Iain Mott[1] and Thibaud Keller[2]

Three-dimensional sound design with Mosca

Spatialisation tri-dimensionnel avec Mosca

Abstract

Mosca is a software extension class of the SuperCollider language for sound synthesis and algorithmic composition. It produces three-dimensional sound fields via an intuitive graphical user interface controlling a variety of hidden internal methods. Drawing from ambisonics' adaptable form of surround sound applicable to wide-ranging loudspeaker configurations and headphones, Mosca is suitable in a variety of applications. Recent improvements to the software include the incorporation of high-order ambisonics and the OSSIA/score project, enabling sophisticated control of audio spatialisation and synchronisation with a variety of media. Additionally, current development will bring location sensitivity to Mosca. Together with existing head-tracking capacities for binaural audio reproduction, the software will be highly useful in mixed and virtual reality projects.

Keywords: ambisonics, surround sound, electroacoustic music, SuperCollider, OSSIA/score

Résumé

Mosca est une extension pour l'environnement de programmation audio SuperCollider. Cette extension permet la production de champs sonores tridimensionnels à travers une interface graphique intuitive et différentes méthodes internes. Tirant profit de la flexibilité du format ambisonique, compatible avec le rendu au casque et une large gamme de configurations d'enceintes, Mosca peut s'adapter à un grand nombre d'applications. Ce programme est actuellement développé par les auteurs pour intégrer des ordres ambisoniques supérieurs, et davantage de techniques de

1 Sound artist, composer and lecturer in sound design and voice at the Departamento de Artes Cênicas, Instituto de Artes, Universidade de Brasília, http://cen.unb.br. Doctorate in arts from the University of Wollongong entitled "Sound Installation and Self-listening". Personal website: https://escuta.org. Institutional email: iainmott@unb.br.

2 Software technician and system administrator at SCRIME, University of Bordeaux, https://scrime.u--bordeaux.fr. Member of the OSSIA team, https://ossia.io. Masters in computer music programming from the university of Saint-Etienne, https://musinf.univ-st-etienne.fr/index.html. Personal page: https://github.com/thibaudk. Organisation page: https://github.com/scrime-u-bordeaux. Institutional email: thibaud.keller@u-bordeaux.fr.

spatialisation, ainsi que le projet OSSIA/score, permettant un contrôle sophistiqué et la synchronisation, avec différents médias. En outre, les développements actuels apporteront à Mosca une localisation GPS de l'auditeur. Avec les possibilités actuelles de suivi du mouvement de la tête pour le rendu binaural, le programme pourra facilement être utilisé pour des projets de réalité virtuelle et mixte.

Mots clefs: ambisoniques, spatialisation, musique électroacoustique, SuperCollider, OSSIA/score

Introduction

Mosca is a software extension class (or quark) of the SuperCollider language for sound synthesis and algorithmic composition, and was initially developed as part of a research project entitled *Cerrado Ambisônico* by Iain Mott, assisted by the MCTI/CNPq in the Edital Universal. The software produces surround sound and offers an intuitive graphical user interface (GUI) for direct control. A variety of ambisonic and spatial audio techniques are used to render three dimensional sound designs on both headphones and wide-ranging loudspeaker configurations. Mosca makes extensive use of the the Ambisonic Toolkit (ATK) code library for ambisonic processing and the Automation quark to sequence control data. Recent work by Thibaud Keller has brought improvements to the GUI and seen the inclusion of additional spatial audio libraries. Furthermore, communication with the multimedia sequencer OSSIA/score is now enabled, facilitating live performances and synchronisation with other real-time multimedia systems. Current work also includes location sensitivity via GPS and accelerometers, derived from Mott's sound mapping installation *Botanica*, presented in the proceedings of #16.Art.

As an extension of SuperCollider, Mosca is open source and runs on Linux, Mac and Windows platforms. This article describes both current capabilities of Mosca and the work in progress. Along with the documentation for the Mosca quark in SuperCollider, this article serves as a practical guide to the software and provides the reader with information to realise their own projects.

