CHAMPS VECTORIELS POUR LE CONTRÔLE DE LA SPATIALISATION

Thibaut Carpentier
CNRS, IRCAM, Sorbonne Université – UMR STMS
1, place Igor Stravinsky, 75004 Paris
thibaut.carpentier@ircam.fr

Andrew Gerzso
IRCAM
1, place Igor Stravinsky, 75004 Paris
andrew.gerzso@ircam.fr

RÉSUMÉ

Cet article décrit un outil logiciel qui exploite, de façon quelque peu métaphorique, le concept de *champ vectoriel* pour générer des mouvements fluides et naturels d'agents autonomes. L'« aptitude à s'orienter librement » de ces agents peut se révéler utile dans des scénarios compositionnels dans lesquels on cherche à explorer un espace de paramètres d'une façon mi-improvisée et mi-déterministe (inspirée par les lois de la physique). L'article se concentre principalement sur le contrôle de la spatialisation sonore, où le logiciel est utilisé pour créer des trajectoires qui évoluent perpétuellement ; toutefois d'autres applications potentielles sont également suggérées.

1. INTRODUCTION

1.1. Les trajectoires spatiales en tant que paradigme compositionel

La spatialité de la musique et la spatialisation des sons constituent un enjeu majeur des compositions électroacoustiques depuis les travaux pionniers de la *musique concrète* dans les années 1950. Depuis lors, les artistes déploient une large palette de techniques d'écriture de l'espace, et parallèlement, de nombreuses technologies de spatialisation ont été développées, permettant aux compositeurs et aux interprètes de contrôler divers paramètres spatio-musicaux. Parmi les techniques employées, les mouvements de sources sonores constituent une des pratiques compositionnelles les plus courantes, attesté par plusieurs sondages [2, 19, 16, 9]. Les déplacements de sons dans l'espace sont typiquement utilisés pour produire une chorégraphie musicale, pour souligner les contrastes entre des éléments statiques et dynamiques, ou pour faciliter la discrimination de flux concurrents. Dans la plupart des compositions, ces mouvements de sources sonores ponctuelles – dans des espaces réels ou virtuels – sont pensés comme des trajectoires spatiales, décrites sous forme de courbes, de trajets, de motifs géométriques, etc. La notion de trajectoire constitue ainsi un paradigme essentiel pour l'écriture et l'organisation spatio-temporelles de la musique.

1.2. Techniques d'écriture de l'espace

Un grand nombre d'outils ont d'ores et déjà été proposés pour créer et manipuler des données spatiales, en vue de produire des trajectoires sonores. Garcia et al. (voir le paragraphe 2.3 dans [9]) ont fait l'inventaire des principaux environnements d'écriture de l'espace, et ils soulignent que, dans la plupart des cas, les outils proposent une séparation claire entre l'interface d'écriture (dans laquelle les compositeurs peuvent générer et éditer des trajectoires) et le moteur de rendu (en charge de la production et du traitement des signaux audio). Très grossièrement, on peut cataloguer les outils d'édition de trajectoires selon trois grandes catégories :

— des éditeurs de trajectoires « de bas-niveau » qui traitent les données spatiales comme des pistes d'automation (à l'instar des stations de travail audionumériques). Typiquement, ces outils permettent une manipulation « précise » des trajectoires, à l'aide de *breakpoint functions*, de courbes splines ou de Bézier, de dessins à main levée, ou tout autre type de représentation temps+coordonnées. Des options de transformations (rotation, translation, mise à l'échelle) sont généralement proposées. On peut notamment inclure dans cette catégorie les logiciels Zirkonium [27], Holo-Edit [20], Ambicontrol [24], SpaceJam [14], Trajectoires [10], etc., ainsi que des plugins (pour stations de travail) tels que Spatium [18] ou ToscA [3].

— des techniques algorithmiques : cette catégorie, relativement vaste, englobe les trajectoires prédéfinies ou paramétriques (ellipses, spirales, etc.) [15], les outils génératifs (courbes de Lissajou, mouvements browniens, processus stochastiques, etc.), la simulation de mouvements de nuées ou d'essaims [13, 7], les modèles physiques [18, 1], la diffraction de particules sonores [23], les moteurs de contraintes [17], les bibliothèques de composition algorithmique ou de *scripting* [25], etc.

