

i-score : retour sur deux ans d'expérimentations autour de l'écriture interactive

Pierre Cochard, Myriam Desainte-Catherine

Réalisateur en informatique musicale, SCRIME-LaBRI

Professeur d'informatique à l'ENSEIRB-MATMECA, Bordeaux INP

Université de Bordeaux, CNRS LaBRI, UMR 5800, F-33400 Talence, INRIA, France,

pierre.cochard@u-bordeaux.fr

myriam.desainte-catherine@u-bordeaux.fr,

1. Introduction

Cet article présente 6 cas concrets d'écriture interactive ou semi-interactive réalisés au SCRIME entre janvier 2015 et février 2017, à l'aide du logiciel i-score. Certains d'entre eux ont été conçus dans le contexte de démonstrations du logiciel, d'autres, dans le cadre de réelles productions intermedia proposées et/ou accompagnées par le SCRIME.

Nous présenterons en premier lieu brièvement le logiciel i-score ainsi que le contexte général de conception des cas d'écriture, puis, chacun de ceux-ci sera exposé en suivant une même trame, détaillant : le projet et son contexte de réalisation ; le matériel et les logiciels "cibles" utilisés (dans un rapport de flux d'entrées-sorties, centrés sur i-score) ; la méthodologie d'écriture ; enfin, une réflexion rétrospective sur la conception du scénario et l'utilisation du logiciel.

1.1 Le logiciel i-score

Le logiciel i-score¹ est un séquenceur intermedia interactif. Il permet d'organiser visuellement des événements statiques et/ou interactifs sur une *timeline* et de coordonner l'exécution de processus de contrôle pilotant différents types de média (audio, vidéo, lumière...) au sein d'un ensemble nommé scénario. Les événements interactifs sont déclenchés dynamiquement lors de l'exécution, à l'inverse des événements statiques, dont le déclenchement dépend uniquement des relations temporelles formalisées dans le scénario. Le logiciel, depuis sa version 0.3, introduit de nouvelles fonctionnalités, telles que le *mapping* (qui consiste en une fonction de transfert entre deux différents paramètres), les conditions, les boucles, l'encapsulation, ainsi que les événements et processus *JavaScript*. Les protocoles de communication entre i-score et les appareils ciblés incluent : MIDI, Open Sound Control (OSC), OSCQuery, Minuit, WebSocket/HTTP, Port Série.

1.2 Contexte général d'écriture et de conception

La médiation vis-à-vis d'i-score et de son mode d'utilisation fait, au SCRIME, partie intégrante du rôle du Réalisateur en informatique musicale (RIM). Celui-ci est en effet chargé de tester, d'éprouver le logiciel et ses fonctionnalités à partir des indications des développeurs, puis d'accompagner les artistes – jusqu'à présent *beta-testeurs* – sur sa prise

¹ i-score.org

en main, puis sur la modélisation et l'écriture de scénarios concrets. Selon les projets, le RIM peut être amené à traduire et à ajuster une partition dynamique préexistante en scénario i-score, de guider l'artiste sur l'écriture d'un scénario, ou bien d'en créer lui-même intégralement le contenu.

2. Démonstration – Echantillonneur granulaire (3^{ème} version)

Cette démonstration fut réalisée entre décembre 2014 et janvier 2015, à l'aide d'i-score 0.2.3. Il s'agit ici de la 3^{ème} version de la série de démonstrations qui nous avons conçues avec i-score à partir de 2015, qui a été choisie pour sa stabilité. Elle a notamment été présentée dans le cadre d'une réunion des équipes du projet ANR INEDIT, en présence du GRAME et de l'IRCAM, mais aussi dans le contexte d'un reportage de TV7 (chaîne de télévision locale à Bordeaux) sur les arts numériques innovants. Elle fut ensuite revue et revisitée quelques mois plus tard avec la version 0.3 d'i-score, à l'occasion d'une présentation aux forums IRCAM en novembre 2015.

2.1 Dispositif

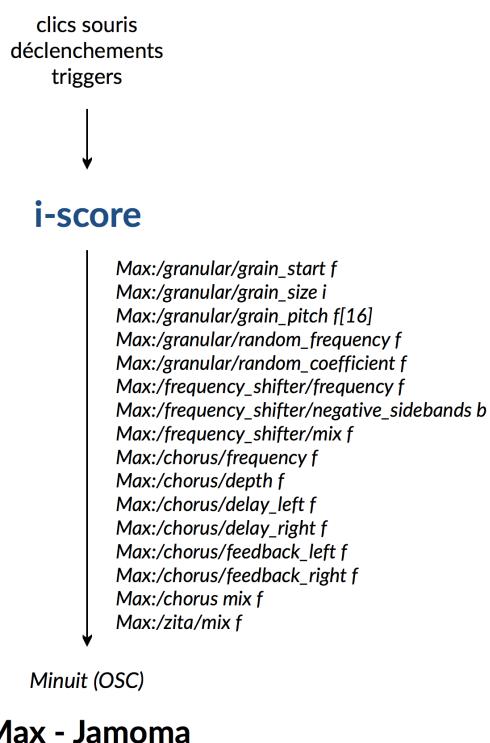


Figure 1 : schéma de dispositif, démonstration n°3

Dans le contexte du dispositif employé, i-score ne reçoit qu'un seul flux d'entrée : le clic de la souris, traité et géré en interne dans le logiciel, permettant de déclencher dynamiquement certains évènements (*triggers*), dont les états sont dépendants d'expressions prédéterminées par l'utilisateur. Il pilote en sortie un échantillonneur granulaire ainsi que 3

modules de traitements de signaux numériques (*frequency shifter*, *chorus*, réverbération), réalisés à l'aide des environnements de programmation Max et Faust (FIGURE 1).

La liaison entre i-score et le *device* Max est effectuée à l'aide du *package* Jamoma², qui permet une communication implicite, simplifiée par le protocole Minuit (système de requêtes *Open Sound Control* automatiques). Ainsi, chacun des paramètres et nœuds OSC créés dans le patch (à l'aide des objets *j.model* et *j.parameter*) se retrouvent - avec leurs différents attributs -- automatiquement exposés dans l'arborescence i-score, représentée dans la fenêtre de gauche (le *device explorer*), et synchronisés.

