UNIVERSITY OF CALIFORNIA

Santa Barbara

Sound Element Spatializer

A project report submitted in partial satisfaction

of the requirements for the degree of

Master of Science

In

Media Arts & Technology

by

Ryan Michael McGee

Committee in charge:

Curtis Roads, Chair

JoAnn Kuchera-Morin

George Legrady

Matthew Wright

December 2010

The project report of Ryan Michael McGee is approved.

_____

Curtis Roads, Committee Chair

_____

JoAnn Kuchera-Morin

_____

George Legrady

_____

Matthew Wright

December 2010

ABSTRACT

Sound Element Spatializer

By

Ryan Michael McGee

Existing sound spatialization paradigms fail to address the needs of many composers and use restraining control methods. This paper presents Sound Element Spatializer (SES), a novel system for the rendering and control of spatial audio. SES provides multiple 3D sound rendering techniques and allows for an arbitrary loudspeaker configuration with an arbitrary number of moving sound sources. Sound sources become "spatial sound elements" possessing individual trajectory information controlled via the Open Sound Control protocol. SES operates as a robust, cross-platform application that can spatialize sound sources from other applications or live inputs in real-time. Also included are novel means for the spatial upmixing of sounds.

CONTENTS

DEFINITION OF TERMS

Digital Audio Workstation (DAW) – popular commercial software packages for multi-track recording, editing, mixing, and mastering of audio. The most common include Pro Tools, Ableton Live, Logic, and Digital Performer.

Sound Element Spatializer (SES) - the spatialization software resulting from this project

Panning Algorithm - any computational method by which sounds are distributed to multiple loudspeakers (Ambisonics or Vector-Based Amplitude Panning, for example)

INTRODUCTION

Spatialization is a dimension of timbre used to contrast and animate sounds in space similar to the use of pitch in the frequency domain. Spatialization can act to clarify dense textures of sounds, perceive greater numbers of simultaneous sound elements, or choreograph complex sonic trajectories [17]. Spatialized sound elements may reside on Macro, Meso, Sound object, or Micro time scales [16], that is, sound elements may consist of entire compositions, organized phrases of sounds, individual sound objects, or sound particles at the threshold of human perception. For effective spatialization one must be able to precisely control the trajectories of spatial sound elements. 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. In fact, the visionary composer, Karlheinz Stockhausen, listed "the decomposition of sound" and "composition of several layers in space" as two of his four criteria for electronic music.

Motivation

Much of the motivation for this project began with my needs as a composer and frustrations with DAW software when creating my piece, *WANTS*, in the winter of 2009. Part of the work called for the spatialization of several voices over an octaphonic loudspeaker arrangement. Using popular DAW software, I discovered that the included surround sound panners did not support anything beyond a 7.1 configuration, so, for me, the need to spatialize sound over an arbitrary speaker layout was born. To realize the piece, I also needed to precisely move several simultaneous sound sources along separate Fibonacci spiral trajectories. At the time I accomplished this through tedious MATLAB scripting that involved generating eight mono sound files for each source. My MATLAB script implemented distance-based amplitude panning [6] with only gain attenuation for distance cue. For more realistic moving sound sources I would need Doppler shift and air absorption filters.

WANTS also called for sound sources moving at speeds over 50 meters per second (100 mph). I was able to accomplish this using my non-real-time MATLAB rendering technique, but I began to wonder if sounds could be moved that quickly in real-time without unwanted artifacts. My following spatialization system prototypes in Pure Data and Max/MSP suffered from a "zipper" noise when moving sounds too quickly. I looked at other existing DAW alternatives for spatialization, but found each of them lacking with regards to usability, scalability, flexibility, or control.

Problem Statement

Many spatialization techniques remain inaccessible to the average computer musician using popular digital audio workstation (DAW) software packages (Logic, ProTools, Live, Digital Performer, etc.). While DAWs do 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, 10.2, etcetera [11]. DAW software packages also lack flexibility with available panning algorithms [11]. In fact, most DAWs only implement exactly one form of spatialization rendering, usually some variation of vector-based amplitude panning (VBAP) [14] or distance-based amplitude panning (DBAP) [6]. Unlike VBAP and DBAP, Higher-Order Ambisonics (HOA) [2] and wave field synthesis (WFS) [20] techniques use soundfield modeling and superposition of wavefronts to provide additional realism and depth to spatialized sounds. Unfortunately, Ambisonics and WFS rendering methods are not present in any DAW.

Most DAW packages do, however, have integrated automation editors for the time dependant control of spatialization parameters such as position and distance. Yet, the automation of spatialization becomes cumbersome when implementing complex geometric (possibly computed or algorithmic) trajectories for sound sources or when manipulating the position of several sound sources simultaneously [8]. Visual artists, on the other hand, are often masters of creating complex spatial trajectories, and their tools (Maya, Processing, etc) are extremely capable trajectory

editors. There is a need to link the spatial practices involved in data visualization with spatialization in computer music to provide a solution for realizing complex sonic trajectories.

Open Sound Control (OSC) is a network protocol for communication among computers and a variety of multimedia devices [24]. To list a few examples, OSC messages can be generated from sound synthesis software, visualization software, hardware controllers, touch-screen interfaces, and mobile devices. OSC provides massive amounts of flexibility and data capacity over the dated MIDI protocol found in all popular DAW software. For maximum flexibility, sound spatialization should be controlled via OSC messages rather than with restricting MIDI based interfaces.

Project Goals

The goals for this project are stated as follows:

- Real-time spatialization of an arbitrary number of simultaneous live or recorded sound sources over an arbitrary loudspeaker arrangement.
- Dynamic selection between multiple panning algorithms with distance cue including Doppler shift, air absorption, and gain attenuation.
- High-speed movement of sound sources (over 50 m/s) without “zipper” noise artifacts.
- Flexible, precise control of sound trajectories using the OSC protocol.
- A standalone application for robustness and ease of compatibility.
- Provide a means for visual artists to use their work to spatialize sound.

RELATED WORK

In response to needs imposed by electroacoustic composers and multi-channel venues, a number of novel spatial audio platforms have already emerged. Of such systems the most extensive include BEASTMulch, Zirkonium, Jamoma, and Spat. While existing systems do greatly extend the compositional capabilities for spatialization well beyond those of DAWs, all lack in at least one area of usability, scalability, flexibility, or control.

Table 1. Features of Current Spatialization Systems

	Arbitrary Speaker Configurations	Arbitrary Number of Sound Sources	Distance Cue and Doppler Shift	OSC Trajectory Control	Multiple Panning Algorithms	Standalone Application	Real-time
DAWs						✓	✓
BEASTmulch [23]	✓	✓			✓	✓	✓
Zirkonium [15]	✓			✓		✓	✓
Jamoma [11]	✓	✓	✓	✓	✓		✓
Spat [5]	✓	✓	✓		✓		✓
ViMiC [10]			✓				✓
HoloSpat/HoloEdit [8]	✓	✓	✓		✓		✓
OMPrisma [18]	✓	✓	✓		✓		✓
SSR [1]	✓		✓		✓	✓	
Granulators [7, 21]		✓				✓	✓
SES	✓	✓	✓	✓	✓	✓	✓

Zirkonium

Zirkonium is a program for sound spatialization developed at the Institute for Music and Acoustics at the ZKM Center for Art and Media in Karlsruhe, Germany [15]. Originally created to control a custom 47-speaker configuration, the program is also designed to work with arbitrary loudspeaker environments. Zirkonium does integrate well with a variety of input sources including DAW output and live instruments, but the user is limited to 16 simultaneous input sources. The application is able to read position information from OSC messages, but only implements a single panning algorithm, VBAP. The program does not implement Doppler shift or an air absorption filter to provide additional distance or motion cue. Nevertheless, Zirkonium is a very useful application and has served as inspiration for this project with its support for flexible speaker configurations and OSC control.

BEASTmulch

The BEASTmulch System is a software tool for the presentation of electroacoustic music over multichannel systems developed at the Electroacoustic Music Studios at the University of Birmingham, United Kingdom [23]. The program is extremely flexible regarding loudspeaker configuration, panning algorithms, and input sources. However, while there are several different versions of 2D and 3D VBAP and Ambisonic panners implemented, there is no option for DBAP. While flexible, the user-interface is a bit cumbersome and may not appeal to the average electronic musician trying to begin with spatialization. There is no direct OSC

control of source positions in BEASTmulch either. It appears one must use another application, MotorBEAST, to read and send OSC control data or make use of an Ethersense hardware controller to send OSC control data based on haptic sensor inputs. I could not find either the MotorBEAST application or any supporting documentation for OSC control online. Nevertheless, BEASTmulch is a comprehensive, highly customizable spatialization environment for experienced computer musicians.

SpatDif

SpatDIF (Spatial Sound Description Interchange Format) is not a spatialization tool, but rather a standardized OSC message format for describing spatial audio trajectories between different spatialization platforms [9]. Currently SpatDIF is implemented in Jamoma, OMPrisma, and SES. Additional implementations will ease the transfer of sonic trajectory information between various spatialization programs and their panning algorithms. An example OSC message in SpatDIF format reads "/SpatDIF/source/3/aed 45.0 -15.0 5.0" meaning that source number 3 has an azimuth of 45 degrees, elevation of -15 degrees, and distance of 5 meters.

Jamoma

Jamoma is a Max/MSP based platform for research in new media art and interactivity consisting of several parallel development efforts [11]. Several

Max/MSP modules for different spatialization techniques (DBAP, Ambisonics, VBAP, ViMiC [10]) have been implemented along with distance cue processing that takes into account Doppler shift, air absorption, and gain attenuation. OSC control is implemented using SpatDIF format. Jamoma may be a solution for expert computer musicians comfortable with graphical programming in the Max/MSP environment.

Spat

Spatialisateur (Spat) is a very thorough Max/MSP spatialization suite developed at IRCAM dating back to 1991 [5]. Spat implements a variety of panning algorithms and is scalable to support an arbitrary number of sources and speakers. In addition to attenuation, time delay, and air absorption, Spat makes use of multi-channel reverb for distance cue, an extension of John Chowning's milestone work in the seventies [3]. Modules in Spat rely on perceptual rather than technical parameters for spatialization. For example, one may specify the brilliance, vivacity, presence, heat, or heaviness of a sound source. Though, unlike Jamoma, Spat does not utilize a simple, standardized OSC trajectory control format.

