Projet de C++ - Transform audio in visual effects

Louis de Charsonville

22 février 2013

Résumé

Cette courte note présente dans un premier temps l'objectif du projet informatique, puis la démarche (notamment mathématique) et enfin quelques mots plus spécifiques sur le code.

1 Objectifs du projet

De plus en plus d'artistes associent des effets visuels à la musique qu'ils jouent lors de performances artistiques (concerts, festivals, exposition). Parmi ses effets visuels, la technique du mapping, qui consiste à projeter des images sur des structures en trois dimensions, se distingue et devient de plus en plus courante. Parmi, les artistes les plus connus, on peut notamment citer Etienne de Crécy, Amon Tobin ou Cut Killer (campagne Adidas All In sur le palais du Pharo).

Bien plus modestement, ce projet se propose d'analyser le son en entrée et de générer des images "en lien" avec ce son et pouvant, être projetées sur une structure en 3D. L'analyse du son se fait en instantané et ne se fonde donc que sur l'intensité sonore mesurée (par un micro) à un instant t (et non pas sur l'analyse d'un fichier son). L'objectif de se projet est de déterminer le rythme de la musique pour l'utiliser lors de la génération de formes géométriques extrêmement simples liées à ce rythme.

2 La démarche

2.1 L'analyse sonore

L'analyse sonore se fonde sur l'intensité sonore mesurée par un micro. Afin de trouver le rythme d'une musique, plusieurs méthodes sont possibles : l'analyse via l'intensité : une "beat" est plus intense que le reste de la musique. En détectant les crêtes du signal, on peut mesurer le rythme d'une musique. Néanmoins, cette méthode n'est pas très satisfaisante car elle est ne différencie pas

les sons, un accord de guitare peut avoir la même intensité qu'une grosse caisse de batterie mais pas la même place dans le rythme d'une musique.

L'algorithme utilisé ici permet une approche fréquentielle de la musique, permettant de distinguer les graves des aigus, etc. Depuis les travaux de Joseph Fourier, on sait qu'une onde périodique se décompose de manière unique en une série d'harmoniques. C'est la méthode choisie ici afin d'obtenir le spectre de fréquence du son en entrée. Grâce à la transformée de Fourier, on obtient une décomposition en fréquence du son en entrée. L'obtention du spectre de fréquence en temps réel constitue la majeur partie du travail présenté ici, la décomposition en fréquence étant ensuite utilisée pour des animations graphiques (comme dans un equalizer par exemple).

L'analyse sonore procède comme suit : prélèvement de 1024 samples par 43ième de seconde, l'intensité sonore est donc discrétisée. (sample rate = 44100/s). On mesure la fréquence de ces 1024 samples et on obtient donc le spectre sonore réparti 1024 fréquences. L'algorithme mesure 43 fois par secondes l'intensité de 1024 fréquences. Ces 1024 fréquences sont ensuite réparties selon 64 sous-bandes de fréquences.

2.2 L'algorithme de Cooley-Tucker

Afin d'obtenir le spectre de fréquence du son, on utilise l'algorithme de Cooley-Tucker, un algorithme de FFT : Fast Fourier Transform (décomposition discrète), très adapté au code (car de complexité temporelle en O(nlog(n)). Cet algorithme est un algorithme récursif fondé sur le principe de diviser pour régner et est relativement simple à implémenter.

2.3 De 1024 fréquences à un equalizer de 64 fréquences

L'algorithme de Cooley)Tucker fournit la décomposition du signal en 1024 fréquences. C'est inutilisable en l'état. Par ailleurs, l'oreille humaine est ainsi faite qu'elle n'accorde pas la même puissance à chaque fréquence. Par ailleurs le spectre va de 40Hz à 20000Hz (fréquence inaudible), il convient donc de rassembler de manière "intelligente" et rapide les fréquences entre elles. Nous avons choisi ici de répartir ces 1024 fréquences en 64 sous-bandes de fréquences selon une méthode pseudo-algorithmique (voir la section le code).

2.4 Les effets visuels

Les effets visuels sont extrêmement basiques et épurés. L'objectif de ce projet de C++ étant plus l'utilisation des concepts inhérents à ce langage que la conception graphiste, nous avons choisit de restreindre ces effets à trois : un equalizer (en réalité deux : un à 64 fréquences et l'autre à 32), un détecteur de beat (très utile pour la projection mapping) et une animation où des cercles bougent en rythme avec la musique.

3 Le code

3.1 Avant-propos sur openframeworks

Ce projet utilise la librairie open source openframeworks. openframeworks est un gestionnaire d'interface qui fonctionnent grâce à des boucles infinies et des méthodes pré-implémentées très intuitives : dans ce projet nous n'en utilisons que cinq :

- setup : qui initialise les différents attributs
- update : qui permet la mise à jour des attributs
- draw : qui dessine les formes
- audioIn : qui récupère le son de(s) entrée(s) audio
- keyPressed : qui réagit à la pression d'une touche

3.2 Les différentes classes

Le code se divise en trois parties : l'analyse mathématique, la post-production et les effets visuels.

L'analyse mathématique est caractérisée par deux classes : la classe FFT qui implémente l'algorithme de Cooley Tucker et la classe complexe. La faible complexité temporelle de cet algorithme est due à la taille de l'échantillon qui est une puissance de 2. L'algorithme de Cooley Tucker prend un vecteur complexe en entrée avec une taille de 1024 et donne en sortie un complexe avec les valeurs des 1024 fréquences.

La post production est faite au sein de la classe equalizer qui récupère ce vecteur et le transforme en un equalizer. La répartition des 1024 fréquences en 64 sous bandes se fait selon une formule mathématique très simple : les 5 premières sous-bandes récupèrent deux fréquences, les 5 suivantes trois, les 5 suivantes, quatre, etc. Ainsi on obtient une répartition pseudo-logarithmique proche de celle de la perception de l'oreille qui perçoit très mal les hautes fréquences. L'equalizer stocke également en mémoire l'historique de l'intensité pour chaque fréquence pendant une seconde. Ainsi l'equalizer est en mesure de détecter une beat, ie la crête du signal sur cette fréquence.

Les effets visuels utilisent le spectre de l'equalizer en paramètre et le détecteur de beat, le plus souvent pour la taille de l'objet : l'objet devient de plus en plus gros si la fréquence est très présente. Il y a trois effets : la représentation du spectre fréquentiel, une méthode (BrokenLines) qui affiche des bandes blanches dès qu'une beat est détectée sur une fréquence, plus de beat sont détectées, plus nombreuses sont ces bandes. Cet effet est fait pour être utilisé sur une structure en 3D verticale derrière un DJ en boite de nuit par exemple. La salle devenant éclairée lors des "temps forts" de la musique mixée.

4 Conclusion

Ce projet s'est concentré sur l'analyse spectrale d'un son en entrée et en direct (des analyses d'une très grande qualité sont possibles lorsque l'input est

un fichier son) afin de l'utiliser à des fins graphiques pour des performances visuelles lors de sets (discipline du vjing).

Tout en utilisant les méthodes d'openframeworks, nous avons pu mettre en valeur quelques spécificités de la programmation en C++ : orientée objet et soucieuse d'allouer au mieux la mémoire (à travers l'utilisation de pointeur et de constantes).