2.2 Scénario

i-score : démonstration n°3

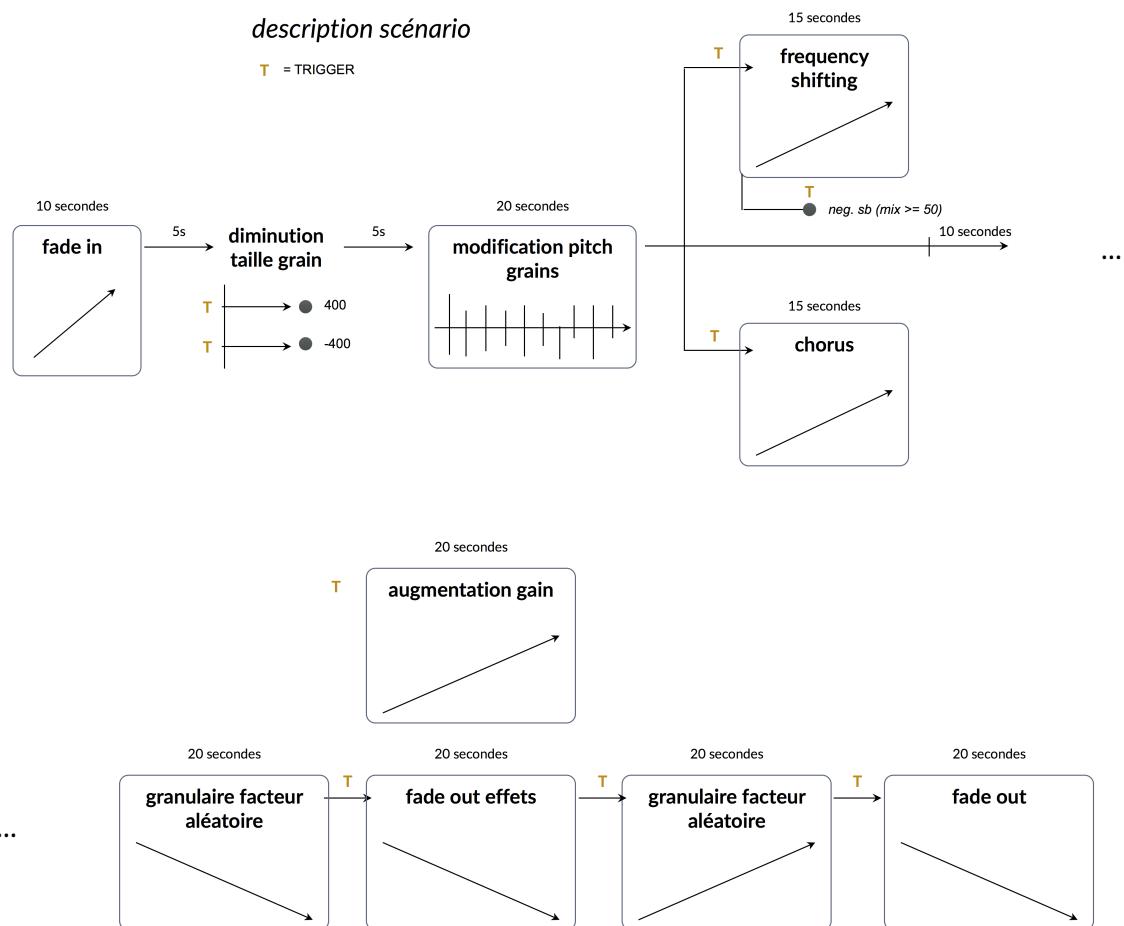


Figure 2 : schéma de scénario, démonstration n°3

Dans ce premier exemple, le scénario (FIGURE 2) est construit de manière simple et linéaire :

² jamoma.org

1. Une automation linéaire ascendante (*fade in*) d'une durée de 10 secondes est appliquée au flux audio de sortie de l'échantillonneur granulaire.
2. Attente de 5 secondes.
3. L'utilisateur doit ensuite choisir entre une modification de la taille du grain de 400 millisecondes et -400 millisecondes (ce qui lui indique une lecture à rebours), ou aucune s'il n'agit pas sur les évènements *triggers* en cliquant sur l'une des deux icônes.
4. Attente de 5 secondes (indépendant du choix précédent).
5. Interpolation des 16 valeurs de hauteur (*pitch*) de chaque grain, de 0 vers les valeurs de destination (indépendantes les unes des autres) - durée de 20 secondes environ.
6. A l'instar de l'étape 3, choix entre l'activation du module de *chorus* ou de *frequency shifting* (ou aucun) – si un choix est effectué – durée de 15 secondes.
7. Attente d'environ 40 secondes (après la fin de l'étape 5 – les effets étant également facultatifs)
8. Automation appliquée au paramètre *random factor* de l'échantillonneur granulaire (durée 20 secondes).
9. Estompelement des effets éventuels appliqués (étape 6), augmentation progressif du gain - déclenchement manuel via *trigger* – durée 20 secondes.
10. Automation *random factor* de nouveau – déclenchement manuel via *trigger* – durée 20 secondes.
11. Fondu final de sortie – déclenchement manuel via *trigger* – durée 20 secondes.

2.3 Méthodologie d'écriture

Dans le cadre de cette démonstration, l'écriture pourrait être décrite comme une écriture de transitions, plutôt qu'une écriture de processus. En effet, le processus *audio* à l'œuvre est constant : il démarre directement à l'ouverture du patch, et se termine uniquement à sa fermeture. Le scénario expose donc des transitions d'état de ce processus, à travers les différents paramètres qui le composent.

L'objectif était dans un premier lieu de parvenir à séquencer de manière souple des évènements sonores relativement complexes et imprévisibles. En effet, l'échantillonneur granulaire présent dans le dispositif Max dépend dans son fonctionnement de deux paramètres aléatoires (fréquence et coefficient appliqués à la position de départ de lecture de chaque grain). Cela signifie qu'il ne jouera jamais deux fois exactement la même chose, et qu'il est en cela impossible de le synchroniser très précisément avec d'autres évènements. C'est dans ce cas de figure qu'une synchronisation manuelle (à l'aide d'un *trigger* – un déclenchement utilisateur) semble être utile. L'écoulement temporel devient alors parfaitement maîtrisé par l'utilisateur à certains endroits du scénario, ce qui amène celui-ci à devenir dynamique.

Dans un second lieu, nous avons voulu expérimenter l'écriture de séquences inspirées des *ossias*, c'est à dire des passages alternatifs pouvant être choisis par l'interprète (aux étapes 3 et 6 du scénario). Bien que l'on ne puisse véritablement parler *d'ossia* au sens strict du terme, dans la mesure où aucune des deux séquences n'est définie « par défaut » et où il est finalement possible de les ignorer toutes les deux et de n'en déclencher aucune, ce mode d'écriture s'est révélé toutefois intéressant dans une optique d'essais comparatifs ou de transformations réactives et/ou parallèles.

2.4 Retours sur expérience

En comparaison avec le système traditionnel des *cue-lists*, sur lequel cette démonstration était au départ basée, i-score apportait déjà à ce stade (dans sa version 0.2) une bien meilleure lecture du scénario, car il permettait notamment de l'appréhender dans sa globalité, sur une échelle de temps bien définie (même si celle-ci peut devenir entièrement dynamique). Il ouvrait ainsi une vraie facilité potentielle d'écriture, qui, en pratique, souffrait beaucoup du manque d'ergonomie et de stabilité du logiciel.

3. Pianotronics #3 – Alain Bonardi (commande SCRIME)

Pianotronics #3 est une pièce pour deux interprètes (piano et électronique temps réel), conçue en hommage à *Archipel 4*, forme ouverte pour piano d'André Boucourechliev (1970). Son fonctionnement repose sur un dispositif de tirages aléatoires de séquences musicales écrites sous forme de blocs de partitions standard. L'originalité et le *challenge* technique qui en découlent viennent du fait que le pianiste ignore au préalable l'ordre des séquences, qui arrivent au fur et à mesure sur l'écran de la partition numérique disposée sur le pupitre de son piano.

Cette pièce d'Alain Bonardi constitue notre toute première expérience d'utilisation d'i-score dans un contexte réel de production en spectacle vivant. Une première ébauche du scénario avait été réalisée lors d'un pré-travail avec le compositeur en février 2015 sur i-score 0.2.3. La pièce a ensuite bénéficié des nouvelles fonctionnalités et de la plus grande stabilité d'i-score dans sa version 0.3, lors d'une deuxième séance de travail en janvier 2016. Celle-ci a abouti à la création de la pièce le 8 avril 2016 à l'occasion du Piano Day, organisé par le SCRIME à la chapelle du CROUS de Bordeaux.

3.1 Dispositif

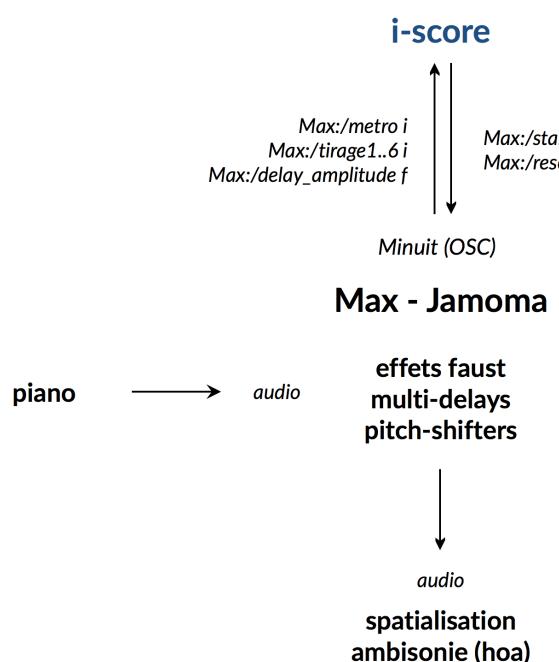


Figure 3 : schéma de dispositif, pianotronics #3

Le son du piano est capté par un ou plusieurs microphones et son signal traité par un ordinateur, à partir d'une collection de modules de traitements numériques conçus par Alain Bonardi à l'aide du langage Faust. Le pilotage de ces effets audio est géré dans Max, par le biais de certains processus automatiques, ou bien par le compositeur lui-même pendant la performance.

i-score reçoit de Max 3 types d'informations (FIGURE 3): l'ordre des séquences générée par le tirage aléatoire préalable (aux adresses */tirage1*, */tirage2*... jusqu'à */tirage 6*), la valeur d'un métronome "dynamique" */metro* chargé d'observer et de réguler les temporalités de chaque séquence, ainsi que la valeur d'amplitude des modules de traitements numériques */delayAmp*. Sa liaison à Max, à l'instar du cas précédent, est assurée par le framework Jamoma.

