OUTILS D'ÉCRITURE SPATIALE POUR LES PARTITIONS INTERACTIVES

Jean-Michaël Celerier
Univ. Bordeaux, LaBRI, UMR 5800, F-33400 Talence, France.
Blue Yeti, F-17110 France.
jcelerie@labri.fr

Myriam Desainte-Catherine
Univ. Bordeaux, LaBRI, UMR 5800, F-33400 Talence, France.
CNRS, LaBRI, UMR 5800, F-33400 Talence, France.
INRIA, F-33400 Talence, France.
myriam@labri.fr

Jean-Michel Couturier
Blue Yeti, F-17110 France.
jmc@blueyeti.fr

RÉSUMÉ

L'écriture de partitions interactives évolue pour inclure du contenu spatial. En effet, dans de nombreuses pratiques de création, musicales ou scénographiques, un travail sur des paramètres à plusieurs dimensions apparaît nécessaire. Cela peut être dû à de la spatialisation du son ou à la nécessité de répondre à des contraintes physiques sur une scène. Il est possible de définir des sous-espaces utiles pour la définition de ces paramètres en tant que zones d'espace : les courbes de trajectoires paramétriques en sont un exemple. Nous présentons les modèles possibles pour l'écriture de contenu spatial. Un modèle mixte est choisi pour l'implémentation dans le séquenceur i-score. Il offre des moyens d'écriture très généraux ainsi que la possibilité de spécialiser et d'optimiser certains cas lorsque cela s'avère nécessaire. Cette implémentation comporte des outils d'édition, de visualisation, ainsi que deux possibles sémantiques d'exécution offrant une forte interopérabilité avec les scénarios interactifs.

1. INTRODUCTION

Les pratiques de création multimédia incluent par définition des composantes spatiales. Celles-ci peuvent se manifester dès qu'un des types de donnée manipulé possède plus d'une dimension. La simple présence de multiples paramètres dans une création multimédia permet d'avoir une notion d'espace dans lequel on met en correspondance un paramètre avec un autre. Il est donc possible de définir de manière géométrique de nombreux éléments courants avec lesquels les compositeurs travaillent.

Cependant, la plupart des outils permettant un travail d'écriture spatiale sont soit spécifiques à la création de trajectoires pour des sons, dans un cadre musical, soit orientés vers la création de jeu vidéo ou de formes spécifiques de médias. De par leur spécialisation pour des projets donnés, ils ne permettent pas toujours une utilisation facile dans un contexte autre que celui qui a été prévu au départ.

Nous avons ici la démarche inverse : nous cherchons des méthodes génériques, permettant de manipuler tout type de données spatiales pour ensuite pouvoir les appliquer aux cas particuliers de la création de scénographies interactives avec des éléments musicaux. Cela a pour avantage une plus grande flexibilité, et une moindre nécessité d'utilisation de différents outils. Cependant, c'est au prix de performances non optimales, que nous essayons néanmoins d'optimiser par la suite.

Sera exposé un état de l'art des outils de création spatiale, puis nous présenterons deux modèles, l'un géométrique et l'autre paramétrique, qui permettent de réaliser différents types de compositions, notamment liées aux relations que l'on peut avoir entre plusieurs zones définies géométriquement. Ces modèles, intégrés au séquenceur i-score, sont appliqués à deux études de cas : une chorégraphie de robots Metabots, et une application de réalité augmentée audio, Sonopluie.

Des détails et problématiques propres à l'implémentation sont présentés, ainsi que les modes de manipulation et de rendu. Notamment, l'intégration d'un processus spatial à la sémantique d'exécution actuelle de i-score est discutée.

2. EXISTANT

Nous présentons ici plusieurs applications opérant principalement dans un domaine spatial, à but créatif. Les thèmes étudiés sont d'abord l'utilisation de contenus spatiaux en musique, en scénographie, et en jeux vidéos, ainsi que les différentes méthodes d'écriture et de représentation qui existent. Une présentation exhaustive de l'état actuel de l'écriture spatiale en musique est donnée dans [16]. Notamment, la question de la notation dans le cadre de partitions impliquant des éléments spatiaux y est abordée.

Ces partitions, graphiques, peuvent être spatiales uniquement dans leur représentation, mais peuvent aussi indiquer des manières d'interpréter dans l'espace, notamment à l'aide de symboles spécialisés [15]. Une taxinomie des possibilités de création dans l'espace en musiques électroacoustiques est présentée par Bertrand Merlier, notamment dans l'ouvrage *Vocabulaire de l'espace en musique électroacoustique [25]*.

Des outils logiciels existent pour ces partitions – ils sont

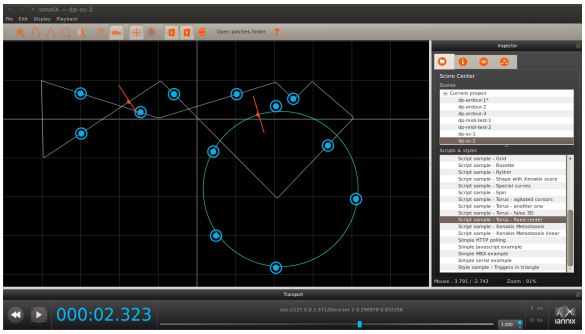

Figure 1. Le séquenceur graphique IanniX, inspiré de l'œuvre de Iannis Xenakis.

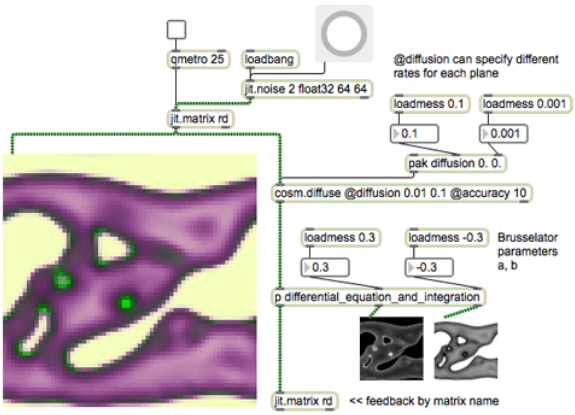

Figure 2. Un patch COSM dans Max/MSP.

souvent spécialisés. Par exemple, la bibliothèque ENP [23] permet de concevoir des partitions graphiques à l'aide d'un éditeur lui aussi graphique et d'un langage basé sur LISP.