— des éditeurs symboliques de « plus haut-niveau » qui proposent une description des caractéristiques spatiales sous la forme de symboles, de métaphores, voire de partitions, en vue de produire des motifs, des figures et des scénarios autonomes de spatialisation. Cette catégorie inclut notamment le Spatialization Symbolic Music Notation framework [8], IanniX [12], i-score [6], ou encore d'autres propositions de notation symbolique de l'écriture de l'espace [11].

Il faut toutefois signaler que la plupart des environnements logiciels sus-mentionnés ne sont pas strictement cantonnés

à une catégorie ; par exemple Holo-Edit est principalement un éditeur de bas-niveau, mais il intègre aussi des fonctionnalités relevant de l'approche algorithmique. De la même manière, les compositeurs ont coutume de jongler entre différentes perspectives, notamment pour s'affranchir de la complexité et des contraintes expressives des outils proposés, et ce, dans le but de bâtir leur propre schéma de représentation des mouvements spatiaux.

2. CHAMPS VECTORIELS ET NAVIGATION AUTONOME

Dans cet article nous présentons un nouvel outil de manipulation de trajectoires, qui partage des similarités avec les logiciels précédemment évoqués, et qui s'inscrit à la fois dans les catégories algorithmique et symbolique. Son fondement repose sur la notion de *champ vectoriel*, une représentation inspirée de la physique mais abordée sous un angle quelque peu métaphorique.

Un champ vectoriel (ou champ de vecteurs) consiste en une collection de points dans l'espace, désignée comme le domaine, et une fonction qui associe à chaque point du domaine un (et un seul) vecteur. Les champs vectoriels sont typiquement représentés dans le plan 2D ou dans l'espace 3D comme un ensemble de flèches, avec une longueur et une orientation donnée, pour chaque point du domaine. Les champs vectoriels sont souvent utilisés en physique et dans les sciences de l'ingénieur, comme outil de modélisation et de visualisation de l'amplitude et la direction de vecteurs (forces, vélocités, etc.) qui évoluent points par points. Ils sont par exemple employés pour représenter des champs magnétiques ou gravitationnels, des fluides en mouvement, ou encore des cartographies de vitesse du vent dans les relevés météorologiques. Les champs de vecteurs proposent une représentation très intuitive d'un flux d'éléments dans un espace. Comme ils peuvent être associés à une équation différentielle, les champs vectoriels présentent souvent des motifs particuliers au voisinage des points d'équilibre, par exemple une « source » (vecteurs émanant d'un nœud spécifique), un « puits » (vecteurs qui s'évanouissent dans un trou), ou un « point selle » (en forme de selle de cheval).

`simone` [1] est un outil logiciel (décrit plus en détails au paragraphe 3) qui s'appuie sur ces paradigmes. Son principe est extrêmement simple : l'outil permet de générer et visualiser un champ vectoriel 2D, considéré comme un champ de vélocité ; lorsque des objets (ou « particules ») sont « lancés » dans le champ, ils se retrouvent animés par le flux vectoriel et se mettent en mouvement dans le domaine, y naviguant comme des agents autonomes. Ceci offre donc un moyen très rapide et intuitif pour définir et explorer un espace, par exemple un espace de paramètres musicaux ; l'exploration est en partie prévisible (le champ vectoriel étant facilement appréhendé à l'œil nu) et en partie « improvisée » (les singularités du champ de vecteurs pouvant faire dévier les particules de leur trajectoire initiale). Pour autant, `simone` n'est pas conçu comme un moteur de simulation physique stricto sensu ; le logiciel est avant tout destiné à des compositeurs, et pour des usages créatifs. C'est pourquoi nous y avons également introduit un ensemble d'opérateurs métaphoriques (voir 3.1), qui peuvent perturber le comportement des particules dans le champ vectoriel, permettant potentiellement de produire des motifs plus variés ou erratiques.