En termes de flux de sorties, le logiciel n'occupe pas du tout le même rôle que sur la démonstration précédente : il est en effet chargé de piloter la trame générale du scénario, à partir de deux messages : le message */start* (qui génère le tirage aléatoire des séquences) ainsi que le message */reset* (qui indique à Max que la séquence musicale en cours est terminée).

i-score : pianotronics #3

description scénario

T = TRIGGER ? = CONDITION

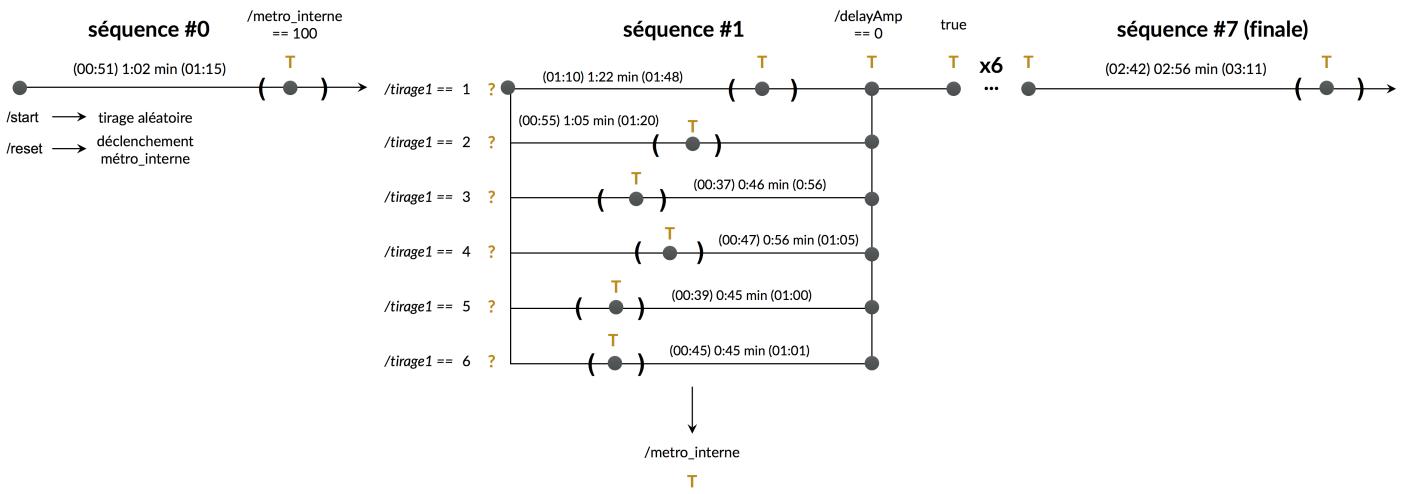


Figure 4 : schéma de scénario, pianotronics #3

Scénario / Méthodologie d'écriture

Le tirage aléatoire ne s'applique ni à la première, ni à la dernière séquence (les séquences 0 et 7), celles-ci restent toujours fixes et invariables. Seules les 6 séquences intermédiaires sont concernées par ce tirage. Pour chacune de celles-ci, le fonctionnement reste identique : i-score interroge la valeur du tirage aléatoire à chaque nouvelle séquence, valide la séquence choisie et invalide toutes les autres. La fin de chaque séquence, quant à elle, est déclenchée sous plusieurs conditions :

- le nombre de battements du métronome dynamique a atteint la valeur maximale spécifiée pour chaque séquence (*/metro*)
- **OU**, le temps maximum spécifié pour chaque séquence a été atteint
- **ET**, le temps minimum spécifié pour chaque séquence a été atteint
- **ET**, dans les deux cas, la valeur d'amplitude des *delays* est égale à zéro.

Chaque séquence est représentée dans le scénario par une contrainte temporelle, dont le début et la fin sont déclenchés dynamiquement. Pour les séquences 1 à 6, elles se démultiplient individuellement en 6 contraintes temporelles, correspondant aux 6 tirages possibles, chacun doté d'attributs qui leur sont propres (condition d'activation, nombre de battements total du métronome dynamique, temps minimum, maximum, et *trigger* de validation). Comme énoncé précédemment, l'un des principes majeurs de la pièce repose sur le *challenge* technique imposé au pianiste, qui dispose pour chaque séquence d'une fenêtre de temps plus ou moins large pour valider sa fin (56 secondes dans le cas de la plus courte, 1 minute et 48 secondes pour la plus longue). Ces fenêtres ont été déterminées empiriquement durant les répétitions en présence de l'interprète et du compositeur, puis fixées sur le scénario i-score à l'aide des outils de fenétrage des évènements *triggers* d'i-score (représentés symboliquement par des parenthèses sur chaque contrainte temporelle). Le pianiste doit idéalement arriver au terme de la séquence en cours entre le temps minimum et maximum impartis. Si la fin intervient avant le temps minimum, la validation de la séquence devra attendre que le curseur temporel l'atteigne, ce qui créera une pause non désirée et un défaut de transition avec la séquence suivante. A l'opposé, si le curseur temporel atteint le temps maximum avant la fin de l'interprétation, la séquence suivante est immédiatement déclenchée, remplaçant la précédente sur la partition numérique du pianiste, qui devra très vite réagir, sous peine d'accumuler du retard sur les séquences suivantes. Le "métronome dynamique" est quant à lui chargé d'observer et de suivre la progression du pianiste dans la partition. Il s'incrémenter en fonction des informations qu'il reçoit (!!!), et valide la fin de la séquence lorsqu'il atteint la valeur indiquée sur le *trigger*. Il est remis à zéro automatiquement à chaque nouveau début de séquence.

Enfin, nous avons conçu le scénario de manière linéaire, avec – outre les séquences immuables de début et de fin – des blocs identiques répétés 6 fois. Il avait été envisagé de représenter ces blocs par une seule contrainte multiple, qui aurait été jouée en boucle (le processus des boucles avait été implémenté et stabilisé dans cette version d'i-score), seulement, nous avons jugé préférable de les garder séquencées, pour disposer d'une meilleure lisibilité quant à la progression globale du scénario, la syntaxe temporelle colorée d'i-score étant particulièrement adaptée et lisible pour cela.

Retours sur expérience

Le logiciel étant toujours en version *alpha* lors de l'écriture et de la performance de *pianotronics #3*, nous sommes restés prudents et ainsi relativement à l'écart d'une trop grande dépendance à ses fonctionnalités, ce qui explique pourquoi son rôle – bien que central – ne s'est pas porté sur le pilotage complet du dispositif (les traitements *audio* notamment ainsi que certains processus interne à Max) qui a été également sécurisé par le compositeur par des interfaces manuelles alternatives dans Max. Néanmoins, la partition qui en ressort semble claire et s'est montrée fonctionnelle lors de la création du *Piano Day*. Enfin, de manière plus globale, il est important de noter que l'évolution d'i-score avec la version 0.3 a permis de gagner beaucoup plus de stabilité, de performances globales, et d'aisance d'écriture, notamment sur la redéfinition des *triggers* et des conditions.

4. ParOral – Georges Gagnéré

ParOral est un projet de lecture augmentée créé par Georges Gagnéré. A l'aide de technologies de reconnaissance vocale et de suivi de texte, le lecteur déclenche par sa voix un décorum visuel, musical et sonore ainsi que certaines transformations vocales dynamiques qui s'activent et évoluent en fonction des situations narratives qu'il parcourt. Georges Gagnéré fait partie des artistes qui ont largement contribué au projet i-score depuis ses débuts (avec le projet Virage, ou ANR INEDIT). Son projet *ParOral*, basé sur le conte d'Andersen, constitue un projet important pour i-score, car il a notamment permis d'explorer et de formaliser un grand nombre de fonctionnalités aujourd'hui disponibles dans le logiciel. La première résidence de travail a eu lieu au SCRIME en mars 2015, et a permis de présenter au public un premier prototype du projet avec i-score 0.2. Un portage - toujours en cours à l'heure actuelle, a été amorcé en mai 2016 avec la version 0.3. La présentation de ce cas d'écriture se focalisera uniquement sur la partie sonore du dispositif.

