TOSCA : UN PLUGIN DE COMMUNICATION OSC POUR LE MIXAGE SPATIALISÉ ORIENTÉ OBJET

Thibaut Carpentier
UMR 9912 STMS IRCAM-CNRS-UPMC
1, place Igor Stravinsky, 75004 Paris
thibaut.carpentier@ircam.fr

RÉSUMÉ

Cet article présente *ToscA*, un *plugin* permettant de communiquer, en lecture comme en écriture, l'automation des paramètres d'une station de travail audionumérique vers d'autres applications, via le protocole OSC. Son application typique concerne la réalisation de mixages spatialisés orientés objet, indépendamment des contraintes des logiciels hôtes.

1. INTRODUCTION

La spatialisation sonore connaît actuellement un certain essor. Certes les contenus discographiques restent majoritairement produits en stéréophonie (ou très marginalement en 5.1), mais la spatialisation joue un rôle croissant dans d'autres secteurs de la production audio : l'industrie cinématographique, notamment, s'oriente vers la diffusion « 3D », et les salles de projection s'équipent progressivement de dispositifs de restitution tridimensionnelle constitués de plusieurs dizaines de haut-parleurs. Parallèlement, les comportements de consommation des contenus multimédia évoluent : avec l'avènement des *smartphones* et tablettes, les programmes audio/vidéo s'écoutent désormais de façon individuelle, sur casque. Ceci incite les producteurs de contenus, en particulier radiophoniques [1], à s'intéresser aux techniques de rendu binaural, qui sont en voie de démocratisation. Enfin, la disponibilité d'équipements audionumériques « massivement multicanaux » (interfaces et protocoles de transmission tels que MADI [2] ou Dante [3]) facilite l'aménagement d'ambitieux dispositifs de spatialisation dans des salles de concert [7] ou dans le cadre d'installations sonores.

En ce qui concerne la production et la transmission de contenus spatialisés, trois paradigmes peuvent être distingués : les techniques orientées canal (« channel-based »), orientées scène (« scene-based ») et orientées objets (« object-based »). L'approche « channel-based » consiste à produire le contenu pour un dispositif de restitution donné et standardisé (par exemple stéréophonique ou 5.1 ITU-R BS 775) ; le contenu est alors stocké dans un format contenant autant de pistes que de points de diffusion, et chaque piste est directement envoyée vers le canal approprié. L'approche « scene-based » appréhende une scène sonore spatiale dans sa globalité, et réalise son encodage dans un formalisme indépendant du dispositif de restitution, typiquement au moyen d'une représentation Ambisonique [3]. Lors de la reproduction, un décodage est nécessaire, et il est par ailleurs possible d'appliquer certaines transformations globales de la scène d'origine (par exemple une rotation d'ensemble). Enfin, les mixages orientés objets considèrent des entités sonores (généralement mono ou stéréophoniques) associées à des méta-données de contrôle (position, orientation, gain, etc.). L'ensemble des données (audio et contrôle afférent) est convoyé jusqu'au lieu de diffusion où un moteur de rendu est en charge de spatialiser les éléments selon la configuration d'écoute (casque ou ensemble de haut-parleurs). Il est ainsi possible pour l'utilisateur final d'interagir voire de re-mixer les objets constitutifs de la scène sonore lors du rendu ; dans le cadre d'une diffusion binaurale, il est par exemple possible, au niveau du terminal de reproduction, de *rendre* le mix avec des HRTFs [4] personnalisées ou adaptées à l'auditeur.

L'émergence (et la multiplication) des nouvelles plateformes de diffusion précédemment mentionnées — cinéma 3D, diffusion binaurale, etc. — semble donc favorable au développement des approches de mixage orientées objets [5], en raison de leur grande flexibilité.

