Compte-rendu à t=2.0y

Jean-Michaël Celerier

23 novembre 2016

Table des matières

Chapitre 1

Introduction

Ce document survole les travaux qui ont été réalisés jusqu'à présent lors de la thèse, étudie les pistes qui sont ouvertes et les possibilités pour la dernière année.

1.1 Mise en relation avec le sujet

Calques audio interactifs : **théorie**, mise en œuvre et usages.

1.2 Articulation et analyse générale

Chapitre 2

Réalisations

2.1 Étude de cas

2.1.1 Sonopluie

Ce cas est constitué de zones pré-écrites qui correspondent à des étapes d'une randonnée géo-localisée.

Le signal d'entrée est un positionnement GPS.

Les ambiances sonores évoluent selon les zones. Il doit y avoir une forme d'interaction multi-pointeurs : par exemple, quand deux personnes sont dans la même zone, la scénarisation locale du son peut évoluer.

Pour l'écriture, il est utile d'avoir accès à une source de données cartographiques, comme Google Maps, MapQuest, etc.

2.1.2 Tableaux sonores

Dans ce cas, on travaille sur des tableaux que l'on désire sonifier, d'une manière artistique et avec une notion de scénarisation.

On interagit à l'aide d'une surface tactile ou d'un capteur de mouvements.

La scénarisation est définie de manière hiérarchique :

— Une ambiance globale peut recouvrir tout le tableau.
— Comme dans le cas Sonopluie, des zones plus spécifiques peuvent porter une ambiance additionnelle.
— Des éléments individuels (objets) peuvent avoir un comportement qui leur est propre.

Bosch

Ce triptyque riche en détails permet d'essayer différentes techniques.

— Spatialisation d'éléments statiques : bruits d'oiseaux, etc.
— Spatialisation d'objets dynamiques : cercle du panneau central.
— Instruments de musique : répartis dans le panneau de droite.
— Historique avec personnages : foule en bas du panneau central.

Tableaux de Vernet

- Profondeur - Refaire en vrai ? - Le tableau ne correspond pas à la réalité

2.1.3 Jardin connecté

2.1.4 Pièces électro-acoustiques spatialisées

2.1.5 Grapholine

2.2 Développements théoriques, publications

2.2.1 États de l'art

- Comparaison avec ce qui existe déjà

2.2.2 Modèle théorique

TENOR2015 : OSSIA

Présente une spécification en réseaux de Petri des travaux réalisés lors du projet OSSIA.

IUI2015 (refusé)

JNMR : Vérification

Traduction du modèle graphique en automates temporisés tels que définis par J. Arias.

On essaye par la suite de permettre aux compositeurs de vérifier des propriétés sur leurs partitions interactives, telles que "deux séquences ne sont jamais exécutées en même temps".

JIM2016 : Interface

Étude d'un scénario complexe (Cas du coureur, au futuroscope).

ICMC2016 : Programmation structurée

"Design patterns" et outils de programmation standard à l'aide des objets présentés précédemment.

Démo : inspiré de Klavierstücke XI.

i-score qui communique par WebSockets avec un serveur Node.JS

Des gens se connectent par téléphone au serveur et peuvent influer sur l'exécution du scénario : prochaine partie à jouer, et vitesse moyenne de lecture.

2.2.3 Espace

Compte-rendu espace

JIM2016 : Espace

Analyse de différentes méthodes pour représenter des objets spatiaux dans le cadre de i-score. -> Conclusion : CAS peu adéquat, dur d'avoir de bonnes performances à un tick rate quelconque. Alternatives : se restreindre aux cas linéaires ? GPU ? Mais latence.

JIM2016 : Démo

Présentation d'une sonification interactive du tableau de Jérôme Bosch, *Le Jardin des Délices*.

2.2.4 Audio

SMC2016 : i-score et LibAudioStream

Intégration de la LibAudioStream avec i-score : traduction de scénarios i-score en expressions LibAudioStream et synchronisation des deux moteurs d'exécution ; outils graphiques pour le son.

Démo : toutes les fonctionnalités.

2.2.5 Répartition

Stage master

Proposition : méthode de répartition en introduisant de nouveaux réseaux de Petri.