Dispositif

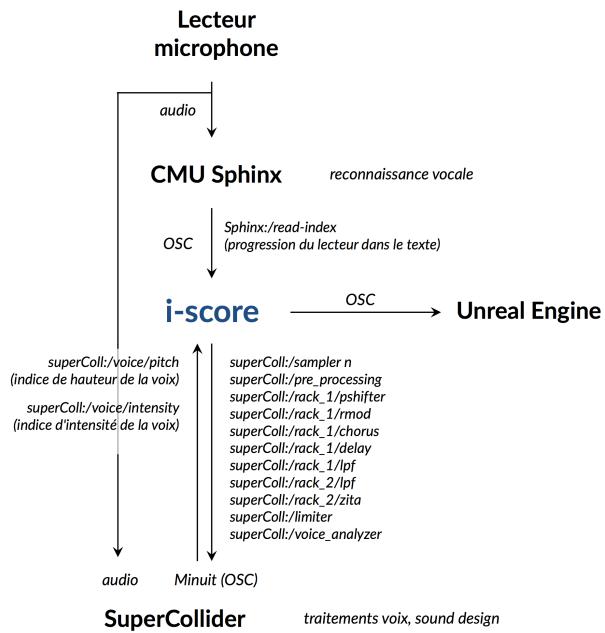


Figure 5 : schéma de dispositif, ParOral

Le dispositif, bien plus complexe que sur les exemples précédents, est constitué de 3 programmes “cibles” :

- Une application audio réalisée avec l’environnement de programmation *SuperCollider*, chargée de piloter les processus d’illustration sonore et de transformations vocales du lecteur.
- Un programme réalisé avec le moteur *Unreal*, chargé de piloter l’animation d’avatars tridimensionnels (prenant la forme de silhouettes, ou d’ombres) placés dans un décorum visuel vidéo-projeté.
- Un programme de reconnaissance vocale, basé sur la bibliothèque *open-source* CMU Sphinx.

i-score reçoit un flux d’entrée provenant du programme “Sphinx”, correspondant à l’incrémantion progressive de l’index des numéros de caractères du texte, représentant la position du lecteur au sein de ce dernier. Il pilote en sortie les programmes *Unreal Engine* et *SuperCollider*, dotés, pour le premier, d’un *plug-in* d’envoi-réception de messages OSC, implémenté par Cédric Plessiet, et, pour le second, d’un *framework* créé par le Réalisateur en informatique musicale (*mgulibSC*) semblable à Jamoma, en cela qu’il permet l’intégration du protocole Minuit.

Scénario / Méthodologie d’écriture

T = TRIGGER

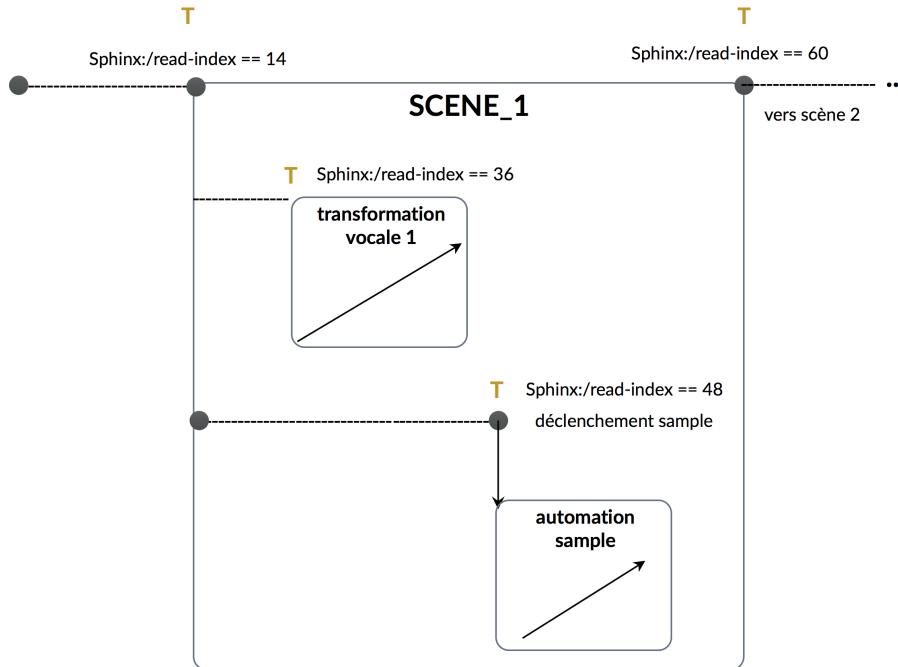


Figure 6 : schéma d'écriture de scène, ParOral

Pour des raisons pratiques et d'organisation du travail, nous avons créé deux scénarios séparés : l'un dédié à la gestion de la partie graphique et visuelle, effectuée par Georges Gagneré, l'autre pour la partie sonore, composée par le RIM, sous la direction du metteur en scène. Ces deux scénarios se retrouvent néanmoins synchronisés naturellement par Sphinx, qui envoie au fur et à mesure des index de position formatés en messages OSC (*Sphinx:/read-index*) à chacune des machines, qui ont également la possibilité de communiquer entre-elles, par le biais d'i-score (avec, à titre d'exemple, des processus de *mapping*).

Les différentes scènes (formalisées sur i-score en sous-scénarios) dans lesquelles sont inclus ces évènements sont dotés – en plus d'un lien séquentiel - d'un lien dynamique (*trigger infini*) avec le *timenode* 0 (le tout premier nœud temporel du scénario), ce qui nous a permis d'effectuer un travail scène par scène, sans avoir à rejouer l'intégralité du scénario. Cette fonctionnalité a été implémentée de manière plus souple dans les versions suivantes d'i-score : il est maintenant possible de lire uniquement le contenu d'un sous-scénario spécifique.

Dans le cas de la partition graphique ou sonore, le projet expose un scénario composé quasi-exclusivement d'évènements et de processus dynamiques, réagissant à la position du lecteur dans le texte ainsi qu'à – éventuellement - certaines propriétés de sa voix (comme l'intensité, ou la hauteur). Le mode d'écriture (FIGURE 6) est ainsi relativement simple, et nous a par conséquent amenés à nous autoriser plus de complexité et de détail sur l'écriture et les relations inter-événemmentielles. Ainsi, pour chaque nouvel évènement dynamique, un *trigger* est positionné sur le *timenode* de départ, soumis à l'évaluation de l'expression choisie (ici, *Sphinx:/read-index == 14*). Bien entendu, nous avons, de la même manière,

cherché à rendre possible l'arrêt dynamique de ces mêmes évènements ou processus, ce qui a fait surgir des difficultés non-négligeables. En effet : il nous était fondamentalement impossible d'anticiper le rythme de l'énonciation du narrateur, et la faible réactivité de Sphinx sur des temporalités très courtes ne nous permettait pas d'extrapoler ce paramètre de manière suffisamment viable. Par conséquent, la fin d'un processus de transformation vocale était déterminée de manière empirique, sur une durée fixe moyenne, à défaut de ne pouvoir être entièrement dynamique. Dans ce contexte, la temporalité des processus prenait le pas sur celle du narrateur, le temps qu'ils se terminent. Le résultat donne toutefois des résultats souvent intéressants, dans la mesure où le lecteur s'adapte très instinctivement à la contrainte temporelle qui lui est imposée, et joue avec les modulations vocales qui surviennent ou disparaissent progressivement.

Retours sur expérience

A l'instar des cas précédents, la lisibilité qu'offre l'écriture sous forme de partition de mise en scène permet de considérer avec plus d'efficacité sa forme et son contenu. Ici, étant donné les grandes possibilités octroyées par le dispositif, doté d'arborescences de paramètres très riches, elle nous a permis de travailler de manière plus intelligible et intelligente entre les différents médias, et de gérer la complexité et les relations entre processus avec plus d'aisance.

5. Le Chant des Nanos – Clément Rossignol-Puech, Laurent Soulié, agence Genre, Antonin Dubuisson, Allain Glykos, Pierre Cochard

Le *Chant des nanos* est un projet Arts et Sciences intermédia visant à mettre en corrélation recherche en nanosciences et musique, graphisme, poésie. Il prend la forme d'une installation sonore (octophonie) et de vidéo-mapping autonome, pilotée par i-score. La création de l'installation s'est déroulée le 13 octobre 2016 au Forum des Arts et de la Culture de la ville de Talence, à l'occasion du lancement du deuxième concours étudiant Arts et Sciences STArt #2.