Problems With Other Spatialization Systems

This project views dependence on Max/MSP as a major drawback for a spatialization framework. Graphical programming in an environment such as Max or OpenMusic should not be a requirement for spatializing audio. Spatialization systems should strive to appeal to the typical DAW user to encourage their usage. Also, systems that rely on their own trajectory editors are seen as lacking. While it is not

expected that typical DAW users should be comfortable programming their own OSC control applications, spatialization systems should utilize OSC to encourage the development of compatible trajectory controllers by expert users or companies.

The Granular Synthesis Model

Granular synthesis can serve as a useful model for sound spatialization. Granulation involves the time-based decomposition of a sound source into individual sound "elements" (also referred to as "grains" or "atoms") [7]. The composer or sound designer can then edit parameters of each element individually with great detail. Some granulation programs already apply an element-based approach to spatialization, and "clouds" of sound are created in part by specifying a unique spatial position for each sound element [7]. However, most implementations are limited to stereo sound, and the precision of positioning is limited to stochastic methods. Though, Scott Wilson's "Spatial Swarm Granulation" is an interesting experiment with BEASTmulch that makes use of a Boid distributed behavior model to organize spatialized sound grains in 3D [22]. When coupled with a multichannel granulation program, SES will allow for the precise positioning of grains or clusters of grains in 3D space according to any process that outputs OSC messages.

DESIGN

Hardware and Software Configuration

Sound Element Spatializer is currently compiled for Max OSX 10.5 or later. Since cross-platform C++ libraries were used for construction, the software can also be compiled and run on Linux or Windows systems. There is no minimum processor speed recommended, but higher clock rates support larger numbers of simultaneous sources and speakers.

Ideally, a spatialization system utilizing SES would distribute the processing over three computers. The first computer would be responsible for running the sound generating applications such as a DAW, granulator, or any other audio application. The second computer would be responsible for spatialization rendering by running SES, and the third computer would send OSC trajectory data to the SES computer over the local network.

Though three computers are ideal, such a configuration is certainly not a requirement to run SES. The sound generation and trajectory computation programs may be run simultaneously on the same machine as SES, but it is best to isolate the most processor intensive task to a separate computer when possible. When using complex visualizations to compute source positions one may often find that the trajectory processing consumes more resources than the sound generation or SES. In that case there should be a dedicated machine to send OSC trajectory messages.

When dealing with pre-recorded sound sources the sound generation processing may be minor and combined with SES or the trajectory computation. It may also be necessary to isolate trajectory computation when using mobile, touch-screen devices to control source positions. Example hardware configurations are shown in following diagrams.

Figure 1. Distributed Processing with SES Over Three Computers

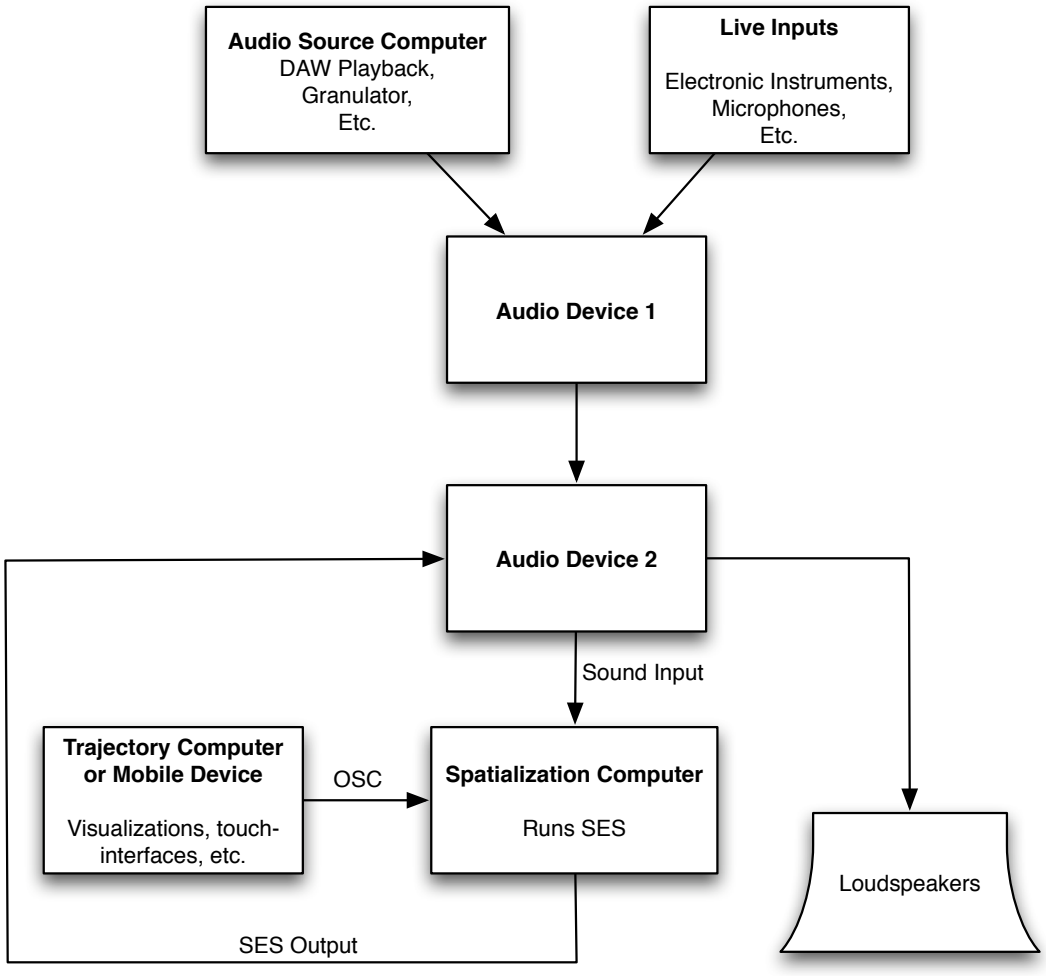
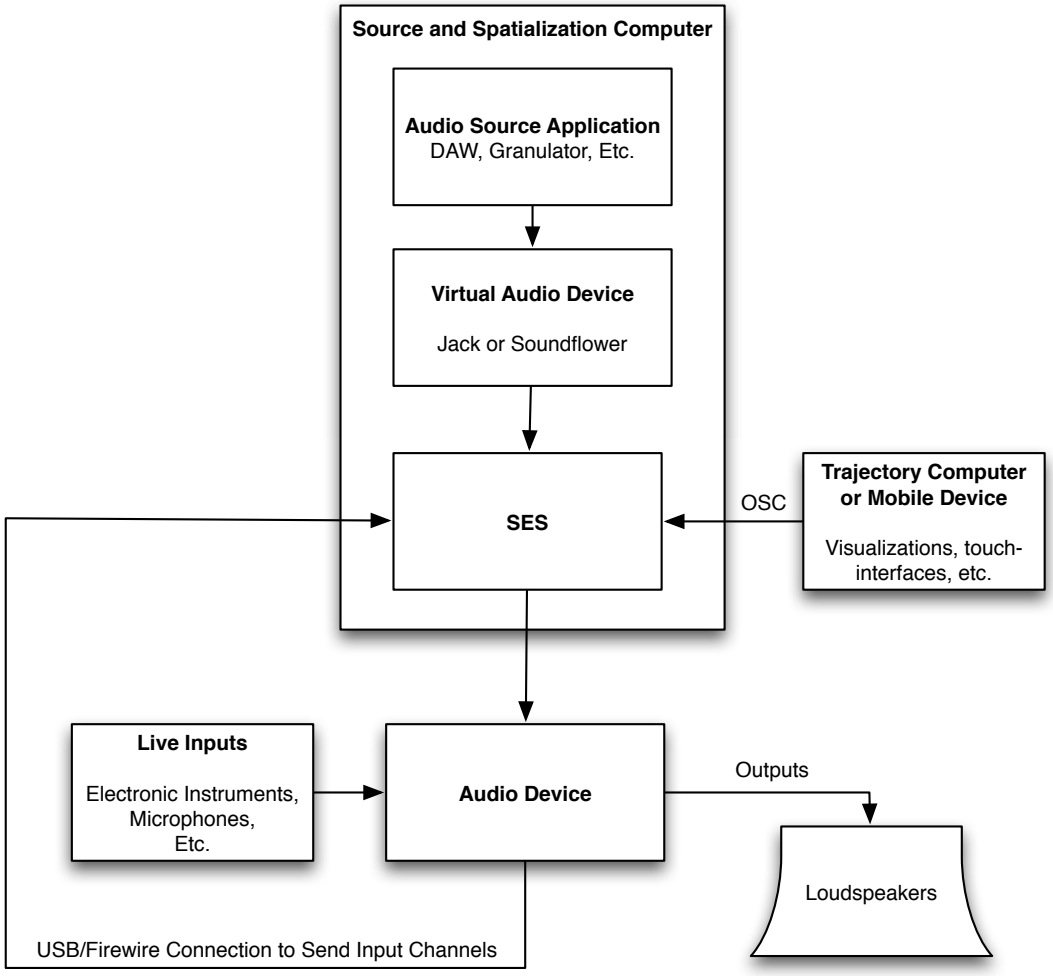
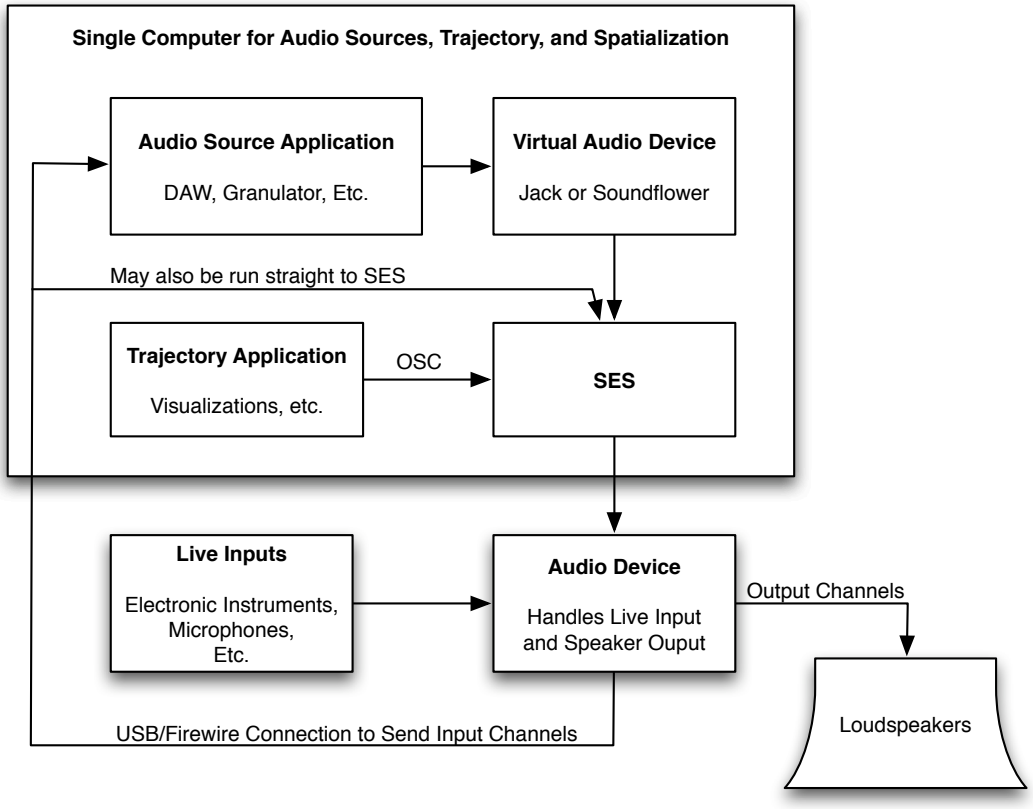


