Projet PonyTracker
Rapport

Guillaume Abramovici
Guy Godfroy
Mickaël Illy
Quentin Martin

Encadrant : Elizabeth Brunet

22 mai 2014

Table des matières

Introduction

La musique est un art ayant traversé les époques, évoluant au même rythme et à la même vitesse que les progrès culturels, sociaux, scientifiques et technologiques de l'Être Humain. Celui-ci a toujours su s'adapter à ces évolutions dans le but de faire progresser, d'améliorer, d'enrichir ses productions musicales. Parallèlement, l'informatique a connu ces dernières années une évolution sans précédent, devenant indispensable dans la vie de tous les jours.

Il n'en fallait alors pas plus pour deviner que le futur de la musique passerait peut-être par l'emploi de cet outil informatique. Traitement du son, des informations, des instruments, physiques ou virtuels,... aujourd'hui les ordinateurs nous offrent la possibilité d'aller plus loin dans tous les domaines artistiques et culturels, notamment dans la dscipline musicale, et de créer des œuvres qui n'auraient seulement pu être envisagées quand l'informatique n'en était qu'à ses débuts.

Il existe alors aujourd'hui un besoin de créer des applications permettant cette création musicale. En particulier, le genre de la musique électronique se prête particulièrement bien à cette idée. L'artiste voulant notamment produire ce genre de musique a désormais besoin d'un outil lui permettant de manier les sons, de les enregistrer, d'utiliser des instruments qu'il peut créer et d'appliquer les effets qu'il souhaite, sans pour autant posséder le matériel physique (les instruments), ou bien même les compétences nécessaires pour y jouer. Il s'agit donc de simuler les instruments et les effets.

L'avenir de la musique pourrait bien passer par l'infinité de possibilités que nous offre désormais l'informatique...

Section 1

Cahier des charges

1.1 Objectifs

Notre objectif est de développer un séquenceur de musique, qui s'inscrit dans la discipline de la MAO (Musique Assistée par Ordinateur). Nous souhaitons répondre au besoin de musique électronique permettant d'écrire des fichiers de musique séquentielle, appelés *modules*, et munir le programme d'une interface graphique simple et intuitive, afin de toucher le plus de passionnés de musique possible. Afin de répondre au besoin, l'utilisateur devra avoir la possibilité de :

- créer ou importer des sons de bases, nommés *samples*, dont l'unique importance en définitive réside en la forme d'onde sonore ;
- configurer des *instruments* virtuels basés sur ces samples.
- entrer des mélodies joués simultanément par ces instruments via une matrice, dans des phrases musicales appelés des *motifs*.
- placer des effets isolément sur chaque note ;
- définir l'agencement temporel des motifs.

Notre but est de créer un programme s'inspirant de ceux déjà existants (par exemple MilkyTracker) en améliorant l'interface et en les simplifiant afin de les rendre plus accessibles, notamment pour les utilisateurs débutants. Par ailleurs, plutôt que de reprendre d'anciens standards de fichiers, qui datent maintenant de plus d'une vingtaine d'années et qui sont devenus flous pour cause de nombreux rajout successifs, il a été convennu que nous créerions un nouveau format de fichier, qui permettra d'implémenter plus simplement les caractéristiques d'un module.

1.2 Contraintes

Le principal objectif du Projet Informatique est de développer la compétence des étudiants en génie logiciel et en gestion de projet en équipe. Ce projet est

réalisé par un groupe de quatre étudiants et est encadré par un enseignant. Nous devrons fréquemment organiser des réunions de suivi avec l'enseignement, afin de présenter l'avancement du projet.

Le temps consacré par chaque étudiant au projet est d'environ 50h. Afin de respecter les échéances imposées par le cadre pédagogique, nous avons établi après analyse de l'ensembles des tâches à réaliser un plan de charge prévisionnel, un planning prévisionnel et un suivi d'activité.

Le langage utilisé pour le développement du projet sera le langage C. L'interface graphique sera réalisée avec l'environnement graphique GTK (notamment via l'utilisation de l'outil de création d'interface Glade). Afin de partager les informations entre les membres du groupe, nous utiliserons Git. La documentation concernant toutes les fonctions et tous les modules codés sera réalisée à l'aide de l'assistant de documentation Doxygen. Nous remplirons également un Wiki tout le long de notre projet.

Le projet achevé, il sera présenté lors d'une soutenance devant un Jury. La soutenance se décomposera en trois phases : une présentation vidéo-projetée, une démonstration du fonctionnement du logiciel développé et la réponse aux questions du jury. Elle aura lieu le 27 ou le 28 mai 2014.