Les processeurs de spatialisation orientés objets sont nombreux (citons par exemple : *Sound Element Spatializer* [6], *SoundScape Renderer* [4], *ICST Ambisonics Tools* [13], *Spatium* [8], VBAP [11], *HoaLibrary* [14], *Zirkonium* [12], *Spatialisateur* [5], etc.), et, d'un point de vue du traitement du signal audio ils sont performants et offrent de grandes possibilités. En revanche, l'édition (*authoring*) de scènes sonores

1. Citons notamment les réalisations de Radio France et de la BBC, respectivement : `http://nouvoson.radiofrance.fr` et `http://www.bbc.co.uk/rd/projects/binaural-broadcasting`.
2. Multichannel Audio Digital Interface
3. `http://www.audinate.com`
4. Head-Related Transfer Functions.
5. Des approches hybrides telles que « object-based + scene-based » ou « object-based + channel-based » sont aussi fréquemment employées (par exemple dans les dispositifs *Atmos*).

spatialisées demeure un défi majeur pour les producteurs de contenus, et ce, pour plusieurs raisons : 1) la plupart des processeurs précédemment mentionnés s'intègrent dans des environnements temps-réel – tels que Max/MSP ou Pure Data – qui sont mal adaptés au mixage ou à l'écriture de la spatialisation car ils ne se prêtent guère à la manipulation de séquences dans le temps. 2) À l'inverse, les stations de travail audionumériques (Digital Audio Workstations ou DAW) possèdent une puissante *timeline* et barre de transport, mais souffrent d'un manque de flexibilité dans la gestion des pistes multicanales : la plupart des DAWs ne supportent que des pistes et des bus « faiblement » multicanaux (stéréo, 5.1, 7.1) et l'insertion de plugiciels (*plugins*) de spatialisation y est donc très contrainte et malaisée. 3) Enfin, l'absence de standardisation ou de consensus pour la représentation des méta-données de contrôle de spatialisation (en dépit de plusieurs initiatives, par exemple [1, 10]) freine la communication entre les différents outils.

Cet article présente *ToscA*, un plugiciel qui permet de communiquer l'automation d'une station de travail vers d'autres applications. Le principal cas d'usage concerne la réalisation de mixages spatialisés, orientés objet, et « massivement » multicanaux.

2. PRÉSENTATION

Nous proposons ici une architecture de travail dans laquelle le moteur de rendu spatialisé est déporté en dehors de la station de travail (Figure 1). L'idée est d'extraire d'une part les pistes audio à traiter, d'autre part les données d'automation de spatialisation, et les transmettre du DAW vers l'outil de rendu. Celui-ci traite les données et délivre les signaux spatialisés. Ces signaux pourront être directement envoyés au dispositif de restitution, rendus dans des fichiers audio, ou encore renvoyés vers le DAW.

Dans la suite de cet article, on désignera par « application auxiliaire » l'environnement dans lequel est inséré le moteur de rendu de spatialisation.

Typiquement les pistes audio à spatialiser sont monophoniques ou stéréophoniques, et les données d'automation portent sur la localisation spatiale (coordonnées, éventuellement orientation et directivité). Les informations de localisation sont le plus souvent stockées en coordonnées cartésiennes ou polaires, en 3D, et elles sont donc stockées sur trois pistes d'automation.

2.1. Transmission audio

L'architecture de rendu « déporté » hors du DAW requiert de transmettre les pistes audio du DAW vers l'application auxiliaire. Plusieurs solutions permettent d'accomplir cette communication inter-applications, et deux cas d'usage doivent être distingués :

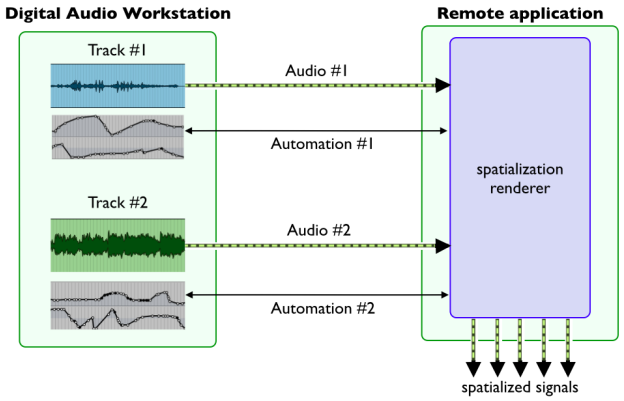

Figure 1. Principe de rendu de spatialisation orientée objet dans une application déportée hors de la station de travail.