Figure 2. Distributed Processing with SES Over Two Computers



Using a virtual audio device such as Jack [27] or Soundflower [28], one may realize a complete spatialization system with SES on a single computer. The user routes output from the sound generating application to SES through the virtual audio device and configures the OSC trajectory controller to use the network address of localhost (127.0.0.1).

Figure 3. SES on a Single Computer

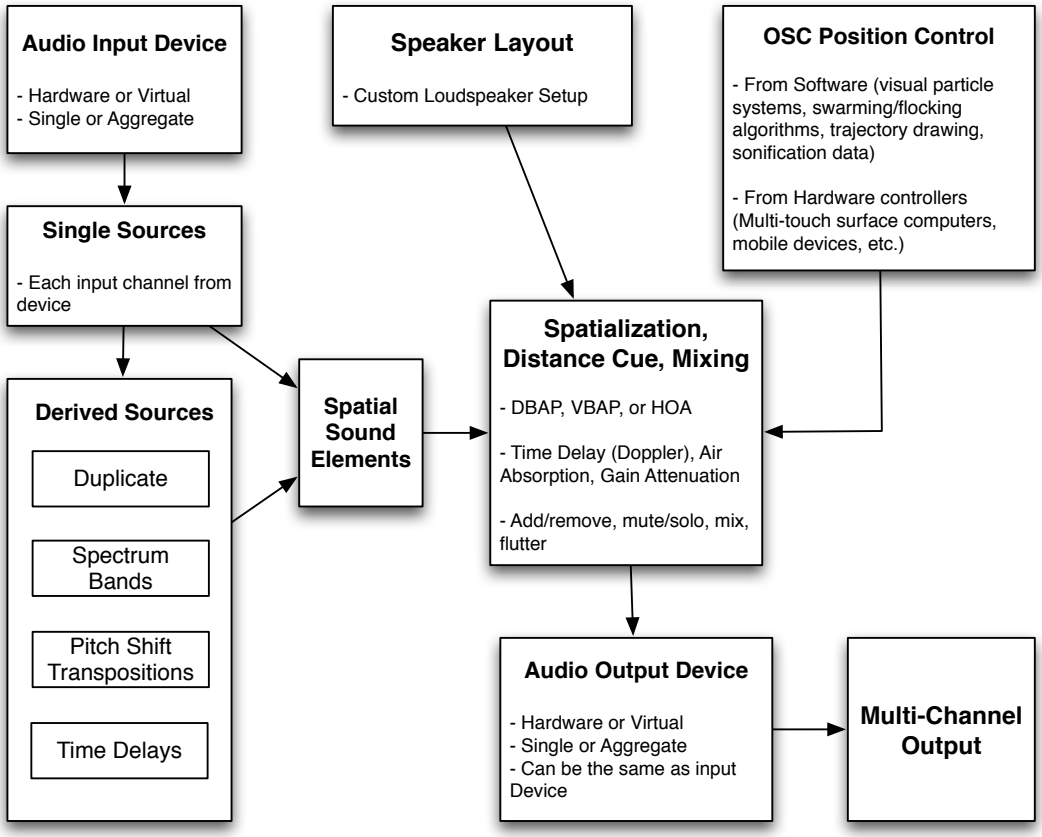


All panning algorithms in SES work with as few as 2 speakers and scale to accommodate any number of speakers, limited only by processing power. Since few audio devices have more than 16 output channels, it may often be necessary to create an aggregate device from 2 or more separate audio devices chained together to act as the final output device connected to the loudspeakers.

Software Design

SES consists of several input sources mapped to spatial sound elements positioned according to incoming OSC messages from a computer on the same network. Spatial sound elements may either be input channels from the audio input device or any number of "derived sources" – a novel feature of SES. Using derived sources the user may create several spatial sound elements from a single input channel, a method known as spatial decorrelation or decorrelated upmixing [2]. There is a graphical user-interface (GUI) to select input/output devices, select input sources, create derived sources, manage elements, load and save speaker layouts, and choose a spatialization algorithm. Figure 4 provides and overview of the entire software.

Figure 4. Abstract Overview of SES Software



Implementation

SES was written in C++ using the Create Signal Library (CSL) [29] for high-level audio processing and the Jules' Utility Class Extensions (JUCE) [30] library for the user-interface and low-level audio processing. There is frequent use of the vector class from the C++ Standard Template Library, and the SoundTouch [31] library was used for adding pitch-shifting functionality to CSL. Finally, the liblo [32] OSC library was used to receive OSC trajectory control messages.

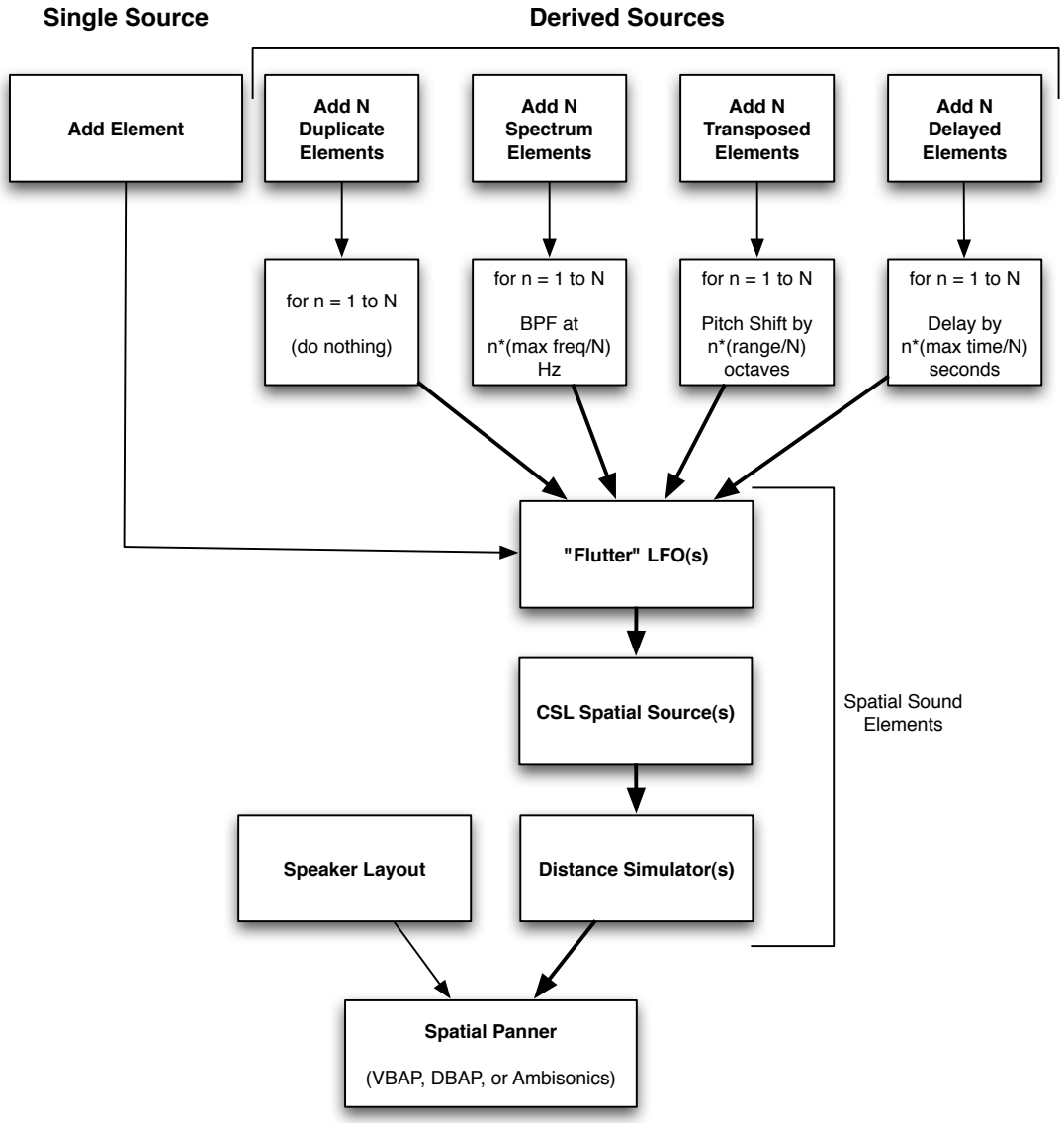
CSL already contained a basic framework for spatial audio panning, but many modifications and additions to the library were needed to realize this project. CSL contained panning classes for VBAP and HOA. Both existing panning classes were modified to smoothly interpolate their sources between positions so that sources could be moved at high speeds without a "zipper" effect from sudden change in the amplitude of the speakers. An additional panning class was created for DBAP, which scales very well to irregular speaker layouts.

