OUTILS D'ÉCRITURE SPATIALE DANS I-SCORE

Jean-Michaël Celerier
Organisme
Adresse électronique

Myriam Desainte-Catherine
Organisme
Adresse électronique

Jean-Michel Couturier
Organisme
Adresse électronique

RÉSUMÉ

Le résumé doit être placé en haut de la colonne gauche et doit contenir entre 150 et 200 mots.

1. INTRODUCTION

Nécessité d'un outil de conception spatiale généralisé répondant aux besoins de la muséographie de la musique spatiale de la robotique des outils de spectacle

Présentation d'un modèle adapté pour l'exécution, intégré avec les développements précédents -> quelle interaction de l'espace et du temps ?

On cherche donc à offrir une intégration entre du temps et des structures spatiales.

Problème de l'écriture et des outils graphiques.

Problème de la visualisation de l'écoulement du temps en 2D (animation).

1.1. Existant

Une présentation de l'état actuel de l'écriture spatiale en musique est donnée dans [5]. Notamment, la question de la notation dans le cadre de partitions impliquant des éléments spatiaux est abordé. Ces partitions 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 [4]. Une taxinomie des possibilités de création dans l'espace en musiques électro-acoustiques est présentée par Bertrand Merlier dans [10]. Elle est étendue dans l'ouvrage *Vocabulaire de l'espace en musique électro-acoustique[9]*.

Des outils logiciels existent pour ces partitions – ils sont souvent spécialisés. Par exemple, la bibliothèque **ENP!** (**ENP!**)[8] permet de concevoir des partitions graphiques telles qu'en fig. **??** 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 notamment 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[13], 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 HoloEdit et HoloSpat permettent de travailler avec des trajectoires, mais sont extrêmement spécialisés pour des objets audio. C'est notamment du à la nécessité de composer en ayant conscience à chaque instant des fortes contraintes techniques du moyen de restitution de l' ?uvre. Il serait intéressant d'utiliser ces trajectoires pour contrôler non pas des sources sonores mais des éléments dans des espaces de paramètres quelconques.

Le logiciel IanniX[6] (fig. **??**) 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.

Une méthode d'écriture de la spatialisation par contraintes est proposée par Olivier Delerue avec le système MusicScpace[3]. Cela permet une approche déclarative à 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 mappings sonores. C'est le cas notamment de la bibliothèque Topos[11], 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.

Enfin, il convient de noter la richesse pour ce qui est des modes d'entrée et d'interaction. Par exemple, il existe plusieurs possibilités de composition musicale à l'aide de tables interactives comme la Reactable[7] et différentes approches dérivées qui peuvent être spécifiquement axées sur la spatialisation du son[12].

Un modèle plus complet d'espace en trois dimensions est fourni par COSM[14]. Implémenté dans Max/MSP, il

offre une grande richesse d'écriture. 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 mathématiquements. 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 peut être contrôlé via une API Python et il est notamment possible de déplacer des éléments et tester pour des 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.

Enfin, le monde des jeux vidéos dispose aussi d'outils adaptés à l'écriture spatiale : la bibliothèque 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'un simple panning gauche-droite à l'utilisation de HRTFs, comme le fait la bibliothèque OpenAL-soft.

2. EXAMPLES

2.1. Example : sonopluie

Utilisation de OpenAL

Description du scénario :

On a plusieurs sources sonores dans un espace. Plusieurs personnes se déplacent et sont géolocalisées. Lorsqu'elles se rapprochent d'une source, elles l'entendent plus fort.

On désire utiliser i-score pour l'écriture de tels scénarios. Par la suite, il faut scénariser : par exemple, une fois qu'une zone a été passée on veut activer d'autres zones pour faire un parcours et non une simple balade.

2.2. Example : robots

Avec logiciel de simulation à côté (simule pannes, etc.).

On conçoit une chorégraphie de robots et drones.

Ils ont chacun une trajectoire qui peut évoluer dans le temps, en fonction de ce qu'il se passe.

Les drones portent des hauts-parleurs tandis que les robots font du rythme en tapant des pieds.

3. MODÈLE

3.1. Conception

On veut :

— une intégration avec l'écosystème existant.
— Décrire finement des évolutions dans le temps (ou en fonction d'autres paramètres).
— pouvoir écrire des scènes 2D, 3D basiques potentiellement en utilisant des guides (GMaps, image).
— animer soit via le temps soit via d'autres paramètres (mapping).