Étude subséquente : se baser sur modèle OSSIA

On essaye d'implémenter le même formalisme, en utilisant non plus sur les réseaux de Petri, mais sur les outils dont on dispose dans le modèle OSSIA : TimeNode, Event, Trigger.

2.3 Conférences, présentations, workshops

— **Cycles SCRIME 2014** Présentation du travail de stage sur la répartition.
— **Cycles SCRIME 2015** Présentation d'un état de l'art sur l'espace.
— **Forum IRCAM 2015** Présentation générale du logiciel.
— **FOSDEM 2016** Présentation générale du logiciel à l'aide d'un example réalisé par Théo en Processing.
— **GDR ESARS** Présentation générale du logiciel.
— **Cycles SCRIME 2016** Présentation du travail sur l'audio.

— **Workshop improvisation LaBRI** Présentation du travail sur l'audio.
— **Workshop DESINC2016** Workshop d'initiation à i-score avec des participants d'horizons variés.

2.4 Développements logiciels

2.4.1 Génie logiciel et généralités

Performances

Comment mettre en valeur l'aspect "performances" du logiciel? Un accent très fort est mis dessus lors du développement.

Par exemple :
— Gestion du cache et de la mémoire via une minimisation des allocations dynamiques.
— Métaprogrammation et utilisation de techniques permettant de déporter le plus de code possible à la compilation plutôt qu'à l'exécution pour améliorer les optimisations possibles par le compilateur.
  — Dans libossia, gestion des types "tableaux" courants (`Vec2f`, `Vec3f`, `Vec4f`) sur la pile plutôt que sur le tas. C'est par opposition à Jamoma ou le type `TTValue` est sur le tas et nécessite une allocation dynamique quoi qu'il arrive.
  — Utilisation de ces types pour les dataspaces.
  — Dans i-score, instantiation des composants du logiciel.
— Structures de données optimisées au cas par cas.
— Caches variés pour améliorer les performances.

Tests

Couverture de code :
— 70 % pour libossia via des tests unitaires et d'intégration.
— 50 % pour i-score via des tests d'intégration.
— 0 % pour les extensions à i-score (audio, espace, etc.).

Idéalement il devrait y avoir au moins 90% de couverture.

Mais plus de tests implique moins de R&D.

2.4.2 i-score

Architecture

Problèmes actuels

Portabilité

Après avoir enlevé la dépendance à Jamoma, on obtient l'exécution sur Android et iOS.

Le logiciel marche donc sur toute les plate-formes courantes.

2.4.3 Extensions à i-score

Édition répartie

Lien : `https://github.com/OSSIA/iscore-addon-network`

Cette extension permet à plusieurs instances d'i-score de partager le même scénario au moment de l'édition.

Ce dépôt contient aussi les bases des travaux de répartition de l'exécution.

Audio

Lien : `https://github.com/OSSIA/iscore-addon-audio`

Contient :
— L'intégration avec la LibAudioStream.
— Les flux développés pour la bibliothèque (Send et Return).
— Les processus i-score correspondants.

— Une intégration avec le format de plug-in LV2.
Utilise Faust et optionnellement la bibliothèque lilv pour LV2.

Automation 3D
Lien : `https://github.com/OSSIA/iscore-addon-autom3d`
Contient un processus qui permet de décrire une automation sur un paramètre à trois dimensions via une spline cubique, et un widget permettant l'édition de cette automation en 3D.
Utilise la bibliothèque VTK.

Espace
Lien : `https://github.com/OSSIA/iscore-addon-space`
Contient le processus Espace qui permet de définir des zones via des équations cartésiennes et des interactions entre ces zones.
Utilise la bibliothèque VTK.

Image
Lien : `https://github.com/OSSIA/iscore-addon-image`

Offre un processus purement utilitaire, qui permet d'afficher une image dans un scénario i-score.

Vidéo
Lien : `https://github.com/OSSIA/iscore-addon-video`

Offre un processus qui lit une vidéo de manière intégrée à la timeline i-score.

Controle à distance
Lien : `https://github.com/OSSIA/iscore-addon-remotecontrol`

Expose une partie de l'exécution i-score via un protocole simple sur WebSockets, qui permet notamment de déclencher les points d'interaction à la volée.
Contient aussi une application simple pour tablette ou téléphone qui permet d'interagir avec ce protocole pour permettre le déclenchement au moment ou les points d'interaction deviennent accessibles.

