Projet d'informatique : rapport final

I Limitation et problème du programme

L'utilisation de mémoire est assez conséquente, pour sortir notre fichier de 2.5MB nous avons un pique d'utilisation de la mémoire RAM atteignant 900MB. Cela signifie donc que nous gardons beaucoup d'informations inutile ou répetitive en mémoire.

II Utilisation de constructions non-déclaratives

Nous ne sommes pas sortis du cadre fonctionnelle pour la réalisation du programme.

III Implémentation surprenante

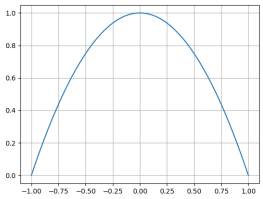
Rien à signaler.

IV Complexité

- Duration : Notre fonction "Duration" parcours une fois la liste contenant les notes et calcule le ratio à appliquer à la durée de chaque note grâce à la formule $\frac{\text{Nbre de secondes souhaitée}}{\text{Nbre de secondes totale}}$. Elle fait appelle à la fonction "Stretch" qui applique ce ratio a chaque durée. Elle ne parcours la liste qu'une seule fois. Si nous appelons "Duration" avec une liste de taille 2n, cela prendra 2 fois plus de temps que pour n, nous avons donc une complexité n.
- Merge : Notre fonction "Merge" utilise une sous fonction qui effectue la somme de l'accumulateur et de la liste suivante multiplié par son facteur(qui correspond à l'intensité voulue de cette musique). Notre nombre d'actions est linéairement dépendant à la taille des différentes listes ainsi qu'à leur nombre.
- Loop : Notre fonction "Loop" cherche d'abord à savoir si la durée demandée est plus petite que la durée de la liste de samples. Si c'est le cas, nous utilisons "List.take" afin de prendre le nombre d'éléments demandé. Si la durée demandée excède la durée (en secondes) de la liste de samples, nous utilisons la fonction "Repeat" qui assemble une liste composée de copie de la liste de samples en utilisant l'opérateur 'l'. Nous utilisons en suite "List.take" afin d'obtenir la partie tronquée de la fonction pour atteindre la durée désirée. Dans le premier cas, la complexité est linéairement dépendante à la durée. Dans le second cas,

V Extensions

- Fonctions complexes :
  - echo(delay :⟨duration⟩ decay :⟨factor⟩ repeat :⟨natural⟩ ⟨music⟩)
    Un filtre d'écho qui permet d'enchainer plusieurs écho. Chaque nouvel écho est un écho du précédent.
  - crossfade(seconds :⟨duration⟩ ⟨music⟩ ⟨music⟩)
    Ce filtre superpose un fondu sortant (fade-out) sur la première musique et un fondu entrant (fade-in) sur la seconde musique.
  - lowpass(⟨music⟩)
    Ce filtre filtre passe-bas permet de laisser passer les basses fréquences et d'atténuer les hautes fréquences, c'est-à-dire les fréquences supérieures à la fréquence de coupure, qui das notre cas vaut 5000Hz.
- Lissage : La fonction lissage permet de lisser une liste de samples qui lui est donnée. Cette fonction applique un lissage correspondant à la fonction : $-x^2 + 1$, de [-1,0] pour le lissage d'entrée. Et de [0,1] pour le lissage de sortie.



Fonction $-x^2 + 1$

Il est commandé par un boolean se trouvant dans la fonction Mix afin de pouvoir exécuter le programme normalement lors des tests. Le boolean en question a pour non "Lissage".

- Instruments : L'implémentation des instruments a été commencée. Il est possible de modifier l'instrument voulu, mais ceci n'a pas d'impact pour la suite car, ce n'est pas pris en charge dans notre fonction Mix.