SOUND ELEMENT SPATIALIZER

Ryan McGee, Matthew Wright

Media Arts and Technology
University of California, Santa Barbara
ryan@mat.ucsb.edu
matt@create.ucsb.edu

ABSTRACT

Sound Element Spatializer (SES) is a novel system for the rendering and control of spatial audio. SES provides simple access to spatial rendering techniques including VBAP, DBAP, Ambisonics, and WFS via OSC. Arbitrary loudspeaker configurations and an arbitrary number of moving sound sources are supported. SES operates as a cross-platform C++ application that can spatialize sound sources from other applications or live inputs in real-time. Also included are means for decorrelated upmixing.

1. INTRODUCTION

Computer music composers have carefully crafted the spatial dimension of sound for decades [2]. It can contrast and animate sounds in space in a way comparable to the use of pitch in the frequency domain. Spatialization can clarify dense textures of sounds, make greater numbers of simultaneous sound elements perceivable, and choreograph complex sonic trajectories [13]. For effective spatialization one must be able to precisely control the trajectories of spatial sound elements, i.e., their spatial positions as functions of time. Dramatic effects can be achieved when the movement of sounds is very fast- the "whoosh" of a high-speed vehicle driving by or the sounds of several insects flying around one's head, for example. With computers we can simulate moving sound sources beyond the scope of reality. For instance, one may simulate a violinist flying around a room at 100mph or decompose a sound into constituent elements to be individually spatialized.

The goal of SES was to operate as a robust, cross-platform, standalone application that would be easy to integrate with any audio application and scale to support spatialization of an arbitrary number of sound sources over an arbitrary loudspeaker configuration, with dynamic selection among spatialization rendering techniques including Vector Base Amplitude Panning (VBAP) [10], Distance Based Amplitude Panning (DBAP) [5], Higher Order Ambisonics (HOA) [6], and Wave Field Synthesis [1]. Control of sound trajectories should not limited to a proprietary GUI or to programming in a given environment, but instead accomplished via Open Sound Control (OSC) [16, 17].

1.1. Digital Audio Workstation Software

Popular digital audio workstation (DAW) software packages such as Logic, ProTools, Live, Digital Performer, etc., though useful for many aspects of layering and shaping sound in time, offer impoverished spatialization functionality. While DAWs sometimes include spatial panning interfaces or plug-ins, these panning methods are limited to sounds in a 2-dimensional plane and are not scalable to accommodate loudspeaker configurations beyond consumer formats such as stereo, quadraphonic, 5.1, 7.1, and 10.2 [8]. DAW packages may include integrated automation editors for the time-dependent control of spatialization parameters, but the automation of spatialization becomes cumbersome when implementing complex geometric trajectories for several sound sources or wanting to specify intricate trajectories procedurally.

1.2. Related Work

Outside the world of DAWs, there are several spatial audio platforms including BEASTMulch [15], Zirkonium [11], Jamoma [8], Spatialisateur [3], and OMPrisma [14]. While these systems do greatly extend the compositional capabilities for spatialization well beyond those of DAWs, all currently lack in at least one area of usability, scalability, flexibility, or control, as shown in Table 1.

Features of Current Spatialization Systems	Arbitrary Speaker Configurations	Arbitrary Number of Sound Sources	Distance Cue and Doppler Shift	Trajectory Control via OSC	Multiple Panning Algorithms	WFS	Cross-platform Standalone App.
DAWs							✓
BEASTmulch	✓	✓			✓		
Zirkonium	✓			✓			
Jamoma	✓	✓	✓	✓	✓		
Spatialisateur	✓	✓	✓		✓		
OMPrisma	✓	✓	✓		✓		
SES	✓	✓	✓	✓	✓	✓	✓

Table 1. Comparison of Current Spatialization Systems

2. SOUND ELEMENT SPATIALIZER

SES positions an arbitrary number of sound elements in real time over an arbitrary speaker arrangement via VBAP, DBAP, HOA, or WFS with distance cues including Doppler shift, air absorption, and gain attenuation. Smooth position interpolation supports movement of sound sources up to the speed of sound. SES is a cross-platform standalone GUI application for easy integration with any audio application and no additional programming, with all control exposed via OSC. Novel features include "derived sources" for decorrelated upmixing and a built-in optional "Flutter" effect to increase spatial localization. Figure 1 provides an overview of the entire software.

2.1. Spatial Upmixing Techniques

SES's novel "derived sources" feature creates spatializable sound elements beyond the number of input channels, to support decorrelated upmixing (a.k.a. spatial decorrelation) [4]. The simplest method, *duplication*, copies a given audio input to any number of individually spatializable sources; this obviously does not provide decorrelation but can produce interesting chorusing effects when many copies of the same sound move with Doppler. The *spectrum bands* method divides an input via Butterworth band-pass filters of varying center frequency. *Pitch shifting* an input produces additional elements transposed by varying pitch intervals, and *delays* produce derived sources following an input with settable time lags.