— les deux applications (DAW et application auxiliaire) sont installées sur deux ordinateurs distincts ; dans ce cas on se contentera de connecter physiquement les interfaces audio des deux machines entre elles,

— les deux applications tournent sur un même ordinateur. Si la connectique le permet, on pourra réaliser un bouclage physique des sorties DAW de l'interface audio vers les entrées de l'application auxiliaire. Un tel routage peut également se faire de façon logicielle, par exemple avec Jack [6], SoundFlower [7], ou autres. Certaines interfaces audio autorisent également un tel bouclage « software », sans latence, au niveau du pilote (*driver*) audio.

2.2. Transmission des automations

Pour la communication des données d'automation, une solution tout d'abord envisagée – et mise en œuvre dans plusieurs cas concrets de productions à l'Ircam et ailleurs – consiste à utiliser le protocole MIDI. En effet, celui-ci est parfaitement supporté dans toutes les stations de travail et dans la plupart des applications auxiliaires. L'approche consiste en général à affecter un canal MIDI à chaque piste à spatialiser, et un numéro de contrôleur MIDI à chaque paramètre de spatialisation. Ainsi, la syntaxe des messages transmis est telle que :

numéro du canal MIDI	$\longleftrightarrow$	numéro de l'objet à spatialiser
numéro du contrôleur MIDI	$\longleftrightarrow$	numéro du paramètre de spatialisation à piloter
valeur du contrôleur MIDI	$\longleftrightarrow$	valeur du paramètre

Au niveau de l'application auxiliaire, une mise en correspondance (*mapping*) des contrôleurs doit être réalisée, ainsi

6. http://jackaudio.org
7. http://rogueamoeba.com/freebies/soundflower

qu'une mise à l'échelle des valeurs des contrôleurs (les données MIDI émises par le DAW étant codées sur la plage [0 – 127]).

Cette approche de communication MIDI, certes fonctionnelle, souffre de nombreuses contraintes :

— la configuration d'une session dans la station de travail est fastidieuse, peu commode, et propice aux erreurs,

— la faible résolution du protocole MIDI (8 bits) peut se révéler contraignante pour encoder certains paramètres de spatialisation tels que la distance d'une source ou son angle d'azimut ; un pis-aller consiste à encoder ces paramètres sur plusieurs contrôleurs MIDI (par exemple un contrôleur pour le poids fort et un contrôleur pour le poids faible) mais ceci alourdit encore considérablement la mise en œuvre,

— le nombre d'objets pilotables est limité par le nombre de canaux MIDI disponibles, c'est-à-dire 16,

— les messages MIDI n'étant pas « typés », les informations échangées entre le DAW et l'application auxiliaire sont peu « lisibles », ce qui nuit à la maintenance et à la pérennité d'une session de travail.

Ainsi nous avons été amenés à développer un outil ad-hoc, *ToscA*, permettant de palier les faiblesses de l'intercommunication via MIDI. L'outil repose sur le protocole Open Sound Control (OSC) [15], qui a été choisi en raison de sa large adoption dans les applications d'informatique musicale, et de sa facilité de mise en œuvre.

3. TOSCA

ToscA [8] s'insère en tant que plugiciel audio dans le DAW, sur chacune des pistes à spatialiser. *ToscA* n'affecte aucunement le signal audio qui est simplement shunté (*bypassé*) ; en revanche le *plugin* expose un certain nombre de paramètres automatisables. Dans un souci de compatibilité avec une vaste gamme de stations audionumériques, le nombre maximal de paramètres exposés par piste est limité à 32.

