# 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

Les pratiques 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 mappe un paramètre à un autre.

La plupart des outils permettant un travail spatial sont soit spécifiques à la création de trajectoires pour des sons, dans un cadre musical, soient orientées vers la création de jeu vidéo ou de formes spécifiques de médias.

Nous cherchons ici des méthodes génériques, permettant de manipuler tout type de données spatiales pour ensuite pouvoir les appliquer aux cas particuliers de la création musicale notamment dans un cadre interactif.

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 dans un cadre musical de nouveaux types de compositions, notamment liées aux relations que l'on peut avoir entre plusieurs zones définies géométriquement.

Ce modèle, intégré au séquenceur i-score, sera appliqué à deux études de cas : une chorégraphie de robots, et une application de réalité augmentée audio.

Nous discuterons ensuite des évolutions possibles qui sont liées à un typage plus fort des données avec lesquelles on travaille : notamment les modes d'édition et de représentation qui peuvent être appliqués, ainsi que de la quesiton de la relation au temps d'objets spatiaux en deux dimensions ou plus.

### 1.1. Existant

Une présentation de l'état actuel de l'écriture spatiale en musique est donnée dans [8]. 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 [7]. Une taxinomie des possibilités de création dans l'espace en musiques électro-acoustiques est présentée par Bertrand Merlier dans [13]. Elle est étendue dans l'ouvrage *Vocabulaire de l'espace en musique électro-acoustique[12].*

Des outils logiciels existent pour ces partitions – ils sont souvent spécialisés. Par exemple, la bibliothèque **ENP!** (**ENP!**)[11] 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[18], 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[9] (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[5]. 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[14], qui permet de capter le mouvement de danseurs et d'en extraire des in-

formations 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[10] et différentes approches dérivées qui peuvent être spécifiquement axées sur la spatialisation du son[16].

Un modèle plus complet d'espace en trois dimensions est fourni par COSM[19]. 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

Cette application interactive utilise de la géolocalisation par téléphone. Plusieurs sources sonores sont apposées dans un espace, dans lequel les participants se déplacent. 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. La spatialisation est réalisée par la bibliothèque OpenAL

Il manque un outil simple d'écriture pour disposer les zones dans l'espace. De plus, on désire 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.

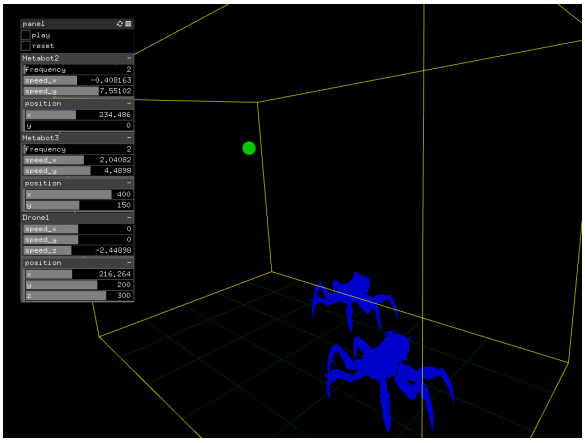


**Figure 1**. Logiciel de simulation de metabots communiquant avec i-score

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

Ce projet consiste en la réalisation d'une chorégraphie de robots musicaux. On dispose d'une flotte de robots open-source Metabot [1] Ces robots se contrôlent en vitesse : on écrit sur le port série des commandes telles que `dx 30` pour indique une vitesse de $30\,\mathrm{cm\,s^{-1}}$. Un logiciel a été conçu pour faire un mapping d'un arbre Minuit vers le port série. En parallèle, un logiciel de simulation 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.

Chaque robot va posséder une trajectoire dans le plan..

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 robot battent un rythme en tapant des pieds.

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

## 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. 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[15] qui produit un objet sur lequel on peut réaliser quelques calculs. 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 qui serait complètement opaque pour le t 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 [3]. 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 : il est possible d'exhiber des calculs spécialisés pour chaque relation entre deux types de zones. Enfin, pour deux zones génériques, non typées, nous pouvons chercher pour l'existance 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, qui permet de minimiser des systèmes non-linéaires. Il existe aussi des méthodes pour mesurer des distances entre deux zones quelconques. Par exemple [6]

Cette information peut ensuite être exposée dans l'arbre i-score, puis être réutilisée par la suite pour concevoir d'autres zones, ou bien être par la suite utilisée dans d'autres logiciels.

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

Une tentative d'implémentation a été réalisée à l'aide du CAS **GiNaC** [2]. 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 thread. L'implémentation actuelle est donc basée sur le parseur de fonctions mathématiques de la bibliothèque **vtk** [17]. D'autres alternatives ont été envisagées : Symbolic C++, Sage, GNU Octave.

### 4.1. Sémantique d'exécution

La sémantique générale d'exécution d'i-score est donnée dans [4]. 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 une cue.

#### *4.1.1. Processus spatial*

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

— 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.