Ambisonic sound

Sound is not a static object. It is liberated from a source and propagated through space by way of compression and rarefaction of air molecules. On a mild day of 20 degrees centigrade, it moves in waves through the air at the speed of approximately 343 metres per second. Various sounds arrive at our ears from all directions, either directly or after having first come into contact with surrounding objects and materials. Ambisonics aims to both capture or synthesise and reproduce such multidirectional sound fields for a certain region in space. Developed by Michael

Gerzon and others in the UK and the USA in the early 1970s [3], ambisonics involves encoding and decoding steps. To encode ambisonics is to process a sound source or sources, for example a solo musical instrument or a flock of birds, by capturing the waveform's inherent phase and level information over a number of channels and along specific spatial axes. Encoding may be done synthetically, by way of digital or analogue processing, or acoustically, using a so-called soundfield[4] microphone (Fellgett, 1975) first developed by Michael Gerzon and Peter Craven in the 1970s (Batke, 2009) with a number of capsules to capture the incoming sound from multiple angles at once. As a convention, sound fields are generally encoded in the multichannel B-format, which may exist at different spatial resolutions, or ambisonic orders (Hollerweger, 2008) and is independent of the loudspeaker configuration used for reproduction. The rendering of an ambisonic sound field is performed in the decoding step, where the signal is processed for a specific loudspeaker array, whether it be a 2-dimensional arrangement of loudspeakers surrounding an audience, or a three-dimensional array of loudspeakers and even headphones, delivering a full spherical sonic experience.

Sound Sources

Mosca takes a flexible approach to encoding and decoding. Sound sources may be any combination of mono, stereo or B-format material and the signals may originate from file (loaded into memory or streamed from disc), from hardware inputs (physical or from other applications like a DAW via the Jack Audio Connection Kit (Jack)[5]) or from sound synthesis processes inside SuperCollider itself (McCartney, 1996; Wilson, Cottle, & Collins, 2011). A particular source is first selected by right-clicking in a blank section in the GUI (Figure 1), then selecting its index from the drop-down menu, with the total number of sources defined when initialising Mosca. Once a source is selected, it can be assigned an input and the defined sources appear as numbered circles in the GUI. The centre of the larger blue circle defines the location of the listener or audience and its periphery marks the maximum audible distance from the listening point. Sources may be positioned and moved in space by clicking and dragging.

---

3 For information on the origins and nature of ambisonics and extensive lists of early publications on the subject, see: www.michaelgerzonphotos.org.uk. Ambisonics is under ongoing development and many of the leading researchers participate in the email discussion group "Sursound": https://mail.music.vt.edu/mailman/listinfo/sursound. The website http://www.ambisonic.n*et al*soprovides resources and the Wikipedia page on ambisonics provides an excellent technical introduction: https://en.wikipedia.org/wiki/Ambisonics.

4 For information about encoding conventions see: https://www.ambisonic.net/fileformats.html

5 http://jackaudio.org

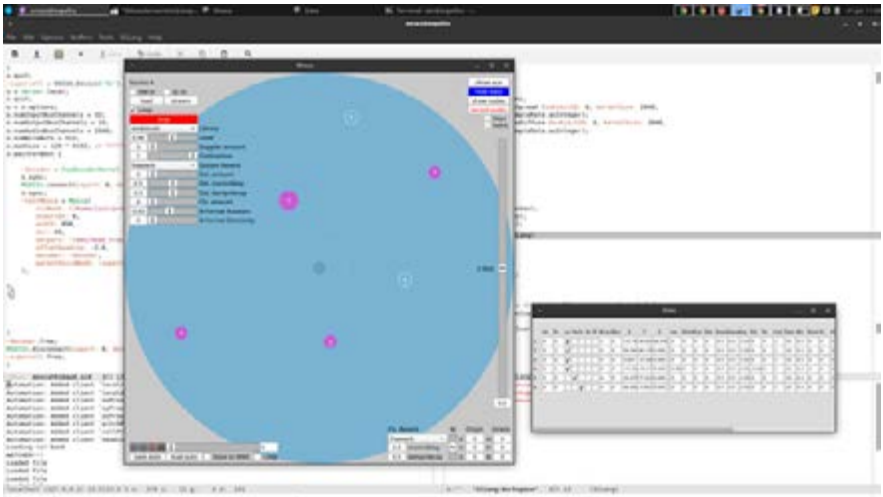

Figure 1. Typical Mosca work session

In order to spatialise the sound sources, mono and stereo inputs are processed on a per-source basis by a selection of algorithm set from the "Library" pull-down menu in the GUI. Users may choose between the ATK[6], HOALib[7] (Guillot, Paris, & Deneu, n.d.), Ambitools (Grond & Lecomte, 2017)[8], ADT(Heller & Benjamin, 2014)[9], BF-FMH, Josh and VBAP libraries[10]. The ATK and Josh are both restricted to 1st order ambisonics, BF-FMH can either encode in 1st or 2nd order whereas Ambitools, HOALib and ADTB offer up to 5th order ambisonics. The default ambisonic order of Mosca is 1, however the user may enter a maximum order as an instantiation argument to perform higher order encoding and decoding. Josh is a simple *ambisonic granulator* effect. VBAP on the other hand is a non-ambisonic method of sound spatialisation unaffected by the "maxorder" argument. The acronym stands for *vector based amplitude panning* (Pulkki, 1997) and involves the panning of sound sources between adjacent loudspeakers in an array, the details of which—angular

