant et formateur pour tous les membres de l'équipe par rapport à notre cursus de formation d'ingénieur informaticien.

Nous remercions M. David Janin, M. Pierre Hanna ainsi que M. Matthias Robine pour les conseils qu'ils nous ont prodigués.