Trajectory Score Library : A Tool for Algorithmic Spatialisation with Antescofo

Nadir Babouri

TRAJECTORY SCORE LIBRARY: A TOOL FOR ALGORITHMIC SPATIALISATION WITH ANTESCOFO

Nadir Babouri
Spectacle Vivant & Art Sonore
info@nadirbabouri.fr

ABSTRACT

This paper presents Trajectory Score Library, an Antescofo library of scripts, converting mathematical parametric functions into curves in order to control the Spat5 sources trajectories. These scripts are written in the Antescofo language, a timed synchronous language which allows the use of relative time to synchronize the trajectories with musical events. This language can also provide a convenient way of creating and transmitting automation parameters to the Spat5 through the Open Sound Control protocol implemented in Ascrentescofo. Trajectory Score Library is an additional tool amongst existing spatialization solutions. In the context of this presentation these solutions are considered as remote-control software. Trajectory Score Library thus aims to use Max as a unified real-time environment to unite different computer music processes.

1. INTRODUCTION

Antescofo [8] combines a real-time listening machine with a reactive and timed synchronous language [9]. The language is used for the authoring of music pieces involving live musicians and computer processes. The real-time system assures its correct performance and synchronization. In our case, the computer process is a refined control of sound source trajectories in the Ircam Spatialisateur, a part of the spatial audio rendering [1]. Spatialization, as part of the writing of a piece of music, is designed to create for the listener a given « spatial impression »; a generic term that actually brings together several notions grouped into two families:

— Position and spatial quality of the sound sources: absolute or relative orientation of the listener, distance from the listener, width and apparent depth/location accuracy, etc.
— Perception of the sound space: feeling of envelopment, perception of the dimensions of the sound space, reverberation, etc.

All of these perceptual aspects of spatial sound are strongly linked to geometric parameters, like the physical position of the sources or the geometry of the room, and to physical parameters, like the acoustic properties of the walls of the sound stage, real (concert situation) or virtual (created by studio mixing) [2].

2. FOUNDATIONS

Trajectory Score Library [2] is a tool conceived in the context of composing for spatial audio involving dense streams of various control data, among other electronic processes within a score. Until now the control of sound source trajectories was handled through remote applications, so the aim of this library is to gather the different electronic processes into the Max real-time environment [13], augmented with a time structure provided by the reactive and timed synchronous language in Antescofo.

An Antescofo score is a text file that is used for real-time score following [3] and triggering electronic actions as written during composition. It is common in contemporary instrumental writing for composers to include bits of code for the electronic part in their compositional process. Considering spatialization, the author previously relied on remote applications with GUI interfaces, allowing the drawing of trajectories or including some script to generate trajectories. Aside from this, extra programming was needed within Max, the main real-time environment, in order to communicate with the remote application, trigger the trajectories, collect the stream automated data, scale it in some cases and synchronize it with musical events which required accessing the transport and timeline of the remote application. Additionally, going back and forth between environments can be quite time consuming. Fortunately, most of these applications were implemented with the OSC [4] protocol which enhanced the interaction between them. These considerations do not mean that the remote applications are not efficient. Each one of them is useful according to a technical or artistic process i.e. whether we are writing, rehearsing, playing or recording music.

[1] Other parts of spatial rendering could be the creation of virtual acoustic spaces, auditory scenes, interactive content, 3D mixing and diffusion.

[2] See forum.ircam.fr/projects/detail/trajectory-score-library.

[3] Detecting the position and tempo of live musicians in a given score

[4] See cnmat.berkeley.edu/OpenSoundControl/.

3. SPAT5 SPATIALISATION CONTROL: A STATE OF THE ART

Spat5 [5] [7] is a library of multiple software modules and algorithms that users can combine in order to realize the spatial audio treatment they need. Spat5 tends to be completely flexible and addresses any type of audio setup environment from stereo to complex and ambitious ones [6]. As a real-time spatial audio processor, Spat5 allows composers, sound artists, performers, and sound engineers to control the localization of sound sources in 3D auditory spaces.