Dispositif

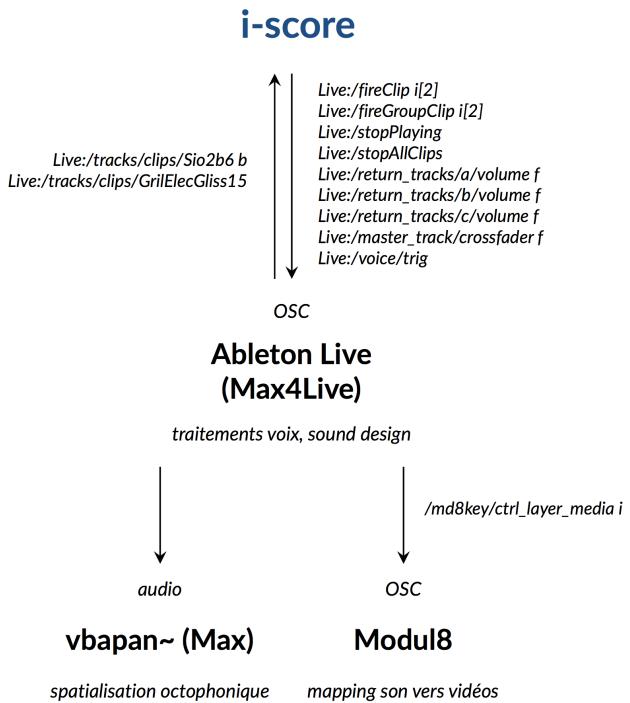


Figure 7 : schéma de dispositif, Le Chant des Nanos

Le dispositif consiste en une communication bidirectionnelle entre i-score et Ableton Live, via le programme d'extension Max For Live, qui est chargé de faire la traduction des messages OSC envoyés par i-score en appels de fonctions de l'API d'Ableton Live (déclenchement de *clips* ou de scènes, paramétrage des volumes audio des pistes de retour, *master crossfader*...). A l'inverse, sont envoyés à i-score des messages destinés à l'informer de l'état de lecture de certains *clips* audio (Live:/tracks/clips/Sio2b6 et Live:/tracks/clips/GrilElecGliss15).

Dans le séquenceur, les clips audios sont empilés dans leurs pistes respectives, et il leur a été attribué une *follow-action* aléatoire, ce qui signifie qu'à chaque fin de lecture d'un *clip* sur une piste donnée, un autre *clip* (déterminé aléatoirement) sera déclenché immédiatement à la suite. Par conséquent, si un premier *clip* sur une piste donnée est déclenché à partir d'i-score, la piste continuera à jouer des séquences aléatoires de *clips* jusqu'à l'arrêt du scénario. A partir de cette méthode, i-score déclenche donc en réalité des séquences de *clips*, plutôt que des *clips* individuels.

Scénario / Méthodologie d'écriture

i-score : chant des nanos

description scénario

T = TRIGGER
? = CONDITION

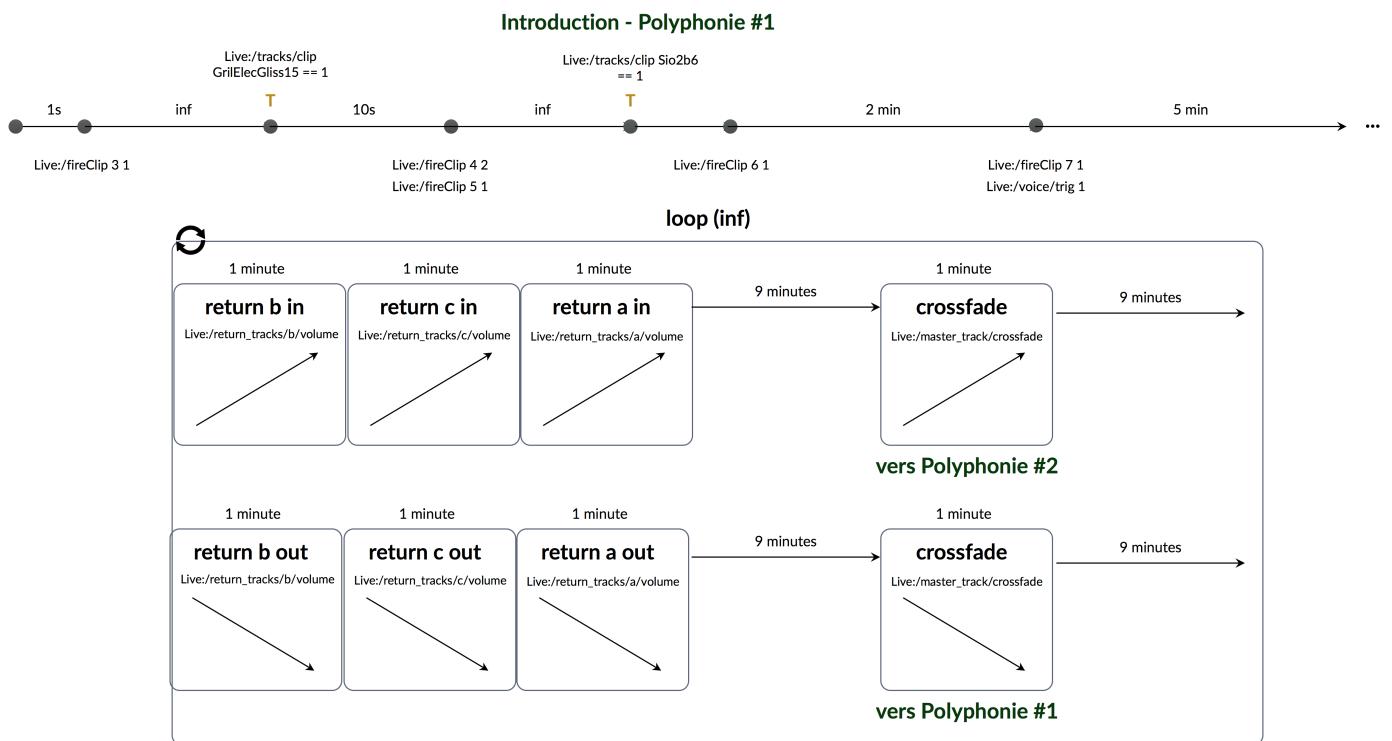


Figure 8 : schéma de scénario, Le Chant des Nanos

Le projet n'avait à l'origine pas été conçu pour être spécifiquement piloté par i-score. En nous décrivant la manière dont il considérait le développement du scénario de l'installation, le compositeur nous a interrogé sur la possibilité de déployer un outil simple permettant la construction d'une telle narration. Ableton Live pouvant être piloté par protocole OSC à l'aide de l'extension M4L ou de scripts Python, nous avons décidé de faire un portage de ce scénario sur i-score, qui a été effectué par le Réalisateur en informatique musicale. Laurent Soulié nous a ainsi adressé la conduite suivante :

- 1. Initialisation de précaution (stop global, stop de tous les clips)**
 - Volume des pistes de retour A, B, C à zéro
 - Master crossfader à gauche
- 2. Mise en route de la première polyphonie**
 - Lancer le clip *GrilElec10* sur la piste 4
 - Attendre le déclenchement aléatoire de *GrilElecGliss15* (piste 4)
 - Au déclenchement de *GrilElecGliss15*, attendre 10 secondes, lancer le clip *Crakels7* (piste 5) et le clip *S3* (piste 6)
 - Au déclenchement de *Sio2b6*, attendre 5 secondes, lancer le clip *Moteur 10* (piste 7)
 - Attendre 2 minutes, lancer le clip *S11* (piste 8)
- 3. Polyphonie 1 en l'état**
 - Attendre 5 minutes
- 4. Mise en route des effets (pendant la polyphonie) (POINT D'ENTREE BOUCLE)**

- Monter à 0dB le volume de la piste de *retour B* en 1 minute
- Monter à 0dB le volume de la piste de *retour C* en 1 minute
- Monter à 0dB le volume de la piste de *retour A* en 1 minute

5. Mise en route de la seconde polyphonie

- Attendre 9 minutes
- Déclenchement du groupe 2 (scène 1)
- *Master crossfader* passe à droite (point B) en 2 minutes

6. Polyphonie 2 statique

- Attendre 9 minutes

7. Estompement des effets

- Baisser à -inf le volume de la piste de *retour B* en 1 minute
- Baisser à -inf le volume de la piste de *retour C* en 1 minute
- Baisser à -inf le volume de la piste de *retour A* en 1 minute

8. Polyphonie 2 statique

- Attendre 9 minutes

9. Transition vers première polyphonie

- *Master crossfader* passe à gauche (point A) en 2 minutes
- Attendre 9 minutes
- **POINT DE SORTIE BOUCLE**

Le scénario est donc composé de 3 grandes séquences : une première, introductory, qui se place sur une temporalité courte (de quelques secondes à quelques minutes pour chacune des contraintes statiques) une seconde et troisième se basant sur une temporalité plus longue (une dizaine de minutes par polyphonie). Ces deux dernières séquences sont placées à l'intérieur d'une boucle globale infinie.

