Rapport technique final

Adonis Najimi,
Valentin Stern,
Vincent Albert,
Théo Gerriet

31 mars 2014

1 Client C++

1.1 Description technique

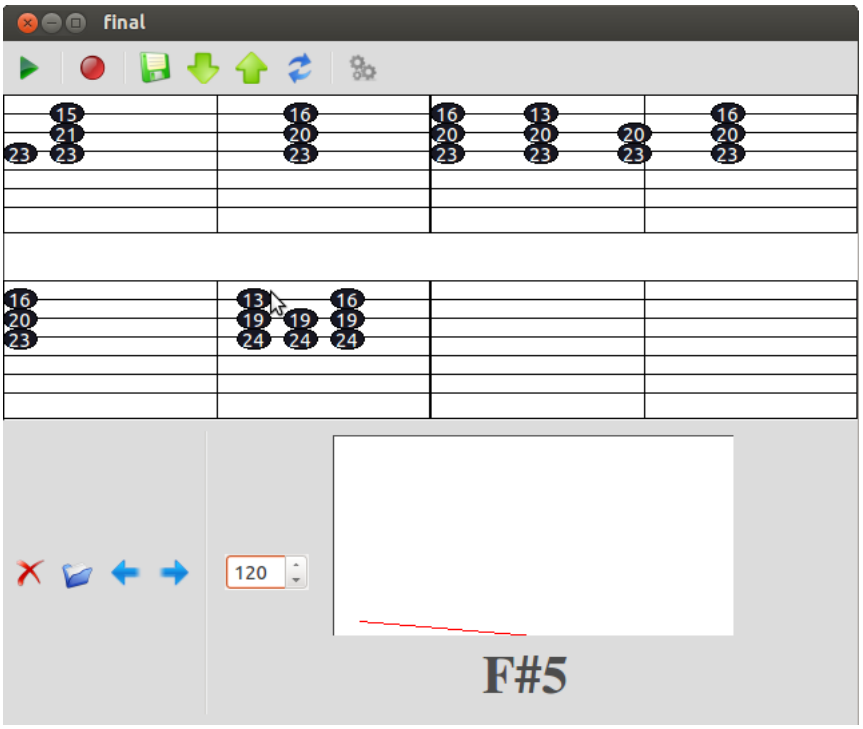

FIGURE 1 – Rendu final de l'application

Afin de lier l'application cliente et l'application web, nous utilisons un format de sauvegarde partagé par les deux langages (C++ et JS), le JSON. Nous avons donc conçu un simple parser JSON afin de pouvoir sauvegarder une partition soit sur son PC local, soit dans la base de donnée distante pour la partager avec le monde. Nous avons choisi ce format d'enregistrement car il est aisément lisible sur le serveur qui peut alors traiter rapidement les informations pour les afficher au utilisateurs, et car c'est un format de plus en plus répandu. Nous aurions pu prendre du XML, mais il aurait été moins bien traité côté serveur. De plus, le format JSON consomme moins de place en mémoire que le format XML, car il est légèrement moins verbeux. L'application possède donc des fonctionnalités de sauvegardes sous forme de fichier et de sauvegarde sur le c̈loudr̈epresenté par le site web. Il est possible de réouvrir une partition sauvegardée en local.

La bibliothèque FMOD Ex nous permet de récupérer le son entrant de l'application. Elle a été choisi car elle est simple d'utilisation tout en étant efficace. Cela nous a permis de nous concentrer sur le traitement de ce son, qui constitue une des parties les plus importante du projet. Nous avons suivi l'algorithme du rapport de conception, et y avons apporté quelques modifications pour l'implémentation réelle.

La bibliothèque Qt ne crée pas de nouveau thread pour sa partie graphique comme Swing par exemple, nous avons donc du créer deux nouveaux thread pour l'enregistrement de partitions et l'accordeur. Il a été aisé d'implementer ces thread car Qt nous permet d'en faire un héritant d'une de ses classes. Sans ces threads, l'applications serait par exemple bloquée si l'on appuie sur le bouton d'enregistrement, car celui ne serait plus accessible une fois l'enregistrement lancé.

La quantité de fréquences a traiter étant importante (8192 tous les pas de temps), nous avons décidé d'utiliser le tri rapide pour un maximum de performance. Nous avons due y ajouter une surcouche sauvegardant les places de chaque fréquences car FMOD assigne chaque volume de fréquence à sa place dans le tableau. Si nous avions uniquement trié le tableau de fréquences en fonction du volume, nous n'aurions plus la possibilité de savoir quelle fréquence a le volume le plus haut.