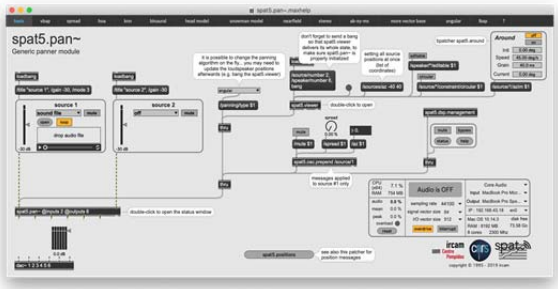

Figure 1 shows a Spat5 external called Spat5.pan~ and its help patcher included in the Max Spat5 Overview.

Spat5 provides a powerful reverberation engine that can be applied to real and virtual auditory spaces. The processor receives sounds from instrumental or synthetic sources, adds spatialization effects in real-time and outputs signals for reproduction on an electro-acoustic system (loudspeakers or headphones). Spat5 provides rich and multifold interfaces for monitoring real-time spatialization in Max. The Spat5 environment does not provide a time-based interface or transport for composing spatialization. Fortunately, the OSC protocol is fully embedded, and communication with remote applications is robust and convenient due to the url-like syntax. Some of these applications are considered, in this paper, as host environments.

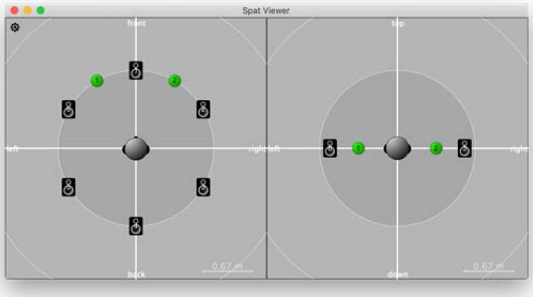

Figure 2. Spat5 viewer for sources and speaker visualization and manipulation

Authoring tools for spatial audio are influenced by the orientation of these host environments or of the technological frameworks they fit in, be it from computational, representational or user interaction points of view. In the following subsection, we present some applications used in different contexts.

3.1. Max

Here is an approach to control the trajectories of audio sources within Max, using an implementation of an *lcd*, a *buffer*, a *waveform* and *function* objects [7]. Drawing a trajectory in the *lcd* object will generate, via a program, a script with the convenient Spat5 prepended syntax. It can be triggered or saved to control the trajectory of sound sources. This configuration is quite useful during composition or rehearsals. The lack of a timeline makes it difficult to be precise in the drawing of a trajectory, besides the time format is absolute. The waveform display is useful in order to correlate a spatialization with the amplitude variation of the sound source.

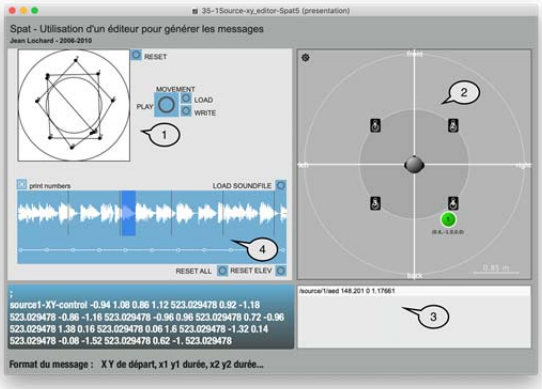

Figure 3 shows a Max patch used to generate spat5 trajectories. 1–An lcd object, 2–Spat5 Viewer, 3–Spat5 coordinates, 4–An audio source waveform.

3.2. Holo-Edit

Holo-Edit is a graphical and algorithmic editor of sound trajectories [1]. It is possible, within the application, to draw and record position in real time from an external program or device via the OSC protocol, and to graphically edit trajectories across a complex sound system. Holo-Edit allows for precise positioning of sound sources in time and space defined by a set of speakers. In order to do so, it associates sound sources to trajectories, a set of points defined by their position in space and their time stamp. The application also offers different ways to create, transform, and manipulate sound trajectory.

These actions can be launched from both sides. More calculations are needed to achieve synchronization with musical events, and Holo-Edit's timeline is in absolute time.

[5] The focus of this article is on the Ircam Spatialisateur. Nevertheless, the Trajectory Score Library is generic and produces coordinates that can control other spatialization engines.

[6] Ambisonics, wavefield field synthesis.

[7] Max native objects.