CSL also contained a distance processor that took into account air absorption and gain attenuation. However, to achieve realistic motion simulation it was necessary to add time delay for sources by implementing a variable delay line. Doppler shift occurs naturally from the time delay when sources move towards or away from the listener. As with the panning in CSL, smooth interpolation was added to the distance cue to eliminate unwanted artifacts in the audio when moving sources quickly.

SES creates spatial sound elements either directly from audio input sources or from the software's novel "derived sources" feature, which currently provides four means of spatial upmixing: duplication, spectrum bands, pitch shifting (transpositions), and delays. Duplication is accomplished by repeatedly adding the desired number of sources via a code loop in C++. Sources can be divided into several frequency spectrum bands by applying CSL's Butterworth band-pass filter to several copies of the same source and incrementing the center frequency each time. It was necessary to add a new effect class to CSL to achieve real-time pitch shifting of sound sources. The SoundTouch library operates on audio buffers in the new CSL PitchShift class to increase or decrease the pitch in increments over a user specified range. CSL's multi-tap RingBuffer was used to create multiple time delays of a single sound.

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 flutter LFO waveform to use: sine, square, sawtooth, or triangle.

Figure 5. Creation of Spatial Sound Elements in SES



Audio input sources and derived input sources are instantiated as "spatial sources" in CSL. This project added the ability to control the amplitude of each spatial source with a LFO to achieve the "flutter" effect. The spatial sound elements of SES are contained as a STL vector of CSL spatial sources. The vector class allows

for the dynamic addition and removal of spatial sources, and after the number of sources is modified the chosen panning class is updated with the new element vector.

JUCE was chosen for the user-interface programming because of its use in CSL and cross-platform compatibility. While CSL contains many high-level audio processing classes (filters, panning, effects, etc.), JUCE is actually the core of CSL's low-level processing (writing audio buffers to the sound card and device management). Using JUCE, the added complexity of another C++ library was avoided and compatibility with Mac, Linux, and Windows systems was ensured. JUCE GUI components in SES allow for the dynamic selection of panning algorithm and speaker layout used to spatialize all sound elements.

Using liblo, SES reads trajectory control messages according to the SpatDIF [9] OSC format outlined below. It is important to note that "sources" in SpatDIF are analogous to spatial sound "elements" in SES.

"/SpatDIF/source/*source number*/aed *azimuth elevation distance*"

The combination "aed" indicates that the source will be positioned in terms of polar coordinates. SpatDIF also incorporates Cartesian messages in "xyz" format, but these are not used in SES. The exact message in SES for placing element number 5 at an azimuth of 330 degrees, elevation of 25 degrees and distance of 12 meters would read:

"/SpatDIF/source/*5*/aed 330 25 12"

The azimuth angle ranges from 0 to 360 degrees starting directly in front of the listener and measured clockwise. The elevation ranges from directly below the

listener, -90 degrees, to directly above the listener, 90 degrees. Hence, the elevation is fixed at 0 degrees for 2D trajectories. The distance is assumed to be in meters and ranges from 0 to 344. Since the speed of sound is approximately 344 meters per second, a maximum distance of 344 meters meant that a 1 second delay line for each source would be sufficient for the time delay component of distance cue. 344 meters is also a sufficient maximum distance because sounds are practically silent at that distance due to the gain attenuation of distance cue.

User Interface

Figure 6. Typical Order of Operations for SES GUI

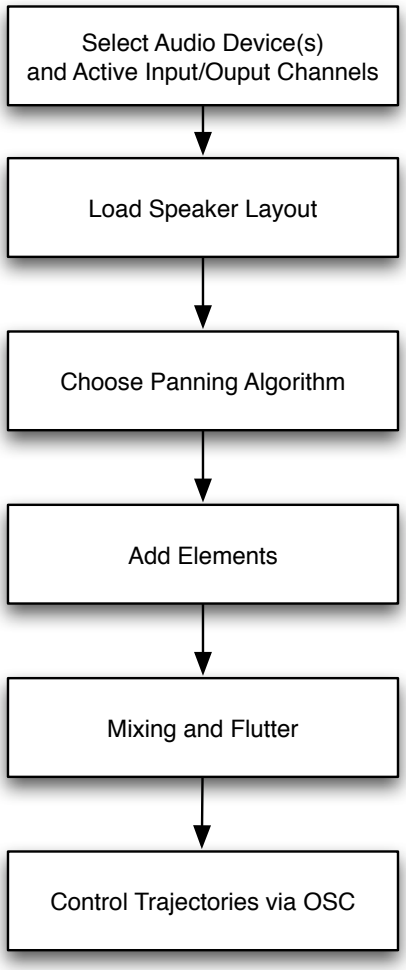
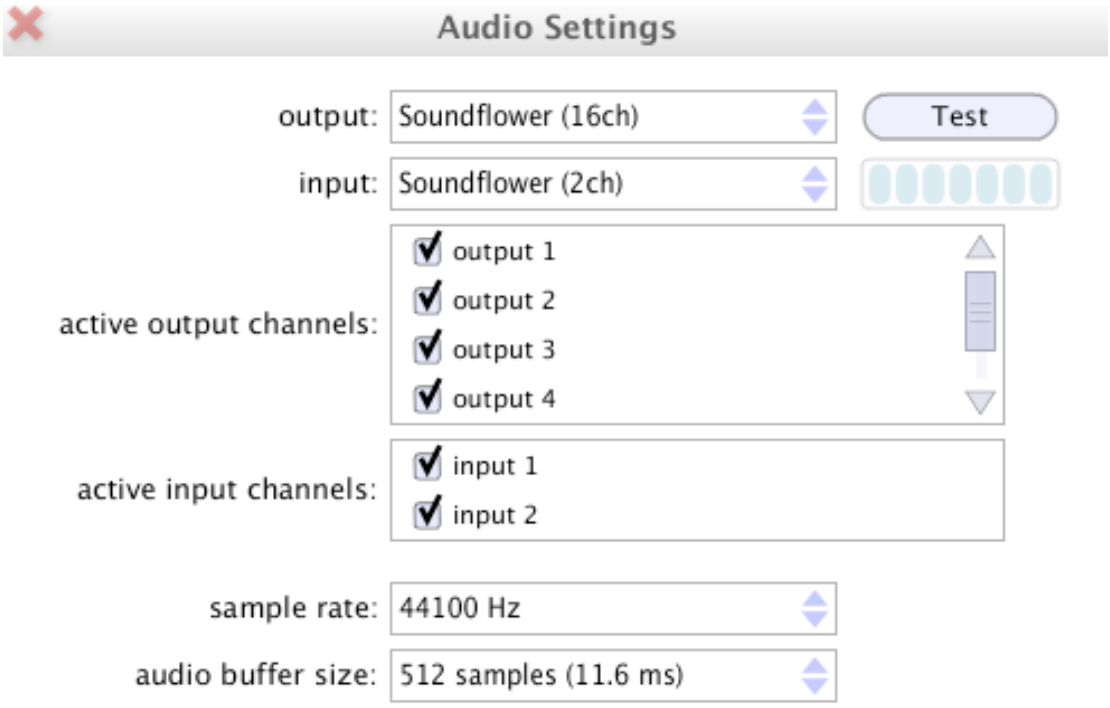


Figure 6 shows the typical order of operations within the SES GUI, though audio devices, speaker layouts, panning algorithm, and elements can all be managed dynamically after initial setup. The user begins by selecting devices for audio input and output as seen in Figure 7 below. It is possible to select different devices for input and output and specify the channels used on each device.

Figure 7. Audio Device Selection in SES



Next, the user can select the speaker layout editor from the main GUI window. The azimuth, elevation, and radius for each specific loudspeaker can be specified and saved to a file for easy loading in the future. Layouts are saved as ".spklyt" files and are plain text for editing outside of SES. SES will not allow a speaker configuration to be loaded if there are not enough available channels on the output device. Figures 8 and 9 show the speaker layout editor and .spklyt file for an octaphonic speaker layout.

