Steering Behaviors for Spatial Sound Authoring

Thibaut Carpentier, Andrew Gerzso

Steering Behaviors for Spatial Sound Authoring

Thibaut Carpentier
IRCAM, CNRS, 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

ABSTRACT

This paper describes a software tool that metaphorically uses the concept of vector field in order to generate fluid, lifelike, motions of autonomous agents. The "steering behaviors" of such agents is potentially useful in a number of compositional contexts, where one wants to navigate through a parameter space in a physically-inspired and improvisational manner. The present article focuses mainly on spatial sound authoring, wherein the tool is used to create constantly evolving trajectories of sonic objects in a virtual space.

1. INTRODUCTION

1.1 Spatial trajectories as a compositional paradigm

Music in space and spatialized sounds have been a preoccupation of electroacoustic composers since the early experiments of *musique concrète* in the 1950s. From that point onwards, artists have explored a wide range of spatial composition techniques, and a number of spatial audio technologies have been developed in order to allow composers and performers to control spatio-musical attributes. Amongst the used techniques, movements of sounds have been reported one of the most popular (contemporary) compositional practices by several surveys [1–4]. Moving sound sources are typically used to introduce a choreography of sonic objects, to provoke a contrast between static and dynamic layers, and to help with the segregation of streams. In many compositional works, motions of sound sources in real or virtual spaces are apprehended as spatial trajectories: paths, curves, geometrical patterns, etc. constitue a key paradigm for the control of the spatiotemporal sound organization.

1.2 Techniques for spatial authoring

Within the framework of spatial trajectories, a great variety of tools have been proposed for the creation and manipulation of spatial data. Garcia et al. (section 2.3 in [4]) offers a review of spatial authoring frameworks, and also emphasizes the explicit separation of the authoring environment (in which composers edit or generate trajectories) from the rendering engine (that actually performs the audio processing). Broadly speaking, the approaches taken by trajectory authoring tools can be classified in three main categories:

- "low-level" trajectory editors that essentially treat spatial data as automation tracks (as found in Digital Audio Workstations). Typically, these tools allow for the "accurate" manipulation of trajectories through the use of breakpoint functions, spline or Bézier curves, free-hand drawing, quantized time-coordinate tuples, etc., optionally with transformative functionalities (rotation, translation, scaling). This category includes – but is not limited to: Zirkonium [5], Holo-Edit [6], Ambicontrol [7], SpaceJam [8], Trajectoires [9], etc., as well as DAW plugins such as Spatium [10] or ToscA [11].
- algorithmic-based techniques: this is a broad category that encompasses parametric or predefined trajectories (ellipse, spiral, etc.) [12], generative tools (Lissajou, Brownian, stochastic process, etc.), simulation of flocking and swarming movements [13, 14], physical models [10, 15], scattering sound particles [16], constraints engines [17], scripting or algorithmic compositional toolboxes [18], etc.
- symbolic editors offering a somewhat "higher-level" description of the spatial features represented as graphical symbols, spatial metaphors, or scores, in order to produce patterns, figures, or autonomous spatial scenarios. This notably includes the Spatialization Symbolic Music Notation framework [19], IanniX [20], i-score [21], and other attempts at symbolic notation for composing sound spatialization [22].

It must be noted that most authoring environments are not limited to one category or another, as they often provide mixed approaches. Similarly, composers are used to combining multiple perspectives, in order to alleviate the complexity and expressivity constraints of each tool, and to build their own idiosyncratic representation of spatial movements.

2. VECTOR FIELD AND STEERING BEHAVIORS

The work herein presented deals with a novel trajectory-based tool that shares similarities with the aforementioned editors, and borrows from both the algorithmically-driven and symbolic approaches. At its core is the notion of vector field, a physically-inspired representation, but approached from a somewhat metaphoric angle.

A vector field consists of a collection of points in space, called the domain, and a function that maps each point of the domain to one (and only one) vector. Vector fields are typically visualized in the 2D plane or in the 3D space, as a collection of arrows with a given length and direction, each attached to a point of the domain. They look like a "needle diagram". Vector fields are intensively used in physics as a way to model and depict the magnitude and orientation

of vectors, such as forces, velocities, etc. Typical examples include electric fields, magnetic or gravitational fields, fluid flows, or wind speed in weather charts. Vector fields offer a rather intuitive representation, e.g. of the velocity of a moving flow in space. When associated with a differential equation, vector fields often exhibit certain common shapes, such as a "source" (vectors emanating out of one point), a "sink" (vectors disappearing into a hole), or a "saddle point" (which looks like a horse's saddle).