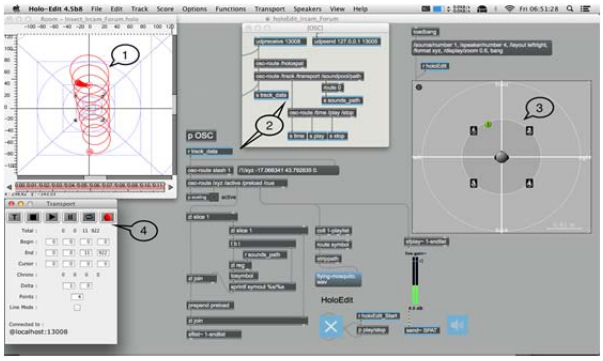

Figure 4. Spat5 interaction with Holo-Edit through Max. 1–Holo-Edit's Room, 2–OSC interface, 3–Spat5 Viewer, 4–Holo-Edit's Transport.

The automated parameters can also be read off-line from a Holo-Edit exported XML [8] file, through a Max program implementing the use of RegExp [9] formatting and parsing data in order to fit the Spat5 syntax.

3.3. Iannix

IanniX [12] can be used as a tool for the creation and the performance of musical scores with a graphic representation. Through various communication protocols, it synchronizes single events as well as continuous data to external environments. Many object attributes, as well as various mapping modes, allow the user to match the characteristics and the behavior of cursors, curves, and triggers to sound and music parameters and several MIDI messages. Specific usages of IanniX include the control of sound spatialization, both for the definition of virtual sound trajectories and the routing of audio signals in complex sound projection systems.

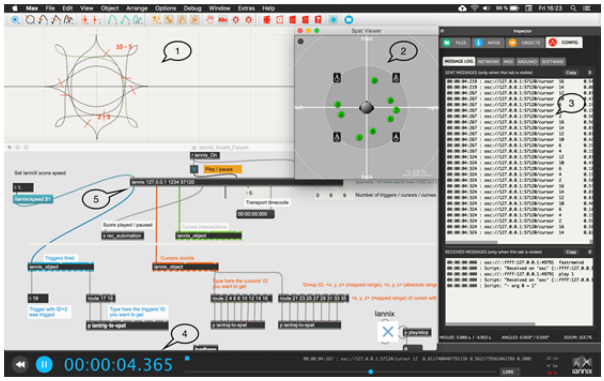

Figure 5. Spat5 interaction with Iannix. 1–Iannix score editor, 2–Spat5 Viewer, 3–OSC messages, 4–Iannix transport, 5–Iannix Max object.

A set of Max patches is provided by Iannix developers to link Iannix to the Spat5 engine via OSC. The remote application transport can also be triggered from both sides. More calculations are needed to achieve synchronization with musical events and Iannix's timeline is in absolute time.

3.4. OpenMusic

OM-Spat is a library for the creation and rendering of spatial scenes in the computer-aided composition software, OpenMusic [3]. This library implements different objects that allow the generation and storage of source trajectories and spatial attributes. The rendering is done off-line by the Spat kernel [10] as a binaural or transaural audio file. A sampling of the trajectories can also be exported as sequences of data frames in SDIF format [11]. This file can be played and streamed to Spat by the Spat-SDIF-Player [4], a stand-alone Max application. The articulation with time consists of a list of durations, in milliseconds, on which sound trajectories are mapped.

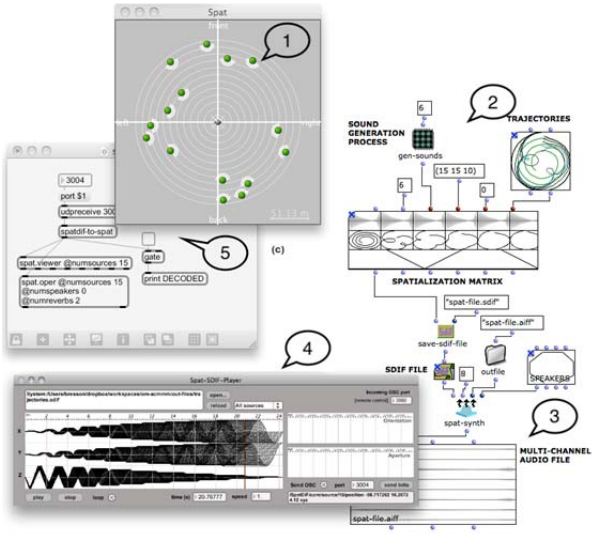

Figure 6. OM-Spat environment. 1–Spat5 Viewer, 2–OM trajectories generation, 3–Spat Renderer, 4–Spat-SDIF-Player, 5–UDP communication.