L'approche ainsi adoptée dans `simone` n'est pas sans rappeler la notion de *steering behaviors* — que l'on peut maladroitement traduire par « aptitudes à s'orienter » ou « comportements de navigation ». « Steering behaviors » est un terme essentiellement en usage dans le domaine du jeu vidéo, et tiré des travaux fondateurs de Reynolds [22] : l'expression fait référence à l'aptitude de personnages à se déplacer dans leur environnement de façon autonome et réaliste. Ces capacités de locomotion sont pilotées par un ensemble de forces et d'interactions qui, combinées entre elles, engendrent des déplacements naturels et fluides des personnages. Cela ne repose pas sur une formulation globale des actions des personnages (comme leur destination ou leur objectif à long terme), mais utilise au contraire des informations locales pour mettre à jour les trajectoires des agents de façon incrémentale, étape par étape. Ceci permet de produire des mouvements relativement complexes, mais avec une implémentation très simple. Reynolds formalise différents types de comportements, et donne des exemples tels que : la poursuite (personnage suivant une cible mouvante), la fuite (s'éloigner d'une position donnée), la déambulation (navigation aléatoire), l'évitement de collisions (contournement d'obstacles), le suivi de tracé (parcours le long d'un chemin), le suivi de flux (alignement d'un mouvement sur la tangente locale, comportement qui correspond le mieux aux champs vectoriels), les mouvements de groupes tels que les nuées (implémenté par exemple dans le fameux algorithme *boids* [21]), le suivi d'un meneur, etc.

3. OUTIL PROPOSÉ

L'interface graphique de `simone` est représentée sur la Figure 1. La fenêtre affiche un champ vectoriel 2D, constitué de N lignes et M colonnes, formant un domaine de $N \times M$ cellules (dans notre exemple $N = M = 17$). Chaque cellule contient un vecteur (c'est-à-dire une flèche) caractérisé par sa direction et sa vélocité. Contrairement aux représentations traditionnelles des champs de vecteurs, le paramètre de vélocité n'est pas visualisé par la longueur de la flèche ; ici, toutes les flèches sont visualisées avec la même longueur (arbitraire), tandis que le facteur de vélocité est symbolisé par un cercle rouge clair (le rayon du cercle étant proportionnel à la valeur de vélocité). Ce parti pris a été adopté pour faciliter la lisibilité et surtout l'édition des cellules. La couleur d'arrière-plan des cellules – ici, un damier de gris – n'a aucune signification ; ce choix est uniquement motivé par un souci de lisibilité et discrimination visuelle des cellules.

1 . Une première mouture, alors intitulée `xvf`, a été présentée oralement par Andrew Gerzso lors des ateliers du Forum Ircam en novembre 2010 ("`xvf` : A Max/MSP Graphics Object for Drawing and Generating 2D Trajectories"), toutefois cette version du logiciel n'avait pas fait l'objet d'une diffusion à la communauté.

Les caractéristiques de chaque cellule (orientation et vélocité du vecteur) peuvent être spécifiées via des messages, ou contrôlées à l'aide de la souris et de raccourcis clavier. Dans le présent exemple, le champ de vecteurs a une forme de spirale-vortex (avec des vitesses tirées aléatoirement) ; ce motif a été généré par un jeu d'équations paramétriques, très simples, codées en javascript. L'allure du vortex peut être modifiée par les paramètres α et β (voir boutons rotatifs dans la Figure 1) qui régissent les composantes de gradient selon les axes x et y respectivement. D'autres équations peuvent être facilement implémentées.