Une des problématiques actuelles pour la représentation de l'écriture musicale est celle du geste, et de son lien avec la partition : comment annoter le geste du musicien avec précision ? Et, inversement, comment à partir d'un geste créer un son correspondant ? Ces questions sont abordées dans la description de Soundstudio 4D [31], dans le cadre d'un système de conception de trajectoires pour spatialisation à l'aide d'interactions en trois dimensions.

Une autre question est l'association entre l'aspect graphique et le résultat. Ainsi, des outils tels que Holo-Edit et Holo-Spat [9] permettent de travailler avec des trajectoires, mais sont orientés vers un travail sur de l'audio. C'est dû à la nécessité de composer en ayant conscience à chaque instant des fortes contraintes techniques du moyen de restitution de l'œuvre ; par exemple, la configuration de hauts-parleurs pourra être fixée par l'auditorium et l'artiste doit alors travailler avec.

Le logiciel IanniX [20] (fig. 1) dispose aussi de nombreuses possibilités d'écriture spatiale : les partitions sont des ensembles d'éléments graphiques définis paramétriquement ou bien à l'aide d'un langage de programmation dédié, que des curseurs vont parcourir. L'information de position de chaque curseur est envoyée en OSC, ce qui permet l'intégration à d'autres logiciels, ainsi que des compositions par feedback.

Une méthode d'écriture de la spatialisation par contraintes est proposée par Olivier Delerue dans MusicSpace [13]. Cela permet une approche déclarative de l'écriture de partition, en spécifiant des contraintes telles que « deux objets ne doivent jamais être à plus de deux mètres l'un de l'autre » ou bien « l'angle entre deux objets et l'auditeur doit être supérieur à 90 degrés ». Les objets peuvent être notamment des sources sonores. Une édition graphique de ces contraintes est proposée, et elles sont représentées en termes de cercles et de segments reliant les objets qu'elles contraignent.

Des données spatiales peuvent aussi être utilisées directement pour créer des mises en correspondance (*mappings*) sonores. C'est le cas de la bibliothèque Topos [26], qui permet de capter le mouvement de danseurs et d'en extraire des informations pouvant être utiles pour la conception de pièces de musique interactives. Une fois que le mouvement du danseur est capturé via un périphérique externe, il devient possible d'extraire des informations telles que le volume occupé par le danseur, sa vitesse, ou bien diverses mesures relatives à l'évolution de deux ou quatres points dans le temps, comme l'instabilité ou les collisions entre différentes parties du corps. Ces données peuvent ensuite être réutilisées dans Pure Data pour de la génération de musique.

Un modèle plus complet d'espace en trois dimensions est fourni par COSM [34], en fig. 2. Implémenté dans Max/MSP, il offre une grande richesse d'écriture mais n'a pas été mis à jour pour les dernières versions de Max. En plus de lieux et de trajectoires, il est possible d'écrire l'interaction dans une certaine mesure, ainsi que la communication entre différents agents. Une nouveauté est la possibilité de travailler avec des champs définis mathématiquement. Ces champs peuvent varier dans le temps et être sonifiés par la suite.

Une approche de contrôle spatial est possible via le logiciel Blender, qui sert à l'origine à réaliser des images et films de synthèse. Blender a déjà été utilisé à d'autres occasions pour de la spatialisation de son [28] et de la simulation acoustique [8]. Blender peut être contrôlé via une API Python et il est notamment possible de déplacer des éléments et tester la présence de collisions ou d'autres propriétés géométriques. Néanmoins, cela se fait à la vitesse de son moteur d'exécution qui est fixée à 60 Hz. Les messages reçus entre deux trames sont accumulés.

Le monde des jeux vidéos dispose aussi d'outils adaptés à l'écriture spatiale : l'interface OpenAL a été développée à l'origine pour offrir aux jeux une couche d'abstraction permettant de bénéficier de spatialisation simplement en donnant une position et une orientation à des sources sonores ponctuelles. Cette position et orientation peuvent évoluer dans le temps. Par la suite les implémentations sont libres d'utiliser les méthodes qu'elles souhaitent pour réaliser la spatialisation : cela peut aller d'une simple panoramique gauche-droite à l'utilisation de HRTFs, comme le fait la bibliothèque OpenAL-soft.

De manière générale, les moteurs et éditeurs de jeux, tels que Unreal Engine ou Unity offrent des possibilités

d'interaction très riches pour disposer et animer ces sources dans le temps.

Enfin, il existe de nombreuses alternatives aux traditionnels claviers, souris, joysticks pour les modes et périphériques d'entrée et d'interaction permettant de décrire des données spatiales dans le monde physique. Par exemple, il existe plusieurs possibilités de composition musicale à l'aide de tables interactives comme la Reactable [22] et différentes approches dérivées qui peuvent être spécifiquement axées sur la spatialisation du son [29]. On notera aussi ShapeTape [18], un ruban déformable qui permet d'obtenir sa torsion en 32 points pour reconstruire la courbe en 3D dans un logiciel. Un vocabulaire de mouvements possibles (torsions, étirements, poussées, ...) est défini par les auteurs.

3. CAS D'UTILISATION

Afin d'avoir des possibilités de test de notre système, deux cas pratiques d'applications nécessitant de l'écriture de contenus spatiaux ont été étudiés.

Dans les deux cas, on veut écrire une famille de scénarios interactifs qui manipulent des types d'objets fixés : des sources sonores virtuelles dans le premier cas et des robots dans le second cas.

3.1. Example : Sonopluie

Cette application interactive utilise de la géolocalisation par téléphone. Plusieurs sources sonores sont apposées dans un espace, dans lequel les participants se déplacent. Cet espace virtuel est mappé au monde réel lors du parcours. La mise en correspondance est réalisée en positionnant les zones sur une carte issue d'un service web. Le système mesure la distance de chaque participant aux sources, et va jouer les sources sonores plus ou moins fort en fonction de cette distance.