L'écriture sur i-score s'est faite très naturellement, car la conduite était très clairement décrite, et correspondait parfaitement à la syntaxe d'i-score. Les attentes sont formalisées par des contraintes temporelles vides de tout processus, les attentes dynamiques de déclenchement par des évènements *triggers*, et les envois aux pistes de retour ainsi qu'au *master crossfader* par de simples automations linéaires. Enfin les deux dernières séquences sont incluses dans un processus de boucle, qui est doté d'un *trigger* infini sur son *timenode* de fin.

Retours sur expérience

Malgré quelques problèmes de performances lors de l'envoi de certaines données relatives aux courbes d'automations, ainsi que quelques difficultés liées à l'OSC sur M4L, le portage du scénario s'est réalisé sans grande difficulté, et a permis de tester la fiabilité d'i-score sur une temporalité plus longue qu'à l'accoutumée (plusieurs heures) en installation autonome.

Gerer tout via i-score : follow-actions + pattern-matching vers Modul8 ?

10. quarrè

quarrè est une installation sonore immersive et interactive dont le déroulement scénaristique peut être simultanément orienté et interprété (de manière tactile et gestuelle)

par plusieurs utilisateurs, via une application dédiée – destinée aux tablettes et *smartphones*. Un premier prototype de l'installation a été présenté les 22, 23, 26 et 27 septembre 2016 au LaBRI, dans le cadre du festival *Les Campulsations*.

Dispositif

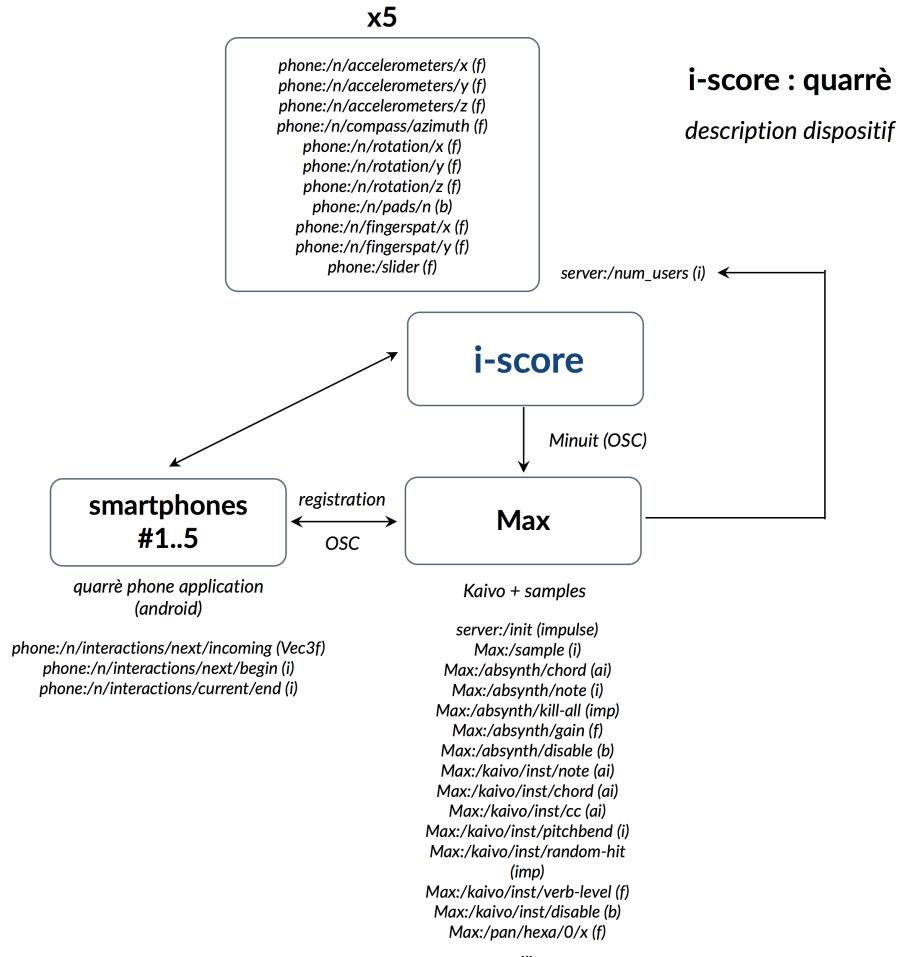


Figure 9 : schéma de dispositif, quarré

L'installation débute préalablement avec l'enregistrement et le référencement des *smartphones* des utilisateurs. Cette première étape (gérée via Max) vise à obtenir un premier lot d'informations : le nombre d'utilisateurs pour la session s'apprêtant à être démarrée, l'adresse IP, les capteurs fonctionnels et disponibles pour chaque appareil. Ces informations sont gardées en mémoire dans Max. La valeur du paramètre “nombre d'utilisateurs” (*server :/num_users*) est transmise à i-score, cette variable étant utilisée à l'intérieur même du scénario pour déterminer certaines activations et attributions des interactions.

Le dispositif sonore de l'installation consiste en un patch Max, contrôlant de nombreux lecteurs d'échantillons, synthétiseurs et modules de spatialisation sonore.

En fonction de la progression du scénario, i-score envoie aux différents *smartphones* connectés une référence et un contexte d'interaction avec la composition (identifiant, durée et pré-notification de l'interaction). Une fois ces données reçues, l'application *smartphone* les interprète et met à jour l'IHM de l'utilisateur en affichant dans la partie supérieure de la

fenêtre : le titre de l'interaction, une brève description des actions à effectuer, et le module d'interaction correspondant, dans la partie inférieure. Pour ce dernier, il peut s'agir d'un module entièrement tactile (**screenshot**) ou sensoriel / gestuel, ou encore hybride.

Lorsqu'une interaction débute (i-score envoie au *smartphone* cible le message *phone:/n/interactions/next/begin identifiant_de_l_interaction* pour lui notifier son déclenchement), le *smartphone* envoie les données pré-ciblées à i-score, qui peut s'en servir pour les manipuler, puis les transmettre aux différents composants du moteur audio, par le biais de *mappings* ou autres processus et/ou évènements.

Scénario

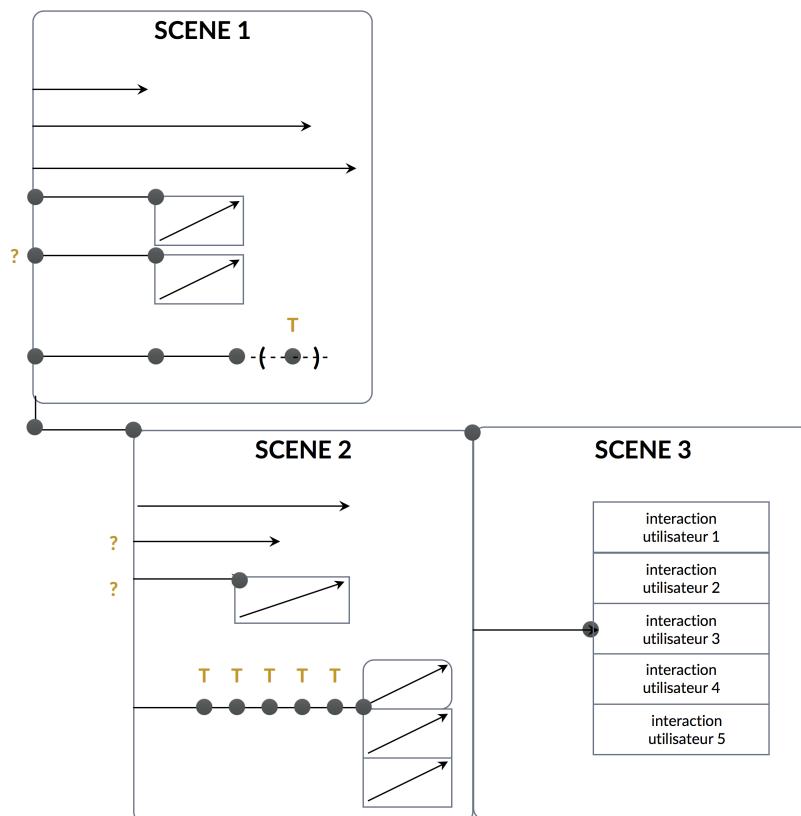


Figure 10 : schéma de scénarisation, quarré

Le scénario a été scindé en 3 scènes, chacune expérimentant une approche interactive différente :

