Rapport sur MaestrOZ, LINFO1104

Balabka Igor : 00611900
Pihet Henri : 66151900

2 mai 2022

1 Limitations et limites du programmes

1. L'input ne doit pas être trop long car cela utilise trop de mémoire.
2. L'input ne doit pas être une liste dans une liste. Exemple: [[A] [B C]]. Avec cette liste là, notre programme ne fonctionnera pas.
   Pour remédier à ce problème, on a crée une fonction "Flatten". Elle permet de prendre en input une liste dans une liste et retourne une simple liste. Reprenons l'exemple ci-dessus: [[A] [B C]]. En passant la liste comme argument, notre fonction "Flatten" renverra [A B C]. Prenons un autre exemple, [[A] [B[C]]]. La fonction "Flatten" donnera comme output [A B [C]]
3. Notre code n'est pas le plus esthétique. On aurait dû utiliser plus de fonction de base comme "Map" afin de le rendre plus lisible et plus simple à comprendre.
4. Notre programme ne contient aucune extension.

2 Les fonctions non déclaratives

Nous avons décider de ne pas utiliser de fonctions déclaratives. Toutes nos fonctions essayent d'utiliser du pattern matching quand c'est possible.Nous n'avons pas implémenter de fonction non-déclaratives parce que nous voulions toujours savoir l'output de sortie et ne pas permettre au programme de choisir l'output. Les fonctions déclaratives sont compliquées à débugger et ne permettent pas de savoir si toutes nos fonctions sont correctes.

3 Les implémentations surprenantes

1. Notre fonction "NodeShifter" est une fonction beaucoup trop longue. Elle fait presque 100 lignes de code. Le seul point positif est qu'elle est simple à comprendre mais n'est pas du tout optimale

```
fun {NoteShifter N I} %L=list extended I=float, shift partition by I semitone, C C# D D# E F F# G G# A A# B
   if I<0.0 then
      case N
      of note(name:Name octave:Octave sharp:Sharp duration:Duration instrument:none) then
         if Name == b then
            {NoteShifter note(name:a octave:Octave sharp:true duration:Duration instrument:none) I+1.0}
         elseif Name == a then
            if Sharp == true then
               note(name:Name octave:Octave sharp:false duration:Duration instrument:none)
            else
               {NoteShifter note(name:g octave:Octave sharp:true duration:Duration instrument:none) I+1.0}
            end
         elseif Name == g then
            if Sharp == true then
               {NoteShifter note(name:Name octave:Octave sharp:false duration:Duration instrument:none) I+1.0}
            else
               {NoteShifter note(name:f octave:Octave sharp:true duration:Duration instrument:none) I+1.0}
            end
         elseif Name == f then
            if Sharp == true then
               {NoteShifter note(name:Name octave:Octave sharp:false duration:Duration instrument:none) I+1.0}
            else
               {NoteShifter note(name:e octave:Octave sharp:false duration:Duration instrument:none) I+1.0}
            end
         elseif Name == e then
            {NoteShifter note(name:d octave:Octave sharp:true duration:Duration instrument:none) I+1.0}
         elseif Name == d then
            if Sharp == true then
               {NoteShifter note(name:Name octave:Octave sharp:false duration:Duration instrument:none) I+1.0}
            else
               {NoteShifter note(name:c octave:Octave sharp:true duration:Duration instrument:none) I+1.0}
            end
         elseif Name == c then
            if Sharp == true then
               {NoteShifter note(name:Name octave:Octave sharp:false duration:Duration instrument:none) I+1.0}
            else
               {NoteShifter note(name:b octave:Octave-1 sharp:false duration:Duration instrument:none) I+1.0}
            end
         else
            error
         end
      [] nil then nil
      end
   elseif I>0.0 then
      case N
      of note(name:Name octave:Octave sharp:Sharp duration:Duration instrument:none) then
         if Name == b then
            {NoteShifter note(name:c octave:Octave+1 sharp:false duration:Duration instrument:none) I-1.0}
         elseif Name == a then
            if Sharp == true then
               {NoteShifter note(name:b octave:Octave sharp:false duration:Duration instrument:none) I-1.0}
            else
               {NoteShifter note(name:Name octave:Octave sharp:true duration:Duration instrument:none) I-1.0}
            end
         elseif Name == g then
            if Sharp == true then
               {NoteShifter note(name:a octave:Octave sharp:false duration:Duration instrument:none) I-1.0}
            else
               {NoteShifter note(name:Name octave:Octave sharp:true duration:Duration instrument:none) I-1.0}
            end
         elseif Name == f then
            if Sharp == true then
               {NoteShifter note(name:g octave:Octave sharp:false duration:Duration instrument:none) I-1.0}
            else
               {NoteShifter note(name:Name octave:Octave sharp:true duration:Duration instrument:none) I-1.0}
            end
         elseif Name == e then
            {NoteShifter note(name:f octave:Octave sharp:true duration:Duration instrument:none) I-1.0}
         elseif Name == d then
            if Sharp == true then
               {NoteShifter note(name:e octave:Octave sharp:false duration:Duration instrument:none) I-1.0}
            else
               {NoteShifter note(name:Name octave:Octave sharp:true duration:Duration instrument:none) I-1.0}
            end
         elseif Name == c then
            if Sharp == true then
               {NoteShifter note(name:d octave:Octave sharp:false duration:Duration instrument:none) I-1.0}
            else
               {NoteShifter note(name:Name octave:Octave sharp:true duration:Duration instrument:none) I-1.0}
            end
         else
            error
         end
      [] nil then nil
      end
   else
      N
   end
end
```