Le projet se décomposera également en plusieurs livrables à rendre :

— Livrable 1 : Pré-rapport. Exposition du projet, cahier des charges, spécification fonctionnelle, plan de charge et planning prévisionnels... *À rendre avant le 12 mars 2014*
— Livrable 2 : Premier prototype du logiciel. *À rendre avant le 10 avril 2014*
— Livrable 3 : Logiciel et rapport. *À rendre avant le 21 mai 2014*
— Livrable 4 : Présentation vidéo-projetable de la soutenance. *À rendre avant le 29 mai 2014*

1.3 Fonctionnalités

Nous avons dressé une liste des fonctions du programme qui se veut la plus exhaustive possible. Nous avons indicé chaque fonction selon leur priorité d'implémentation : $**$ pour les fonctions à implémenter pour la première version à rendre pour le livrable 2, et $*$ pour celles à implémenter pour le produit final. Les autres sont les fonctions moins importantes, voire sujettes à débat, et il est prévu de les implémenter uniquement nous en avons la possibilité technique et temporelle.

Voici dons la liste des fonctionnalités :

1.3.1 Gestion des samples

— Import de samples $**$

— Visualisation graphique des samples en vue d'affinage du son $*$

— Réglages des paramètres de lecture en boucle des samples $**$

1.3.2 Gestion des instruments

— Choix d'un sample pour chaque instrument $**$

— Accord du sample en fonction d'une note de préférence $*$

— Édition du volume d'attaque, de maintien et de chute via une enveloppe sonore $*$

— Positionnement stéréo

1.3.3 Matrice d'édition des motifs

— Entrée/visualisation de la mélodie par clavier physique $**$ ou clavier graphique cliquable $*$

— Présentation de la matrice selon le temps et les instruments $**$

— Choix du nom des notes façon anglo-saxonne ou latine (via fenêtre des préférences)

— Lecture de plusieurs instruments simultanément $*$

— Choix du tempo $**$, de la base de temps $**$ et du nombre de *ticks*[1] $*$

— Gestion des motifs (choix de l'agencement dans le temps, du tempo et de la base de temps) $*$

1.3.4 Effets

— Implémentation de plusieurs effets :

- arpège $**$
- portamento (montant, descendant, intelligent) $*$
- vibrato $*$
- tremolo $*$
- coupure $*$
- délai $*$

— Choix des effets via un menu déroulant en face de chaque note $*$

— Modification dynamique du volume et de la stéréo $*$

1. Voir définition dans le lexique en annexe

1.3.5 Fonctionnalités basiques

— Créer nouveau document ∗
— Ouvrir document ∗
— Sauvegarder document ∗
— Undo/redo ∗
— Maximiser/minimiser/fermer fenêtre ∗
— Export de la musique en fichier son (wav ou mp3) ∗

Section 2

Développement

2.1 Spécification fonctionnelle

L'utilisateur se confrontera à une fenêtre principale, qui permettra l'accès à la plupart des fonctions via des boutons ou via une liste de menus en haut de la fenêtre.

En haut à gauche, la première rangée de boutons permet d'enregistrer un fichier, d'ouvrir un fichier, d'annuler et de rétablir la dernière action. La deuxième rangée de boutons donne accès au contrôles de lecture du module ouvert ainsi que la matrice d'édition des motifs, c'est à dire passer au motif précédent, mettre en lecture, stopper la lecture et passer au motif suivant.

En haut au centre, dans le cadre noir, on aura accès au statistiques générales de lecture, avec affichage des numéros en écriture hexadécimale du motif, du temps et du *tick* en cours de lecture, de la durée depuis le début du module, et d'un VU-mètre par canal stéréo.

À droite, on trouve les boutons pour ouvrir les trois fenêtres de contrôles spécifiques des samples, des instruments et des motifs, un autre bouton en deux partie définissant le mode d'édition de la matrice (entrer des notes ou des effets), et enfin des champs éditables pour le tempo (BPM) et de nombre de *ticks* par temps. On peut aussi choisir l'instrument d'édition via un menu déroulant.

Enfin, dans toute la partie basse de la fenêtre, on trouve la matrice d'édition du motif en cours, l'axe des temps étant vertical et descendant, et le tout organisé en plusieurs canaux de lecture afin de pouvoir jouer plusieurs instruments en même temps. Le plus souvent, on réserve un canal par instrument, mais cette distinction est facultative.