Il manque un outil simple d'écriture pour disposer les zones dans l'espace, leur associer des sons et des comportements. Un des objectifs du projet est d'offrir de l'interaction et de l'évolutivité dans les scénarios : par exemple, après s'être approché d'un point donné, on voudrait pouvoir désactiver des points précédents et activer des points suivants, ou avoir des points qui se déplacent.

Un scénario d'example, tel qu'implémenté dans i-score, consiste en :

- Une source de géolocalisation exposée dans un arbre Minuit.
- Un moteur audio gérant la spatialisation via positionnement de sources. Nous utilisons ici la bibliothèque OpenAL [19], dans son implémentation `OpenAL-soft`. En effet, elle est supportée sur de nombreux appareils mobiles courants. Elle aussi est implémentée en Minuit.
- Un projet i-score qui consiste en :
  - Une boucle principale contenant l'application complète.

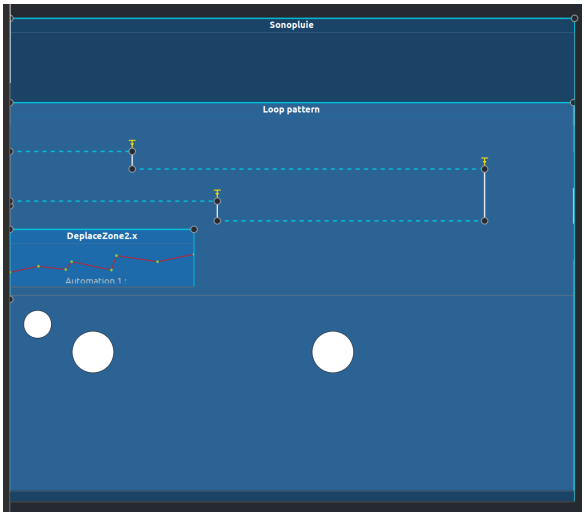

Figure 3. Un scénario de type Sonopluie

  - Dans cette boucle, un processus Espace ou sont instanciées les zones sous forme de disque. Pour cet exemple il y a trois zones. Ce processus est détaillé dans la partie suivante.
  - Un nombre de pointeurs statiques est alloué à l'avance. Leur position correspond à celle des sources de géolocalisation.
  - On va réaliser deux tests : distance au centre d'une zone, collision. La collision va servir à activer le son, et la distance à contrôler un dosage d'effet, par exemple une réverbération.
  - Cette boucle contient aussi un scénario muni de plusieurs contraintes temporelles. Ces contraintes ont pour mission d'animer une des zones en la déplaçant au cours du temps, et d'activer la troisième zone uniquement quand les deux premières ont été visitées.

Ce scénario est visible en figure 3.

3.2. Example : robots

Ce projet consiste en la réalisation d'une chorégraphie de robots musicaux, puis de drones. On dispose d'une flotte de robots open-source Metabot [1] . Ces robots sont contrôlés en vitesse : on écrit sur le port série des commandes telles que `dx 5` pour indiquer une vitesse de $5\,\mathrm{cm\,s^{-1}}$. Un logiciel a été conçu pour faire une mise en correspondance d'un arbre Minuit [2] vers le port série. En parallèle, un logiciel de simulation, présenté en fig. 4 est développé à l'aide d'OpenFrameworks, qui expose le même arbre Minuit. Ce logiciel de simulation permet d'afficher les robots et de détecter les collisions, pour empêcher qu'elles ne se produisent en pratique, avec des robots coûteux. On veut notamment utiliser le rythme de marche des robots pour produire de la musique.

1 . `http://metabot.fr/`

2 . Minuit [12] est un protocole de communication basé sur OSC permettant la découverte via réseau des paramètres contrôlables d'un logiciel multimédia.

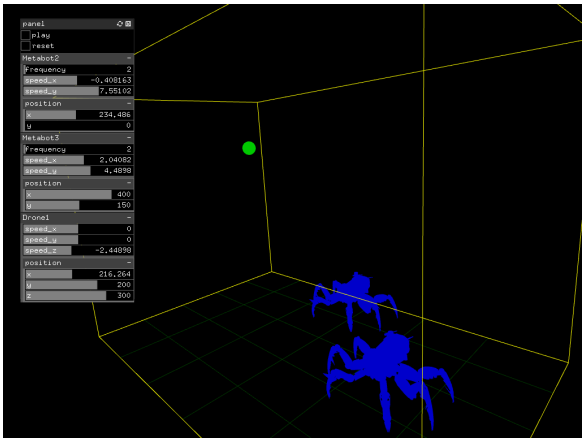

Figure 4. Logiciel de simulation de metabots communiquant avec i-score

Chaque robot va posséder une trajectoire dans le plan. Cette trajectoire pourra potentiellement changer en fonction des choix du metteur en scène, ou bien du comportement des autres robots : par exemple, si un robot tombe en panne en face d'un autre, un changement de trajectoire doit être opéré (ou bien le robot doit s'arrêter). Pour ce faire, un autre logiciel est présent sur le réseau : ce logiciel détecte à chaque instant la position de chaque robot et l'expose via Minuit. L'auteur peut donc utiliser cette information pour prévoir des cas d'urgence, comme mettre fin au spectacle gracieusement si trop de robots tombent en panne.

Un des problèmes principaux est le contrôle en vitesse des robots : il faut à partir d'une trajectoire décrivant une position, calculer la dérivée dans les deux ou trois directions pour l'envoyer aux robots. Dans l'attente d'un modèle de calcul plus complet et générique, cela peut être fait dans le processus qui gère des trajectoires.

4. MODÈLE

4.1. Conception

Les objectifs pour l'intégration des notions spatiales dans i-score sont :

- Une intégration avec l'écosystème existant.
- Support de l'animation : Une description fine des évolutions dans le temps (ou en fonction d'autres paramètres).
- La possibilité d'écrire des scènes 2D, 3D basiques potentiellement en utilisant des guides (sources cartographiques, images).
- Des performances les plus proches possible du temps réel.