3.5. SPAT-SCENE

SPAT-SCENE was built upon observation of some composers' practice at Ircam [10]. Its aim is to help them structure real-time spatialization processing as part of a compositional approach. It is a graphical user interface coupled to a spatial sound processor embedded in OM-sharp, an application derived from OpenMusic [5]. It combines the interface and the rendering engine of the Ircam *Spatialisateur* with compositional and temporal control of spatial scene descriptors, all in a unified framework. The interface provides orthogonal views of the space vs. time dimensions of spatial sound scenes: it allows layouts of sound sources to be specified at specific point, and the timed trajectories, for individual sound sources, to be edited and synchronized.

[8] XML stands for eXtensible Markup Language and is designed to be self-descriptive.

[9] RegExp, short for regular expression.

[10] Spat renderer is an external executable included in the Ircam Spat5 distribution.

[11] Sound Data Interchange Format.

The SPAT-SCENE object can be rendered in different ways. In the context of compositional vs. real-time interactions, it can act as a controller or monitoring interface for the real-time spatialization processes.

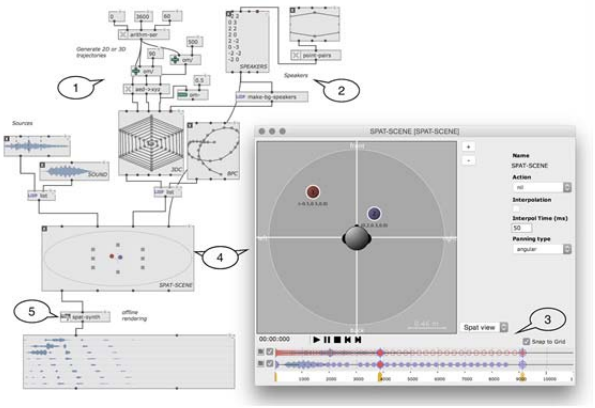

Figure 7. SPAT-SCENE interface in the OM-sharp environment. 1–OM trajectories generation, 2–Speaker settings, 3–SPAT-SCENE transport and timeline BPF, 4–Spat-SCENE, 5–Spat Render.

3.6. OSCar

OSCar [6] is a plugin that uses the OSC protocol to transmit automation parameters between a digital audio workstation and a remote application. The main use case is the production of massively multichannel object-oriented spatialized mixes. The developer proposes a workflow where the spatialization rendering engine [12] lies outside the workstation. OSCar plugin is completely generic and it can control any type of parameters. It can be inserted on an audio track and during playback, active automation tracks and their parameters are read and the corresponding OSC messages are sent over UDP.

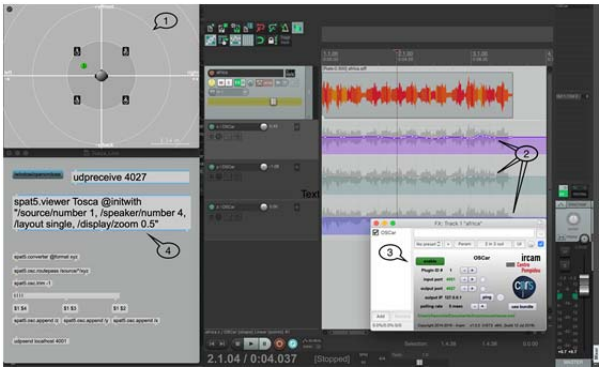

Figure 8. OSCar interface communication between Reaper and Spat5. 1–The Ircam *Spatialisateur* viewer; 2–Spatial coordinates automation lanes; 3–OSC interface, 4–Spat5 Viewer max object connected to a UDP receiver.

[12] OSCar is not tied to a specific spatialization renderer and the exposed automations are generic.

This method affords the fast and intuitive workflow for geometrical editing in the automation lanes and a precise reference to the workstation timeline in absolute or relative time.

3.7. Symbolist

Symbolist is a graphic notation environment for music and multimedia [11]. It is based on an OSC encoding of symbols representing multirate and multidimensional control data, which can be streamed as control messages to audio processing or any kind of media environment. Symbols can be designed and composed graphically. The environment provides tools for creating symbols groups and stave references, by which symbols may be timed and used to constitute a structured and executable multimedia score.

