Rapport de Projet

JaSMEd
JavaScript Music Editor

Etudiants :
Clément Bossut
Valentin Cassat
Jaime Chao
Thibaud Cummings

Encadrant :
Emmanuel Saint-James

Parcours SAR
Université Pierre et Marie Curie
UPMC Paris

2ème semestre 2012

Table des matières

Chapitre 1

Introduction

Nous avons proposé ce projet car il s'inscrit à la fois dans le domaine des applications réparties (web), et reflète nos orientations musicales dans la pratique de l'informatique.

C'est dans un cadre internet en constante transformation que l'on a développé Jasmed, une application client/serveur qui tire partie des nouvelles API d'HTML5.

JaSMEd est un éditeur musical multi-utilisateur temps réel implémenté à partir des nouveaux standards du web, et qui est spécifique de part son approche de composition polyrythmique [1].
Actuellement il n'existe pas d'outils pour créer une polyrythmie musicale de manière collaborative. En général il n'y a pas de moyens simples permettant d'écrire ce genre de rythmes complexes, et encore moins sur internet où la majeure partie des applications musicales se bornent à être de simples gadgets et non pas de véritables outils de composition.
C'est en faisant ce constat que nous est apparue l'idée de développer Jasmed.

Notre travail s'inscrit dans l'ue PSAR, c'est donc un projet fonctionnel mais non encore abouti que nous présentons, et qui s'axe sur une partie répartie pour s'inscrire dans la problématique du master SAR.

Nous avons cherché à travailler de la meilleure façon possible, en nous servant d'outils utiles aux programmateurs souhaitant travailler en groupe. Nous avons proposé le projet, ce qui est une chance mais qui peut aussi être un danger si le sujet se révèle trop ambitieux, nous avons constamment cherché à nous recentrer et nous concerter avec notre professeur référent Emmanuel Saint-James.

1. Superposition de plusieurs parties ayant chacune un rythme différent et dont les accents d'appui ne coïncident pas entre eux“ (Larousse de la Musique, Paris, t. 2, 1982, p. 1247).

Chapitre 2

Etat de l'art

Le HTML est le langage principal du World Wide Web, le W3C [1] est actuellement entrain de travailler sur HTML5, qui est la 5ème révision majeure de ce langage. Il est important de noter que cette version est toujours à l'état d'ébauche [2] et ne cesse d'évoluer. HTML5 fournit une flopée de nouvelles API, notamment la balise <audio> [3] qui permet de fournir un lecteur audio simple, et permet enfin de s'affranchir des players lourds et privés . Malheureusement ce n'est pas suffisant pour un traitement de l'audio plus poussé, c'est ainsi que des librairies audio ont fait leur apparition. Chaque navigateur a développé son api audio, WebAudio pour Google Chrome Audio Data Api pour firefox Il existe plusieurs librairies qui abstraient ces 2 api audiolib

Dernièrement, une quantité de nouvelles aplications web musicales sont apparus sur la toile - éditeur musical -jeux sonores

La majeure partie de ces applications ont un aspect ludique, et ne sont pas vraiment utiles à des compositeurs/musiciens. Si on regarde les logiciels d'édition musicale ils ne permettent pas de créer facilement des musiques poly-rythmiques, mais se prettent plus facilement au binaire. Live,

http ://harmonyseq.wordpress.com/

Notre projet s'inscrit donc au coeur des avancées actuelles du web dans le domaine de l'audio, et il propose une vision d'édition poly-rythmique peu répandue dans les logiciels audio.

1. Consortium du www
2. `http://www.w3.org/TR/html5/index.html`
3. `http://www.whatwg.org/specs/web-apps/current-work/multipage/the-video-element.html#the-audio-element`

Chapitre 3

Technologies utilisées

3.1 Confort de développement

Google docs, git, github, cloud9

3.2 Client

audiolib, jquerry, underscore, backbone, socket.io,

3.3 Serveur

node, socket.io, express, redis

Chapitre 4

Conception

4.1 Général

le client s'exécute sur un navigateur web, la synthèse du son étant gérée par l'API audiolib ; la communication temps-réel entre les deux parties s'effectue via le protocole webSocket.
Le serveur utilise nodejs, une librairie JavaScript ajoutant la programmation évenementielle au langage.
Plusieurs clients peuvent éditer une unique piste musicale de manière concurrente.
Le programme client offre divers outils d'édition et de lecture du morceau, tandis que le serveur s'occupe de la communication entre les différents éditeurs et de la sauvegarde des données.