4.1.1. Modèles pour l'écriture spatiale

Plusieurs modèles ont été envisagés pour écrire des scénarios spatiaux.

Une première possibilité repose sur les méthodes de description qualitative de l'espace, telles que RCC-8 [17]. Elles consistent en l'extension des relations de Allen [2], à plus d'une dimension. Cependant de telles méthodes ne s'avèrent pas efficaces si on désire décrire des dispositions précises, avec des métriques sur les objets que l'on veut manipuler. Par exemple, on peut à l'aide de méthodes qualitatives écrire "A est à l'extérieur de B et lui est tangent, et A et B sont contenus dans C". Mais les pratiques appellent plus souvent une écriture de la forme "A est un cercle de $1u$ centré en $(3, 3)$ dans une pièce de $5u$, et un pointeur B est situé en $(2, 3)$" ou u est une unité de distance ; on peut par la suite obtenir les relations RCC à partir des relations métriques tandis que l'inverse n'est pas possible.

Une seconde possibilité est l'approche orientée objet [32] : on dispose d'une classe abstraite représentant une forme. Les classes dérivées implémentent des méthodes de dessin et de calcul optimisées pour des formes concrètes : sphère, cube, etc. L'avantage est la possibilité d'avoir des exécutions très rapides dans certains cas. Par exemple, calculer la distance entre deux cercles est trivial. En revanche, cette méthode manque doublement de flexibilité :

- Il est difficile de rajouter de nouveaux types de calculs, car il faut modifier tous les types pour rajouter le calcul (par exemple un calcul d'aire pour une zone).
- Si un calcul implique deux (ou plus) zones abstraites, on assiste à une explosion combinatoire lorsque l'on désire rajouter une nouvelle zone car le calcul doit alors être défini entre la nouvelle zone et toutes les zones pré-existantes.

Une troisième possibilité, numérique, et permettant un maximum de flexibilité, a été étudiée : on demande à l'utilisateur de remplir la fonction caractéristique d'une zone à l'aide d'un langage de programmation. Par la suite, on peut tester pour chaque point de l'espace son appartenance à la zone, en ayant la possibilité de faire des approximations. Cela permet d'implémenter des objets spatiaux qui sont difficilement possibles à réaliser uniquement de manière mathématique, avec des objets dont la définition contient des boucles ou des conditionnelles, voire des facteurs aléatoires. En revanche, on perd des possibilités d'analyse par la suite : on n'a en effet pas de forme générale des zones ainsi créées et on ne peut au mieux qu'effectuer des tests, coûteux en temps de calcul, entre zones. Il est cependant toujours possible de prendre un échantillon des points de la zone pour appliquer une méthode de maillage telles que la triangulation de Delaunay [27] qui produit un objet sur lequel on peut réaliser plusieurs opérations plus simplement. Les bibliothèques CGAL [6] et VTK [30] proposent toutes deux des implémentations performantes de cette triangulation. Enfin, les performances ne sont pas adaptées à des évolutions rapides en temps réel : c'est le cas de la géométrie dynamique. cela peut marcher en dimension 1 et pour des fonctions avec un nombre relativement important de points à calculer, mais est lourd en ressources en dimension 2 et quasiment impensable sur un CPU en temps réel en dimension 3.

Finalement, l'utilisation de modèles mathématiques dédiés est possible. En général, il s'agit des modèles paramétriques, ou géométriques. Les modèles géométriques sont

notamment très utilisés dans le domaine de la conception assistée par ordinateur, tandis que les modèles paramétriques sous-tendent la plupart des outils de spatialisation musicale, notamment à l'aide des fonctions spline, ainsi que de nombreux outils pour l'animation et les images de synthèse.

4.1.2. Dimensions non spatiales

Une question qui se pose pour le choix du modèle est celle de l'interaction entre les zones et le temps. En effet, dans des logiciels comme IanniX, on dispose de courbes paramétrisées par des paramètres externes ou par une horloge. Cette approche permet d'avoir dans une seule vue toute l'information possible : on dessine la courbe pour un intervalle inclus dans l'ensemble de paramétrisation. Cependant, l'objectif des processus spatiaux est de posséder plus d'une dimension de paramétrisation. La question de l'affichage d'une variation au cours du temps se pose alors. On peut penser à afficher un dégradé qui serait complètement opaque pour une valeur de paramètre donné, et serait de plus en plus transparent lorsque l'on s'en éloigne ; un exemple d'affichage d'animation d'objet 3D au cours du temps est par exemple donné dans [10]. Il est aussi possible d'afficher uniquement la trajectoire du centre (ou d'un point donné) de l'objet au cours du temps, mais cela nécessite de pouvoir la calculer. Ce n'est pas possible dans le cas d'une zone définie avec des contraintes très faibles. Par exemple, pour un modèle géométrique, défini par $x < 0$, un demi-plan, on ne peut le définir.

Une autre possibilité est d'avoir des boîtes séparées pour définir les animations. Cela permet plus de clarté, et autorise à associer une seule trajectoire à plusieurs objets plus facilement : La boîte contenant la trajectoire va écrire à chaque tic d'horloge sur une adresse OSC. Les zones spatiales peuvent ensuite aller chercher la valeur de cette adresse ; l'auteur peut utiliser ces coordonnées dans l'écriture de l'interactivité de ces zones.

4.1.3. Espaces non cartésiens

Une considération importante est le support des objets définis autrement que dans le système de coordonées cartésiennes. Le système le plus courant sera les coordonées polaires, mais la question des espaces sur des types de paramètres autres a aussi été posée : par exemple, un travail sur des espaces colorimétriques requiérerait des outils adaptés car la conversion entre plusieurs de ces espaces, comme RVB et $\mathrm{L^*a^*b^*}$ est souvent non-linéaire. Pour ce faire, il est possible d'utiliser des outils tels que ceux fournis par la bibliothèque Jamoma [12], offrant notamment des types de données de haut-niveau dans la *DataspaceLib*.

