Ecole Polytechnique de Louvain La Neuve

Rapport final

Projet OZ : Maestroz

Groupe 68 :

Bourez Nicolas
28462000 / nicolas.bourez@student.uclouvain.be

Denis Kervin
26121900 / kervin.denis@student.uclouvain.be

Année 2021-2022
LEPL1503

1 Problèmes connus dans le programme

On a remarqué que Oz contenait quelques imperfections par rapport à sa façon de gérer les floats. Dès lors, si on met en argument de la transformation stretch, une fraction (par exemple $\frac{1}{3}$), notre programme aura des difficultées à gérer cette transformation. Par contre, si on passe comme argument 0.3333 alors il n'y aura aucun problème (on perd juste un peu de précision).

2 Justification des constructions non déclaratives

Au début, nous utilisions des celulles pour pouvoir redéfinir des variables. Cela avait pour avantage de rendre notre code plus lisible pour quelqu'un qui essayerait de comprendre comment certaines fonctions s'exécutent (notamment la fonction `Transpose`). Cependant, on compensait ce gain en lisibilité par de l'espace supplémentaire en mémoire(minime mais présent). Puisqu'Inginious n'acceptait que la programmation déclarative, nous avons décidé de changer notre code pour quelque chose de plus complexe à comprendre. On a pu dès lors recevoir un feedback d'Inginious.

3 Choix d'implementation

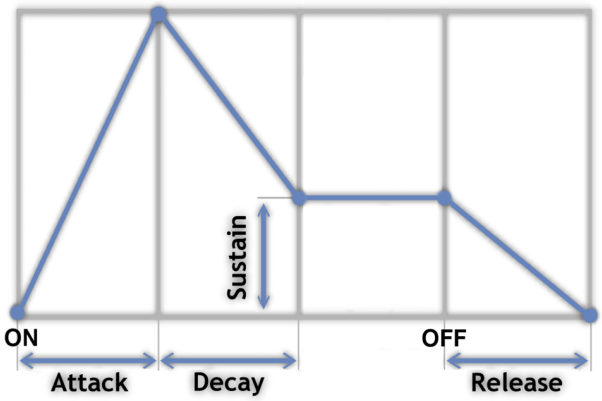
Il n'y a rien de très surprenant dans le code. Les seules choses assez insolites sont nos approximations des floats notamment dans les fonctions `ComplexeLoop`,`BasicLoop` et `HardLoop`. Cela nous permet de pouvoir mieux comparer les floats (un simple == nous donnait une boucle infinie et donc un excès de mémoire).

4 Extentions

Nous avons fait le choix de deux extensions : le lissage et les instruments. Le choix de la première parraissait évident puisqu'il a pour résultat final de sublimer le son. Le choix d' une deuxième vient du fait qu'il nous donne vraiment un panel de créativité pour pouvoir faire notre création finale.
Pour activer les extensions, il suffit de mettre la variable `VerboseExtension` à `true`.
NB : l'activation de nos extensions risquent de ne plus faire passer certains tests notamment celui sur l'échantillonage. Cela vient du fait que nous appliquons le filtre directement dans la fonction échantillon.
Les extensions sont démontrées dans la créativité.

4.1 Lissage

Comme nous l'avons remarqué lors de l'implementation, la synthese des notes bruits desagreables entre chaque note. Pour palier ce probleme, il faut adoucir le debut et la fin des notes. En premier lieu nous avons décidé de réaliser une enveloppe de type "trapèze". Elle fonctionne très bien et on a juste à réutiliser la fonction fade pour pouvoir l'appliquer. Soit N la longueur d'une note, nous avons choisi d'appliquer la croissance de l'instensité sur les $\frac{N}{10}$ première secondes et les $\frac{N}{10}$ dernières. C'est un choix purement arbitraire mais il marche. Nous avons ensuite voulu modeliser une enveloppe sonore plus complexe, plus particulièrement l'enveloppe ADSR, mais on est malheureusement tombé à cours de temps.



4.2 Instrument

Cette extension consiste en premier lieu :

- à ajouter un instrument à une note. La fonction `PartitionToTimedList` prend en charge une transfromation supplémentaire : `instrument(name:<instrument> <Partition>)`. Pour pouvoir appliquer cette transformation, 3 nouvelles fonctions ont été crées : `Instru`,`ChargeInstru`, `ChargeChordInstru`. La première consiste juste à appliquer les deux suivantes sur les notes de la partition. La deuxième se charge de changer l'instrument de la note tandis que la troisième applique `ChargeInstru` à chaque note de l'accord.

En deuxième lieu :

- à lier la note à son fichier wave correspondant. Ceci est fait directement à partir de la fonction `Echantillons`, dans laquelle on a encore accès à toutes les notes. On applique ensuite la fonction `SetInstrument` aux notes et `SetChord` aux accords. La fonction `SetInstrument` va adapter la durée du fichier à la longueur de la note et va appliquer un fade à celle-ci. Nous avons choisi d'appliquer l'évolution linéaire de l'intensité sur les 5 premiers et 5 derniers centièmes de secondes de chaque note. C'est un choix purement arbitraire mais celui-ci ne devait pas être trop long car des notes jouées en dessous de 10 centièmes de secondes n'auraient pas été entendues. La fonction `SetChor` va simplement sommer les enchantillons de chaque note du chord et va prendre leur moyenne.