

Rapport technique final

Adonis NAJIMI,
Valentin STERN,
Vincent ALBERT,
Théo GERRIET

31 mars 2014

1 Client C++

1.1 Description technique

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 cloud représenté 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'implémenter 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.

1.2 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.

1.3 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 chaque note 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 apparaître 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 interdite par Qt. Nous avons donc une erreur, et une amélioration possible serait de permettre cette fonctionnalité.

1.4 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 développement. Une fois installée, elle permet la récupération de toutes les fréquences entrant par le micro, et cela simplement.

2 Application Web

Table des matières

1	Client C++	2
1.1	Améliorations possibles	2
1.2	Ambitions	2
1.3	Choix	2
2	Application Web	3