Lorsque ces pistes d'automation sont activées en lecture, *ToscA* envoie les données de ces pistes en OSC. De même, si les automations sont armées en écriture, *ToscA* peut recevoir des paquets OSC depuis une application distante, et inscrire ces données dans les pistes du DAW.

3.1. Syntaxe des messages

Chaque instance de *ToscA* possède un identifiant (ID) qui est réglable par l'utilisateur (voir Figures 2 et 3). Cet identifiant constitue la racine des messages OSC émis/reçus par *ToscA*. Leur syntaxe prend la forme suivante :
« /ID/NomParamètre ValeurParamètre »

8. ToscA est l'acronyme de Thibaut's OpenSoundControl Automation.

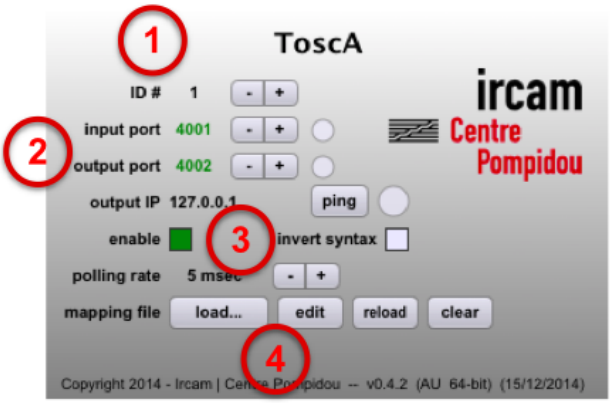

Figure 2. Vue du plugin ToscA. ① sélection de l'ID du *plugin*. ② configuration des ports UDP d'entrée/sortie ; adresse IP de destination. ③ inversion optionnelle de la syntaxe des messages. ④ chargement ou édition du fichier de configuration.

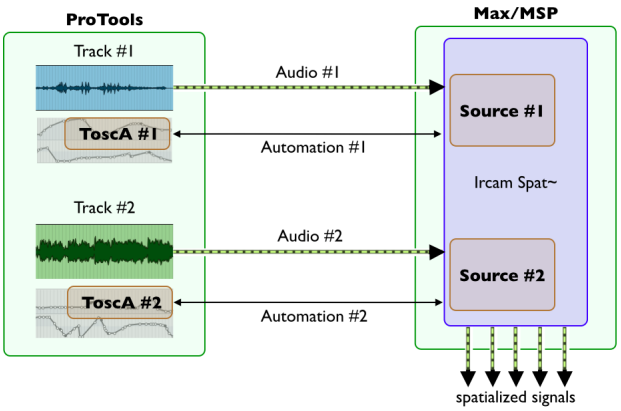

Figure 3. Exemple typique d'utilisation de *ToscA* pour piloter le *Spatialisateur*.

par exemple : « /3/azimuth 135.0 »
Dans un souci de simplicité, les paramètres dans *ToscA* sont tous des nombres flottants en double précision. Les autres types de données possiblement encapsulables dans un paquet OSC (tels que nombres entiers, chaînes de charactères, etc.) ne sont actuellement pas gérés.

3.2. Configuration par fichier de mapping

ToscA n'est pas lié à un moteur de spatialisation en particulier, autrement dit, les paramètres d'automation qu'il expose sont génériques [9]. Par défaut, ces 32 paramètres se nomment *param1*, *param2*, etc. Le libellé des paramètres peut ensuite être configuré. Ceci ce fait par le biais d'un fichier de configu-

9. Au reste, *ToscA* peut parfaitement être employé pour piloter des applications autres que des moteurs de spatialisation.

ration (fichier dit de *mapping*) qui doit être chargé dans *ToscA*. Il s'agit d'un fichier XML dont la syntaxe est présentée sur les figures 4 et 5.