4.2 Modularité

4.3 Jeux de tests

Chapitre 5

Architecture

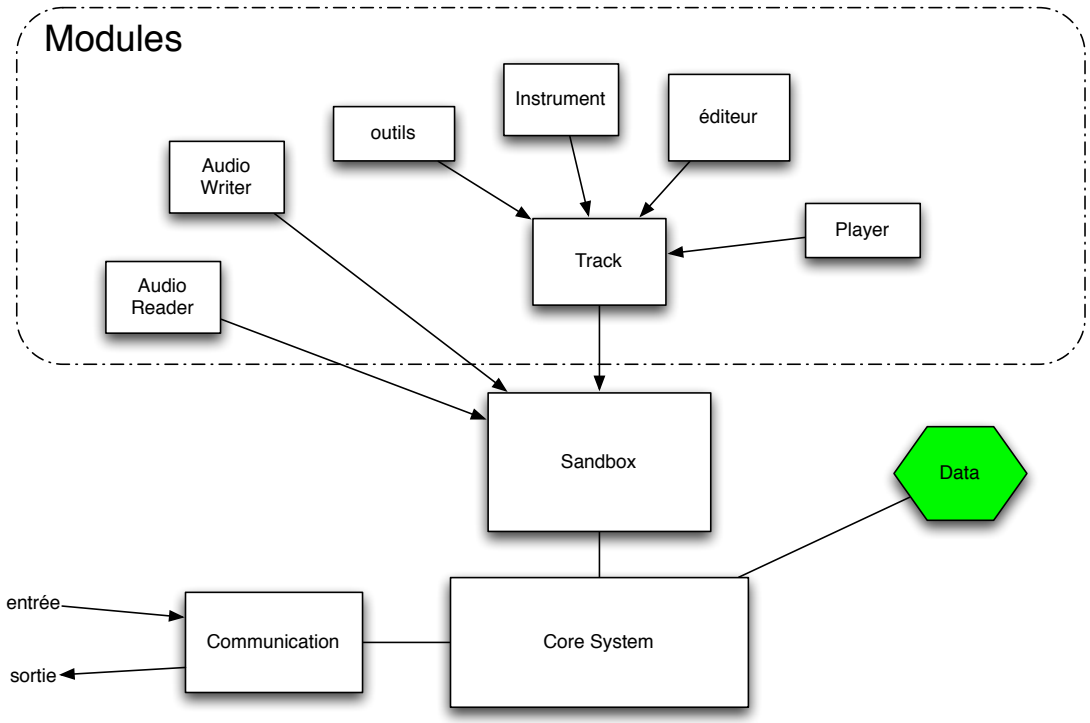
5.1 Serveur

Le serveur centralise et re-distribue les messages d'utilisateurs travaillant dans la même salle - sur le même morceau - tout en gérant les collisions éventuelles. Il implante deux protocoles réseaux :

- HTTP pour l'initialisation de l'application et le chargement de morceaux ;
- Websocket pour toutes les modifications temps-réel effectués par les clients.

5.2 Client

L'idée est de se détacher des notations solfégiques usuelles pour s'orienter vers une écriture musicale plus ludique et intuitive, adaptée notamment aux enfants. Dans ce but, nous nous sommes émancipé de la signature rythmique et avons imaginé une division en blocs (aparentable à des mesures). Tous les blocs ont une durée identique, et sont indépendamment subdivisibles (un bloc s'apparente à une mesure avec une signature et une vitesse d'exécution adapté).



L'utilisateur peut ainsi construire un morceau en accolant des blocs et en spécifiant pour chacun le nombre de temps égaux à insérer dans celui-ci. De cette manière, il est possible de reproduire toute

construction rythmique (à la manière des constructions LEGO).
Un même bloc peut être composé de différents “calques”, un calque représentant ce bloc avec une subdivision particulière : cela permet d’éditer simplement de la polyrythmie.

