Programmation Mobile
Antoine Caillon
Philippe Esling

Soundboard app

Créez votre launchpad virtuelle

2022-05-09
antoine.caillon@ircam.fr
philippe.esling@ircam.fr

Le but de ce projet est d'implémenter une version virtuelle simplifiée d'un sampler / sequencer physique. Ces systèmes permettent le jeu en live d'une banque de sons pré-enregistrés via l'utilisation d'une grille de pads. Dans leur version la plus simple, ces machines sont uniquement utilisable dans leur version *sampler*: les samples sont jouable par pression des pads, à l'instar d'un synthétiseur, mais des versions plus avancées permettent une utilisation de type *sequencer* (l'utilisateur peut créer / éditer une boucle jouée de manière répétée afin de créer des rythmes / patterns).

1. Modalités d'évaluation

Date de rendu ..... 13 juin 2022
Groupes ..... Effectuer le projet *seul* et rendre par dépôt `GitHub`

Système de notation

Rendu attendu

Code source (`React Native`) 60%
Rapport (`PDF`) 40%

Points bonus

Code et documentation en anglais ..... **+1**
Rapport en anglais ..... **+1**

Distribution des points

Vue Sampler **6pts**
Bibliothèque de samples **4pts**
Édition de samples (rognage) **3pts**
Vue enregistrement de sons **3pts**
Vue recherche freesound **3pts**

Critères d'évaluation

Clarté du code (découpage en petites méthodes, bon nommage des variables) .....
Redondance du code (pas de répétition). .....
Extensibilité (ajout facile de fonctionnalités). .....
Justesse du code. .....

2. Préliminaires

Quelques notes générales à suivre pour le projet.

- Préférez implémenter moins de fonctionnalités mais bien.
- Facilitez la vie de l'utilisateur final
  Si vous hésitez entre deux choix, préférez celui qui est ergonomique pour l'utilisateur.
  Si ils sont équivalents, choisissez ce qui est le plus simple à implémenter et relire.
- Documentez votre code, ajoutez des commentaires là où vous pensez qu'il y a besoin d'explications

1) Gestionnaire de version

Pour ce projet, nous vous demandons d'utiliser un gestionnaire de version. Comme vu en cours, nous allons utiliser le service de `github`. Nous rappelons ici les étapes principales à suivre pour celui-ci.

1. Créer un compte sur `http://www.github.com`.
2. Vous pouvez dès lors créer un répertoire à versionner.
3. Lors de la création, ajoutez une licence a votre projet (MIT ou LGPL).
4. Envoyer l'adresse de ce répertoire par mail à `esling@ircam.fr`, vous serez noté grâce à celui-ci.

Vous pouvez installer sur Windows l'exécutable `http://msysgit.github.io/`. Nous rappelons ici quelques commandes principales a connaître, mais vous pouvez suivre un tutoriel plus fourni, comme par exemple `http://git-scm.com/docs/gittutorial`

Cloner un répertoire Permet de télécharger un dépôt, créer le dossier associé, et mettre en place le système de versionnage

git clone `https://github.com/esling/tarot.git`

Ajouter un fichier au projet

git add App.js

Faire un commit l'option `-a` met à jour tous les changements des fichiers qui ont été ajoutés.

git commit -a -m " Initial commit"

Upload des changements locaux (via commit sur le serveur distant)

git push

Download des changements du serveur sur votre machine locale. A priori, seulement utile si vous travaillez sur plusieurs machines ou à plusieurs (pour synchroniser votre pc portable et le pc fixe).

git pull

Il y a des plugins spécifiques pour chaque IDE qui incorporent cet outil directement. Nous vous conseillons néanmoins de vous familiariser d'abord avec la ligne de commande pour saisir les concepts généraux de `git`. Ensuite vous devez respecter ces règles

- Effectuer des petits commit réguliers (un pour chaque fonctionnalité et résolution de bugs).

- Le message dans un commit doit être clair, explicite et relativement court.
- Vous ne devez effectuer un commit que sur du code compilable (pas d'erreur).
- Faites fréquemment des push, au moins une fois par jour, utile en cas de crash et pour les retours.
- Soyez cohérent, si vous codez en anglais, les messages des commit seront en anglais.