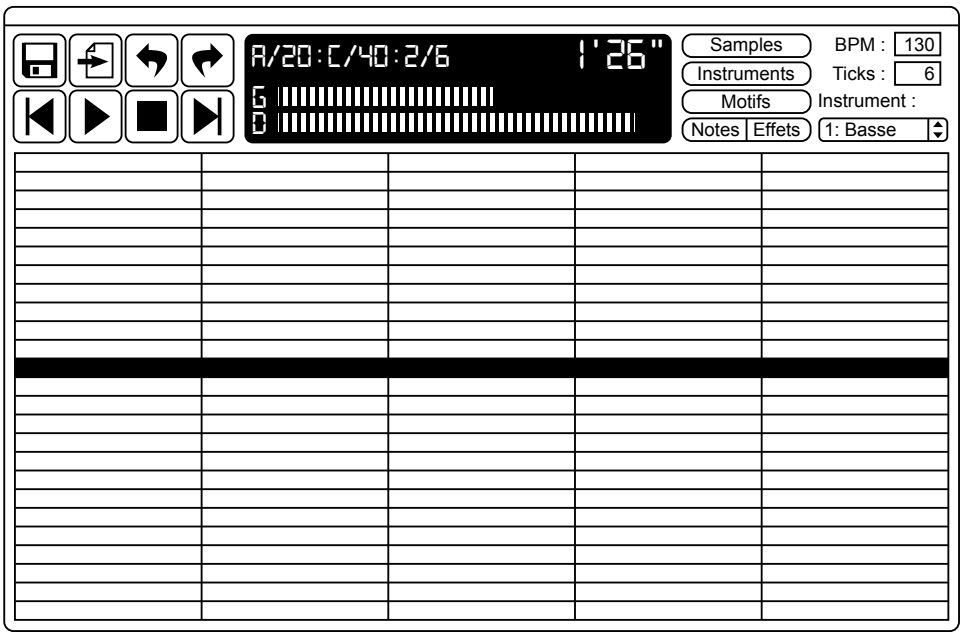


FIGURE 2.1 – Fenêtre principale

La fenêtres d'édition des samples permettra l'importation/édition des samples. Le premier bouton en haut à droite permet d'ajouter un sample à la liste juste en bas, et ouvre en même temps une fenêtre de navigation dans le système de fichiers de l'utilisateur. Le deuxième bouton supprime un élément. Les deux autre boutons démarrent et arrêtent la prévisualisation du sample. Le menu déroulant permet de choisir plusieurs modes de lecture du sample :

— *Pas de boucle* : le sample sera lu une seule fois, sans tenir compte des poignées de sélection définies par l'utilisateur, visibles sur le visualiser de forme d'onde en bas.

— *En boucle* : le sample sera lu en continu et en boucle entre les deux poignées de sélections.

— *En rafale* : similaire à la lecture en boucle, sauf que le sample sera lu à partir du début, puis en boucle entre les deux poignées.

— *En aller-retour* : le sample sera lu en boucle entre les deux poignées alternativement à l'endroit et à l'envers.

En haut à droite, des boutons radio permettent trois mode d'ajustement de la hauteur de la note :

— Si le sample a une hauteur qui correspond exactement à une note de la gamme tempérée, on peut alors directement préciser cette note.

— Si la note n'est pas juste, on peut être plus précis en indiquant plutôt la fréquence de la note.

— Enfin, si on est sûr d'avoir sélectionné uniquement une période du sample, on peut demander au programme de calculer automatiquement la hauteur de la note, en calculant la durée de la période entre les deux poignées.