Chapitre 6

Journal de bord et déroulement

Février

choix du projet, M.Saint-James accepte d'être notre professeur référent Thibaud partage un ensemble de standards web nouvelles générations, des liens vers des exemples (HTML5, socket.io, build large app) Clément rejoint le projet

24/02

rdv avec St-James -> il veut qu'on lui montre une app fonctionnelle et simple rapidement Mars 01/03

rdv : on se met d'accord sur ce qu'on commence chacun à coder pendant le w.e

Thibaud : architecture côté client

Clément : fonction qui traduit les messages du client en midi

Jaime : mini-serveur opérationnel qui envoit les messages valides à la fonction de Clément

Valentin : travail sur graphisme svg, fonction de zoom, rajouter ses collègues dans gitHub

06/03

choix de la façon de stocker les données musicales côté client :

dictionnaire de tableau de tableau

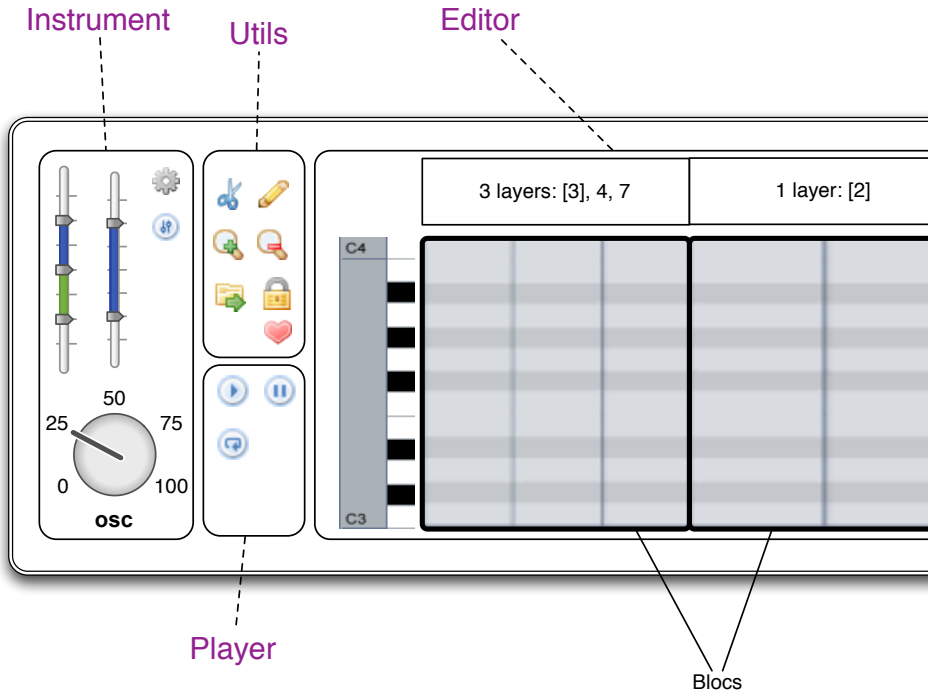
tableau de pistes -> tableau de mesures ->tableau de temps

remise en question du midi car trop contraignant pour peu de bénéfices court terme (->portabilité) possibilité d'écrire une fonction qui traduirait notre représentation musicale en midi

Chapitre 7

Manuel d'utilisation

Le client fournit à l'utilisateur des outils d'édition, de lecture, d'instrumentation, et représente gra-



phiquement la piste en cours d'édition.

7.0.1 Lecture

Les outils de lecture sont simplement constitués de trois boutons :
- lecture/pause
- stop
- répétition de la piste

7.0.2 Instrumentation

Le panneau d'instrumentation permet de choisir l'instrument qui jouera la piste lors de la lecture, ainsi que de modifier le volume et d'y ajouter des effets prédéfinis.

7.0.3 Édition

La partie édition représente une partition musicale à éditer/jouer, spécifiant la hauteur, la longueur, et la position de chaque note. L'éditeur permet la navigation verticale et horizontale :horizon-

talement sur l'axe du temps et verticalement selon la hauteur du son.

Bibliographie

[1] David Flanagan, *JavaScript : The Definitive Guide.*
O'Reilly, 2011.

[2] Jussi Kalliokosky, *Audio Library Application Programming Interface.*
`https://github.com/jussi-kalliokoski/audiolib.js`

[3] W3C (, *Audio Library Application Programming Interface.*
`https://github.com/jussi-kalliokoski/audiolib.js`