`simone` is a software tool (further detailed in section 3) that builds on these paradigms. Its principle is very simple: the tool allows to generate and display a 2D vector field, thought as a velocity field; when objects or "particles" are "thrown" into this field, they become animated, and they navigate through the domain, as autonomous agents. This provides a quick and intuitive means to define and explore a space, e.g. a space of musical parameters, in a way that is partly predictable (a vector field is easily grasped at first glance), and partly "improvised" (singularities in the field can make the particles deviate from their expected trajectory). However, our claim is to make `simone` a creative tool for composers, and not a strictly accurate physical model. Therefore a number of metaphorical operators have been introduced, that can alter the behavior of the particles moving through the vector field, leaving room for further erratic and complex patterns.

Such an approach also closely relates to the notion of "steering behaviors". Steering behaviors is a term mostly used in game development, after the seminal work of Reynolds [23]: steering behaviors refers to the ability of autonomous characters to move in a realistic manner in their environment. These motion behaviors are described by simple forces and interactions that are combined to produce lifelike, improvisational navigation around the characters' domain. They do not rely on a global formulation of the characters' action (ultimate goal or long-term planning), but rather use local information, updating the agents' trajectory by incremental adjustment step by step. This makes them simple to understand and implement, but still able to produce rather complex movement patterns. Reynolds formalizes and gives many examples of steering behaviors, such as: seek (pursuit of a target), flee (inverse of seek), wander (random steering), collision avoidance (dodging around obstacles), path following (steer along a predetermined path), flow field following (aligning its motion with the local tangent, a behavior that clearly corresponds to the vector field situation), flocking (as implemented in the well-known *boids* algorithm [24]), leader following, etc.

3. PROPOSED TOOL

The graphical user interface of `simone` is presented in Figure 1. The window displays a 2D vector field, made of N rows and M columns, forming a domain of $N \times M$ cells (in this example $N = M = 17$). Each cell contains a vector (i.e. an arrow) characterized by its direction and velocity. Unlike traditional representations of vector fields, the velocity parameter is not depicted by the length of the arrow. Here, all arrows are drawn with the same (arbitrary) length, while the velocity factor is visualized by the light red circle (the value being proportional to the radius of the circle). This choice was made to improve readability and ease edition of the cells. The background color of the cells – here grey colors in a checkerboard pattern – has no meaning; this design was chosen only for readability and fast discrimination of the cells.

The vector (orientation and velocity) in each cell can be specified via message, or controlled with the mouse and a set of keyboard modifiers. In the current example, the vector field has a vortex-like shape (with randomized velocities) that was generated by a set of simple parametric equations implemented in javascript. The topology of the vortex can be distorted by the α and β parameters (dial buttons in Figure 1). Other sets of equations can be easily implemented.

