École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. (33)2-47-36-14-14
Fax (33)2-47-36-14-22
www.polytech.univ-tours.fr

Département Informatique

Rapport de projet S7

Synthétiser un signal sonore et l'enregistrer dans un fichier son

Auteur(s)

Julien Amoros
[julien.amoros@etu.univ-tours.fr]
Florent Flécheau
[florent.flecheau@etu.univ-tours.fr]

Encadrant(s)

Maxime Martineau
[maxime.martineau@univ-tours.fr]

Polytech Tours
Département DI

Version du 12 octobre 2016

Table des matières

Table des figures

Introduction

Le but de ce projet est de créer un synthétiseur musical. Nous devrons, à partir d'un fichier texte rédigé dans une syntaxe bien particulière, et créer un signal sonore (sinusoïdal, carré ou triangulaire par exemple). Par la suite, on pourrait enregistrer ce signal dans un fichier son. Une autre option à gérer une partition polyphonique afin de générer plusieurs signaux à la fois. Ce sujet nous a assez vite séduit étant donné qu'il concernait la musique, un domaine nous intéressant. Suite à notre rencontre avec notre encadrant, nous avons décidé de réaliser notre projet en suivant la méthode "Agile". Elle consiste à faire une première version du projet, de la présenter à notre client, puis de voir quelles fonctionnalités supplémentaires peuvent être implémentées dans un laps de temps donné, puis de reprendre rendez-vous avec le client, et de recommencer jusqu'à obtenir une version finale de l'application.

Chapitre 1

Itération 1

1.1 Recherche et première approche

1.1.1 Définition des besoins/Spécification

Le but de ce projet est dans un premier temps, de pouvoir créer une musique à partir d'un fichier texte avec une syntaxe similaire à celle utilisée avec Lilypond, c'est à dire de la forme suivante : "a2 a a b4 c" par exemple. Nous vous renvoyons à la documentation Lilypond pour plus de détails. Avec cette notation :

- "a" correspond à la note "la"
- " b" correspond à "si"
- "c" à "do",
- "d" à "ré"
- "e" à "mi"
- "f" à "fa"
- "g" à "sol"

Afin d'exploiter cette notation, nous devons être capable de lire ces lettres, de trouver les notes correspondantes, et ainsi d'y associer leurs fréquences fondamentales. En créant une sinusoïde avec ces fréquences, nous devrions pouvoir émettre une mélodie.

L'idée d'avoir une possibilité de pouvoir manipuler le son (pouvoir modifier la fréquence ou la durée des notes par exemple) a aussi été évoquée.

Une autre piste à développer après avoir le "corps" du projet est de pouvoir lire plusieurs voix différentes à la fois et de les lire en même temps pour s'approcher d'une vraie mélodie.

Dans la finalité de ce projet, nous aimerions également pouvoir générer un fichier audio (en format mp3 par exemple) contenant la partition lue.

1.1.2 Recherches

Pour ce projet, nous avions le choix du langage de programmation, tout en restant de préférence sur un langage assez basique. Afin d'orienter ce choix, nous avons commencé par une phase de recherche.

Dans un premier temps, nous avons eu un peu de mal à nous représenter comment réussir à émettre un signal sonore à partir d'un fichier texte. Nous avons donc regardé les fréquences correspondant aux différentes notes des octaves afin d'avoir une idée un peu plus claire d'un algorithme. Nous avons également consulté la documentation et testé Lilypond. Il s'agit d'une application permettant de créer, de la même façon que notre projet, à partir d'un fichier texte, une partition lisible par n'importe quel musicien, un LaTeX pour la musique. En poursuivant avec Lilypond, nous avons trouvé un éditeur correspondant au format de Lilypond : Frescobaldi. Il contient également un lecteur pour les partitions ainsi créées.