6 http://www.ambisonictoolkit.net

7 http://hoalibrary.mshparisnord.fr/en

8 https://github.com/sekisushai/ambitools

9 https://bitbucket.org/ambidecodertoolbox/adt

10 The last three libraries are provided by the official supercollider plugin repository

location and distance—must be passed to Mosca in the creation arguments for the Mosca instance. Regardless of the chosen technique for a particular source, stereo signals are treated as two distinct mono signals, separated spatially by an adjustable angle parameter.

B-format signals up to 5th order, when used as input to Mosca, have no need to undergo ambisonic encoding. The signal can be oriented and transformed directly in various ways. The user can load or stream existing archival B-format material or create their own audio files, either by using Mosca itself to record B-format files from manipulated sources (mono, stereo or B-format) or record their own material acoustically with a sound field microphone. Most commonly available soundfield microphones are so-called A-format devices. Recorded signals using these microphones will therefore need to be processed into B-format for use. The original Mosca project has made use of such a microphone, the Core Sound Tetramic along with a TASCAM DR-680 digital recorder and B-format material has been generated from raw recordings using the Linux software Tetraproc (Adriaensen, 2007). The methodology used and a sample processing script are available online.[11]

Mosca uses two techniques for positioning B-format sources in space. Either the *push* transformation from the ATK library or the *Beam formation* techniques available in Ambitools (Lecomte, Gauthier, Langrenne, Berry, & Garcia, 2016). Both effects diminish as the source approaches the centre point, surrounding the listener and allowing the inherent ambisonic coding of the original signal to dominate. With the ATK, this signal can also be gradually striped of its directional attributes with the "Directivity" control. Fully rolled back, this parameter renders B-format inputs as omnidirectional signals with constituent sounds surrounding the listening point equally from all directions. Additionally, rotation of incoming ambisonic sound fields around their Z-axis can be performed with the "Rotation" parameter.

Mono and stereo sources are fully *contracted* by default, setting a focused position in space. When fully de-contracted with the *contraction* control in the GUI, the signal becomes omnidirectional. B-format sources, on the other hand, are de-contracted by default and contraction causes them to become spatially focussed. The parameter thus offers continuous control over the source's *width*, from a narrow point to an enveloping mass. When mono or stereo signals are de-contracted, the *spread* and *diffuse* GUI options of the ATK library offer two different types of spectral smearing over the spherical ambisonic image. Both have distinct rendering qualities and users may find that one suits a specific class of sounds better than the other, as is the case with all available libraries.

11 http://escuta.org/en/proj/research/ambiresources/item/222-shell-script.html

Distance and reverberation

Along with angular location, Mosca's source signals are attenuated proportionally to their distance from the origin, inside an audible radius of one-hundred metres. In this way, as sources move from the centre of the GUI towards the periphery, they become quieter before being silenced once outside the large blue circle. All sources are subject to high-frequency attenuation with distance. Additionally, a *proximity effect*[12] is available via the ATK as well as a *near field compensation*[13] through Ambitools.

Reverberation plays an important psychoacoustic role in the perceived distance of a sounding object. Typically, as a sound approaches, the ratio of direct to reflected sound—for example from walls, floor and ceiling—increases. In other words, distant sounds generally appear more reverberant than close sounds. Mosca provides two reverberation controls, close and distant, each with a selectable reverberation types. Along with the Freeverb[14] and a plain all-pass option, convolution reverberation can be used to accurately simulate the resonance of natural environments. The Mosca project has created B-format *tail* room impulse responses (RIR) to that effect using Linux based software written by Fons Adriaensen (2007). The methods used have been documented online. [15] Banks of such RIRs may be passed to Mosca as initiation arguments.

