Audio et i-score

Jean-Michaël Celerier

8 février 2016

1 Introduction

Ce document présente différentes possibilités pour l'écriture et l'utilisation de son dans i-score, et les environnements qui peuvent lui être adjoints.

1.1 Problématique

Les questions auxquelles ce document tente d'apporter une ou plusieurs réponses sont les suivantes :

- Quel doit être l'agencement des responsabilités entre i-score et les outils annexes pour avoir des possibilités d'écriture maximale. Cette question de partage de responsabilités se pose aussi au niveau des utilisateurs de ces outils : qui utillise i-score ? Qui utilise wWise ? Qui utilise Unity ?
- Quelles sont les contraintes techniques qui peuvent s'appliquer lors de l'interopérabilité avec différents environnements, et comment les résoudre. Par exemple, comment faire en sorte que l'environnement fonctionne sur mobile.
- Comment doit fonctionner la gestion des médias dans un projet complet.
- Comment doit fonctionner la répartition dans un cas ou l'on désire produire une œuvre distribuée ; cela pose la question de la séparation du moteur d'édition et d'exécution dans le cas du son, ainsi que de leur communication.
- Quelles sont les problématiques d'écriture qui se pose lorsque l'on désire écrire des scènes sonores ? (Mute de certaines parties ? Collaboration à l'écriture ?)
- Un modèle de calcul a-t-il sa place dans l'environnement, et si oui, qui doit le fournir, et où ces calculs sont-ils écrits ?

2 Problématiques techniques

2.1 Fonctionnement sur mobile / plate-formes embarquées

Il n'y a généralement pas ou peu d'IPC sur ces plate-formes : tout doit être contenu dans une seule application. Exception : les applications audio sur iOS, depuis iOS 7 [1] [2]. La méthode standard est de communiquer via internet.

Plusieurs approches : Audiobus, Ableton Link.

Pour Android il n'y a pas encore de standard pour le faire [3] [4] [5].

3 Types de moteurs audio

Il existe de nombreuses possibilités pour écrire une application produisant du son. Les outils que l'on considère peuvent se répartir dans différentes catégories (et faire partie de plus d'une catégorie à la fois).

— Application graphique (Max/MSP, Ableton Live)
— Bibliothèque, framework, API (PortAudio, Jamoma)
— Middleware (wWise, Unity)
— Domain Specific Language (DSL) (FaUST, Max/MSP)
— Application de contrôle et de routage (Audiobus, JACK)
— Plug-in pour une autre application (ils sont nombreux)

Les interactions peuvent être nombreuses entre ces formats : Par exemple, un middleware peut fournir une API pour intégrer des plug-ins. Un de ces plug-ins peut être écrit dans un DSL qui est compilé en C.

Les API vont être de plusieurs types :

— API d'intégration : pour rajouter des outils audio à un système complet (ex. : API VST).
— API de création : pour créer ses propres outils audio (ex. : STK).

Un autre axe est la main que prend l'API sur la gestion du son :

— API maîtresse : le programmeur n'a pas le contrôle du flot audio. Ex. : OpenAL, Unity.
— API esclave : le programmeur contrôle le flot audio (il n'est pas géré par l'API). Ex. : PortAudio, API bas-niveau.

Certaines API peuvent fonctionner des deux manières.

1. https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/Inter-AppCommunication/Inter-AppCommunication.html
2. http://www.musicradar.com/tuition/tech/audiobus-vs-inter-app-audio-which-is-best-620144
3. http://developer.samsung.com/galaxy#professional-audio
4. http://superpowered.com/
5. http://discchord.com/blog/2013/9/4/patchfield-audio-architecture-for-android.html

On trouve de plus plusieurs modes d'exécution (qui là encore peuvent fonctionner de concert) :

— Déclaratif : on décrit à l'avance les traitements qui vont avoir lieu dans un langage propre à l'API (souvent un Embedded Domain Specific Language (EDSL)).
— Graphe audio : cas déclaratif le plus courant. On décrit la manière dont des blocs audio qui appliquent un traitement sont reliés entre eux.
— Push : très utilisé dans les moteurs de jeu ; l'utilisateur doit régulièrement appeler une fonction, généralement dans la boucle `update()`, qui va mettre à jour le son.
— Pull : l'API appelle régulièrement une fonction fournie par l'utilisateur qui remplit un buffer audio. C'est l'approche la plus courante pour du code bas-niveau ou devant être performant car elle correspond au fonctionnement des drivers audio.

Autre possibilitié intéressante issue de Faust : utiliser LLVM pour recompiler et optimiser à la volée du patch ?

4 Modèle de données

5 Gestion de la spatialisation

6 Briques en présence

Les outils dont on dispose sont :

— Des moteurs de scénarisation. Un moteur de scénarisation permet de décrire une évolution du temps, en prenant en compte des évènements extérieurs. i-score et wWise en sont.
— Des scènes spatiales. Ce sont des descriptions et agencements d'objets en deux ou trois dimensions qui peuvent évoluer.
— Des moteurs de son.
— Des moteurs de physique : prennent à un instant t une scène spatiale et la transforment en une autre scène après application des lois physiques en vigueur (gravité, etc.).
— Des sources de données spatiales.

7 Possibilités d'implémentation

Questions : * qui gère la sortie son ? * qui gère la spatialisation en sortie (sur les hauts-parleurs) * qui gère la spatialisation en entrée (des objets) * qui applique des

effets * qui contrôle l'écoulement du temps scénaristique * qui fait office de source sonore * qui communique avec qui

- SuperCollider comme moteur audio ? - Tout dans i-score ? - libaudiostream et la place de FaUST ? - Grapholine ? - Problème du contrôle si deux moteurs.

Cas 1. Séquenceur intégré à i-score i-score devient live Question principale : gestion de l'horloge Possibilité : - Un processus s'enregistre aurpès du mixeur - Quand le processus est démarré, le mixeur appelle pull() dessus - Problématique : si le processus est déclenché via un évènement interactif, il est nécessaire de le prévoir (par exemple en commençant à appeler pull dès que l'on est dans la zone interactive, ou bien en utilisant des mutex / compteurs atomiques, ou bien en rajoutant de la latence, ou bie en mangeant le début.) - Problématique : bufferisation si effet sonore met du temps à s'applique

Cas 2. Utilisation d'un séquenceur externe

Cas 3. Utilisation de i-score à haut niveau pour grands scénarios et à bas niveau pour objets sonores. Entre les deux communications avec objets qui gèrent le son.

Cas 4. Utilisation de wWise Implémentation de i-score / grapholine / autre comme plug-in wWise ?

Pb. de la bufferistaion : temps-réel, latence ?

8 Questions sémantiques

Possibilté d'utilisation d'un outil comme OWL ?

9 Utilisation pour les applications possibles

- Tableaux, installs, etc. - Morceaux de musiques interactifs que l'on peut distribuer par internet ?