Nous avons choisi de fournir deux classes sous le design pattern du singleton car cela simplifiait beaucoup le développement, et car plusieurs instances de ces deux classes serait inutiles. La classe Notes fournit donc toutes les notes possiblement enregistrable dans la musique, et la classe Guitar fournit la possibilité de convertir les notes jouées en frettes pour permettre l'affichage de la tablature. Nous nous sommes donc plus concentré sur les tablatures pour guitare et non les partitions en général, mais il serait possible de l'adapter pour tous les instrument. Il est également possible de changer l'accordage de la guitare, mais uniquement en changeant le code, faute d'un manque de temps pour développer cette fonctionnalité.

1.2 Ambitions

La lecture d'une partition faisait partie d'une des fonctionnalités à priorité haute de l'application. Après de nombreuses recherches nous n'avons pas pu récupérer des fichiers MIDI correspondant aux sons de chaques notes de musique possible pour permettre la lecture de l'application, et c'est pour cela que la fonctionnalité n'a pas été réalisée.

Nous souhaitions faire apparaitre les notes dans la partition en temps réel, mais cela a été impossible à cause du fait que l'enregistrement de la partition se fait dans un Thread, et la création de nouveaux objets est interdit par Qt. Nous avions donc une erreur, et une améliorations possible serait de permettre cette fonctionnalité.

Nous nous sommes surtout concentré sur l'enregistrement de partition à partir du micro, nous n'avons donc pas fini de développer complètement la modification des partitions après cet enregistrement.

La répartition des taches pour le client a un peu été changée à la fin du projet. Nous avions pensé que le dévelopemment du modèle serait aussi important que celui de la partie graphique, or en fin de projet, nous avons été plus nombreux à développer la partie graphique. Cela est surtout du au fait qu'il fallait lier la

partie graphique et la partie des données, et nous avons donc plus pu travailler en collaboration à ce moment là. L'intégration n'a pas engendré trop de difficultés, car nous avions conçu dès le début les deux parties de sorte que tous les éléments soient là pour nous faciliter la jointure.

1.3 Choix

Le langage C++ a été un choix apprécié pour l'élaboration du client. Le langage est puissant et rapide ce qui permet un traitement des fréquences très rapide, malgré une quantité d'informations importante. L'étude d'un nouveau langage au sein de ce projet a donc été une chose bénéfique car le langage a directement été appliqué à la pratique.

Pour l'analyse des sons entrant, le choix de la bibliothèque FMOD n'a pas été regretté pour le developpement. Une fois installée, elle permet la récupération de toutes les fréquences entrant par le micro, et cela simplement.

1.4 Améliorations possibles

Au niveau de la prise de sons, il y a la possibilité d'améliorer celle ci et obtenir plus de précision en implémentant un algorithme permettant de détecter quels sont les seuils à prendre en compte pour la prise de son et obtenir une analyse beaucoup plus fine du son entrant et ainsi des partitions correspondant mieux à la réalité.

Au niveau de la sauvegarde des partitions, il serait possible de compresser un peu plus chaque fichier car il y a beaucoup d'espaces. Cela consisterait à améliorer le parser JSON des partitions. En effet, sur un petit fichier de par exemple 300 caractères, représentant ainsi une partition de 2 accords comportant en tout 7 notes, le gain possible en enlevant les espaces est de 40%. Ainsi, le remplissage de la base de données serait moins rapide, et les performances seraient également plus grandes.

```
{  " name " : " capture.tab " ,  " tempo " : " 120 " ,  " chords
" : [  {  " volume " : 0.0262736 ,  " duration " : 7 ,  " notes
" :  {  " F#2 "  }  }  ,  {  " volume " : 0.0455792 ,  "
duration " : 14 ,  " notes " :  {  " F#2 "  ,  " F2 "  ,  " A#2
"  ,  " B2 "  ,  " G2 "  ,  " G#2 "  }  } ]  }
```

FIGURE 2 – Fichier representant une partition

Une grande amélioration serait également de rajouter la possibilité d'enregistrer plusieurs pistes. Cela permettrait donc de composer des musiques plus complètes avec par exemple une guitare rythmique et une guitare solo.

2 Application Web

Table des matières

1 Client C++ **2**
- 1.1 Description technique 2
- 1.2 Ambitions 3
- 1.3 Choix 3
- 1.4 Améliorations possibles 3

2 Application Web **5**