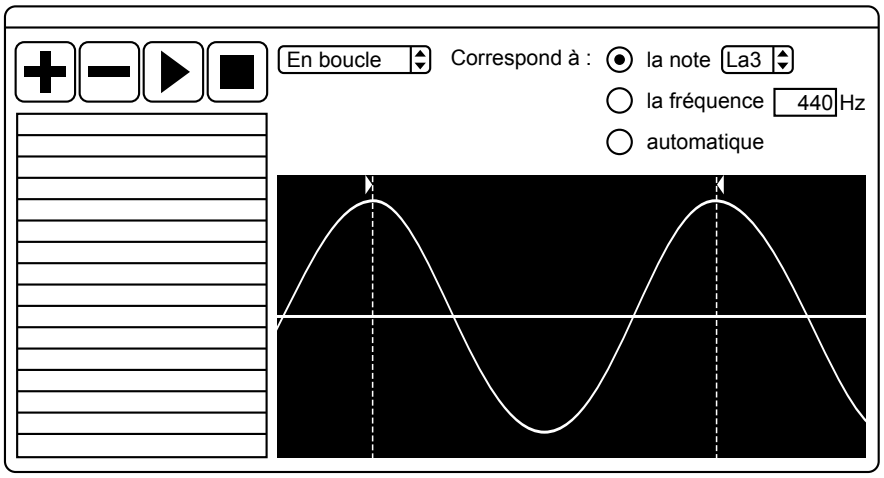


FIGURE 2.2 – Fenêtre d'édition des samples

La fenêtre d'édition des instruments permettra la création/édition des instruments par l'utilisateur. Elle présentera la liste des instruments couplée aux sample correspondants (dont le choix peut bien sûr être modifié) en vis-à-vis d'un cadre d'édition ou il sera possible d'éditer l'enveloppe sonore (activée ou non), d'ajouter une réverbération et de modifier les réglages propres à l'instrument, c'est à dire le volume et la balance de base.

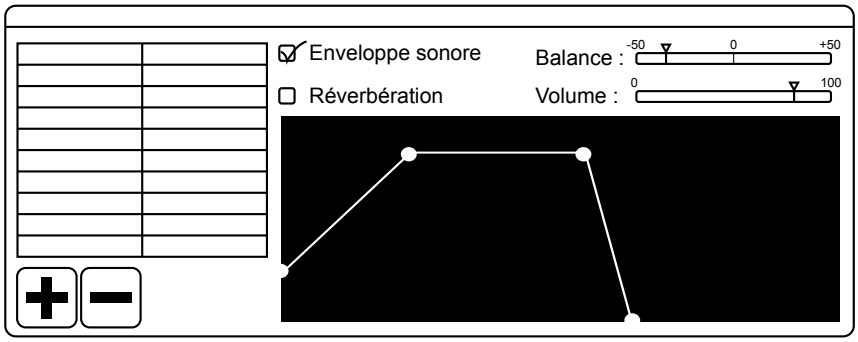


FIGURE 2.3 – Fenêtre des instruments

La fenêtre des motifs présentera une liste de mise en correspondance entre le motif joué à un moment donné du module et le motif effectif, permettant

ainsi de créer facilement des répétitions. On pourra choisir le nombre de temps par motif, et choisir si on veut que le module revienne au début après avoir lu tous les motifs.

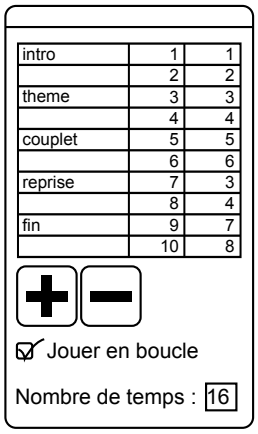


FIGURE 2.4 – Fenêtre des instruments

On implémentera éventuellement une fenêtre de préférences, permettant notamment de régler la convention de nommage des notes façon latine/anglo-saxonne. Nous attendons d'avoir plus de besoins de configuration avant d'envisager créer cette fenêtre.

2.2 Conception préliminaire

Nous avons organisé le développement de ce programme en plusieurs pôles. Çi-après un schéma de cette organisation.

Par ailleurs, nous nous servirons de la librairie PortAudio, qui est une librairie écrite en C, et qui permet la manipulation, l'entrée et la sortie du son.

