



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 JaSMEd 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 qui tire sa spécificité de part son approche de composition polyrythmique¹. 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 ne sont 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 programmeurs 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 W3C¹ 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'ébauche² 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 enfin de se passer des vieux lecteurs et codecs lourds et privés (Windows media player, realPlayer, quicktime). 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.

Les navigateurs Firefox et Google Chrome ont ainsi développé leur propre API audio, permettant de Ces API ont été créés dans le but d'offrir au web game developer un vrai environnement sonore pour la conception de WebAudio est un peu plus poussée que Firefox, de même elle bénéficie d'un plus grand engouement de la communauté. D'après les listes de diffusion du working draft du W3C, l'API de Google serait même candidate à la standardisation.

WebAudio, largement inspiré du système CoreAudio d'Apple, pour Google Chrome Audio Data Api pour firefox

Il existe déjà beaucoup de librairies javascript qui abstraient ces API et permettent de construire des applications audio sur le web plus facilement⁴. Certaines, comme `audiolib.js[?]` ou `audionode.js`, permettent d'abstraire les différences entre les 2 API concurrentes et ainsi grandement faciliter la portabilité entre navigateurs.

Une quantité de nouvelles applications web musicales sont apparues sur la toile avec l'arrivée d'HTML5. La majeure partie de ces applications ont un aspect ludique (ou se bornent à être de simples gadgets), et ne sont pas vraiment utiles à des compositeurs/musiciens. On en a recensé un échantillon :

- jeux sonores
- éditeur musical

Si on regarde les logiciels d'édition musicale ils ne permettent pas de créer facilement des musiques poly-rythmiques, mais se prettent plus facilement à composer des tempos binaires. Les plus connus d'entre eux :

Ableton Live

<http://harmonyseq.wordpress.com/>

<http://www.technitone.com/#>

1. World Wide Web Consortium

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>

4. https://wiki.mozilla.org/Audio_Data_API#JavaScript_Audio_Libraries

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.

Chapitre 3

Technologies utilisées

3.1 Confort de développement

Google docs, git, github, cloud9

3.2 JavaScript

3.3 Librairies

3.3.1 Client

audiolib, jquery, underscore, backbone, socket.io,

3.3.2 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 événementielle 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 Architecture