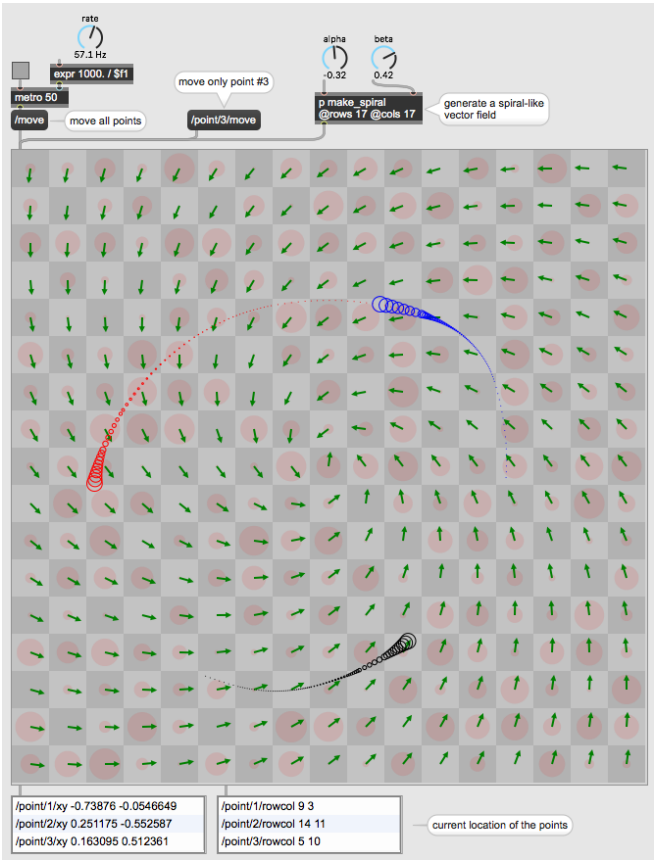

Figure 1: Graphical user interface of `simone`.

Given this vector field, one can "throw" particles (simply referred to as "points") in the field, by setting their initial position. The example in Figure 1 contains three points, represented in red, black, and blue respectively. At each computation tick the trajectory of the points is updated according to the cell they are currently over: the steering direction and the speed of the trajectory is altered by the vector of the cell.

The computation tick is triggered by sending the `/move` message, typically from a metronome. Changing the rate of the metronome allows to alter the speed of the trajectory. The `/move` command can be sent globally – applied to all particles – or independently for each point, so that each trajectory can operate at its own rate. At each computation step, the `simone` object delivers output messages with the current position and current cell for each point (see bottom of Figure 1). Cartesian coordinates are arbitrarily expressed in the $[-1; 1]$ range, and can later be scaled to a

more convenient or meaningful domain.

The "history" of each point (i.e. its previous locations) is depicted by a tail of dots (the older the history, the smaller the dots). History serves as a visual cue, and can further be used to alter the computation step: user can specified an "inertia" parameter (expressed in 0–100% range) that takes history into account in the update formulae. Higher inertia makes the trajectory smoother, while points with low value of inertia might suddenly change direction if the vector field exhibits singularities. The capacity of the history buffer (how many previous locations are retained) can be modify, which would in turn alter the impact of history in the inertia law.

The proposed vector field approach therefore offers a powerful and intuitive way to generate spatial motifs. Furthermore, trajectories can be flexibly modulated, by changing their rate, or via transformations of the underlying vector field. Indeed, the characteristics of the field can be updated in real-time, by several means:

- properties of the parametric equations (α and β variables in the present example) can be varied,
- external data (e.g. real-time weather charts) could be used to derive a new vector field,
- the mouse cursor can be turned into a "magnet" (either attractive or repulsive) so as to alter the directions of the field arrows; such magnetic manipulation can be applied on the whole domain, or only in user-defined sub-regions,
- other algorithmic strategies can be easily implemented. For instance, a point could modify the direction and/or velocity of each of the cells it has already visited, turning the field into a constantly evolving environment, etc.

As previously mentioned, `simone` is not intended to be a physical-model simulator; it is rather thought as an open environment that intuitively offers a wide range of patterns and behaviors. In order to further extend possibilities, a number of metaphorical operators have been introduced, digressing from the classical vector field paradigm.

3.1 Cell operators

Each cell in the domain can be assigned an operator, used in the computation iteration. In the example of Figure 1, every cell of the domain was assigned the same operator, a vector arrow representing a direction and a velocity value. Other types of operators are proposed, and depicted with graphical symbols (see Figure 2).

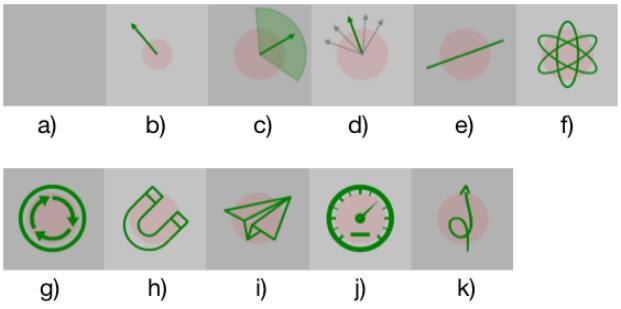

Figure 2: Various cell operators.