2.2. Flutter

Another novel feature in SES is the ability to instantly "flutter" any element, which can increase spatial perception. Flutter simply applies an individual low frequency oscillator (LFO) to the amplitude of each sound element. The rate of the flutter LFO is adjusted by the user in the range of 1hz to 30hz, with 12 to 16hz typically being the most effective. The user can also select the shape of the flutter waveform: sine, square, sawtooth, or triangle. With large swarms or clusters of several simultaneous moving sound sources, flutter is extremely effective at localizing a particular source within the mix.

2.3. Distributed Processing

SES is designed to run on a single system or distribute processing amongst multiple systems. While SES handles spatialization rendering and spatial upmixing, the source sounds and OSC trajectories must be produced elsewhere. We often run a separate sound source application (DAW, granulator[1], etc.) on the same computer and connect its audio outputs to SES's audio inputs using a virtual audio device such as SoundFlower[2] or Jack[3]. We can also simultaneously run an OSC trajectory control application on the same machine. Using a hardware audio device SES may read source sounds from live sources, a multichannel recording, or another computer. OSC control applications can send trajectory data to SES from another computer over a network. An example processing scheme for running SES on a single machine is shown in Figure 2

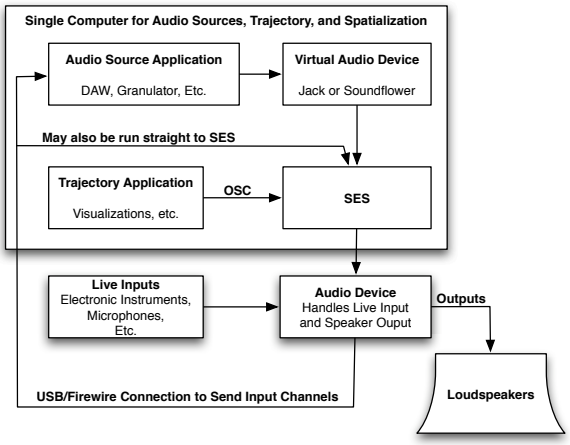

Figure 2. SES Running on a Single Machine

2.4. Trajectory Control

The use of OSC in SES allows for an extremely broad range of software and hardware controllers. For example, each sound element may map to the position of a node in a visual flocking algorithm, or receive control messages from a mobile touch-screen device. SES users may implement custom trajectory controllers in any application that can send OSC, or use one of the provided trajectory editors. We envision an ever-growing collection of trajectory controllers, editors, and sequencers shared among the SES user community. In particular, visual artists working with particle systems can easily make their programs into SES trajectory controllers simply by having each visual particle output OSC to control the spatial position of a given audio source. OSC messages can also be used to control parameters for derived sources, to control flutter, to add and remove elements, to mix elements (level, mute, solo, master gain), and to dynamically switch between panning algorithms and speaker layouts. SES reads trajectory control messages in polar or spherical coordinates according to the SpatDIF [9] OSC format.

3. GRAPHICAL USER INTERFACE

The main window (Figure 3) shows a table of all sound elements with mute, solo, level, and flutter controls. The

[1] Granular synthesis [12] programs already allow each grain to have its own spatial position. However, most implementations are limited to stereo sound, with the precision of positioning often limited to stochastic methods [7]. When coupled with a multichannel granulation program, SES allows for the precise positioning of grains or clusters of grains in 3D space according to any process that outputs OSC messages.