The implementation of convolution reverberation, under the close control, can be described as a 2nd order diffuse A-format reverberation. This technique produces reverberation weighted in the direction of sound events encoded in the dry ambisonic signal and involves conversion to and from A-format in order to apply the effect. The encoded 2nd order ambisonic signal is converted to a 12-channel A-format signal and convolved with a B-format RIR which has been *upsampled* to 2nd order and converted to A-format impulse spectrum, a process that is performed automatically on first initialisation. A final step converts this reverberated A-format signal back to 2nd order B-format for decoding and audition (Anderson, 2011).

Presently in Mosca, the *distant* reverberation control attenuates a type of reverberation described by John Chowning as *local*, whereby the return signal from the reverberation process[16] is mixed back with the source signal before the spatialisation phase, producing a tightly spatially focused reverberant signal at the same angular location as the source. Conversely in our implementation, the *close* reverberation control attenuates what

12 http://doc.sccode.org/Classes/FoaProximity.html

13 http://www.sekisushai.net/ambitools/hoa_encoder

14 https://ccrma.stanford.edu/~jos/pasp/Freeverb.html

15 http://escuta.org/en/proj/research/ambiresources/item/222-shell-script.html

16 In the case of Mosca, fed by the w component of the source's B-format encoded signal.

Chowning described as global reverberation, a type of reverberation that envelopes the listener, albeit, in our implementation, with a source-directed weighting. The overall effect is that as the source recedes from the listener, and as it moves from close to *distant* control, the width of the reverberant field narrows (Chowning, 1977). In the future, Mosca will also offer the option of applying A-format convolution reverberation under distant as well as close control.

Movement, control and automation

Mosca is well disposed to create sound fields with moving sound sources and its name was inspired in part by ambisonic recordings of flies—*moscas* in Portuguese—made inadvertently during field recordings in the national park of Chapada dos Veadeiros [17]. It is also a reference to Trevor Wishart's recordings of bluebottle flies for his electroacoustic composition Red Bird. After attempts of recording a fly affixed to a substrate, he discovered that only with a fly in motion—achieved by waving a bluebottle attached to a stick in a stereo field—"could the aural image 'fly' be recreated" (Wishart, 1996, p. 151). To reproduce this characteristic of moving sound, a scalable Doppler effect is implemented in Mosca for each source.

Creating movements and animating parameters in Mosca can be done in a number of ways. A quick and intuitive method relies on the *Automation* quark [18] and direct interaction with the GUI. If users select *record* in the automation transport without selecting play, any changes made with the interface—for instance loading particular audio files, positioning sources, adjusting various level controls—are recorded as a single register in memory and may be saved and later recalled as such, in a named directory. If play is selected while recording, users may record the spatial movement of sources and changes to the controls, as a continuous stream of data. The transport may be rewound any number of times to overlay additional changes or to override prior alterations of the controls. Again, the recorded data may be saved and later reloaded from a named directory for playback. The transport may be synchronised to a digital audio workstation (DAW) using Midi Machine Control (MMC) messages by selecting the *slave to MCC* checkbox in the GUI [19]. This allows Mosca to spatialise multitrack audio compositions, the individual tracks being sent to Mosca via Jack to individual sources with *HW-in* selected[20].

17 B-format audio recordings from this field work are available online on an interactive map: https://escuta.org/mapa

18 https://github.com/neeels/Automation

19 This method has been tested to work with the open source DAW Ardour on Linux.

20 When selected in the GUI, the user must enter the number of channels and the starting bus number.

Mosca can run without an active GUI by reading stored Automation data[21] and may also be controlled entirely through coded commands by way of proxies for the GUI elements. Calling "myMoscaInstanceName.xboxProxy[i].value", for example, will return the current x-coordinate of the source at index "i". With the "valueAction" method, new values can be set in real time for any given parameter proxy. A full list of available proxies is given in the help file for Mosca.

A major focus of Mosca's rework was to enable communication with the *intermedia sequencer* OSSIA/score[22] (Celerier, 2018). This open source project can be described as a graphical language to formulate software interactions in time. The approach expands on a traditional sequencer interface with the addition of flexible durations, parallel timelines, conditional branches and a host of communication protocols. Developed as a virtual conductor, OSSIA/score allows the creation of scenarios where events and processes obey "when", "while" and "switch case" statements represented as *Triggers*, *Loops* and *Conditions*. Open Sound Control (OSC) (Schmeder, Freed, & Wessel, 2010) and OSCQuery[23] compliant applications like Mosca can then be remotely controlled and synchronised together inside a single scenario. Exposed on the network with the OSSIA quark[24] for supercollider, all of Mosca's parameters appear within the *device explorer* of OSSIA/score. This new feature not only greatly improves the integration with other systems, it also provides high level functionalities for editing large-scale spatial audio compositions and *real time* interactive projects of all kinds.