a) void cell: a void cell has no effect; it does not modify the nominal direction or the speed of a point reaching this cell.
b) simple vector, as previously described, imposes the new direction and velocity of a point, optionally affected by inertia.
c) ranged vector: similar to the simple vector, except that, for each computation tick, a new direction is randomly steered within the angular range.
d) radial cell: similar to the ranged vector, except that the new direction is randomly chosen within a discrete set of allowed paths.
e) paddle: provokes a reflection, depending on the incident angle of the trajectory.
f) random cell: steers a new direction randomly; similar to a ranged vector, with a 360° range.
g) repeller: provokes a perpendicular reflection of the incident trajectory.
h) attractor: produces an orbital path around the cell.
i) teleport: instantly transport to another (user-defined) cell or Cartesian position.
j) speed: changes the trajectory's velocity, without affecting its incident direction.
k) deviation: deviates the incident direction by a given amount (i.e. offset in steering direction).

Combining these operators in the domain allows to produce a wide range of improvised motions, similar to some of Reynold's steering behaviors. For example, "wander" or random walk can be implemented by means of ranged and radial cells.

Finally, one needs to deal with points leaving the domain (i.e. getting out of bounds of the $N \times M$ field). Available options include: let the trajectory disappear, "wrap" the point to the origin, to a given Cartesian position, to a given cell, or to a random location.

3.2 Software implementation

`simone` is a Max external object, written in `C++`, and built with the Juce framework. It is freely distributed with the Ircam `spat~` package [25]. Benefiting from the latest improvements in the `spat~` software architecture [26], `simone` features import/export of presets, snapshot compatibility, and OSC communication. In the context of vector field, the use of OSC pattern matching appears very handy and efficient as it allows to concisely address a subset of the domain (e.g. `/row/[2-5]/col/*/type repeller` will assign the 'repeller' type to all cells in the rows 2 to 5).

4. CONCLUSION AND PERSPECTIVES

We have presented a novel software tool that relies on the concept of vector field to generate spatial trajectories. With the inclusion of metaphorical operators, the trajectories act as autonomous agents, navigating through a domain in a lifelike and improvisational manner. The underlying vector field itself can be generated with various techniques, e.g. mathematical equations, hand-drawing, or data mapping, etc. The tool is intuitive to use, and is generic enough to be applied in a variety of contexts. Spatial sound authoring constitues a primary field of application, as trajectory-based technique is a most popular compositional practice. Another

obvious application of the proposed technique is corpus-based concatenative synthesis [27], wherein snippets of sounds are (re-)composed by navigating through a space of (musically meaningful) descriptors. Future work will investigate the usefulness of the tool in other generative compositional processes. New developments are also considered to further extend the expressivity of the tool, e.g. by adding new kinds of symbolic operators, or constraints between multiples trajectories so that several points can produce collective steering behaviors.

5. REFERENCES

[1] M. A. Baalman, "Spatial Composition Techniques and Sound Spatialisation Technologies," *Organised Sound*, vol. 15, no. 3, pp. 209 – 218, Dec 2010.

[2] N. Peters, G. Marentakis, and S. McAdams, "Current Technologies and Compositional Practices for Spatialization: A Qualitative and Quantitative Analysis," *Computer Music Journal*, vol. 35, no. 1, pp. 10 – 27, 2011.

[3] F. Otondo, "Contemporary trends in the use of space in electroacoustic music," *Organised Sound*, vol. 13, no. 1, pp. 77 – 81, April 2008.

[4] J. Garcia, T. Carpentier, and J. Bresson, "Interactive-compositional authoring of sound spatialization," *Journal of New Music Research*, vol. 46, no. 1, pp. 74 – 86, 2017.

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

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

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

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

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

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

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

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

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

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

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

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

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

[18] M. Schumacher and J. Bresson, "Spatial Sound Synthesis in Computer-Aided Composition," *Organised Sound*, vol. 15, no. 3, pp. 271 – 289, Dec 2010.

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

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

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

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

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

[24] ——, "Flocks, Herds, and Schools: A Distributed Behavioral Model," *Computer Graphics*, vol. 21, no. 4, pp. 25 – 34, 1987.

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

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

[27] D. Schwarz, "Corpus-Based Concatenative Synthesis," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 92–104, March 2007.