University of California
Santa Barbara

Scanning Spaces: Paradigms for Spatial Sonification and Synthesis

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Media Arts and Technology

by

Ryan Michael McGee

December 2015

The Dissertation of Ryan Michael McGee is approved.

Professor Curtis Roads

Professor George Legrady

Professor JoAnn Kuchera-Morin, Committee Chair

December 2015

Scanning Spaces:

Paradigms for Spatial Sonification and Synthesis

To my grandfathers.

Acknowledgements

A deep thank you to the following people and organizations for supporting this dissertation.

- Dissertation Committee: Dr. JoAnn Kuchera-Morin, Dr. Curtis Roads, George Legrady
- Professors and Mentors: Matthew Wright, Andres Cabrera, Stephen Travis Pope, Clarence Barlow, Marcos Novak
- The UCSB Media Arts and Technology Community and AlloSphere Research Group: Reza Ali, Yuan-Yi Fan, Lance Putnam, Graham Wakefield, Karl Yerkes, Pablo Colapinto, Dennis Adderton, Myles Sciotto, Charlie Roberts
- Collaborators: Daniel Ashbrook, D.V. Rogers, Damon Seeley
- A generous Graduate fellowship from the Robert W. Deutsch Foundation

Curriculum Vitæ

Ryan Michael McGee

Education

2015	Ph.D. in Media Arts and Technology, University of California, Santa Barbara.
2010	MS in Media Arts and Technology, University of California, Santa Barbara.
2008	BS in Electrical Engineering, The University of Texas at Dallas

Academic Experience

2013-15	Teaching Associate, Department of Art, University of California, Santa Barbara
2012-14	Graduate Student Researcher, AlloSphere Research Group, University of California, Santa Barbara
2011	Graduate Student Researcher, Department of Physics, University of California, Santa Barbara
2010-11	Teaching Assistant, Department of Physics, University of California, Santa Barbara
2009-11	Teaching Assistant, Media Arts and Technology, University of California, Santa Barbara

Professional Experience

2012-Present	Independent Commercial C++ Software Developer, Multiple Clients and Projects in Los Angeles and San Francisco Involving Audio Plug-ins, Multi-touch interfaces, and New Media Installations
2012-Present	Owner and Software Developer, Unfiltered Audio LLC, Santa Barbara, CA
2011	Research Intern, Nokia Research Hollywood
2007-08	Electrical Design Engineer, Polatomic, Inc., Richardson, TX
2007	Student Engineer, Southwest Research Institute, San Antonio, TX

Publications

McGee, R., Spatial Modulation Synthesis. Proceedings of the International Computer Music Conference (ICMC). 2015.

McGee, R., VOSIS: a Multi-touch Image Sonification Interface. Proceedings of New Interfaces for Musical Expression (NIME). 2013.

McGee, R., Dickinson J., Legrady G. The Voice of Sisyphus: an Image Sonification Multimedia Installation. Proceedings of the International Conference on Auditory Display (ICAD). 2012.

McGee, R., Ashbrook D., White S. SenSynth: a Mobile Application for Dynamic Sensor to Sound Mapping. Proceedings of New Interfaces for Musical Expression (NIME). 2012.

McGee, R. and Wright, M. Sound Element Spatializer. Proceedings of the International Computer Music Conference (ICMC). 2011.

McGee, R., Fan, Y.Y., and Ali, S.R. BioRhythm: a Biologically-inspired Audio-Visual Installation. Proceedings of New Interfaces for Musical Expression (NIME). 2011.

McGee, R., van der Veen, J., Wright, M., Kuchera-Morin, J., Alper, B., and Lubin, P. Sonifying the Cosmic Microwave Background. Proceedings of International Conference on Auditory Display (ICAD). 2011.

Abstract

Scanning Spaces:

Paradigms for Spatial Sonification and Synthesis

by

Ryan Michael McGee

In 1962 Karlheinz Stockhausen's "Concept of Unity in Electronic Music" introduced a connection between the parameters of intensity, duration, pitch, and timbre using an accelerating pulse train. In 1973 John Chowning discovered that complex audio spectra could be synthesized by increasing vibrato rates past 20Hz. In both cases the notion of acceleration to produce timbre was critical to discovery. Although both composers also utilized sound spatialization in their works, spatial parameters were not unified with their synthesis techniques. This dissertation examines software studies and multimedia works involving the use of spatial and visual data to produce complex sound spectra. The culmination of these experiments, Spatial Modulation Synthesis, is introduced as a novel, mathematical control paradigm for audio-visual synthesis, providing unified control of spatialization, timbre, and visual form using high-speed sound trajectories.

The unique, visual sonification and spatialization rendering paradigms of this dissertation necessitated the development of an original audio-sample-rate graphics rendering implementation, which, unlike typical multimedia frameworks, provides an exchange of audio-visual data without downsampling or interpolation.

Contents

Chapter 1

Introduction

1.1 Scanning Spaces

"Scanning Spaces" is the body of work leading to and including Spatial Modulation Synthesis. This dissertation explores work traversing several disciplines including music composition, sound art, synthesis, sonification, signal processing, and computer science. The content of each chapter deals with both technical and aesthetic concepts along the notion that artistic creativity drives the development of technology. In fact, a majority of the work in this dissertation originated from desires as a composer and sound designer to accurately simulate high-speed moving sound sources and to make the visible audible.

The companion website for this dissertation containing several multimedia examples is online at `http://www.spatialmodulation.com`.

1.2 Problem Statement

Composers Karlheinz Stockhausen and John Chowning both created a continuum between acceleration and timbre while making extensive use of sound spatialization in their works, but technical limitations ultimately left spatial parameters separate from their timbral and synthesis control paradigms. However, distance cues based on physical first principles such as Doppler shift and distance-based gain-attenuation provide an intrinsic, mathematical connection between moving sound sources and modulation of their frequency and amplitude. While distance cues are often simulated at relatively low speeds (under 100 m/s), spatial modulation synthesis aims to be the first rigorous control paradigm utilizing this physical connection between space and timbre up to the speed of sound by overcoming technical obstacles of accurate software implementation.

The Doppler shift approximation formula ubiquitous in the field of digital audio is physically inaccurate at high-speeds over approximately 100 m/s (225 mph). While this approximation and speed limit have been sufficient for existing Doppler-based effects, a true, physically accurate Doppler implementation is needed to explore the full frequency range of Doppler-based frequency modulation at high velocities.

Current multimedia software frameworks provide inadequate control rates for spatialization trajectories which limit accurate representation of high-speed oscillating motion for virtual sound sources. These frameworks also involve separate threads for audio and graphics rendering as the two are assumed to be either independent or operate with audio parameters modulating graphics. However, spatial modulation synthesis desires the opposite – spatial, graphical data used to modulate sound, and thus, requires a novel, unified audio-visual rendering paradigm to synthesize and visualize audio-rate frequency and amplitude modulation from simulated spatial sound motion.

1.3 Outline of the Dissertation

After a brief historical overview of music, art, and technology involving spatial sound in Chapter 2, Chapter 3 describes the original work culminating in the theory of spatial modulation synthesis, which is described and evaluated in Chapters 4 and 5. Future development directions for spatial modulation synthesis are discussed in Chapter 6, and conclusions regarding its impact are stated in Chapter 7. Appendices A and B summarize the mathematical formulas and audio-visual timbres resulting from spatial modulation synthesis respectively.

Chapter 2

Background

2.1 Sound Spatialization in Composition

Beginning with Karlheinz Stockhausen's monumental *Gesang der Junglinge* in 1956, several 20th century works of electronic and computer music have utilized sound spatialization as an independent parameter of composition choreographed alongside timbre, pitch, intensity and duration. In 1958, Edgar Varese's *Poeme Electronique* premiered at the historic Phillips Pavilion using a multi-channel spatialization system controlled by rotary telephone dials to send sounds around the complex architecture of the venue in 3 dimensions. While early choreography of spatial trajectories in music relied simply on amplitude panning between speakers, John Chowning's 1977 paper, The Simulation of Moving Sound Sources [1], detailed various techniques to implement localization cue. Thereafter, many spatial compositions have utilized more realistic, physically based distance simulation including Doppler shift and gain attenuation with distance. Chowning's *Turenas* is a prime example of "kinetic" music that features sounds following trajectories of Lissajous curves around a lister centered in a quadraphonic loudspeaker arrangement [2]. Indeed, the piece's title, an anagram of "natures," suggests sounds that "tour" around a listener. Jean-Claude Risset's *Songes* experiments with sounds that "swiftly fly through space," hinting at an illusion of birds flying around the listener [2]. As Henry Brandt describes in his 1967 article,"Space as an Essential Aspect of Music Composition," [3] spatialization of sound is used by composers to disperse and clarify dense textures of several sound sources.

In addition to spatial composition on the macro level, granular synthesis techniques allow for spatialization at the level of micro sound. Techniques such as Pulsar Synthesis [4] involve the spatialization of grains as a fundamental parameter. Granular implementations are typically stereo, but Scott Wilson's Spatial Swarm Granulation [5] utilizes a 3D point-source model for placing sound grains, akin to the spatial model used in most 3D sound spatialization paradigms [6].

2.2 Sound Synthesis Utilizing Spatial Motion

In the 1930's Donald Leslie began developing a rotating speaker for a Hammond organ to more closely emulate a pipe organ. Similarly, between 1958 and 1960 Stockhausen's *Kontakte* used a rotating speaker captured by four microphones to produce sounds affected by the speed of rotation. In the modern digital sound domain, emulations of rotating speakers, simply termed "Leslie" effects, are used to add a rich chorusing effect to sounds. These effects also involve many other considerations for echo reflections, filtering, distortion, and speaker characteristics that occur with a physically rotating speaker. However, what is shared between Leslie, Stockhausen, and the ideas expressed in this dissertation is a common notion of physical motion to affect timbre.

In 2000 Bill Verplank and Max Mathews introduced scanned syntheses, which involves the playback of non-audio data at faster rates to produce audible frequencies [7]. Initially, Verplank experimented with a string model oscillating at haptic frequencies and scanning the motion fast enough to produce tones. His work involved the transformation of visible spatial frequencies into the audio domain. Similarly, wave terrain synthesis is a form of scanned synthesis that scans 3D terrains to provide a connection between space and timbre [8].

The multi-dimensional pixel position and color spaces within digital images have also been used for sound synthesis. Iannis Xenakis' UPIC and popular commercial software such as MetaSynth and Adobe Audition use a "time-frequency" approach to scanning images as sound that has been termed graphic synthesis [9]. These pieces of software consider an image much like a musical score in which the vertical axis directly corresponds to frequency and the horizontal axis to time. Usually the image is drawn, but some software like Audition allows the use of bitmap images and considers color as the intensity of frequencies on the vertical axis, using the image as a spectrograph for the synthesized sound. MetaSynth offers a number of other visual-to-sound mapping possibilities including the association of color to stereo position.

2.3 Spatial Sound Art

Traditionally, sound composition has been temporally focused as music. However, many 20th century artists brought spatial elements to sound composition. Robin Minard is one such artist that conditions architectural spaces with layers of sound. Rather than animating and moving sounds in space, he relies on the listener to move through the sonic spaces he creates to modulate the sound. His 1985 piece, *Music for Passageways* involves spatial localization of separate timbres over 32 speakers and has been installed in several public areas such as foyers. Though on a much slower time scale than a rotating speaker or simulated moving sound sources, there is again the notion of spatial movement to create timbre in Minard's work.

Max Neuhaus is another artist using motion to morph sounds in his work. In 1967 he created *Drive In Music* which involved seven radio transmitters positioned along a highway so that a car's direction and speed would determine the piece. Like Minard, the listener's motion was critical to the produced sound.

Edwin van der Heide takes an alternative approach to motion than Minard and Neuhaus. Rather than relying on the listener's motion, his 2000 piece *Spatial Sounds* physically moves sound sources up to 100 km/h as listeners observe from a relatively fixed position. Using a speaker mounted on a 3 meter long rotating arm, the effects of Van der Heide's work are similar to Leslie speakers, but with much deeper Doppler shift and amplitude modulation due to the greater distance of motion for the speaker.

2.4 Audio-Visual Composition

Along with the spatialization of sound in the 20th century, visual arts became more temporally focused from pioneering audio-visual artists and the concept of visual music. Sergei Eisenstein was a 20th century Russian filmmaker who documented various ideas of visual music which he referred to as "montage." He broke down visuals into fundamental units of structure that he termed "cells" and composed on metric, rhythmic, and tonal layers with these cells [10]. The metric layer refers to the time durations of cells, while the rhythmic is concerned with larger ordering structures such as phrases. The tonal layer is concerned with the color, texture, and shape of the visuals. Many parallels regarding the decomposition of media elements into smaller, organized constituents can be observed between Eisenstein's visual philosophy and Stockhausen's concept of unity in electronic music. Eisenstein is also credited with the idea of "vertical montage" – a unified method of audio-visual composition with the visual score as simply another line (vertically) above the musical score.

Adriano Abbado has defined three correspondences between abstract animation and synthetic sound [11]. The first correspondence is timbre-shape. In music we use timbre to identify what the sound is or what produced the sound. If a trumpet and clarinet are playing the same pitch at the same loudness then each instrument's timbre encompasses the differences in their sounds. Timbre is the defining characteristic of a sound source and correlates with visual shape because shape tends to be a more defining characteristic than color for any visual. Color is typically of less importance and is often used to enhance the perception given ultimately by shape. The second correspondence is perceived location. Here, Abbado takes the uncommon approach of mapping visual space to auditory space. Typically in music we correlate visual space to pitch, using up or down to represent higher or lower pitches and left-to-right to represent advancing time. However, Abbado suggests the idea of hearing visuals as sound sources from the direction seen in a projection. The final correspondence Abbado discusses is perceived intensity. He correlates intensity with loudness in sounds and brightness in images. This implies bright visuals for loud sounds and dark visuals for quiet sounds. This approach seems to agree with most people's intuition and the fact that both black visuals and silent sounds are represented by a value of 0 in the digital domain.

2.5 Sonification

Sonification can be defined as the process of making data audible in order to perceive relationships that facilitate communication and interpretation [12]. This can be accomplished by the direct conversion of data to sound (audification), the mapping the of data to control variables of some audio source (parameter mapping), or the use of some model as a sound generator requiring interaction to produce sound (model-based) [12]. Audification is simply a rescaling of any data set to fall between digital audio amplitude values of -1 and +1 played back at a variable rate fast enough to produce audible frequencies (above 20 Hz). Parameter mapping sonification literally maps data to control the parameters of one or more sound generators. The parameters may be fundamental elements of oscillators such as amplitude, frequency, and envelope or may be more abstract such as pleasantness, brightness, or roughness of a sound. Data can also be mapped to more complex synthesis processes to control the density of sound grains, frequency modulation index, or spatial position over multiple speakers. Parameters may exist on larger time scales as well such as tempo and meter. Thus, sound spatialization falls within the realm of parameter mapping sonification in the sense that it is a mapping from some trajectory data to a sound's spatial position. Lastly, model-based sonifications are digital sound instruments behaving according to some model, which may or may not be physical. For instance, the VOSIS application described in Section 3.5 uses a non-physical model for image sonification and requires user interactivity to produce and temporalize sound.

Chapter 3

Practice

"Scanning Spaces" refers to the series of sound spatialization and sonification experiments between 2009 and 2014 that has ultimately lead to spatial modulation synthesis. The following sections of this chapter describe both artistic and technical projects involving the acceleration of spatial trajectories and/or the acceleration of spatial data. The final section describes how the results and desires stemming from the practice drove the development of software for spatial modulation synthesis. All of the works have been demonstrated publicly, resulting in the following performances and publications:

Specific links to relevant media are provided in each chapter, but all media can be found online at `http://www.spatialmodulation.com`.

3.1 *W.A.N.T.S.* (2009)

W.A.N.T.S. is an 8 channel sound composition that made use of custom spatialization software to simulate Fibonacci spirals of sound moving around a listener. The piece used high-speed sound trajectories, rotating hundreds of times per minute, to create amplitude modulation and granulation effects. *W.A.N.T.S.* was played publicly in UCSB's Lotte Lehmann Concert Hall on February 25th, 2010. A stereo recording is online at `https://soundcloud.com/lifeorange/wants`.

3.1.1 Concept

The idea for this piece came to me as I was taking my weekly motorcycle ride down California's Pacific Coast highway. While riding at highway speeds for most the 90 mile trip from Santa Barbara to Santa Monica, hints of musical structure started to appear in the environmental noise reaching my ears through the helmet. I began to associate particular speeds, geographic locations, and riding positions with the modulation of wind noise. I decided to capture field recordings of this experience to create the first, musique concrète[1] layer of sound comprising the work.

The second layer of sound material was comprised of vocal samples with lyrics derived from states of self-reflection in which I often found myself while riding. This involved thoughts about how my personality and surroundings (social and geographical) have drastically changed since moving to California. I thought about parts of personality were no longer present, such as the cautious part that would never have owned a motorcycle because they were too dangerous. I thought about the the variety of landscapes and geographical features in California that contrast the completely flat, oceanless landscape of my previous home in Texas. So, the piece expresses a simple mantra, "we are not the

[1]`https://en.wikipedia.org/wiki/Musique_concrte`

same" in which "we" refers to dualities present within any person experiencing a drastic change of surroundings. The acronym for this phrase is also a duality in the sense that it refers to "wants" or desires for change.

The piece was realized as an assignment for a graduate course in which spatialization over 8 loudspeakers was a requirement. The particular loudspeaker arrangement provided was a circle, which lead me to imagine the listener as the rider in the center of a ring of speakers providing two layers of sound: the concrète wind and engine recordings outside the helmet, and the voices of thought inside the helmet. The idea of literal high-speed acceleration to modulate field recordings would also manifest itself virtually through custom spatialization rendering software that could pan sounds at audio rates (above 20Hz). I found that correlating the literal speeds achieved during the field recordings to the speeds of trajectory spatialization for the voices provided a connection between the two layers.

3.1.2 Recording Techniques

I have somewhat jokingly referred to *W.A.N.T.S.* as an 8 channel piece for soprano and motorcyclist because the production involved several high-speed motorcycle rides while capturing wind and engine recordings via stereo in-ear microphones. These recording sessions were captured by driving at accelerating speeds, near mountains, in the open, around corners, ducked under the windscreen, and completely exposed to wind. On the other hand, vocals for the abstract layer were captured from a trained opera singer within a completely controlled, noise isolated recording studio. She was told to sing, read, and whisper the phrase "we are not the same" along with the individual words of the phrase in English and French. I did not direct the key of singing, but simply gave instructions as to whether she should sing higher, lower, louder, quieter, and with more or less vibrato.

I also specified whether I would like a more varying and dramatic phrase, or a more monotone reading.

Once all sounds had been collected, the next task was to edit the best takes and create a scheme of organization. I began by cutting up the field recordings and vocal samples in Ableton Live. No warping or sonic manipulation of any kind was done the recordings before spatialization. I simply divided the rides and words into different categories based on speed, acceleration, deceleration, location, language, and word. This process produced a number of audio files that I would feed into a custom spatialization script for final processing.

3.1.3 Spatialization Technique

Given the 8 channel ring of speakers on which I was tasked to realize the piece, I decided to begin experimenting with simple circular trajectories of sound. While a variety of commercial digital audio workstation software did support 8 channels in the form of 7.1 surround, no software supported a simple octaphonic ring layout, much less any custom, arbitrary speaker layout. So, the initial, seemingly simple, idea of sound circles was not simple to realize using existing commercial software. Even given a 7.1 panner with automation, it would be extremely tedious to realize a sound that circles quickly - hundreds or thousands of times within a minute.

Coming from a signal processing background, I was comfortable with MATLAB[2] and decided that I could write a simple script to pan sound around a circular array of speakers. Once I had the script rendering circles around the speakers, it became trivial to experiment with increasing the speed of panning and to add a changing radial distance to create spiral sound trajectories. Finally, I had the script simulating Fibonacci spirals of sound moving around the listener (Figure 3.1).

[2] `http://www.mathworks.com/products/matlab/`

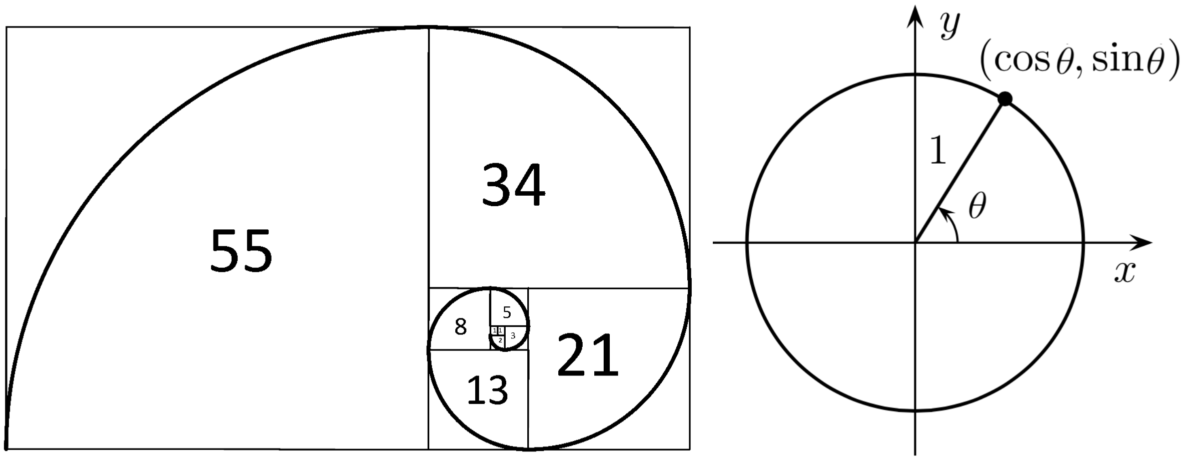


Figure 3.1: Fibonacci Spiral (Left) and Unit Circle (Right) [3]

Fibonacci spirals of sound are circles with a continuously decreasing radius. For each 90 degree arc, the radius of the spiral decreases from one Fibonacci number to the previous. Thus, 4 times the number of complete 360 degree spiral rotations gave the index of the highest Fibonacci number needed to construct the spiral. For example, a 720 degree (twice around) spiral would require $2 * 4 = 8$ Fibonacci numbers. However, since the script divides the gain of the sound by the radius, the first Fibonacci number, 0, would result in infinite gain at the center of the spiral. So, the script ceases when the radius is 1.

The MATLAB script assumes eight speakers are placed equidistantly on a unit circle (Figure 3.1). The script takes as input the number of spirals or circles the sound should complete over the course of its duration and computes an angle, θ, for each sample of the sound. Using the angle and radial distance of the sound, the script computes the distance to the nearest speaker location. Then, a gain reduction multiplier is computed as $1 - 0.5 * distance$. So, for a sound at the opposite end of the circle from the speaker ($distance = 2$), the multiplier is 0, and for a sound at the speaker location ($distance = 0$),

[3] `https://en.wikipedia.org/wiki/Golden_spiral` and `https://en.wikipedia.org/wiki/Unit_circle`

the multiplier is 1.

Running the MATLAB script for each input audio file resulted in 8 new mono sound files, one for each channel. The spatialized sound files were imported into Ableton Live's multi-tracking view. For example, the first section of the piece involves 2 Fibonacci spirals and 4 circles simultaneously, for a total of $2 * 8 + 4 * 8 = 48$ tracks. Configuring a single slider on a midi controller to simultaneously control all 8 channels of a particular sound allowed me to mix the spirals and circles as if they were single tracks. The mixes were then recorded down to a final octaphonic version as well as a stereo version. See Appendex C.1 for the complete MATLAB spatialization script used in *W.A.N.T.S.*.

3.1.4 Results

In *W.A.N.T.S.* I found that I was able to modify the timbre of the vocal recordings through spatialization alone due to the amplitude modulation produced from rapidly spiraling sounds. Higher speeds correlated to higher frequencies of modulation, and larger spirals produced a greater index of modulation due to increasing amplitude attenuation with distance. Spatialization was key to creating effects in which acceleration was the correlating parameter between the physical layer of sound (recorded at specific speeds) and the vocal sounds inside the rider's head (simulated at specific speeds). The spirals of voices were programmed to move at rates varying from about 2 rotations per minute to 7000 rotations per minute, which were roughly correlated to match the RPM of the motorcycle's engine during accelerations. The AM and tremolo effects of the resulting from the high-speed spatialization often complemented the hum of the motorcycle engine recordings, but it was by experimenting with extreme speeds that I first noticed the opportunity for a sort of spatial synthesis - or the ability to produce timbral changes from spatialization.

The MATLAB spatialization script did not include Doppler shift, so the timbral effects from spatialization were completely based on amplitude modulation. My later work would include Doppler shift for not only a more physically accurate representation of sound source movement, but also as a way to create frequency modulation effects from high speed spatialization. The methods by which *W.A.N.T.S.* was created were cumbersome, so I began to imagine a piece of software that would allow for more flexible, real-time spatialization with Doppler shift. Though processing was slow and tedious, the MATLAB spatialization script did allow for precise, sample-rate control of a complex mathematical sound trajectory - another notion that would tie into my later work.

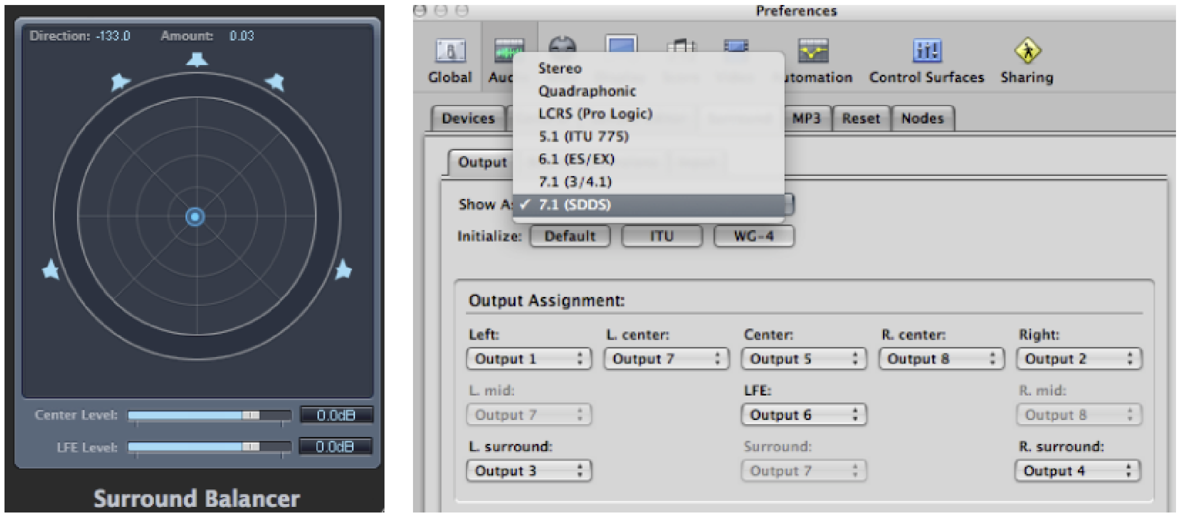
3.2 Sound Element Spatializer (2010)

Sound Element Spatializer (SES) is a sound spatialization server that easily integrates with any existing audio application and scales to support an arbitrary number of moving sound sources over an arbitrary loudspeaker configuration. SES used multiple distance cues in including Doppler shift and provides dynamic selection of multiple spatialization rendering techniques including Vector Base Amplitude Panning (VBAP) [13], Distance Based Amplitude Panning (DBAP) [14], and Higher Order Ambisonics (HOA) [15]. Control of sound trajectories is not limited to a proprietary GUI, but instead accomplished via Open Sound Control (OSC) [16, 17], making it possible to create any number of spatialization controllers for specific applications. Audio is routed into SES via live inputs or a virtual audio device that can process output from individual DAW tracks in real-time. SES also includes features for decorrelated upmixing, making it possible to create multiple, independent sound sources from a single input to produce creative spatial effects.

SES was published in the Proceedings of the 2011 International Computer Music Conference [18] and used for several other sound works, including a 3-month spatial light and sound installation at the Los Angeles Architecture and Design Museum (Section 3.3).

3.2.1 Digital Audio Workstation Limitations

Popular digital audio workstation (DAW) software packages such as Logic, ProTools, Live, Digital Performer, etc., though useful for many aspects of layering and mixing sound, 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 [19]. 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. Figure 3.2 shows the surround panning interface of Logic Pro, which is typical of most DAWs - the formats and their speaker layouts are fixed, and sources must be controlled individually.



Direction: -133.0 Amount: 0.03

Center Level: 0.0dB

LFE Level: 0.0dB

Show As: ✓ 7.1 (SDDS)

Initialize: Default ITU WG-4

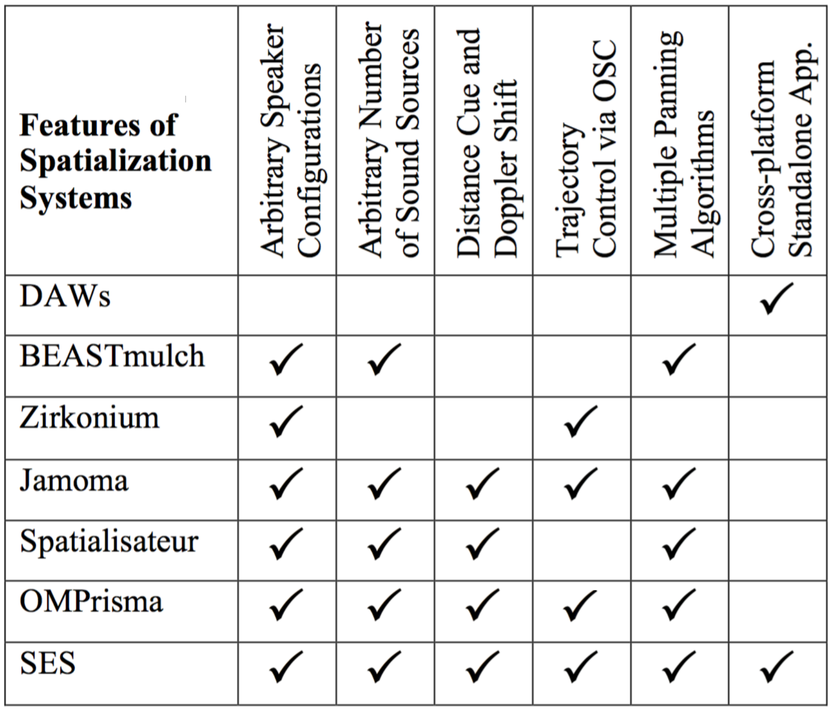
Output Assignment:

Left:	L. center:	Center:	R. center:	Right:
Output 1	Output 7	Output 5	Output 8	Output 2
L. mid:		LFE:		R. mid:
Output 7		Output 6		Output 8
L. surround:		Surround:		R. surround:
Output 3		Output 7		Output 4

Figure 3.2: Typical DAW Surround Mixing Interface

3.2.2 Related Work

Outside the world of DAWs, there are several spatial audio platforms including BEASTMulch [20], Zirkonium [21], Jamoma [19], Spatialisateur [22], and OMPrisma [23]. 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 3.1.



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

Table 3.1: Comparison of Current Spatialization Systems

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. 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.

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. 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.

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 [24] 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. 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 [1]. 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.

OMPrisma

Omprisma is a comprehensive spatialization suite implemented in the Open Music language [25]. While it implements nearly every desired feature of SES, there is significant overhead in ease of use for the computer music layman due to the dependency on Open Music. SES aims to make such spatialization abilities easy to access through a simple, organized GUI within a standalone application.

Custom Max/MSP and PureData

It is also common to use graphical programming environments such as PureData [4] and Max/MSP [5] to create custom spatial audio patches that support custom speaker configurations and OSC messaging. However, these environments do not scale easily and become cumbersome to manage when patches become too complicated. Early prototypes for SES were constructed in PureData (Figure 3.3) and Max/MSP, but as features grew it became necessary to use a a true programming language to create an organized application rather than a complicated patch.

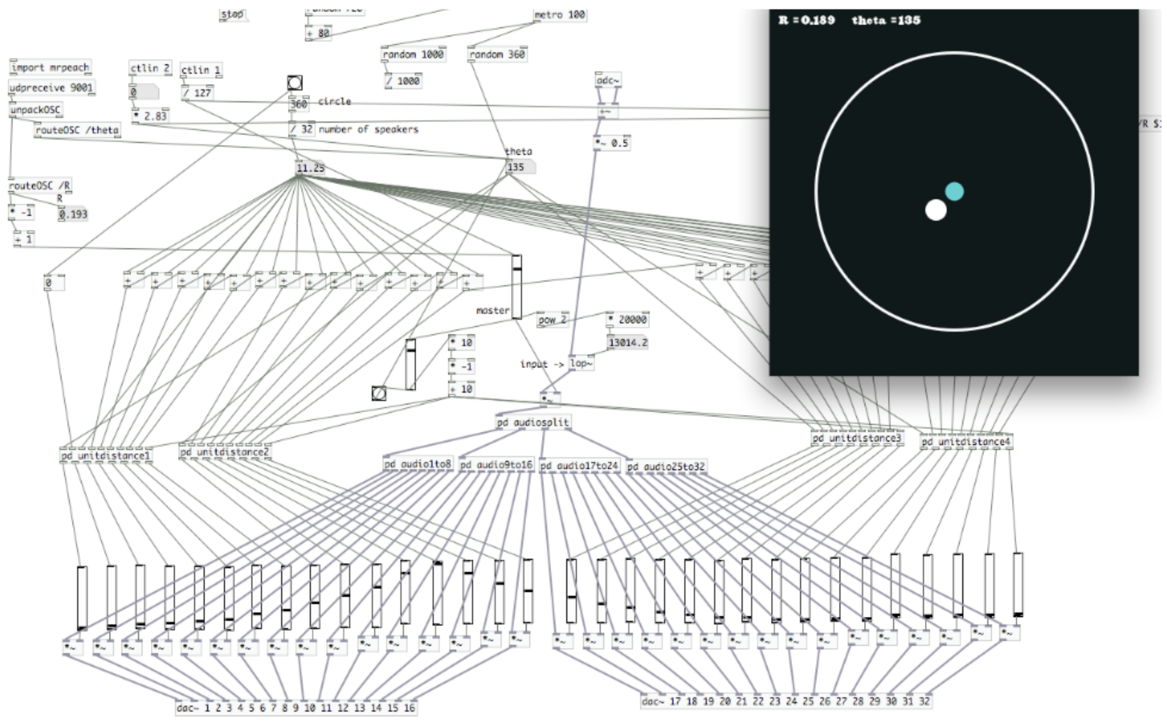


Figure 3.3: SES Prototype Spatializer in PureData with OSC Controller

[4]https://puredata.info
[5]https://cycling74.com/products/max/

3.2.3 Goals

SES had a list of predefined goals based on the features lacking in DAWs and the desires arising after creating *W.A.N.T.S* (Section 3.1):

- 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 without "zipper" noise artifacts.
- Flexible, precise control of sound trajectories using the OSC protocol, which allows for the creation of custom trajectory controllers.
- Robust and easy to integrate with DAW software or any audio application
- Cross-platform standalone application in C++. (Easy to use, no complicated Pure-Data or Max/MSP programming)
- Provide a link between visual and sonic worlds using OSC. SES should encourage collaboration and exploration for both visual and sound artists with spatial coordinates as unifying, shared parameters between graphics and audio.

3.2.4 The Sound Element Spatializer Paradigm

SES positions an arbitrary number of sound elements in real-time over an arbitrary speaker arrangement via VBAP, DBAP, or HOA with distance cues including Doppler shift, air absorption, and gain attenuation. Smooth position interpolation supports high-speed movement of sound sources without zipper-noise. SES is a cross-platform, standalone GUI application that easily integrates with any audio application exposes all controls via OSC. Novel features include "derived sources" for decorrelated upmixing and a built-in optional "flutter" effect to increase spatial localization. Figure 3.4 provides an overview of the entire software.