Figure 8. Speaker Layout Editor in SES

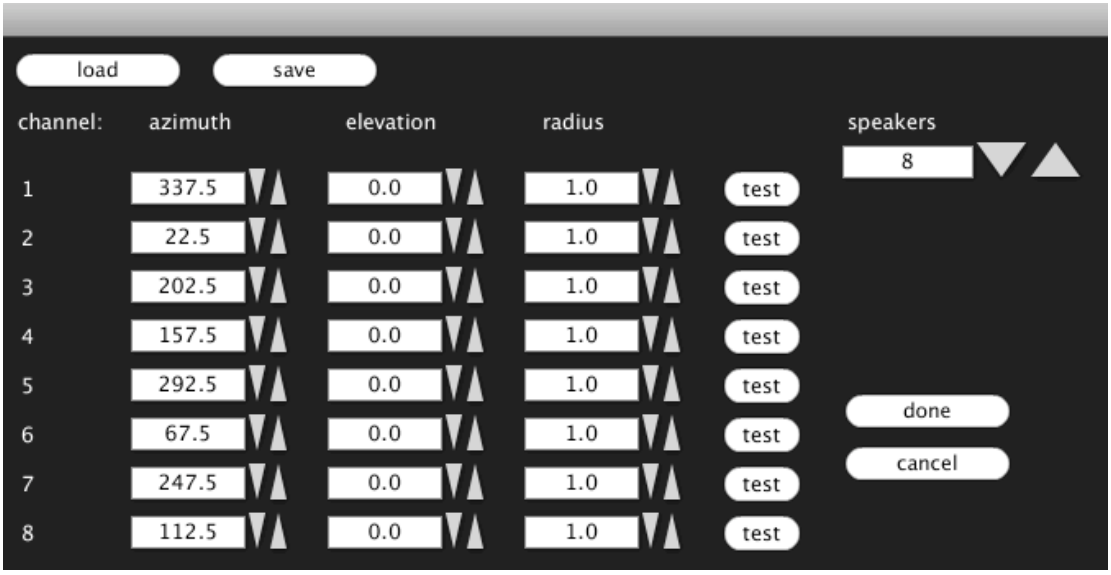
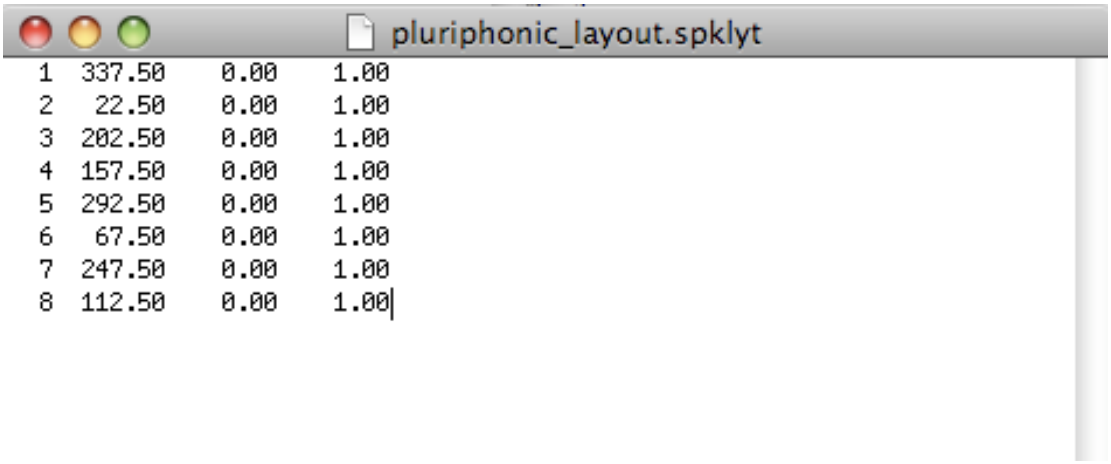


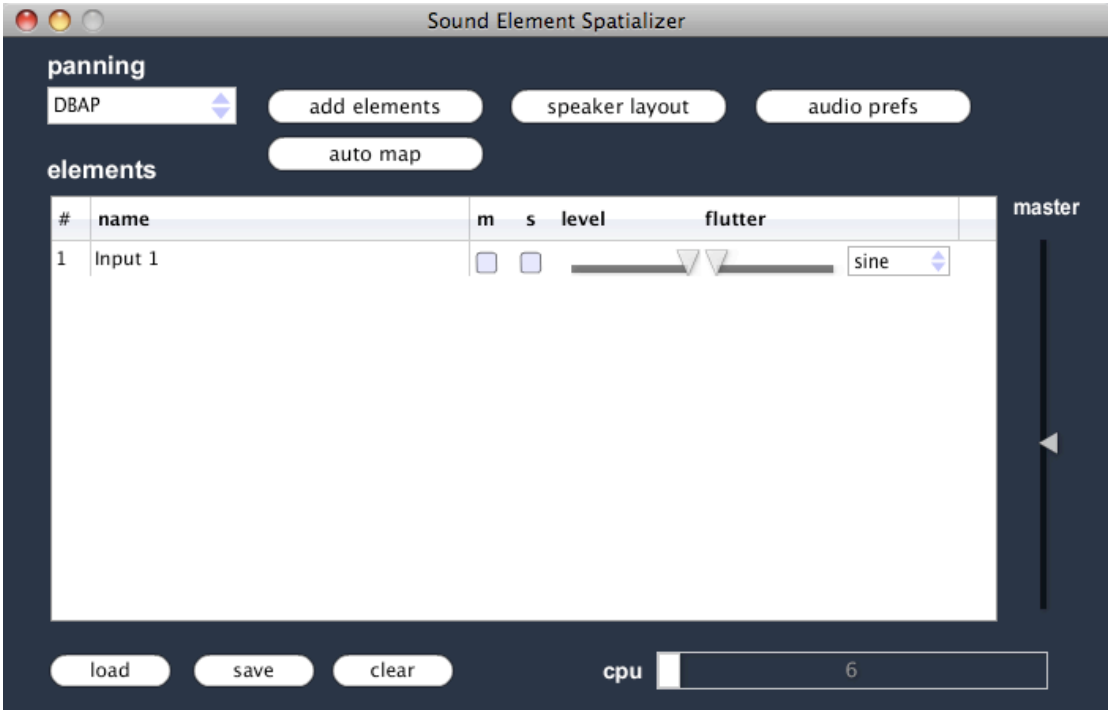
Figure 9. Example .spklyt File



The simple column based text format of .spklyt files allows for easy tweaking in a text editor outside of SES, or one may paste the columns into a spreadsheet application like Excel for more advanced editing. The example in Figure 9 corresponds to the layout in Figure 8. The first column is the speaker number followed by the azimuth, elevation, and distance for each speaker.

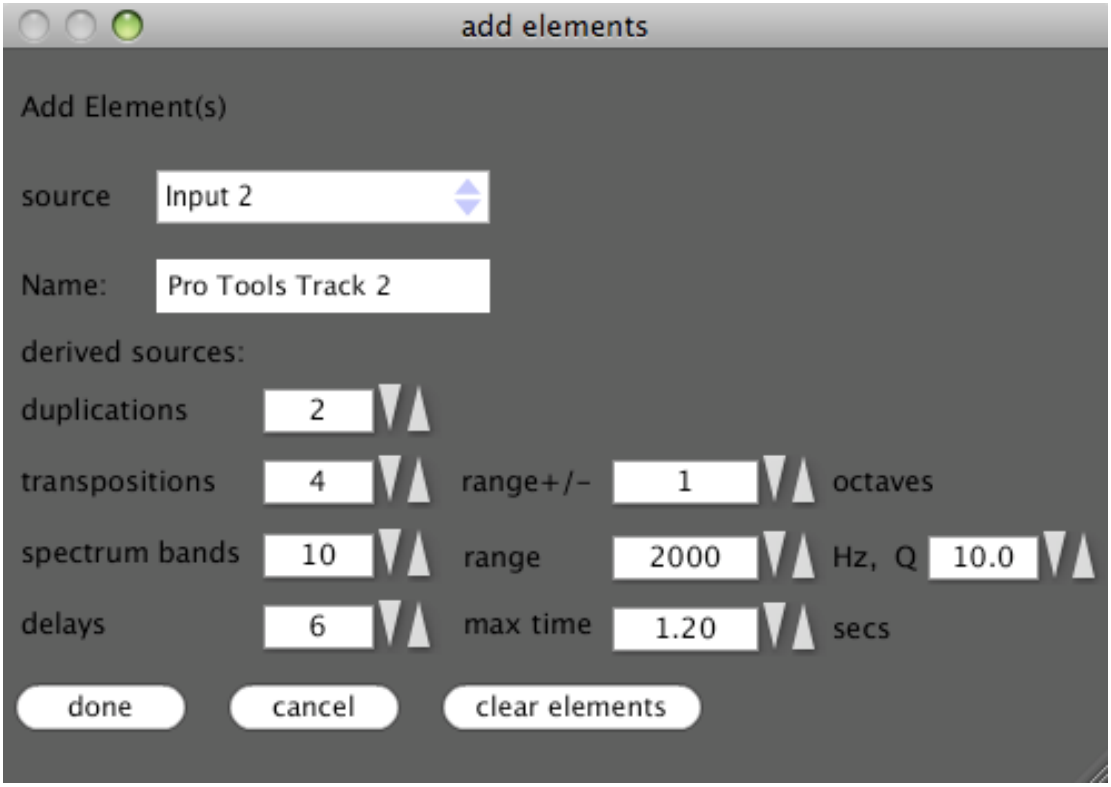
After the speaker layout is set the software returns to the main window shown in Figure 10.

Figure 10. The SES Main Window



By default the first input from the audio device is mapped to a single element. To add more elements, the user can click the "add elements" button on the main window, which reveals the window shown in Figure 11. Alternatively, the user may select the "auto map" button, which will automatically map each input channel of the audio device to a separate element.

Figure 11. Adding Elements in SES

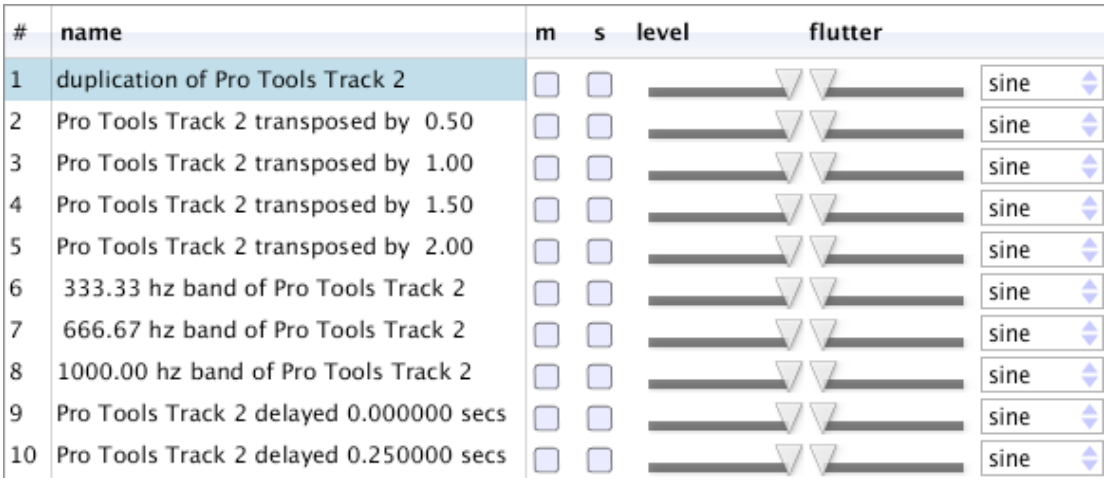


In the "add elements" window, the source drop-down menu will display all available inputs on the current audio input device. By using a virtual device such as Jack or Soundflower one may route the output from DAW software or another audio application into SES for real-time spatialization. Using a hardware input device, one may spatialize live audio inputs in real-time. If one has created an aggregate device (a combination of two or more devices) on their system then it is possible to receive virtual and hardware inputs simultaneously. The user can assign a name to each element for clearer identification during later editing.