- Scène 1 – “passive-fermée”
- Scène 2 – “active-fermée”
- Scène 3 – “active – ouverte”

La scène A consiste en une scène introductory, au sein de laquelle l'ambiance sonore s'installe progressivement, et où il y a peu d'interactions-utilisateur. Ces quelques interactions sont par ailleurs plutôt “superficielles” : elles n'impliquent pas ou peu de modifications ou de possibilités d'interprétation majeures sur le scénario.

La scène **B** introduit plus d'interactivité, notamment pour l'utilisateur chargé de prendre en main les déclenchements des notes percussives du synthétiseur en modèle physique. Celui-ci devient déterminant dans la mesure où il prend en quelque sorte le rôle de métronome du scénario, sur lequel les autres processus vont venir se greffer au fur et à mesure de la progression.

La scène **C** est une forme ouverte dans laquelle est assignée à chaque utilisateur une interaction fixe et immuable. Il n'y a, mise à part la durée globale de la scène, aucune restriction : les évènements ne sont ni séquencés ni temporisés.

Méthodologie d'écriture

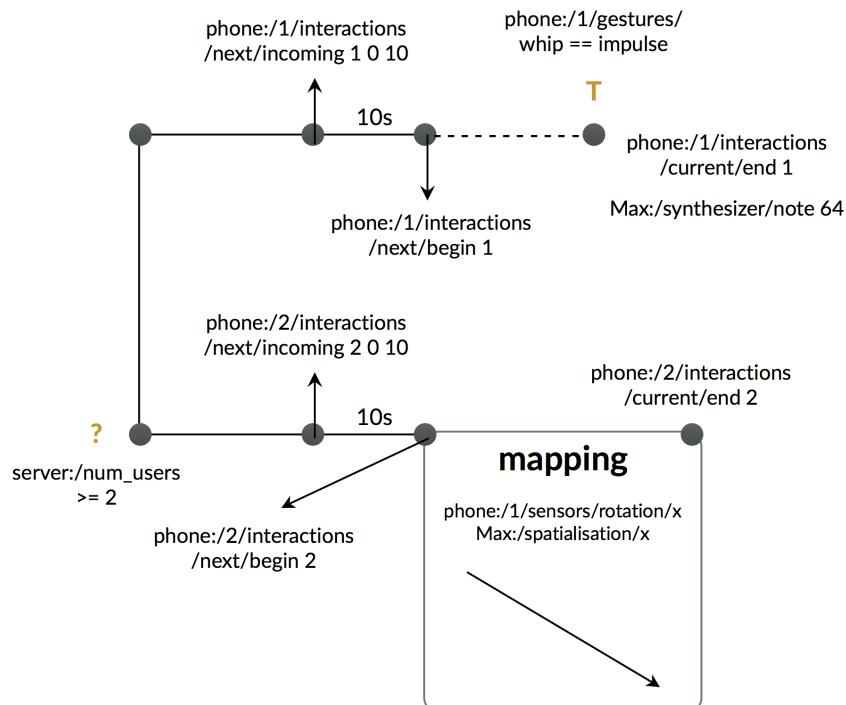


Figure 11 : schéma, exemple d'écriture des interactions, quarré

Les 3 différentes scènes sont représentées par des sous-scénarios qui, dans le cas des deux premiers, se chevauchent au lieu d'être séquencés, permettant d'introduire plus de souplesse et de possibilités de réajustements dans les transitions entre les deux ambiances sonores. Les interactions continues des utilisateurs sont représentées par des processus de *mapping* (les flux d'entrées envoyés par les différents téléphones sont mis à l'échelle, puis envoyés aux modules de traitement audionumériques), les interactions ponctuelles, quant à elles, sont formalisées par des *triggers*. Les différents échantillons sonores lus sont représentés par des contraintes temporelles "vides" marquant pour chacun leur durée, ce qui permet de visualiser les chevauchements.

L'attribution dynamique des interactions a pu être formalisée grâce à des contraintes ou des groupes de contraintes temporelles conditionnelles, réagissant au paramètre "nombre d'utilisateurs" (server:/num_users).

Retours sur expérience

Pas de possibilité d'avoir un processus de *pattern matching* autrement qu'avec un processus JavaScript, nécessitant une connaissance basique du langage, peu performant dans ce type de contextes (signal en impulsions, non continu).

Il aurait été souhaitable de pouvoir implémenter le processus préalable de gestion des connexions directement dans le scénario, de façon à pouvoir lancer ce dernier dès qu'un ensemble de conditions sont réunies.

OSCQuery, TCP vs UDP.

i-score plugin audio -> visualisation des formes d'onde, reprises de lecture à des endroits précis pour faciliter la composition.

11. L'Arbre intégral

L'arbre intégral est un spectacle intermédia dont la thématique gravite autour de l'arbre, de la technologie et du réseau. Il mêle à la fois danse, poésie, musique et vidéo 3D / réalité virtuelle.

Dispositif

i-score : arbre intégral (v. 2b - ambarès)

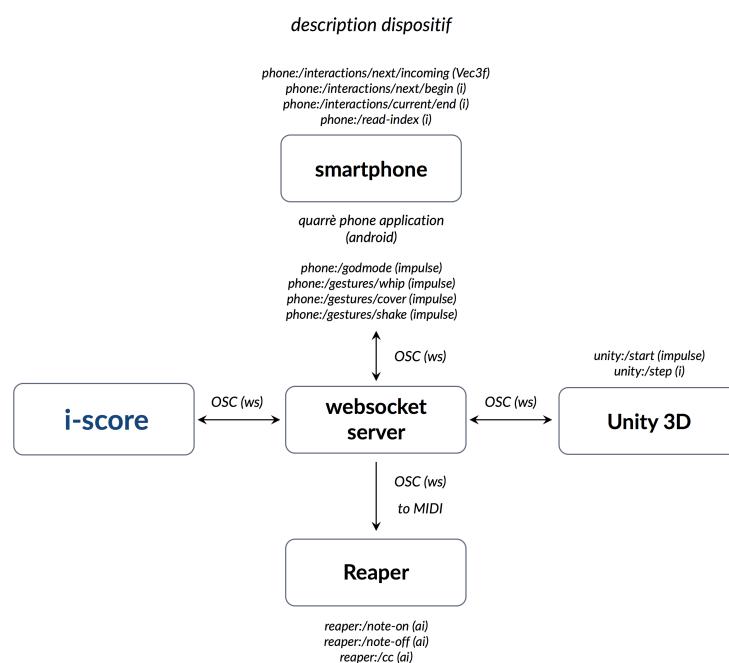


Figure 12 : schéma de dispositif, *L'Arbre intégral*

Le dispositif scénique est constitué de plusieurs appareils, tous reliés en réseau par Wi-Fi :

- Un ordinateur chargé du pilotage du visuel 3D / interactions réalité virtuelle (par l'intermédiaire du logiciel Unity 3D)

- Un ordinateur chargé du pilotage de la partie sonore et musicale (par le biais de Reaper ou d’Ableton Live)
- Un smartphone, utilisé sur scène par le poète dans le contexte d’interactions vocales, doté de l’application **quarrè** (cf. cas précédent)
- Un ordinateur “serveur”, sur lequel sont installés i-score et un serveur local **WebSocket**, chargé de gérer les flux de données échangés entre i-score et les différentes machines.