3.1.1. Choix du modèle

Plusieurs modèles ont été envisagés pour écrire des scénarios spatiaux. Une première approche s'est portée sur les méthodes de description qualitative de l'espace, telles que RCC-8. 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 préciser "A est à l'extérieur de B et lui est tangent, et A et B sont contenus dans C". Mais il est plus commode d'écrire "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 distances métriques tandis que l'inverse n'est pas possible.

Une seconde méthode, permettant un maximum de flexibilité, a été étudiée : demander à l'utilisateur de remplir la fonction caractéristique d'une zone en code. 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 notamment d'implémenter des objets spatiaux qui sont difficilement possibles à réaliser uniquement de manière mathématique, avec notamment des objets dont la définition contient des boucles ou des conditionnelles. 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 entre zones. Enfin, les performances ne sont pas adaptées à des évolutions rapides en temps réel ; cela peut marcher en dimension 1 et pour des fonctions avec un nombre relativement important de points à calculer, mais est très lent en dimension 2 et impensable en dimension 3.

Une troisième question qui se pose 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 (souvent le temps). Cette approche permet d'avoir dans une seule vue toute l'information possible. Cependant, ici nous pouvons avoir 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 [2] qui serait complètement opaque pour le t donné, et serait de plus en plus transparent lorsque l'on s'en éloigne. Ou bien uniquement afficher la trajectoire du centre de l'objet au cours du temps, mais cela nécessite de pouvoir la calculer. Ce n'est pas dans le cas d'une zone définie avec des contraintes très faibles. Par exemple, pour une zone définie par $x < 0$, un demi-plan, on ne peut le définir.

Une autre possibilité est d'avoir des boites séparées pour définir les animations. Cela permet plus de clarté, et permet aussi d'associer une seule trajectoire à plusieurs objets plus facilement : la trajectoire va écrire sur l'adresse `/trajectoire/position` que les zones spatiales iront

chercher à chaque tick d'horloge ; elles peuvent par la suite utiliser les coordonnées ainsi obtenues comme bon leur semble.

Nous offrons dans le logiciel les deux possibilités, mais l'affichage dans le premier cas n'est pas implémenté (le temps sera ceci dit pris en compte lors de l'exécution).

3.2. Description

Approche paramétrique, non focalisée sur le son.

On définit des zones par un ensemble d'équations que l'on peut paramétriser - chaque équation est une contrainte.

Processus d'espace : possède un viewport. Un viewport possède des dimensions. Ces dimensions peuvent être spatiales, et il peut y avoir une dimension temporelle.

La dimension temporelle est en fait une dimension de mapping pour l'exécution : (citer article OSSIA) la fonction d'exécution de i-score prend le temps en entrée, et sort un état. Cependant on pourrait imaginer que cette fonction prendrait d'autres valeurs en argument, on aurait ainsi un mapping simple à plusieurs dimensions. On affecte chacune de ces dimensions à un espace graphique : x, y, (par la suite z), potentiellement animation ainsi que des bornes potentielles.

Dans ce viewport, on définit des zones par un ensemble d'équations. Par exemple, $x < y; x + y <= 2 + a$ forme une zone. On associe chaque variable de la zone soit à des composants de l'espace défini, soit à des paramètres. Un paramètre peut avoir une valeur par défaut, ainsi qu'une adresse présente dans l'arbre de paramètres i-score.

Le logiciel fournit aussi des zones par défaut, pour lesquelles un rendu plus propre peut être offert, en utilisant les primitives du moteur de rendu graphique (Qt). Les zones possibles sont cercle, pointeur. Sinon, le rendu est actuellement fait par pixellisation / voxelisation. Comme cette opération est couteuse, dans ce cas les zones n'évoluent pas en temps réel avec le changement des paramètres et s'affichent avec leur valeur par défaut.

Enfin, on peut définir des calculs de relations entre les différentes zones. Par exemple, on peut 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. Là encore, une opération entre deux zones génériques sera couteuse en temps de calcul. Cette information peut ensuite être exposée dans l'arbre i-score, puis être réutilisée par la suite pour concevoir d'autres zones.

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 tick d'horloge.

De la même manière, on peut utiliser les automations à une, deux, ou trois dimensions pour modifier les zones lors de l'exécution.