Analyse statique
Lien : `https://github.com/OSSIA/iscore-addon-staticanalysis`
Contient le code utilisé pour :
— La traduction en automates temporisés de l'article JNMR.
— Un algorithme de génération aléatoire de scénario i-score.
— Des outils de mesure de métriques de code appliqués aux scénarios i-score, notamment les métriques de complexité de Halstead.

Extension "Preset"
Lien : `https://github.com/OSSIA/iscore-addon-presetstate`
Prototype permettant de charger un état via un fichier de configuration.

PureData
Lien : `https://github.com/jcelerier/iscore-addon-pd`
Prototype permettant d'utiliser des patches PureData dans un scénario i-score. Pour l'instant ne gère que des patchs avec une entrée et une sortie fixe. Utilise libpd.

Segments
Lien : `https://github.com/OSSIA/iscore-addon-segments`
Doit servir de base pour la brique d'intégration entre i-score et le projet Segments. Pour l'instant contient juste un exemple d'utilisation de quelques interfaces d'i-score.

2.4.4 libossia

Architecture

Problèmes actuels

- temps de compilation

Portages

C

Csharp et Unity

Qt

Java

Javascript

2.4.5 OSCQuery

2.4.6 coppa

2.4.7 Études et développements mineurs

External RealSense

Outils pour graphe de calcul

DisPATCH

RaftLib

Contribution à d'autres projets open-source

LibAudioStream

FAUST

Jamoma

Contributions mineures

- Placeholder/Nodeeditor
- verdigris
- fmt
- Qt-color-widgets
- jni.hpp
- quazip
- QRecentFilesMenu
- ModernMIDI
- libsamplerate
- ofxMSAPhysics
- Cotire

2.5 Projets liés

2.5.1 Audio

Stage Magali Chauvat

Objectifs - Il faut encore faire la brique d'intégration dans i-score

2.5.2 Robots

Stage Nicolas 2015
Stage Kinda Al Chahid 2015
Stage Paul Breton 2016
Stage Maëva 2016
Projet TM - Robot 2015 - 2016
Objectifs

Groupe TM

Groupe Robots

Projet TM - Robot 2016 - 2017
Objectifs

Groupe TM

Groupe Robots

PFA2016 - 2017
Objectifs

2.6 Cours et TDs donnés

2.6.1 TIM

2.6.2 TAP

Chapitre 3

Objectifs à venir

3.1 Système réparti

3.1.1 Exécution répartie

3.1.2 Répartition des protocoles

3.2 Audio

3.2.1 Article dans CMJ ?

Pour que ce soit convainquant : offrir en plus la possibilité de réutiliser les flux passés. Et bien tout modéliser.

— Utiliser la modélisation de la LibAudioStream (Flux) ?
— Utiliser la modélisation de OSSIA (Petri) ?

3.2.2 Signatures temporelles

3.2.3 Support audio étendu

- Gestion des pistes (entrées / sorties virtuelles comme dans un vrai séquenceur) - Intégration travail TrackListModel de Magali - Spatialisation / autre chose que stéréo -> pistes de sortie 1, 2, 3, N canaux

3.2.4 Question des queues de reverb

Possibilité :

— Au début, lire tous les streams à l'infini dès qu'ils commencent.
— Question des boucles ? que doit-il se passer ?

Dans un DAW normal on boucle la lecture de contenu (fichiers audio, notes midi), mais les effets sont hors de la boucle.

Ici les deux choses sont possibles, c'est à l'auteur de choisir (en mettant les effets dans la boucle ou hors de la boucle).

- Compensation de délai : certains effets (compresseur, limiteur, certaines réverbs) impliquent un délai dans le traitement du signal. La plpuart des DAW compensent ce délai au prix d'une latence d'entrée plus élevée : les autres pistes sont décalées de manière à tenir compte du délai des plug-ins. Ableton Live offre le choix : compenser le délai ou avoir une latence d'entrée plus faible. Si le délai n'est pas prix en compte, deux pistes qui étaient synchronisées à la base ne le seront plus après application de l'effet, ce qui peut poser problème pour des sections rythmiques par exemple.