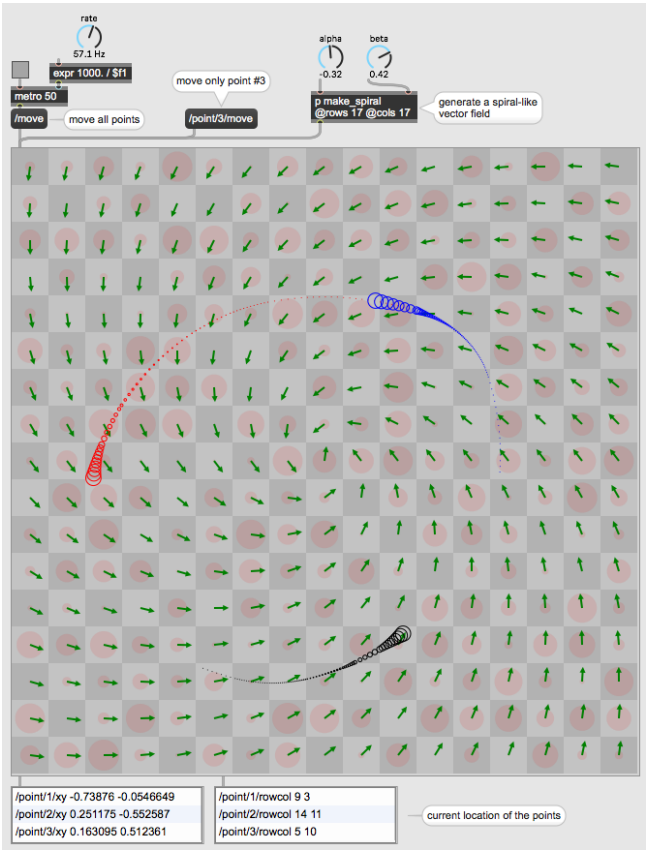

Figure 1. Interface graphique de `simone`. Partie supérieure : messages de contrôle (paramétrisation du champ vectoriel et mise en mouvement des particules). Partie inférieure : messages en sortie (positions instantanées des particules). Partie centrale : représentation du champ vectoriel et des particules y évoluant.

Étant donné ce champ vectoriel, on peut « lancer » des particules (simplement désignées comme des « points ») dans le champ, en spécifiant leur position initiale. L'exemple de la Figure 1 présente trois points, respectivement rouge, noir et bleu. À chaque pas de calcul, les trajectoires des points sont mises à jour en fonction de la cellule sur laquelle ils se trouvent : la vitesse et la direction des trajectoires sont pondérées par le vecteur de la cellule courante.

Une itération de calcul est déclenchée par l'envoi d'un message `/move`, typiquement asservi à un métronome. Changer la fréquence du métronome permet de varier la vitesse des trajectoires. Le message `/move` peut être utilisé globalement – s'appliquant à toutes les particules – ou de façon indépendante pour chaque point, de sorte que chaque trajectoire peut évoluer à sa propre cadence.

À chaque pas de calcul, `simone` délivre un ensemble de messages contenant les coordonnées et les indices des cellules courantes pour chacun des points (voir en bas de la Figure 1). Les coordonnées sont exprimées dans un repère Cartésien, sur un domaine arbitraire de $[-1; 1]$; bien entendu il convient ensuite de mettre ces données à l'échelle pour opérer dans une plage de valeurs plus pratique ou plus pertinente, selon l'application visée.

L'« historique » de chacun des points est visualisé par une queue en pointillés (de diamètre de plus en plus petit avec leur ancienneté). L'historique sert de repère visuel, mais peut également influer sur le calcul des trajectoires : l'utilisateur peut assigner un facteur d'« inertie » (exprimée entre 0 et 100%) pris en compte dans les calculs d'itération. Augmenter l'inertie rend les trajectoires plus fluides, tandis que les particules ayant une faible inertie peuvent changer subitement de direction si le champ vectoriel présente des singularités.

L'approche de champ de vecteurs ainsi proposée permet donc de générer des motifs spatiaux d'une manière puissante et intuitive. En outre, les trajectoires peuvent être modulées, en modifiant leur vitesse, ou bien en transformant le champ vectoriel sous-jacent. En effet, les caractéristiques du champ vectoriel peuvent être mises à jour en temps réel et ce, de plusieurs façons :

- les propriétés des équations paramétriques (les variables α et β dans notre exemple) peuvent être modifiées,
- des données extérieures (e.g. des bulletins météorologiques temps-réel) peuvent être utilisées pour générer un nouveau champ vectoriel,
- il est possible de transformer le curseur de la souris en « aimant » (attractif ou repoussant) afin d'altérer les directions du champ de vecteurs ; cette opération de magnétisme peut s'opérer sur l'ensemble du domaine, ou sur une ou des sous-régions définies par l'utilisateur,
- toute autre stratégie algorithmique peut être aisément implémentée. Par exemple, un point peut perturber la vélocité et/ou la direction des cellules qu'il a déjà visitées, faisant ainsi du champ vectoriel un environnement en perpétuelle évolution, etc.