2. La fonction "Loop" est une fonction compliquée. On a du mal à la comprendre mais en l'ayant testée, elle nous donne le résultat attendu. Etant donné que la fonction est correct, on a décidé de ne pas la changer pour éviter d'avoir une fonction qui bug

```
fun {Loop Duration Music Acc}
    local Nofl in
        %NumberOfLoop
        Nofl = Duration/({IntToFloat {List.lenght Music}}/44100.0) %In float
        if (Nofl =< 1.0) then
            {Cut 0.0 Duration Music 0.0}
        else %Ex if Nofl = 4.67 => 4*Repeat + Cut 0.0 Duration-4
            if (Nofl - {IntToFloat {FloatToInt Nofl}} >= 0.0) then %53.3-53 => 53 repeat and cut form 0.0 to 0.3
                {Append {Repeat {FloatToInt Nofl} Music} {Cut 0.0 (Duration-({IntToFloat {FloatToInt Nofl}}*({IntToFloat {List.lenght Music}}/44100.0)))
                Music 0.0}}
            else %Ex if Nofl = 53.99 => 54-1 repeat and cut from 0.0 to 53.99 - (54-1)
                {Append {Repeat {FloatToInt Nofl} Music} {Cut 0.0 (Duration-({IntToFloat {FloatToInt Nofl}-1}*({IntToFloat {List.lenght Music}}/44100.0)))
                Music 0.0}}
            end
        end
    end

end
```

3. La fonction "PartitionToTimedList" appel une fonction auxiliaire qui elle à plus d'argument. Malheureusement ces arguments ne sont plus nécessaires mais comme on ne veut pas créer de nouveaux bugs, on les laisse. Nous n'avons pas envie de toucher ce qui fonctionne comme il faut.

```
fun {PartitionToTimedList L}
    {PartitionToTimedListAux L 1.0 1.0}
end

fun {PartitionToTimedListAux L T E} %L=List (Partiton) T=Time of the note (T=1.0 means that the note will last 1.0 sec)  E=stretch factor (ex: if T=1.5 an
    case L
    of H1|T1 then
        case H1

        of H2|T2 then
            %({Append {PartitionToTimedListAux H2 T E} {PartitionToTimedListAux T2 T E}})|{PartitionToTimedListAux T1 T E} %Need to make a proper ChrodToT
            {ChrodToTimedList H1}|{PartitionToTimedListAux T1 1.0 1.0} %much simpler
        [] silence(duration:R) then
            H1|{PartitionToTimedListAux T1 T E} %pass silence to Mix, Mix pass silence to SilenceToAi, SilenceToAi make 0.0 sound
        [] nil then
            {PartitionToTimedListAux T1 T E}
        [] duration(1:P seconds:R) then
            {Append {SetDuration {PartitionToTimedListAux P T E} R} {PartitionToTimedListAux T1 T E}}
            %{Append {PartitionToTimedListAux P (R/{IntToFloat {Lenght P}})*E E} {PartitionToTimedListAux T1 T E}}%Need to make a proper DurationToTimedLi
        [] stretch(1:P factor:R) then
            %{Browse P}
            {Append {Stretch {PartitionToTimedListAux P T E} R} {PartitionToTimedListAux T1 T E}}
            %{Append {PartitionToTimedListAux P T*R R} {PartitionToTimedListAux T1 T E}} %Need to make a proper Stretch func
        [] drone(note:Nc amount:N) then
            {Append {Drone {PartitionToTimedListAux Nc T E} N 0} {PartitionToTimedListAux T1 T E}}
        [] transpose(1:P semitones:I) then
            {Append {TransT {PartitionToTimedListAux P T E} I 0} {PartitionToTimedListAux T1 T E}}
        [] Atom then
            {NoteToExtended H1 T}|{PartitionToTimedListAux T1 T E}

        else
            H1|{PartitionToTimedListAux T1 T E} %for robustness
        end
    else
        nil
    end
end
```

4. Pour la fonction "Fade" nous avons eu du mal à implémenter la fondue à la fin de la musique. Pour cela, on a inversé les échantillons de la musique afin de faire un fondu sur la fin de la musique avec l'aide de la fonction FadeIn.

```
fun {FadeIn Factor Music Acc}
    case Music of H|T then
        if (Acc =< 1.0) then
            H*Acc | {FadeIn Factor T Acc+Factor}
        else
            music
        end
    else
        nil
    end
end

fun {Fade In Out Music} %if 5 sample to fade then 1/5 = 0.2 => factors = [0.8 0.6 0.4 0.2 0.0]
    if (In > 0.0) then
        {FadeIn 1.0/In*44100.0 Music 0.0}
    elseif (Out > 0.0) then
        {Reverse {FadeIn 1.0/In*44100.0 {Reverse Music} 0.0}} %So dirty but no time to do it the proper way
    else
        Music
    end
end
```

4 Les extensions implémentées

Nous n'avons pas implémenter d'extensions. Nous avons préférer réussir d'autre projet en priorité plutôt que celui là. Nous avons commencer à travailler un peu tard et nous sommes vite arrivé à cours de temps. Nous avons essayer de réaliser une base solide plutôt que d'avoir un "PartitionToTimeListed" et un "Mix" qui ne fonctionne qu'une fois sur deux.