[2] http://cycling74.com/products/soundower/

[3] http://www.jackosx.com/

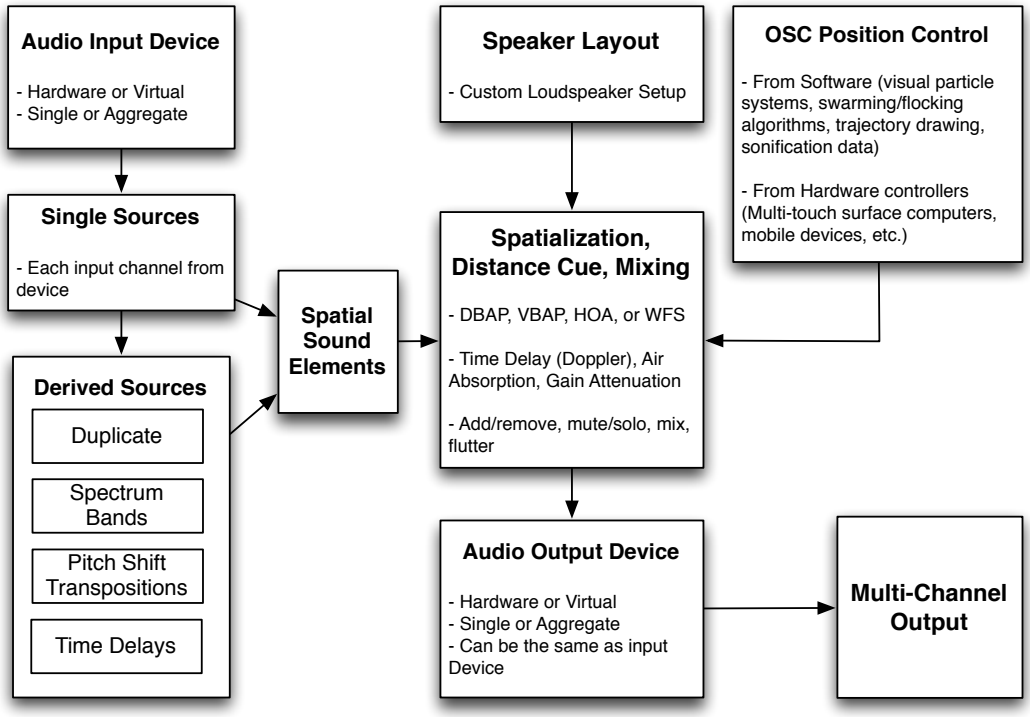

Figure 1. Sound Element Spatializer Overview

"auto map" button maps each input on the connected audio device to a separate element, e.g., to quickly spatialize individual tracks from a DAW. There are buttons to bring up the program's other windows alongside a drop down menu to dynamically select the panning algorithm. Load and save buttons provide means to store and retrieve complex lists of elements, and there is a CPU meter and master gain slider. The "add elements" window (Figure 4) allows users to map an audio input channel to a single element or to many derived sources as described in Section 2.1, as well as to name the new element(s). Finally, the simple speaker layout editor allows the user to specify azimuth, elevation, and radial position for any number of speakers and to save and load these layouts as simple text files.

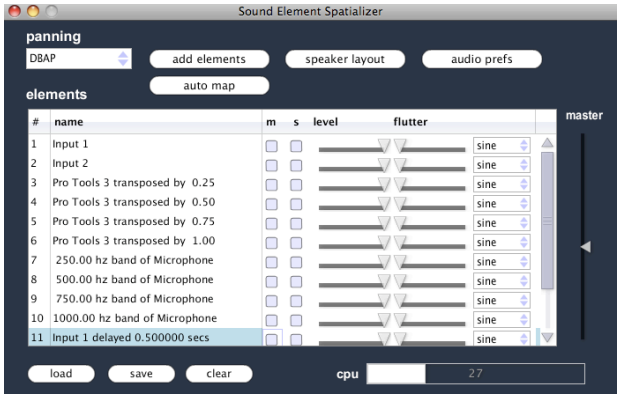

Figure 3. SES Main Window

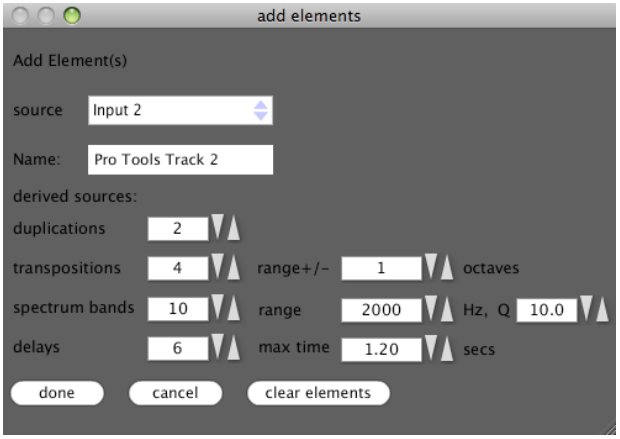

Figure 4. SES Add Elements Window

4. IMPLEMENTATION

SES was written in C++ using the CREATE Signal Library[4] (CSL) for high-level audio processing, WFSynth[5] for real-time WFS, JUCE (Jules Utility Class Extensions)[6] for the graphical user interface and low-level audio processing, the SoundTouch library[7] for real-time pitch shifting, and the Lightweight OSC library (liblo)[8] to receive OSC messages.