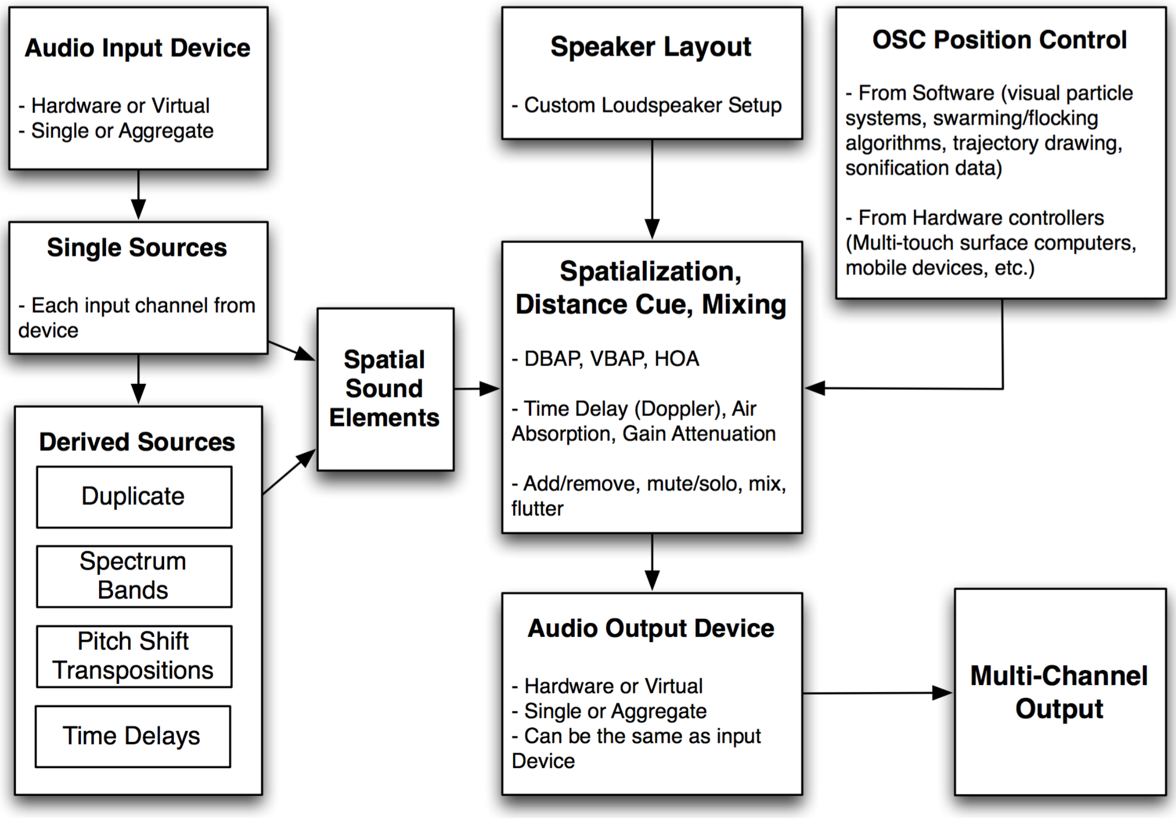

Audio Input Device
- Hardware or Virtual
- Single or Aggregate
Single Sources
- Each input channel from device
Derived Sources
Duplicate
Spectrum Bands
Pitch Shift Transpositions
Time Delays
Spatial Sound Elements
Speaker Layout
- Custom Loudspeaker Setup
OSC Position Control
- From Software (visual particle systems, swarming/flocking algorithms, trajectory drawing, sonification data)
- From Hardware controllers (Multi-touch surface computers, mobile devices, etc.)
Spatialization, Distance Cue, Mixing
- DBAP, VBAP, HOA
- Time Delay (Doppler), Air Absorption, Gain Attenuation
- Add/remove, mute/solo, mix, flutter
Audio Output Device
- Hardware or Virtual
- Single or Aggregate
- Can be the same as input Device
Multi-Channel Output

Figure 3.4: Sound Element Spatializer Overview

Spatial Upmixing

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) [26]. The simplest method, *duplication*, copies a given audio input to any number of individually spatializable sources; this obviously does not provide decorrelation on its own 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.

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.

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. Often a separate sound source application is run (DAW, granulator[6], etc.) on the same computer and its audio outputs are connected to SES's audio inputs using a virtual audio device such as SoundFlower[7] or Jack[8]. An OSC trajectory control application is simultaneously run 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 3.5, while a distributed spatialization rendering setup using two machines is shown in Figure 3.6.

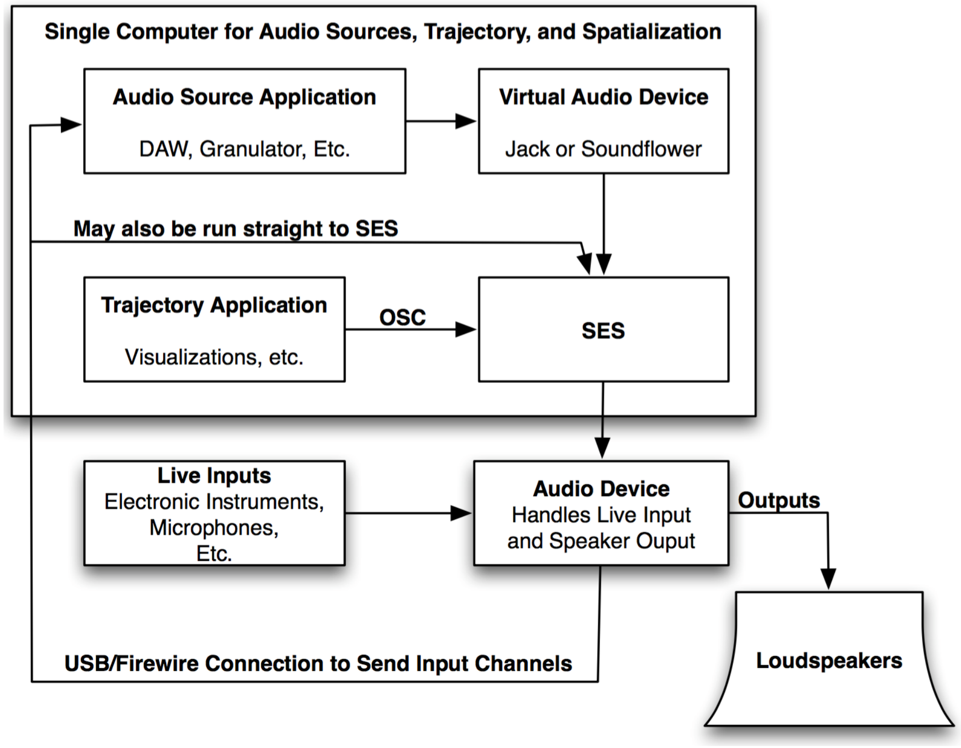

Single Computer for Audio Sources, Trajectory, and Spatialization
Audio Source Application
DAW, Granulator, Etc.
Virtual Audio Device
Jack or Soundflower
May also be run straight to SES
Trajectory Application
Visualizations, etc.
OSC
SES
Live Inputs
Electronic Instruments, Microphones, Etc.
Audio Device
Handles Live Input and Speaker Ouput
Outputs
Loudspeakers
USB/Firewire Connection to Send Input Channels

Figure 3.5: SES Running on a Single Machine

[6]Granular synthesis [27] 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 [28]. 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.

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

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

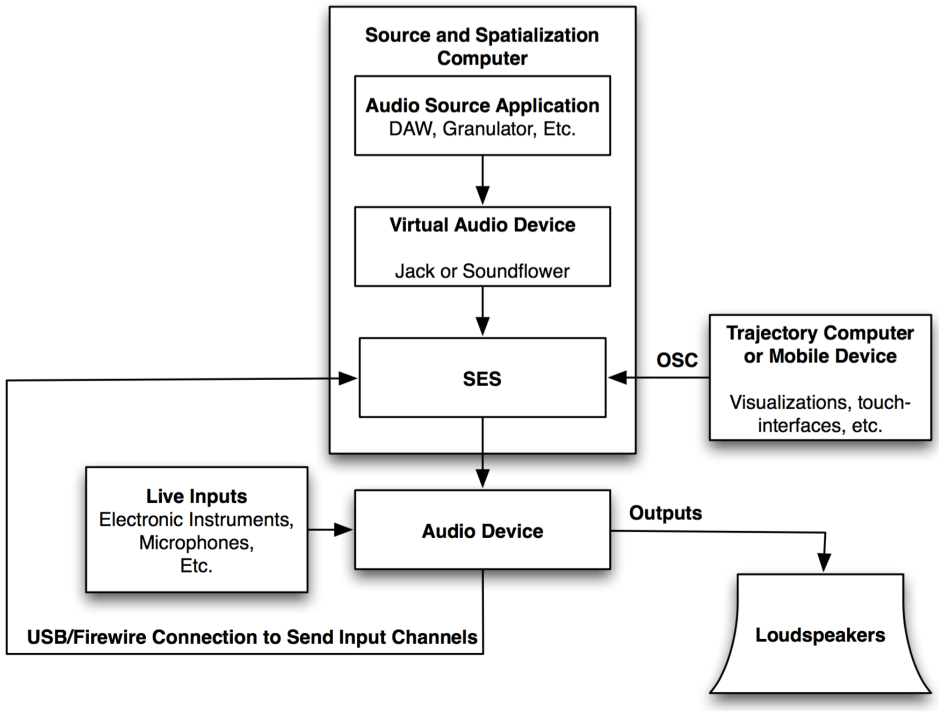
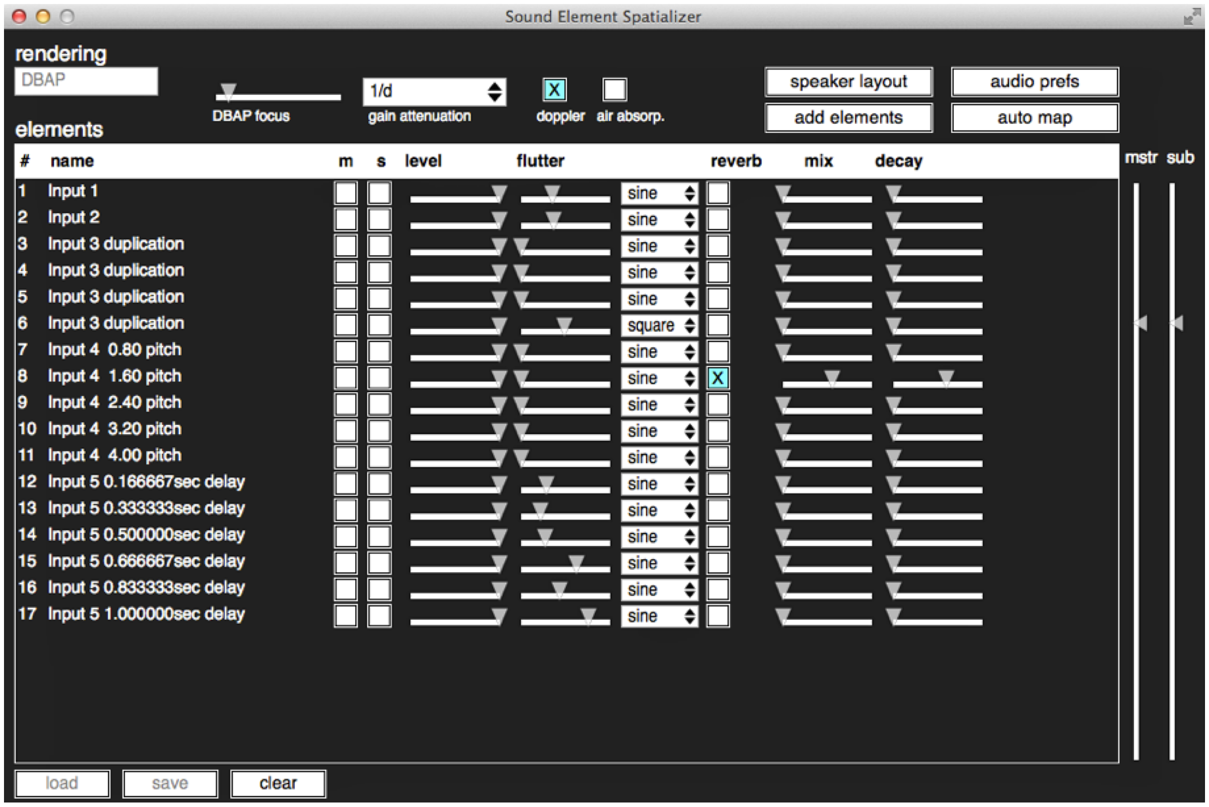

Source and Spatialization Computer
Audio Source Application
DAW, Granulator, Etc.
Virtual Audio Device
Jack or Soundflower
SES
OSC
Trajectory Computer or Mobile Device
Visualizations, touch-interfaces, etc.
Live Inputs
Electronic Instruments, Microphones, Etc.
Audio Device
Outputs
Loudspeakers
USB/Firewire Connection to Send Input Channels

Figure 3.6: SES Running with Separate Audio and Trajectory Machines

Graphical User Interface

The main window (Figure 3.7) shows a table of all sound elements with mute, solo, level, and flutter controls. The “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 3.8) allows users to map an audio input channel to a single element or to many derived sources as described in Section 3.2.4, as well as to name the new element(s). Finally, the simple speaker layout editor (Figure 3.8) 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.



Sound Element Spatializer

rendering

DBAP

DBAP focus

1/d gain attenuation

[✓ doppler] air absorp.

speaker layout | audio prefs | add elements | auto map

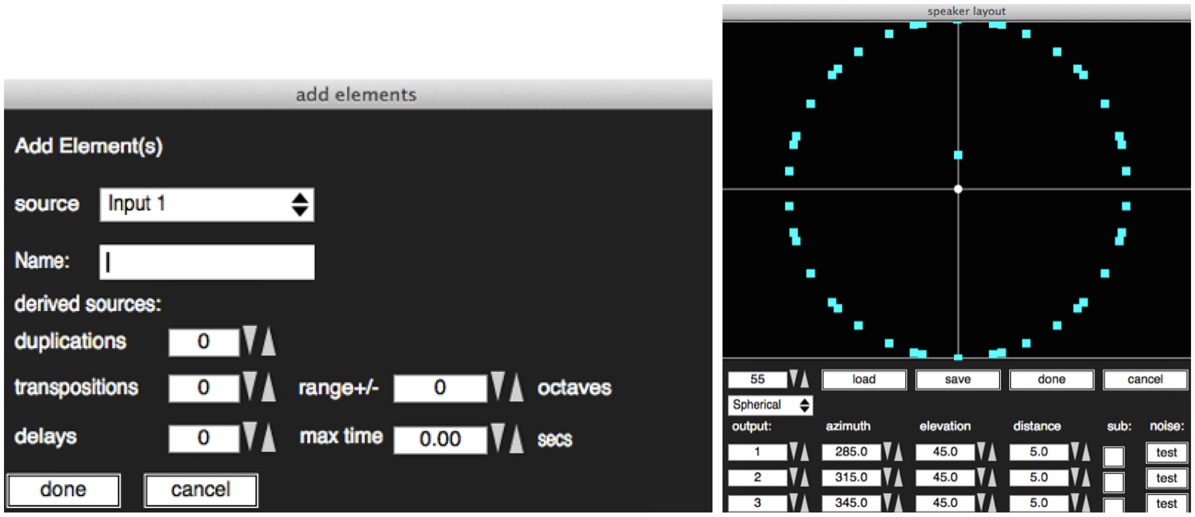
elements

#	name	m	s	level	flutter		reverb	mix	decay
1	Input 1					sine			
2	Input 2					sine			
3	Input 3 duplication					sine			
4	Input 3 duplication					sine			
5	Input 3 duplication					sine			
6	Input 3 duplication					square			
7	Input 4 0.80 pitch					sine			
8	Input 4 1.60 pitch					sine	X		
9	Input 4 2.40 pitch					sine			
10	Input 4 3.20 pitch					sine			
11	Input 4 4.00 pitch					sine			
12	Input 5 0.166667sec delay					sine			
13	Input 5 0.333333sec delay					sine			
14	Input 5 0.500000sec delay					sine			
15	Input 5 0.666667sec delay					sine			
16	Input 5 0.833333sec delay					sine			
17	Input 5 1.000000sec delay					sine			

mstr sub

load save clear

Figure 3.7: SES Main Window



add elements

Add Element(s)

source: Input 1

Name:

derived sources:

duplications 0

transpositions 0 range+/- 0 octaves

delays 0 max time 0.00 secs

done cancel

speaker layout

55 load save done cancel

Spherical

output:	azimuth	elevation	distance	sub:	noise:
1	285.0	45.0	5.0		test
2	315.0	45.0	5.0		test
3	345.0	45.0	5.0		test

Figure 3.8: SES Add Elements Window (Left) and Speaker Layout Editor (Right)

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 controllers. 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 [24] OSC format.

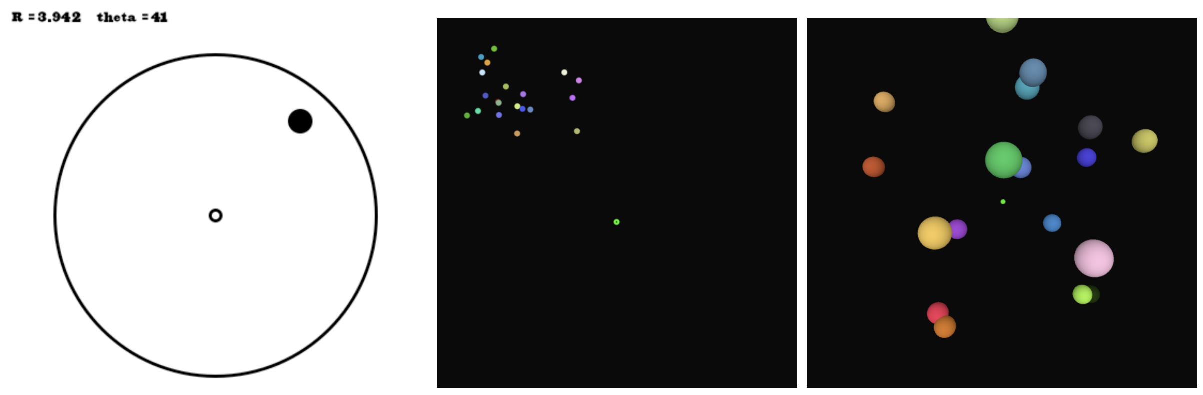


Figure 3.9: Simple 2D Controller (Left), Multi-Element 2D Controller (Middle), and Multi-Element 3D Controller (Right)

3.2.5 Implementation

SES was written in C++ using the CREATE Signal Library[9] (CSL) for high-level audio processing, JUCE (Jules' Utility Class Extensions)[10] for the graphical user interface and low-level audio processing, the SoundTouch library[11] for real-time pitch shifting, and the Lightweight OSC library (liblo)[12] to receive OSC messages.

Many features were added to CSL's existing basic framework for spatial audio panning, including a new DBAP panning class and continuous interpolation of source positions to avoid "zipper" effects when moving sources at high speeds.

3.2.6 Results

SES provided means for flexible, precise control of spatialization for an arbitrary number of sound sources over an arbitrary speaker layout and was the first native application available to allow dynamic switching between DBAP, VBAP, and HOA. Much work was done to remove common limitations found in other similar systems, such as smooth interpolation of high-speed source movement without "zipper" noise artifacts. SES's novel derived sources and flutter features allowed for experimentation with clusters of moving sound sources as used in *No Heritage* (Section 3.6).

Equipped with a Doppler simulation, I began to formulate the idea of Doppler FM - using precise trajectories to produce FM timbres via high-speed Doppler shift. Though SES's interpolation would allow for smooth panning and Doppler at high-speeds, I would eventually discover that the results would not be physically accurate at high speeds due to buffer-rate interpolation of OSC trajectories. Doppler FM is fully realized in Spatial Modulation Synthesis (Chapter 4) and requires sample-rate trajectory control.

[9]http://fastlabinc.com/CSL/
[10]http://www.rawmaterialsoftware.com/juce.php
[11]http://www.surina.net/soundtouch/
[12]http://liblo.sourceforge.net/

3.3 *Skate 1.0 (2011)*

Skate 1.0 is an abstract virtual skateboard park manifested through an immersive light and sound installation using 60 fluorescent lights and a 12.2-channel sound system. Skater sounds travel around and through the viewer while the lights register their movement and intensity on the space and on viewer's bodies. *Skate 1.0* was exhibited at the Los Angeles Architecture and Design Museum from July to September, 2011. A video of the installation is online at `https://vimeo.com/26356626`.

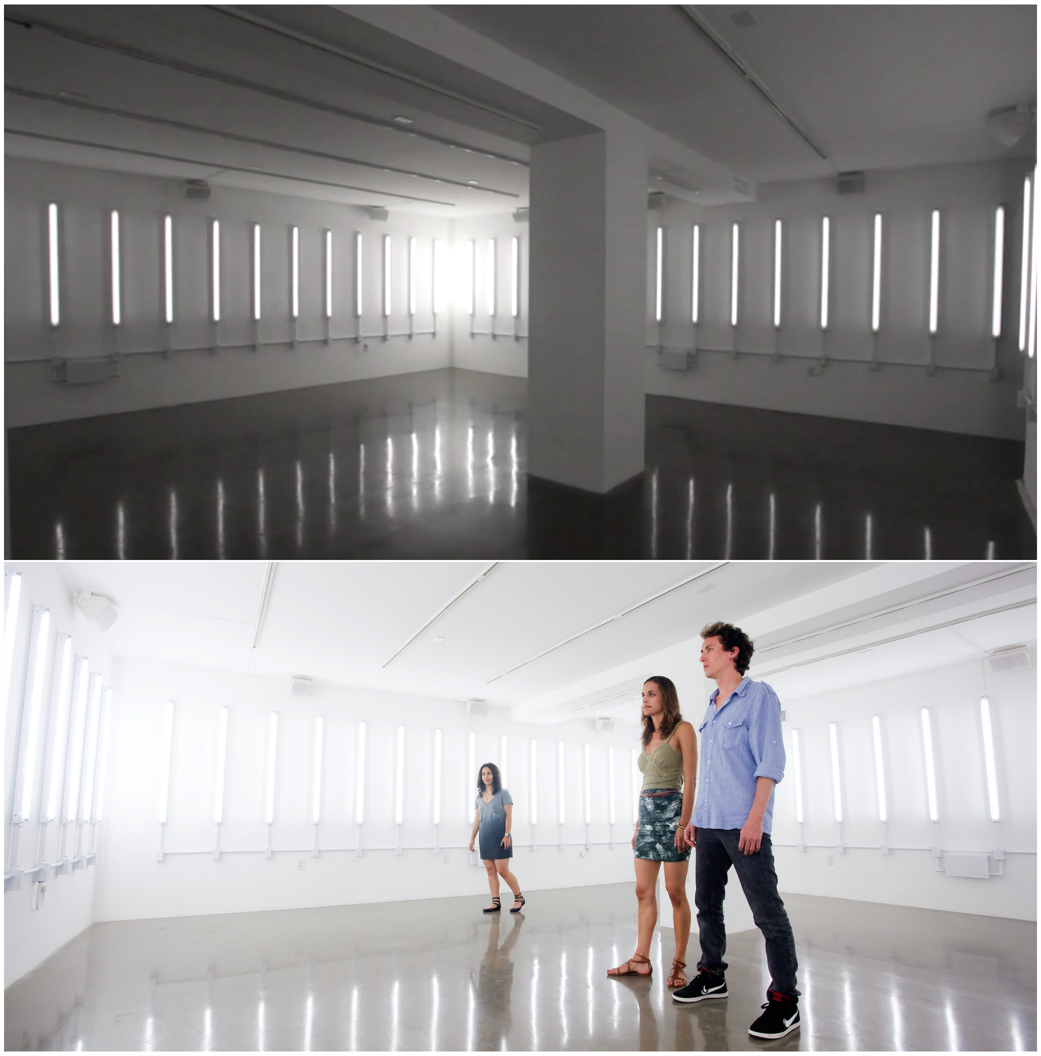


Figure 3.10: Skate 1.0 Installation

3.3.1 Spatialization Control

Skate 1.0 used Sound Element Spatializer (Section 3.2) to synchronize moving sound sources with moving light sources. The agency, Electroland[13], provided the lighting hardware and custom lighting control software that could control the individual brightness of 60 fluorescent lights. A number of moving light sequences were preprogrammed, and the role of SES was to use the moving light trajectories as moving sound trajectories. Through the simple definition of a shared audio-visual spatial coordinate system Electroland was able to quickly send OSC sound trajectory messages from their software to SES, which was configured to accommodate the 12.2 speaker layout for the venue.

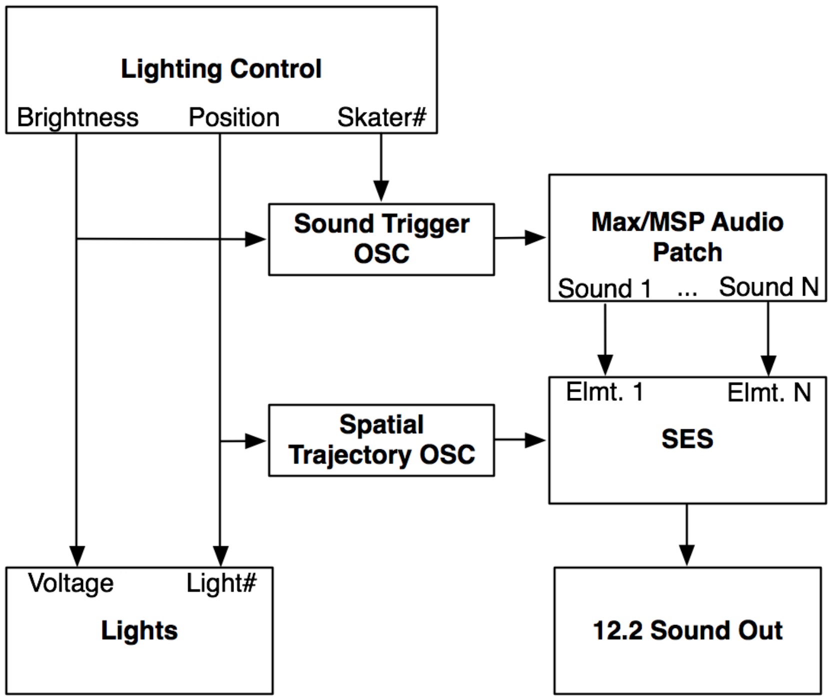


Figure 3.11: *Skate 1.0* Lighting Controlled Sound Spatialization

Several overlapping trajectories played simultaneously to represent several skaters moving in space. Each skater was sonically represented by an independently controlled

[13]http://www.electroland.net

spatial sound element in SES. The sound source for each element was a mono field recording of skateboard sounds sent to an individual virtual audio output using Soundflower. SES easily mapped separate audio inputs from Soundflower to individual sound sources for spatialization.

3.3.2 Results

Initially, all moving light and sound sources were programmed to traverse the space at realistic skateboarding speeds (around 5 meters per second). However, we experimented with increasing speeds with and without Doppler simulation to create the final sequences. As with *W.A.N.T.S.* (Chapter 3.1), timbral changes from high-speed spatialization were observed and used as effects. SES ran continuously for the 3-month duration of the installation with no crashes.

3.4 SenSynth (2011)

SenSynth is a mobile application that allows for arbitrary, dynamic mapping between several sensors and sound synthesis parameters. In addition to parameter mapping sonification using synthesis techniques commonly found on mobile devices, a scanned synthesis source for the audification of sensor data was implemented. A novel instrument based on the audification of accelerometer data is presented along with a new means of mobile synthesis control via a wearable magnetic ring. SenSynth was published in the proceedings of the 2012 New Interfaces for Music Expression conference [29].

3.4.1 Background

In 2002 Hunt stated that an electronic instrument was more than just an interface coupled with a sound generator, and referred to the mapping between interface and sound as the "essence" of an instrument [30]. With today's mobile devices we see a plethora of sensors available for interaction coupled with powerful processors that can rival the sound synthesis capabilities of the laptop. Yet, we often only see instruments exploiting either the sensor or synthesis potential of these devices. For example, applications like Cosmovox[14] and AirVox[15] make innovative use of the camera and sensors on a phone, but are rather limited in synthesis abilities and do not allow one to reconfigure the sensor-to-sound mapping. Users of these applications are not free to explore the "essence" of other mappings. While commercial mobile applications such as Synthstation[16] and Nanostudio[17] bring powerful synthesis to mobile phones, they still rely on interaction via touch screen or external MIDI adapter and fail to make use of the many available mobile

[14]http://leisuresonic.com/cosmovox/
[15]http://www.yonac.com/AirVox/
[16]http://www.akaipro.com/synthstation
[17]http://www.blipinteractive.co.uk/

sensors on modern phones such as the accelerometer, gyroscope, and magnetometer.

Also in 2002, Wessel and Wright suggested a NIME design criteria with "initial ease of use coupled with a long term potential for virtuosity. . . and clear and simple strategies for programming the relationship between gesture and musical result" [31]. Many existing mobile instruments such as the Ocarina[18] take the "hyperinstrument" approach to accomplishing initial ease of use and potential for virtuosity by modeling traditional instruments. Overholt and Roads suggested an alternate approach of designing around modern synthesis techniques rather than creating instruments with familiar interfaces – "the mimicking of these older designs does not take into account the predilection for the electronic composer to work inside the sounds." [32] Likewise, the emphasis with SenSynth was to allow the user to easily create sensor-to-sound mappings rather than to mimic interactions with traditional instruments. Overholt and Roads state that new instruments will require a "new type of virtuosity" from an "interdisciplinary composer/programmer" [32]. However, initial ease of use is still considered when designing new interactions with SenSynth, which results in a simple mapping interface, the inclusion of preset instruments, and an optional pitch quantizer to ease the playing of notes in a musical scale using various sensors.

Another category of applications includes RjDj[19], Inception[20], and ZooZBeat[21]. These augmented music applications do bring an innovative use of onboard sensors as interfaces to interact with songs or preset patches, but overall do not stand as instruments themselves. Pre-smartphone systems such as Sonic City [33] and PuDAC [34] synthesized and composed musical experiences using a number of wearable sensors as a user walked through a city or engaged in exercise. Both projects viewed themselves as augmented

[18]`http://ocarina.smule.com/`
[19]`http://rjdj.me/`
[20]`http://inceptiontheapp.com/`
[21]`http://www.zoozmobile.com/zoozbeat.htm`

musical experiences rather than instruments for musical performance. One goal of SenSynth was to create an interface that is able to stand as both a performance instrument and augmented music application. A key feature to realize this versatility, which all other aforementioned applications lack, is the ability to quickly and dynamically create custom sensor to sound mappings. Without this ability to remap, performers are constrained to a fixed form of interaction and the new musical interface becomes no more versatile than a traditional musical instrument. Likewise, listeners of augmented musical experiences should be able to create their own interactions instead of downloading other's presets.

The Speedial [35], urMus [36], and Momu [37] projects are probably the most similar to SenSynth in terms of ideology. Momu is a programming toolkit to aid in the rapid mapping between mobile sensors and audio. Speedial and urMus are mobile applications that allow for rapid sensor to sound mapping in a manner very similar to SenSynth. However, SenSynth provides a GUI front end to all instrument parameters so that sliders and buttons may be used to control parameters that are not mapped to any sensor. urMus allows the scripting of GUI's that may be uploaded to the phone, but SenSynth's approach augments a variety of provided synthesis interfaces with customizable sensor control. We believe providing initial synthesis interfaces from which to map lowers the barrier of entry for the electronic music layman to create custom instruments.

SenSynth is a mobile interface with an emphasis on arbitrary mapping between an array of sensor inputs and several sound synthesis techniques. There was no aim to model any familiar modes of interaction nor an attempt to define any modes of interaction for existing synthesis techniques - only an aim to exploit every sensor to sound mapping possibility on a modern mobile device. While included preset modes of interaction and a global pitch quantizer provide instant usability, users desiring more control are free to design their own instruments or abandon the automatic tuning. SenSynth redefines itself as an instrument through user-created mappings and compositions created by the user's motions.

3.4.2 SenSynth Interface

SenSynth allows the user to arbitrarily map a mobile phone's sensor output values to parameters for sound synthesis and effects. For example, the accelerometer might be connected to control the modulation depth of a FM synthesizer, so that when when the user shakes the phone, more partials appear in the FM sound. SenSynth is a mobile platform for sonic experimentation and the results of such mappings may be used equally for musical performance and sonification. The ease of use of the GUI enables rapid experimentation with both the mappings and the instrument itself. A feature completely unique to SenSynth is that every onscreen button or slider can be controlled via any sensor on the phone.

GUI

SenSynth initially presents the user with a simple mixer page to activate and adjust the gain of each available sound source. Each source has its own page in which all parameters can be adjusted via touch sliders and buttons. There is also an FX page with on/off switches and the appropriate sliders for each audio effect. Figure 3.12 shows the main menu and Figure 3.13 shows the Oscillators page as an example of the variety of parameters available for mapping.

The mapping page allows the user to dynamically map any sensor on the phone to control any source or effect parameter. The user is presented with a list of all available sensors and once a selection is made all available sources and effects are displayed. After making a source or effect selection, a list of parameters that can be controlled by the chosen sensor are displayed. While mappings are arbitrary for the most part, there are

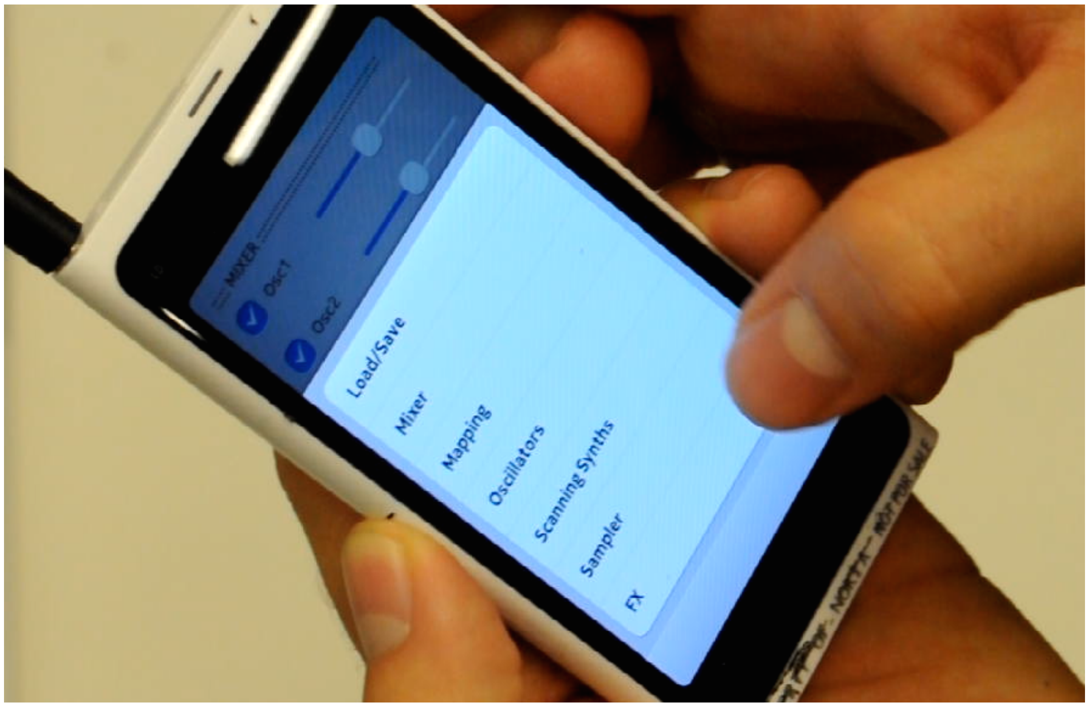


Figure 3.12: SenSynth Main Menu

some limitations. For example, the proximity sensor can only output a boolean on/off, so it is not able to control the frequency of an oscillator. A single sensor can control any number of sound parameters for a one-to-many mapping. If a user tries to assign multiple sensors to a single parameter the last sensor will gain all control and the previous mappings to that parameter will be removed. Figure 3.14 shows the selection of a sensor and several completed mappings.

Once a mapping is created it is displayed in a list with two buttons: one to remove the mapping and one to invert the sensor control. For example, if the gyroscope is mapped to control the frequency of an oscillator, tilting the phone upwards (increasing the gyroscope's pitch) will cause an increase in the frequency of an oscillator by default, but applying the invert button will cause a decrease in the oscillator's frequency when the phone is tilted upwards. Series of several mappings can be named and saved as presets on the Load / Save page.

