Université CathOzlique de Louvain

EcOzle Polytechnique de Louvain

OZ Player

Rapport de PrOzjet

LINFOz1104

Groupe 5

Christophe KAFROUNI 37961800
Tom KENDA 32001700

Titulaire du cours

Peter VAN ROY

Assistants

Thomas WIRTGEN
François DE KEERSMAEKER

Année académique 2021-2022
2 mai 2022

1. Limitations et problèmes du programme

Nous avons pris beaucoup de plaisir à implémenter cet *Oz Mixer*. Nous avons veillé à ce qu'il soit le plus intuitif et simple d'utilisation dans le cas où d'autre utilisateur venait à en faire usage. Si les spécifications de chacune des fonctions sont bien respectées il ne devrait donc pas y avoir de problème.

Une des limitations est justement que le programme suppose une bonne compréhension des spécifications et très peu de traitements des erreurs (*error handling*) ont été implémentés. La gestion des erreurs en Oz n'étant pas très étoffée non plus, cela pourrait rendre la compréhension des erreurs un peu compliqué.

Afin de rendre le code robuste et efficace nous avons préféré l'utilisation de construction récursive terminale. Pour les cas où nous avons eu besoin de fonction *Fold* nous avons veillé à utiliser la fonction *FoldL*[1] plutôt que *FoldR*. En effet, cette dernière n'est pas récursive terminal puisque qu'elle doit toujours aller chercher d'abord dans le membre de droite pour faire le calcul et elle doit ainsi garder en mémoire un grand nombre d'élément avant de pouvoir faire le calcul sur le premier membre de gauche. Il ne devrait donc pas y avoir de problème de mémoire ou de long temps de calcul même dans le cas où nous devrions jouer de très longue partition (i.e. de très longues listes et donc beaucoup d'élément à garder en mémoire).

Une deuxième limitation de notre programme est qu'il est écrit entièrement de manière séquentielle. Cela pourrait limiter sa vitesse de traitement des partitions. Une manière d'améliorer la vitesse du programme est probablement d'utiliser des *threads* à différents endroits car certains calculs peuvent être faits en parallèle. Mais, en pratique, le programme sera réellement plus rapide uniquement si l'ordinateur dispose de plusieurs cœurs.

2. Constructions non-déclaratives

Aucune construction non-déclarative n'as été utilisée durant ce projet. Puisque c'était toujours possible nous avons évité l'utilisation des cellules et nous avons utilisé des fonctions récursives terminales plutôt que des boucles. Les fonctions *Map* et *FoldL,* prédéfinie dans Oz, nous ont particulièrement été utile à de nombreuses reprises pour éviter les boucles et adopté une écriture plus déclarative.

3. Choix d'implémentation surprenant

Selon nous, rien de très surprenant n'a été réalisé dans notre code. Nous avons essayé d'utiliser un maximum de fonctions de la librairie standard de Oz pour elles sont efficaces[2] et plus facilement compréhensible par le lecteur.

---

[1] Fonction prédéfinie dans Mozart : lien vers la documentation

[2] Au départ nous avions implémenté nous même quelques fonctions sur les listes sans aller lire toute la documentation Oz, mais nous nous sommes vite rendu compte que les fonctions prédéfinies étaient plus efficaces !

4. Extensions | créativité

Pour cette partie nous avons été challengés par mon grand-père pianiste. Nous lui avons expliqué le projet et il nous a mis au défi de coder la partition de Scott Joplin "The Entertainer".

Au vu de la longueur de la musique nous nous sommes limités au début (d'autant plus qu'il y a beaucoup de répétitions). Le code pour réaliser la partition est fourni dans le fichier *creation.dj.oz* dans la section « Creative Bonus » de INGInious. Il est accompagné du fichier *creation.wav* qui contient la composition musicale entière, ainsi que du fichier *intro.wav* qui sert juste pour faire les 6 premières secondes de la musique.