4.2. Processus spatiaux dans i-score

Nous choisissons d'utiliser plusieurs des modèles exposés précédemment. Le cœur de l'approche repose sur un modèle mixte, géométrique et orienté objet pour la définition de scènes spatiales, ainsi que sur des processus représentant des objets paramétriques, en raison de leur practicité pour l'écriture de trajectoires, très courantes dans les domaines d'application ciblés.

Il n'y a pas de focalisation directe sur le son : les processus opèrent toujours sur des données numériques quelconques.

4.2.1. Processus Espace

Dans le premier cas, on définit une classe abstraite de zone, qui contient une liste d'équations et d'inéquations. Ces équations et inéquations servent à contraindre la zone. Par exemple, on peut donner l'équation cartésienne d'un cercle, ou d'une parabole.

De là, il est possible d'offrir un comportement générique pour l'affichage et les relations entre plusieurs zones, mais il est aussi possible de spécialiser la classe de zone à des fins d'optimisations.

Ces zones sont contenues dans un processus Espace. Le processus Espace consiste en :

— Une liste de dimensions bornées.
— Une liste de zones.
— Une liste d'opérations entre zones.
— Une fenêtre (*viewport*).

Lorsque la définition d'une zone est donnée, on sépare en deux ses inconnues. Pour les exemples qui suivent, on se place dans un espace à deux dimensions x, y.

Par exemple, dans le cas de l'équation d'un disque :

$$(u - x_0)^2 + (v - y_0)^2 \leq r^2 \tag{1}$$

On permet à l'utilisateur d'attribuer à chaque inconnue, u, v, x_0, y_0, r, soit une dimension, soit un paramètre. Les zones prédéfinies par héritage offrent une décomposition simple par défaut. Ainsi, pour le disque, on utilise comme dimensions :

$$\begin{cases} x \to u \\ y \to v \end{cases} \tag{2}$$

.

L'utilisateur peut par la suite utiliser pour les paramètres une application de la forme :

$$\begin{cases} 10.0 \to x_0 \\ 10.0 \to y_0 \\ \texttt{/adresse/OSC} \to r_0 \end{cases} \tag{3}$$

Actuellement, les primitives prédéfinies sont disque et pointeur. Le pointeur correspond à :

$$\begin{cases} u = x_0 \\ v = y_0 \end{cases} \tag{4}$$

Sont prévues par la suite des implémentations spécifiques au moins pour les droites, segments, plans et demi-plans, polygones, cercles, sphères, et tores.

Un paramètre peut être une constante, ou bien être un nœud de l'arbre des paramètres i-score et donc provenir d'une source externe : OSC, MIDI, etc. Cela permet l'interactivité : il est possible de mapper des capteurs externes

à des positions, ou à n'importe quel autre paramètre. Il est aussi possible d'utiliser l'écoulement du temps à l'exécution pour faire de l'animation spatiale.

Enfin, on peut dans une certaine mesure définir des objets paramétriques :

$$\begin{cases} u = t \\ v = sin(t) \end{cases} \tag{5}$$

avec t un paramètre externe, pouvant être le temps. Cependant dans un tel cas, le processus ne considère qu'un point à la fois car le fonctionnement est par contraintes. Une autre approche paramétrique plus adaptée aux besoins courants de l'écriture spatiale est décrite en section 4.2.2. Il est important de noter que la seule équation $v = sin(t)$ ne suffit pas. En effet, si on suppose qu'on applique la même transformation qu'en 2, ici, la variable d'espace x est non-contrainte. On a donc $y = k$ avec k une constante à un instant donné. Le résultat sera une zone ayant pour forme une droite horizontale qui variera verticalement selon la fonction sinus. De la même manière, une zone qui ne contraindrait aucune des dimensions correspond à l'intégralité de l'espace.

L'autre implémentation :

$$v = sin(u) \tag{6}$$

est entièrement statique car elle ne dépend pas de paramètres externes.

Ensuite, il est possible de définir des calculs de relations entre les différentes zones. De la même manière que pour la définition des zones, on opère avec une méthode générique qui peut ensuite être sous-classée pour gérer des cas soit très courants, soit ne pouvant s'exprimer simplement de manière mathématique, ou enfin pouvant être optimisés par une implémentation en *C++*. Par exemple, il est possible d'obtenir l'information de collision entre zones. Pour le cas du pointeur on va simplement évaluer les valeurs actuelles des informations de dimension par rapport aux autres zones, ce qui est une opération très rapide. De même pour les zones de types connus : il est possible d'exhiber des calculs spécialisés pour chaque relation entre deux types de zones. Enfin, pour des cas impliquant une zone générique, non typée, nous pouvons chercher l'existence de solutions au système composé de l'ensemble de leurs équations. Ceci nécessite cependant un solveur capable de résoudre des systèmes d'inéquations. Un outil potentiel pour cette application est *nlopt* [21], qui permet de minimiser des systèmes non-linéaires.

Une autre approche pour les collisions est d'utiliser les méthodes issues du jeu vidéo, avec des notions de cube englobant ou sphère englobante. De nombreuses approches plus perfectionnées pour la gestion des collisions sont présentées pour référence dans [24]. Des moteurs physiques tels que *Bullet* ou *Open Dynamics Engine* [5] pourront par la suite être utilisés à cette fin.

Le résultat des calculs est ensuite exposé dans l'arbre i-score, pour être réutilisé par la suite afin de concevoir d'autres zones, ou bien en tant que sortie appliquée à un autre logiciel ou matériel.