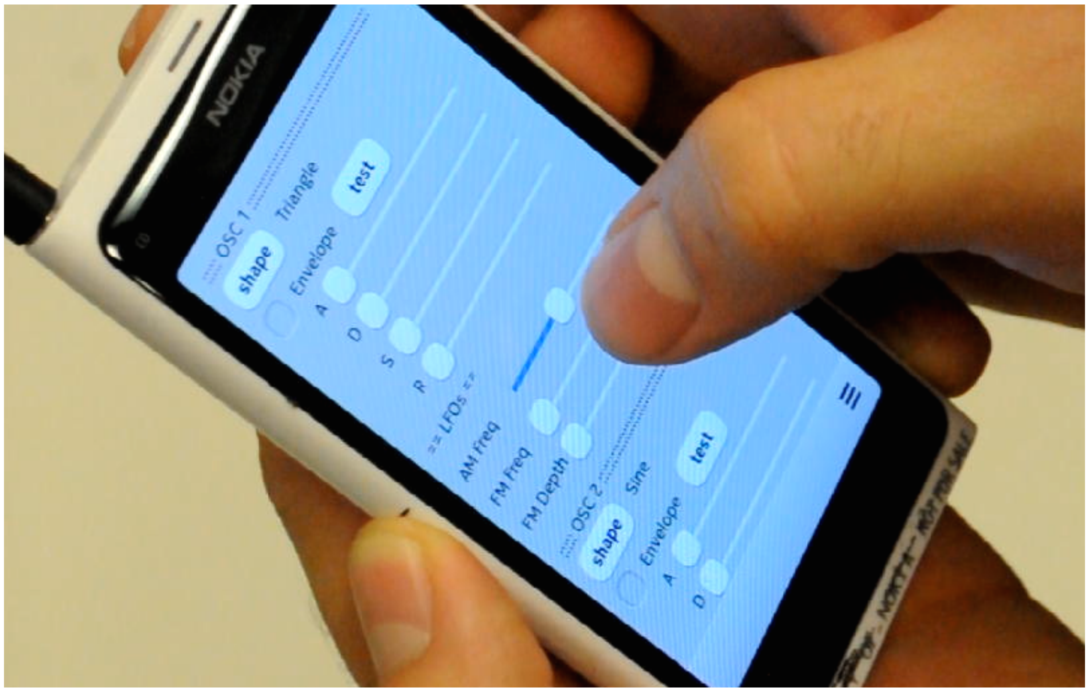
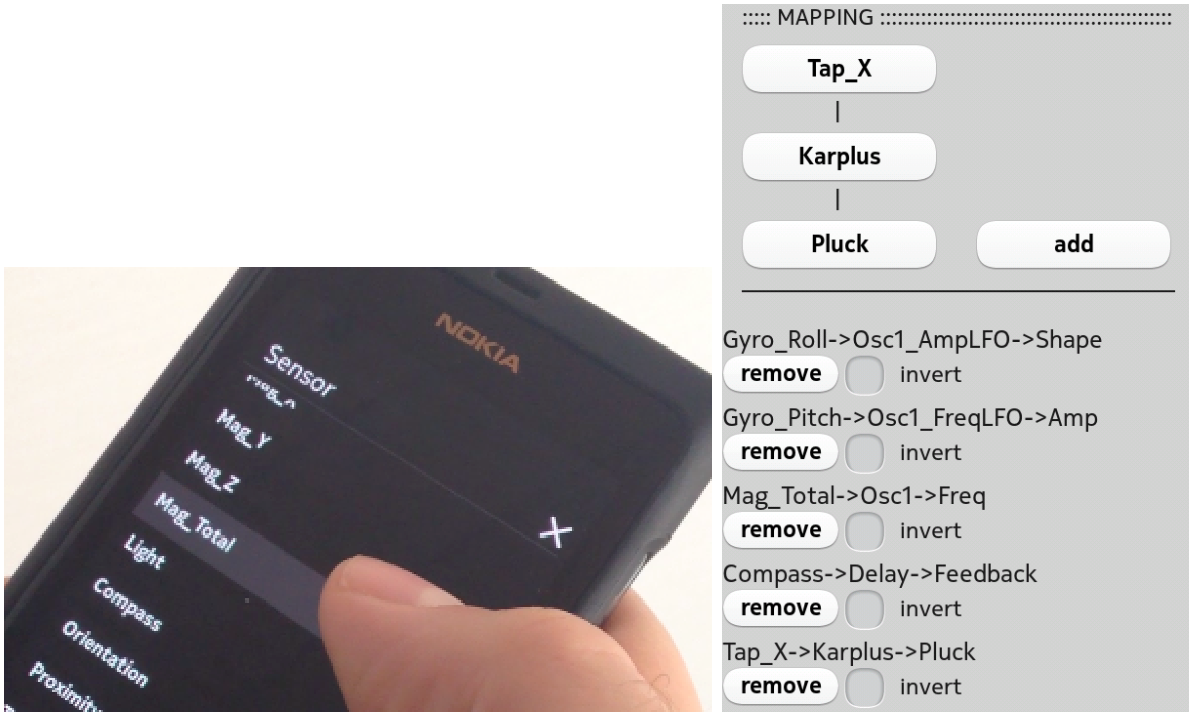


Figure 3.13: SenSynth Oscillators Page

Sensor to Sound Parameter Mapping

Figure 3.15 shows all available sensors and sound parameters as well as their data types. The center boxes indicate which sensor information can control which sound parameters. For example, it is possible for floating-point outputs to control integer and boolean parameters. So, if the gyroscope roll is mapped to control the waveshape of an oscillator the floating-point output will be quantized to 5 values, one for each of the 5 possible waveshapes. As the user twists the phone, the waveshapes will change as the values fall into the quantized ranges. For floating-point to boolean control, "true" is triggered whenever the value surpasses a fixed threshold and outputs "false" otherwise. For instance, one could record into the sampler by twisting the phone one direction and stop recording by twisting the opposite direction.



:::: MAPPING ::

Tap_X | Karplus | Pluck add

Gyro_Roll->Osc1_AmpLFO->Shape
remove invert

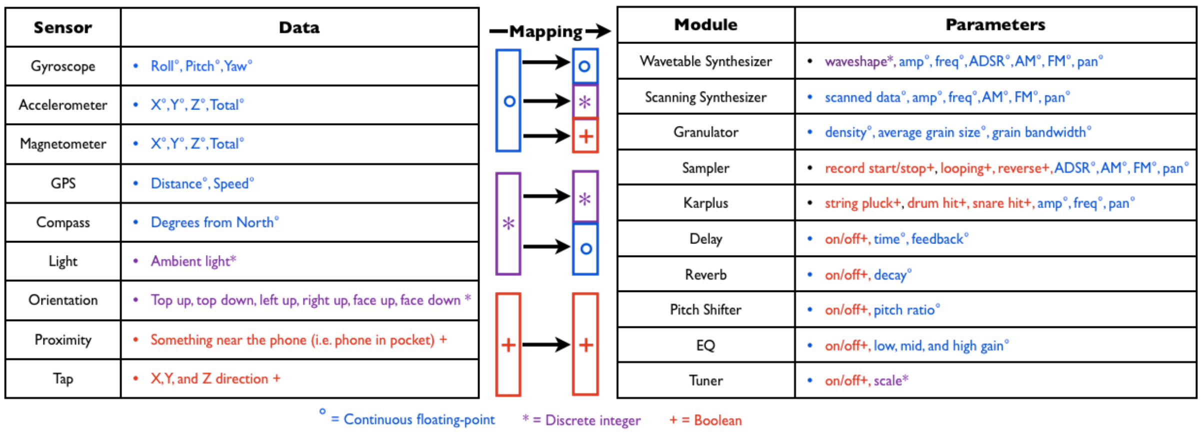
Gyro_Pitch->Osc1_FreqLFO->Amp
remove invert

Mag_Total->Osc1->Freq
remove invert

Compass->Delay->Feedback
remove invert

Tap_X->Karplus->Pluck
remove invert

Figure 3.14: Selecting a Sensor (Left) and an Example Instrument (Right)



Sensor	Data
Gyroscope	• Roll°, Pitch°, Yaw°
Accelerometer	• X°, Y°, Z°, Total°
Magnetometer	• X°, Y°, Z°, Total°
GPS	• Distance°, Speed°
Compass	• Degrees from North°
Light	• Ambient light*
Orientation	• Top up, top down, left up, right up, face up, face down *
Proximity	• Something near the phone (i.e. phone in pocket) +
Tap	• X, Y, and Z direction +

Module	Parameters
Wavetable Synthesizer	• waveshape*, amp°, freq°, ADSR°, AM°, FM°, pan°
Scanning Synthesizer	• scanned data°, amp°, freq°, AM°, FM°, pan°
Granulator	• density°, average grain size°, grain bandwidth°
Sampler	• record start/stop+, looping+, reverse+, ADSR°, AM°, FM°, pan°
Karplus	• string pluck+, drum hit+, snare hit+, amp°, freq°, pan°
Delay	• on/off+, time°, feedback°
Reverb	• on/off+, decay°
Pitch Shifter	• on/off+, pitch ratio°
EQ	• on/off+, low, mid, and high gain°
Tuner	• on/off+, scale*

° = Continuous floating-point * = Discrete integer + = Boolean

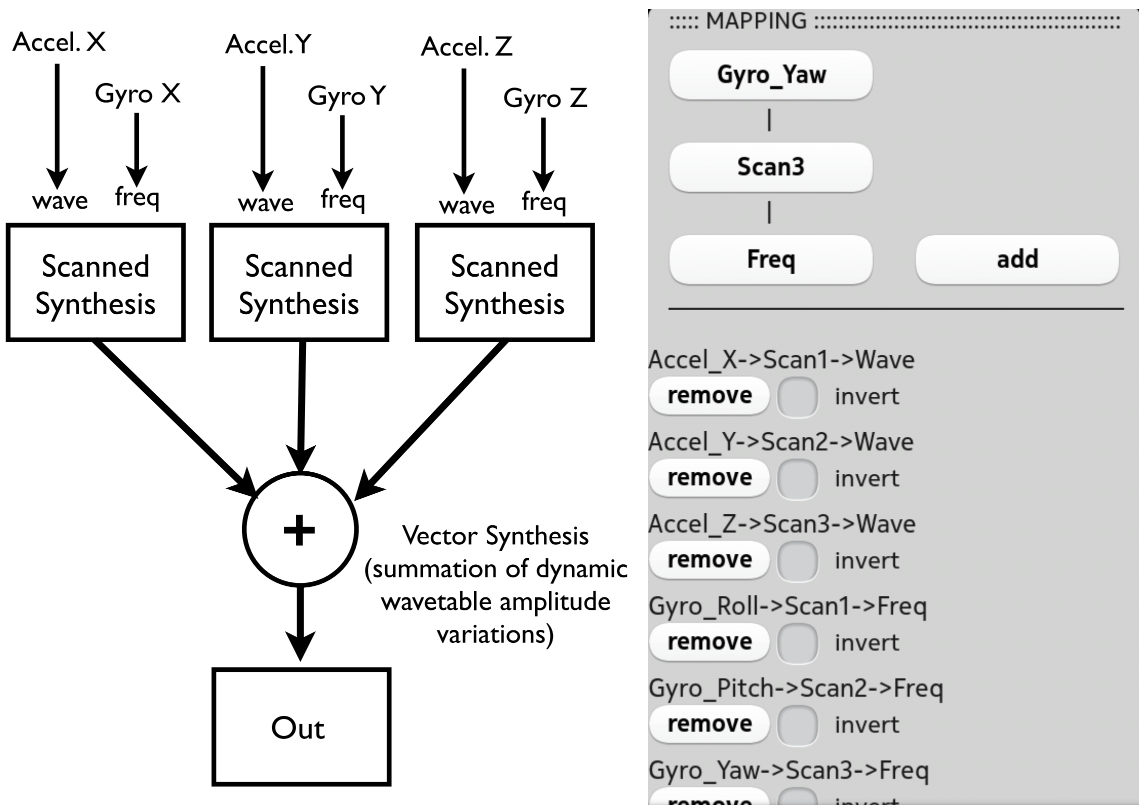
Figure 3.15: Available Sensors, Sound Parameters, and Mapping Options in SenSynth

3.4.3 The Kinetic Synthesizer

In addition to the common oscillators found in other mobile sound synthesis applications, SenSynth includes 3 scanned synthesis sources for the direct sonification of sensor data. Scanned synthesis reads periodic motion-generated data at haptic rates (sub 15Hz) and literally scans the data as an audio waveform at rates fast enough to enter our range of hearing. Thus, one may directly manipulate the spectrum of a sound by human movements [7]. Scanned synthesis may also be classified as a form of audification– the act of reading arbitrary data as an audio waveform [38]. Using scanned synthesis with the accelerometer and gyroscope, a preset instrument called the "kinetic synthesizer" is introduced.

All 3 scanned sources are mapped to read the accelerometer x, y, and z axis respectively. Dynamic amplitude variations within each of the three summed accelerometer waveforms results in vector synthesis, a technique in which multiple wavetables are rapidly crossfaded with each other [39]. Then the gyroscope's roll, pitch, and yaw are mapped to the frequency (directly related to the scan rate) of each scanned source. The resulting combination of motion-produced, scanned vector synthesis has been termed the "kinetic synthesizer" to emphasize the relationship between physical, human motion and sound timbre. Creating this instrument can be accomplished in seconds using SenSynth's mapping interface (Figure 3.16).

The musical pitch of each source is controlled by the gyroscope's 3 dimensions of tilt. With the pitch quantizer active one can easily construct harmonic triads since all frequencies will be quantized to match the nearest note in the selected musical scale. Without the quantizer the gyroscope gives a subtle vibrato effect when trying to hold the phone in a certain plane and shaking it to produce the scanned waveforms, and these effects can be stretched to FM-like sounds with the combination of rapid shaking and tilting of the phone. The experience is reminiscent of Hunt's accidental theremin discovery in which it was determined that necessitating movement to make sound results in a more interesting instrument [30]. Playing the instrument, users immediately become aware of the connection between their physical vibrations and the resulting sound. As O'Modhrain shows [40], such haptic feedback incorporated within a computer-based instrument can improve players' abilities to learn new behaviors.



Accel_X->Scan1->Wave

Accel_Y->Scan2->Wave

Accel_Z->Scan3->Wave

Gyro_Roll->Scan1->Freq

Gyro_Pitch->Scan2->Freq

Gyro_Yaw->Scan3->Freq

Figure 3.16: The Kinetic Synthesizer Abstract (Left) and SenSynth Mapping (Right)

3.4.4 Magnetic Ring Control

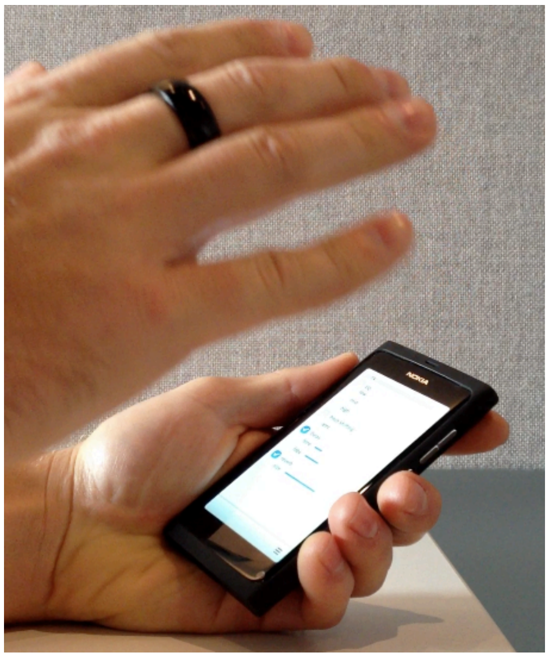


Figure 3.17: Interaction Using the Magnetic Ring

Using a simple neodymium magnet shaped as a wedding ring a high degree of control (down to about 1mm of hand movement) is obtained when mapping the phone's magnetometer to various sound parameters. Such magnetic rings have also been successfully used for selection of audio and visual targets [41]. The initial test with the ring was to use hand motion to control the frequency of an oscillator much like a theremin, but the most powerful results were obtained by using the ring to control expressive parameters such as amplitude modulation (tremolo), panning, granulator density, grain size, EQ levels, sampler playback, delay feedback, and reverberation amount. For example, moving a hand towards the phone could simultaneously increase granulator density, reverberation amount, and cause a note to trigger from one of the oscillators. To produce a greater magnetic effect one may simply attach additional small neodymium magnets to the ring, increasing the magnetic field and allowing one to interact from a greater distance.

3.4.5 Implementation

SenSynth was built for Nokia N9 and 950 models running the MeeGo operating system[22]. Unfortunately, the N9 and MeeGo were never released in the United States and the app was built in collaboration with Nokia Research Hollywood before plans to abandon MeeGo in favor of a Microsoft OS were announced in 2011. Thus, the app never saw a public release. However, it could now be realized on a number of iOS or Android devices. Programming was accomplished in C++ using the QT Quick package and QML. The sound synthesis library used was custom but greatly inspired by the CREATE Signal Library (CSL)[23].

3.4.6 Results

SenSynth provided an open approach to mobile synthesis interaction by allowing for arbitrary mapping of all sensors to sound parameters and was the first mobile app capable of audification of its own sensor data as well as the first app to introduce the idea of 3D control via a wearable magnetic ring. Using the Kinetic Synthesizer in *No Heritage* I again became aware of a connection between spatial motion and sound timbre as I had with the projects in previous sections of this chapter. However, SenSynth dealt with the generation of sounds through audification and sonification of spatial data rather than applying spatial data (trajectories) to exiting sounds. SenSynth was also my first step into instrument design and development. The "Kinetic Synthesizer" literally accelerated sensor data from physical motion to produce sound, and I began to correlate the 3 axes of motion used to for micro sensor vibrations with the 3 axes of motion used for macro level sound source spatialization in SES (Section 3.2). Spatial Modulation Synthesis (Chapter 4) would eventually create a continuum between these two concepts of spatial sound.

[22]`https://en.wikipedia.org/wiki/Nokia_N9`
[23]`http://fastlabinc.com/CSL/`

3.5 Image Sonification (2011-2013)

My experiments with image sonification began as a custom piece of software necessary to realize a multimedia installation by George Legrady entitled *Voice of Sisyphus* (2011), which explored the sonification of black and white photographs. What began as a simple tool for the audification of image pixel data eventually grew into a parameter-rich, polyphonic sound generator that could be completely controlled via Open Sound Control (OSC) to create audio-visual compositions in which the image and sound were not mapped to control each other, but were literally the same data. Eventually, the question arose about the potential of performing such a piece live, and it was quickly realized that the use of a computer mouse was not at all conducive to manipulating multiple regions of an image simultaneously. Multi-touch screens provided the ability to move and manipulate the parameters of multiple image regions for a performance scenario, thus creating the instrument, VOSIS, which currently runs as both an iPad application and a desktop application that can be used with large multi-touch overlays such as those by PQ Labs[24]. VOSIS is an image sonification software instrument that creates complex wavetables by raster scanning greyscale image pixel data to audify visual frequencies, creating inherent audio-visual relationships as image filters are applied to ultimately affect the resulting sound.

Papers describing the image sonification techniques present in *Voice of Sisyphus* and VOSIS have been published in the Proceedings of the International Conference on Auditory Display [42] and in the Proceedings of New Interfaces for Musical Expression [43] respectively. A demo video of VOSIS along with links to download the software for iPad and OS X are available at `http://www.imagesonification.com`. VOSIS has over 10,000 downloads internationally.

[24]`http://www.multitouch.com`

Background

Most experiments examining relationships between sound and image begin with sounds or music that influence visuals. Chladni's famous 18th century "sound figure" experiment involves visual patterns generated by playing a violin bow against a plate of glass covered in sand[44]. 20th century visual music artists often worked by tediously synchronizing visuals to preexisting music. Though, in some cases, the sounds and visuals were composed together as in *Tarantella* by Mary Ellen Bute. Today, visual artists often use sound as input to produce audio-reactive visualizations of music in real-time.

Less common are technical methodologies requiring images as input to generate sound. However, in 1929 Fritz Winckel conducted an experiment in which he was able to receive and listen to television signals over a radio[44], thus resulting in an early form of image audification. Rudolph Pfenninger's *Tonende Handschrift* (Sounding Handwriting), Oskar Fischinger's *Ornament Sound Experiments*, and Norman McLaren's *Synchromy* utilized a technique of drawing on film soundtracks by hand to synthesize sounds. VOSIS continues in the tradition of the aforementioned works by using visual data to produce sound.

Related Work

A vast majority of existing image sonification software uses the so-called "time-frequency" approach [9] in which an image acts as the spectrograph for a sound. These systems include Iannis Xenakis' UPIC and popular commercial software such as MetaSynth and Adobe Audition. Their shared approach considers the entire image much like a musical score where the vertical axis directly corresponds to frequency and the horizontal axis to time. Usually the image is drawn, but some software like Audition allows the use of bitmap images and considers color as the intensity of frequencies on the vertical axis. MetaSynth uses the color of drawn images to represent the stereo position of the sound. In any case, all of the aforementioned software reads images left to right at a rate corresponding to the tempo. Reading an entire image left-to-right as a means to image sonification has been termed as *scanning* by Yeo[45].

However, the approach with VOSIS was to focus on different regions within an image over the course of the performance or composition. Yeo has termed this approach *probing* [45]. Thus, unlike scanning, the horizontal axis of the image is not related to time. The peformer's *probing* of regions over time advances the composition non-linearly. The goal is a more literal translation of images to sound than the typical spectrograph scanning approach. It is felt that, although novel in their own right, spectrograph scanning approaches adhere too closely to a traditional musical score. VOSIS is a departure from the common practice of viewing images as time-frequency planes and provides a means to listen to variations between different regions of an image. The resulting sounds unfold as one explores areas of an image in a non-linear fashion– first noticing some region, person, or object and then shifting the focus to other objects within the scene.

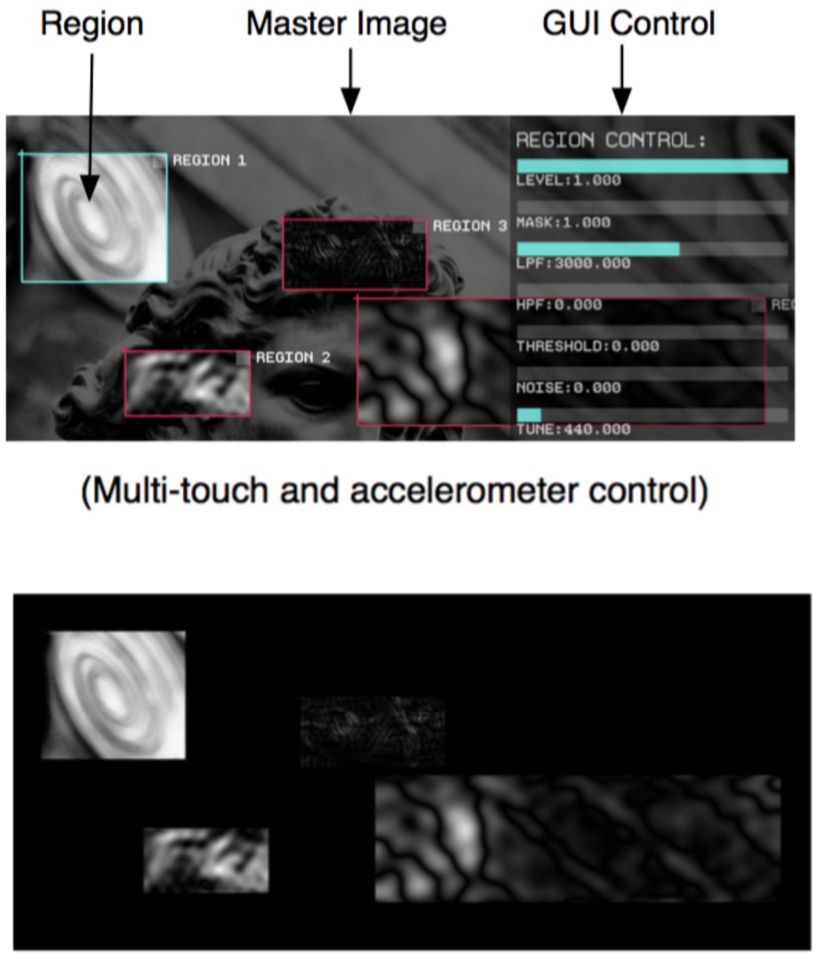
One convenient constraint with the initial project motivating VOSIS was that the software did not have to consider color since all source images were greyscale. The possible color-sound relationships with image sonification are numerous and, for now, beyond the scope of this project. While VOSIS has evolved into a more generic tool that can read any image or video, the lack of color consideration remains to serve as a useful limit of scope until current methods are refined and the multiple dimensions of color can be added to the algorithm without causing "parameter overload."

3.5.1 Implementation

VOSIS is designed to be used with both still images and video. For still images, the image itself is the "master" image, while for recorded or streaming video, the master image changes at the frame rate of the video recording or camera. Greyscale pixel values within a region of the master image are read into an array, filtered, drawn as a new image, and read as an audio wavetable. Regions can be selected either by drawing rectangular boxes on the screen or simply by touching an area of an image with a selectable region size. Each region can be thought of as a note with its frequency dependent on the frequency content of the pixels within that region of the image and the rate at which the image pixels are scanned. Chords can be formed by selecting multiple regions at once. There is also a segmentation mode which subdivides large regions and plays them back as a sequence of notes. Consideration was taken for real-time manipulation of region locations and sizes during a performance or installation without introducing unwanted audio artifacts.

Interface

Users start by selecting an input source, either an image file, video file, or video camera. Regions are added in either "draw" mode or "touch" mode. In draw mode the user can touch to draw outlined rectangular regions of any size. Sounds from regions are looped indefinitely and up to 10 regions can be added in draw mode. Once a region is added its filter parameters (described in detail in section 3.5.1) can be adjusted via sliders on a GUI panel or mapped to be controlled by a multi-touch gesture or accelerometer direction (when running on an iPad). With multiple regions, touching within a region will make its parameters available for editing on the GUI panel. Thus, it is possible to draw several static regions with various parameters to create a dense sound. In touch mode the user simply touches anywhere on the image to play the sound of an adjustable-sized region centered at the touch location. An ADSR envelope turns touch mode into a keyboard-like instrument. When in touch mode the parameters on the GUI panel will affect every region added via touch mode. So, a typical usage scenario may be to add several regions in draw mode to build up a foundation on which to improvise while in touch mode.



REGION CONTROL:
LEVEL:1.000
MASK:1.000
LPF:3000.000
HPF:0.000
THRESHOLD:0.000
NOISE:0.000
TUNE:440.000

Figure 3.18: VOSIS Interface: Edit Mode (Top) and Performance Mode (Bottom)

Sounds resulting from regions added to a video or camera feed are inherently dynamic as the content of the region changes at the video frame rate. With both still images and video sources dynamics can be achieved by moving regions to hear the varying amplitude and frequency content within of different areas of the master image. Mapping the filter parameters to multi-touch gestures such as two-finger horizontal and vertical movement and pinch in/out can create a familiar type of sound performance interaction akin to an

XY pad or pitch or modulation wheel. Mapping a regions level or scan rate to the iPads accelerometer can result in expressive tremolo and vibrato of the multi-touch "chords." Segmentation mode subdivides regions over an adjustable size threshold into equally sized smaller regions and plays their sounds back in left-to-right, top-down order similar to a step sequencer at an adjustable tempo.

Presentation mode removes the GUI panel and region outlines from sight, making the application suitable for visual music performance. For instance, an iPad can be connected to a projector so that the montage of audio-visual movements can be displayed to an audience without the performers hands covering any visuals. The user is also able to adjust the opacity of the background master image source, making it possible to only display visuals as they are created and played via touch. Thus, it is easy to create live visual music performances by probing and filtering image regions. Figure 3.18 shows performance mode in comparison to the standard editing mode.

Synthesis Technique

Figure 3.19 outlines the image-to-sound synthesis algorithm in VOSIS. As described in section 3.5 VOSIS only deals with greyscale images, and any color or other format images imported to the software will first be converted to 8-bit greyscale. Once a region is selected, the synthesis algorithm begins with a back-and-forth, top-down raster scanning of the greyscale pixel values, which range from 0 to 255 (black to white respectively). Simply scaling these values to obtain a waveform of floating-point audio samples in the -1.0 to 1.0 range results in harsh, noisy sounds without much variation between separate regions in most images. These initial noisy results were not at all surprising given that the greyscale variation of an arbitrary image will contain a dense, broad range of frequencies. For instance, given a picture of a landscape, analyzing variations in each pixel value over a region containing thousands of blades of grass would easily

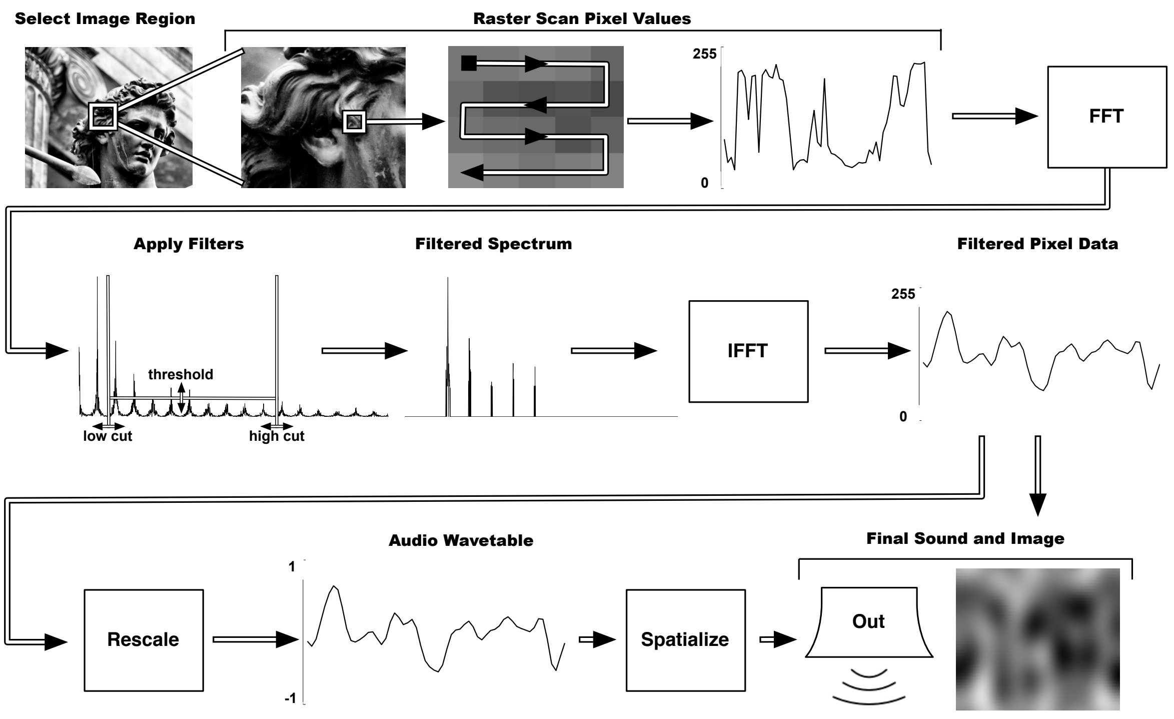


Figure 3.19: Sound Synthesis Algorithm in VOSIS

produce a noisy spectrum with no clear partials. Of course, images could be specifically produced to contain particular spectra and result in tonal sounds[46], but the interest of this project is to explore the sounds resulting from different regions of any arbitrary image. When initially experimenting with this form of image sonification one might be tempted to ask "What does a face sound like compared to a window?" However, the ability to determine high-level descriptions of image regions such as a "face" or "window" is a problem of feature recognition in computer vision, and not contained in the scope of this project. Rather, VOSIS examines the objective differences in the pixel data between faces and windows rather than what sounds someone may associate with each of those objects. In many cases the spectral-based filters in VOSIS help to produce less noisy sounds with greater distinguishability between regions.

A selection of frequency domain filters is applied to the audification of pixel data by implementing a short-time Fourier transform (STFT) for each region. The STFT is obtained by computing a fast Fourier transform (FFT) of each region at the graphics' frame rate. Each FFT gives amplitudes and phases for frequencies contained in that region at that time. Manipulation of these amplitudes and phases allows controls the spectrum of the image and, therefore, the resulting sound in real-time. Zeroing the amplitudes of frequencies above or below a cutoff produces a low-pass or high-pass filter respectively, while scrambling the phases of an FFT scrambles the pixels in an image and removes temporal cues from the sound without affecting is spectrum. The key filter was the implementation of a variable amplitude threshold, below which all frequencies are removed, thus leaving only the most prominent partials present to accentuate tonal differences between otherwise similar sounding image regions. Implementing this threshold denoises the resulting sounds, leaving clear tones that change as the region is moved or resized. The pixel data of regions is continuously updated to show the effect of the filters so the observer is always seeing and hearing the same data. As the sound becomes clearer from the removal of frequencies, the image becomes blurry. An interesting conclusion from this process is that most perceptually coherent images sound like noise while perceptually clear, tonal sounds result from very abstract or blurry images.

To obtain the final image and sound data after applying filters in the frequency domain an inverse short-time Fourier transform (ISTFT) is computed for each region, which gives the filtered pixel values. These new values are then scaled to the range -1.0 to 1.0 and read as an audio wavetable via scanned synthesis, a technique that can be used to scan arbitrary wavetables of audio data at variable rates using interpolation[7]. A control for the scan rate of these wavetables affects the fundamental pitch of the resulting sounds. However, the perceived pitch also changes as regions are moved and resized, causing new partials appear and disappear from the spectrum.

Before computing the FFT the pixel data can also be scaled to effect the brightness of the resulting image and, therefore, amplitude of the sound. A masking effect can also be applied at this point, which acts as a bit reduction to the image and sound by quantizing amplitude values. Overall, it is important to note that the software only manipulates the image data and not the audio data. Since the audio data is continually produced in the same manner (scanning the IFFT results), changes in the sound are always directly produced from changes in the image. Simply put, using VOSIS one is always seeing and hearing the same data. Figure 3.20 summarizes the sonic effects of the image filters.

Performance with VOSIS demands rapid movement and resizing of regions which initially caused discontinuities in the wavetables, resulting in an unwanted audible popping noise. To account for the resizing of images, all resulting audio wavetables, originally a length equal to the number of pixels in an image region, are resampled to a fixed size before an interpolated read of the table at the desired frequency. Wavetables are then cross-faded with each other at the audio buffer rate to prevent discontinuities from the dynamically changing wavetables resulting from the movement and resizing of regions. If the region's position and size are unchanged, then the wavetable is simply looped. Scrambling the phases of an image region can be used to obtain perceptually continuous sounds rather than loops.

Regions' sounds are spatialized according to their location within the image. If a region is segmented, then the spatialization algorithm updates the position of the sound as each subsection is played. The method of spatialization is similar to that used in vOICe[47], an augmented reality project for the totally blind. Sounds are stereo panned left-to-right according to their region's position in the horizontal image plane. With multiple regions present, the spatialization gives clarity to the mix and provides cues as to the location of sounds within the master image.

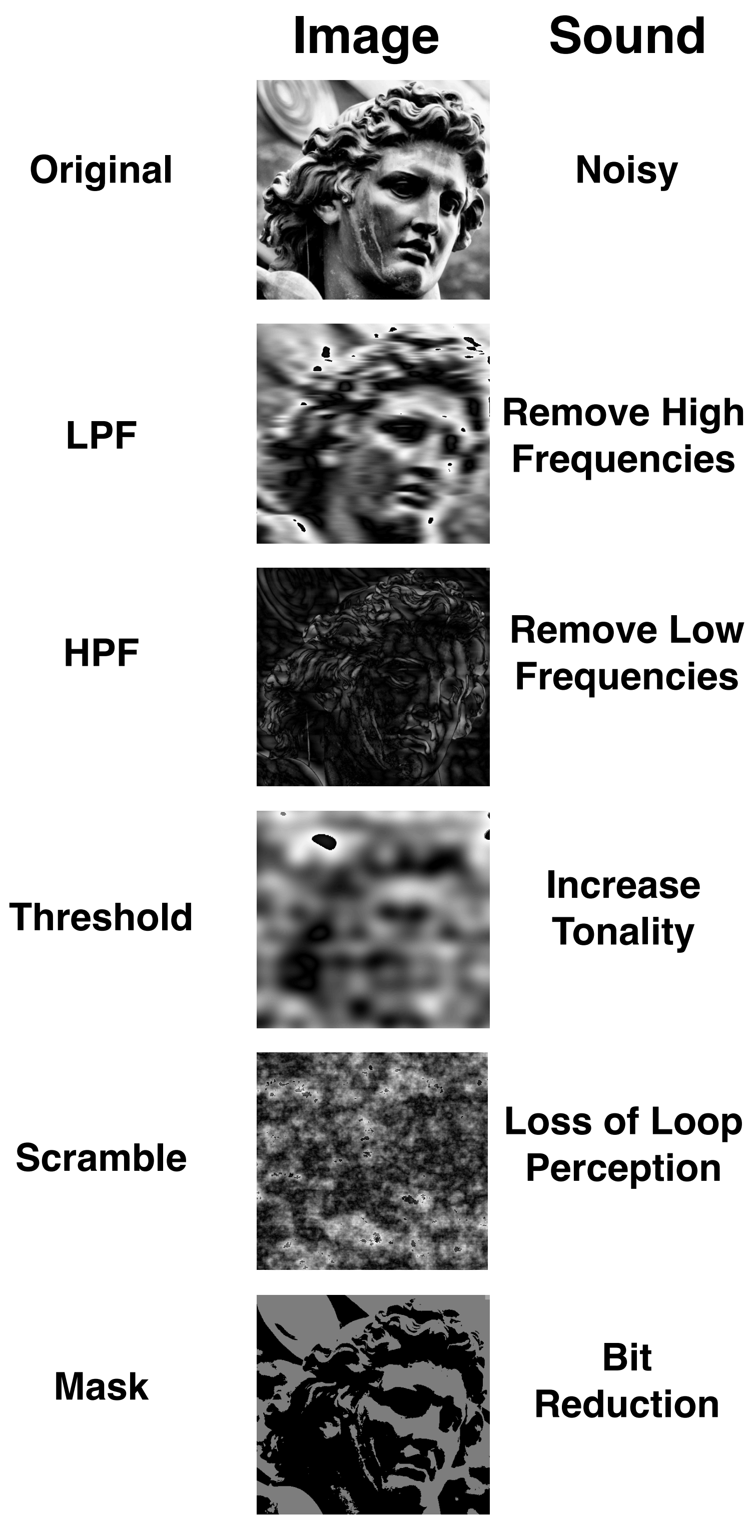


Figure 3.20: Effects of Image Filters on Sound

3.5.2 Results

VOSIS synthesis sound using 2-dimensional image pixel data. As with the sensor data in SenSynth (3.4), the image data must be scanned at sufficient speed to enter the audio domain. VOSIS also pans sound using each pixel's horizontal location within the image as a stereo left/right percentage. Thus, a non-linear probing of images can create a spatial audio-visual composition. The visual relationships observed emphasize shape-timbre correlations in which smooth shapes are more tonal with less noise and partials while rough, unfiltered images sound harsher. VOSIS creates a powerful sychresis [48] between a montage of images and their resulting sounds. All of these aforementioned features were influential for the highly visual component of spatial modulation's control interface and accompanying shape-timbre relationships. The concept of multi-dimensional visual form as the source for both a sound's spatial position and waveform is again present in spatial modulation synthesis (Section 5.2.1).