Ensuite, nous avons commencé à rechercher des librairies que nous pouvions utiliser pour traiter le son, et ce dans plusieurs langages : Java et C/C++, ces trois langages étant ceux que nous avons étudié en 3ème année.

Pour le C et le C++, nous avons trouvé 2 librairies assez importantes, OpenALl et FMOD.

Pour Java, nous avons trouvé une librairie permettant de gérer les sons assez facilement, l'API Java Sound.

1.1.3 Choix des librairies et outils

En faisant nos recherches nous avons testé plusieurs librairies. Nous avons réussi à prendre en main l'API java assez vite pour l'utilisation que nous voulions en faire, nous nous sommes donc arrêtés dessus, et elle correspondait à ce dont nous avions besoin. Le fait que nous ayons trouvé une partie de code permettant de générer une sinusoïde pendant nos recherches, que nous pouvions ensuite réutiliser et adapter pour intégrer dans notre propre code nous assurait un gain de temps conséquent et ainsi de pouvoir se concentrer sur des fonctionnalités plus avancées de l'application.

Et donc, par conséquent, nous avons choisi de réaliser notre programme en Java.

1.2 Définition des classes

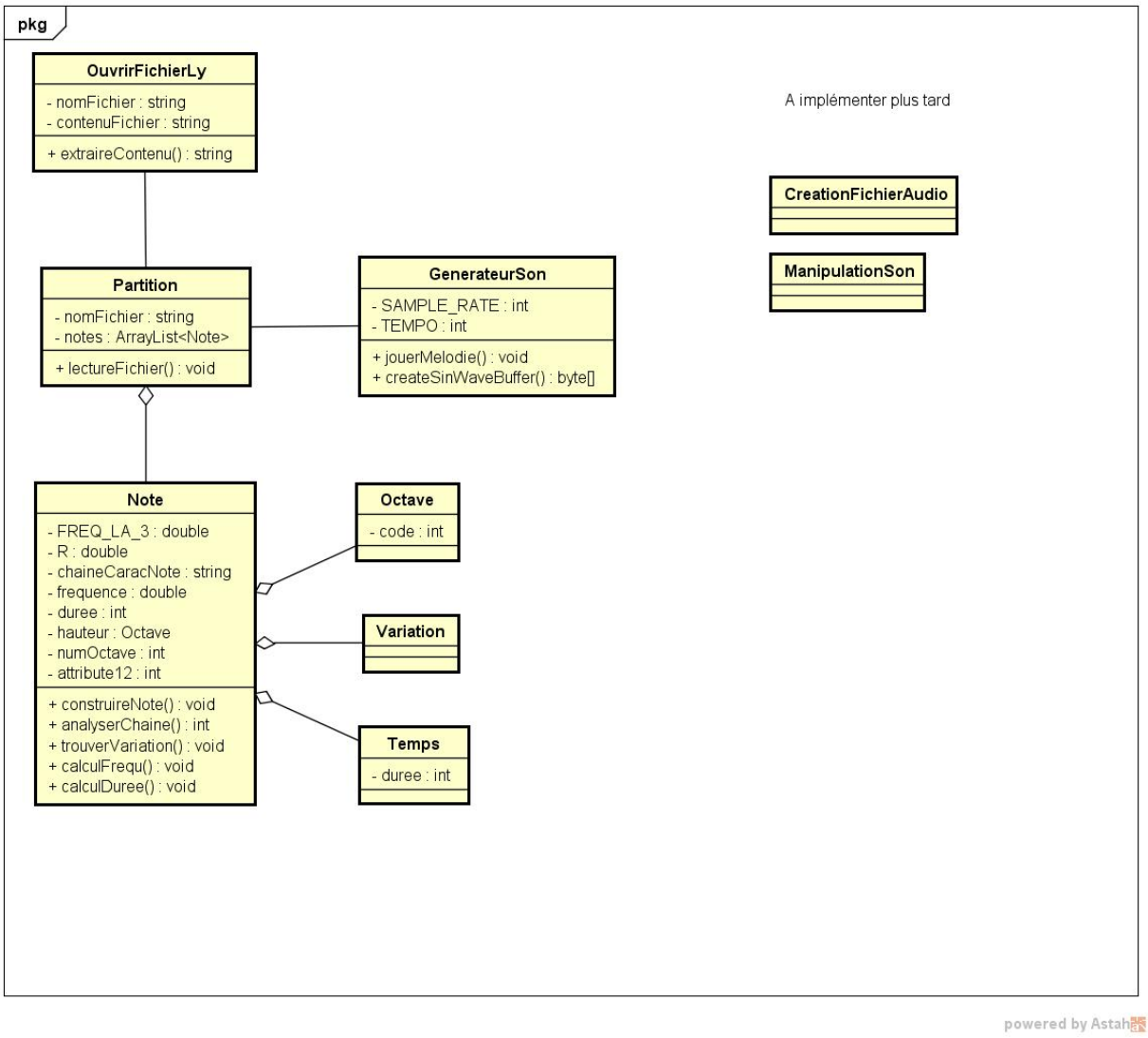

FIGURE 1.1 – Diagramme de classe première version

1.2.1 Classe Partition

Le but de cette classe est de décrire la structure de partition.
Attributs :

- nom du fichier contenant le morceau
- ArrayList de notes contenant la liste des notes à jouer dans l'ordre chronologique

Constructeur : Cette méthode possède un constructeur prenant en paramètre le nom du fichier, qui sera mis dans les attributs. Il lance ensuite la lecture du fichier, en appelant la méthode correspondante.
Accesseurs :

- getNotes() permet de récupérer l'ArrayList des notes.

Méthodes :

- lectureFichier() initialise la structure OuvrirFichierLy, puis récupère la chaîne de caractère qui a été lue dans le fichier, elle la sépare ensuite en fonction des espaces (appel de la

méthode split), puis lance la construction des notes pour enfin les ajouter à l'ArrayList en attribut.

1.2.2 Classe Notes

Le but de cette classe est de construire caractériser une note, c'est à dire de calculer sa fréquence et sa durée en ms à partir de la chaîne contenue dans le fichier texte qui la décris. Attributs :