Par souci de simplicité (et de rapidité de configuration de la session de travail), les noms des paramètres sont les mêmes pour toutes les instances du plugiciel, c'est-à-dire pour toutes les pistes dans le DAW : lorsqu'un fichier de configuration est chargé (ou édité) dans une des instances de *ToscA*, toutes les autres instances se mettent conformément à jour.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<tosca version="0.4">
    <parameter index="1" name="x" min="-10" max="10" scaling="linear"/>
    <parameter index="2" name="y" min="-10" max="10" scaling="linear"/>
    <parameter index="3" name="z" min="-10" max="10" scaling="linear"/>
</tosca>
```

Figure 4. Exemple de fichier de configuration : déclaration de trois paramètres d'automation correspondants à des coordonnées cartésiennes.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<tosca version="0.4">
    <parameter index="1" name="azim" min="-180" max="180" scaling="linear"/>
    <parameter index="2" name="elev" min="-90" max="90" scaling="linear"/>
    <parameter index="3" name="gain" min="-60" max="0" scaling="linear"/>
    <parameter index="4" name="aperture" min="0" max="180" scaling="linear"/>
    <parameter index="5" name="orientation" min="-180" max="180" scaling="linear"/>
</tosca>
```

Figure 5. Autre exemple de fichier de configuration.

Le fichier XML de configuration permet en outre de spécifier, pour chacun des paramètres, une mise à l'échelle (« scaling ») des valeurs d'automation : dans l'environnement du DAW, les pistes d'automation sont codées en flottant sur [0 – 1] ; *ToscA* réalise un *scaling* de [0 – 1] vers le domaine [min – max] choisi par l'utilisateur. Cette mise à l'échelle affecte les données sortantes et l'opération inverse est appliquée aux données entrantes. Le fait que *ToscA* gère directement le *scaling* a deux avantages :

— l'utilisateur n'a pas à le faire au niveau de l'application auxiliaire (contrairement à l'approche MIDI présentée au paragraphe 2.2),

— il est possible « re-dimensionner » un mixage existant, simplement en modifiant la plage [min – max] d'un ou plusieurs paramètres dans le fichier de configuration. Cela répond à une problématique courante en spatialisation : une même œuvre pouvant être donnée dans plusieurs lieux de dimensions ou d'acoustiques très disparates, le compositeur est souvent amené, pour adapter le rendu à la salle, à exagérer ou diminuer certains paramètres de spatialisation (voir par exemple [9]), notamment le rayon des trajectoires des objets spatialisés.

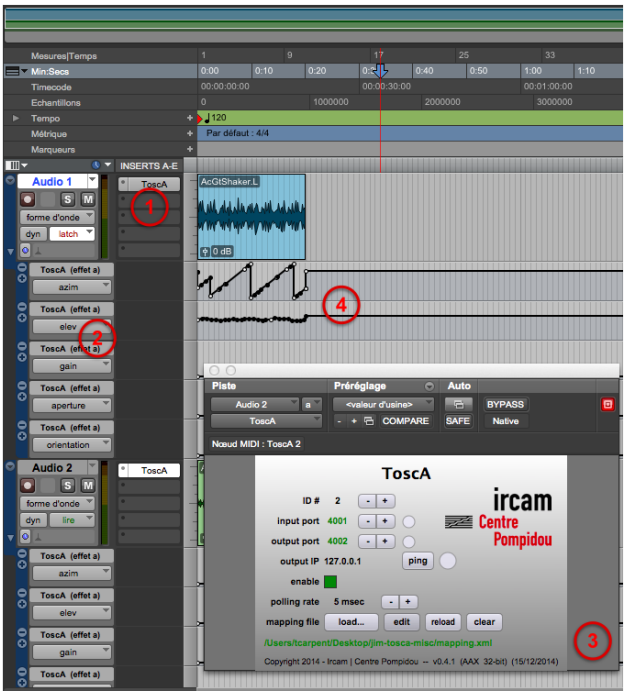

Figure 6. Vue du *plugin ToscA* dans ProTools. ① *plugin* en insert sur la piste 1. ② liste des paramètres exposés. ③ fenêtre du *plugin*. ④ pistes d'automation.