3.6 *No Heritage* (2011)

This piece was a study utilizing the synthesis techniques and custom software tools developed in this chapter thus far. I considered each technique the basis for each of the 3 sections comprising the piece. The first section used the Kinetic Synthesizer instrument created in SenSynth, which generates sound via audification of a mobile phone's accelerometer. The second section used sounds created via image sonification, which is explores the notion of probing a static image to generate sound. The images used were photographs of various historic cars from the 2011 Concours D'elegance show in Pebble Beach. The final section emphasized field recordings of people and vehicles (also recorded at the Concours D'elegance) processed through SES to create virtual, high-speed motion which resulted in extreme frequency modulation and granulation of the sounds due to simulated Doppler and distance based gain attention respectively. The work explores the sound of mechanical motion and the simulated motion of mechanical sounds.

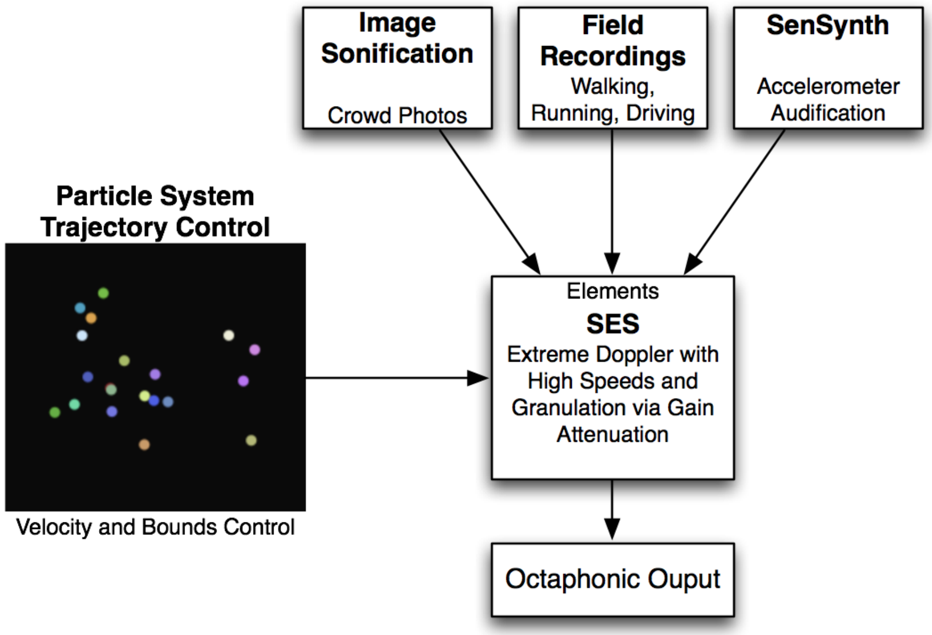


Image Sonification
Crowd Photos
Field Recordings
Walking, Running, Driving
SenSynth
Accelerometer Audification
Particle System Trajectory Control
Velocity and Bounds Control
Elements
SES
Extreme Doppler with High Speeds and Granulation via Gain Attenuation
Octaphonic Ouput

Figure 3.21: *No Heritage* Workflow

3.6.1 Results

In addition to the use of SenSynth and VOSIS, the piece rapidly spatialized swarms of moving sound sources (using SES's derived sources feature, Section 3.2.4), and exposed the potential of spatialization to create granulation effects in addition to amplitude and frequency modulation. The control for these spatial swarms was a simple visual particle system (written in Processing[25], sending OSC) which contained several bouncing circles representing each sound source. The primary controls for the system were sliders for velocity and bounds - i.e. how fast the sources moved and how far before they bounced back towards the listener. The center of the swarm could be controlled with the position of the mouse. Figure 3.21 shows a snapshot of the simple particle system used to produce spatial granulation and swarming effects via SES. In the end, I was able to produce a variety of effects and tones along a continuum from spatialization to spatialized granulation to rough FM tones by controlling only velocity and bounds parameters for a system of moving sound sources. Velocity and bounds would become the basis for control of spatial modulation synthesis (Chapter 4).

No Heritage was performed publicly in UCSB's Lotte-Lehmann Hall on October 13th, 2011. While the public performance was ocatphonic, a stereo version is available online at at `https://soundcloud.com/lifeorange/no-heritage`.

[25]`https://processing.org`

3.7 Seismic Sonification (2012-2014)

In 2012 I began a collaboration with with the Australian artist D.V. Rogers[26] during his residency at the UCSB AlloSphere to sonify 3-axis seismic data recordings. Supported by the Australian Arts Council, the goal of the research was to accentuate sonic differences in the audification of individual seismic events around the world. Unlike the audification of image pixel data (Section 3.5.1), seismic events are physical perturbations obeying the general wave equation, making them naturally conducive to audification. Increasing the playback rates of seismic recordings and rescaling the data values to match those of digital audio samples (straight audification) produces eerily realistic door slamming and explosion sounds. While others have explored the audification of seismic data [49], the variety of sounds produced lacks enough variety to be engaging for either interpretive sonification or for music. Rogers commissioned two pieces of electronica-style music from me, with the intention that this would push me to find ways of extracting more variety from the sounds while producing new seismic music that could be used to engage the public. Together, we also created a spatialized audio piece for the AlloSphere at UCSB and collaborated on a long-term seismic sound installation in Los Angeles.

[26]`http://allshookup.org/`

3.7.1 Audification Process of Seismic Data

Since seismic waves obey the general wave equation [27], the process of making them audible is simply a matter of rescaling the seismometer data values to the range of digital audio samples, [-1.0, 1.0], and playing the result back at a rate fast enough enter our range of hearing (20Hz - 20kHz). The typical range for seismic waves is 0.1-3 Hz, so increasing the playback rate by a factor of 100-1000X is suitable for making them audible.

Seismometers record activity 3-dimensionally along vertical, East-West, and North-South axes as shown in Figure 3.22. Seismic recording stations categorize their recordings by sample rate, gain sensitivity, and orientation. For instance a channel code of **BHZ** would indicate a **b**road band, **h**igh gain, vertically (**z**-axis) orientated recording. Broad band channels are indicative of a 10Hz-80Hz sampling rate (specified in the header of each recording), which are desirable for audification as they is the highest available sample rates for any event. Likewise, high gain channels are desirable to produce more amplitude resolution in the resulting sounds. Differences amongst the audification of separate axes are subtle (Figure 3.24), so the vertical, Z, channel it typically used by default, but it is possible to synchronize and mix the audifications of all 3 orientations together to produce a fuller sound.

[27] `http://en.wikipedia.org/wiki/Wave_equation`

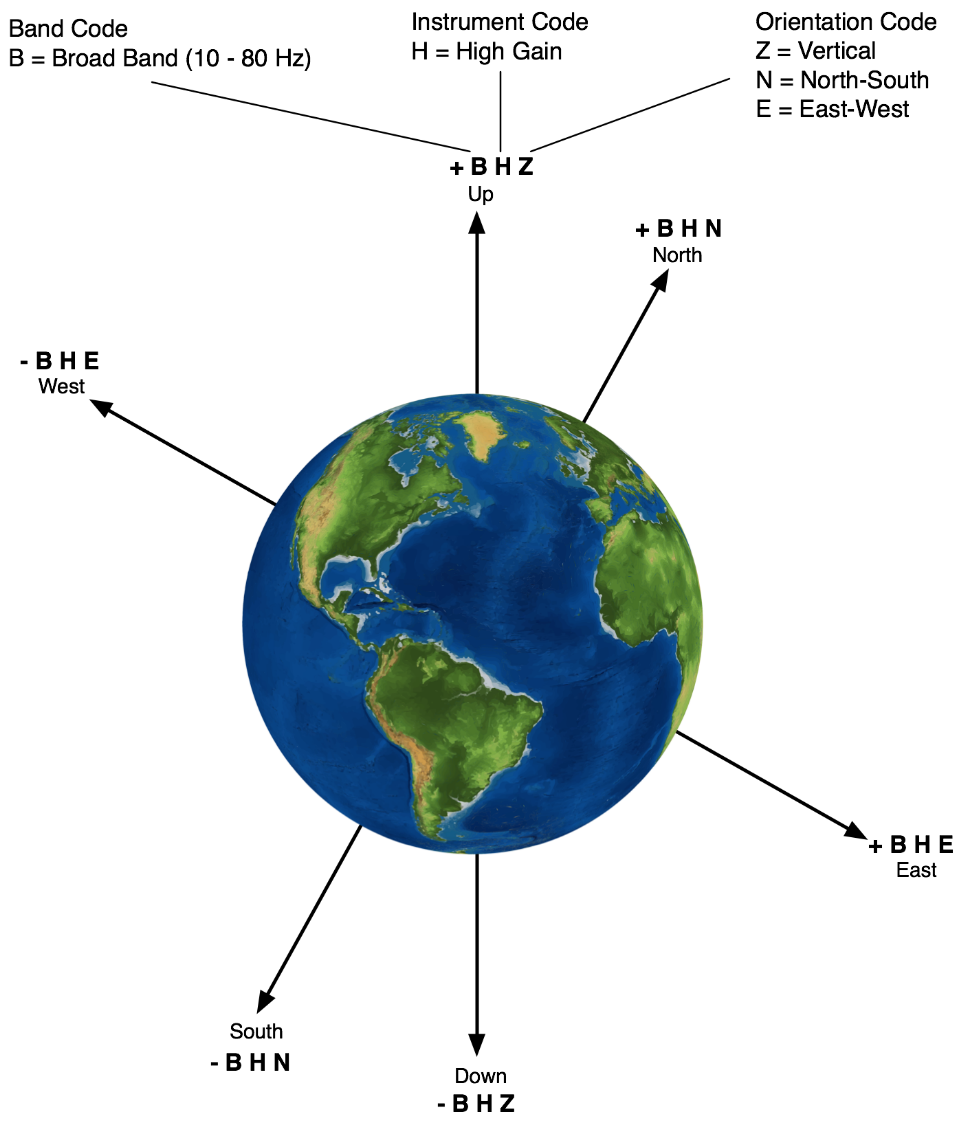
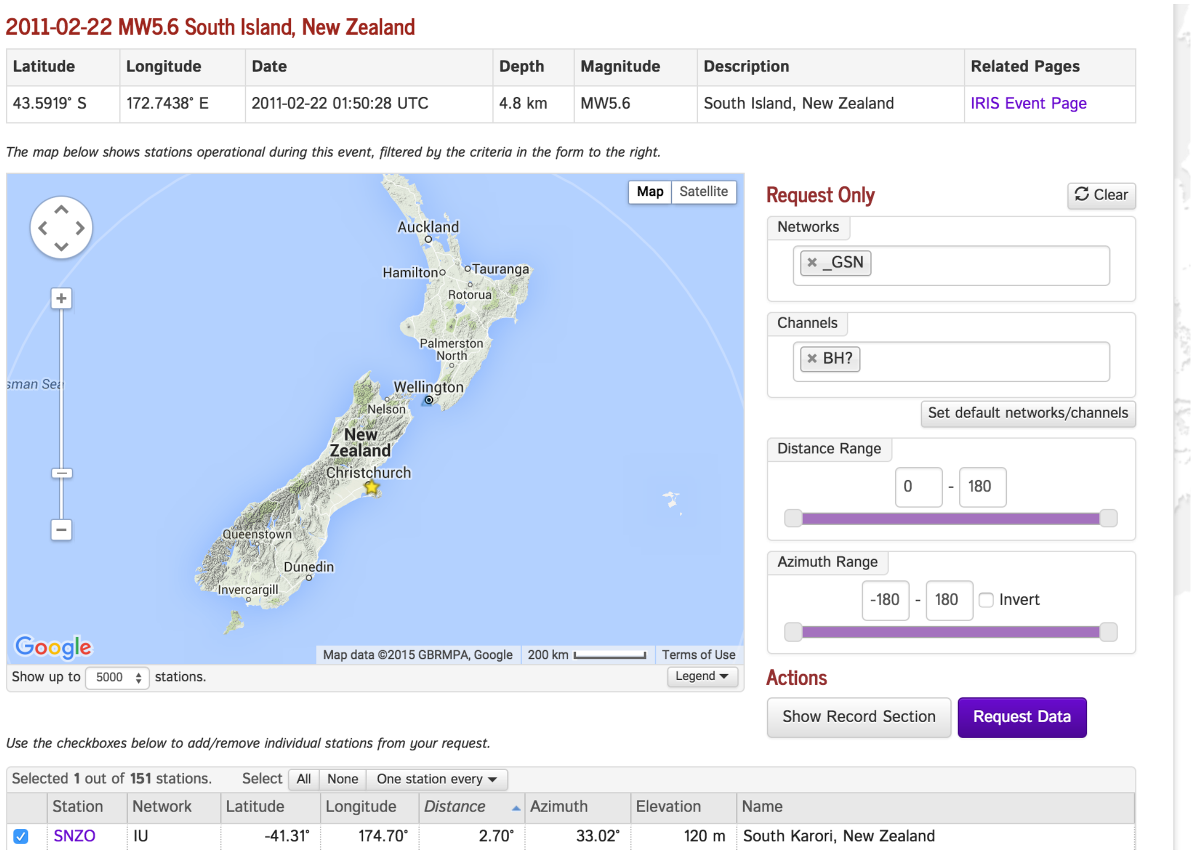


Figure 3.22: Orientation of Seismic Recording Channels

Our experiments began with data collected from the February 21st, 2011 magnitude 6.1 Christchurch, New Zealand Earthquake. IRIS (Incorporated Research Institutions for Seismology)[28], provides an online interface for accessing a database of current and past seismic events around the world[29] as seen in Figure 3.23. Figure 3.24 shows the waveforms produced from audification at 276 times the original speed of the Christchurch quake event along each axis.



2011-02-22 MW5.6 South Island, New Zealand

Latitude	Longitude	Date	Depth	Magnitude	Description	Related Pages
43.5919° S	172.7438° E	2011-02-22 01:50:28 UTC	4.8 km	MW5.6	South Island, New Zealand	IRIS Event Page

The map below shows stations operational during this event, filtered by the criteria in the form to the right.

Show up to 5000 stations.

Request Only

Networks: _GSN

Channels: BH?

Set default networks/channels

Distance Range: 0 - 180

Azimuth Range: -180 - 180 Invert

Actions

Show Record Section | Request Data

Use the checkboxes below to add/remove individual stations from your request.

Selected **1** out of **151** stations. Select All None One station every

	Station	Network	Latitude	Longitude	*Distance*	Azimuth	Elevation	Name
✓	SNZO	IU	-41.31°	174.70°	2.70°	33.02°	120 m	South Karori, New Zealand

Figure 3.23: Iris Wilber Seismic Database Interface

[28]`http://www.iris.edu`
[29]`http://ds.iris.edu/wilber3/find_event`

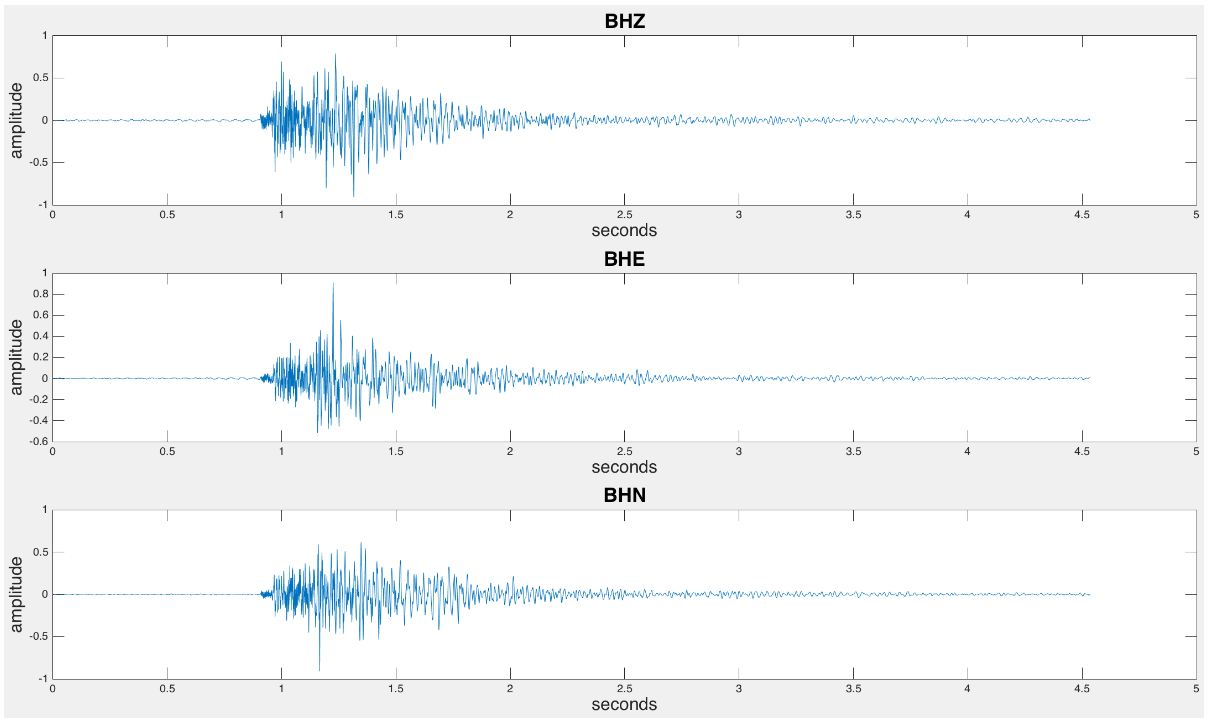


Figure 3.24: Audified Seismic Channels

3.7.2 Sonification Processes of Seismic Data

Wanting to expose more sonic variation for each seismic event without straying from the data, I explored several means of effects processing without using parameter mapping so that the seismic data sets would remain as the only sound generators. This is an important distinction within the field of sonification since most techniques involve the mapping of data to parameters of subjectively chosen sound generators. With this work, the original seismic waveforms are accelerated and scaled to generate sound, and variety is achieved by resampling, filtering, granulation, time-stretching, and pitch shifting. Since the definition of audification limits processing to resampling and scaling, any other modifications to the sound enter the realm of sonification. The following processes were used heavily in the creation of the *Christchurch* and *Haiti* compositions (Section 3.7.3).

Several sound examples to accompany the following sections are online at `http://i-e-i.wikispaces.com/Auditory+Display`.

Synchronous Granulation

Granulation of sound is the process of slicing a sound into several sound "grains" creating segments lasting 1 to 100 milliseconds. If the grains are played back in order then the original sound results. One can repeat each adjacent grain a number of times to result in a new sound 5 times longer than the original. Choosing an arbitrary duration for each repeated grain (1 to 100ms) will result in several discontinuities in the sound. For example, a grain's amplitude may start at 0.23 and end on -0.72. When repeated, this jump in amplitude would produce an undesirable click in the sound. To solve this a short amplitude window (envelope or ramp) is applied to each grain so each always starts and ends at amplitude 0. Next, the grains are overlapped so there is less audible beating from the windowing. In this work, synchronous granulation refers to methods of granulation that involve several grains of identical duration and windowing. Audified earthquakes can be characterized by an initial high frequency, high amplitude sound that decays over time like hitting a snare drum. Synchronous granulation has the effect of time-stretching these sounds, repeatedly emphasizing each grain, which emphasizes the unique decay of each earthquake.

Asynchronous Granulation Based on Zero-Crossings

Asynchronous granulation implies that each sound grain will have have different characteristics. In this case, the duration of each grain varies over time based on an algorithm that chooses the start and end points for each grain based on the location of zero crossings within the sound, which are points where the wave's amplitude is equal to 0. Zero crossings may occur often, sometimes even less than 1ms apart, so a minimum time for each grain is also specified. A convenience of using zero crossings is that windowing is not needed since the grains will already start and end on 0. Another quality of zero crossings is that they usually indicate the beginning of an impulse or large transient within the sound. When asynchronous grains are repeated multiple times, major impulsive portions of each earthquake are emphasized.

Time-Stretching, Pitch Shifting, and Filtering via Phase Vocoding

The phase vocoder is essentially a more advanced granulation tool. Its process breaks the sound into multiple segments of equal duration and uses a Fast Fourier Transform to analyze the frequency spectrum of each segment. One may interpolate multiple spectra between two segments to extend the duration of a sound while maintaining its frequency content. If one time-stretches a sound in such a fashion and then alters the playback rate, the result becomes a change in pitch without a change in duration (unlike audification). The spectra of each segment can also be manipulated to apply filtering effects. Removing frequencies below a certain amplitude threshold has the effect of de-noising a sound, leaving only the most prominent frequencies. This de-noising can be taken to extremes to leave only a few partials in each sound, producing unique tones for a seismic event.

3.7.3 Compositions and Installations

Christchurch (2012)

Christchurch uses a single seismic recording from the nearest station to the February 21st, 2011 Christchurch earthquake. The piece begins with a build-up of several reversed audifications of the event, time-stretched at different speeds. Then, the strong impact from the raw audification is heard, followed immediately by a chaotic granulated version emphasizing the loudest points in the impact. A tone fades in that is an extremely time-streched, pitch shifted version of the event with all but the most dominant partials filtered out of the sound. Other versions of this tone eventually overlap at manually coordinated harmonic pitch intervals. The event is played back using several different time-strech factors and synchronous granulations during the course of the piece. Timing becomes more ordered and apparent until rhythmic granulations and tones lead to a final build-up, ending with another raw audification.

The composition can be found online at `https://soundcloud.com/seismicsounds/christchurch-earthquake` and has over 30,000 unique plays.

Haiti **(2012)**

Haiti uses seismic recordings of the 12th January, 2010 magnitude 7.0 Haiti event and explores the variety of sounds coming from the same event recorded by the nearest 12 stations. The piece begins with granulated audifications of each station played in succession from the furthest to nearest station at the same playback rate. A brief recording of sensor noise from each station is played along with the granulations. Since each station has its own distinct sound, these noises represent signatures of each station as the listener moves nearer to the quake. A slow melody plays in the background that was generated by filtering out all but the most prominent frequency from the spectra of each noise signature. A rhythmic sound in the background was generated from the impact of each station played back at high-speed in succession again from furthest to nearest. As this rhythm gradually increases in intensity a lower noise grows in the background, which is a time stretched recording of the impact played in reverse to further emphasize the backwards (far to near) build-up of the piece. At the climax the impact from the nearest station plays, followed by impacts from the other stations – this time increasing in distance. In the background the low growl of the time-stretched recording fades away. The sounds become chaotic after the main event using asynchronous granulation based on zero-crossings.

The composition can be found online at `https://soundcloud.com/seismicsounds/haiti-earthquake-12th-january` and has over 4,000 unique plays.

Shadow Zone Shadows **(2012)**

Shadow Zone Shadows was a sound study orchestrated by D.V. Rogers that spatialized seismic audifications according to the geographic location of their impact and simulation of seismic propagation through the earth. This piece was realized in the USCB AlloSphere, allowing the listener to imagine being placed at the center of the earth while experiencing a series of earthquakes sounding and moving around them. A variety of international seismic events were used ranging in magnitude from 5.5 (Los Angeles) to 9.1 (Sumatra). The piece used custom spatialization software that allowed us to program trajectories synchronized with specific points in the seismic data (start of event and reflections). Just as the seismic data was sped up to become an audible audification, the spatial trajectories were simulated at speeds much faster than actual seismic waves propagate through the earth.

A recording along with more information on the events used can be found online at `https://soundcloud.com/seismicsounds/shadow-zone-shadows`.

DOMUS (2014)

DOMUS was an experimental piece of anti-seismic architecture by D.V. Rogers incorporating spatialized seismic sound and light (Figure 3.25). 8 speakers and 8 subwoofers played back seismic audifications and the *Christchurch* and *Haiti* compositions continuously from 10am to 10pm for the 7-month duration of the installation (October-May 2014) at Materials and Applications, Los Angeles[30]. The low frequency audifications

[30]`http://www.emanate.org/past-exhibitions/domus`

literally rumbled the structure, emphasizing their seismic nature. A light chandelier by Rene Christian[31] was comprised of a LED matrix that visualized the audifications through mappings between sound frequency and amplitude to light color and brightness.

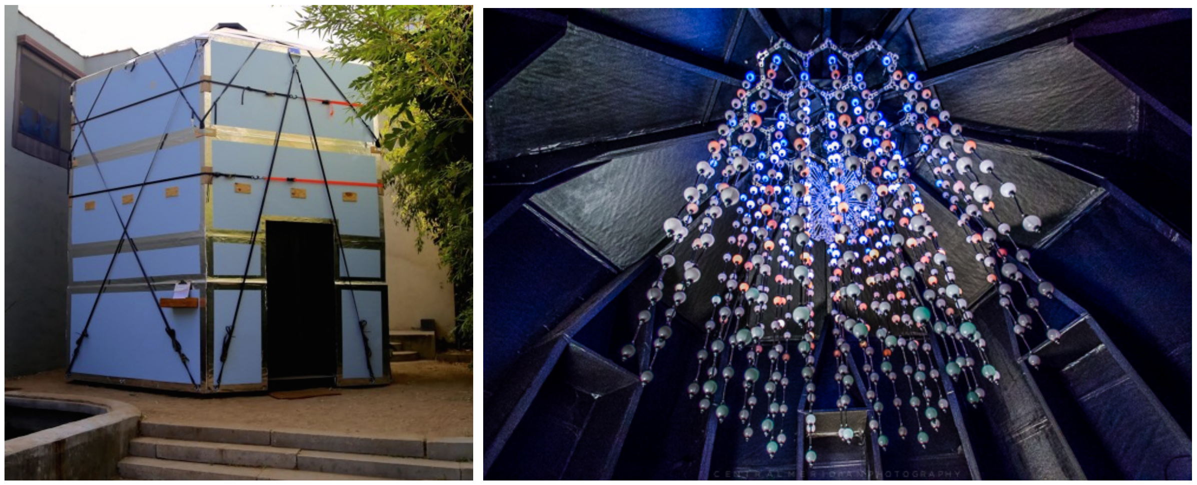


Figure 3.25: DOMUS Seismic Sound and Light Architecture

More information on DOMUS is online at `http://domus.urbanaction.org`.

3.7.4 Results

Work with seismic sonification emphasized the notion of accelerating 3-axis spatial data - not only by increasing audification rates, but through spatialization of these sounds according to geographical location and propagation. However, the means by which I could synthesize sounds from seismic data and spatialize the result were not unified - meaning I had created separate software applications to realize spatial sonification. As with image sonification (Section 3.5.1) I becaome interested in considering a waveform derived from spatial data as both a sound source and trajectory. The heavy and effective use of granular synthesis in this work created further desire to unify granulation with spatialization and synthesis.

[31]http://renechristen.net

3.8 *Kinetic* (2014)

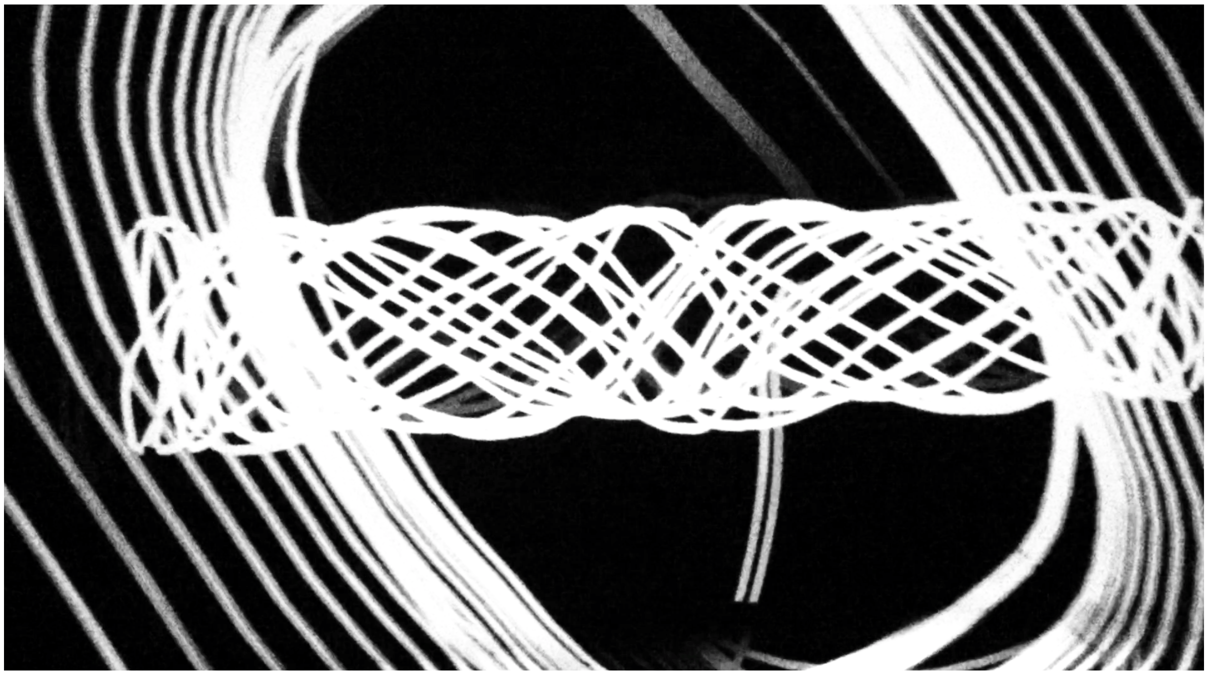


Figure 3.26: Photograph of Expanding Orbit from *Kinetic* in the AlloSphere

Kinetic was a spatial, audio-visual composition exhibited in the AlloSphere[32] at UCSB as part of the Media Arts and Technology End of Year Show, 2014[33]. Through the use of a 54.1 channel sound system and full surround 3D graphics, the audience was completely immersed inside complex sonic trajectories as Figure 3.27 shows for scale. The software implementation, control parameters, and audio-visual relationships emerging from *Kinetic* combined desires and results of past projects and became the basis for spatial modulation synthesis (Chapter 4).

[32]`http://www.allosphere.ucsb.edu`
[33]`http://show.mat.ucsb.edu/2014`

3.8.1 Sample Rate Spatialization Control

The underlying goal of this work was to control a sound spatialization paradigm such that the Doppler shift produced from a moving sound source could be used to produce FM tones when as the source moves rapidly back and forth across a listener. While rough effects using Doppler for frequency modulation were previously realized in compositions using SES, I had not yet produced a means of sample rate spatialization control with Doppler. Using the open-source C++ multimedia library, AlloSystem[34], I was able to modify existing code to attenuate, pan, and Doppler shift an arbitrary sound source at sample rate as opposed to the existing buffer rate control. While control at audio buffer rate is sufficient for most spatialization applications, the high speeds demanded by *Kinetic* necessitated sample rate representation of sound trajectories. Chapters 4 and 5 detail the specific speeds required and insufficiencies of preexisting rendering paradigms.

Also critical to the realization of the piece was the ability to visualize the high-speed sound trajectories. Typically, spatial sound trajectories would begin their existence on a graphical thread controlled by a GUI editor, DAW automation, or OSC and a separate audio rendering thread would receive control updates from the graphics thread at the start of each buffer and interpolate the position for each sample of sound. So, typically, sound trajectories exist as visuals that control sound. The problem with that approach is that visuals are rendered at graphics frame rates (30-90 Hz) that are only a fraction of audio sample rates (44.1-192 kHz), so resolution is sacrificed when sounds move with extreme non-linear velocities (acceleration). To accurately represent Doppler shift produced from accelerating high-speed sound sources, *Kinetic* needed to compute trajectory updates for each sample of audio. A history of spatial coordinates for each sound had to be stored into a buffer periodically read by the graphics thread for display, akin to how streaming

[34]`https://github.com/AlloSphere-Research-Group/AlloSystem`

audio is visualized. Thus, trajectories were computed with full resolution on the audio thread and displayed without loss of resolution on the graphics thread through the use of a shared buffer. Section 5.4 goes into more detail regarding this novel audio-visual rendering as it is applied to spatial modulation synthesis.

3.8.2 Compositional Techniques

The source material for *Kinetic* consisted of two sound samples and 1 sinusoidal oscillator. One sample was of train noise recorded at Berlin's Hauptbahnhof, and the other was a recording of a train conductor's announcements while on a high-speed train departing from the station. Each of the three samples was treated as a moving sound source spatialized using the software described in Section 3.8.1. As the trajectories were controlled from the audio thread, there as no GUI to the software and trajectory control was completely preprogrammed though rendered in real-time. The piece is explores the extreme acceleration of sound sources moving within variable bounds. Everything begins with a quite literal, audio-visual accelerando in which the initially spatially separated, slow moving sound sources increase in velocity to the speed of sound, becoming unrecognizable and spatially smeared both audibly and visually. The bounds of the oscillator is reduced to form a FM tone and resulting visual orbit that is then suddenly expanded to break the tone into a deeply modulating sound (Figure 3.26) that expands further into granular stream (Figure 3.27). The sampled sound sources slowly decelerate until all sounds return to their original low speeds. Simply controlling high-speed velocity and bounds changes created a diverse sound palette that allowed for the morphing between FM and granular sounds spatially and visually.

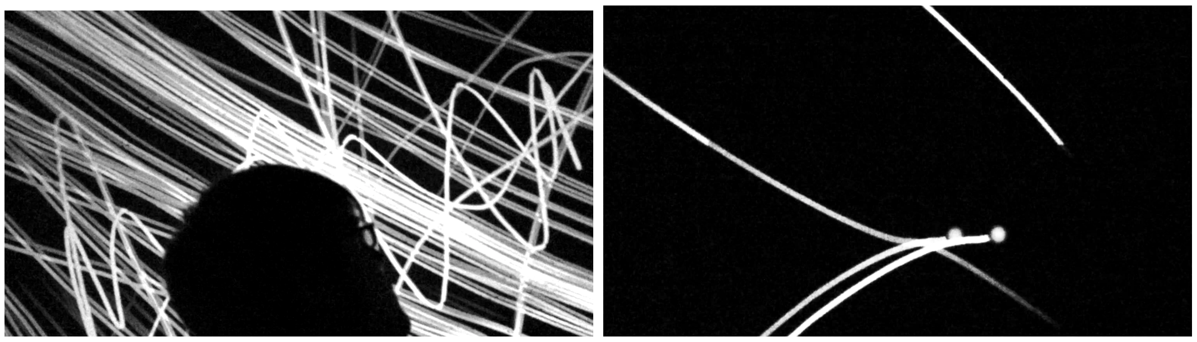


Figure 3.27: Photograph of Granular Trails (Left) and (Right) from *Kinetic* in the AlloSphere

3.8.3 Results

The realization of *Kinetic* proved that through the sample-rate control of sound spatialization and the simulation of precise, accelerating trajectories for moving sound sources it is possible to sculpt the Doppler curve produced from sounds crossing and changing directions with respect to a listener. It was found that when these sounds move at extremely high speeds (over 100 meters per second), periodically crossing a listener several times per second, the frequency of the source is modulated at audio rates and depths akin to traditional FM synthesis. By controlling the bounds of sound source movement, timbres could smoothly morph between tones and granulations with increasing distance. *Kinetic* explored high-speed motion of sound sources sonically and visually – by drawing 1 second history trails of sound trajectories, it became possible to see complex patterns in motion (Figure 3.26) otherwise too fast to be observed by the eye. Relationships between motion, visual patterns, and sound timbre were discovered, leading to the theory of spatial modulation synthesis described in Chapters 4 and 5. The software implementation of *Kinetic* using AlloSystem led to the development the novel audio-visual rendering paradigm used for spatial modulation synthesis and described in Section 5.4.

3.9 Towards Spatial Modulation Synthesis (2009-2015)

The culmination of the experiments in this chapter is the formal theory of Spatial Modulation Synthesis (SM) 4, a novel control continuum between sound spatialization and synthesis emphasizing high-speed sound trajectories to morph spatial choreography into FM tones and granulation. All of my previous experiments have elements of SM, which were ultimately brought together in a software realization of SM (Section refchap:SMplug). The most important and significant common thread in the work leading to SM is the notion of acceleration to produce timbre, meaning the acceleration of simulated spatial sound trajectories and the acceleration of spatial data.

W.A.N.T.S. introduced amplitude modulation effects from high-speed panning as well as the ability to program sample-rate spatialization trajectories, though not in real-time and without any form of visual control. Sound Element Spatializer provided real-time, visually controlled spatialization with Doppler leading to frequency modulation effects, though control was not at sample-rate and thus not physically accurate for high-speeds. SenSynth and Seismic Sonification dealt with accelerated 3-axis data to synthesize sound, and showed that trajectory data could produce timbres if scanned at sufficient speeds. SenSynth also explored the creation of various instruments with 3D spatial control. Likewise, image sonification experiments accelerated 2-dimensional image data into the audio domain to produce sound from photographs, leading to correlations between visual shape and sound timbre. Finally, *Kinetic* demonstrated audio-visual relationships between high-speed moving sound sources and their visual trajectories, creating a wide range of timbres and shapes using only controls of velocity and bounds controlled at sample rate.

