LINFO1104 - Concepts, paradigmes et sémantique des langages de programmation

Projet Son'OZ

Vincent Meessen David Mutaku Fwala

Groupe: F

02 Mai 2025

Table des matières

1 Introduction

Ce projet a été réalisé dans le cadre du cours "**LINFO1104** - Concepts, paradigmes et sémantique des langages de programmation" qui consiste à

Ce projet a été réalisé dans le cadre du cours "**LINFO1104** – Concepts, paradigmes et sémantique des langages de programmation". L'objectif de cette année est de faire vibrer la Son'OZ de Mozart à travers un projet en deux parties, axé sur le traitement musical en langage Oz.

Dans la première partie du projet, nous avons implémenté la fonction **PartitionToTimedList**, dont le rôle est de transformer une partition musicale en une liste structurée de notes et d'accords, incluant leurs durées. Cette étape permet de préparer les données musicales en vue de leur traitement audio par parties **partition(⟨partition⟩)** de la fonction **Mix**.

La seconde partie du projet consiste en l'implémentation de la fonction **Mix**, qui permet de transformer une musique(<music>) en une liste d'échantillons(<samples>) utilisé par la suite pour produire un fichier audio au format **.wav**. La fonction **Mix** prend une liste de morceaux (<parts>) détaillé dans la section **Mix**(**??**)

Ce projet comprend plusieurs fichiers oz pour les sources, les tests, un main ainsi que des exemples. Il contient également un Makefile pour automatiser l'exécution, ainsi que des fichiers .wav utilisé dans nos exemples. Le fichier plot_wave.py permet de visualiser un fichier .wav et finalement le projet comprend le rapport PDF.

La partie **PartitionToTimedList** a été fait par David et la partie **Mix** ainsi que les extension on été fait par Vincent.

2 Limitations et problèmes connus du programme

Malheureusement nous avons eu des difficultés pour la partie **PartitionToTimedList**, avec quelques fonctions qui ne fonctionnent pas correctement. Nous n'avons pas su effectuer les tests convenablement pour cette même partie. En revanche avec ce que nous avons nous pouvons utiliser le mix et donc faire la plus grosse partie du projet

3 Constructions non-déclaratives utilisez

Nous n'utilisons que les exceptions (try/catch) afin de vérifié que lorsque l'on lis un fichier il n'y pas eu de problème comme par exemple ; fichiers non trouvé ou permission insatisfaisante. On peut retrouvé ceci dans la fonction **Wave** du fichier **Mix.oz**

4 Choix d'implémentation surprenant

Comme demander **make run** permet de lancer le projet avec la musique du fichier **exemple.dj.oz**.

En revanche il est aussi possible de lancer des exemple avec **make exemple** utilisant la musique du fichier **exemple.dj.oz** et plusieurs autres musiques du fichier **Expl_mixing.oz**.

L'utilisation des extensions nécessite l'argument "–extension" lors de la compilation pour tester ceci faite **make extension** ce qui lance le projet avec une variable booléen laissant passer certaines fonctions de mix (reverse,crossfade,muffle) ce qui est la raison pour laquelle **make exemple** donne par exemple pour **out_crossfade.wav** une music vide hors en utilisant les extensions **make extension** alors **out_crossfade.wav** utilise bien la fonction crossfade de mix comme demander dans la partie Extensions du projet. Il va aussi lancer l'extension Créativité utilisant la musique du fichier **bonus/creation.dj.wav** et donner l'audio **bonus/creation.wav**

5 Extensions apporté

Pour les extension nous avons décidé de faire les **Effets Complexes** et **Creation** en ajoutant 3 nouveau filtres au Mix :

`reverse(⟨music⟩)` : permet d'inverser une music.

`crossfade(seconds:⟨duration⟩ ⟨music⟩1 ⟨music⟩2)` : permet une transition fluide entre deux music.

`muffle(start:⟨duration⟩ finish:⟨duration⟩ intensity:⟨factor⟩ ⟨music⟩` : Permet d'ajuster l'intensité entre le temps start et finish

Ainsi que la création d'une petite musique dans **bonus/creation.dj.wav** qui crée l'audio **bonus/creation.wav**

Pour activer les extension et lancer les exemples il faut compiler le projet en utilisant :

```
>make extension
```