Comme nous l'avons déjà signalé, `simone` n'est pas conçu comme un simulateur rigoureux de modèle physique ; il se veut plutôt un environnement ouvert et flexible permettant d'engendrer rapidement une vaste palette de motifs et de comportements. Afin d'accroître encore ses possibilités, nous avons introduit divers opérateurs symboliques qui s'éloignent du paradigme conventionnel de champ vectoriel.

3.1. Opérateurs de cellule

Il est possible d'assigner un opérateur, utilisé pour l'itération de calcul, à chaque cellule du domaine. Dans l'exemple de la Figure 1, toutes les cellules été configurées avec le même opérateur, un vecteur représentant une direction et une vélocité. D'autres types d'opérateurs sont disponibles, et représentés par les symboles suivants (voir Figure 2).

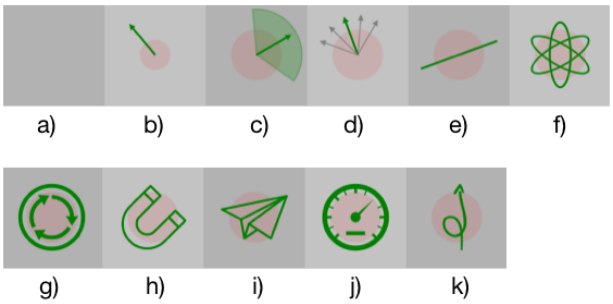

Figure 2. Différents opérateurs de cellule.

a) cellule vide (« void ») : une cellule vide (sans opérateur) n'a aucun effet ; elle ne modifie ni la direction nominale ni la vitesse des points qui la parcourent.
b) un simple vecteur, ainsi que précédemment décrit, impose une nouvelle direction et une nouvelle vélocité, en tenant compte du paramètre optionnel d'inertie.
c) « ranged vector » : similaire au vecteur simple, sinon que pour chaque pas de calcul, une nouvelle direction est tirée aléatoirement dans la plage symbolisée par l'arc de cercle vert.
d) « radial vector » : similaire au « ranged vector », mais la nouvelle direction est choisie aléatoirement parmi un ensemble discret de possibilités.
e) réflecteur (« paddle ») : provoque une réflexion, en fonction de l'angle d'incidence de la trajectoire.
f) opérateur aléatoire : une direction quelconque est choisie ; ceci est donc équivalent à un « ranged vector » ayant une variance de 360°.
g) opérateur répulsif : engendre une réflexion orthogonale à l'incidence de la trajectoire.
h) attracteur : produit un mouvement orbital autour de la cellule.
i) téléportation : transporte instantanément vers une autre cellule ou vers une position Cartésienne spécifiée par l'utilisateur.
j) changement de vitesse : modifie la vélocité de la trajectoire sans influer sur sa direction.
k) déviation : dévie la direction incidente d'une certaine valeur (offset).

En combinant ces opérateurs sur l'ensemble du domaine, il est possible de produire une gamme très variée de mouvements, et de simuler la plupart des « steering behaviors » proposés par Reynold. Par exemple, un comportement déambulatoire peut s'implémenter au moyen d'opérateurs « ranged » et « radial ».

Enfin, il est nécessaire de traiter les points qui sortent du domaine. Différentes options sont proposées : il est possible de laisser la trajectoire disparaître (elle devient alors inactive), de « replier » le point vers l'origine, ou vers une cellule ou une coordonnée Cartésienne donnée, ou encore vers une position aléatoire.