Certains types d'effets préviennent à l'avance du décalage imposé, tandis que d'autres non. Par exemple : - Les effets VST supportent un délai statique (fixé au chargement du plug-in) - Les effets Reaper, Pro Tools supportent un délai dynamique. - Ce doit être fait manuellement par l'auteur pour FaUST.

VST / VSTi
LV2
Format de plug-ins qui permet l'analyse en temps réel de données.

3.3 Embedding de i-score

3.3.1 DLL dans d'autres moteurs d'exécution

Max
VST
Unity

3.3.2 Scénarios compilés

3.3.3 Web

3.3.4 IncludeOS pour devices ?

3.4 Unification temps - espace

- Lister les cas possibles : comprendre ce que "appliquer les structures de i-score dans l'espace" veut dire. Ex :

— Si on considère qu'une contrainte temporelle est toujours sur une seule dimension, de temps ?
— Si on considère qu'une contrainte temporelle est toujours sur une seule dimension, d'espace ?
— Si on considère qu'une contrainte temporelle est sur plusieurs dimensions d'espace ?
— Si on considère qu'une contrainte temporelle est sur toutes les dimensions à la fois (on fait "progresser" l'espace temps et on arrive dans des espaces différents selon les outcome du scénario i-score).
— Si on considère non pas l'avancement dans un scénario comme quelque chose de global, mais quelque chose de lié à un acteur. On a dès lors besoin de multiples curseurs de temps.

Alternative "intégration dans unity" : on applique à un objet existant dans Unity, un scénario i-score. Ce scénario devrait pouvoir interagir avec le reste des paramètres ? On voit juste un aspect d'un scénario global ?

3.5 Modèle par graphe de noeuds pour calcul par tranches

On veut à chaque instant (tick) avoir un graphe de calcul différent, en fonction des processus qui sont exécutés.

Il faut :

— Changer le moteur d'exécution afin qu'il fonctionne réellement sous la forme d'un graphe.
— Changer l'implémentation des processus et le type `ossia::state` pour qu'ils renvoient un calcul à effectuer plutôt que le résultat de ce calcul ; c'est une forme d'évaluation paresseuse.
— Changer les processus pour qu'ils spécifient les adresses qu'ils utilisent en entrée - sortie. Par exemple pour le processus Javascript cela permettrait d'activer le listening sur les adresses qu'il utilise en entrée, plutôt que de faire un "pull" dessus qui peut être plus long.

3.6 Édition à l'exécution

Actuellement, les structures de données utilisée à l'exécution sont recréées à chaque fois que le bouton "exécuter" est pressé. C'est du au fait que les structures de données de l'API OSSIA ne sont pas thread-safe : il est actuellement dangereux de changer par exemple la durée d'une contrainte pendant qu'elle s'exécute car ces deux choses se font dans des threads distincts.

La première possibilité est de rajouter des mutex aux endroits impactés par l'édition (c'est à dire à peu près partout).

La seconde (ayant ma préférence) est d'utiliser une file de commandes pour l'édition : des commandes d'édition sont envoyées et mise dans une file sans verouillage (lock-free queue).

Le moteur d'exécution peut appliquer les commandes d'édition dans le cadre de sa boucle d'exécution, une fois que le tick d'exécution courant est terminé. Cela permet de garder les performances du code sans mutex, tout en prévenant les crashs.

Par la suite, il est nécessaire de changer la manière dont l'API est appelée depuis i-score pour qu'elle garde les structures en mémoire plutôt que de les recréer à chaque exécution, et qu'elle réagisse aux changements dans l'interface graphique.

Une perspective intéressante serait d'offrir par la suite une API d'édition par le réseau (OSC) ou via Javascript, pour pouvoir écrire des partitions auto-génératives.

3.7 Polyphonie

- Faire une liste des évènements communs à tous les types de polyphonie

3.7.1 Polyphonie dans les scénarios

On voudrait qu'une boite puisse être exécutée en même temps par plusieurs curseurs.

