Rapport de projet

Par

François Poitras

Présenté à

Olivier Bélanger

Dans le cadre du cours de

Création Musicale en Language Python 2 (MUS3327)

22 avril 2017

Université de Montréal

1 Rapport de travail

Le projet réalisé dans le cadre du cours est un constructeur de progressions d'accord. L'utilisateur a le choix parmis les douzes tonalités et une panoplie de qualités d'accords pour développer une progression. Pour des raisons de simplicité, l'ajout et la suppression d'accords fonctionne comme une pile. C'est à dire qu'un ajout *push* et qu'une suppression *pop* sur la progression. L'utilisateur est aussi limité à un accord par mesure. Il s'agit des seules informations que l'utilisateur doit fournir au programme. Celui-ci génère ensuite la progression à l'aide de notes MIDI et génère également une ligne de basse en fonction de l'accord courant.

1.1 Résumé du développement

Les fonctionnalités principales développées dans le plan de travail ont été respectées. Les fonctionnalités plus complexes, comme la synthèse d'instrument réels a été changée. Il y a aussi quelques fonctionnalités qui ont été tout simplement abandonées. Il n'y a pas eu de problèmes majeurs dans le reste des fonctionnalités, à l'exception de quelque crashs inexpliqués qui ont pu être résolus.

1.2 Divergences par rapport au plan

Le plan initial prévoyait une génération de mélodie pour accompagner les accords. En raison de la complexité de cette fonctionnalité, celle ci a été abandonée. Il serait toutefois possible d'ajouter celle-ci dans un développement futur de l'application. Le programme génère une ligne de basse simple pour aller avec les accords. Il s'agit d'une simple arpégiation de l'accord qui est utilisé dans une mesure donnée. Comme la génération n'a pas été implémentée, la synthèse des accords s'en retrouve simplifiée, car il y a seulement un cas d'utilisation.

Le plan mentionnait aussi une synthèse plus ou moins avancée d'instruments réels. Il était entre autres question d'un piano et d'un instrument à vent pour les basses. La synthèse d'instruments étant un domaine extrêmement complexe, les deux synthétiseurs de l'application essaient plus ou moins d'imiter des instruments réels. Le premier synthétiseur est une imitation approximative d'un marimba. Ce synthétiseur est utilisé pour les accords. Un deuxième synthétiseur est utilisé dans les basses. Ce dernier ne se veut pas une imitiation d'un instrument réel et est donc beaucoup plus simple. Il est toutefois assez efficace dans les basses fréquences.

1.2.1 Problèmes rencontrés

Le principal problème a été un *segmentation fault* dans le séquenceur. A première vue, il semblait que le problème venait de pyo directement. Le crash se produisait au moment Toutefois, une mise à jour de la librairie n'a pas permis de règler entièrement le problème. La solution a donc été d'utiliser moins de pyo pour réussir à faire fonctionner la librairie. Pour remplacer les objets *Iter*, le séquenceur utilise des *TrigFunc*. Le principal inconvénient est que le séquenceur dépend maintenant de Python (et non de pyo)

en raison des appels à des fonctions lors de triggers. Puisque le projet n'utilise pas beaucoup de processus audio, le problème de timing n'est pas vraiment remarquable.

2 Perspectives

2.1 Mélodie

La suite logique du projet est de développer l'aspect qui a été mis de coté. La complexité de cet élément – il s'agit après tout d'établir une mélodie intéressante basée sur une progression arbitraire d'accords – fait en sorte que ce projet pourrait faire l'objet à part entière d'un autre projet, basé sur celui existant. Pour avoir une mélodie qui semble correcte, une stratégie possible est d'utiliser l'ensemble des notes de la progression et de déterminer quelle gamme est la plus appropriée à l'ensemble. Autrement dit, on cherche à maximiser les notes communes des accords fournis et d'une gamme. Ensuite, pour générer une mélodie intéressante, on peut développer quelques thèmes au préalables et les développer. Ce ne sont là que des idées préliminaires et dans le cas d'un projet plus poussé, d'autres options seront sans doute évaluées.

2.2 Basse

La ligne de basse est plutôt primitive et utilise une simple arpégiation de l'accord courant. De plus, si l'accord est plutôt complexe (cinq voix et plus), on perd les notes les plus hautes de l'accord car elles sont jouées séquentiellement. C'est à dire que le premier temps est toujours la fondamentale, le deuxième la tierce, le troisième la quinte et le quatrième la septième de l'accord. Il pourrait être plus intéressant d'essayer de construire une ligne de *walking bass* plus typique, par exemple en utilisant des tons d'approche pour anticiper le changement ou en changent l'ordre des notes qui sont jouées.

2.3 Style musical

Une autre amélioration est de développer différents styles musicaux. Dans la version actuelle du programme, les accords sont des blanches. Il y en a donc deux par mesure. La basse est d'une note par temps. Ce manque de flexibilité pourrait être corrigé en permettant aux usagers de placer eux-mêmes les accords dans une mesure. Ainsi, ils pourraient explorer les accords sur des contre-temps, par exemple, ou encore avec des durées différentes.

2.4 Éléments graphiques

L'interface graphique en tant que tel est assez intuitive à utiliser, mais la construction elle même de progression d'accords pourrait être révisée. Le plus simple serait de demander à l'usager à quelle position il veut placer l'accord qu'il a choisit. Cela pourrait alourdir l'interface, mais il s'agit d'une solution simple à implémenter. Une solution plus complexe est d'implémenter un système de *drag and drop*, tel que mentionné dans le plan initial. De cette façon, l'utilisateur pourrait déplacer à sa guise les accords qu'il a placé et cela garderait l'esprit intuitif de l'interface.