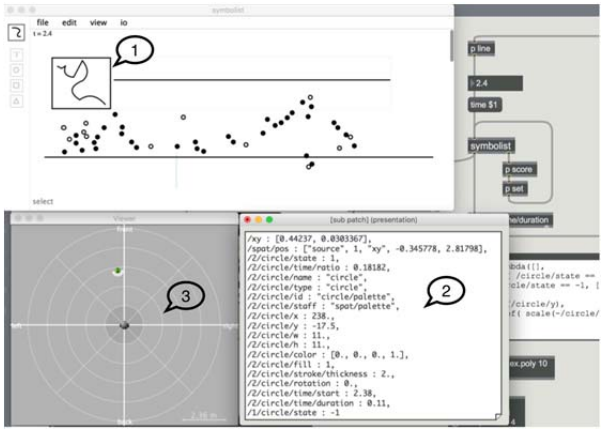

Figure 9. Symbolist graphic interface and source trajectory and some audio synthesis. 1–A notation example to control a source trajectory in spat5, 2–OSC messages, 3–Spat5 Viewer.

4. COMPOSING WITHIN A UNIQUE ENVIRONMENT

The implementation of the trajectory algorithms through Antescofo brings the control of audio spatialization to a unified framework for the composition of mixed music. See Figure 10.

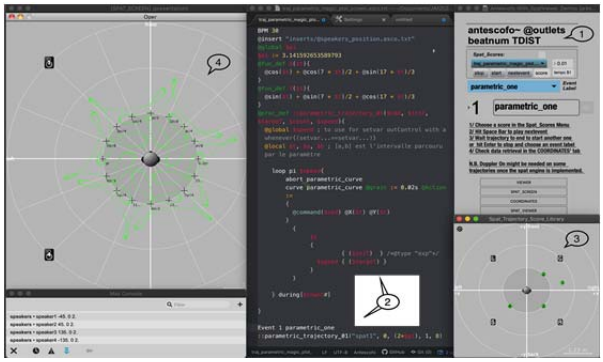

Figure 10. Trajectory Score Library demonstration: 1–Antescofo max object, 2–An electronic score and a defined trajectory algorithmic process, 3–The Ircam

Spatialisateur viewer, 4–An LCD interface showing a source trajectory generated by the algorithm.

Indeed, the Antescofo engine and its reactive timed synchronous language enable fluid process for composing several layers of refined synthesis and control over the electronic part of a composition. The language can be used to do sequencing over messages to be sent to Max, using musical values expressed in beats, and relative to the initially declared score tempo.

4.1. Antescofo Language Scripts

Antescofo uses scores [13] to control different computer processes in mixed music. These scores are simple text files [14] or scripts, used for real-time score following as well as the computation and triggering of electronic and musical events as conceived during composition. Trajectory Score Library is a collection of scripts, written in the Antescofo language that includes several methods, actions, functions, definitions, and processes used to program the algorithmic control of spatialization. One of these definitions is called `@fun_def` [15] where an intentional function f is defined by arbitrary rules (i.e. by an expression) that specify how an image `f(x)` is associated to an element x. Its simple formula is:

```
@fun_def @name($arg1, $arg2, ...){expression}
```

One of the spatial algorithms, included in the library, defines a circle trajectory, and the user has access to its parameters inputs to control the source movement spatially and temporally.

The cartesian equation of a circle:

$$x^2 + y^2 = r^2 \quad (1)$$

In order to implement the circle mathematical equation, it has to be transformed into a parametric equation,

$$\cos(\theta) = \frac{x}{r}, \sin(\theta) = \frac{y}{r} \quad (2)$$

$$x = r * \cos(\theta), y = r * \sin(\theta) \quad (3)$$

then declared within an Antescofo language script ready to be used as an expression:

```
@fun_def X($t, $r, $offsetX){$r *@cos($t)+$offsetX}
@fun_def Y($t, $r, $offsetY){$r *@sin($t)+$offsetY}
```

Where t ∈ [0, 2π] and offsetX and offsetY are the coordinates of the center of the circle.