Figure 2. OSSIA/score scenario example

21 Using initialisation arguments.

22 ossia.io

23 https://github.com/Vidvox/OSCQueryProposal

24 https://github.com/OSSIA/ossia-sclang

Decoding

A variety of options are available for decoding, each dependent on the rendering system used (binaural, 2D ring or 3D array). A SuperCollider decoder object, chosen from a number of types, may be passed to the Mosca instance in an initialisation argument. If none is provided, raw ambisonic components of the chosen order are outputted for processing with an external decoder, such as Ambdec (Adriaensen, 2011). The AmbiDecoderToolbox library for Octave and Matlab (Heller & Benjamin, 2014) may also be used and is recommended for higher-order decoding. It enables users to create custom decoders calibrated specifically for their setup. The scripting and compiling steps for ADT are described in the SC-HOA quark tutorials (Grond & Lecomte, 2017). As mentioned, VBAP involves neither ambisonic encoding nor decoding, however Mosca does require VBAP to be initialised with an array of loudspeaker coordinates (2D or 3D) passed as an initialisation argument.

When a binaural ambisonic decoder is used for headphone reproduction, it may be advantageous to use a head-tracking device. In this way the sound field rotates correspondingly with rotations of the head and the listener may better audition each source. Mosca provides support for the Arduino 9-Axes Motion Shield[25] sensor with the Arduino Uno[26] microprocessor. The Mosca help file provides information on how the head-tracker should be installed and configured. Movements of the head are displayed in the GUI as values of *heading, pitch* and *roll* as well as corresponding rotations of sound sources. These *orientation* controls, like the *origin* parameters, only affect the relative spherical coordinates for every source, leaving the absolute Cartesian coordinates intact. Both types of coordinates are available in OSSIA/score when exposed to the network.

Current work

Work is currently underway to bring location sensitivity to Mosca using SuperCollider code developed in the sound art project *Botanica* (Mott, 2017). The code will enable Mosca to import a map and then calibrate it to position a mobile listener in accordance with their physical location. Longitude and latitude serial data may be used as input, be it from GPS or from any location sensing device. The system developed uses a Ublox NEO-6M GPS module[27] which is wired directly to the Arduino Uno with 9-Axes Motion Shield. Coupled with head tracking, listeners will be able to walk in a direction of their choice, or travel by some other means, to explore soundscapes spanning physical space. Together with existing head-tracking capacities for

25 https://store.arduino.cc/usa/9-axis-motion-shield

26 https://store.arduino.cc/usa/arduino-uno-rev3

27 https://lastminuteengineers.com/neo6m-gps-arduino-tutorial/

binaural audio reproduction, location sensitivity will lend Mosca broad application in mixed and virtual reality projects. The OSSIA quark is also under development to take over from the now deprecated ossia-supercollider[28] project, currently the only way to enable OSCQuery communication. It will soon be reconfigured and based on Pierre Cochard's wsclang[29] fork of SuperCollider and the changes then submitted to the official SuperCollider repository.

Resources

The home page for Mosca is http://escuta.org/mosca. To use Mosca, SuperCollider must be installed on a Linux, Mac or Windows computer along with SuperCollider's assortment of plugins, including HOA[30] and the ATK[31]. Like SuperCollider and the ATK, the Mosca source code and accompanying Arduino source for head-tracking, are available on the Github site.[32] As well as including the SuperCollider source code for Mosca and help files, the repository, as mentioned above, contains the source code for the Arduino-based head-tracker and should be loaded onto the device using the Arduino IDE[33] software. The Mosca quark and additional prerequisite quarks including the ATK may be loaded via the Quarks.gui interface in SuperCollider. Once loaded, the user may access the Mosca help file and guide with full instructions and code examples, by running the help command on the class name Mosca. Additionally, the moscaproject.zip file contained within the git source may be used. Once extracted, the archive contains the basic file structure for a Mosca project as well as an example RIR file. A much larger project directory with B-format audio files is also available on the site Escuta.org.[34]

Acknowledgements

Participation in this event was made possible with assistance from the Fundação de Apoio a Pesquisa do Distrito Federal (FAP DF). The authors gratefully acknowledge Joseph Anderson (ATK), Pierre Lecomte (Ambitools), Florian Grond (SC-HOA), Pierre

