IN104 - Rapport intermédiaire

Transcription automatique de la musique de piano

Groupe 10 : Tadeusz Plagué-Makowiecki , Alexandre Dréan

1. Decomposition du projet

```
Main(){
  //Entrer un fichier audio en ligne de commande

  //1. Appeler la fonction piano_audio_to_piano_note qui
renvoie : une liste de temps auxquels sont joués des notes et
une liste des notes en questions

  //2. Appeler la fonction piano_notes_to_video qui génère une
fenêtre SDL et joue le son ainsi que les notes qui s'activent
successivement sur le piano.
}
```

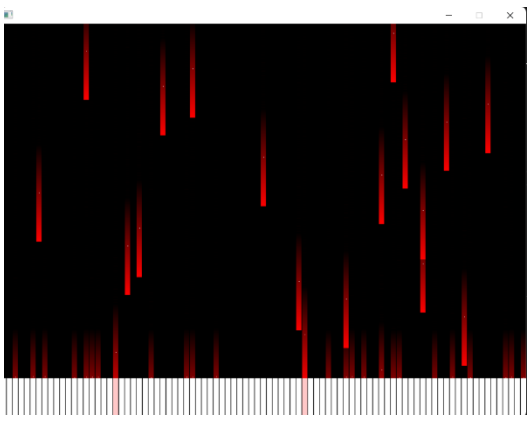
2. Avancement

- `piano_notes_to_video` :

Cette fonction a été réalisée en tout premier lieu. Elle a été testé en générant des listes de temps associés avec des listes de notes aléatoires. Temporairement, la fonction n'est pas définie dans des fichiers séparés : elle est pour l'instant définie dans le fichier `main` avec toutes les fonctions intermédiaires qui servent à son fonctionnement. Nous avons mis de côté les éventuelles améliorations possibles de cette fonction pour nous concentrer sur la lecture audio, mais nous n'excluons pas la possiblilité de revenir sur cette fonction plus tard éventuellement pour rajouter des éléments personnalisables (couleurs des notes en fonctions de leur fréquence, nombre de notes, longueur d'une animation de note, etc...)

.

Le programme peut être testé dans le fichier bin/prog.exe



- `piano_audio_to_piano_note` :

Cette fonction est en cours de développement pour la lecture d'une seule note à la fois. Elle est décomposée en plusieurs sous-fonctions contenues dans les fichiers suivants:

(i) `lect_audio.c`

Contient les fonctions pour lire l'audio WAVE et renvoyé le nombre d'échantillons, un tableau des temps d'échantillonage et un tableau des amplitudes en dB. Ces fonctions s'inspirent du programme fournit par l'université de Nice pour lire des fichiers WAVE. Elles ont toutes été testés et comparé avec des fonctions effectuant la même lecture en Matlab et elles ont donnés des résulats concluants.

(ii) `Fenetrage_hamming.c`

Contient la fonction permettant de renvoyé une nouvelle liste d'amplitude qui a été fenêtré entre des temps t1 et t2 avec un fenêtrage de hamming.

(iii) `Transformee_Fourier.c`

Contient la fonction qui permet de calculer la transformée de fourrier rapide d'un vecteur en utilisant la bibliothèque GSL. Cette fonction a été comparé avec la fonction fft de Matlab et elle fournit les mêmes résultats.

(iv) `Freq_preponderante.c`

Fonction qui calcule la fréquence prépondérante dans un spectre en utlisant la formule $argmax \log \prod_{i=1}^{H} |X(if)|^2$ sachant que cette fréquence doit être supérieur à un certain seuil pour être considéré comme une note.

Remarque : A l'issue de cette dernière séance nous avons mis toutes les fonctions des fichiers .c dans le fichier main.c avant la fonction main. Cela est plus simple pour nous afin de tester les fonctions sans avoir de problème causé par #include «fichier.c ». A la fin du projet nous optimiserons peut-être la compilation en créant des fichiers .h.

3. Problèmes rencontrés

Dans la fonction `piano_audio_to_piano_note`, afin de pouvoir trouver les fréquences prépondérantes à différents instants, on créer un boucle de temps avec un

fenêtre de temps qui se déplace [t1,t2] = [0, 0.2] puis [0.2,0.4] etc... Et pour chaque tour de boucle on récupère un tableau d'amplitude fenêtrée en utilisant le fenêtrage de hamming, ce tableau doit ensuite passé par la fonction transformée de fourrier puis par la fonction Freq_preponderante, ces dernières fonctions prennent trop de temps à s'exécuter et elles fournissent quasiment toujours une fréquence fausse. Nous sommes donc en phase de deboggage. Une solution retenue pour améliorer le temps de calcul était de calculer les transformée de fourrier à partir de tableau d'amplitude en short au lieu de float. Le calcul se fait alors en un temps bien plus raisonnable mais les résulats sont encore erronés.

4. Ce qui reste à faire

Lorsque la fonction `piano_audio_to_piano_note` aura été corrigée. Nous serons alors capable de calculer les notes prépondérantes à chacuns des intervalles de temps : [0, 0.2], [0.2,0.4] . On pourrait alors détecter celle qui est la plus forte (en terme d'amplitude) avec la méthode qui précède, puis on pourrait filtré la fréquence lié à cette note dans notre signal puis voir si une autre note est joué en simultané et recommencé tant qu'on détecte des note sur le spectre de cette fenêtre.

Des eventuelles modification de `piano_notes_to_video` pourraient être nécessaire pour ajuster la longueur de l'animation d'une note en fonction de la longueur d'une note (au sens du temps où le doigt reste appuyé sur cette note)

5. Conclusion

Le plus dure reste de débogger la fonction `piano_audio_to_piano_note` puis de créer une fonction qui filtre une fréquence. Il faudra aussi qu'on ré-orgagnise le code (i.e : séparer `piano_notes_to_video` du fichier main et homogénéiser le code (tabulation, position des accolades, etc..))