Any call to `@X()` and `@Y()`, anywhere in the action language, where an expression is allowed (inside messages, etc.), will be replaced by its value at run-time. We will use an action language called `Curve` [16] to apply these definitions and generate x and y values at run-time.

```
Curve receiver [17], starting_point, final_point,
duration
```

Curves in Antescofo allow for the definition of continuously sampled actions on break points and detailed control of the interpolation between them. As time passes, the curve is traversed, and the corresponding action fired at the sampling point.

Then we create a process to consolidate the control of the action of the curve with all the necessary parameters and make it possible to call the process through an inserted script including the math.

```
@global $speed, $pi
$pi := 3.141592653589793
@fun_def X($t, $r, $offsetX){$r*@cos($t)+ $offsetX}
@fun_def Y($t,$r,$offsetY){r*@sin($t) + $offsetY}
@proc_def circle($cmd, $iniT, $targeT, $r, $offsetX,
$offsetY, $count, $speed){ curve circleCurve
@grain:=0.01 @Action := { @command($cmd)
@X($t,$r,$offsetX) @Y($t,$r,$offsetY)}
{$t
{($initT*$pi)}
$speed{($targetT*$pi)}}}
```

We finally call the process from the score with a simple command line. In the case of the circle script the parameters are:

```
::circle($cmd,$iniT,$targeT,$r,$offsetX,$offsetY,$cou
nt,$speed)
```

`$cmd` : The sound source.
`$iniT`: Source initial position.
`$targeT`: Source target position.
`$r` : The circle's radius.
`$offsetX` : Circle position according to xx' axe.
`$offsetY`: Circle position according to yy' axe.
`$count` : N circle loop.
`$speed` : The duration of the trajectory.

[13] See `support.ircam.fr/docs/Antescofo/manuals/UserGuide/structure/#structure-of-an-antescofo-score`

[14] That can be edited using a syntax highlighting for an editor like Atom. Developed by Nadir Babouri., Clément Poncelet and Benjamin Levy. See `atom.io/packages/atom-antescofo`.

[15] See `support.ircam.fr/docs/Antescofo/manuals/Reference/functions_def/`.

[16] See `support.ircam.fr/docs/Antescofo/manuals/Reference/4-compound/#compound-actions`.

[17] It can be a Max receiver, Supercollider, or any other computer music environment.

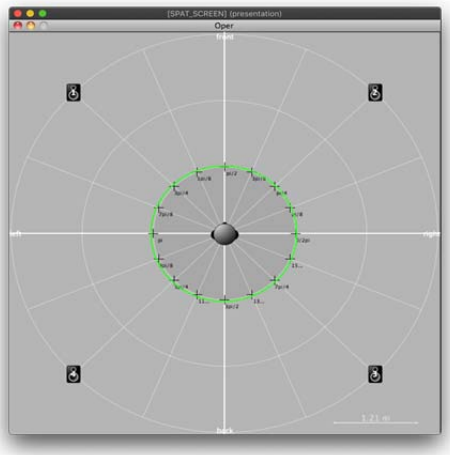

Figure 11. A circle trajectory process

The outcome of the computation will be forwarded to the Spat engine through OSC and prepended with the Spat5 syntax: `/source/1/xy $1 $2`. With the process: `::circle("spat1", 0, 2, 1, 0, 0, 1, 4)` the source spat1" will make one turn counter-clockwise with a radius of 1 during 4 beats. The ready-made trajectories in the Trajectory Score Library can be easily transformed through the inputs to the algorithm [18]. In the example of the circle trajectory process, if we would like to change the radius and the position of the circle, we need to change the `$r` and the `$offset` parameters:

```
::circle("spat1", 0, -2, 1, 0, 0, 1, 4)
::circle("spat2", 0, -2, 0.5, -2, 2, 1, 4)
::circle("spat3", 7/4, 2+7/4, 1, 1., 1., 1, 4)
::circle("spat4", 0, -2, 3, 0, 0, 1, 4)
```

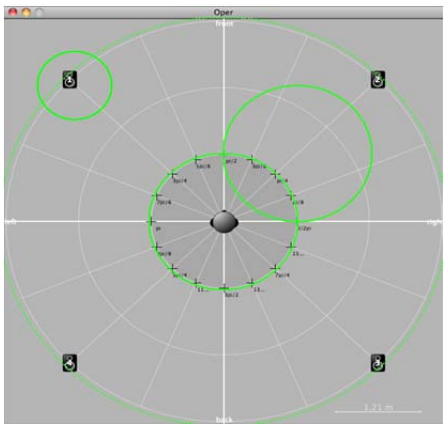

Figure 12 shows the produced trajectories in the Spat Viewer.