- Fréquence de la note La (octave 3) a partir de laquelle toutes les autres notes sont calculées
- R, le coefficient de la gamme pythagoricienne, multiplier la fréquence d'une note par cette constante la fait monter d'un demi ton, la diviser par la même constante la fait descendre. Sa valeur exacte est racine douzième de 2, on prends ici une valeur approchée.
- La chaîne de caractère caractéristique de la note (cf notation Lilypond)
- La fréquence de la note (en Hz)
- La durée de la note (en ms)
- La hauteur (do, ré, mi, mi# etc...)
- Le numéro de son octave (3 de base)
- La variation (bémol, dièse ou neutre)

Constructeur : Le constructeur mets la chaîne en paramètre dans les attributs puis initialise la construction de la note qui se déroule en 3 étapes : interprétation de la chaîne de caractère, calcul de la fréquence, et calcul de la durée.
Accesseurs : On peut récupérer dans la structure : la fréquence, la durée et sa hauteur.
Méthodes :

- construireNote() appel successivement analyserChaine(), calculFrequ() (si la note n'est pas un silence) et calculDuree().
- analyserChaine() va tout d'abord regarder la première lettre (= note en notation anglaise, de a à g, avec c = DO, d = RE etc...), si c'est un n, la fonction s'arrête en construisant un silence (frequence de 1, hauteur = NONE...) sinon, calcul du numéro de la note (voir enum Octave), appel de la fonction pour trouver la variation (voir trouverVariation()), modification de numNote si nécessaire enfin modification du numéro de l'octave si nécessaire.
- trouverVariation() retire la première lettre de la chaine (qui correspond à la note qui a déjà été lue) puis recherche dedans les marqueurs des variations ('d' et 'is' pour le dièse, 'b' et 'es' pour le bémol)
- calculFrequ(), prend d'abord comme fréquence de base celle du La3, puis la ramène à la bonne octave en multipliant ou divisant par un multiple de 2, puis multiplie par R le nombre de fois nécessaire pour arriver à la bonne fréquence.
- calculDuree(), détecte le numéro à la fin de la chaine correspondant à la durée de la note.