28 https://github.com/OSSIA/ossia-supercollider

29 https://github.com/pchdev/wsclang

30 https://github.com/scrime-u-bordeaux/sc3-pluginsHOA

31 http://www.ambisonictoolkit.net/download/supercollider

32 https://github.com/escuta/mosca

33 https://www.arduino.cc

34 http://escuta.org/tmp/moscaproject.zip. The archive includes a B-format audio recorded by Iain Mott in Chapada dos Veadeiros and Brasilia as well as a Spitfire recording by John Leonard, provided with kind permission.

Guillot (HoaLib), Neels Hofmeyr (Automation), Jean-Michël Celerier (OSSIA/score), Pierre Cochard (ossia-supercollider & wsclang) and members of the SuperCollider users and dev lists for their assistance and valuable suggestions.

References

Adriaensen, F. (2007). **A Tetrahedral Microphone Processor for Ambisonic Recording.** Presented at the Linux Audio Conference, Berlin.

Adriaensen, F. (2011). **Ambdec 0.4.2 User Manual.** Retrieved from http://kokkinizita.linuxaudio.org/linuxaudio/downloads/ambdec-manual.pdf

Anderson, J. (2011). **Authoring complex Ambisonic soundfields: An artist's tips & tricks.** Presented at the Digital Hybridity and Sounds in Space Joint Symposium, University of Derby, UK. Retrieved from https://www.researchgate.net/publication/273944382_Authoring_complex_Ambisonic_soundfields_An_artist%27s_tips_tricks

Batke, J.-M. (2009). The B-Format Microphone Revised. **Ambisonics Symposium**, 5.

Celerier, J.-M. (2018). Authoring interactive media: **A logical & temporal approach** (PhD thesis). University of Bordeaux, France.

Chowning, J. M. (1977). The Simulation of Moving Sound Sources. **Computer Music Journal**, 1(3), 48–52.

Fellgett, P. (1975). Ambisonics. Part one: General system description. **Studio Sound**, 17(8), 20–22, 40.

Grond, F., & Lecomte, P. (2017). **Higher Order Ambisonics for SuperCollider.** Presented at the Linux Audio Conference.

Guillot, P., Paris, E., & Deneu, M. (n.d.). La bibliothèque de spatialisation HOA pour Max/MSP, Pure Data, VST, FAUST. **Revue Francophone d'Informatique et Musique.** Retrieved from https://revues.mshparisnord.fr:443/rfim/index.php?id=245

Heller, A. J., & Benjamin, E. M. (2014). **The Ambisonic Decoder Toolbox: Extensions for Partial-Coverage Loudspeaker Arrays.** Presented at the Linux Audio Conference, Karlsruhe, Germany.

Hollerweger, F. (2008). **An Introduction to Higher Order Ambisonics.** Retrieved from http://flo.mur.at/writings/HOA-intro.pdf

Lecomte, P., Gauthier, P.-A., Langrenne, C., Berry, A., & Garcia, A. (2016). Filtrage directionnel dans un scène sonore 3D par une utilisation conjointe de Beamforming et

d'Ambisonie d'ordre élevé. **13ème Congrès Français d'Acoustique Joint Avec Le 20ème Colloque VIbrations, SHocks and NOise, CFA/VISHNO**. Presented at the Le Mans, Sarthe, France. Le Mans, Sarthe, France.

McCartney, J. (1996). SuperCollider: A New Real Time Synthesis Language. **International Computer Music Conference**, 257–258. Hong Kong: International Computer Music Association.

Mott, I. (2017). **Botanica: Navigable, immersive sound art.** Presented at the 16o Encontro Internacional de Arte e Tecnologia, Porto, Portugal.

Pulkki, V. (1997). Virtual Sound Source Positioning Using Vector Base Amplitude Panning. **Journal of the Audio Engineering Society**, 45(6).

Schmeder, A., Freed, A., & Wessel, D. (2010). **Best Practices for Open Sound Contro**l. Presented at the Linux Audio Conference.

Wilson, S., Cottle, D., & Collins, N. (Eds.). (2011). **The SuperCollider Book**. Cambridge, Massachusetts & London: MIT Press.

Wishart, T. (1996). **On Sonic Art** (2nd ed.; S. Emmerson, Ed.). Amsterdam: Harwood Academic Publishers.