We can at the same time control, for example, the aperture and the yaw of all the sound sources [19] using another Antescofo construction called Group [20] in the score [21]:

```
Group position-orientation-diffusion {
oscsend src_aperture "localhost" :
4072"/source/*/aperture" source_aperture 10
oscsend src_yaw "localhost" : 4072 "/source/*/yaw"
src_yaw 180}
```

4.2. Time and Score Performance

Speaking about time is very easy in Antescofo [22] [9]. An Antescofo score can be seen as a sequencer, where all the actions are organized in time that we specify in a text file. We can write an electronic score with the beat notation and decide later to change the tempo. In this case we don't have to rewrite all the durations. Antescofo does the translation. This allows us to change the tempo with the `BPM` declaration at any place in a score. We can also work within different times simultaneously using the `@tempo` attribute in order to write complex polyrhythms. We can accordingly control trajectories sequentially and synchronize them precisely to electronic and musical events. When the time unit is declared, through the `BPM` assignment, all the delays between the actions and the durations of the processes become relative to that `BPM` whether it is fixed or depending on the tempo of the musician and its computation by the listening machine.

```
BPM 60
@insert"inserts/@rectilinear.asco.txt"
@insert "inserts/@lissajou.asco.txt"
Event 1 spat1_mvt_01
::rectilinear("spat1", 0, 2, -2, 2, 2, 0, 1, 1)
1
::rectilinear("spat1", 0, 2, 2, 2, 0, -2, 1, 1)
1
::rectilinear("spat1", 0, 2, 2, -2, -2, 0, 1, 1)
1
::rectilinear("spat1", 0, 2, -2, -2, 0, 2, 1, 1)
4
::lissajou("spat1", 1/3, 3, 2, -2, 4, 3, 1, 16)
```

In the example above, once the event is detected or launched by the musician, the audio source "spat1" will first achieve four-line trajectories of a duration of 1 beat. Each will be triggered one after the other with a delay of one beat. A square trajectory is thus programmed. Four beats later, a Lissajous trajectory with a duration of 16 beats is launched moving the sound source in the loudspeaker projection setup. We can accordingly synchronize different processes relative to the declared BPM and to the notation score played by a musician or to a running electroacoustic piece. When events are

[18] See `support.ircam.fr/docs/Antescofo/manuals/Reference/10-process/`.

[19] The aperture is a ratio between direct and reverberated sound. Increasing the Aperture will ultimately diffuse it wider, excite more the room and then increase the reverb sensation. The orientation of the source is controlled by Yaw. It is the ability to turn the direction of the source 360 degrees.

[20] The group construction gathers several actions logically within one block that shares common properties of tempo, synchronization and errors handling strategies in order to create polyphonic phrases.

[21] We can also construct a process to control the change of the aperture and the yaw smoothly with a curve action.

[22] Voir `support.ircam.fr/docs/Antescofo/manuals/Reference/5-synchro/`.

detected by the listening machine in real time, and while the tempo of the musician is fluctuating, there exists some synchronization strategies that defines the temporal evolution of a process depending on the musician, to help the composer achieve the precision or evolution of sound he needs in the performance.

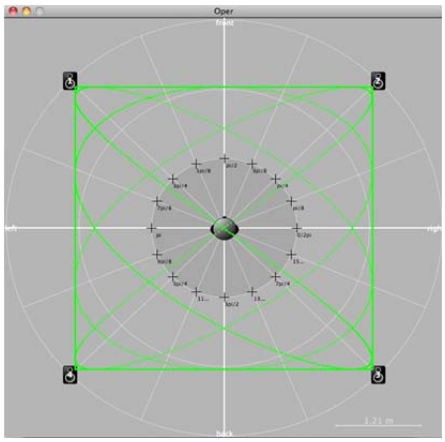

Figure 13 shows the produced trajectories of the spat1_mvt_01 Antescofo event.