Ce dernier a été implémenté pour plusieurs raisons : i-score (version 1.0.0 b1) ne gère nativement l’OSC qu’avec le protocole UDP, qui n’est pas sécurisé pour des flux de déclenchement (*on/off*), car le protocole ne peut garantir sa bonne réception par le destinataire. En revanche, il est possible d’instancier un *device* WebSocket, qui, quant à lui, garantit l’ordre et la réception des messages envoyés, au prix d’une latence supplémentaire, qui ne s’est toutefois pas montrée handicapante lors des répétitions et des représentations. Le device Websocket i-score est un *device* client, il ne gère pas de flux de connexions, c’est pourquoi nous avons conçu une application serveur détachée à l’aide du *framework* Qt, ainsi que d’autres applications ou *plug-ins* client pour les différentes machines et logiciels (Unity, Reaper / Live, l’application quarrè).

Scénario / Méthodologie d’écriture

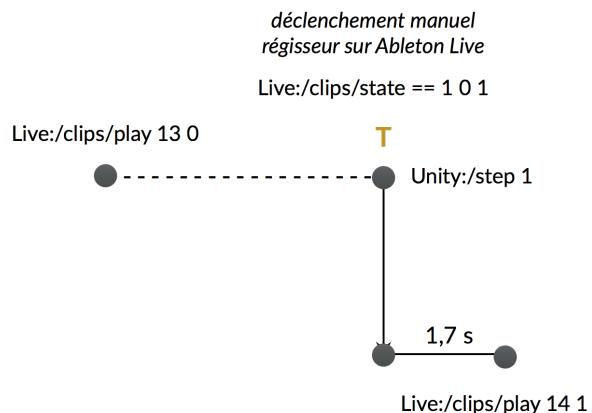


Figure 13 : schéma d’écriture n°1 – démarrage du spectacle, *L’Arbre intégral*

Après initialisation des paramètres de chaque appareil, la lecture du scénario déclenche immédiatement la lecture en boucle d’un clip dans Ableton Live, qui fait office de prologue au spectacle et dure jusqu’à ce que le public soit installé et la lumière éteinte. Le “top” initial, symbolisé sur scène par le mot “*login*” est déclenché manuellement par le régisseur via un contrôleur, lorsque le moment lui semble opportun (FIG. 13). Ableton Live informe i-score du déclenchement du clip, qui déclenchera à son tour le processus vidéo 3D continu sur Unity (*unity:/step 1*) ainsi qu’un premier clip musical dans Live, quelques secondes après. De manière plus générale, le scénario est conçu sous forme de pistes, symbolisées par des séquences de sous-scénarios et de codes-couleur. Les différentes séquences vidéo ne contiennent à ce stade aucun processus, ils figurent dans le scénario dans l’optique d’avoir une visualisation de la progression des processus vidéo et pour permettre des

synchronisations avec les autres médias. En effet, la génération en temps réel des formes graphiques de synthèse et des séquences interactives en réalité virtuelle implique une temporalité non maîtrisable : des latences et des accélérations peuvent survenir en fonction du temps de calcul de la machine, ce qui signifie qu'à ce stade, Unity devient la référence temporelle de tous les processus du scénario, qui viennent se greffer à certaines *keyframes*, formalisées par des *triggers* déclenchés par Unity, par le biais du message *unity:/step*.

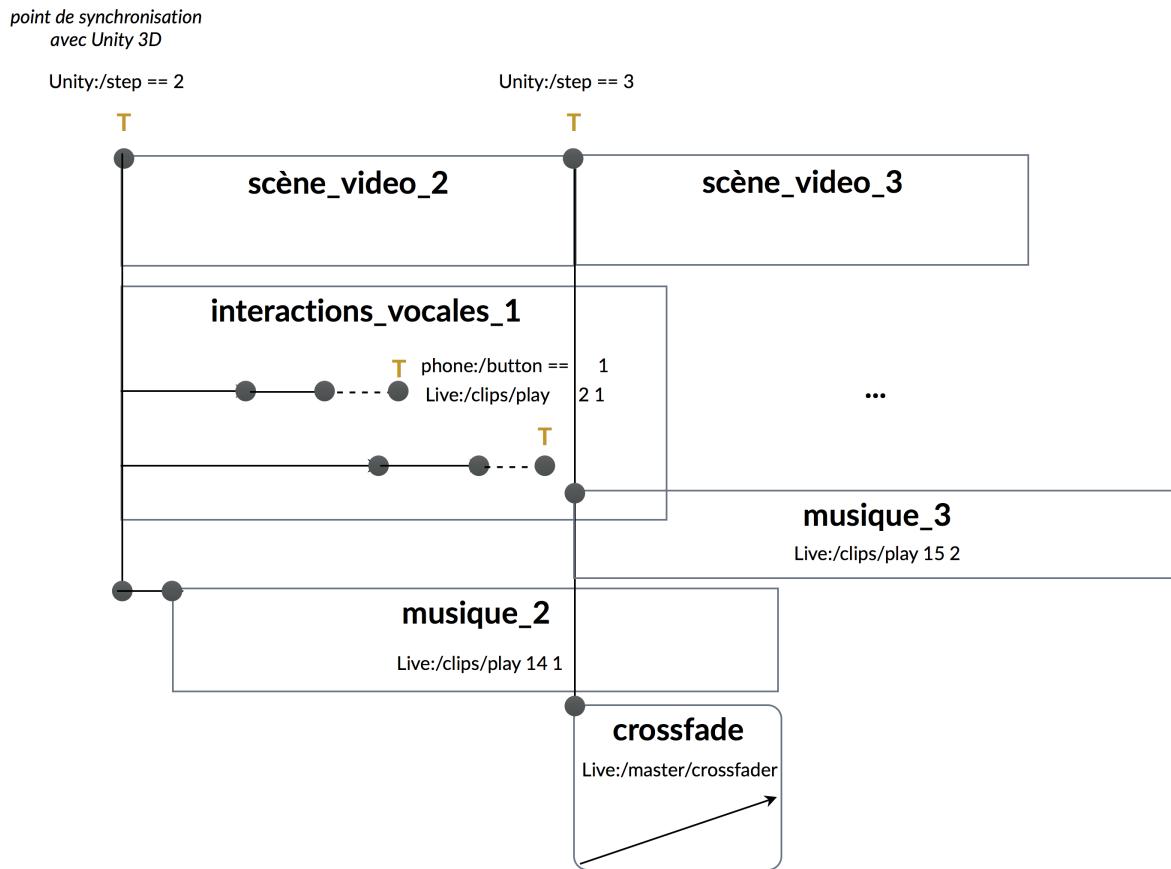


Figure 14 : schéma d'écriture n°2, exemple général de liaisons intermédiaires, L'Arbre intégral

Les séquences d'interaction vocale, gérées par le *smartphone* du poète sur scène, sont symbolisées en bleu sur le scénario, et sont formalisées de la même manière que sur l'installation *quarré* (cf. cas précédent), c'est-à-dire : notification d'une prochaine interaction, activation et validation de l'interaction (*triggers*). Le poète dispose de plusieurs moyens pour exécuter cette dernière : soit par un geste reconnu par l'application, soit en appuyant sur un bouton tactile matérialisé sur l'IHM du téléphone. Ces validations déclenchent par l'intermédiaire d'i-score des *clips* audio, des activations de processus de spatialisation...

Les séquences musicales, symbolisées en rouge, sont formalisées par des sous-scénarios dont la longueur correspond à la durée des *clips* lancés sur Ableton Live. Ils peuvent contenir des automations de volume ou de *crossfading*, permettant d'ajuster le mixage et les transitions entre différentes séquences.

Retours sur expérience

A ce stade, nous n'avons pas encore eu la possibilité de développer une écriture des dispositifs "lumière", qui serait tout à fait possible avec une interface appropriée (i-score ne gère pas encore le protocole DMX ainsi que les appareils USB qui permettent son traitement et acheminement). Néanmoins, cet exemple concret montre les potentiels d'une écriture scénique interactive qui pourrait contrôler tous types d'appareils utilisés dans le contexte d'une performance artistique. Par ailleurs, la démarche expérimentale du spectacle ayant conduit à certaines difficultés quant à la représentation de l'enchaînement des évènements entre les différents médias, le fait de disposer d'une partition scénique de référence a pu permettre d'optimiser le dialogue entre artistes et techniciens, car elle s'est montrée suffisamment explicite graphiquement pour être appréhendée par toute l'équipe.

CONCLUSION GENERALE

Biblio :

boids

Mémoire de Master de Thomas Meyssonnier

Jean-Michaël

Autre article