Within the "add elements" window, if no derived sources are created then the selected input source becomes a single element for spatialization. However, if the

user selects any number of derived sources then that number of elements will be created and require individual control for spatialization. For example, one may choose to create 1 duplication, 4 transpositions, 3 spectrum bands, and 2 delays from a single sound. This would bring the total to 10 sources to be spatialized according to 10 separate OSC control messages. The main window would update the table of elements as seen in Figure 12. Note that this is a modest example since SES can process over 100 elements simultaneously. The derived sources feature is extremely useful when mapping one sound to a large number of trajectory nodes. Additionally, derived sources can be used to create several spatial effects from the amplitude and frequency modulations that result from panning and Doppler shift.

Figure 12. SES Table Demonstrating Derived Sources



As seen in figure 12, the level, flutter rate, and flutter waveform can be dynamically controlled for each element. Traditional mute and solo options are also available for monitoring results. Elements can quickly be removed by selection (as with element 1 above) and pressing delete or backspace.

Currently, users are able to select between DBAP, VBAP, and HOA panning. An implementation of WFS in SES is forthcoming. The panning algorithm can be changed dynamically from the drop down box on the main window. This allows sound artists to quickly experiment with the different panning techniques to find the one that best suits their space or composition. Table 2 briefly summarizes the advantages of each panning algorithm.

Table 2. Features of Panning Algorithms

	Arbitrary Speaker Positions	**Arbitrary Listener Position**	**Efficient Computation**	**Synthesis of Wavefronts**
DBAP	✓	✓		
VBAP			✓	
HOA			✓	✓
WFS		✓		✓

DBAP computes the gain for each speaker based on the distance from the sound source to that speaker without considering the position of the listener. DBAP also assumes the every speaker is active all of the time, so while a speaker's gain may be zero, it is still necessary to compute its gain with each position update. Thus, DBAP is ideal for irregular speaker layouts and non-fixed listeners, but is not as computationally efficient as VBAP, which only needs to compute gains for at most 3 speakers at any given time. VBAP simply triangulates the speaker configuration to determine the best 3 speakers to represent a source's position. However, VBAP

works best for hemisphere configurations in which each speaker is equidistant from a listener at a central point.

Unlike DBAP and VBAP, HOA and WFS techniques attempt to physically synthesize the wavefronts emitted from a sound source. These techniques allow for sound sources to appear to originate from points closer than any speaker. While HOA is efficient and works with as few as 4 speakers, it works best when the speakers are places on the axis of the Cartesian coordinate system. WFS, on the other hand, requires large amounts of speakers (usually at least 16) and computing resources. WFS allows for multiple listeners to perceive the same source, while HOA assumes the listener is positioned at a central "sweet spot." Though WFS has been credited as the most realistic spatial rendering technique by experienced composers [2], current real-time implementations are limited to sound in a 2D horizontal plane. True 3D WFS has not been perfected and will require even greater hardware and computational resources to operate in real-time.

Trajectory Control

Spatial sound elements are controlled by OSC messages in SpatDIF format, which allows for an extremely broad range of software and hardware controllers. For example, each element may map to the position of a node in a visual flocking algorithm, or receive control messages from a touch-screen device such as an iPad or iPhone. The possibilities for trajectory control are virtually limitless, though every user of SES is not expected to have the technical knowledge required to code their own trajectory controller. Consequently, there is a need for developers of trajectory controllers, editors, and sequencers. Visual artists may fill this need by simply adding SpatDIF OSC output to their code. I will present some basic trajectory control interfaces that are by no means exhaustive of the trajectory control possibilities for SES. The examples presented in Figures 13 through 17 were all coded in the Processing environment [33].

Figure 13. A Single Element Trajectory Controller Using the Mouse

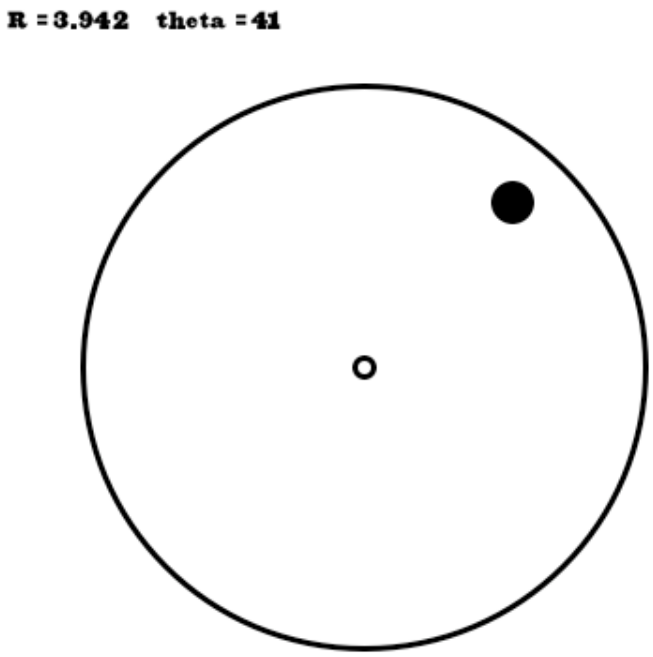


Figure 13 shows a very simple trajectory control example. We see a black dot representing an element at an azimuth angle of 41 degrees as measured counter-clockwise from the listener's front and at a distance of 3.942 meters from the listener. In this example the mouse is clicked and dragged to control the element's position. The OSC message sent from this controller would be:

"SpatDIF/source/1/aed 41.0 0.0 3.942"

Figure 14. A Multiple Element Trajectory Controller Using Boundary Collision

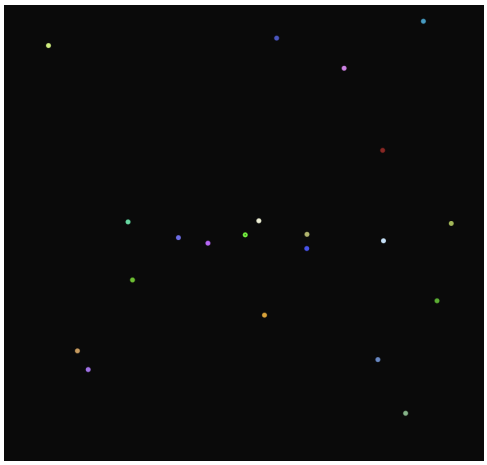
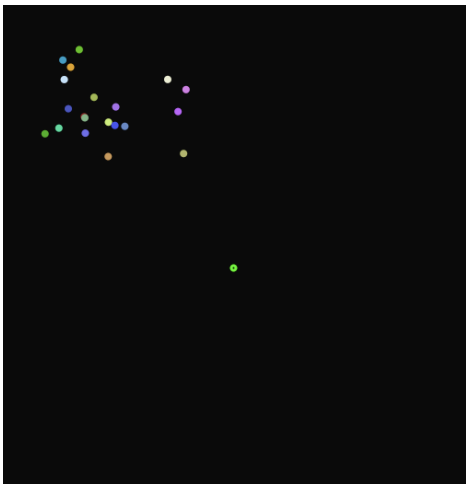


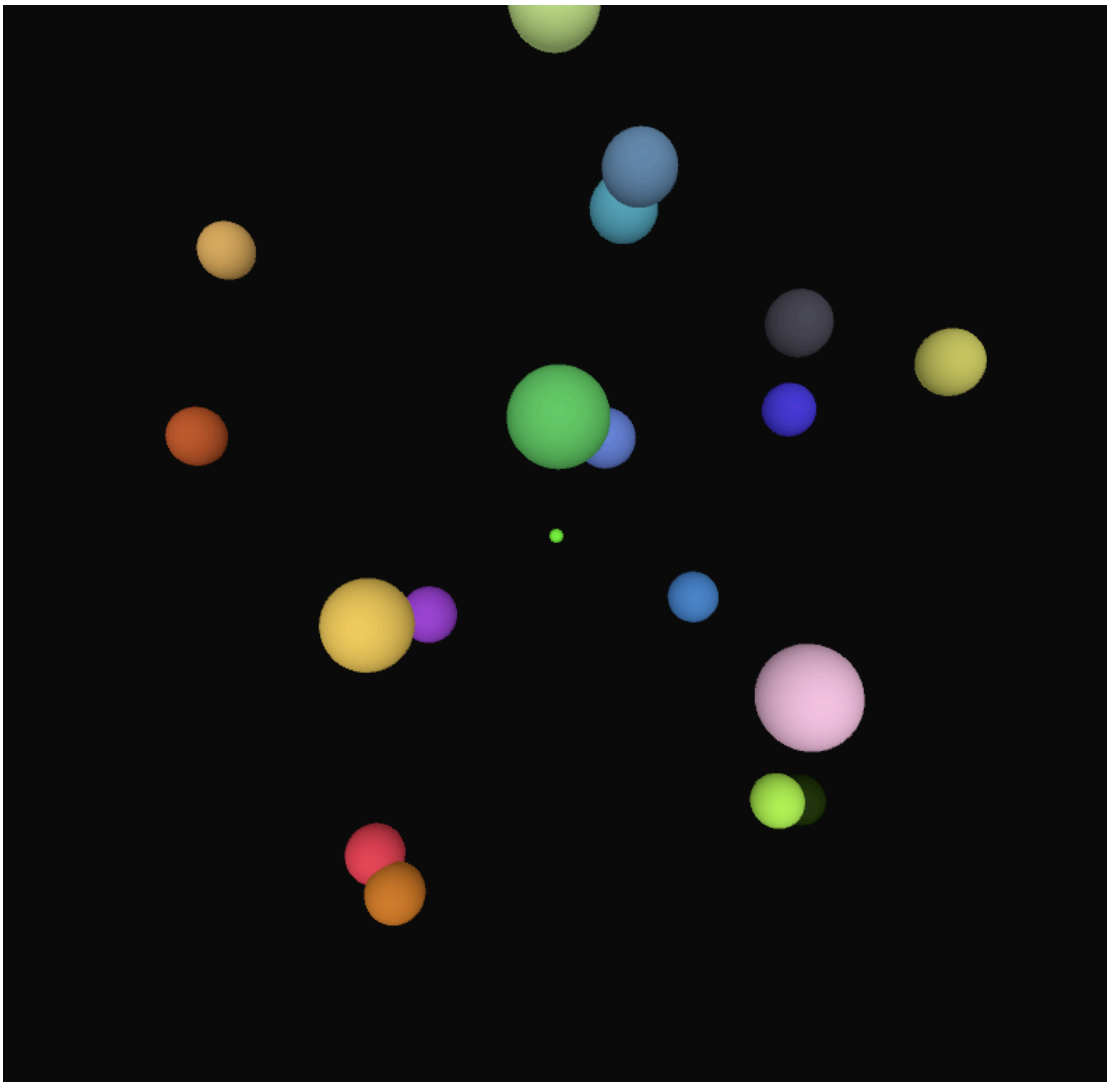
Figure 14 shows a more complex trajectory controller that utilizes more of SES's power. We see 20 different colored nodes representing separate spatial sound elements. Each node moves according to a random initial velocity and changes directions when colliding with a boundary. The velocities and boundary restraints are controlled dynamically. The composer many have a broad spatial distribution of sounds around the listener as shown above, or may wish to constrain the elements to a smaller boundary centered at some point around the listener as shown below.