3.2. Aspects logiciels

`simone` est diffusé sous forme d'objet externe Max, développé en `C++` à l'aide du framework Juce. Il est distribué gratuitement via le package Ircam `spat~` [5]. L'objet bénéfice des dernières améliorations structurelles de l'architecture logicielle de `spat~` [4] : import/export de presets, compatibilité avec les « snapshots » de Max, et avec le protocole OSC [28], etc. Dans le contexte des champs de vecteurs, les fonctionnalités de « pattern matching » OSC se révèlent particulièrement utiles et puissantes, notamment pour configurer une sous-région du domaine, via une syntaxe concise. Par exemple, le message `/row/[2-5]/col/*/type repeller` permet d'assigner le type « opérateur répulsif » à toutes les cellules comprises entre les lignes 2 et 5.

4. CONCLUSION ET PERSPECTIVES

Dans cet article, nous avons présenté un nouvel outil logiciel qui permet de générer des trajectoires spatiales en se basant sur le concept de champs vectoriels. Les trajectoires explorent un domaine tels des agents autonomes, évoluant d'une façon naturelle et semi-improvisée. L'ajout de différents opérateurs métaphoriques permet en outre d'enrichir le registre des mouvements et comportements simulés. Le champ vectoriel sous-jacent peut être généré via diverses techniques, paramétrisé par des équations mathématiques, dessiné à la main, ou issu d'un *mapping* de données. L'outil est d'un usage intuitif, et il est suffisamment générique pour être déployé dans de nombreux contextes. Le contrôle et l'écriture de la spatialisation sont le premier champ d'application visée, puisque le paradigme de trajectoires sonores est l'une des pratiques compositionnelles les plus communes. Toutefois d'autres cadres applicatifs sont envisagés ; en particulier, `simone` peut se révéler utile pour le contrôle de la synthèse concaténative par corpus [26], dans laquelle des fragments d'échantillons audio sont recomposés par navigation dans un espace multi-dimensionnel de descripteurs musicaux. Les travaux à venir étudieront l'intérêt potentiel de `simone` pour d'autres processus de composition générative. Enfin, de futurs développements sont également envisagés pour accroître encore l'expressivité de l'outil, par exemple en ajoutant de nouveaux opérateurs symboliques, ou en permettant des mécanismes de contraintes entre trajectoires, de sorte à produire des « steering behaviors » collectifs.

5. REFERENCES

[1] Baalman, M.A., "Application of Wave Field Synthesis in electronic music and sound installations", *Proc. of the Linux Audio Conference (LAC)*, Karlsruhe, 2004.

[2] Baalman, M.A., "Spatial Composition Techniques and Sound Spatialisation Technologies", *Organised Sound*, volume 15(3) :pp. 209 – 218, Dec 2010.

[3] Carpentier, T., "ToscA : An OSC Communication Plugin for Object-Oriented Spatialization Authoring", *Proc. of the 41st International Computer Music Conference (ICMC)*, pp. 368 – 371, Denton, TX, USA, Sept. 2015.

[4] Carpentier, T., "A new implementation of Spat in Max", *Proc. of the 15th Sound and Music Computing Conference*, pp. 184 – 191, Limassol, Cyprus, Juy 2018.

[5] Carpentier, T., Noisternig, M., Warusfel, O., "Twenty Years of Ircam Spat : Looking Back, Looking Forward", *Proc. of the 41st International Computer Music Conference*, pp. 270 – 277, Denton, TX, USA, Sept. 2015.

[6] Celerier, J.M., Desainte-Catherine, M., Couturier, J.M., "Outils d'écriture spatiale pour les partitions interactives", *Proc. of Journées d'Informatique Musicale (JIM)*, pp. 82 – 92, Albi, France, Mar 2016.

[7] Davis, T., Rebelo, P., "Hearing emergence : towards sound-based self-organisation", *Proc. of the 31st International Computer Music Conference (ICMC)*, pp. 463 – 466, Barcelona, Spain, 2005.

[8] Ellberger, E., Perez, G.T., Schuett, J., Zoia, G., Cavaliero, L., "Spatialization Symbolic Music Notation at ICST", *Proc. of the 40th International Computer Music Conference*, Athens, Greece, Sept. 2014.