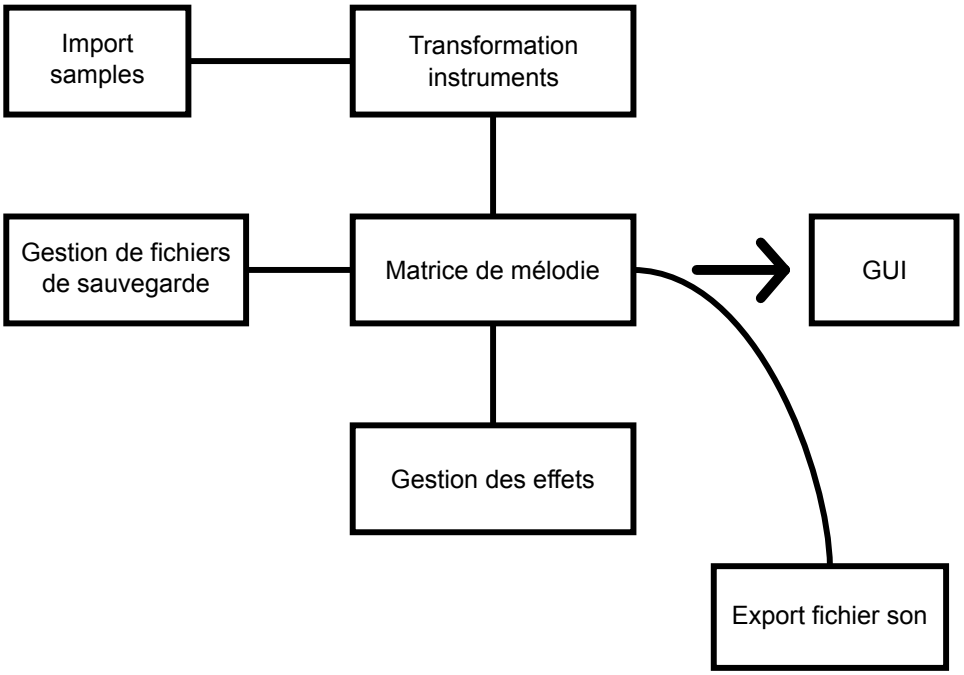


FIGURE 2.5 – Plan d'organisation des pôles de développements

Section 3

Tests unitaires

On va définir des tests unitaires pour chaque pôle de développement.

3.1 Import de samples

On crée un petit algorithme de vérification qui va se charger de lancer la procédure d'import de sample pour un sample donné, et qui va ensuite vérifier octet par octet si le fichier de sauvegarde contient à l'endroit donné le même sample.

3.2 Instruments

De même, après avoir créé un instrument, on vérifie que le fichier contient bien la structure attendue représentant l'instrument. On peut jouer une note et comparer le flux audio en fréquence pour constater que la note est la même que celle que l'on a entré.

3.3 Sauvegarde

On crée un nouveau fichier, que l'on modifie selon un ordre à définir, puis l'on sauvegarde. Puis on relance le programme on ouvre le fichier de sauvegarde, et on compare si les modifications sont identiques.

3.4 Effets

C'est la partie la plus délicate à tester, vu que les effets sont des éléments subjectifs. On sera obligé de tester à l'oreille les différents effets.

3.5 Mélodie

On entre dans le programme une mélodie, puis en comparant le flux audio aux points de vue de la justesse et de la durée des notes, on constate que la mélodie est respectée.

3.6 Export

On compare le flux audio lu et le fichier audio produit.

Section 4

Manuel d'utilisation

Au lancement du programme, un projet est déjà chargé.
Il est alors possible de lire la mélodie via le bouton lecture, ou éditer la mélodie en cliquant directement sur les notes, ce qui ouvre une fenêtre d'édition demandant d'entrer une note, un instrument, etc...

Conclusion

4.1 Échecs

Nous n'avons pas encore pu implémenter certaines fonctions :

— La fonction de sauvegarde et de chargement de projet n'est pas encore opérationnelle.

— Nous n'avons pas encore de possibilité de resampling.

— Il est impossible de créer des instruments et des samples via l'interface graphique.

4.2 Succès

Nous avons atteint certains de nos objectifs :

— Système de lecture efficace sur plusieurs canaux simultanés et changement de pitchs automatiquement

— Création de l'ensemble de l'interface graphique

— Affichage du motif en cours d'édition

— Édition du motif via une interface dynamique.

Il serait donc intéressant d'améliorer les point qui n'ont pas été traités, comme les fenêtre inefficaces et la sauvegarde de fichiers. Il serait également souhaitable d'utiliser une bibliothèque libre pour la gestion du son.

Ce projet a été très enrichissant quant à la gestion de sons et de l'interface graphique. Nous avons une plus grande connaissance des outils mis à disposition pour le traitement du son. Nous avons connus beaucoup de problèmes liés aux liens des bibliothèques, ce qui a amélioré notre compréhension du processus de compilation. Nous nous sommes confrontés également aux problèmes classiques de gestion de projet en équipe, comme la répartition du travail.

