UFR 919 Informatique – Master Informatique

Spécialité STL – UE 5I553 – PPC

TME 12 — Scala

Carlos Agon

0 Install

1. Installer sbt, soit via le web ou en chargeant le fichier `sbt.zip`.
2. Charger le fichier `scalaHelloWorld.zip` et tester le projet.

1 Représentations musicales en SCALA

Pour permettre des opérations sur la musique au niveau symbolique, nous devons commencer par représenter ce qu'est une musique. Nous allons diviser les objets musicaux en deux types : simples ou composés. Les objets simples sont des notes ou des silences. Une note est définie par une hauteur (entier entre 0 et 127, voir le codage midi), une durée (exprimée en millièmes de seconde) et un volume (entier entre 0 et 127). Les objets composés, à leur tour, sont constitués d'une liste d'autres objets musicaux ordonnés en sequence (e.g. une melodie) ou en parallèle (e.g. un accord).

Question 1 Charger le fichier TME12_code.zip et dans le projet TME12 inspecter les diverses `case classes` décrites dans le paragraphe précédent et en se basant sur ce type créez une variable `exemple` reprxtensentant l'objet de la Figure 1.

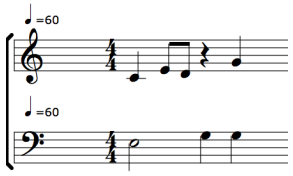

FIGURE 1 – Un objet musical

Question 2 Par la suite, nos aurons besoin d'outils nous permettant d'inspecter et modifier nos objets musicaux. Tester la function `duration` qui rend la durée de n'importe quel objet musical avec la variable exemple (e.g. duration exemple = 4000).

Question 3 Pour écouter nos objets musicaux nous utilisons la function play. En lisant le code, vous aller remarquer l'instruction :

```
remote ! MidiNote(p,v,d,at)
```

qui envoie un message un l'acteur instrument. Cet acteur est créé en executant le programme PLAYER.

Tester le player en écrivant la méthode receive du BachActor ainsi :

```
def receive =  case "START" => { play(exemple) }
```

Lisez et tester les autres fonctions données dans le code (i.e. copy, note_count, stretch).

NOTE : avant de faire run dans le projet PLAYER, executer la command suivante :
`set fork in run := true`

Question 4 La figure 2 montre le groupe des transformations classiques appliquées en musique. Inspecter et tester les fonctions `transpose`, `retrograde` et `mirror` qui implémentent ces transformations.

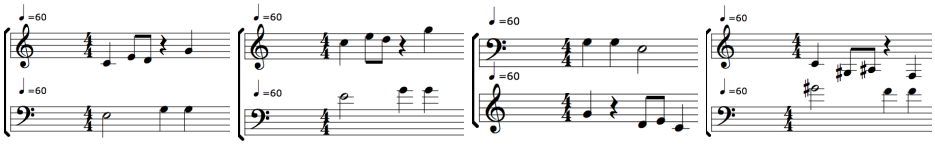

FIGURE 2 – Identité ; Transposition (12 tons) ; Retrograde ; Mirroir (centre = 60)

D'autres fonctions vous seront utiles pour générer des séquences musicales plus complexes. Implementer les fonctions :

— `repeat` qui prend un objet musical et le concatène avec lui même n fois.
— `canon` qui prend un objet musical et le met en parallèle avec lui même, avec un decalage de n millièmes de seconde.
— `concat_mo` qui concatène deux objet musicaux l'un après l'autre.

Créer une variable `exemple2` pour la figure 3.

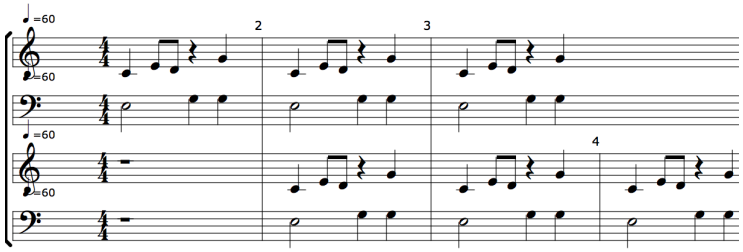

FIGURE 3 – canon (repeat exemple 3) 1000

Question 5 Nous allons vérifier notre implémentation en écoutant le fameux Canon par tons ou Canon perpetuel (BWV 1079) de Johann Sebastian Bach que vous pouvez écouter et comprendre à l'adresse

`www.youtube.com/watch?v=A41CITk85jk`

Le principe de base est le suivant :

Vous disposez d'une voix (le thème du roi, figure 4) que vous allez répéter 6 fois l'une apres l'autre, mais à chaque répétition vous allez faire une transposition d'un ton (2 unités midi).

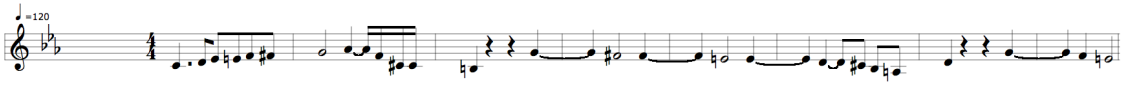

FIGURE 4 – Première voix

La voix du roi est accompagnée par un canon à deux voix dont le thème (en do mineur) est presenté dans la figure 5.

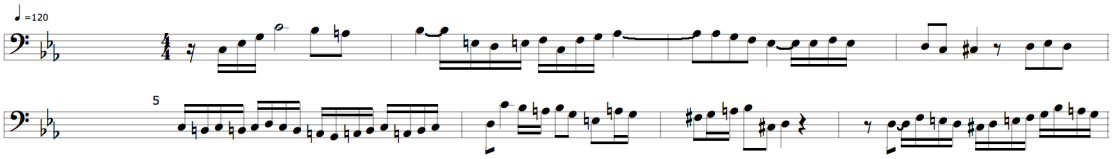

FIGURE 5 – Deuxième voix

Cette voix est aussi repetée 6 fois et, à chaque fois, transposée d'un ton. La troisième voix que l'on répète 6 fois (en transposant d'un ton à chaque fois) rentre une mesure après (2000 ms) et elle est identique à la deuxième voix mais transposée d'une quinte (7 unités midi).

Le résultat final est montré dans la figure 6

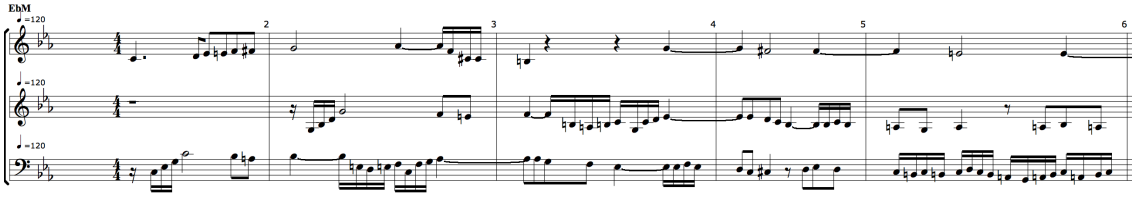

FIGURE 6 – Le résultat final

Ecrivez la fonction `canon_bach` qui construit ce canon.