3.3. Modes touch/latch

Lors de l'*authoring* de scènes sonores spatialisées, il est souvent nécessaire de « re-toucher » les trajectoires des objets (ou d'autres paramètres) à plusieurs reprises. Pour cela on pourra utiliser les modes *touch* ou *latch* des DAWs qui permettent d'éditer partiellement une courbe d'automation. Il est nécessaire que l'application auxiliaire signale au DAW le début et la fin du *touch*. À cette fin, la syntaxe suivante a été développée dans *ToscA* :
« /ID/NomParamètre touch 1 » : le *touch* démarre
« /ID/NomParamètre ValeurParamètre » : la valeur du paramètre est mise à jour
« /ID/NomParamètre touch 0 » : le *touch* s'arrête.

Les modes *touch* ou *latch* pourront notamment se révéler très utiles dans le cas de l'édition de trajectoires spatiales à partir d'une interface tactile compatible OSC telle que Liine's Lemur [10] ou Hexler's TouchOSC [11].

3.4. Syntaxe alternative des messages

Certaines applications de spatialisation compatibles OSC adoptent une syntaxe de messages « inversée » par rapport à *ToscA*. Par exemple le *Sonic Wave I* de SonicEmotion [2]

10. http://liine.net/en/products/lemur
11. http://hexler.net/software/touchosc

utilise : « /Set/SrcPos/Azimuth/3 90.0 »
pour spécifier la position azimutale de la source #3. Afin d'assurer la compatibilité de *ToscA* avec de telles applications, une option « d'inversion de syntaxe » a donc été introduite (Figure 2). Cette option permet de remplacer la syntaxe traditionnelle :
« /ID/NomParamètre ValeurParamètre »
par :
« /NomParamètre/ID ValeurParamètre ».
Notons qu'il est tout à fait licite d'utiliser un ou plusieurs caractères "/" dans le nom des paramètres d'automation. Dans l'exemple ici présenté, on aura pris « Set/SrcPos/Azimuth » comme « NomParamètre ».

3.5. Développement logiciel

ToscA a été développé avec la bibliothèque logicielle (« framework ») Juce [12]. Juce comprend une bibliothèque d'enveloppe (« wrapper ») de plugiciels qui permet, à partir d'une base de code commune, de générer des *plugins* dans différents formats (VST, AudioUnits, etc.). Ainsi, *ToscA* est disponible sous MacOS et Windows, en format VST, VST3, AU, AAX, en mode 32 ou 64 bit. Il est distribué librement via le Forum Ircam [13].

4. CONCLUSION ET PERSPECTIVES

Nous avons développé *ToscA*, un *plugin* qui permet la communication en OSC entre une station de travail et une application auxiliaire déportée dans un autre environnement. En particulier, nous avons présenté le cas d'usage d'un mixage spatialisé.

ToscA se veut tout à fait « œcuménique », et permet de contrôler tout type de paramètre. En pratique, les besoins des compositeurs/mixeurs concernent essentiellement l'*authoring* de trajectoires spatiales. C'est pourquoi un nouveau plugiciel est en cours de développement : *ToscA*-Spat (Figure 7) reprend les principes de *ToscA* (i.e. il ne traite pas l'audio mais il lit/écrit les automations) et intègre en outre, dans le DAW, une fenêtre de visualisation de la scène sonore. Cela permettra une mise en œuvre encore plus rapide et intuitive dans la chaîne de production (par exemple le fichier xml de « mapping » sera automatiquement généré en fonction de la scène sonore construite).

5. REFERENCES

[1] Bresson, J., Schumacher, M., « Representation and interchange of sound spatialization data for compositional applications », *International Computer Music Conference*, Huddersfield, Royaume-Uni, 2011.