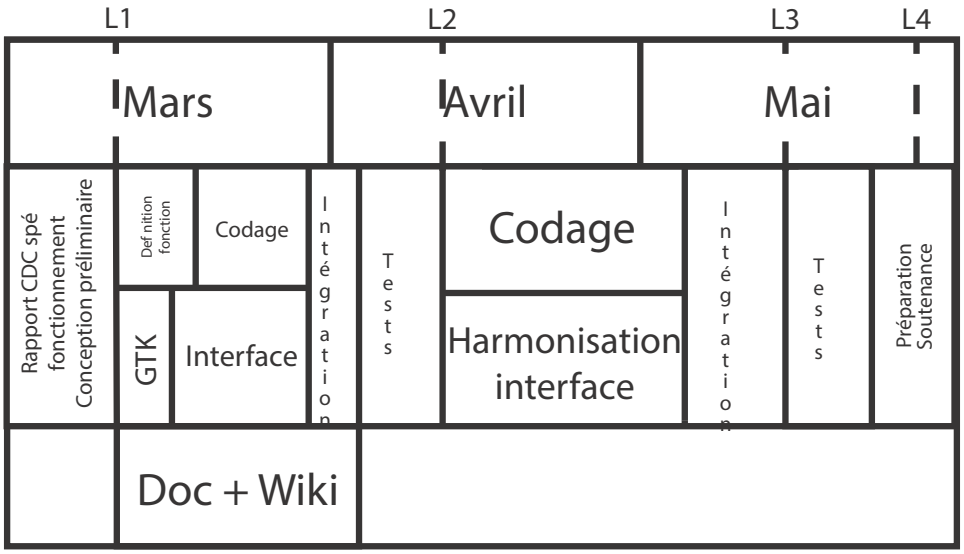
Annexe A

Gestion de projet

A.1 Plan de charges

Description de l'activité	Charge en %	Charge en H	Charge en H / Participant			
			G.A.	G.G.	M.I.	Q.M.
Total	**100 %**	**243**	**57**	**64**	**64**	**58**
Gestion de projets	**37 %**	**91**	**19**	**26**	**26**	**20**
Réunion de lancement	2 %	4	1	1	1	1
Planning prévisionnel et Suivi d'activités	2 %	4	1	1	1	1
Réunions de suivi	10 %	24	6	6	6	6
Rédaction	10 %	24	4	8	8	4
Outils collaboratifs (svn, wiki)	14 %	35	7	10	10	8
Spécification	**3 %**	**8**	**2**	**2**	**2**	**2**
Définition des fonctionnalités	3 %	8	2	2	2	2
Conception préliminaire	**12 %**	**28**	**7**	**7**	**7**	**7**
Définition d'un modèle de données	2 %	4	1	1	1	1
Définition d'un format de fichiers associé au modèle de données	2 %	4	1	1	1	1
Définition des fonctionnalités	5 %	12	3	3	3	3
Définition des différents modules (.h)	3 %	8	2	2	2	2
Conception détaillée	**16 %**	**40**	**10**	**10**	**10**	**10**
Définition des structures de données	2 %	4	1	1	1	1
Définition des fonctions	8 %	20	5	5	5	5
Définition des tests unitaires	2 %	4	1	1	1	1
Auto-formation Gtk et Glade	5 %	12	3	3	3	3
Codage	**20 %**	**48**	**12**	**12**	**12**	**12**
Ecriture des interfaces (.h)	2 %	4		2	2	
Ecriture du makefile	2 %	4	2			2
Ecriture des différentes fonctions	13 %	32	8	8	8	8
Tests unitaires	3 %	8	2	2	2	2
Intégration	**5 %**	**12**	**3**	**3**	**3**	**3**
Intégration des différents modules	3 %	8	2	2	2	2
Tests d'intégration	2 %	4	1	1	1	1
Soutenance	**7 %**	**16**	**4**	**4**	**4**	**4**
Préparation de la soutenance	5 %	12	3	3	3	3
Soutenance	2 %	4	1	1	1	1

A.2 Planning prévisionnel



Annexe B

Code source

Annexe C

Lexique

Module Morceau de musique séquentielle, contenant à la fois les samples, les instruments les motifs, ainsi que les divers réglages associés.

Sample Unité élémentaire de son, ne contenant le plus souvent qu'une ou deux période d'onde sonore, destiné à être répété en boucle dans le but de produire un son continu homogène et qui en détermine ainsi le timbre.

Instrument Entité permettant d'effectuer un mélodie, constitué d'un ensemble de réglages sur un sample, comme une enveloppe sonore, le volume et la balance.

Motif Phrase musicale, délimitant des passages dans le module et permettant au musicien d'organiser sa musique. Un motif contient une mélodie constituée de notes couplées à des réglages individuels d'effets sonores et de volume.

Tick Élément temporel atomique de l'édition musicale. On peut définir le nombre de ticks par temps, en général entre 3 et 10.