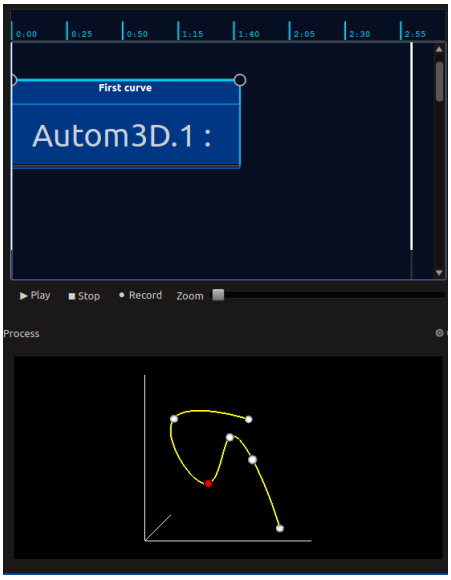

Figure 5. Une spline 3D dans i-score.

i-score étant un environnement ouvert, il est aussi possible de sortir des contraintes du langage mathématique pour les calculs. En effet, un processus Javascript est offert : il permet de réaliser des opérations complexes à chaque tic d'horloge. Il est alors possible de récupérer des informations d'un processus spatial, de les traiter en Javascript, puis de réutiliser leur résultat au tic suivant. Les automations sont utilisables de la même manière. Néanmoins, à terme, un modèle de calcul permettant du chaînage dans un seul tic sera nécessaire.

4.2.2. Automations multi-dimensionnelles

En plus du processus Espace, des possibilités d'écriture paramétrique, plus simples et adaptées à l'écriture de trajectoires, sont aussi présentes.

Il y a plusieurs types d'automation dans i-score :

— Les automations à une dimension sont une fonction de transfert du temps t vers un paramètre u. On note $f : t \in [0;1] \rightarrow D(u)$ ou D est le domaine de définition de la valeur du paramètre ou du service, tel que décrit dans [3].

— Les applications à une dimension sont une fonction de transfert entre deux paramètres u, v. On note $f : D(u) \rightarrow D(v)$. Les automations et applications à une dimension sont des fonctions définies par partie ; chaque partie peut être une fonction vide, un segment de droite, ou bien la fonction $f : x \mapsto x^k$. Des fonctions supplémentaires peuvent être rajoutées par le biais de plug-ins.

— Les automations à deux ou trois dimensions : nous utilisons ici la bibliothèque *vtk* pour afficher et éditer une spline à l'écran, à l'aide de l'objet : `vtkSplineWidget`. Ces processus sont visibles en fig. 5

Dans les cas temporels, le temps d'un processus est celui de sa contrainte parente. On a donc pour une para-

métrisation sur $t \in [0; 1]$, $t = 0$ au début de la contrainte temporelle, et $t = 1$ à la fin de cette contrainte.

Il est possible de choisir entre l'écriture d'un n-uplet de coordonnées, ou bien de plusieurs composantes sur différentes adresses de l'arbre i-score. Cela soulève la question d'un langage de plus haut niveau sur les paramètres, qui rendrait ces choix transparents pour l'utilisateur.

5. IMPLÉMENTATION

La majeure partie de la complexité de l'implémentation provient de la gestion des zones génériques. Une tentative d'implémentation pour les calculs génériques sur les zones a été réalisée à l'aide du CAS *GiNaC* [4]. Cependant, cela ne s'est pas révélé être adapté : en effet la bibliothèque n'est pas thread-safe ce qui empêche de réaliser des calculs déportés sur un autre fil d'exécution. L'implémentation actuelle est donc basée sur le parseur de fonctions mathématiques de la bibliothèque *vtk* [30]. D'autres alternatives ont été envisagées : SymbolicC++ [33], GNU Octave [14], Sage [3] .

5.1. Sémantique d'exécution du processus Espace

La sémantique générale d'exécution d'i-score est donnée dans [11]. Nous rappelons simplement qu'i-score permet d'ordonnancer temporellement des processus qui peuvent être définis dans des plug-ins. Lors de la lecture, une horloge globale va demander récursivement à tous les processus du scénario qui sont en cours d'exécution quel est leur état actuel. Un état est un ensemble de messages qui peuvent être envoyés par réseau, comme un rappel (*cue*).

Ainsi, s'il y a rétroaction, elle se fait au tic d'après. Deux étapes d'exécution sont présentées en fig. 6 et fig. 7.

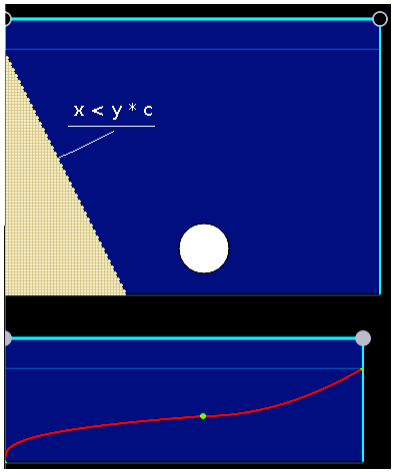

Figure 6. Processus spatial à $t = 0$s. La courbe d'automation est appliquée au paramètre c de l'équation de la zone triangulaire.

L'exécution du processus spatial se déroule de la manière suivante :

3. http://www.sagemath.org/

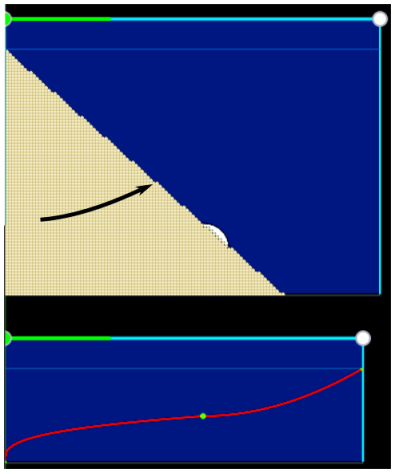

Figure 7. Processus spatial durant l'exécution, à $t = 10$s. c a augmenté, et la zone triangulaire est mise à jour en conséquence. Il est possible de récupérer l'information de collision entre le disque et la zone pour déclencher des évènements ailleurs dans i-score.

— Pour chaque zone dont un des paramètres est défini par une adresse externe, récupérer la valeur actuelle de cette adresse.
— Effectuer tous les calculs entre éléments de la zone puis les renvoyer dans l'état.

Dans le cas d'un processus générique, ces calculs peuvent prendre du temps. Ainsi, un mécanisme de requête / réponse sur plusieurs fils d'exécution a été implémenté. Cela permet de ne pas bloquer toute l'exécution du programme ; en revanche le résultat de l'exécution sera décalé de plusieurs tics. Si le processeur est surchargé, les requêtes sont ignorées pour éviter un blocage complet de l'application.