Figure 15. A Bounded Cluster of Spatial Sound Elements



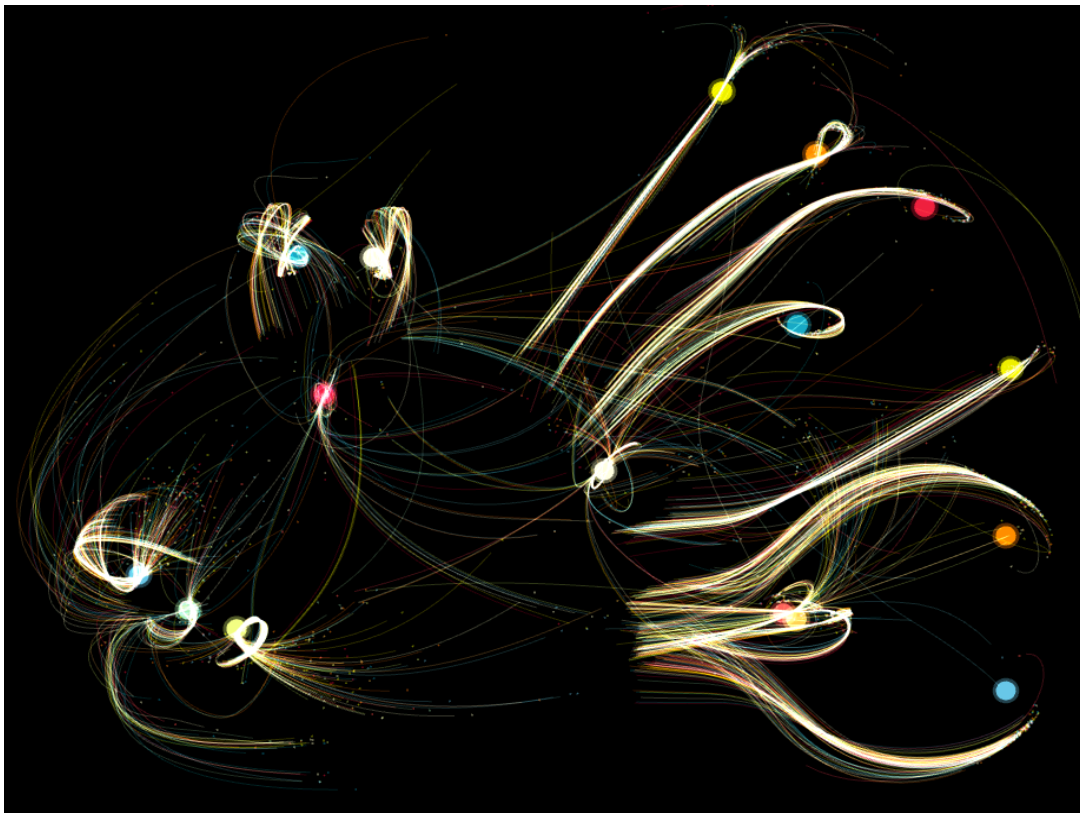
The multiple element controllers shown in Figures 14 and 15 can be extended to 3D as shown in Figure 16.

Figure 16. 3D Multiple Element Trajectory Controller



Beyond simple visualizations of point particles, one may utilize the nodes of a more complex, aesthetic visualization to control sound trajectories. Figure 17 shows such an example with the nodes representing each sound highlighted as colored circles for clarification.

Figure 17. Complex Visualization Used for Trajectory Control



The work of Visual Artist Reza Ali

Touch surface computers and mobile devices may also serve as trajectory controllers. Often to achieve live spatialization composers will adjust the levels of a mixer during the performance of a piece. While mixer sliders are limited to 1-dimensional up and down movements, touch surfaces can provide unrestricted 2D movements over a large surface (Figure 18). 3D trajectories can be obtained by using the gyrometers and accelerometers found in some mobile devices. Also, devices such as the P5 Glove (Figure 19) have the potential to provide 3D OSC trajectory coordinates [25].

Figure 18. The Media Arts and Technology Touch Table at UCSB [26]

Figure 19. The P5 Glove [25]

APPLICATIONS

Music

As mentioned, current DAW systems are heavily lacking support for spatial audio. There is a need for a spatial audio interface that easily integrates with existing software familiar to musicians and recording artists. In addition to electronic and computer music is the potential for classical musicians, vocalists, and anyone with an analog instrument and microphone have their sound spatialized for live performance.

Sound Design

Spatial tremolo and vibrato effects can be heard when moving sounds back and forth from the listener due to gain attenuation and Doppler shift in SES respectively. Adding several duplications of the same sound and moving them independently leads to a spectral blurring effect. The faster the motion, the more spectral blurring will be heard. Thus, we can use the speed of sounds to control their clarity. Interesting compositions may be made by exploring the contrast between speeding-up sound elements to form a blurred tone and slowing-down the sounds to reveal the individual elements. Using any of the SES's "derived source" methods to create multiple elements from a single sound leads to decorrelated upmixing, the process of separating a mono sound into non-similar parts and spatializing the parts separately. Decorrelated upmixing is typically used to convert mono to stereo or stereo to a consumer surround-sound format (5.1, 7.1 etc), but SES allows for upmixing to an arbitrary number of channels.

Doppler FM

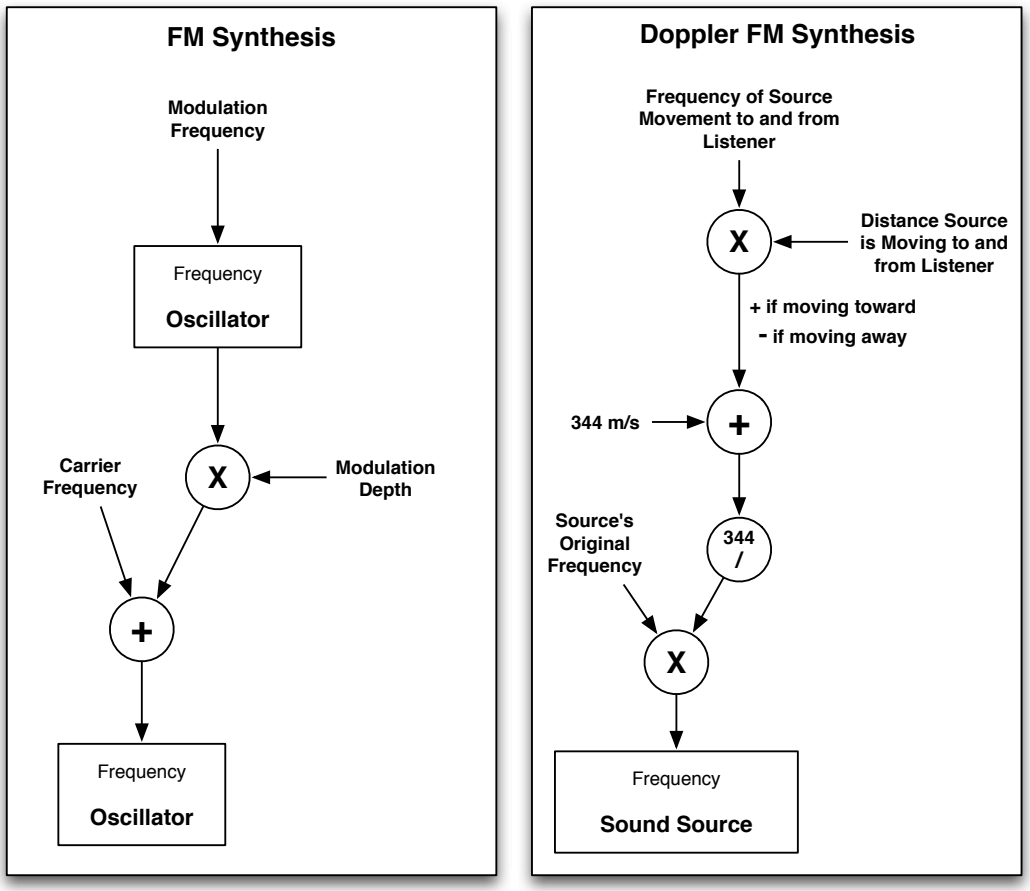
When experimenting with the early prototypes of SES I became interested in how quickly I could move sounds. After implementing Doppler shift I observed the modulation of a sound's frequency components when moving the sound quickly to and from the listener, a vibrato-like effect. By moving the sound back and forth even faster I found a new way to achieve FM synthesis using high-speed trajectories. I call this technique Doppler FM. Just as classic FM synthesis involves the modulation of an oscillator's frequency at rates in the audio domain (roughly greater than 20Hz), moving a sound towards and away from a listener over 20 times per second will modulate the frequency of the sound due to Doppler shift. Traditional FM synthesis also involves a modulation depth, or how much frequency deviation is applied during modulation, which is akin to the speed at which the sound is traveling when using Doppler FM.