5. CONCLUSION AND PERSPECTIVES

The author first presented a non-exhaustive set of external software utilities used to create and control audio sources trajectories in the Ircam Spatialisateur. Going back and forth between some of these host environments and Max can be time consuming. And extra programming is needed to trigger the trajectories, collect the stream automated data, scale it to our needs in some cases, and synchronize it with our musical events. To improve these operations, the author presented a first prototype of Trajectory Score Library. A set of scripts developed for editing and executing control data streams for audio spatialization. The aim of the library is to embed the compositional process, with the digital signal processing into a unified interactive system. The author would like to carry the development of this library due to the evolution of the Antescofo language and the Spat5 library. The notion of "object", widespread in language programming, can now be implemented, in order to organize code by gathering values together into a state, and making the possible interactions with this state explicit through the notion of methods [23]. This will help constructing varying and continuous complex trajectories. We can also write, from now on, the trajectory processes with the default parameters which will make the code more readable. Indeed, a process was declared this way:

```
::circle("spat1", 0, 2, 1, 0, 0, 1, 4)
```

Instead, the name of explicit parameters will be added, which will facilitate the reading and probing of the code:

```
::circle(/source, spat1, /iniT, 0, /targeT, 2,
/radius, 1, /offsetX, 0, /offsetY, 0, /count, 1,
/speed, 4)
```

Finally, extra inputs may be added to the algorithms to react to an external environment, such as gesture following.

6. ACKNOWLEDGEMENTS

The author would like to strongly acknowledge the kind support of Thibaut Carpentier, José-Miguel Fernandez, Jean-Louis Giavitto, Benjamin Levy and Clément Poncelet.

7. REFERENCES

[1] Bascou, C. « Adaptive Spatialization and Scripting Capabilities in the Spatial Trajectory Editor Holo-Edit », Proceedings of the 7th Sound and Music Computing Conference, Barcelona, 2010.

[2] Baskind, A. « Quelques notions sur la spatialisation », Ircam - Centre Pompidou, Formation Spatialisateur, Paris, 2008.

[3] Bresson, J., Schumacher, M. « Representation and Interchange of Sound Spatialization Data for Compositional Applications », Proceedings International Computer Music Conference, Huddersfield, UK, 2011.

[4] Bresson, J. « Spatial Structures Programming for Music », Spatial Computing Workshops (SCW), Valencia, Spain, 2012.

[5] Bresson, J., Bouche D., Carpentier T., Schwarz D., et Garcia J. « Next-generation Computer-aided Composition Environment: A New Implementation of OpenMusic », Proceedings of the International Computer Music Conference (ICMC), Shanghai, China, 2017.

[6] Carpentier, T. « ToscA: An OSC Communication Plugin for Object-Oriented Spatialization Authoring », Proceedings of the 41st International Computer Music Conference, Denton, USA, 2015, p. 368-371.

[7] Carpentier, T. « A new implementation of Spat in Max. », Proceedings of the 15th Sound and Music Computing Conference (SMC), Limassol, Cyprus, 2018, p. 184-191.

[8] Cont, A. « ANTESCOFO: Anticipatory Synchronization and Control of Interactive Parameters in Computer Music », International Music Conference (ICMC), Belfast, Ireland, 2008, p. 33-40. [Hal-00694803.]

[23] Voir `support.ircam.fr/docs/Antescofo/manuals/Reference/actors/#introduction-process-as-object`

[9] Echeveste, J., Giavito J.-L., Cont, A. *A Dynamic Timed-Language for Computer-Human Musical Interaction*. [Research Report] RR-8422, INRIA, 2013.

[10] Garcia, J., Carpentier T., Bresson J. « Interactive-compositional authoring of sound spatialization », *Journal of New Music Research* 46/1 (2017), p. 74-86.

[11] Gottfried, R., Bresson, J. « Symbolist: An Open Authoring Environment for End-user Symbolic Notation », International Conference on Technologies for Music Notation and Representation (TENOR'18), Montreal, Canada, 2018.

[12] Jacquemin, G., Coduys T., Ranc M. « Iannix 0.8 », Actes des Journées d'Informatique Musicale, Mons, Belgique, 2012.

[13] Puckette, M. « Combining Event and Signal Processing in the MAX Graphical Programming Environment », *Computer Music Journal* 15/3 (1991).

Texte édité par Tom Mays