12. http://www.juce.com
13. http://forumnet.ircam.fr/product/spat/tosca/

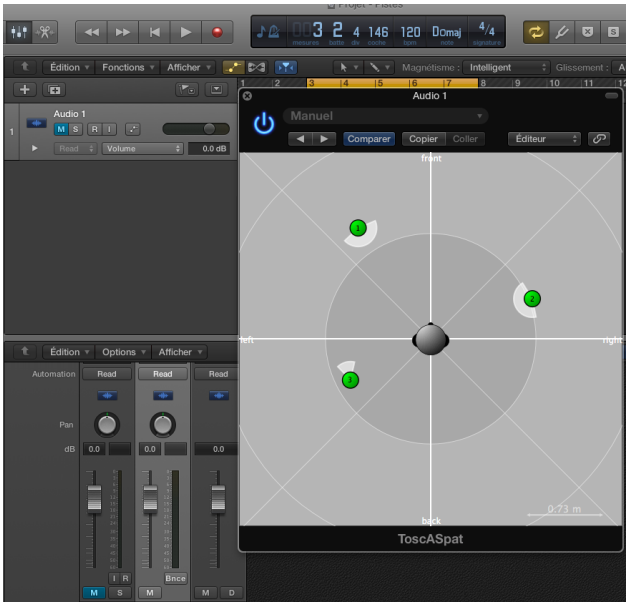

Figure 7. Vue du *plugin ToscA*-Spat dans Logic Pro.

[2] Corteel, E., Glaettli, P., Foulon, R., Frauly, R., Hahn, I., Heiniger, R., Pellegrini, R., « 3D speaker management systems – Mixer integration concepts », *28th Tonmeistertagung VDT International Convention*, Cologne, Allemagne, 2014.

[3] Daniel, J., « Représentation de champs acoustiques, application à la transmission et à la reproduction de scènes sonores complexes dans un contexte multimédia », thèse de doctorat de l'Université Paris VI, 2001.

[4] Geier, M., Spors, S., « Spatial Audio Reproduction with the SoundScape Renderer », *27th Tonmeistertagung VDT International Convention*, Cologne, Allemagne, 2012.

[5] Jot, J.-M., « Real-time Spatial Processing of Sounds for Music, Multimedia and Interactive Human-Computer Interfaces », *ACM Multimedia Systems Journal (Special issue on Audio and Multimedia)*, 7(1), 1997, p. 55 – 69.

[6] McGee, R., Wright, M., « Sound Element Spatializer », *International Computer Music Conference*, Huddersfield, Royaume-Uni, 2011.

[7] Noisternig, M., Carpentier, T., Warusfel, O., « Espro 2.0 – Implementation of a surrounding 350-loudspeaker array for sound field reproduction », *Spatial Audio in today's 3D World - AES 25th UK Conference*, York, Royaume-Uni, 2012.

[8] Penha, R., Pedro Oliveira, J., « Spatium, tools for sound spatialization », *Sound & Music Computing Conference*, Stockholm, Suède, 2013.