1.2.3 Classe GenerateurSon

Attributs :

- L'échantillonage (nombre d'évalutaion de la valeur de l'onde sur un temps donné)
- Le tempo (nombre de noires par minutes)

Méthodes :

- jouerMelodie(Partition part) initialise les structure de l'API java.Sound, puis parcours

le tableau de notes de la partition en paramètre de la partition pour générer l'onde correspondante et la jouer sur la sortie standard, enfin, la méthode ferme le flux de sortie.
— createSinWaveBuffer(double freq, int ms) crée un tableau de byte qui caractérise l'onde de fréquence passée en paramètre.

1.2.4 OuvrirFichierLy

Attributs :
— Nom du fichier contenant la description du morceau
— Chaîne de caractère lue dans le fichier
Constructeur :
— met le nom de fichier passé en paramètre dans les attributs, appel la fonction pour extraire le contenu du fichier.
Accesseurs :
— getContenu() permet de récupérer ce qui a été lu dans le fichier.
Méthodes :
— extraireContenu() permet de lire le fichier texte et de stocker le résultat dans l'attribut coresspondant.

1.3 Perspectives pour la suite

Pendant nos recherches, puis parallèlement à la rédaction de cette première version du code, nous avons également commencé à réfléchir à la façon de traiter la polyphonie avec cette application. Nous avons eu l'idée d'utiliser des Threads afin de traiter plusieurs tâches et donc lire plusieurs voix à la fois. La question que nous avons dû nous poser était la suivante : les Threads permettent-ils une lecture synchronisée des différentes voix ? Nous avons alors effectué quelques tests, et nous en avons déduits que nous pouvions totalement utiliser les Threads pour de la polyphonie. Nous adapterons notre code plus tard, il ne s'agit pas d'une fonctionnalité requise pour notre première version, il s'agissait plutôt de savoir si nous devions faire d'autres recherches ou si nous allions pouvoir utiliser les Threads dans le code.

Chapitre 2

Itération 2

Bibliographie

[1] http://www.commentcamarche.net/forum/affich-590149-lire-un-fichier-texte-en-java

[2] http://stackoverflow.com/questions/8632104/sine-wave-sound-generator-in-java

Annexe A

Liens utiles

Voici une petite liste d'url intéressantes au sujet de ce projet :

- www.polytech.univ-tours.fr
- Syntaxe Lilypond : http://lilypond.org/text-input.html
- Documentation complète Lilypond : http://lilypond.org/doc/v2.18/Documentation/learning.pdf
- http://docs.oracle.com/javase/7/docs/technotes/guides/sound/programmer_guide/contents.html
- https://www.openal.org/documentation/openal-1.1-specification.pdf
- http://www.fmod.org/documentation/

Synthétiser un signal sonore et l'enregistrer dans un fichier son

---

Rapport de projet S7

Résumé : A compléter

Mots clé : ???,????,?????,??????????,??,????

Abstract : To complete later

Keywords : ???,????,?????,??????????,??,????

Auteur(s)

Julien Amoros
[julien.amoros@etu.univ-tours.fr]
Florent Flécheau
[florent.flecheau@etu.univ-tours.fr]

Encadrant(s)

Maxime Martineau
[maxime.martineau@univ-tours.fr]

Polytech Tours
Département DI

École Polytechnique de l'Université de Tours
64 Avenue Jean Portalis, 37200 Tours, France
http://www.polytech.univ-tours.fr