— Comment l'écrire ? actuellement il n'y a qu'un curseur de temps. Faut-il introduire des exécutions parallèles sur le même scénario ?
— Peut-on synchroniser ces exécutions parallèles, avec les outils dont on dispose ?
— Peut-on mettre en parallèle avec le travail sur la répartition : plusieurs scénarios s'exécutent indépendamment ? Problème de l'approche : on fixe à l'avant les utilisateurs ? Ou pas...
— Problématique : "stateful processes". Par exemple un tween.
— Ce n'est pas utile d'émettre des informations sur les mêmes adresses. Pour la polyphonie il est nécessaire de paramétrer les adresses. Ne peut-on pas le faire avec un script JS par exemple ?
— Y-a-t-il des processus qui pourraient avoir du sens uniquement de manière polyphonique ?
  — Automations : il faut que les adresses soient différentes. Si on a n participants, on peut mappers aux adresses [1 ; n]. e.g. foo.0, foo.1, ... avec le système d'instances ? Sinon il faut scripter... objet qui déduplique ? mais on se retrouve avec Max.
  — Interpolations : il faut que les adresses soient différentes
  — Mapping : il faut que les adresses soient différentes
  — JS -> avoir un tableau avec tous les curseurs de temps en entrée ? Ou exécuter chaque script de manière indépendante ?
  — Scénarios, boucles : récursivité
  — Space : multi-pointeurs
  — Fichier son : bof.
  — Effect : bof.
  — "Group" mappings (cf rapport master). Mapping $[1 \to n]$, $[n \to 1]$, $[n \to n]$, $[m \to n]$

Deux cas :

— Cas d'un scénario maître avec plein de petits scénarios esclaves : téléphones ?
— Cas de scénarios entièrement indépendants dans l'exécution : qu'est-ce qu'on y gagne par rapport à lancer plusieurs fois i-score.

Si on s'autorise des curseurs spatiaux. Plusieurs scénarios qui s'exécutent.

Possibilité de déclarer un curseur à partir de n'importe quel noeud de l'arbre.

3.7.2 Polyphonie dans l'espace

- interaction entre deux agents / utilisateurs

3.7.3 Polyphonie dans le son

Première étape : piste virtuelle.

Buffers d'entrée : sorte de polyphonie ? - plusieurs sons - plusieurs auditeurs

3.7.4 Spatialisation du son

- plusieurs enceintes En général deux méthodes : - Spatialisation par piste - Spatialisation par objet - Intégration temps / espace

3.8 Scripting de l'interface en Javascript

3.9 Interaction

- Extraction des informations. Informations de quoi ? Les dataspace le font déjà dans un sens.

3.10 Langage

Modéliser les développements réalisés via un langage.
Approche escomptée : langage réactif ou les variables sont des processus.
Comment le langage peut-il rendre compte de l'aspect "plug-in" du logiciel ?
Très dur de faire un langage extensible au niveau de la syntaxe / sémantique ...
Approche DSL dans un langage existant ?
Le langage doit-il produire un scénario i-score ou bien directement s'exécuter ?
Compiler en code C++ qui correspond ?

3.11 Problématiques d'UI / de développement

- Afficher un processus sous plusieurs aspects (e.g. pour une courbe 3D, les automations x, y, z ainsi que l'allure générale que ça donne). - Alias dans arbre (par exemple trois noeuds addrR, addrG, addrB => une seule addresse addr) - Problématique de l'édition réciproque.

3.11.1 Dataspace

- Gérer les dataspace qui wrap (0 -> 360)

3.11.2 Plug-in marketplace

3.11.3 Bibliothèque

3.12 Lien entre différents aspects

Comment spatialiser un son efficacement en combinant les processus espace, etc.
Comment tout exécuter dans navigateur : il faut du webaudio pour la libaudiostream.

3.12.1 Objets sonores interactifs

-Objets sonores interactifs "self-contained" -> "Chien qui aboie" -> Tonneau qui tombe ; une fois qu'il est tombé, on ne peut plus interagir avec (jusqu'à quand ?).

-> On doit pouvoir les échanger, les stocker... -> Des fichiers sons peuvent être associés. -> Question de propriété intellectuelle ? Empêcher l'édition et ne laisser que certains paramètres accessibles.

Chapitre 4

Conclusion

4.1 Emploi du temps
4.2 Discussion