Ainsi, s'il y a rétroaction, elle se fait au tick d'après.

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

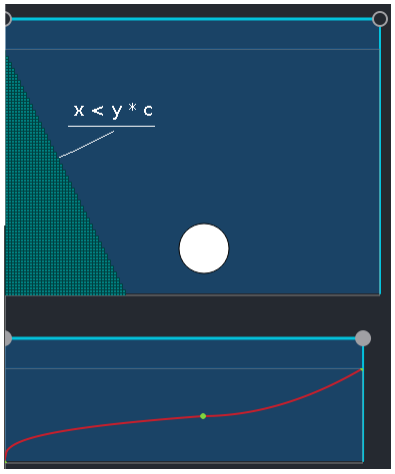


**Figure 2**. Processus spatial au début. La courbe d'automtion est mappée au paramètre $c$ de l'équation de la zone triangulaire.

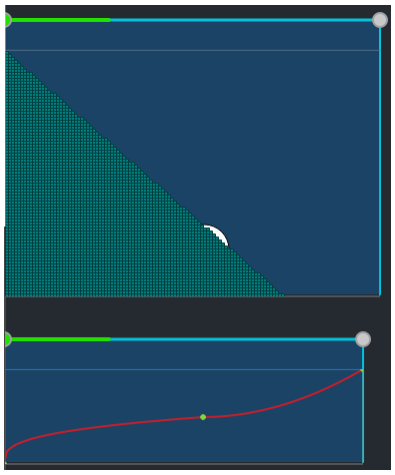

**Figure 3**. Processus spatial durant l'exécution. Il est possible de récupérer l'information de collision entre le disque et la zone

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 d'équation $(x - x_0)^2 + (y - y_0)^2 <= r_0$ les paramètres $x_0, y_0, r_0$ 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.

La première sémantique permet d'avoir une visualisation à l'exécution, tandis que la seconde permet d'avoir une visualisation à l'édition, en permettant le maintien des contraintes décrites par l'auteur.

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 Au-

tomation 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 mappings 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.

#### *4.1.2. Automations multi-dimensionnelles*

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

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

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

— Les automations à plus d'une dimension : nous utilisons ici la bibliothèque **vtk** pour afficher et éditer une spline à l'écran.

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

L'affichage se fait de manière spécialisée pour les types connus. Pour les types génériques, on réalise pour l'instant une pixelisation à partir , en testant pour chaque point de la zone d'affichage.

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

C'est notamment important si on désire avoir un rendu fluide à l'exécution. Le nombre d'images par seconde 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 à l'aide de bibliothèques spécialisées comme ROOT de CERN ou

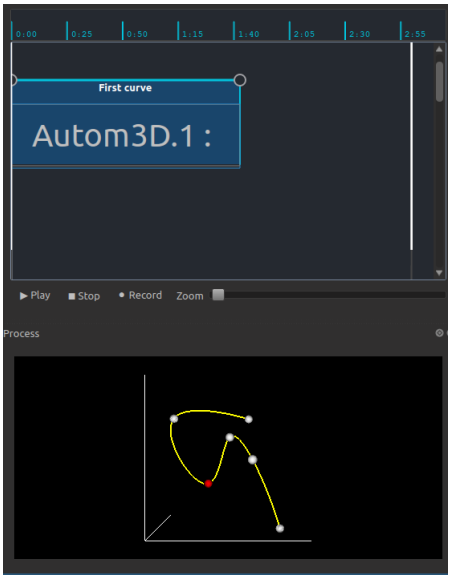


**Figure 4**. Une spline 3D dans i-score. On affiche la courbe dans le panneau inférieur

CGAL qui permet de faire de la triangulation de Delaunay d'une fonction donnée.

Mappings d'espaces de donnée ? Couleur, temps, etc.

### 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

Perspectives : gestion d'espaces de données, comme couleurs, etc. Gestion des types à plus d'une dimension. Il est nécessaire d'avoir une notion de paramètre de plus haut niveau. Pour ce faire, il serait possible d'étudier l'applicabilité de protocoles orientés objet, tels que D-Bus ou ZeroC ICE, pour exposer les objets i-score.

## 6. REMERCIEMENTS

ANR groupe metabots

## 7. REFERENCES


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

[2] 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.

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

[4] Jean-Michaël Celerier, Blue Yeti, Pascal Baltazar, Clément Bossut, Nicolas Vuaille, Jean-Michel Couturier, and Myriam Desainte-Catherine. OSSIA : TOWARDS a UNIFIED INTERFACE FOR SCORING TIME AND INTERACTION.

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

[6] Gershon Elber and Tom Grandine. Hausdorff and minimal distances between parametric freeforms in\ mathbb r^ 2 and\ mathbb r^ 3. In *Advances in Geometric Modeling and Processing*, pages 191–204. Springer, 2008.

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

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

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

[10] 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.

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

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

[13] 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.

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

[15] 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.

[16] 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.

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

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

[19] 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.