4. IMPLÉMENTATION

L'interprétation des équations se fait par le CAS GiNaC Problème avec GiNaC : pas de multi-thread. Autres bibliothèques envisagées : Symbolic C++, Sage, GNU Octave ? Par la suite, utiliser VTK ? -> vtkfunctionparser

4.1. Sémantique d'exécution

Sémantique d'exécution : résultats obtenus au tick d'après. Si plusieurs processus on utilise la sémantique d'i-score (les processus sont empilés). Discuter les multiples sémantiques possibles : mise à jour en temps réel, ou bien "State".

Dans un tick, on calcule d'abord les valeurs actuelles des zones.

Puis on exécute les computations qui écrivent dans l'arbre.

Nécessité d'un graphe d'exécution pour l'ordre ? -> modèle de flot en perspective pour la suite.

4.1.1. Processus spatial

Dans le premier cas : Le logiciel i-score expose son arbre et les processus spatiaux sont des parties intégrantes de l'arbre ; elles se mettent à jour de manière réactive lorsqu'un message est reçu et déclenchent une chaine d'évaluation qui va de calculs en calculs.

Dans le second cas, on suit la sémantique des processus i-score qui est par "State" agissant à l'exécution. Cela pose le problème de la visualisation : on ne peut pas voir lorsqu'on édite un scénario l'effet des changements de paramètres.

Cependant, si on a envie de voir le rendu en temps réel à l'écriture, il faut que l'outil lui-même s'expose de manière permanente via OSC et soit à l'écoute des données de l'arbre.

Ce problème se pose à moindre échelle pour les automations : par exemple, les automations ont une adresse de sortie qui leur est affectée. Cependant, on pourrait aussi vouloir avoir une approche ou l'on dispoe d'une automation, et ou plusieurs autres processus vont lire la valeur actuelle de l'automation.

4.1.2. Automations multi-dimensionnelles

Animation : la durée d'une boite correspond à la durée d'une spline d'animation. Les splines sont paramétrées entre 0 et 1.

Ainsi, puisque les boites d'i-score ont une durées déterminées, cette durée est mappée directement à la valeur de la spline.

Il reste à choisir pour l'automation si on veut qu'elle envoie : - un tuple à une adresse donnée. - chaque composante sur une adresse distincte.

4.2. Rendu

nécessité de trouver une bibliothèque pour l'affichage rapide

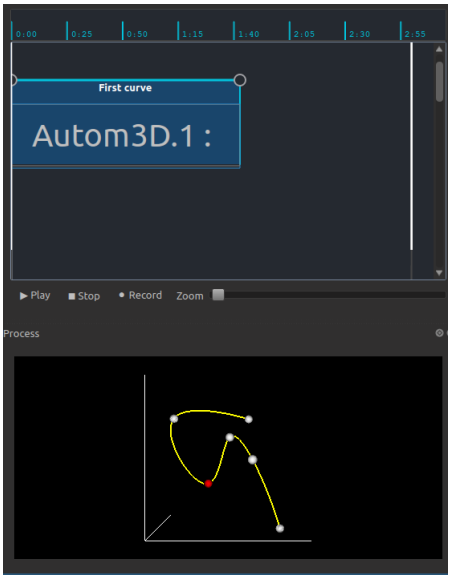

Figure 1. Une spline 3D dans i-score. On affiche la courbe dans le panneau inférieur

4.3. Édition

L'édition va modifier pour les formes connues des paramètres définis.

Les opérations de base sont déplacement, translation, rotation.

On applique les transformations à l'objet :

— La translation est triviale.

— Rotation par un angle thêta dans le plan :

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

Une formule générale de la rotation en n-dimension est donnée dans [1].

— Mise à l'échelle : pas de formule générale ; on fait un rendu et on met à l'échelle par la suite ?

5. CONCLUSION

6. REFERENCES

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

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

[3] M Olivier DELERUE. Spatialisation du son et programmation par contraintes : le système MusicSpace.

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

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

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

[7] 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.

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

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

[10] Bertrand Merlier. VOCABULAIRE DE l'ESPACE ET DE LA SPATIALISATION DES MUSIQUES ÉLECTROACOUSTIQUES : PRÉSENTATION, PROBLÉMATIQUE ET TAXINOMIE DE l'ESPACE. In *EMS : Electroacoustic Music Studies Network ?Beijing 2006*, page 247.

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

[12] 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.

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

[14] 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.