Spatial Modulation Synthesis (SM) (Chapter 4) will introduce a theory of precise, mathematical control of timbre from sample rate spatialization utilizing Doppler, distance-based gain attenuation, and panning. The implementation of SM (Section 5.3) is the first

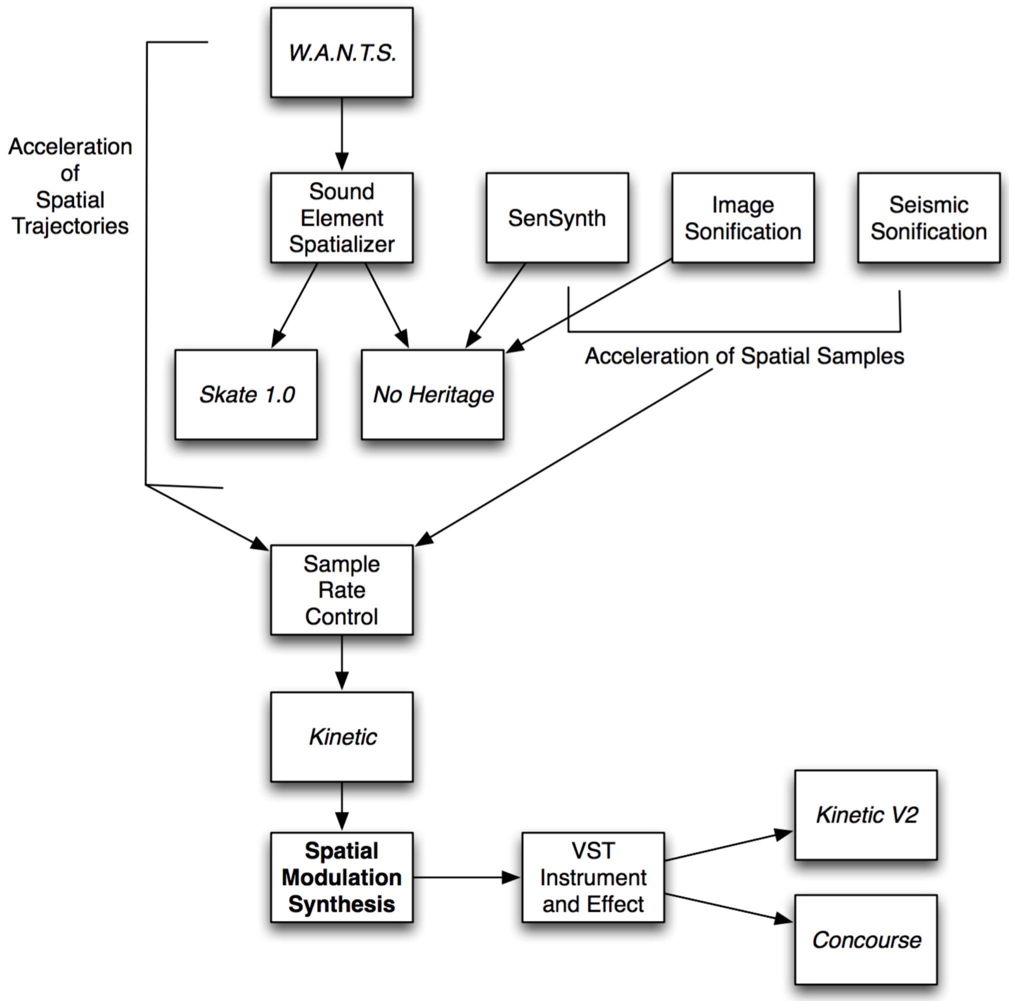

W.A.N.T.S.
Acceleration of Spatial Trajectories
Sound Element Spatializer
SenSynth
Image Sonification
Seismic Sonification
Skate 1.0
No Heritage
Acceleration of Spatial Samples
Sample Rate Control
Kinetic
Spatial Modulation Synthesis
VST Instrument and Effect
Kinetic V2
Concourse

Figure 3.28: Practice Leading to Spatial Modulation Synthesis

software plug-in to provide 3-axis sample rate modulation of spatial sound position, creating a unity between spatialization and synthesis. *Kinetic V2* and *Concourse* (both described in (Section 5.3) demonstrated the use of the SM plug-in for both multichannel synthesis and and spatialization applications within research and commercial venues.

Chapter 4

Spatial Modulation Synthesis

Spatial modulation synthesis (SM) is a novel sound synthesis control paradigm based on the spatialization of moving sound sources at high velocities along periodic orbits. Simulated Doppler shift of high-speed moving sounds is used to achieve audio-rate frequency modulation and is termed Doppler FM. Amplitude envelopes generated from distance-based gain attenuation decompose high-velocity sounds within large boundaries into a stream of spatialized grainlets. By rendering the visual trails of the aforementioned trajectories, it becomes possible to correlate visual patterns and symmetry to sound timbre and harmony. Thus, spatial modulation synthesis provides an inherently visual means of synthesizing timbre using spatial, rather than auditory control parameters.

This chapter focuses on the mathematical theory and components of SM, while Chapter 5 evaluates the limits, implementation, and novel effects produced through SM. Spatial Modulation Synthesis was published in the Proceedings of the 2015 International Computer Music Conference [50].

4.1 Introduction

SM is a paradigm through which modulation synthesis and granulation effects can be produced via high-speed sound spatialization within periodic sound trajectory orbits. Hence the name, spatial parameters are modulated to result in frequency and amplitude modulation of an input signal rather than modulating those parameters directly. Spatialization algorithms include a means for distributing the output to each loudspeaker, such as VBAP [51] or Ambisonics [52], and are typically paired with multiple distance cues including Doppler shift and a distance-based gain attenuation model. Additional distance cues may include air absorption filtering, presence filtering, and reverberation. For now, the concept of spatial modulation synthesis focuses only on Doppler shift and gain attenuation, leaving other possible cues as future research. Figure 4.1 shows the derivation of spatial modulation synthesis from the spatialization of a sound source according to a bounded, periodic trajectory moving at high-speed. The source's velocity and bounds are integral parameters of the timbres resulting from modulation of its spatial position.

While the spatialization of sounds along calculated trajectories and frequency modulation (FM) synthesis have been used together in many monumental compositions (Section 2.1), spatialization has not been used to produce FM synthesis through precise control of simulated sound source trajectories and the resulting Doppler shift. Similarly, while the spatialization of sound grains has been implemented by various composers, spatialization itself has not been used to create granular streams and clouds through rapid motion modulating a distance-based gain attenuation. Using existing spatialization, Doppler, and attenuation techniques, spatial modulation synthesis provides a new control and visualization paradigm through which space and timbre are connected.

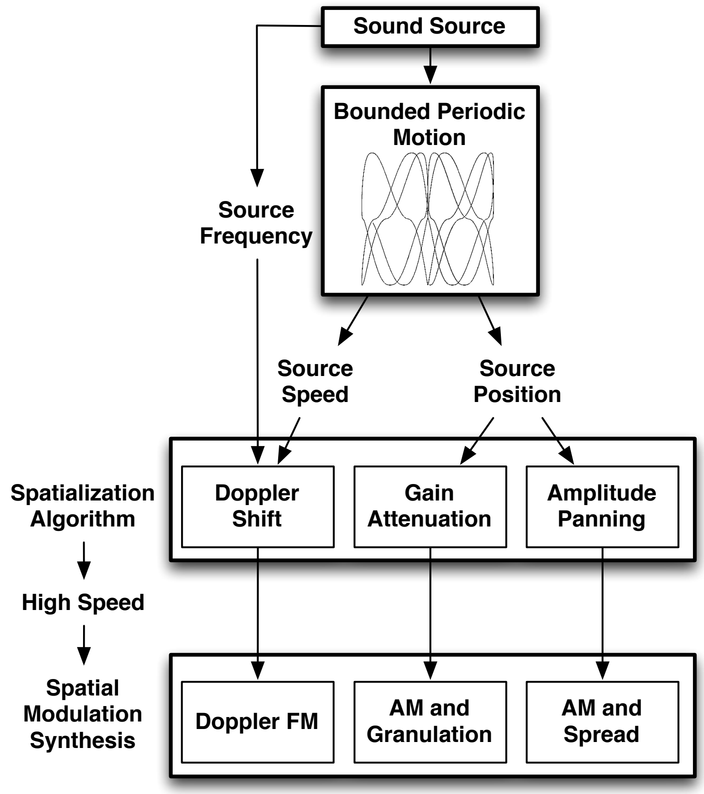


Figure 4.1: Spatial Modulation Synthesis

4.2 Doppler FM

Doppler shift is the phenomena of increased pitch for sound sources moving toward a listener and decreased pitch for sound sources moving away from a listener. Through the simulation of precise trajectories and velocities of moving sound sources it is possible to sculpt the Doppler curve produced from sounds crossing and changing directions with respect to a listener. The term Doppler FM is used to describe audio rate frequency modulation resulting from high-velocity sound source movement oscillating across a listener. First, the simplest case involving the motion of a sound source with a constant magnitude of velocity moving back and forth along one dimension is demonstrated using a common approximation of the Doppler shift formula. Then, the sinusoidal acceleration required to produce typical FM is described. Next, we observe differences between a physically accurate Doppler implementation versus the approximation ubiquitous in digital audio which becomes inaccurate for high speed sources. While this section begins by describing the concept of Doppler FM in one dimension for simplicity, Section 4.2.7 extends the concept to produce complex 3D spatial modulation, producing more interesting timbral and visual results.

4.2.1 Doppler Shift Approximation

The true, physically accurate Doppler shift for a stationary listener and moving sound source is given by equation 4.1.

$$f = \left(\frac{c}{c \pm v_s}\right) f_s \tag{4.1}$$

f is the observed frequency, f_s is the frequency of the source, c is the speed of sound ($\approx 340m/s$) and v_s is the speed of the moving sound source, positive when moving away from the listener and negative when moving towards the listener. For $v_s \ll c$ the magnitude of the shift is roughly equal as the source moves towards and away from the listener at a constant v_s. However, for higher velocities, the change in frequency, $\Delta f = |f - f_s|$, will be uneven and greater when the source is approaching as shown in figure 4.2. For $v_s = c$ the observed frequency goes to ∞ and, in reality, a sonic boom occurs.

However, most digital audio Doppler simulations use an approximation for Doppler shift that is relatively accurate for $v_s \ll c$ and gives an equal shift of frequency both towards and away from the listener [53]. This Doppler approximation is given by Equation

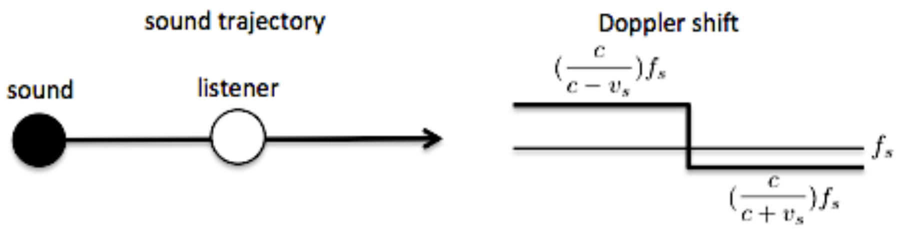


Figure 4.2: Asymmetrical True Doppler Shift of Moving Sound Source

.

4.2 and the resulting shift is shown in Figure 4.3.

$$f = \left(1 \pm \frac{v_s}{c}\right) f_s \tag{4.2}$$

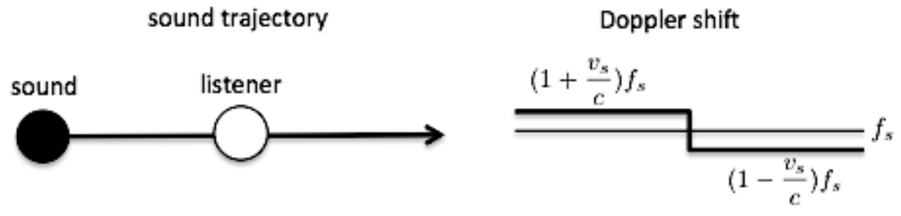


Figure 4.3: Symmetrical Doppler Shift Approximation of Moving Sound Source

.

Although equation 4.2 is not physically accurate for the high values of v_s used in spatial modulation synthesis (which can exceed $100m/s$), the resulting sound effect is sufficient to give the illusion of an approaching source and conveniently provides a symmetrical frequency modulator to use as a starting point for SM as described in the following sections. Doppler FM using the true, physically accurate frequency shift is described thereafter.

4.2.2 Square Wave Modulator

Consider the scenario of a single sound source with frequency, f_s, moving on a horizontal x-axis at a constant velocity, v_s, past a listener at $x = 0$. If the motion of the sound source oscillates such that the direction of its velocity is inverted at a bounds at distance B in either direction from the listener then the resulting Doppler shift becomes periodic. For this back and forth motion at constant velocity, the resulting Doppler curve will resemble a square wave (Figure 4.4).

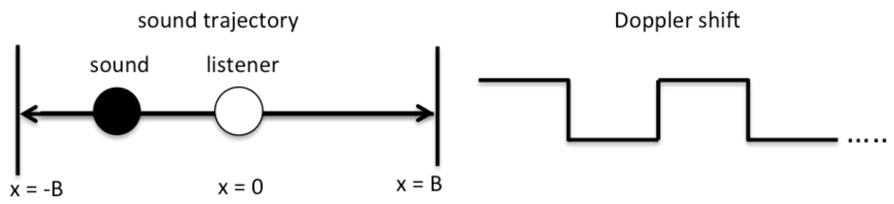


Figure 4.4: Square Wave Doppler FM from Constant Velocity

.

Now, considering the Doppler curve as a frequency modulator for our source with a rate of modulation determined by the time it takes the source to move a distance of $2B$ and a modulation depth directly related to the sound source's velocity (Equation 4.2), the modulation frequency (f_m) and depth (Δf) and are given by

$$f_m = \frac{v_s}{2B} \tag{4.3}$$

$$\Delta f = \frac{v_s}{c} f_s \tag{4.4}$$

From Equation 4.3 it is observed that the upper limit of f_m is dependent on maximizing v_s while minimizing B. Equaton 4.4 shows that the depth of modulation using the Doppler approximation will equal the source frequency at the speed of sound. The limits of these parameters are discussed in Section 5.1.

4.2.3 Sinusoidal Modulator

Typical FM synthesis involves a sinusoidal modulator and the instantaneous frequency is given as

$$f(t) = f_s + \Delta f sin(2\pi f_m t) \tag{4.5}$$

Classical means of FM and Equation 4.5 assume an amplitude and frequency adjustable oscillator as the modulator. SM, however, uses simulation of physical motion to modulate the frequency through Doppler shift. By applying a time-varying velocity (acceleration) to Equation 4.2 we can, instead, obtain $f(t)$ in terms of Doppler shift.

$$f(t) = \left(1 \pm \frac{v_s(t)}{c}\right) f_s \tag{4.6}$$

Thus, we can derive the acceleration required of our sound source to produce a sinusoidal Doppler curve by setting equations 4.5 and 4.6 equal and solving for $v_s(t)$.

$$v_s(t) = \pm \frac{\Delta f sin(2\pi f_m t) c}{f_s} \tag{4.7}$$

To determine v_s at a 1-dimensional horizontal position x between $-B \leq x \leq B$ Equation 4.7 can be rewritten in terms of position.

$$v_s(x) = \pm \frac{\Delta f}{f_s} \left| sin(\frac{x}{B}\pi) \right| c, -B \leq x \leq B \tag{4.8}$$

The sinusoidal velocity in equation 4.8 required to produce a sinusoidal modulator implies acceleration from rest at both boundaries $\pm B$ and at the listener ($x = 0$) (Figure 4.5). The depth of modulation remains the same as for square wave modulation (Equation

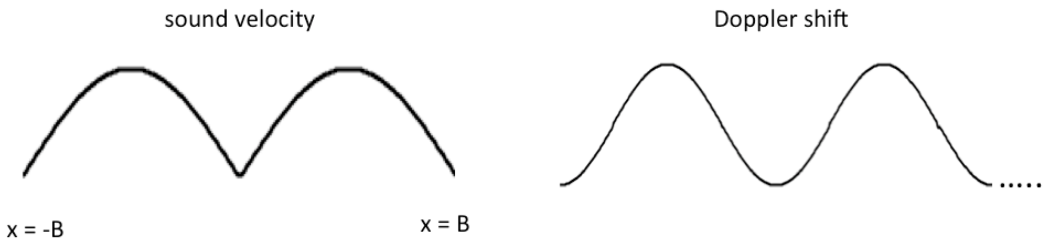


Figure 4.5: Sinusoidal Doppler FM from Accelerating Motion

4.4), but the frequency of modulation is divided by a factor of $\frac{\pi}{2}$.

$$f_m = \frac{v_s}{\pi B} \tag{4.9}$$

$$\Delta f = \frac{v_s}{c} f_s \tag{4.10}$$

4.2.4 Other Doppler Modulators

In addition to the square and sinusoidal frequency modulators produced by linear velocity and sinusoidal acceleration respectively, other accelerations produce complex frequency modulation curves such as triangle and sawtooth waveforms. Due to slower linear acceleration curves, the frequency of modulation for a given bounds will be lower than square wave and sinusoidal motion. The resulting modulation frequency for these modulators is half that of square wave modulation (Equation 4.3).

$$f_m = \frac{v_s}{4B} \tag{4.11}$$

Appendix A contains a summary of Doppler modulation types and their respective formulas.

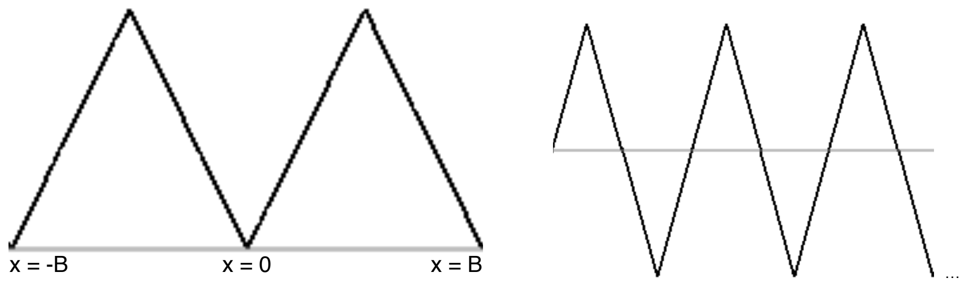


Figure 4.6: TriangleVelocity (Left) and Resulting Modulator (Right)

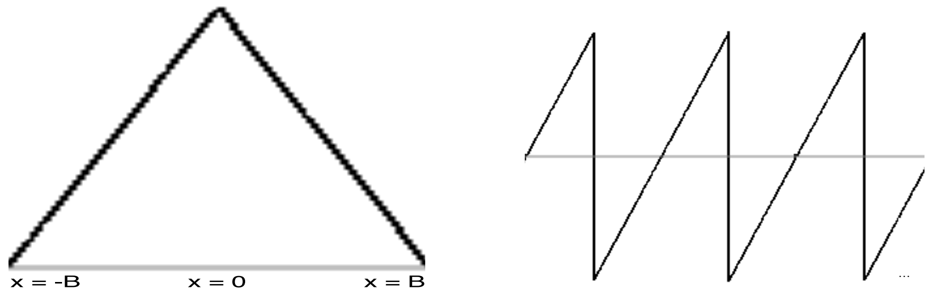


Figure 4.7: Sawtooth Velocity (Left) and Resulting Modulator (Right)

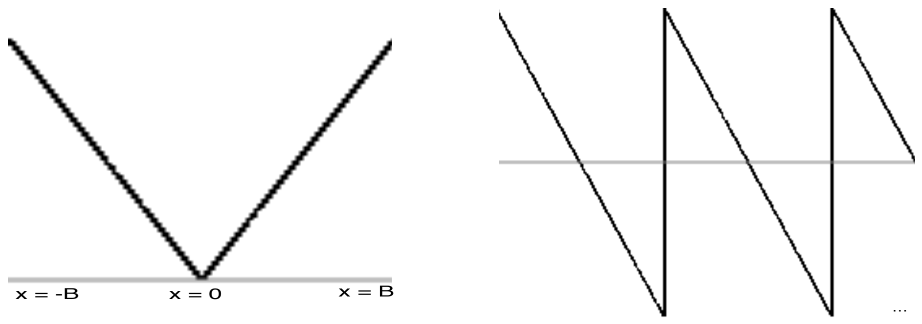


Figure 4.8: Inverse Sawtooth Velocity (Left) and Resulting Modulator (Right)

4.2.5 Asymmetrical Bounds

Thus far, for simplicity, SM has been explained with the assumption that the maximum and minimum boundaries of trajectory oscillation are equal in distance. However, asymmetrical bounds may be used to further shape the frequency modulators resulting from Doppler FM. Magnitude of bounds, B, is now replaced by B_1 and B_2 , $B_1 \neq B_2$.

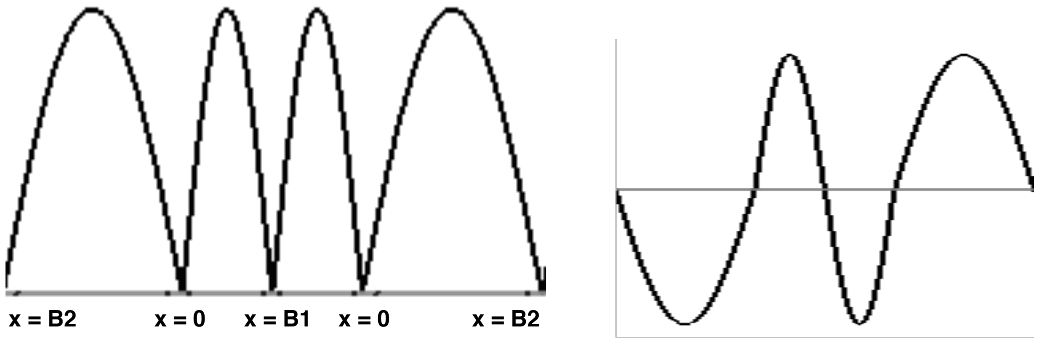


Figure 4.9: Sinusoidal Velocity Within Asymmetrical Bounds (Left) and Resulting Modulator (Right), *B2 = 2B1*

Asymmetrical bounds introduces asymmetries in the duty cycles of the resulting modulators, producing phase distortion. Thus, the equations for modulation frequency change for each type of motion. Appendix A summarizes the changes of of modulation index for each motion type with asymmetrical bounds, though modulation depth remains the same.

$$f_{m_{square}} = \frac{v_s}{2(B_1 + B_2)} \tag{4.12}$$

$$f_{m_{sine}} = \frac{v_s}{\pi(B_1 + B_2)} \tag{4.13}$$

$$f_{m_{triangle/saw}} = \frac{v_s}{4(B_1 + B_2)} \tag{4.14}$$

4.2.6 Physically Accurate Doppler Modulation

Another assumption made for simplicity of explanation thus far has been the use of the equal depth Doppler approximation given by Equation 4.2. Considering the true, physically accurate Doppler shift given by Equation 4.1, the resulting frequency modulators have a greater depth when the source is approaching the listener versus moving away. Importantly, the true Doppler shift is capable of infinite modulation depth rather than being limited to a depth equal to source frequency at the speed of sound. Section 5.1.3 shows the spectral capabilities of physically accurate Doppler with SM.

Equal Velocity, Unequal Depth

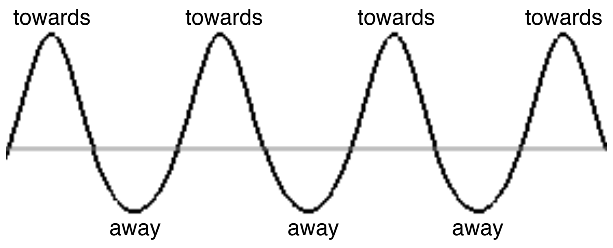


Figure 4.10: Physically Accurate Doppler FM For Low Source Velocity (100 m/s)

Implementing the true Doppler shift while maintaining equal magnitudes of velocity towards and away from the listener ($|v_s| = v_t = v_a$) gives a greater upwards shift in frequency when the source approaches the listener. Figure 4.10 shows the uneven depth of modulation when $v_t = v_a = 100m/s$. Above approximately $100m/s$ the difference of the true Doppler equation becomes more evident as seen in Figure 4.11 where $v_t = v_a = 300m/s$.

Equations 4.15 and 4.18 give the physically accurate Doppler frequency modulation

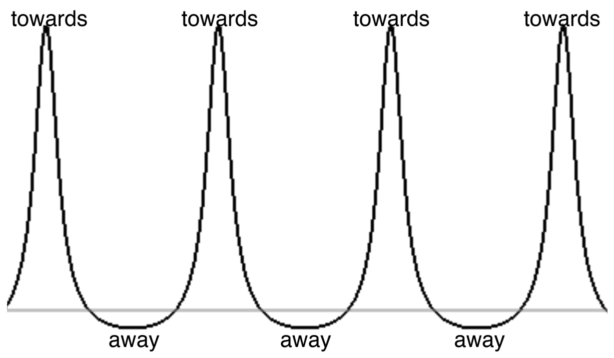


Figure 4.11: Physically Accurate Doppler FM For High Source Velocity

depth when the source is moving towards and away from the listener respectively.

$$\Delta f_{toward} = f_s\left(\frac{c}{c - v_s}\right) - f_s = \frac{f_s v_s}{c - v_s} \tag{4.15}$$

$$\Delta f_{away} = f_s - f_s\left(\frac{c}{c + v_s}\right) = \frac{f_s v_s}{c + v_s} \tag{4.16}$$

Unequal Velocity, Equal Depth

To achieve equal depth of modulation the source velocity can be slowed as it approaches the listener, $v_t < v_a$. Setting Equations 4.15 and 4.18 equal to each other allows for the calculation of v_t in terms of v_a. Since v_t approaches infinity for high speeds, v_t must be decreased as opposed to increasing v_a to achieve an equal Doppler shift towards and away from the listener.

$$v_t = \frac{v_a c}{2v_a + c} \tag{4.17}$$

$$\Delta f = \Delta f_{toward} = \Delta f_{away} = \frac{f_s v_a}{c + v_a} \tag{4.18}$$

While the modulation depth is now equal, the modulator becomes shaped by unequal

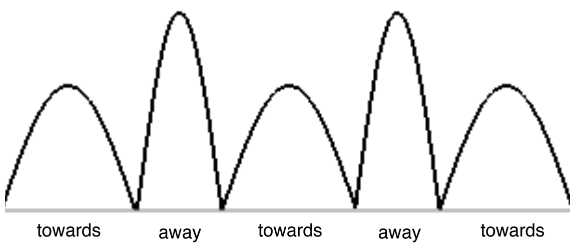


Figure 4.12: Velocity Producing Equal Depth Physically Accurate Doppler FM

duty cycles as the source moves towards and away from the listener. Figure 4.13 shows the modulation shaping (phase distortion) for $v_a = 100m/s$ and Figure 4.14 shows the same effect for $v_a = 300m/s$. Asymmetrical velocity towards and away from the listener complicates the calculation of modulation frequency, which is given for each type of motion in Appendix A.

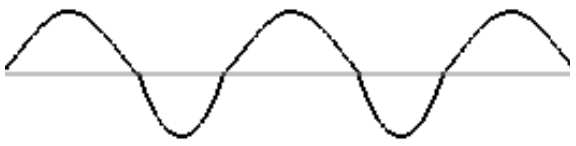


Figure 4.13: Equal Depth Physically Accurate Doppler FM for v_a = 100 m/s

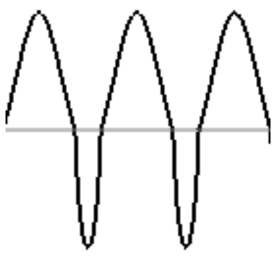


Figure 4.14: Equal Depth Physically Accurate Doppler FM for v_a = 300 m/s

4.2.7 Multi-Dimensional Motion

Expanding source velocity to a 3-dimensional vector, $\langle v_{sx}, v_{sy}, v_{sz} \rangle$, produces complex motion and modulation equivalent to parallel multiple modulator frequency modulation (PMMFM) [54]. Not only do the timbres become more complex, but the resulting 2D and 3D trajectory orbits produce complex visual trails that modulate in correlation with the changing FM parameters. When sinusoidal modulation is used the resulting orbits exhibit the complex harmonic motion of Lissajous curves[1], and other spatial modulator shapes produce similar complex curves whose motion and complexity correlate with the resulting SM timbres.

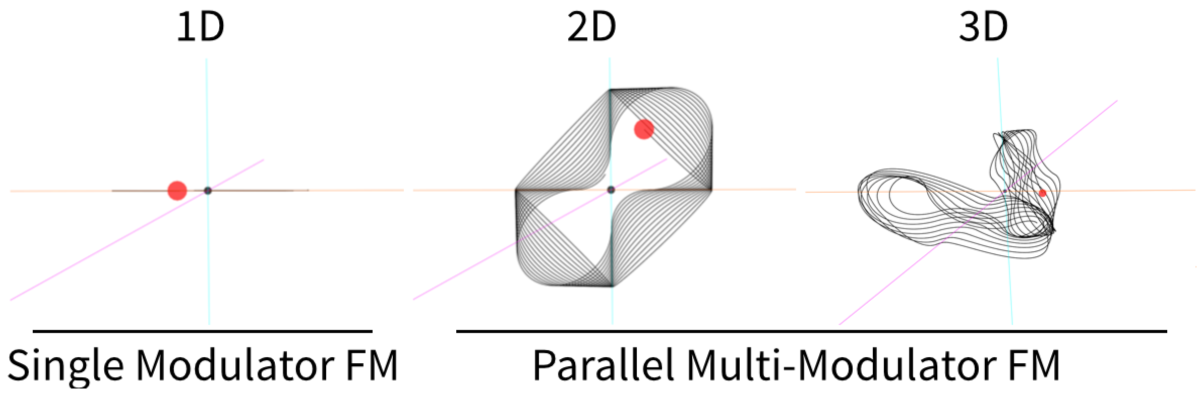


Figure 4.15: SM Multi-Dimensional Motion

Although the oscillating motion of one-dimensional Doppler FM may vary according to the waveforms and types of motion discussed in Section (ref), any visualization of the source trajectory would simply result in a straight line of varying width as seen on the left of Figure 4.15. Multi-dimensional Doppler FM not only increases the breadth of timbres possible with SM, but exposes visual harmonics in the trajectories. Figure 4.16 shows one-second snapshots of the 2D trajectory path for an orbit with equal velocity components and nearly equal boundaries.

[1]`https://en.wikipedia.org/wiki/Lissajous_curve`

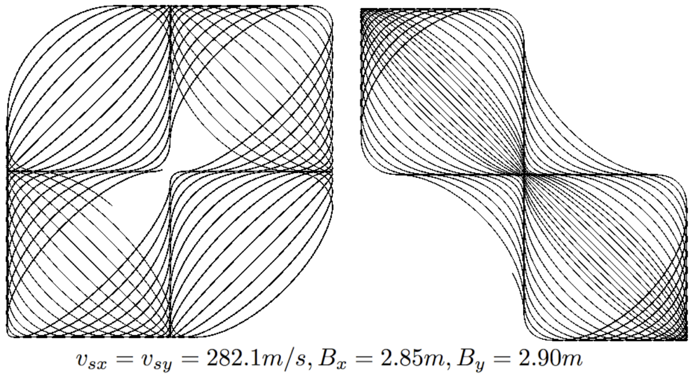


Figure 4.16: 2D Motion From Slightly Asymmetric Bounds

Along with separate velocities in each direction, individual bounds (B_x, B_y, B_z) for each dimension can be applied to literally shape the resulting timbres. SM shapes and sounds will constantly animate due to the slight differences in velocities and/or bounds, which creates slight differences in $f_{mx}/f_{my}/f_{mz}$. Figure 4.17 is an example of extending SM boundaries and velocities into 3 dimensions. Again, one-second snapshots of the same trajectory path at different times are shown. As with Lissajous figures, the ratio of modulation frequency and relative phase between each direction of motion will affect the symmetry of the trajectory.

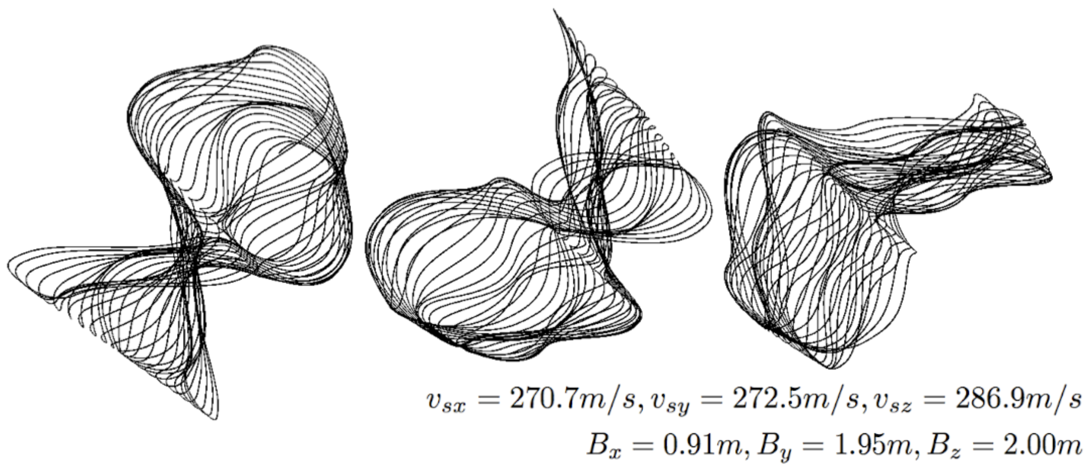


Figure 4.17: 3D Motion From Asymmetric Velocity and Bounds

If the source frequency is an integer multiple of the spatial modulation frequency in all directions, the resulting sound will be harmonic. Because SM provides a connection between modulation frequency, velocity, and bounds, the bounds and velocity of the trajectory can be adjusted in correlation with varying source frequency to maintain harmonic timbre and trajectory. Section 5.2.1 demonstrates this concept as well as several of the examples in Appendix B, and this is implemented as a critical feature of the SM plug-in described in Section 5.3. Appendix A describes how the Doppler FM formulas adjust to accommodate multi-dimensional spatial modulation (MDSM).

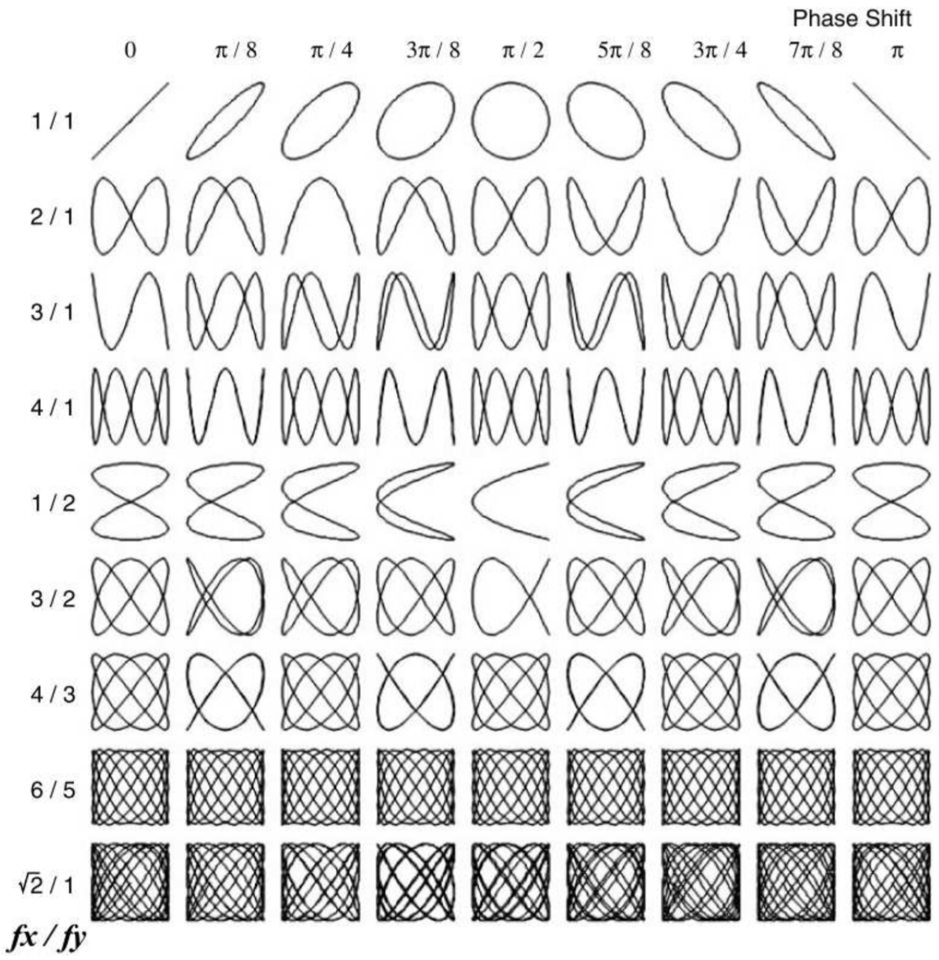


Figure 4.18: Examples of 2D Lissajous Orbits Possible with Multi-Dimensional SM[2]

[2]Image credit: `http://cns-alumni.bu.edu/~slehar/HRezBook/Chap2.pdf`

4.3 Distance-Based AM and Granulation

While Doppler shift is used to create the frequency modulation component of spatial modulation synthesis, spatialization also involves distance-based gain attenuation modulates the amplitude of a source as it moves towards and away from the listener. The amplitude envelopes produced from SM can result in low-depth tremolo at near distances to AM and granulation with increasing bounds and velocity.