3. Soundboard app

1) Architecture du projet

L'application proposée va être sensiblement plus complexe que ce que vous avez pu faire jusqu'ici en terme de contenu et d'organisation. Avant de vous lancer dans son implémentation, commencez par écrire une version conceptuelle de l'application, sous la forme d'un graphique expliquant les relations entre chaque vue / bouton / fonction (voir figure 1) que vous présenterez dans votre rapport. Les éléments à inclure dans l'application sont présentées dans les parties suivantes

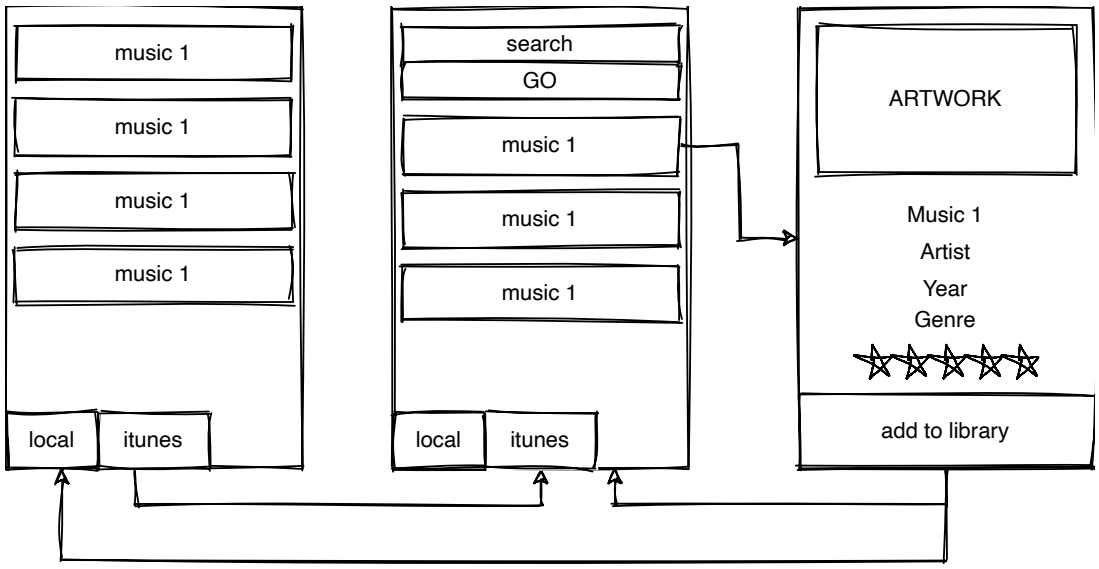

Figure 1: Exemple de conceptualisation de l'application iTunes Seeker

2) Sampler

La base de l'application est la vue *sampler*, contenant une grille de pads, permettant à l'utilisateur de lancer la lecture de samples[1]. Les pads présents dans la grille devront réagir différemment en fonction de la durée de la pression:

- Une pression courte lance la lecture du sample
- Une pression longue ouvre une vue de modification du sample, présentée dans la section 3)

[1] Vous pouvez vous référer à https://docs.expo.io/versions/latest/sdk/av/ pour la partie audio.

3) Edition du sample

La vue d'édition du sample doit permettre à l'utilisateur de modifier le sample présent dans le pad sélectionné. Les interactions sont les suivantes:

- Rognage du sample présent (choix d'un début et d'une fin)[2]
- Choix d'un nouveau sample parmi la bibliothèque locale.
- Enregistrement d'un nouveau sample via le micro (voir section i.)
- Recherche d'un sample par mot-clé en utilisant l'API de *freesound* (voir section ii.)

Les samples présents dans la bibliothèque locale doivent pouvoir être triés **en fonction de leur origine** (enregistré, par défaut et freesound).

i. Micro

La source **micro** est une vue qui permet à l'utilisateur de s'enregistrer via le micro de l'appareil et d'insérer l'enregistrement produit comme élément de la banque de son, en y ajoutant un titre et une description.

ii. Freesound

Le site web *freesound* propose une API permettant une recherche dans leur base de donnée (voir leur documentation). Vous aurez besoin de vous créer un compte et d'obtenir une API Key afin de pouvoir effectuer des requêtes au sein de votre application. Vous devez ajouter le même système de titre + description du sample que pour la vue micro i..

4) Persistance

De manière à pouvoir conserver votre bibliothèque de samples, vous intégrerez un système de persistance de l'état de votre application. Vous pouvez au choix directement enregistrer les samples audio personnalisés ou alors leur URL s'ils proviennent de *freesound*.
Vous intégrerez également un système de *presets* pour la vue sample 2), de manière à pouvoir charger et sauvegarder une certaine disposition de sons.

5) Bonus - Sequencer

Si vous avez implémenté tous les points précédents, vous pouvez rajouter un système de sequencer à votre application, de manière à pouvoir enregistrer des boucles, et les éditer par la suite. Ce point étant optionnel, vous avez totale liberté en ce qui concerne l'étendu des fonctionnalités présentes.

[2] Pour gérer l'édition vous pouvez utiliser react-native-trimmer