For example, consider a listener in the center of a room 10 meters wide and a sound with a frequency of 300 Hz. If the sound is bounded to the room and moves towards and away from the listener 20 times per second, then the sound is covering 10 meters in 0.05 seconds, which corresponds to a speed of 200 meters per second. The change in frequency of the sound assuming a stationary listener is given by the formula for Doppler shift:

$$f = \left(\frac{v}{v + v_s} \right) f_0$$

where *f* is the resulting frequency, *v* is the speed of sound in air (344/ms), *vs* is the speed of the moving sound (positive when moving away from the listener and negative when moving towards the listener), and *fo* is the original frequency of the sound. So, for our example the resulting frequency would be (344/(344+200))*300 = 189.9 Hz when moving away from the listener and (344/(344-200))*300 = 716.7 Hz when moving toward the listener. We can obtain the modulation depth in each case by looking at the difference between the original and resulting frequencies, 110.1Hz and 416.7 Hz, respectively.

Figure 18. FM Synthesis and Doppler FM



Further, we see that for sounds with constant velocity the modulating oscillator resulting from Doppler FM will resemble a square wave with half a cycle at the "moving-towards" modulation depth and the other half at the "moving-away-from" modulation depth. To better model classic FM synthesis the moving sound should have sinusoidal acceleration and deceleration to and from its maximum velocity, resulting in a sinusoidal modulation. Also, to balance the different modulation depths it would also be necessary to have the sound move slower when approaching the listener.

Multimedia Venues

Venues for multimedia playback and performance have to potential to benefit from SES along with the artists creating works for these venues. Movie theaters and concert venues can position speakers freely and optimally within their space without worrying about standardized "surround sound" configurations. The DBAP spatialization method, for instance, responds well to irregular speaker layouts [6]. Venues also can experiment with various panning algorithms to achieve the best possible sound for their space.

Visualization

"Adding sound to data visualizations will be like the addition of sound to the silent films of the early 20th century." – Thomas Hermann [4]

Many visual artists are unconsciously creating sound spatialization algorithms through their visual work. For example, visual particle systems and flocking algorithms work well for sound spatialization. Visual artists often have a deep understanding of the physics behind the motion of moving elements. Using SES, simple OSC message output is all that is needed to transform visualization software into an audio trajectory interface. Considering this novelty, SES should inspire many visual artists to experiment with sound and vice-versa.

Sonification

Sonification involves the representation of data with sound to foster interpretation. Just as one reads a graph to discern patterns and movement in data, one may listen to a sonification to obtain similar interpretation. Sonification usually relies heavily on the frequency domain (changes in pitch) to convey information. However, with advanced sound spatialization techniques one may use the position of sounds as an added dimension for data interpretation.

Like time, space is an objective quality shared between audio and visual domains. While "sound color" (or timbre) and sound brightness may have intuitive mappings to visual stimuli, they are subjective, and thus, there is no definitive way to map the color or brightness of a sound to visual form. However, moving a sound exactly 15 meters into the distance at an angle of 30 degrees along the azimuth does have an objective mapping to a visualization since it is possible to position visual stimuli according to such parameters.

FUTURE WORK

The current version of SES stands as a proof-of-concept application and many additions are planned for future versions. First on the list is the addition of WFS to the available panning algorithms. Next, is utilization of multi-core and multi-processor systems by implementing multithreading in the C++ code. Since most of the processer intense tasks in SES depend on several code loops, multithreading should significantly increase performance on systems with more than a single processor core.

There is much more potential for the user-interface of SES. Elements should be able to be dynamically modified and there should be dynamic controls for the parameters of derived sources. There are also several other means of creating derived sources that have not been explored (onset detection, for example).

Spatialization rendering could take many more factors into account. For one, reverberation should be added as part of distance cue and possibly even as part of the speaker layout editor. Applying different amounts of reverb to individual speakers could help create new virtual spaces in which the sound elements move. Also, as new panning algorithms are invented there will always be consideration for adding them to SES.

A means for OSC upsampling would be useful when using trajectory controllers are not capable of sending position information at audio block rate. Currently, the resolution at which controllers can move sounds is limited by the rate

at which they send OSC position messages. For an audio sampling rate of 44.1kHz and block size of 512 samples, trajectory messages would need to be sent at least 86 times per second for seamless panning. Though SES's interpolation would prevent any artifacts in the sound, the position of the sound will jump between positions unless position updates are sent at block rate. Visualizations, for example, typically update 30 times per second, so upsampling by a factor of 3 is required for smooth operation with the above audio specifications.

CONCLUSION

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. It is my hope that the development of a variety of trajectory controllers from visualization software, touch-interfaces, and mobile devices will cultivate an organized theory of spatial relationships since such a grammar does not currently exist [17]. With the ability to decompose sounds and precisely control the trajectories of individual sound elements composers can consider space as significant as any other expression of timbre.

REFERENCES

[1] J. Ahrens, M. Geier, and S. Spors, *Introduction to the SoundScape Renderer (SSR)*, 2010.

[2] M. a J. Baalman, "Spatial Composition Techniques and Sound Spatialisation Technologies," *Organised Sound*, vol. 15, Oct. 2010, pp. 209-218.

[3] J.M. Chowning, "The simulation of moving sound sources," *Computer Music Journal*, 1977, p. 48–52.

[4] T. Hermann, "Taxonomy and Definitions for Sonification and Auditory Display," *Proc. of the 14th ICAD, Paris*, 2008.

[5] J.-M. Jot and O. Warusfel, "Spat~: A Spatial Audio Processor for Musicians and Sound Engineers."

[6] T. Lossius, P. Baltazar, and T. de La Hogue, "DBAP-Distance-Based Amplitude Panning," *Proceedings of 2009 International Computer Music Conference, Montreal, Canada*, 2009.

[7] A. McLeran, C. Roads, B.L. Sturm, and J.J. Shynk, "Granular sound spatialization using dictionary-based methods," *Proceedings of the 5th Sound and Music Computing Conference, Berlin, Germany*, 2008.

[8] J. Nixdorf and D. Gerhard, "Real-time sound source spatialization as used in Challenging Bodies: implementation and performance," *Proceedings of the 2006 conference on New interfaces for musical expression*, IRCAM—Centre Pompidou, 2006, p. 318–321.

[9] N. Peters, S. Ferguson, and S. McAdams, "Towards a Spatial Sound Description Interchange Format (SpatDIF)," *Canadian Acoustics*, vol. 35, 2007, p. 64–65.

[10] N. Peters, T. Matthews, J. Braasch, and S. McAdams, "Spatial sound rendering in Max/MSP with ViMiC," *Proceedings of the 2008 International Computer Music Conference*, 2008.

[11] N. Petersa, T. Lossiusb, J. Schacherc, P. Baltazard, C. Bascoue, and T. Placef, "A stratified approach for sound spatialization," *Proceedings of 6th Sound and Music Computing Conference*, 2009.

[12] S.T. Pope, "Interchange Formats for Spatial Audio," *Proceedings of the 2008 International Computer Music Conference*, 2008.

[13] S.T. Pope, *Audio in the UCSB CNSI AlloSphere*, 2005. http://www.create.ucsb.edu/~stp/AlloSphereAudio_01.pdf

[14] V. Pulkki, "Virtual sound source positioning using vector base amplitude panning," *Journal of the Audio Engineering Society*, vol. 45, 1997, p. 456–466.

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

[16] C. Roads, "The Nature of Sound," *Composing Electronic Music*, Oxford University Press, 2010. (draft, publication forthcoming)

[17] C. Roads, “Articulating Space,” *Composing Electronic Music*, Oxford University Press, 2010. (draft, publication forthcoming)
[18] M. Schumacher and J. Bresson, “Spatial Sound Synthesis in Computer-Aided Composition,” *Organised Sound*, vol. 15, Oct. 2010, pp. 271-289.
[19] A. Sontacchi and R. Höldrich, “Getting Mixed Up With WFS, VBAP, HOA, TRM…’ from Acronymic Cacophony to a Generalized Rendering Toolbox,” *DEGA Wave Field Synthesis Workshop*, 2007.
[20] S. Spors, R. Rabenstein, and J. Ahrens, “The theory of wave field synthesis revisited,” *124th AES Convention*, 2008.
[21] B. Sturm, C. Roads, A. McLeran, and J. Shynk, “Analysis, Visualization, and Transformation of Audio Signals Using Dictionary-based Methods,” *Journal of New Music Research*, vol. 38, Dec. 2009, pp. 325-341.
[22] S. Wilson, “Spatial Swarm Granulation,” *Proceedings of the 2008 International Computer Music Conference*, SARC, ICMA, 2008, pp. 4-7.
[23] S. Wilson and J. Harrison, “Rethinking the BEAST: Recent developments in multichannel composition at Birmingham ElectroAcoustic Sound Theatre,” *Organised Sound*, vol. 15, Oct. 2010, pp. 239-250.
[24] “Introduction to OSC.” http://opensoundcontrol.org/introduction-osc
[25] “P5 Glove.” http://www.vrealities.com/P5.html
[26] “MAT 200C Student Projects,” 2010. http://mat.ucsb.edu/~ryan/200C_site/
[27] “Jack OS X.” http://www.jackosx.com/
[28] “Soundflower.” http://cycling74.com/products/soundflower/
[29] “CREATE Signal Library.” http://fastlabinc.com/CSL/
[30] “Raw Material Software.” http://www.rawmaterialsoftware.com/juce.php
[31] “SoundTouch Sound Processing Library.” http://www.surina.net/soundtouch/
[32] “liblo: Lightweight OSC implementation.” http://liblo.sourceforge.net/
[33] “Processing.” http://processing.org/