Une autre sémantique est possible : i-score dispose d'un arbre de paramètres interne qu'il est aussi possible de manipuler. Les processus peuvent exposer des paramètres de contrôle à cet arbre, qui sont disponibles à l'édition. i-score est donc introspectible depuis son interface graphique. Notamment, le processus spatial expose les valeurs des paramètres présents pour chacune des zones. Par exemple, pour un disque dont l'équation a été vue en 1, les paramètres x_0, y_0, r sont exposés dans l'arbre interne. De même, les résultats des calculs définis sont exposés. Il est possible de configurer le processus pour que les calculs enregistrés soient réalisés à la réception d'un message sur le nœud correspondant, et mis à jour dans leurs nœuds respectifs.

Cette question, qui est fondamentalement la différence entre un mode *push* et un mode *pull*, se pose aussi pour les autres objets d'i-score. Par exemple, le processus Automation contient une adresse à laquelle il écrit ses valeurs. Cependant, il est aussi possible de simplement laisser s'exécuter la courbe et de permettre à d'autres outils d'aller chercher l'information à son adresse dans l'arbre interne.

Cela pose la question, à plus long terme, d'un graphe de calculs si on désire avoir des rétroactions et des mises

en correspondances complexes. Actuellement, il est possible de réaliser des calculs chaînés à la main, en créant des adresses spécifiques et en ordonnant les processus : l'exécution de ces calculs est globale. Mais par la suite, un modèle de calcul plus complet sera nécessaire pour rendre manifeste les liens qui peuvent exister entre différents processus, à la manière d'environnements tels que OpenMusic [7], Max/MSP ou PureData.

5.2. Rendu

On affecte chacune des dimensions de l'espace à une dimension fixée de la zone graphique : x, y pour l'instant avec pour objectif du fonctionnement en 3D.

L'affichage se fait de manière spécialisée pour les types connus : cela permet d'utiliser les primitives de la bibliothèque Qt à des fins d'optimisation. Pour les types génériques, on réalise pour l'instant une sur-pixelisation, en évaluant la formule pour plusieurs points et en plaçant un rectangle si le point vérifie toutes les contraintes.

Comme c'est une opération lourde, chaque zone de ce type effectue le rendu dans un thread séparé. Lorsqu'un calcul termine, les rectangles ainsi calculés sont envoyés à la bibliothèque d'affichage.

C'est important pour avoir un rendu fluide à l'exécution. Le nombre d'images par seconde du processus spatial sera potentiellement faible, mais il n'y aura pas de blocage de l'interface graphique pendant le rendu.

D'autres méthodes sont à étudier, par exemple en utilisant des algorithmes de triangulation qui ont été évoquées plus tôt. Cela aurait l'avantage de produire un format de données que les cartes graphiques peuvent consommer beaucoup plus aisément.

Enfin, la question de l'affichage pour des dimensions non-spatiales, telles que le temps ou une échelle de couleur, reste ouverte.

5.3. Édition

La création d'objets se fait actuellement par un panneau de configuration. L'édition va modifier pour les formes connues des paramètres définis.

Les opérations de base sur les zones sont déplacement, rotation et mise à l'échelle.

On applique les transformations aux variables d'espace de l'objet.

— La translation et la mise à l'échelle sont de simples transformations affines.

— La rotation par un angle θ dans le plan s'exprime par le changement de coordonnées suivant en dimension deux :

$$\begin{cases} x' = x\cos\theta - y\sin\theta \\ y' = x\sin\theta + y\cos\theta \end{cases}$$

Une formule générale de la rotation en n dimensions est donnée dans [1] et pourra être appliquée par la suite.

On peut ensuite contrôler ces transformations à la souris, à l'aide d'outils, comme le font la majorité des logiciels d'images de synthèse et de jeux vidéo, ou bien y avoir accès depuis l'arbre de paramètres.

6. CONCLUSION

L'écriture spatiale dans un cadre scénographique ou musical présente de nombreux choix en terme de modélisation, d'implémentation, et de représentation. Nous avons présenté un état de l'art des outils d'écriture spatiale dans ces domaines, dont une version plus complète est présente dans le rapport du projet de recherche OSSIA. Par la suite, les différentes méthodes pour aboutir à une écriture spatiale précise sont présentées et discutées. Le choix d'implémentation pour le logiciel i-score se porte sur deux modèles : le premier est un modèle mixte orienté géométrie, permettant des définitions de zones spatiales génériques ainsi que spécialisées, dont l'intérêt repose dans sa capacité à représenter une scène spatiale dont on peut extraire des informations sur les relations entre les objets qui la composent. Le second est un modèle paramétrique traditionnel adapté à l'écriture de trajectoires. L'utilisation d'objets génériques n'offre actuellement pas des performances optimales. Cependant, le fait de travailler dans le domaine mathématique offre à terme les plus grandes possibilités pour des améliorations et optimisations. Ces deux modèles sont utilisés dans des projets de scénographie interactives mêlant du son ainsi que d'autres medias.

Ces modèles sont tous deux adaptés à un travail dans des espaces cartésiens, sur des paramètres linéaires. Cependant, un travail reste à faire sur un outil permettant de traiter des paramètres ou espaces de paramètres différents, comme les espaces colorimétriques. Cela implique un travail sur un typage plus fort des paramètres manipulés dans i-score. Une dynamicité de l'écriture pourra aussi être intéressante : par exemple, créer des zones lors de l'exécution pourrait offrir de nouvelles perspectives d'écriture.

Un travail sur l'extension en trois dimensions des splines, les surfaces NURBS, pourra aussi étendre les possibilités de la méthode, notamment en permettant une intégration avec des logiciels dédiés à la création de contenus 3D.

7. REMERCIEMENTS

Les auteurs tiennent à remercier les élèves ayant réalisé le logiciel de simulation pour Metabots : Maxime Paillassa et Akané Levy. Ces travaux sont issus du projet de recherche OSSIA, financé par l'Agence Nationale de la Recherche sous la référence ANR-12-COORD-0024.