Thus far, the effects of gain attenuation on sound sources as they move up to a distance B meters from the listener have been neglected. From basic physics, the Inverse-square law tells us that the intensity of sound is inversely proportional to the square of the distance of the source to the listener. However, for digital audio, sound amplitudes are typically attenuated simply as the inverse distance as that tends to sound more natural. The implementation of SM (Section 5.3) provides both attenuation options. For one-dimensional oscillating motion the gain attenuation from SM would produce amplitude modulation of depth $1 - 1/B$ or $1 - 1/B^2$ using either attenuation policy respectively. The frequency of AM is equivalent to that of the corresponding Doppler FM.

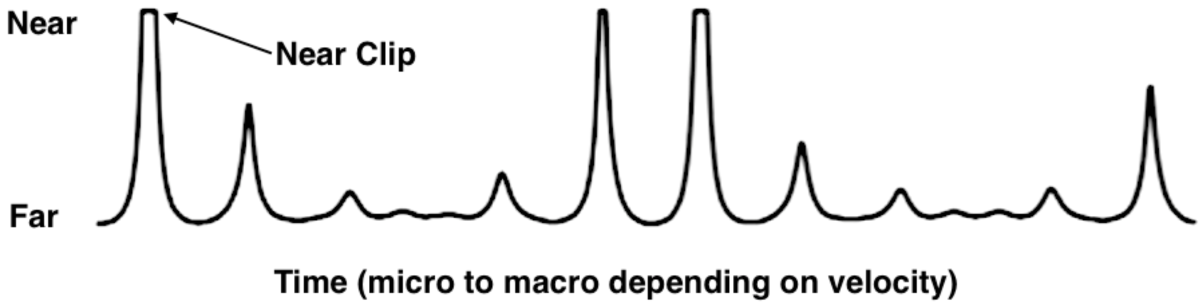


Figure 4.19: Amplitude Envelope from Distance-Based Gain Attenuation

For distances less than 1 meter, gain would rise to ∞ rather than attenuate, so the amplitude of the source is not modulated within 1 meter (the "near-clip") of the listener, though modulation from multi-channel panning may be present (Section 4.4). As the bounds and resulting depth of modulation increase, the AM becomes a pulse train envelope (Figure 4.19) with perceptible moments of silence at far distances though the amplitude never truly reaches 0. With extremely high velocities, large bounds, and multi-dimensional movement, these pulse trains resemble asynchronous granular envelopes. Sections 5.2.2 and 5.2.3 describe the technique of spatial granulation as SM combines these granular envelopes with multi-channel spatialization.

4.4 Spatialization

The final component of SM is a spatialization algorithm for multi-channel amplitude panning. The spatialization algorithm may be as simple as equal power stereo panning or expanded to full 3D panning via DBAP, VBAP, or Ambisonics. Any form of multi-channel panning will add additional AM and spread effects to SM timbres. Due to the variety and complexity of the aforementioned panning algorithms, it is difficult to quantify their added modulation effects. Suffice it to state that at low SM velocities multi-channel panning will result in perceptible localization of sound over a loudspeaker array, while at high velocities the rapid panning will spatially smear the sound across the array – be it a pair of headphones or a large venue with several loudspeakers. Figure 4.20 shows an example of how SM may rapidly modulate amplitude across 16 channels, specifically as a sound source increases velocity within a fixed bounds over a duration of 30 seconds. This and other spatially complex effects can typically be accomplished using a single slider on the SM plug-in (Sections 5.2 and 5.3).

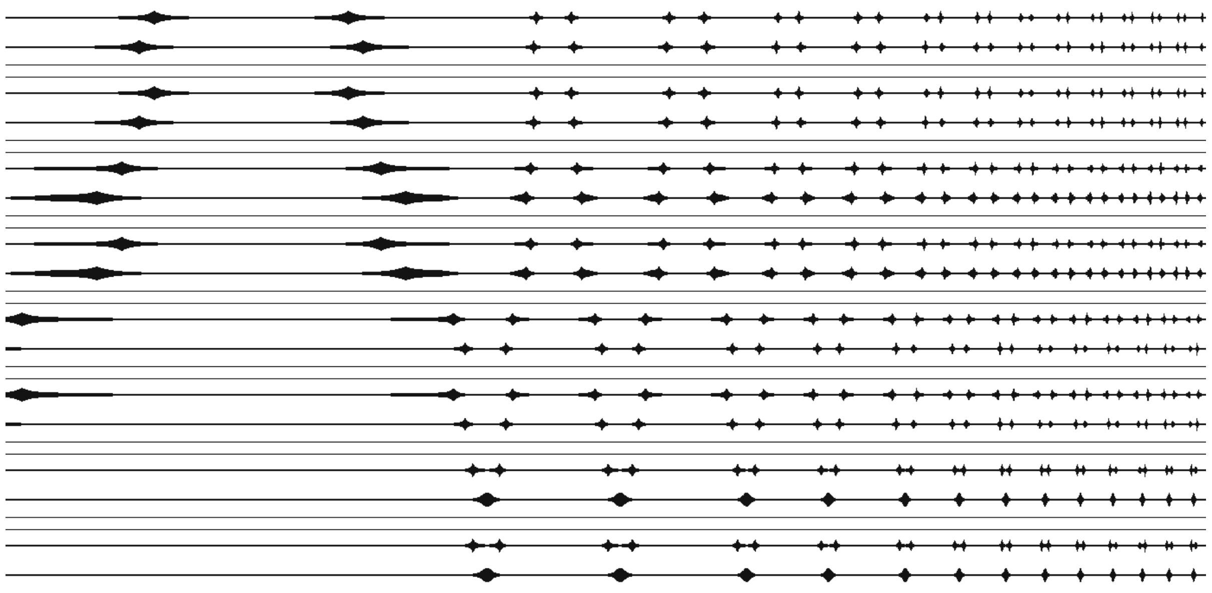


Figure 4.20: Spatialization of SM Waveform Over 16 Channels

Chapter 5

Evaluation of Spatial Modulation Synthesis

This chapter begins with a comparison of Doppler FM to traditional FM, which discusses the physical and technical limits of SM along with its potential for replicating typical FM timbres. Next, novel spatial and visual control techniques made possible with SM are demonstrated, followed by a description of the software implementation of SM as a versatile audio plug-in useful for spatialization and synthesis in arbitrary venues. Lastly, the novel software rendering paradigm developed to realize SM is introduced as a high-resolution approach to the control and visualization of spatial audio trajectories.

Several demo videos of SM presets and a stereo VST plug-in are online at `http://www.spatialmodulation.com`

5.1 Comparison to FM

Using sinusoidal motion (Equation 4.8), a sinusoidal source, and the symmetrical Doppler approximation (Equation 4.2) it is possible to replicate classic sinusoidal carrier and modulator FM with SM. Figure 5.1 compares the Doppler FM component of SM to classic 2 oscillator FM synthesis. Unlike FM, there is no direct control for modulation depth and frequency as those parameters become are by the source's velocity and bounds of motion. Because SM uses a modulating delay line to shift source frequency, it can operate on arbitrary sound source input, though for comparison to classic FM, we consider the sound source to be a sinusoidal oscillator of frequency, f_s.

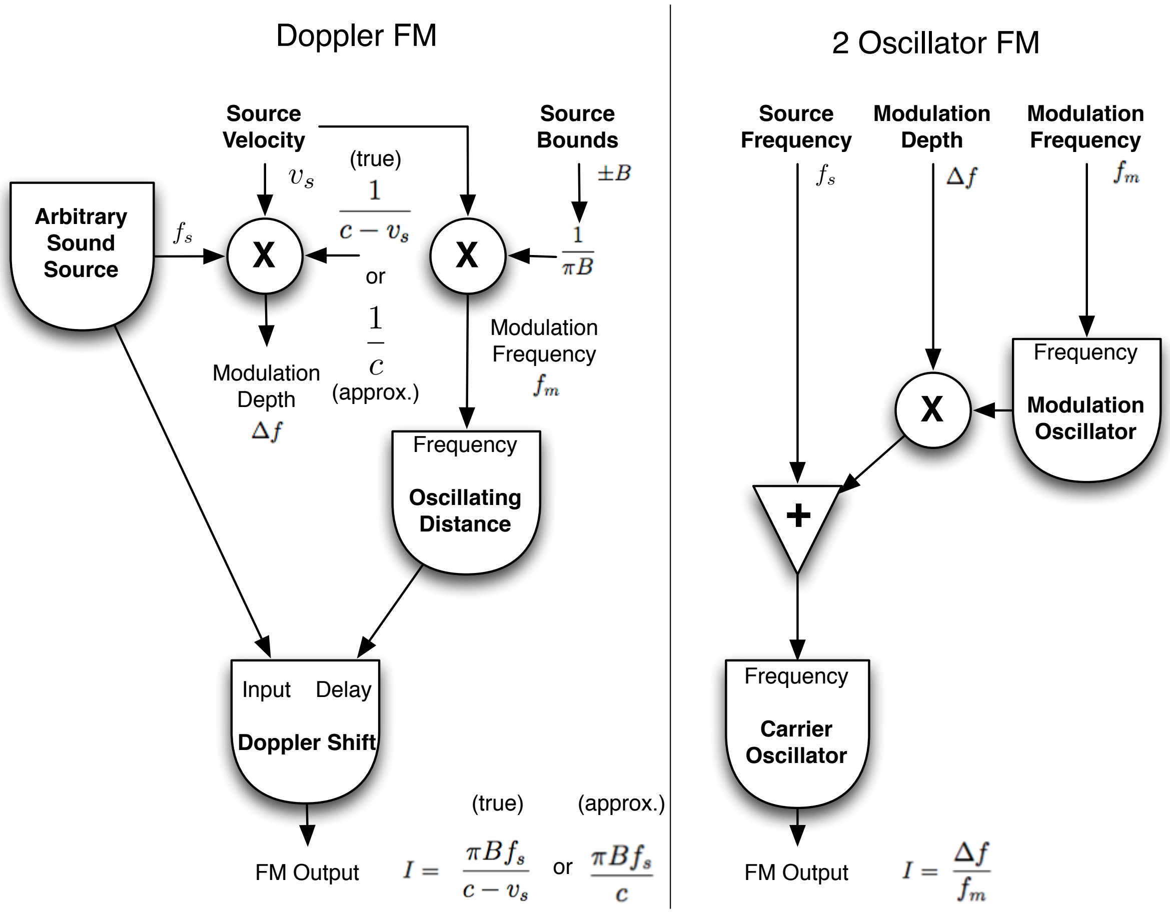


Figure 5.1: Doppler FM Components Compared to 2 Oscillator FM

While classic FM has independent parameters of modulation depth and modulation frequency, SM has independent parameters of source velocity and bounds of source motion. The source velocity can be simulated up to the speed of sound (SOS) (340 m/s) and the minimum bounds of motion is around 0.1 meter depending on the sample rate used. Here we analyze the ranges of velocity and bounds and their impact on modulation depth, index, and frequency. As with FM, carrier to modulator ratio is also an important parameter of SM and is analyzed in Section 5.2.1.

5.1.1 Modulation Frequency

Using SM, Modulation frequency will be maximum when the velocity is maximum and the bounds are minimum (Equation 4.9). Vibrato will begin to morph into FM tones when the resulting modulation frequency exceeds 20 Hz. For sinusoidal, one-dimensional SM within a ± 1 meter bounds this will occur at around $65m/s$ ($\approx$ 145mph). Increasing velocity to the speed of sound (SOS) ($\approx$ 760mph) will give a maximum modulation frequency of 108 Hz within a 1 meter bounds.

Not only do the actual physics of sound transmission break down at the SOS, but computational problems are encountered as well. In reality, a sonic boom occurs at the SOS, which corresponds to the observation that Δf goes to ∞ for $v_s = c$ using the true Doppler shift (Equation 4.1). Moving beyond the SOS, Doppler shift will alias frequencies with negative depth until the effective modulation depth nets to 0 at twice the SOS (Mach 2). While a corresponding digital implementation of the formula would be correct with zero frequency shift at Mach 2, in reality the sound would be heard in reverse since the source is moving faster than its sound waves. The complications of accurately simulating sound sources past the SOS is of future interest, but for now we turn to minimizing bounds to maximize modulation frequency. Also, given that infinite modulation depth can be produced at the SOS using the true Doppler shift formula (Section 5.1.2), there is no need to simulate velocities past the SOS if bounds can be minimized.

The limits of bounds are dependent on the the sample rate used for processing SM. Even though the delay line modulation producing Doppler shift may be controlled at sample rate, the sample rate itself imposes a minimum distance to sample related to the speed of sound. For example, at a sampling rate of 44.1 kHz the distance to sample (distance corresponding to 1 sample of delay) is $340/44100 = 0.00771$ meters. For a $\pm$ 1 meter bounds rendered at 44.1 kHz, the source's frequency would modulate over a total of 2 meters or $2/0.00771 \approx 260$ samples. This is the number of samples available to render one cycle of the modulator. At 44.1 kHz using less than $0.1 meters \approx 26$ samples there is not enough resolution available to accurately represent the modulator and, due to phase noise, the resulting SM spectra stray from the predicted FM values. Figure 5.2 shows the loss of modulator resolution with decreasing bounds. Doubling the sample rate will in turn double the amount of available resolution, so the minimum usable bounds for SM at 96 kHz would be 0.05 meters. Table 5.1.1 summarizes the maximum modulation frequency values for SM using various modulators and sample rates.

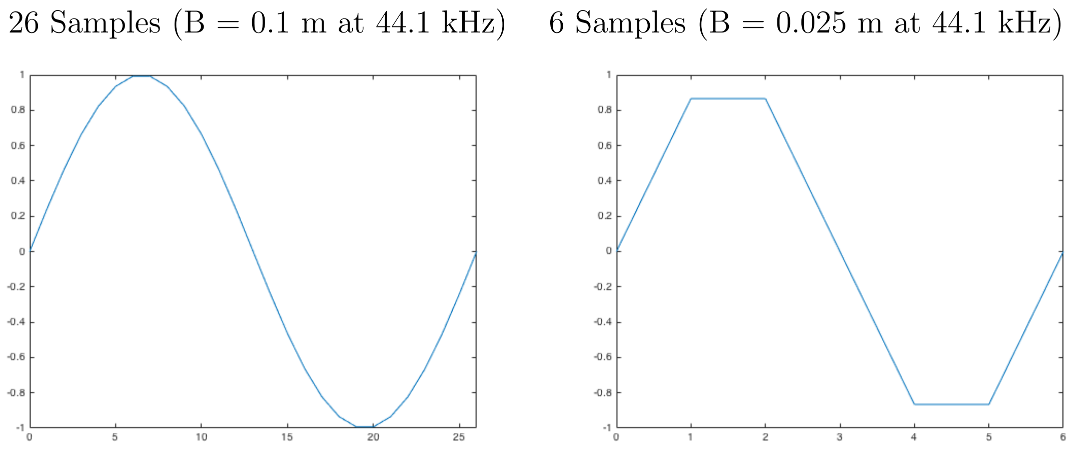


Figure 5.2: Effect of Decreasing SM Bounds on Frequency Modulator

	Max F_m (Square)	Max F_m (Sine)	Max F_m (Tri/Saw)	Min B
44.1 kHz	1700 Hz	1083 Hz	850 Hz	0.1 m
48 kHz	1850 Hz	1178 Hz	925 Hz	0.11 m
96 kHz	3700 Hz	2356 Hz	1850 Hz	0.05 m
192 kHz	7400 Hz	4712 Hz	3700 Hz	0.025 m

Table 5.1: Maximum Modulation Frequencies and Minimum Bounds for SM Plug-in

5.1.2 Modulation Depth and Index

A critical property of FM synthesis is the modulation index, which is the proportion of modulation depth to modulation frequency and indicates the number of sidebands present in the spectrum. Using SM, due to the physics of Doppler shift, the source or carrier frequency, f_s, is directly linked to the index of modulation. This direct relationship between carrier frequency and modulation index is a departure from typical FM in which the two parameters are controlled separately or even varied inversely [54] [55]. However, a direct relationship between carrier frequency and modulation index was observed by Lazzarini et. al in implementing delay-line phase modulation suitable for Adaptive FM (ADFM) [56], a variety of FM that tracks the fundamental frequency of a complex carrier in order to dynamically adjust the modulation frequency to maintain a specific carrier to modulator ratio.The Doppler simulation in SM is also technically a form of phase modulation since a modulatiing delay-line is used to render a source at a given distance from a listener. Thus, while SM may not be capable of covering the entire timbral space of classic FM, it has the ability to operate on both simple and complex carrier signals.

Modulation index, I, using the symmetrical Doppler approximation within symmetrical bounds ($B = B_1 = B_2$) (Section 4.2.3) is shown by Equation 5.1. Interestingly, v_s cancels out of the result, meaning that, using the symmetrical Doppler approximation, modulation index will be constant for a given bounds since both the frequency and depth of modulation vary linearly with v_s.

$$I_{DopplerApprox} = \frac{\Delta f}{f_m} = \frac{\frac{v_s}{c} f_s}{\frac{v_s}{\pi B}} = \frac{\pi B f_s}{c} \tag{5.1}$$

For low values of $v_s < 100m/s$ Equation 5.1 will hold for a true Doppler implementation as well. However, with increasing velocity, the non-linearity of the true Doppler shift becomes more apparent and allows for higher modulation indices that vary based on both velocity and bounds. To compute the true Doppler index the highest modulation depth as the source approaches the listener is used (Equation 4.15) resulting in Equation 5.2 for sinusoidal motion. Appendix A contains index formulas for all variations of motion and Doppler type.

$$I_{DopplerTrue} = \frac{\Delta f_{toward}}{f_m} = \frac{\pi B f_s}{c - v_s} \tag{5.2}$$

	Max $\|\Delta f\|$	Max I
Symmetrical Doppler	f_s	$\frac{f_s}{f_m}$
True Doppler	∞	∞
True Doppler (=)	$\frac{f_s}{2}$	$\frac{f_s}{2f_m}$

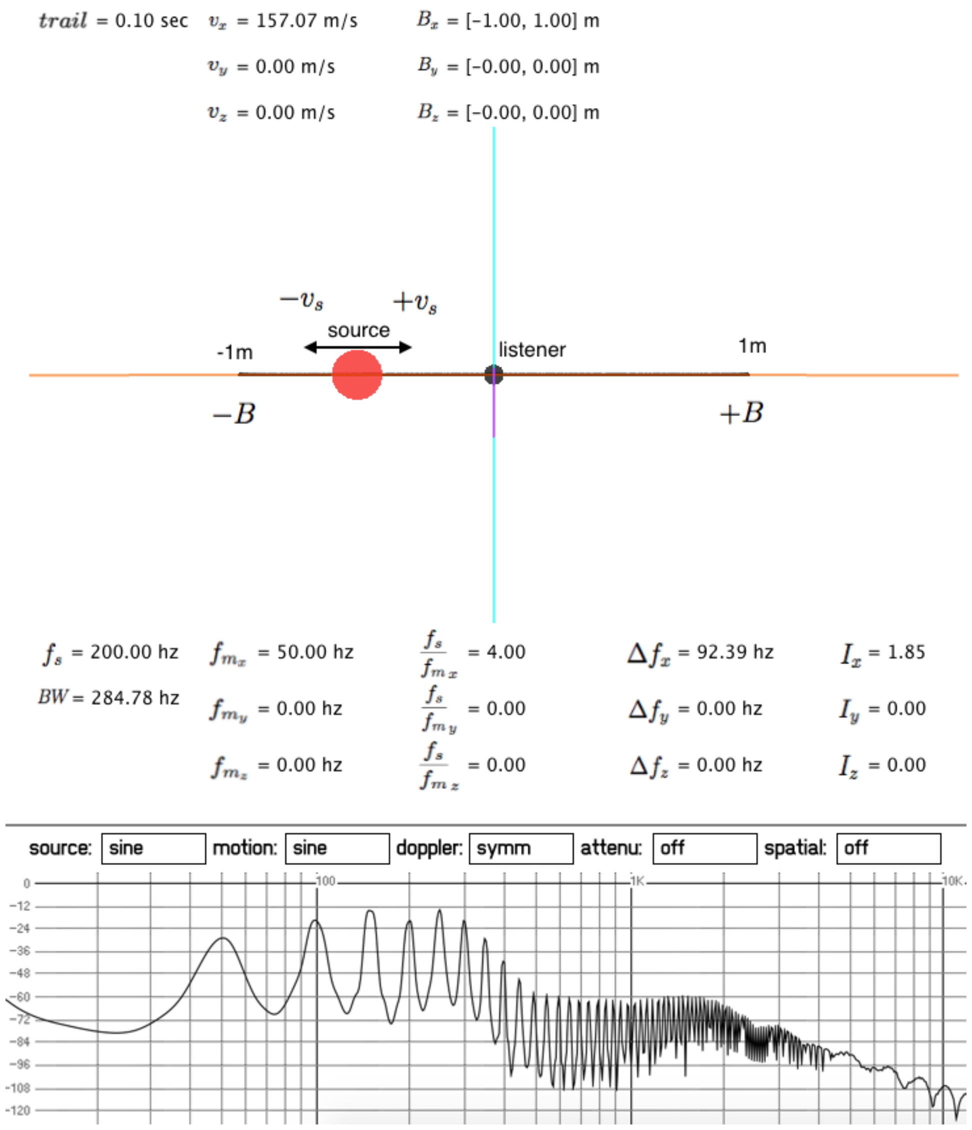
Table 5.2: Maximum Modulation Depth Values for SM

Maximum modulation depths and indices vary according to the Doppler simulation used as shown by Table 5.1.2. The Doppler approximation gives a symmetric depth modulator ideal for replicating FM timbres, but has a maximum depth limited to the carrier's frequency. This implies that the maximum modulation index for the symmetric Doppler approximation is identical to the carrier to modulator ratio. The true Doppler simulation allows for infinite upwards modulation depth, but is asymmetrical with a downwards depth limited to one half the carrier frequency. Lastly, the velocity-varying, equal depth, true Doppler simulation (Section 4.2.6, indicated by (=)) provides a symmetric depth with physical accuracy, but is limited in depth to one half of the carrier frequency. Thus, true Doppler, though producing asymmetrical modulation depth, must be used to cover the range of FM timbres with high indices of modulation.

For instance, using the typical symmetrical Doppler approximation with SM, simulating a brass-like sound with $I = 5$, $f_s : f_m = 1$, and $f_s = 440$ would require a bounds of $\pm$ 0.615 meters, but a source velocity of 1700 m/s, 5 times higher than the maximum speed of 340 m/s. However, an oboe sound with $I = 2$, $f_s : f_m = 3$, and $f_s = 440$ could be produced with $B = 0.25$ meters and $v_s = 230$ m/s. On the other hand, using a true Doppler implementation, it is possible to simulate a brass-like sound because a modulation index of 5 can be obtained from a speed of 272 m/s within the same bounds. As spectral analysis will show in Section 5.1.3, the asymmetry of the modulator resulting from true Doppler does not prohibit the replication of high index FM timbres.

5.1.3 Spectral Analysis

Figure 5.3 shows the initial spectral result of one-dimensional SM using the Doppler approximation. Bounds of ± 1 meter were used so that gain attenuation effects (Section 4.3) would not be present. Spatialization rendering (Section 4.4) was disabled as well, though the spectra would reflect the sum of all channels if a form of multi-channel panning was enabled.



$trail$ = 0.10 sec v_x = 157.07 m/s B_x = [-1.00, 1.00] m
v_y = 0.00 m/s B_y = [-0.00, 0.00] m
v_z = 0.00 m/s B_z = [-0.00, 0.00] m

f_s = 200.00 hz f_{m_x} = 50.00 hz $\frac{f_s}{f_{m\,x}}$ = 4.00 Δf_x = 92.39 hz I_x = 1.85
BW = 284.78 hz f_{m_y} = 0.00 hz $\frac{f_s}{f_{m\,y}}$ = 0.00 Δf_y = 0.00 hz I_y = 0.00
f_{m_z} = 0.00 hz $\frac{f_s}{f_{m\,z}}$ = 0.00 Δf_z = 0.00 hz I_z = 0.00

source: sine motion: sine doppler: symm attenu: off spatial: off

Figure 5.3: One-dimensional SM and Resulting Spectrum

Dynamic Bandwidth Filter

Though the spectrum in Figure 5.3 shows partials as expected for computed FM parameters, there is a significant amount of phase distortion noise present in the higher frequencies. This high frequency noise is the result of artifacts from the modulating delay line used to implement SM. Although a 1 meter bounds allows for enough samples to accurately represent sinusoidal motion without interpolation (Section 5.1.1), interpolation is necessary to read samples from a delay line at fractional indices derived from the source's sinusoidally modulating distance. The presence of phase distortion resulting from interpolated delay line reads is well documented [57]. However, this high frequency noise can be suppressed using a steep low-pass filter that dynamically adjusts its cutoff frequency based on predicted bandwidth. Carson's rule states that 98% of FM signal power will be contained within a bandwidth equal to twice the sum of the modulation depth and frequency (Equation 5.3) [58]. The implementation of SM utilizes a steep low-pass filter with a cutoff frequency (Equation 5.4) that varies with velocity and bounds according to the bandwidth continuously computed using Carson's law. As Figure 5.4 shows, this filter is critical to suppress artifacts from delay line interpolation.

$$BW = 2(\Delta f + f_m) \tag{5.3}$$

$$Cutoff = f_s + \Delta f + f_m \tag{5.4}$$

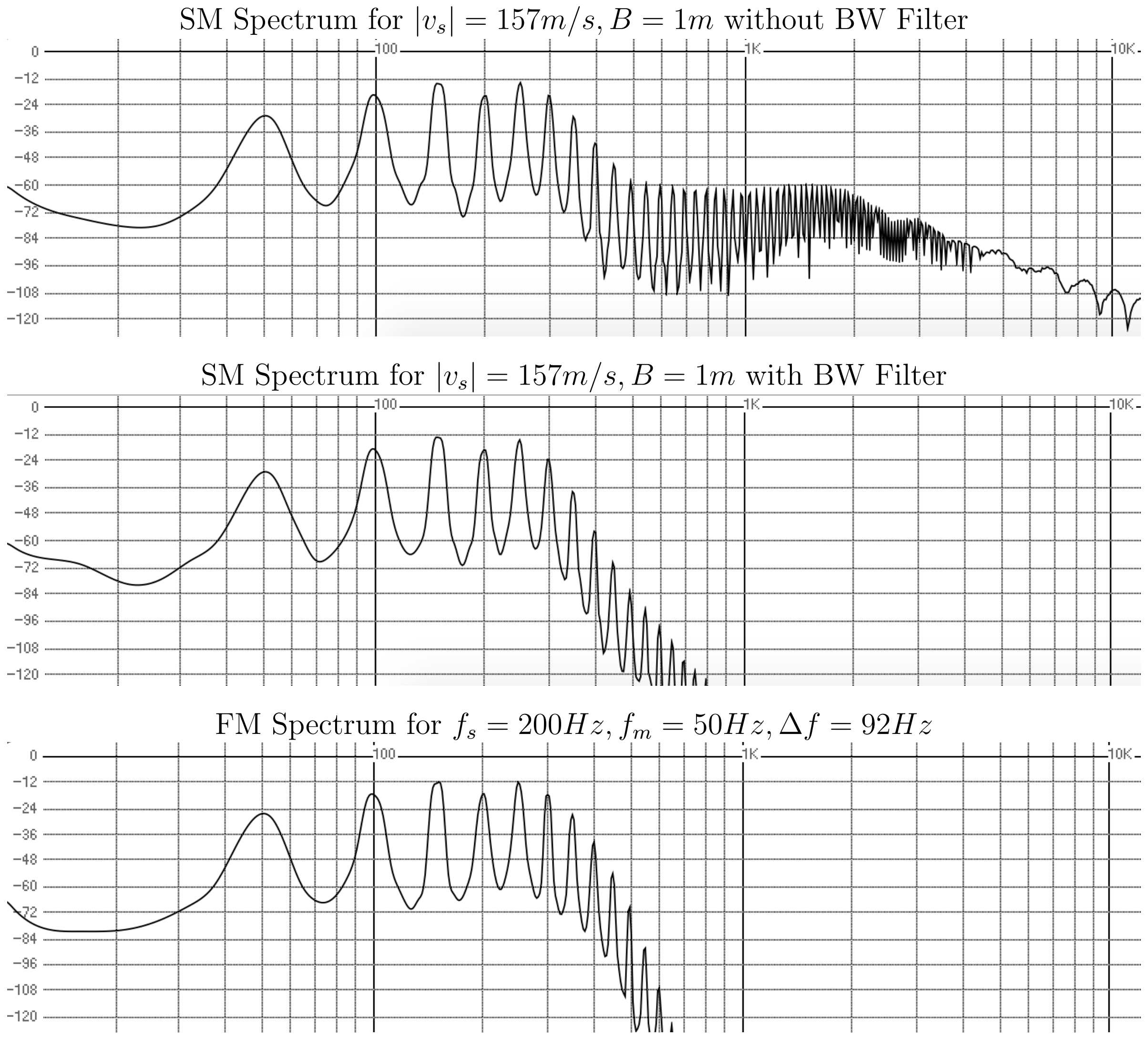


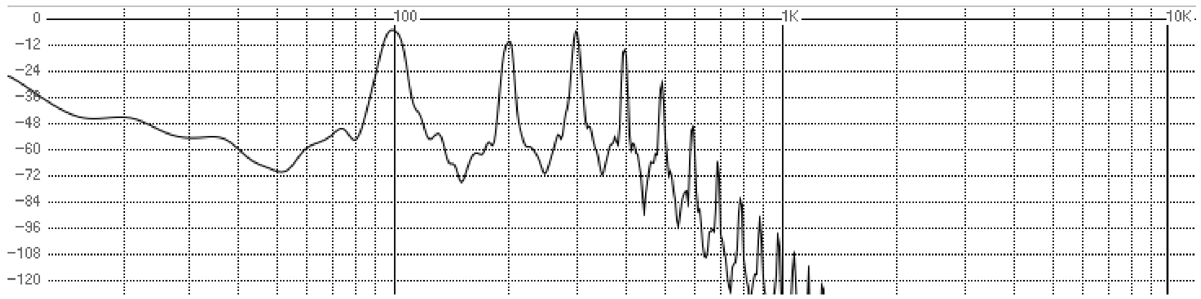
Figure 5.4: Effect of Dynamic Bandwith Filter and Comparison to FM at 157 m/s

Figure 5.4 also shows that the use of a dynamic lowpass filter placed at the predicted FM bandwidth according to Carson's rule allows SM to replicate spectra generated from a classic 2 oscillator FM instrument.

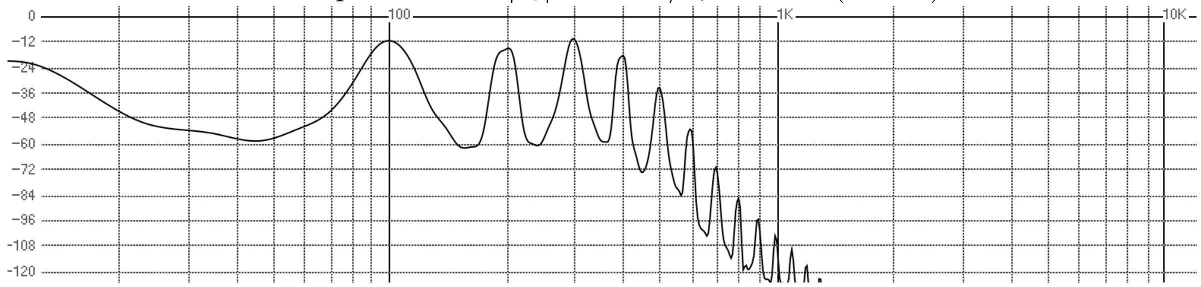
Increased Sampling Rate

However, as velocities increase and approach the speed of sound, low-level amplitude noise becomes present around the SM partials as seen in Figure 5.5. Here we see that increasing the sampling rate allows SM to render a cleaner set of partials with less noise at high speeds to again replicate typical FM. Though various forms of linear, cubic, and all-pass interpolation [57] were used in attempt to minimize amplitude noise around partials, none were as effective as increasing the sampling rate.

SM Spectrum for $|v_s| = 314m/s, B = 1m$ (44.1kHz)



SM Spectrum for $|v_s| = 314m/s, B = 1m$ (96kHz)



FM Spectrum for $f_s = 200Hz, f_m = 100Hz, \Delta f = 185Hz$ $(44.1kHz)$

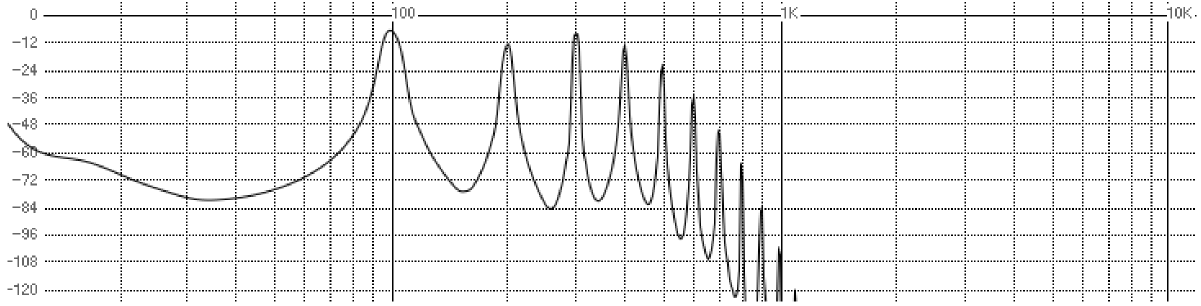


Figure 5.5: Comparison of FM to SM at 314 m/s

True Doppler

Figure 5.6 shows true Doppler spectra using the same velocity and bounds as the previous example using the Doppler approximation. We can observe that the true Doppler implementation results in a much greater modulation depth and index at equivalent speeds. We can again see the benefit of using a higher sampling rate and the ability of SM to replicate FM even though true Doppler produces asymmetrical depth modulation (Section 4.2.6).

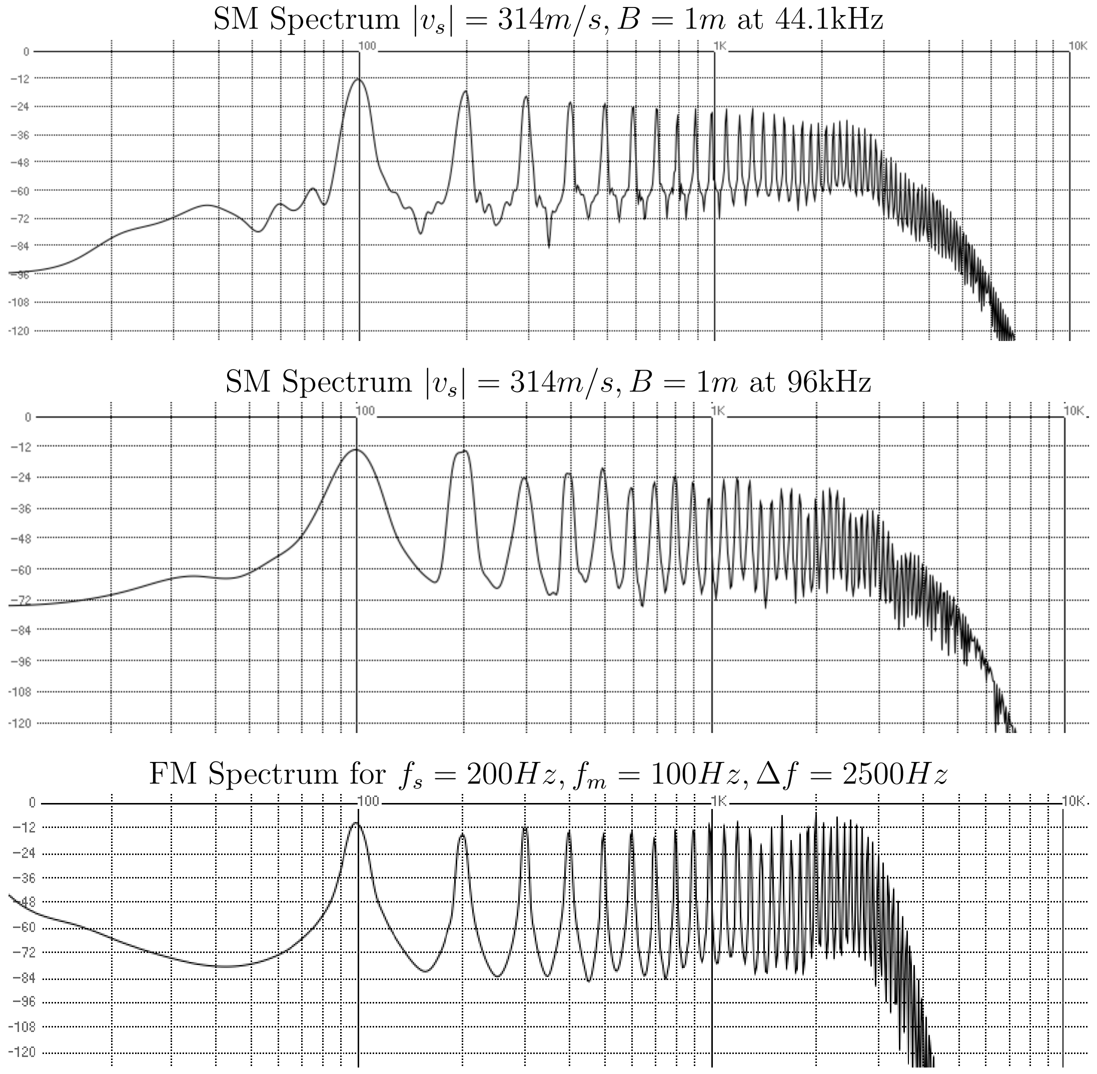


Figure 5.6: Comparison of SM Using True Doppler and FM Spectra

5.2 Spatial Synthesis Control

The novelty of spatial modulation synthesis lies not in the sounds it can produce, but in the control paradigm through which it synthesizes sounds. Parameters of velocity and bounds to control a 3-dimensional spatialization model based on the physical propagation of sound including Doppler shift and distance-based gain attenuation to produce a continuum from simple panning and spatial effects at low velocities to FM tones and granulation at high velocities. Simple adjustments of bounds can produce complex spatial effect that would be tedious to synchronize using other techniques.