[4] http://fastlabinc.com/CSL/
[5] http://wfsynth.sourceforge.net/
[6] http://www.rawmaterialsoftware.com/juce.php
[7] http://www.surina.net/soundtouch/
[8] http://liblo.sourceforge.net/

5. CONCLUSIONS AND FUTURE WORK

SES provides means for flexible, precise control of spatialization for an arbitrary number of sound sources over an arbitrary speaker layout. Much work has been done to remove common limitations found in other similar systems. Expert users are free to program OSC trajectory controllers for SES in the environment of their choice, while there also exists a need for the development of generic OSC trajectory controllers and sequencers for the average DAW user wishing to implement advanced spatialization. Linking the spatial practices involved with data visualization to sound spatialization may be a good first step to realizing and controlling complex sonic trajectories. Future work on SES includes improved interface look-and-feel, multi-core/processor support, and additional means of "derived sources."

6. ACKNOWLEDGEMENTS

We would like to thank Stephen Travis Pope, Jorge Castellanos, Graham Wakefield, and Lance Putnam for their contributions to CSL and technical assistance. We thank Will Wolcott for his wonderful WFS API. We are also grateful to Curtis Roads and JoAnn Kuchera-Morin for their inspiration and guidance throughout the project.

7. REFERENCES

[1] A. J. Berkhout, D. de Vries, and P. Vogel, "Acoustic control by wave field synthesis," *The Journal of the Acoustical Society of America*, vol. 93, no. 5, pp. 2764–2778, 1993.

[2] J. Chowning, "The simulation of moving sound sources," *Computer Music Journal*, no. 3, pp. 48–52, 1977.

[3] J.-M. Jot and O. Warusfel, "A Real-Time Spatial Sound Processor for Music and Virtual Reality Applications," in *Proceedings of International Computer Music Conference*, 1995, pp. 294–295.

[4] G. S. Kendall, "The Decorrelation of Audio Signals and Its Impact on Spatial Imagery," *Computer Music Journal*, vol. 19, no. 4, pp. 71–87, Jan. 1995.

[5] T. Lossius, P. Baltazar, and T. de La Hogue, "DBAP-Distance-Based Amplitude Panning," in *Proceedings of 2009 International Computer Music Conference, Montreal, Canada*, no. 1, 2009.

[6] D. G. Malham and A. Myatt, "3-D Sound Spatialization using Ambisonic Techniques," *Computer Music Journal*, vol. 19, no. 4, pp. 58–70, 1995.

[7] A. McLeran, C. Roads, B. Sturm, and J. Shynk, "Granular sound spatialization using dictionary-based methods," in *Proceedings of the 5th Sound and Music Computing Conference, Berlin, Germany*, no. 1, 2008. [Online]. Available: http://smc.afim-asso.org/smc08/images/proceedings/session2_number4_paper20.pdf

[8] N. Peters, T. Lossiusb, J. Schacherc, P. Baltazard, C. Bascoue, and T. Placef, "A stratified approach for sound spatialization," in *Proceedings of 6th Sound and Music Computing Conference*, 2009. [Online]. Available: http://spatdif.org/papers/Spatialization-SMC09.pdf

[9] N. Peters, S. Ferguson, and S. McAdams, "Towards a Spatial Sound Description Interchange Format (SpatDIF)," *Canadian Acoustics*, vol. 35, no. 3, pp. 64–65, 2007. [Online]. Available: http://www.music.mcgill.ca/~nils/papers/2007_CAA.pdf

[10] V. Pulkki, "Virtual sound source positioning using vector base amplitude panning," *Journal of the Audio Engineering Society*, vol. 45, no. 6, pp. 456–466, 1997. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=7853

[11] C. Ramakrishnan, "Zirkonium." [Online]. Available: http://ima.zkm.de/~cramakri/zirkonium/ZirkoniumManual.pdf

[12] C. Roads, *Microsound*. MIT Press, 2001.

[13] ——, "Articulating space," January 2011, draft of chapter from forthcoming book.

[14] M. Schumacher and J. Bresson, "Spatial Sound Synthesis in Computer-Aided Composition," *Organised Sound*, vol. 15, no. 03, pp. 271–289, Oct. 2010.

[15] S. Wilson and J. Harrison, "Rethinking the BEAST: Recent developments in multichannel composition at Birmingham ElectroAcoustic Sound Theatre," *Organised Sound*, vol. 15, no. 03, pp. 239–250, Oct. 2010. [Online]. Available: http://www.journals.cambridge.org/abstract_S1355771810000312

[16] M. Wright and A. Freed, "Open Sound Control: A New Protocol for Communicating with Sound Synthesizers," in *Proceedings of International Computer Music Conference*, 1997, pp. 101–104.

[17] M. Wright, A. Freed, A. Lee, T. Madden, and A. Momeni, "Managing Complexity with Explicit Mapping of Gestures to Sound Control with OSC," in *Proceedings of International Computer Music Conference*, 2001, pp. 314–317.