[9] Garcia, J., Carpentier, T., Bresson, J., "Interactive-compositional authoring of sound spatialization", *Journal of New Music Research*, volume 46(1) :pp. 74 – 86, 2017.

[10] Garcia, J., Favory, X., Bresson, J., "Trajectoires : a Mobile Application for Controlling Sound Spatialization", *Proc. of CHI EA'16 : ACM Extended Abstracts on Human Factors in Computing Systems*, pp. 3671 – 3674, San Jose, CA, USA, May 2016.

[11] Gottfried, R., "SVG to OSC Transcoding as a Platform for Notational Praxis and Electronic Performance", *Proc. of the 1st International Conference on Technologies for Music Notation and Representation (TENOR)*, pp. 154 – 161, Paris, France, May 2015.

[12] Jacquemin, G., Coduys, T., Ranc, M., "Iannix 0.8", *Proc. of Journées d'Informatique Musicale (JIM)*, pp. 107 – 115, Mons, Belgium, May 2012.

[13] Kim-Boyle, D., "Spectral and Granular Spatialization with Boids", *Proc. of the 32nd International Computer Music Conference (ICMC)*, pp. 139 – 142, New Orleans, LA, USA, 2006.

[14] Madden, A., *Developing spaceJam : The New Sound Spatialization Tool for an Artist and Novice*, Master's thesis, New York University, Nov 2014.

[15] Normandeau, R., "Octogris2 et ZirkOSC2 : outils logiciels pour une spatialisation sonore intégrée au travail de composition", *Proc. of Journées d'Informatique Musicale (JIM)*, Montreal, Canada, May 2015.

[16] Otondo, F., "Contemporary trends in the use of space in electroacoustic music", *Organised Sound*, volume 13(1) :pp. 77 – 81, April 2008.

[17] Pachet, F., Delerue, O., "Constraint-Based Spatialization", *Proc. of the International Conference on Digital Audio Effects (DAFx)*, Barcelona, Spain, 1998.

[18] Penha, R., Oliveira, J.P., "Spatium, tools for sound spatialization", *Proc. of the Sound and Music Computing Conference*, pp. 660 – 667, Stockholm, 2013.

[19] Peters, N., Marentakis, G., McAdams, S., "Current Technologies and Compositional Practices for Spatialization : A Qualitative and Quantitative Analysis", *Computer Music Journal*, volume 35(1) :pp. 10 – 27, 2011.

[20] Pottier, L., "Dynamical spatialization of sound. HoloPhon : a graphic and algorithmic editor for Sigma1", *Proc. of the International Conference on Digital Audio Effects (DAFx)*, Barcelona, Spain, 1998.

[21] Reynolds, C.W., "Flocks, Herds, and Schools : A Distributed Behavioral Model", *Computer Graphics*, volume 21(4) :pp. 25 – 34, 1987.

[22] Reynolds, C.W., "Steering Behaviors For Autonomous Characters", *Proc. of the Game Developers Conference*, pp. 763 – 782, San Jose, CA, USA, 1999.

[23] Roads, C., *Microsound*, The MIT Press, 2001.

[24] Schacher, J.C., "Seven years of ICST ambisonics tools for MaxMSP – a brief report", *Proc. of the 2nd International Symposium on Ambisonics and Spherical Acoustics*, Paris, France, May 2010.

[25] Schumacher, M., Bresson, J., "Spatial Sound Synthesis in Computer-Aided Composition", *Organised Sound*, volume 15(3) :pp. 271 – 289, Dec 2010.

[26] Schwarz, D., "Corpus-Based Concatenative Synthesis", *IEEE Signal Processing Magazine*, volume 24(2) :pp. 92–104, March 2007.

[27] Wagner, D., Brümmer, L., Dipper, G., Otto, J.A., "Introducing the Zirkonium MK2 System for Spatial Composition", *Proc. of the International Computer Music Conference (ICMC) / Sound and Music Computing*, pp. 823 – 829, Athens, Greece, Sept. 2014.

[28] Wright, M., "Open Sound Control : an enabling technology for musical networking", *Organised Sound*, volume 10(3) :pp. 193 – 200, Dec 2005.