	SM Velocity	**SM Bounds**
FM	Depth and Frequency	Frequency
AM	Frequency	Depth and Frequency
Granulation	Duration and Density	Duration and Density
Spatialization	Position	Room Size and Spread

Table 5.3: Control of Synthesis Using Spatial Modulation

As interesting as the timbres possible with SM are the resulting shapes of the sound source trajectories. Velocity and bounds simultaneously control a graphical synthesis of a sound's orbit or 3-dimensional trajectory around a listener. Just as it's possible for SM timbres to morph between different categories of synthesis, the resulting orbits can morph from dense, symmetric shapes to sparse, asymmetric trails.

5.2.1 Spatial Pitch and Timbre

Section 4.2.7 introduced the concept that harmonic relationships between SM timbres and their Lissajous trajectory curves will occur when the source frequency is an integer multiple of the modulation frequency in each direction of spatial modulation. A fundamental property of FM is that integer ratios of the carrier to modulator frequencies will produce harmonic spectra [55]. Likewise, by using symmetrically bounded SM, harmonic spectra can be predicted by the following equations for each modulator shape.

$$\frac{f_s}{f_{m\,square}} = \frac{2Bf_s}{v_s} \implies B_{square} = \frac{v_s}{2f_m} \implies v_{s_{square}} = 2Bf_m \tag{5.5}$$

$$\frac{f_s}{f_{m\,sine}} = \frac{\pi Bf_s}{v_s} \implies B_{sine} = \frac{v_s}{\pi f_m} \implies v_{s_{sine}} = \pi Bf_m \tag{5.6}$$

$$\frac{f_s}{f_{m\,tri/saw}} = \frac{4Bf_s}{v_s} \implies B_{tri/saw} = \frac{v_s}{4f_m} \implies v_{s_{tri/saw}} = 4Bf_m \tag{5.7}$$

Equations 5.5-5.7 also show that it is possible to compute the required bounds (given velocity) or velocity (given bounds) to maintain a fixed modulation frequency ratio regardless of source frequency. This is significant because it creates a relationship between pitch (source frequency) and size (bounds) or speed (velocity). Higher pitches will produce smaller trajectory orbits for fixed velocities or require faster velocities around a fixed size orbit. Likewise, lower pitches will have larger and slower orbits respectively. The software implementation of SM (Section 5.3) implements an optional parameter lock for either velocity or bounds in each direction in order to provide an adjustable carrier to modulator ratio. Using a frequency controllable oscillator (rather than complex input) with a lock enabled produces changes in either bounds or velocity in order to maintain the desired source to modulator ratio in each direction of modulation (Figure 5.7).

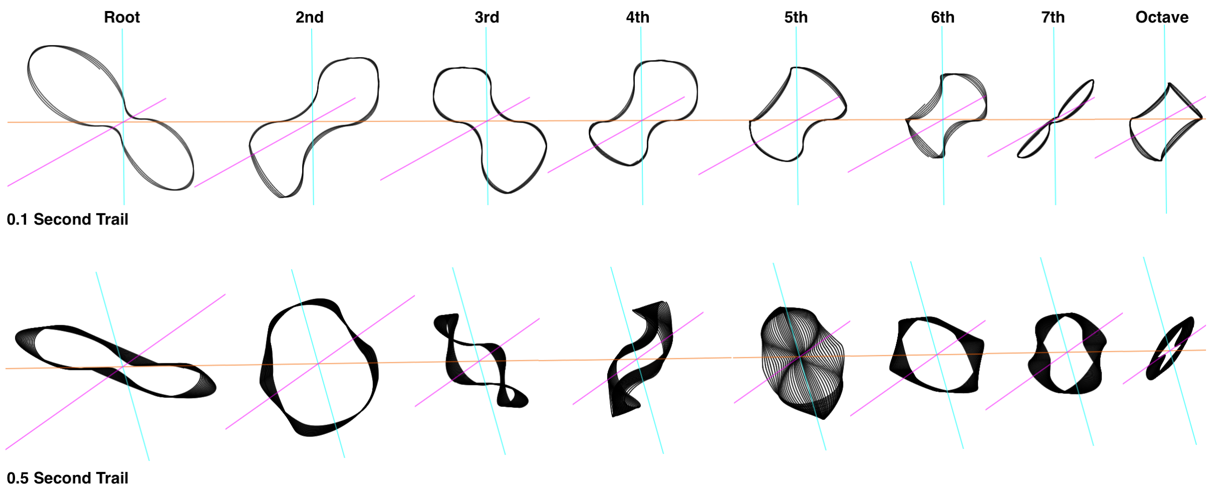


Figure 5.7: Octave of SM Bounds for Fixed Velocity and Source to Modulator Ratio

The selection of several motion oscillators (Section 4.2.4) coupled with independent control of carrier to modulator ratio in each direction (Section 4.2.7) also produces relationships between timbre and the overall shape of the trajectory. Changing the motion modulation type while maintaining fixed carrier to modulator ratios in each direction (Figure 5.8) produces shape timbre relationships. For instance, Square wave SM produces harsher sounds with more energy in high frequency partials than sine wave SM.

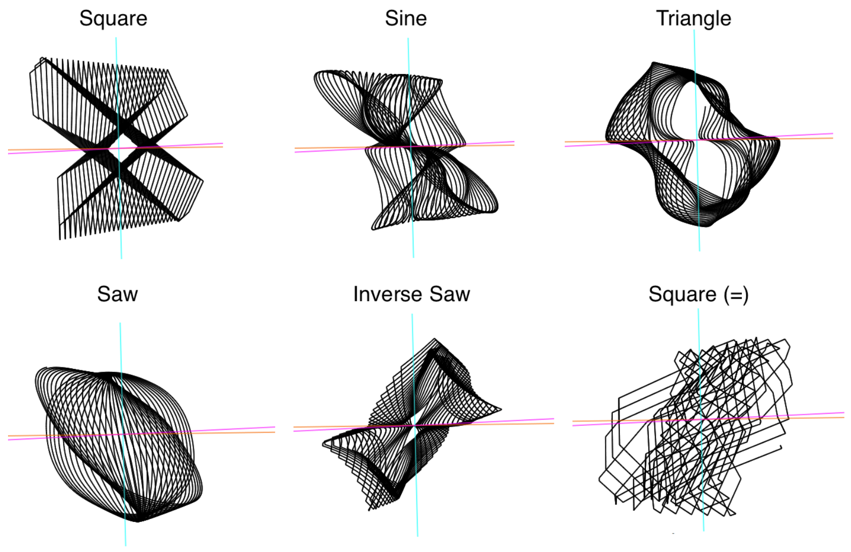
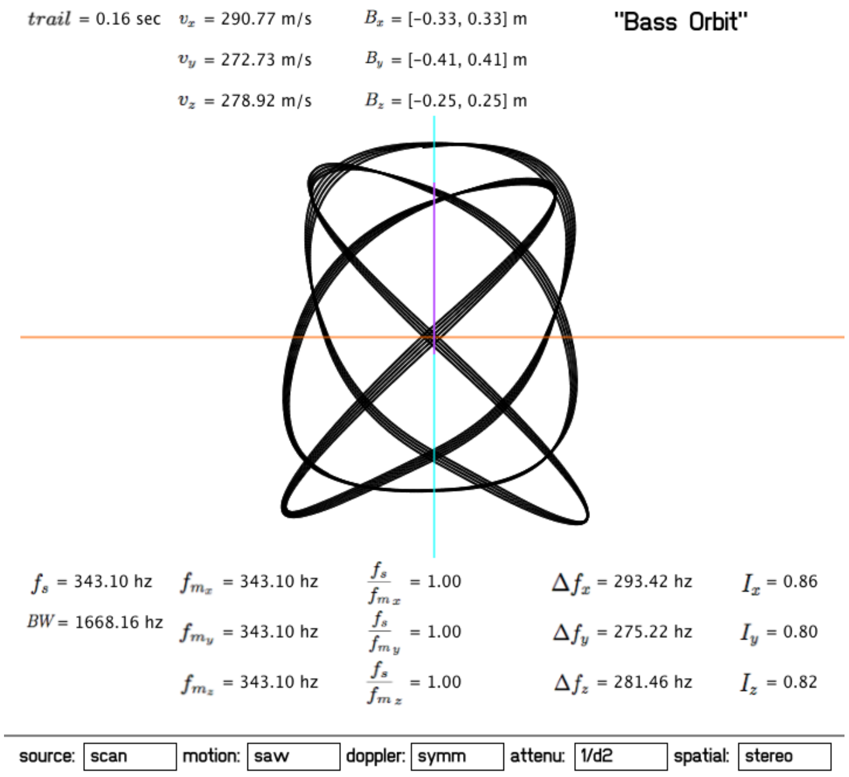


Figure 5.8: Effect of Changing Modulation Shape for Given SM Orbit

Feedback SM

Because SM trajectories are computed at sample rate, the orbits themselves can be scanned as the source to be spatially modulated, resulting in feedback SM (FBSM) which always has a 1:1 source to modulator ratio. Th instantaneous distance from the sound source to the listener is normalized according to the bounds of the orbit and read as audio samples which are individually positioned according to the azimuth, elevation, and radial distance of the source in meters.


"Bass Orbit"

$trail$ = 0.16 sec

v_x = 290.77 m/s
v_y = 272.73 m/s
v_z = 278.92 m/s

B_x = [−0.33, 0.33] m
B_y = [−0.41, 0.41] m
B_z = [−0.25, 0.25] m

f_s = 343.10 hz
BW = 1668.16 hz

f_{m_x} = 343.10 hz
f_{m_y} = 343.10 hz
f_{m_z} = 343.10 hz

$\frac{f_s}{f_{m\,x}}$ = 1.00
$\frac{f_s}{f_{m\,y}}$ = 1.00
$\frac{f_s}{f_{m\,z}}$ = 1.00

Δf_x = 293.42 hz
Δf_y = 275.22 hz
Δf_z = 281.46 hz

I_x = 0.86
I_y = 0.80
I_z = 0.82

source: scan motion: saw doppler: symm attenu: 1/d2 spatial: stereo

Figure 5.9: A Feedback SM Preset: Bass Orbit

Spatial Modulation Cookbook

Appendix B contains a library of several shape/timbre recipes using SM.

5.2.2 Spatial Granulation and Rhythm

Whereas Doppler FM requires extreme velocities within relatively small bounds, spatial granulation requires both extreme distances and extreme velocities. By simply extending the boundary of a Doppler FM sound, it can morph into a series of spatial sound particles. With larger boundaries imposing greater gain attenuation with distance, the amplitude modulation effects described in section 4.3 become short duration amplitude envelopes that will effectively silence the sounds at large distances while returning to full amplitude near the listener. When the velocity of source movement is sufficiently high, the duration of the resulting amplitude envelopes can enter the granular domain ($\leq$ 100 ms). Oscillating movement within a bounds produces a train of these envelopes that varies in duration, shape, and spatial location as the source moves in multiple dimensions.

The maximum distance used in the implementation of SM is currently 100 meters, which is far enough to perceptually silence sounds using the inverse-square gain attenuation law. The minimum distance for gain attenuation to occur is 1 meter and the frequency of amplitude modulation will equal that of Doppler frequency modulation for a given bounds, velocity, and motion type. When the amplitude modulation frequency is maximum ($B = 1m$ and $v_s = 340m/s$) the depth is minimum and inaudible. AM becomes audible with sufficient depth beginning at 2 meters using the inverse squared law, so the practical maximum frequency for amplitude modulation due to distance attenuation is around 54 Hz using sinusoidal motion or 85 Hz using square motion. As the bounds increase to 100 meters the AM slows to around 0.5 Hz, but the steep gain-attenuation morphs into a series of discrete granular envelopes with a minimum duration of approximately 100ms using square motion at the speed of sound. Figure 5.10 shows the granular envelope that results from distance-based gain attenuation as a sound source oscillates between bounds expanding to a maximum 100 meters in one dimension using square wave motion.

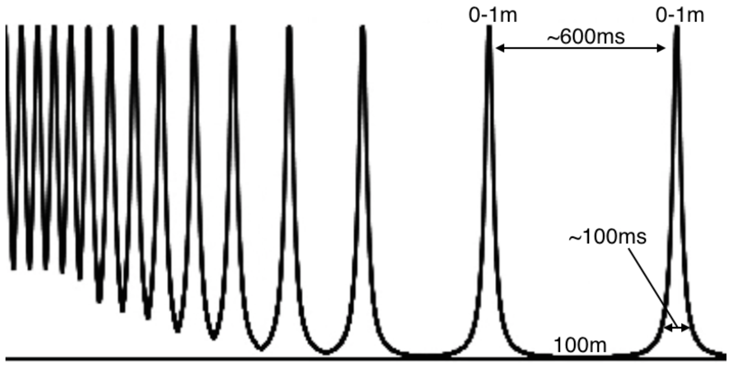


Figure 5.10: AM to Granulation as Bounds Approach 100m at 340m/s

Given the combined effect of Doppler shift, the resulting sound becomes especially similar to glisson synthesis, a form of granular synthesis in which each sound particle is considered a vector of duration, frequency, and amplitude [59]. More generally, the result could be termed a form of grainlet synthesis, another granulation technique in which one parameter of a sound grain is linked to another parameter [59]. In the case of SM, velocity links grain frequency to duration. Polyphony of spatial modulation synthesis can produce several simultaneous granular streams or grain clouds. The density of such clouds can be altered by adding voices or adjusting the velocity or bounds of spatial modulation.

When using high-velocity multi-dimensional spatial modulation, spatial rhythms and synthetic drum sounds can be produced as the amplitude envelope pulse trains gain a larger duty cycle with more complex varying modulation depth. Figure 5.11 shows a 3 second snapshot of the amplitude envelope produced from a "Spatial Drums" SM preset (Figure B.3). Using an 80 Hz triangle wave as input, the resulting sound is reminiscent of a 60 bpm Roland 808 kick drum with a short delay.

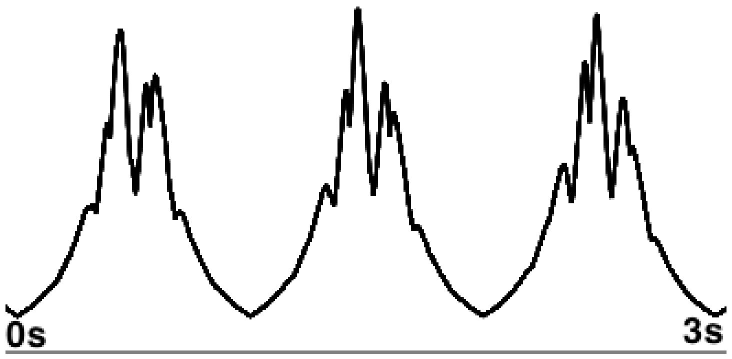


Figure 5.11: Spatial Rhythm Amplitude Envelope

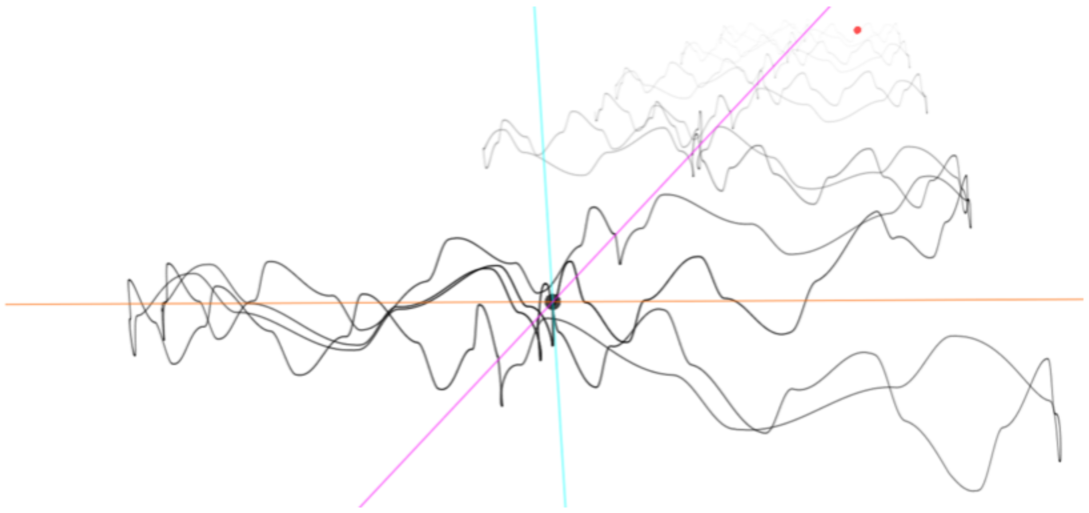


Figure 5.12: Spatial Granulation Trajectory

Figure 5.12 shows an example of a 3D SM trajectory capable of producing complex rhythmic or granular amplitude envelopes – as the source moves quickly into the distance its trail fades out corresponding with gain attenuation (Section 4.3). Appendix B contains more examples of granular and percussive sounds possible with SM.

5.2.3 Spatial Morphing

Expanding SM bounds simultaneously lowers modulation frequency while increasing depth of amplitude modulation, which decomposes a tone into a series of spatial glissons [59]. Synchronously compressing SM bounds and slowing velocity will return a SM tone back to its original input source or single carrier frequency (when the input is a simple oscillator). Increasing SM velocity within fixed bounds allows continuos control of spatialization up to the speed of sound, increasing modulation frequency and depth simultaneously while spreading the amplitude over any number of loudspeakers according to a spatialization algorithm (Section 4.4).

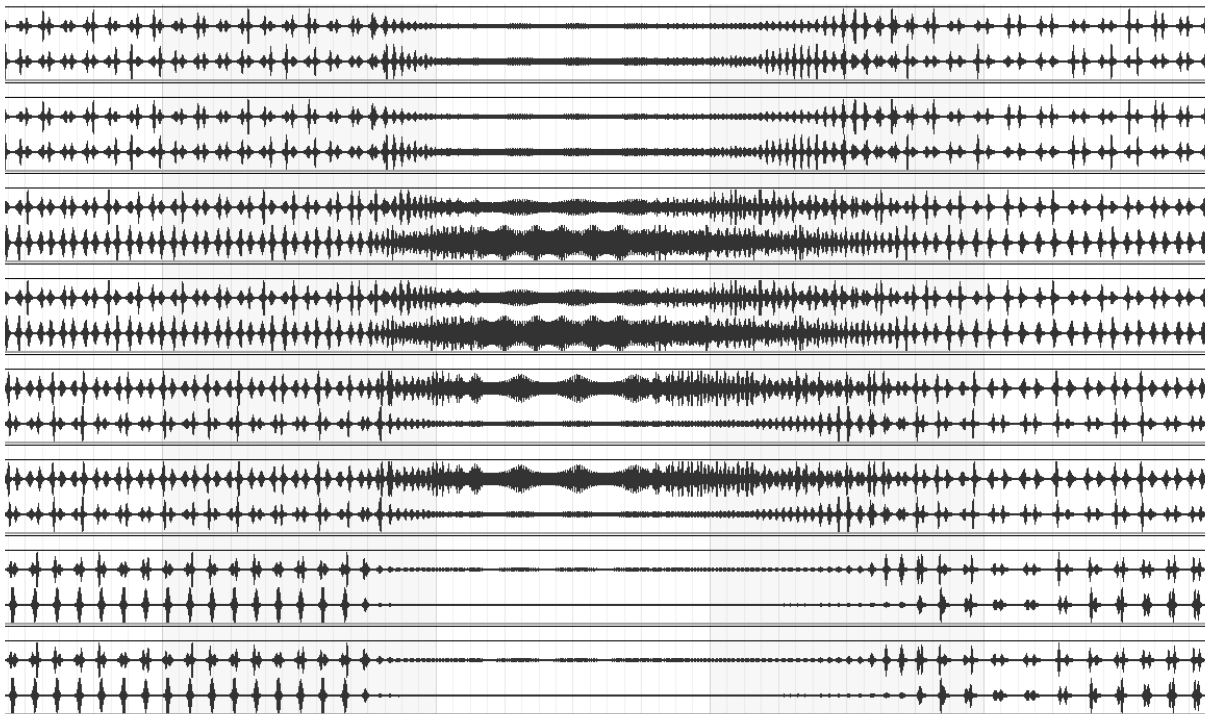


Figure 5.13: Compression and Expansion of SM Bounds Over 16 Channels

Figure 5.13 shows the result of SM bounds compression and expansion over 16 channels while maintaining the same velocity of source motion during a 30 second time interval. When the bounds are compressed towards the center of the figure, the multichannel sound fuses into a tone that spatially morphs back to a serious of granular amplitude envelopes as the bounds are expanded once again. This technique is useful to simulate explosion effects and to gain a unified control between tone, rhythm, and granulation using only single slider or MIDI knob with the SM plug-in (Section 5.3). Figure 5.14 shows the result of asymmetrical SM bounds in which lower bounds is increased as the upper bounds decreases, inverting the bounds of spatial modulation. For a brief moment at the midpoint the bounds are equal, creating a spatially centered tone, which breaks into granular stream biased towards the opposite side of the speaker array as one of the bounds continues to increase. Composers and performers of computer music can use this technique to easily focus, shift, or invert the spatial image of sounds across a speaker array of arbitrary size using only a few sliders[1]. Appendix A.4 summarizes the ranges of SM velocity and bounds useful to morph between spatial effects and synthesis.

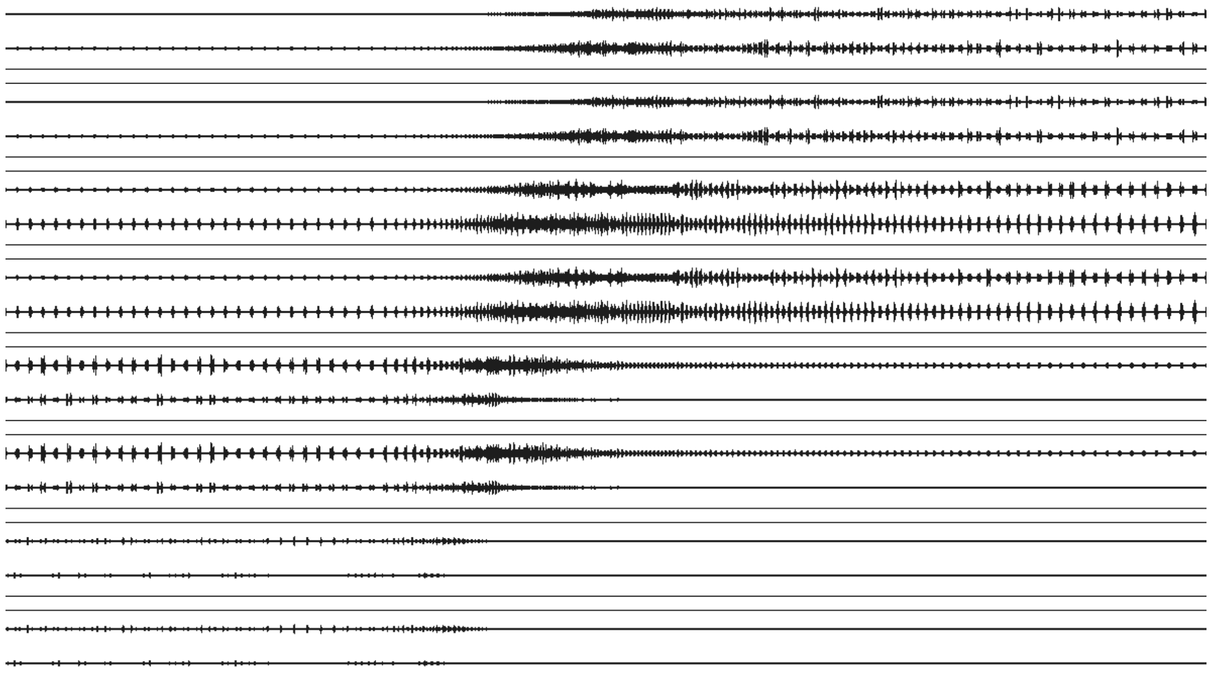


Figure 5.14: Inversion of SM Bounds Over 16 Channels

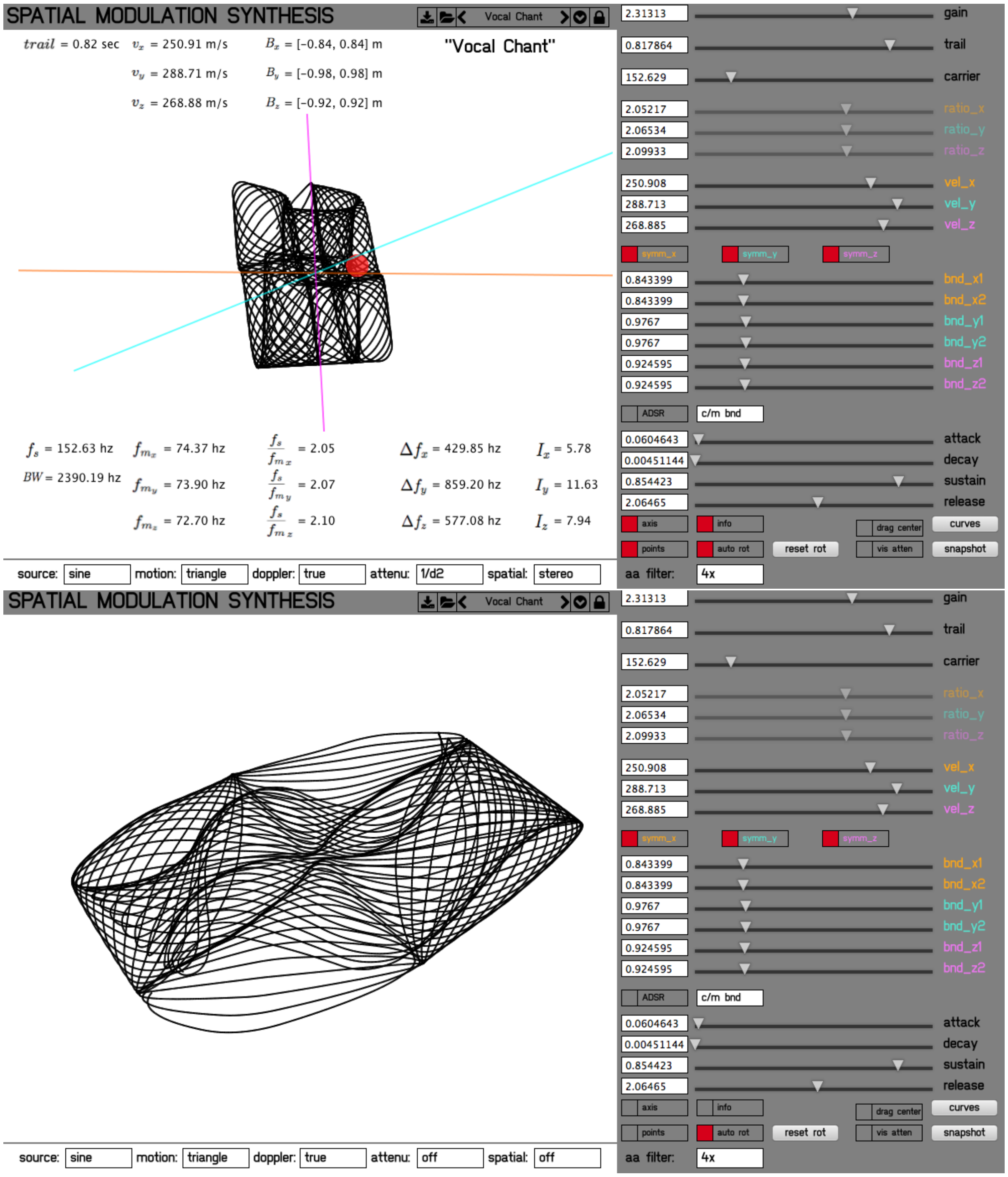
[1]2 to 6 sliders required: min and max bounds for each dimension desired

5.3 Implementation

Spatial modulation synthesis is currently implemented as a VST[2] plug-in using the open-source AlloSystem C++ API[3] developed by the AlloSphere Research Group at the University of California, Santa Barbara. Though AlloSystem included abstractions for spatial audio scenes and audio-visual agents within the scene, several modifications were made to the API to realize SM, most notably the ability for sample rate trajectory control to realize high-speed, physically accurate Doppler shift. Implementing the software as a plug-in allows for timeline automation of all SM parameters, full MIDI control, simple preset saving, and audio device management by the host DAW. The use of AlloSystem for graphics rendering enables compatibility with immersive, distributed 3D displays such as the AlloSphere Research Laboratory[4] . This chapter will describe how the SM Plug-in can act as a spatialization interface, modulation effect, and audio-visual synthesis instrument. A demo of the plug-in is available for download at `www.spatialmodulation.com`.

The SM Plug-in interface (Figure 5.15) exposes all of the control parameters for SM and provides additional features which aid use as a synthesizer instrument such as an ADSR envelope and the ability to control the source (or carrier) to modulation ratios in each dimension (Section 5.2.1). The heart of the interface is the large sound trajectory display which traces the position history (trajectory) of the sound over a given time (*trail*). There are toggles to control the display of FM parameters resulting from the chosen Doppler simulation, the source position (red), and x/y/z axes (orange, blue, and violet, respectively). A full list of parameters along with their description and ranges is given in Table 5.3.

[2]`https://en.wikipedia.org/wiki/Virtual_Studio_Technology`
[3]`https://github.com/AlloSphere-Research-Group/AlloSystem`
[4]`http://www.allosphere.ucsb.edu`



"Vocal Chant"

$trail$ = 0.82 sec v_x = 250.91 m/s B_x = [−0.84, 0.84] m

v_y = 288.71 m/s B_y = [−0.98, 0.98] m

v_z = 268.88 m/s B_z = [−0.92, 0.92] m

f_s = 152.63 hz f_{m_x} = 74.37 hz $\frac{f_s}{f_{m\,x}}$ = 2.05 Δf_x = 429.85 hz I_x = 5.78

BW = 2390.19 hz f_{m_y} = 73.90 hz $\frac{f_s}{f_{m\,y}}$ = 2.07 Δf_y = 859.20 hz I_y = 11.63

f_{m_z} = 72.70 hz $\frac{f_s}{f_{m\,z}}$ = 2.10 Δf_z = 577.08 hz I_z = 7.94

source: sine motion: triangle doppler: true attenu: 1/d2 spatial: stereo

2.31313	gain
0.817864	trail
152.629	carrier
2.05217	ratio_x
2.06534	ratio_y
2.09933	ratio_z
250.908	vel_x
288.713	vel_y
268.885	vel_z
0.843399	bnd_x1
0.843399	bnd_x2
0.9767	bnd_y1
0.9767	bnd_y2
0.924595	bnd_z1
0.924595	bnd_z2
0.0604643	attack
0.00451144	decay
0.854423	sustain
2.06465	release

aa filter: 4x

source: sine motion: triangle doppler: true attenu: off spatial: off

2.31313	gain
0.817864	trail
152.629	carrier
2.05217	ratio_x
2.06534	ratio_y
2.09933	ratio_z
250.908	vel_x
288.713	vel_y
268.885	vel_z
0.843399	bnd_x1
0.843399	bnd_x2
0.9767	bnd_y1
0.9767	bnd_y2
0.924595	bnd_z1
0.924595	bnd_z2
0.0604643	attack
0.00451144	decay
0.854423	sustain
2.06465	release

aa filter: 4x

Figure 5.15: Spatial Modulation VST Plug-in Interface (Top Showing Formula and Axis Display

Parameter	**Description**	**Range/Options**
source	sound source: oscillator, arbitrary audio input from DAW, or scanned synthesis (audification) of the sound trajectory (feedback modulation)	sine, saw, square, triangle, noise, audio, scan
motion	motion type for sound source (Section 4.2.4)	square, sine, triangle, saw, inverse saw
doppler	Doppler simulation type (Section 4.2.6)	symmetrical, true, true (=), off
attenuation	distance-based gain attenuation curve (Section 4.3)	1/d, 1/d2, off
spatialization	spatialization algorithm (Section 4.4)	stereo, dbap, vbap, ambisonics, off
trail	length of trajectory history to draw	[0, 1] seconds
carrier	source frequency, f_s, not applicable when *ADSR* is On	[0, 10000] Hz
ratio x/y/z	carrier to modulator (c/m) ratio for x/y/z modulators, automatically set based on computation from *vel* and *bnd* when *c/m lock* is off, when *c/m lock* is on these sliders set *vel* and/or *bnd* to achieve the desired c/m ratios	[0, 100]
vel x/y/z	velocity for x/y/z motion	[0, 340] meters/sec
bnd x1/y1/z1	lower bounds for x/y/z motion (left/-down/back respectively)	[0.1, 100] meters
bnd x2/y2/z2	upper bounds for x/y/z motion (right/up/front respectively)	[0.1, 100] meters
c/m lock	maintain c/m ratios (*ratio x/y/z*) dynamically by setting *bnd* and/or *vel* according to carrier frequency (*carrier*) or note frequency when *ADSR* is On (Section 5.2.1)	"off", "bnd", "vel"
ADSR	attack/decay/sustain/release amplitude envelope, if On then *carrier* is set via MIDI note input	On/Off, [0, 4] seconds

Table 5.4: Spatial Modulation Plug-in Parameters

Spatializer and Synthesizer

The SM plug-in is unique in the sense that it has the ability to operate as spatializer, effect, or synthesizer and traverses this array of uses by means of multi-dimensional velocity and bounds control (Appendix A.4). Depending on velocity and bounds, the visual trajectories may be seen as perceivable spatialization paths or as complex shapes indicative of the resulting synthesized timbres. Trajectories and sounds can be composed in stereo and have the ability to scale to multi-channel venues of arbitrary size. Two works created with the SM plug-in demonstrate its flexible usage.

Kinetic V2 **and** ***Concourse*** **(2015)**

Kinetic V2 used the SM plug-in for live, improvisational synthesis of SM timbres over 54.1 channel sound system. The as with *Kinetic* (Section 3.8) the audience was immersed inside the 3D high-speed trajectory orbits of the sounds. However, the use of a simple MIDI keyboard with knobs allowed melodies to be played that changed the shape and size of the orbits in real-time.

Concourse was a collaborative audio-visual installation with Reza Ali[5] that ran continuously from September 28th until December 1st, 2015 in the lobby of Dolby's downtown San Francisco headquarters. Unlike *Kinetic V2*, the SM plug-in was used to spatialize existing complex sounds over a 2D, overhead 30.30 speaker array. Like SES 3.2 the SM plug-in is capable of receiving OSC to act as a spatialization server. In this case precise spatial coordinates were sent from the visual application (Figure 5.17 Bottom) to control the position of sounds representing abstract organic audio-visual elements traversing a digital landscape over a 100 foot wide led display.