4.2.1 Modularité

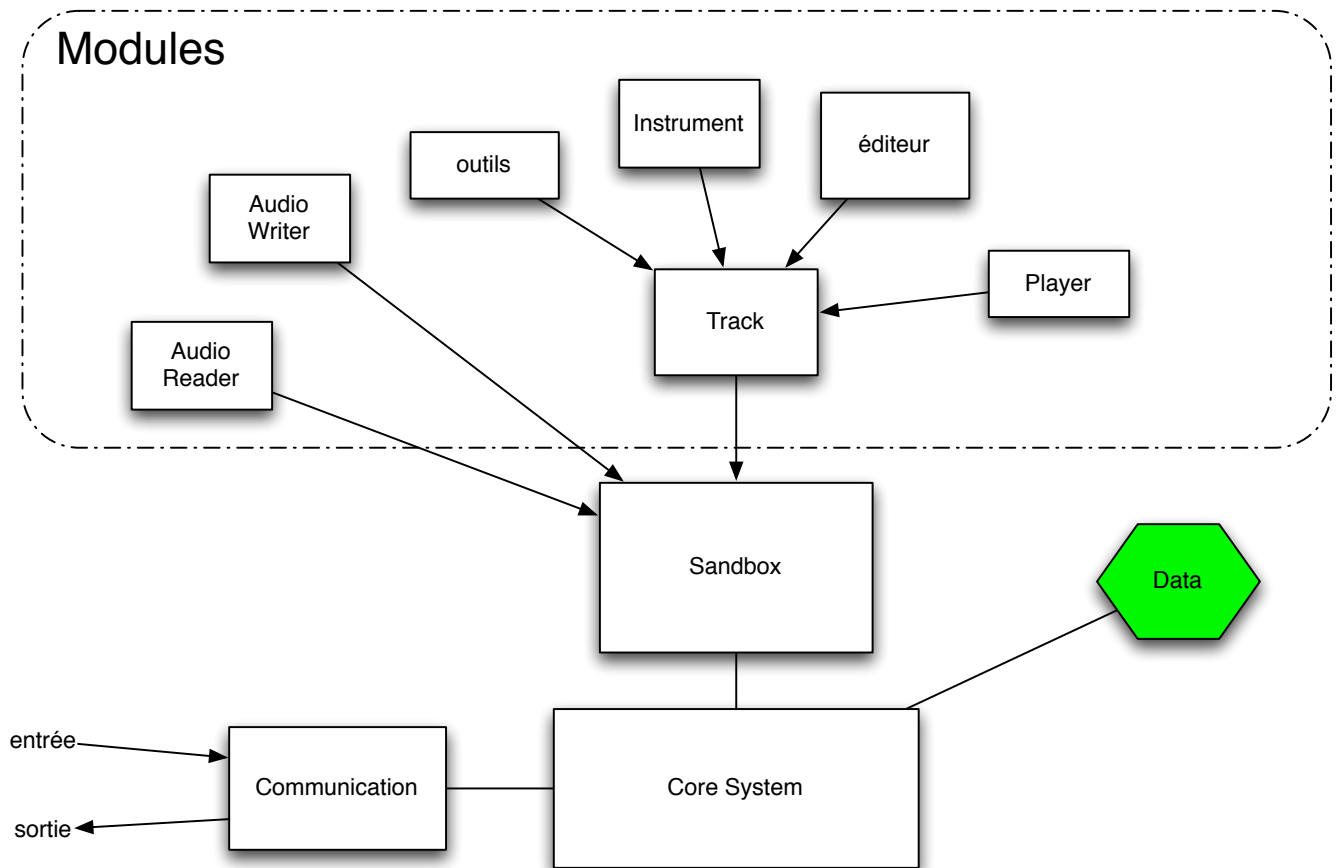
4.2.2 Serveur

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.

4.2.3 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.

4.3 Jeux de tests

Chapitre 5

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 envoie 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

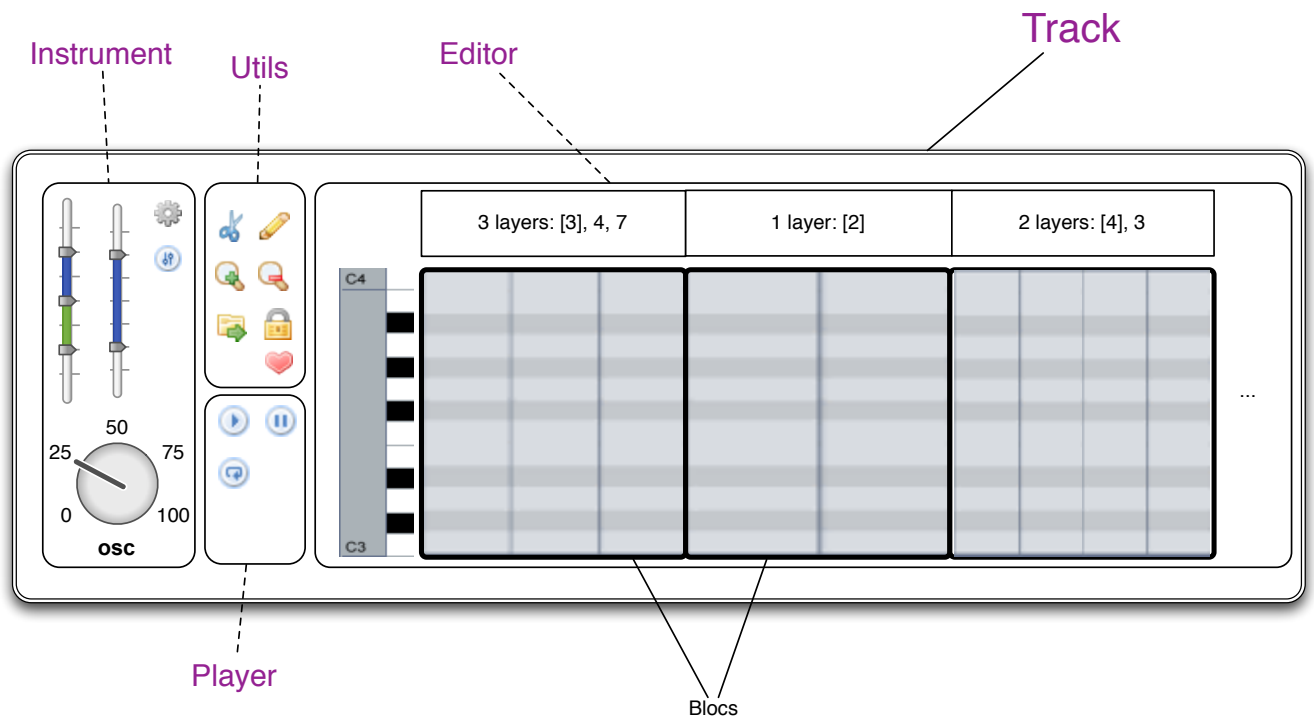
tableau de pistes -> tableau de mesures -> tableau de temps

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 6

Manuel d'utilisation

Le client fournit à l'utilisateur des outils d'édition, de lecture, d'instrumentation, et représente graphiquement la piste en cours d'édition.



6.1 Lecture

Les outils de lecture sont simplement constitués de trois boutons :

- lecture/pause
- stop
- répétition de la piste

6.2 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.

6.3 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 :horizontalement 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 Kalliokoski, *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>