8. REFERENCES

[1] Antonio Aguilera and Ricardo Pérez-Aguila. General n-dimensional rotations. 2004.

[2] James F Allen. Towards a general theory of action and time. 23(2) :123–154.

[3] Pascal Baltazar, Théo de la Hogue, and Myriam Desainte-Catherine. i-score, an interactive sequencer for the intermedia arts.

[4] Christian Bauer, Alexander Frink, and Richard Kreckel. Introduction to the ginac framework for symbolic computation within the c++ programming language. *Journal of Symbolic Computation*, 33(1) :1–12, 2002.

[5] Adrian Boeing and Thomas Bräunl. Evaluation of real-time physics simulation systems. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, pages 281–288. ACM, 2007.

[6] Jean-Daniel Boissonnat, Olivier Devillers, Monique Teillaud, and Mariette Yvinec. Triangulations in cgal. In *Proceedings of the sixteenth annual symposium on Computational geometry*, pages 11–18. ACM, 2000.

[7] Jean Bresson, Carlos Agon, and Gérard Assayag. OpenMusic : visual programming environment for music composition, analysis and research. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 743–746. ACM.

[8] Emil Brink, Eskil Steenberg, and Gert Svensson. The verse networked 3d graphics platform. In *Linköping Electronic Conference Proceedings. SIGRAD Conference*, pages 44–48. Citeseer, 2006.

[9] Benjamin Cabaud and Laurent Pottier. Le contrôle de la spatialisation multi-sources-nouvelles fonctionnalités dans holophon version 2.2. *Proc. Actes des Journées d ?Informatique Musicale*, 2002.

[10] Dan Casas, Margara Tejera, J. Guillemaut, and Adrian Hilton. 4d parametric motion graphs for interactive animation. 19(5) :762–773.

[11] Jean-Michaël Celerier, Pascal Baltazar, Clément Bossut, Nicolas Vuaille, Jean-Michel Couturier, and Myriam Desainte-Catherine. Ossia : Towards a unified interface for scoring time and interaction.

[12] Théo De La Hogue, Julien Rabin, and Laurent Garnier. Jamoma modular : une librairie c++ dediee au developpement d ?applications modulaires pour la creation. *Proc. of the 17es Journées d ?Informatique Musicale, Saint-Etienne, France*, 2011.

[13] Olivier Delerue. Spatialisation du son et programmation par contraintes : le système MusicSpace.

[14] John Wesley Eaton, David Bateman, and Søren Hauberg. *Gnu octave*. Network thoery, 1997.

[15] Emile Ellberger, Germán Toro Perez, Johannes Schuett, Giorgio Zoia, and Linda Cavaliero. Spatialization symbolic music notation at ICST.

[16] Dominique Fober, Jean Bresson, Pierre Couprie, and Yann Geslin. Les nouveaux espaces de la notation musicale. In *Journées d'Informatique Musicale*.

[17] Alfonso Gerevini and Bernhard Nebel. Qualitative spatio-temporal reasoning with RCC-8 and allen's interval calculus : Computational complexity. In *ECAI*, volume 2, pages 312–316.

[18] Tovi Grossman, Ravin Balakrishnan, and Karan Singh. An interface for creating and manipulating curves using a high degree-of-freedom curve input device. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 185–192. ACM.

[19] Garin Hiebert. Openal 1.1 specification and reference. 00010.

[20] Guillaume Jacquemin, Thierry Coduys, and Matthieu Ranc. Iannix 0.8. pages 107–15.

[21] Steven G Johnson. The nlopt nonlinear-optimization package, 2014.

[22] M Kaltenbranner, Sergi Jorda, Gunter Geiger, and Marcos Alonso. The reactable : A collaborative musical instrument. In *Enabling Technologies : Infrastructure for Collaborative Enterprises, 2006. WETICE'06. 15th IEEE International Workshops on*, pages 406–411. IEEE.

[23] Mika Kuuskankare and Mikael Laurson. Expressive notation package. 30(4) :67–79.

[24] Ming Lin and Stefan Gottschalk. Collision detection between geometric models : A survey.

[25] Bertrand Merlier. *Vocabulaire de l'espace en musiques électroacoustiques*.

[26] Luiz Naveda and Ivani Santana. "topos" toolkit for pure data : exploring the spatial features of dance gestures for interactive musical applications.

[27] L Nonato, Rosane Minghim, MCF Oliveira, and Geovan Tavares. A novel approach for delaunay 3d reconstruction with a comparative analysis in the light of applications. In *Computer Graphics Forum*, volume 20, pages 161–174. Wiley Online Library, 2001.

[28] Natanael Olaiz, Pau Arumi, Toni Mateos, and David Garcia. *3D Audio with CLAM and Blender ?s Game Engine*. na, 2009.

[29] Yuya Sasamoto, Michael Cohen, and Julian Villegas. Controlling spatial sound with table-top interface. In *Awareness Science and Technology and Ubi-Media Computing (iCAST-UMEDIA), 2013 International Joint Conference on*, pages 713–718. IEEE.

[30] Will J Schroeder, Bill Lorensen, and Ken Martin. *The visualization toolkit*. Kitware, 2004.

[31] James Sheridan, Gaurav Sood, Thomas Jacob, Henry Gardner, and Stephen Barrass. Soundstudio 4d : A VR interface for gestural composition of spatial soundscapes. In *ICAD*.

[32] Paul S Strauss and Rikk Carey. An object-oriented 3d graphics toolkit. In *ACM SIGGRAPH Computer Graphics*, volume 26, pages 341–349. ACM, 1992.

[33] Kiat Shi Tan, Willi-Hans Steeb, and Yorick Hardy. *SymbolicC++ : An Introduction to Computer Algebra using Object-Oriented Programming : An Introduction to Computer Algebra using Object-Oriented Programming*. Springer Science & Business Media, 2012.

[34] Graham Wakefield and Wesley Smith. *Cosm : A toolkit for composing immersive audio-visual worlds of agency and autonomy*. Ann Arbor, MI : MPublishing, University of Michigan Library. 00007.