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

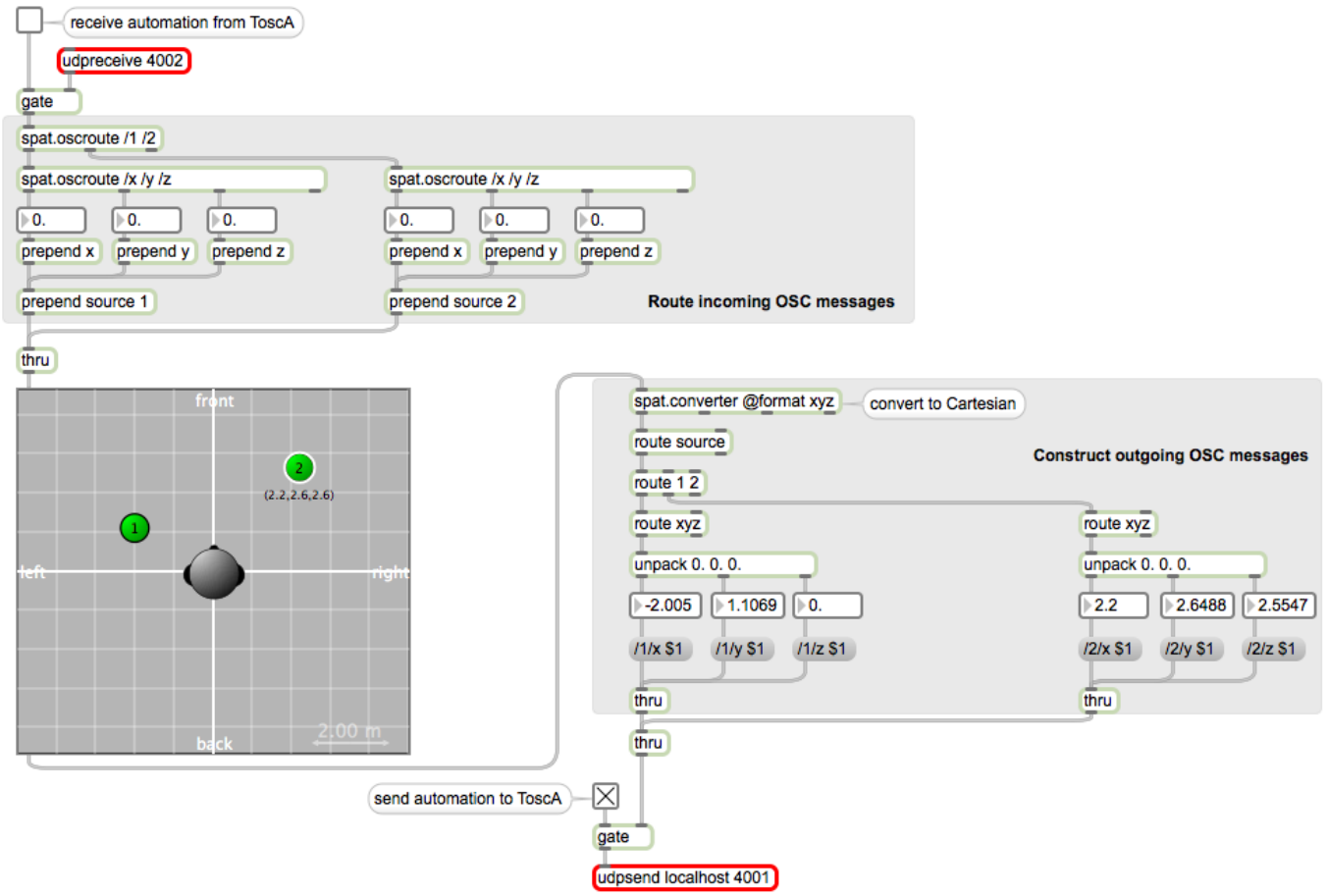

Figure 8. Interfaçage avec le *Spatialisateur* dans Max/MSP : exemple basique du *spat.viewer* (avec deux sources) en communication avec *ToscA*.

[10] Peters, N., Lossius, T., Schacher, J., « The Spatial Sound Description Interchange Format : Principles, Specification, and Examples », *Computer Music Journal*, 37(1), 2013, p. 11 – 22.

[11] Pulkki, V., « Generic Panning Tools for Max/MSP », *International Computer Music Conference*, Berlin, Allemagne, 2000.

[12] Ramakrishnan, C., « Zirkonium : Non-invasive software for sound spatialisation », *Organised Sound*, 14(3), 2009, p. 268 – 276.

[13] Schacher, J., « Seven years of ICST ambisonics tools for Max/MSP - A brief report », *2nd International Symposium on Ambisonics and Spherical Acoustics*, Paris, France, 2010.

[14] Sèdes, A., Guillot, P., Paris, E., « The HOA library, review and prospects », *International Computer Music Conference / Sound & Music Computing conference*, Athènes, Grèce, 2014.

[15] Wright, M., Freed, A., Momeni, A., « Open Sound Control : State of the Art 2003 », *International Conference on New Interfaces for Musical Expression*, Montréal, Canada, 2003, p. 153 – 159.