[5]`http://www.syedrezaali.com`

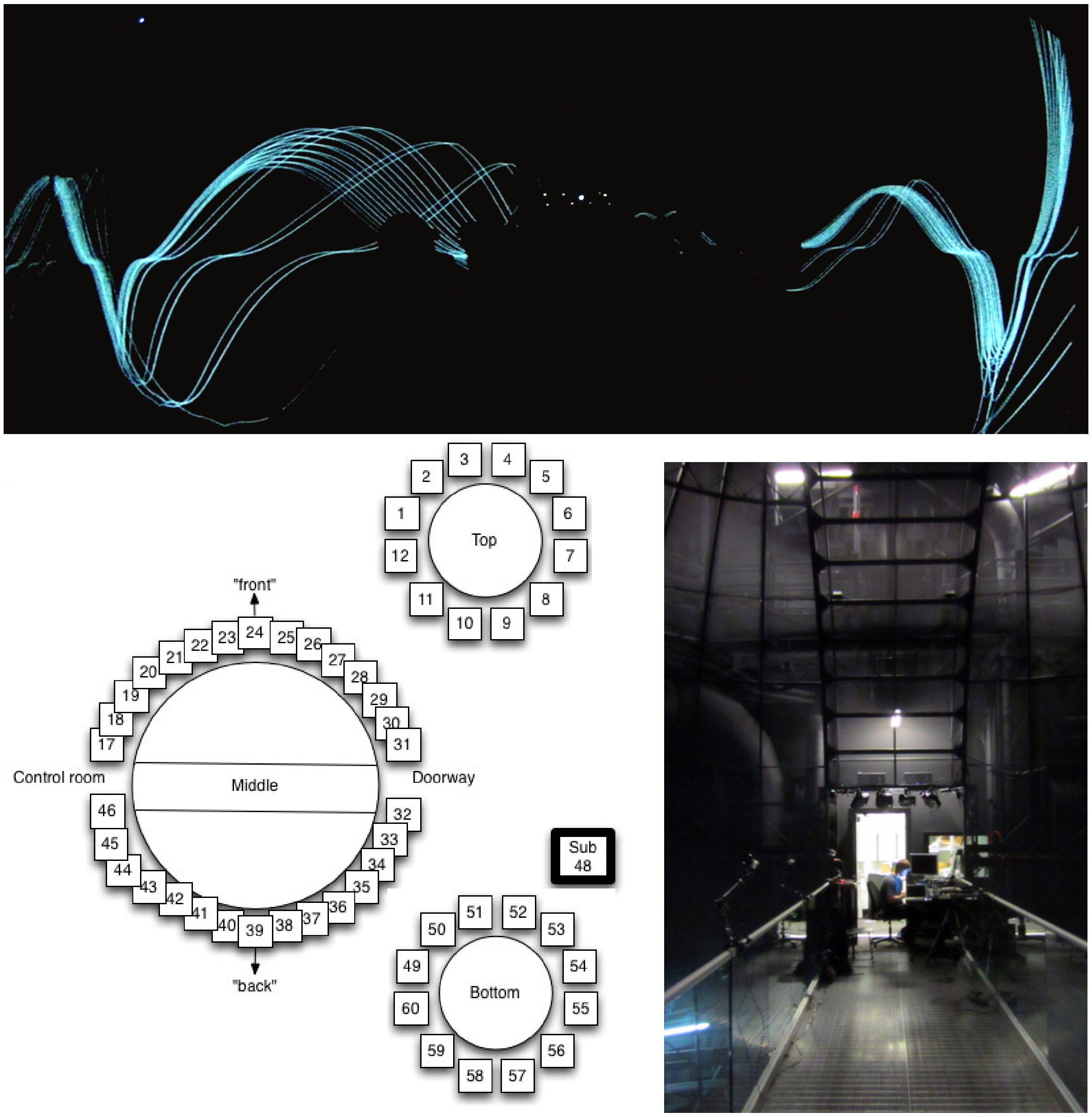
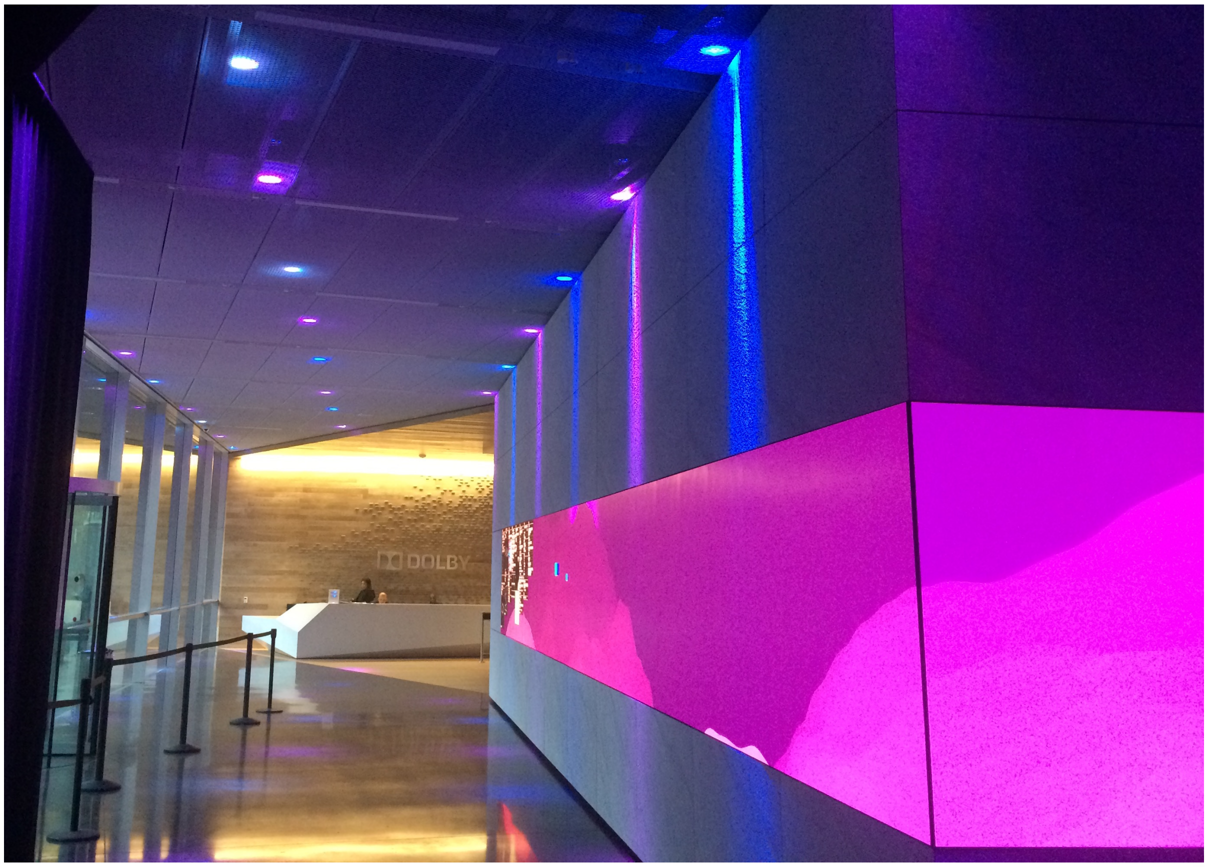


Figure 5.16: *Kinetic V2* (Top), the 54.1 Channel AlloSphere System (Bottom Left), and AlloSphere Audience Bridge (Bottom Right)



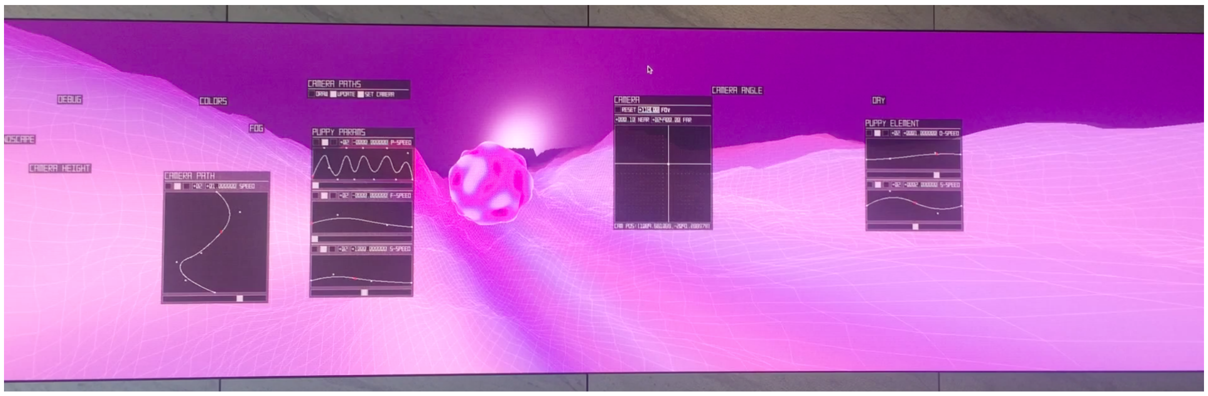


Figure 5.17: *Concourse* Installation at Dolby Headquarters in San Francisco (Top), *Concourse* Spatial Element Trajectory Interface (Bottom)

5.4 Unified Audio-visual Rendering

Most multimedia software frameworks have separate threads for sound and graphics rendering. The origins of this separation derive from the fact that modern computers have separate hardware for sound and graphics output. Between the hardware layer and programming language the operating system manages separate threads for writing audio samples and pixels to the sound card and display respectively. Though these are separate threads, it is not uncommon for there to be communication between audio and graphics threads to synchronize multimedia. For example, audio events may occur as a result of a graphical action in a computer game or vice versa. What is uncommon and unconsidered in existing software frameworks are situations in which audio and graphic output are not only synchronized, but are the same data.

Popular creative coding frameworks such as openFrameworks[6] and Cinder[7] are used heavily by VJs and new media artists for creating synchronized audio-visual art. Cases in which visuals are synchronized to sound are common amongst VJs and this process if often termed as creatng "audio-reactive" art. Because of audio's much higher sampling rate, obtaining instantaneous sonic information involves a form of subsampling the audio samples (averaging, for instance) to obtain information to be displayed at the graphics' frame rate. Or, an entire buffer of audio data can be stored between graphics frames for display during the next frame.

Situations involving sound synchronized to creative visuals are less common, but graphical control of audio is common in the sense that any situation involving a graphical user interface (GUI) to control sound (any software instrument) involves a conversion of graphics rate data to audio control data. In these situations there is graphical data at a control rate which must be interpolated if it is to affect the audio continuously. For

[6]`http://www.openframeworks.cc`
[7]`http://www.openframeworks.cc`

instance, when a slider is moved on the GUI of a software synthesizer instrument to control amplitude, the control values are sent at a rate usually corresponding to the graphics display rate (around 60 Hz) and are interpolated for continuous control at audio sample rate (44100 Hz, for example).

Some creative coding libraries have the concept of an audio-visual "Agent" [60], an entity that produces synchronized graphics and sound such as a character in a computer game. The Agent is aware of its location in space and the audio and graphics that need to be rendered for a given location in space or for a given event (for example, when the character moves). In terms of programming language implementation the Agent is commonly a C++ or Java object with methods for audio and graphics rendering and a number of member variables shared between the two rendering callback methods (Figure 5.18). While this provides an organized container for audio-visual data and rendering, it does not account for a situation in which the audio and graphic data need to be identical without interpolation or subsampling.

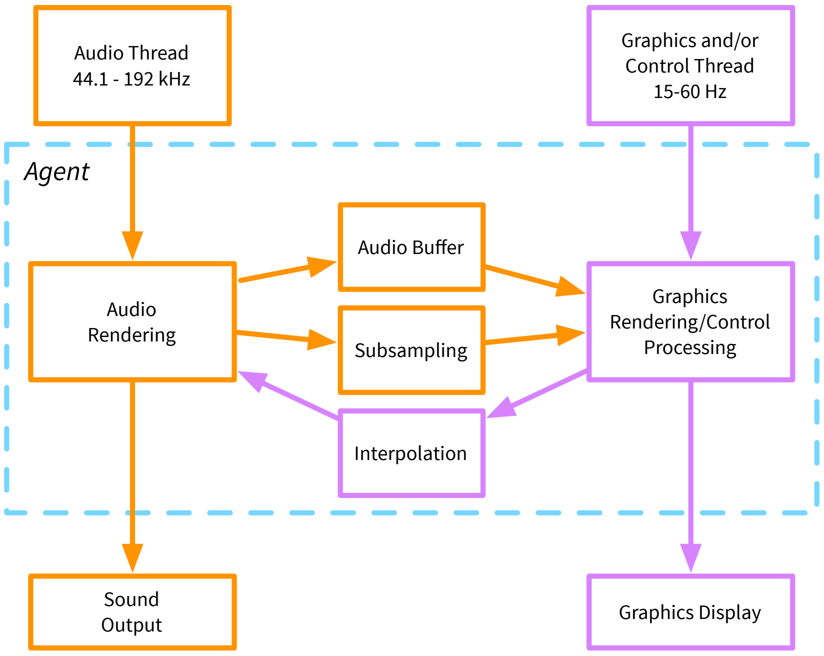


Figure 5.18: A Common, Separated Audio-Visual Rendering Paradigm

The implementation of the spatial modulation and image sonification sound synthesis paradigms described in this dissertation necessitated a new rendering paradigm in which there is a single unified audio-visual rendering method. Because of the literal spatial connection between the graphics and sounds of SM and the need for the simulation accurate high-speed trajectories, control of sound spatialization at graphics rate with interpolation would not suffice. To overcome this limitation both audio and graphics are rendered in a single method called by the audio thread and a graphics buffer is used to store several frames of the trajectory until the next display call by the graphics thread. This allows for the sharing of audio sample-rate control data between sound and graphics rendering (Figure 5.19).

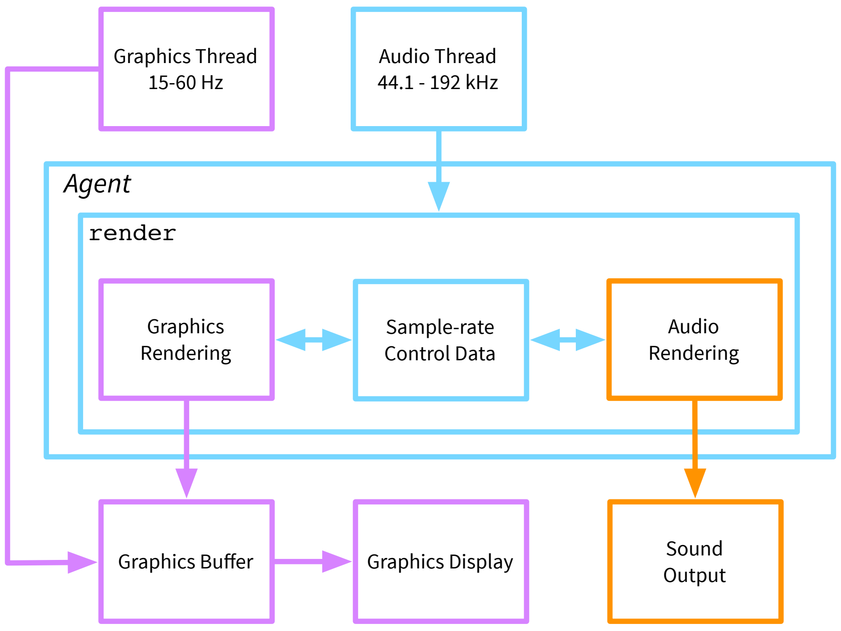


Figure 5.19: Unified Audio-Visual Rendering Paradigm

The unified rendering paradigm requires the use of an Agent with a single render method that updates motion, spatialization, and any audio-rate control shared between between sound and graphics. For instance, within the same render method a vertex can be computed and a sample of audio rendered from the same floating point number without interpolation or subsampling. In addition to sample rate trajectory control for SM, this unified rendering paradigm was needed for image sonification experiments in which images had to be filtered and rendered on the audio thread since their pixel data was needed to render audio samples.

An ideal future paradigm would involve not only unified rendering, but a unified sample rate between sound, graphics, and other possible sensory data (haptics, for example). While human vision may limit the perception of visual spatial motion faster than graphics frame rate (60-120 Hz), SM shows that the rates at which we can perceive spatial sound motion are much higher as perceivable motions transform into perceivable tones when controlled at audio-rate (44100 Hz). As described in Section 5.1.1, even higher sampling rates are necessary to simulate sound source motion below 0.1 meters. Increasing sampling rates to the megahertz range would allow for micrometer bounds of motion for SM.

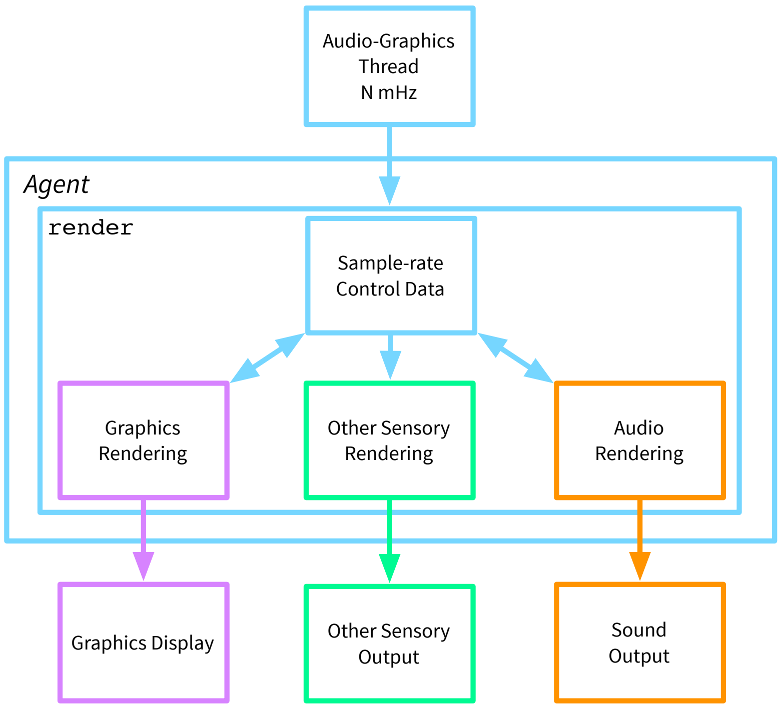


Figure 5.20: Future Hypothetical Unified Audio-Visual Rendering Paradigm

Chapter 6

Future Research

6.0.1 Additional Distance Cues

The current model for spatial modulation synthesis uses only two distance cues, Doppler shift and distance-based gain attenuation. However, many other techniques exist including presence filtering and global versus local reverberation. It may also be interesting to consider a model for representing a sonic boom at the speed of sound.

6.0.2 Additional Sound Orbits

In addition to the current selections of motion types and orbital trajectories used for spatial modulation, several other forms of complex 3 dimensional motion should be explored. These include knots[1], varieties of orbital mechanics, and geometrical algebra formations.

[1] `https://en.wikipedia.org/wiki/Knot_theory`

6.0.3 Additional SM Control Features

Theres are several possibilities for novel spatial control features to add to the SM plug-in (Section 5.3).

- Per x/y/z Motion Type
- Modulation Index ADSR applied to modulate bounds or velocity as notes at a fixed source to modulation ratio are played
- Multi-touch control of the SM plug-in to literally stretch and compress bounds with physical gestures
- Adaptive FM implementation to adjust spatial modulation frequency in correlation with complex input to maintain a desired source to modulator ratio
- Ability to modulate listener position rather than source position

Chapter 7

Conclusions

- Original work exploring the acceleration of spatial trajectories and spatial data (Chapter 3) led to 4 publications in conference proceedings, 3 original multi-channel compositions with public performances, 3 spatial audio software contributions to public art installations, and 2 novel mobile synthesis applications for sonification of spatial data. Throughout this work a notion of high-speed spatialization to create and shape timbres led to the theory of Spatial Modulation Synthesis.

- Spatial Modulation Synthesis (SM) (Chapters 4 and 5) has been introduced as a novel control paradigm unifying space with intensity, duration, pitch, timbre, and visual form by simulating high-speed periodic motion of an arbitrary sound source using 2 primary controls: velocity and bounds. The audio-visual relationships resulting from SM are based on the physical first principles of Doppler shift and distance-based gain-attenuation rather than extraneous mappings or extrinsic synchronization. The implementation of SM exists as a tool traversing uses for spatial mixing, effects, and synthesis – proving simple, synchronized controls replacing an otherwise complex chain of effects and mappings to achieve spatial AM, FM, and granular synthesis. The current Doppler approximation formula ubiquitous in digital audio was not sufficient for covering the entire timbral space made possible by SM. An implementation of physically accurate Doppler shift along with a steep anti-aliasing filter was used to achieve high-index FM timbres from simulated high-velocity source motion.

Just as Chowning took vibrato to new rates to create FM synthesis, spatial modulation synthesis takes sound spatialization to new speeds to create a complex form of modulation and granular synthesis. Using parameters of space and motion to morph sounds into new tones and decompose them into grains expands Stockhausen's Concept of Unity to include spatialization alongside timbre, pitch, intensity, and duration. Visualization of the sound trajectories producing spatial modulation synthesis provides further unity between visual space and sound timbre.

- Implementation of SM required development of an audio-rate graphics rendering paradigm to accurately control high-speed sound trajectories (Section 5.4). This rendering approach was in contrast to mainstream techniques that render spatial audio trajectories at graphics frame rate. It was shown that in order to realize the full potential of SM (minimum bonds, maximum modulation frequency) the highest possible sampling rate for the unified rendering of audio and graphics is needed.

Appendix A

Doppler FM Formulas

A.1 Variables

The formulas and variables in this appendix consider one-dimensional Spatial Modulation and symmetrical bounds. For asymmetric spatial motion (Section 4.2.5), B may be replaced with $B1 + B2$ where $B1$ is the minimum bounds and $B2$ is the maximum bounds. Multi-dimensional spatial motion implies v_s is a vector (Section 4.2.7), and, in general, the primary frequency of modulation, f_m, will be the lowest frequency of modulation for any dimension while the depth of modulation will be the highest depth of any dimension.

Table A.1 defines the variables used in the Doppler FM formulas.

Variable	**Definition**	**Range**
v_s	source velocity, positive when moving towards listener (v_t), negative when moving away from listener (v_a)	0 to 340 m/s c
c	speed of sound in air	340 m/s
f_s	source frequency or carrier frequency	0 to 20 kHz
B	bounds of symmetric spatial motion	0.1m (depending on sample rate) to 100m

Table A.1: Doppler FM Variables

A.2 Modulation Depth,

	Doppler Approximation	True Doppler	True Doppler Equal Depth
Δf	$\frac{v_s}{c} f_s$	$\frac{f_s v_s}{c - v_s}$	$\frac{f_s v_s}{c + v_a}$

Table A.2: Modulation Depth for All SM Motion Types

A.3 Modulation Frequency and Index

A.3.1 Square Motion (Constant Velocity, No Acceleration)

	Doppler Approximation	True Doppler	True Doppler Equal Depth
f_m	$\frac{v_s}{2B}$	$\frac{v_s}{2B}$	$\frac{v_a^2 + v_a c}{2B(2v_a + c)}$
I	$\frac{2Bf_s}{c}$	$\frac{2Bf_s}{c - v_s}$	$\frac{2B(2v_a + c) f_s v_s}{(v_a^2 + v_a c)(c + v_a)}$
$f_s : f_m$	$\frac{2Bf_s}{v_s}$	$\frac{2Bf_s}{v_s}$	$\frac{2B(2v_a + c) f_s}{v_a^2 + v_a c}$

A.3.2 Sinusoidal Motion (Sinusoidal Acceleration)

	Doppler Approximation	True Doppler	True Doppler Equal Depth
f_m	$\frac{v_s}{\pi B}$	$\frac{v_s}{\pi B}$	$\frac{v_a^2+v_a c}{\pi B(2v_a+c)}$
I	$\frac{\pi B f_s}{c}$	$\frac{\pi B f_s}{c-v_s}$	$\frac{\pi B(2v_a+c) f_s v_s}{(v_a^2+v_a c)(c+v_a)}$
$f_s : f_m$	$\frac{\pi B f_s}{v_s}$	$\frac{\pi B f_s}{v_s}$	$\frac{\pi B(2v_a+c) f_s}{v_a^2+v_a c}$

A.3.3 Triangle / Saw / Inverse Saw Motion (Linear Acceleration)

	Doppler Approximation	True Doppler	True Doppler Equal Depth
f_m	$\frac{v_s}{4B}$	$\frac{v_s}{4B}$	$\frac{v_a^2+v_a c}{4B(2v_a+c)}$
I	$\frac{4B f_s}{c}$	$\frac{4B f_s}{c-v_s}$	$\frac{4B(2v_a+c) f_s v_s}{(v_a^2+v_a c)(c+v_a)}$
$f_s : f_m$	$\frac{4B f_s}{v_s}$	$\frac{4B f_s}{v_s}$	$\frac{4B(2v_a+c) f_s}{v_a^2+v_a c}$

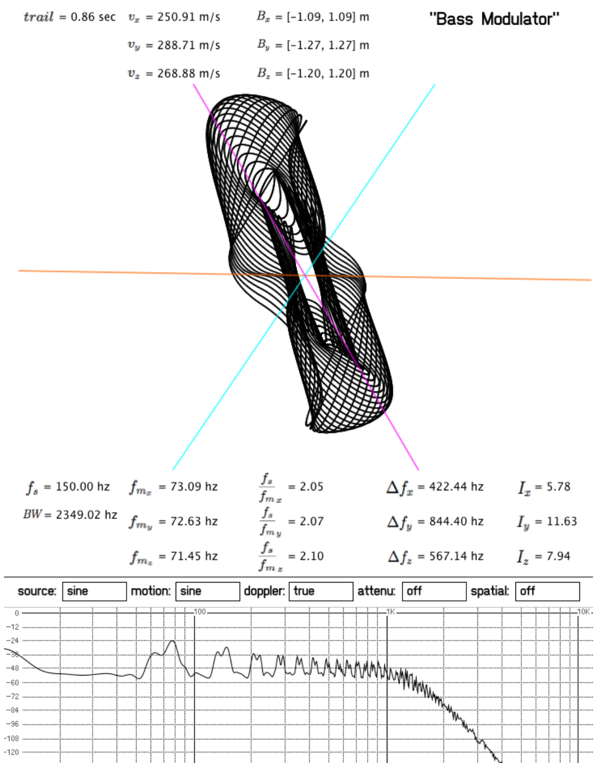
A.4 SM Velocity and Bounds Ranges

	Personal Bounds $B \leq 1m$	**Room Bounds** $1m < B \leq 10m$	**Field Bounds** $10m < B \leq 100m$
Human Velocity $\lvert v_s \rvert \leq 10m/s$	Auto Pan	Auto Pan	Pass-by Effects
Car Velocity $10m/s < \lvert v_s \rvert \leq 50m/s$	Vibrato	Auto Pan	Pass-by Effects
Plane Velocity $50m/s < \lvert v_s \rvert \leq 200m/s$	Rough Tone	Rhythm - Rough Tone	Modulation Panning
Jet Velocity $200m/s < \lvert v_s \rvert \leq 340m/s$	Tone	Rhythm - Rough Tone	Granular

Appendix B

Spatial Modulation Cookbook

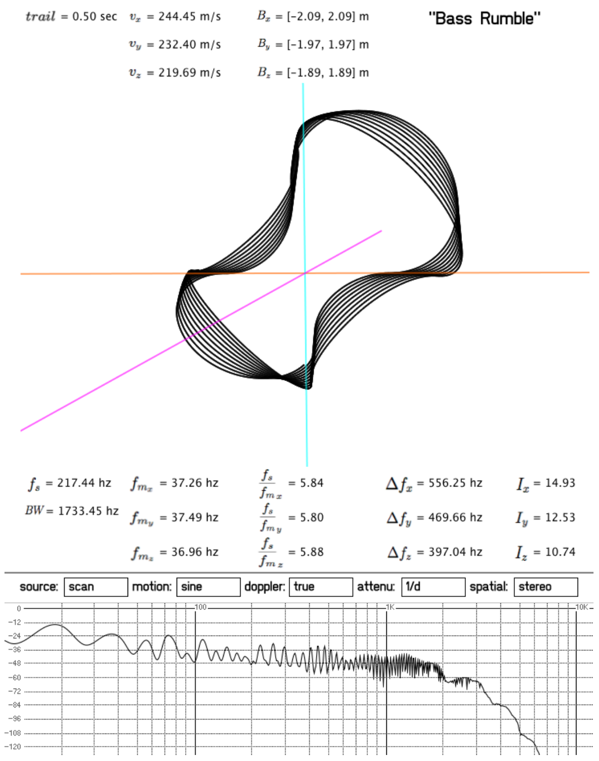
B.1 Bass


"Bass Modulator"

trail = 0.86 sec v_x = 250.91 m/s B_x = [−1.09, 1.09] m
v_y = 288.71 m/s B_y = [−1.27, 1.27] m
v_z = 268.88 m/s B_z = [−1.20, 1.20] m

f_s = 150.00 hz f_{m_x} = 73.09 hz $\frac{f_s}{f_{m\,x}}$ = 2.05 Δf_x = 422.44 hz I_x = 5.78
BW = 2349.02 hz f_{m_y} = 72.63 hz $\frac{f_s}{f_{m\,y}}$ = 2.07 Δf_y = 844.40 hz I_y = 11.63
f_{m_z} = 71.45 hz $\frac{f_s}{f_{m\,z}}$ = 2.10 Δf_z = 567.14 hz I_z = 7.94

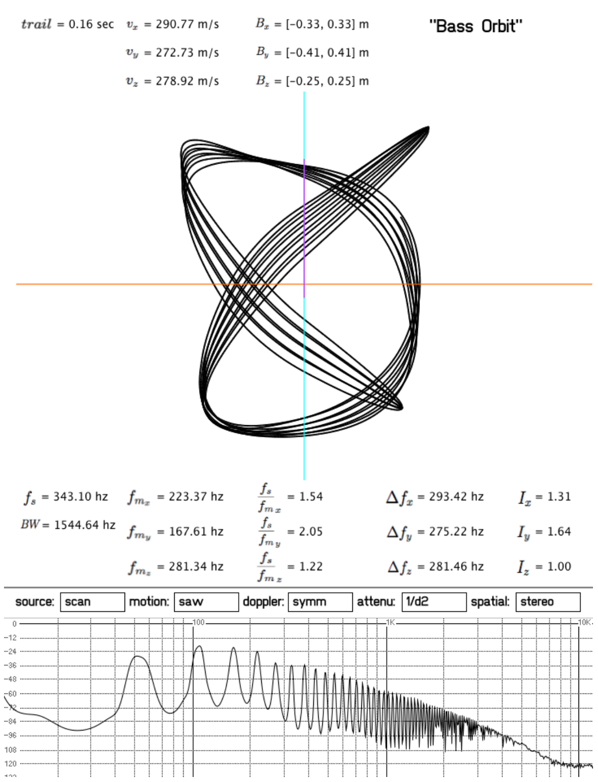
source: sine motion: sine doppler: true attenu: off spatial: off


"Bass Rumble"

trail = 0.50 sec v_x = 244.45 m/s B_x = [−2.09, 2.09] m
v_y = 232.40 m/s B_y = [−1.97, 1.97] m
v_z = 219.69 m/s B_z = [−1.89, 1.89] m

f_s = 217.44 hz f_{m_x} = 37.26 hz $\frac{f_s}{f_{m\,x}}$ = 5.84 Δf_x = 556.25 hz I_x = 14.93
BW = 1733.45 hz f_{m_y} = 37.49 hz $\frac{f_s}{f_{m\,y}}$ = 5.80 Δf_y = 469.66 hz I_y = 12.53
f_{m_z} = 36.96 hz $\frac{f_s}{f_{m\,z}}$ = 5.88 Δf_z = 397.04 hz I_z = 10.74

source: scan motion: sine doppler: true attenu: 1/d spatial: stereo

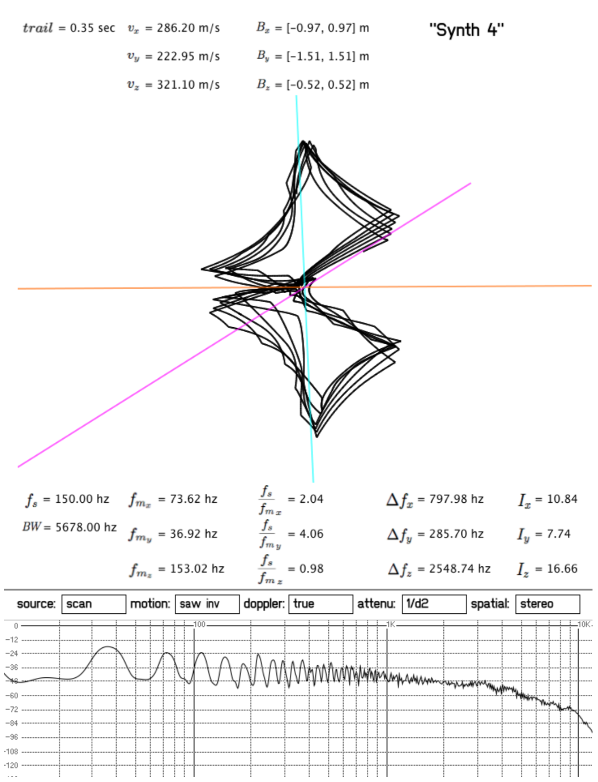


"Bass Orbit"

$trail$ = 0.16 sec v_x = 290.77 m/s B_x = [−0.33, 0.33] m
v_y = 272.73 m/s B_y = [−0.41, 0.41] m
v_z = 278.92 m/s B_z = [−0.25, 0.25] m

f_s = 343.10 hz f_{m_x} = 223.37 hz $\frac{f_s}{f_{m\,x}}$ = 1.54 Δf_x = 293.42 hz I_x = 1.31
BW = 1544.64 hz f_{m_y} = 167.61 hz $\frac{f_s}{f_{m\,y}}$ = 2.05 Δf_y = 275.22 hz I_y = 1.64
f_{m_z} = 281.34 hz $\frac{f_s}{f_{m\,z}}$ = 1.22 Δf_z = 281.46 hz I_z = 1.00

source: scan motion: saw doppler: symm attenu: 1/d2 spatial: stereo

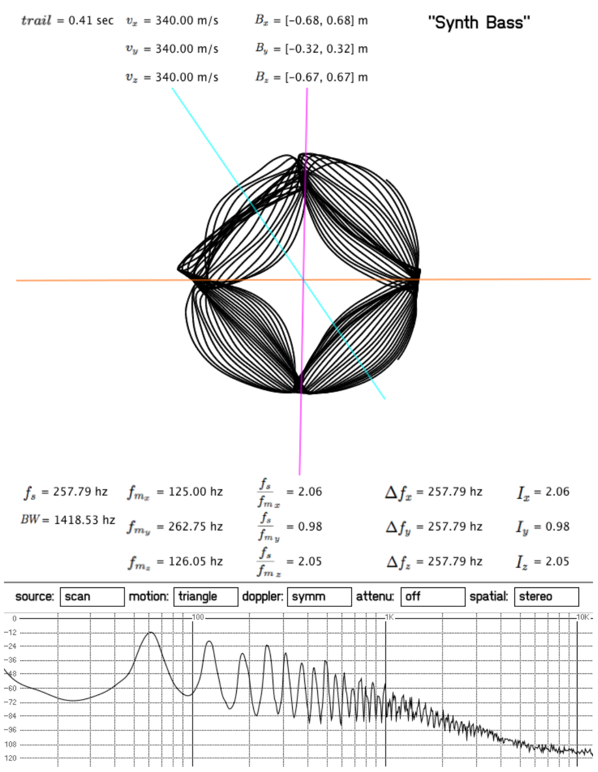


"Synth 4"

$trail$ = 0.35 sec v_x = 286.20 m/s B_x = [−0.97, 0.97] m
v_y = 222.95 m/s B_y = [−1.51, 1.51] m
v_z = 321.10 m/s B_z = [−0.52, 0.52] m

f_s = 150.00 hz f_{m_x} = 73.62 hz $\frac{f_s}{f_{m\,x}}$ = 2.04 Δf_x = 797.98 hz I_x = 10.84
BW = 5678.00 hz f_{m_y} = 36.92 hz $\frac{f_s}{f_{m\,y}}$ = 4.06 Δf_y = 285.70 hz I_y = 7.74
f_{m_z} = 153.02 hz $\frac{f_s}{f_{m\,z}}$ = 0.98 Δf_z = 2548.74 hz I_z = 16.66

source: scan motion: saw inv doppler: true attenu: 1/d2 spatial: stereo

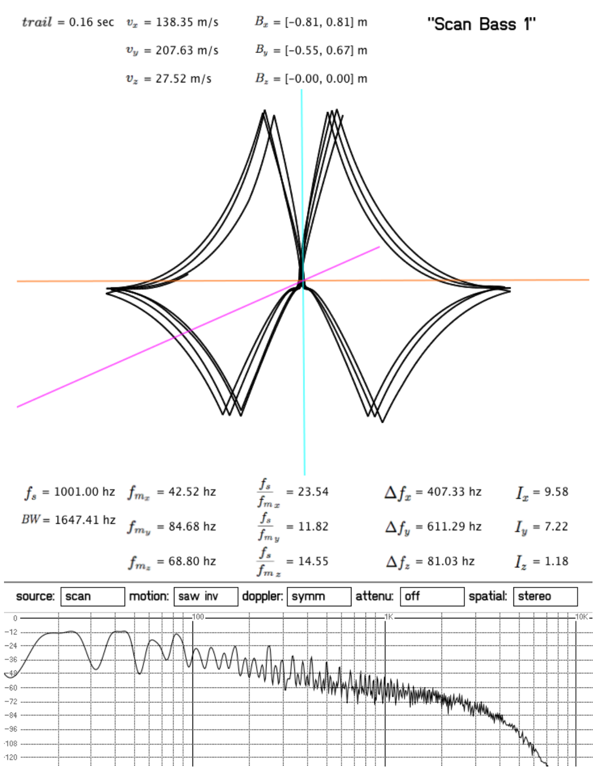


"Synth Bass"

$trail$ = 0.41 sec v_x = 340.00 m/s B_x = [−0.68, 0.68] m
v_y = 340.00 m/s B_y = [−0.32, 0.32] m
v_z = 340.00 m/s B_z = [−0.67, 0.67] m

f_s = 257.79 hz f_{m_x} = 125.00 hz $\frac{f_s}{f_{m\,x}}$ = 2.06 Δf_x = 257.79 hz I_x = 2.06
BW = 1418.53 hz f_{m_y} = 262.75 hz $\frac{f_s}{f_{m\,y}}$ = 0.98 Δf_y = 257.79 hz I_y = 0.98
f_{m_z} = 126.05 hz $\frac{f_s}{f_{m\,z}}$ = 2.05 Δf_z = 257.79 hz I_z = 2.05

source: scan motion: triangle doppler: symm attenu: off spatial: stereo



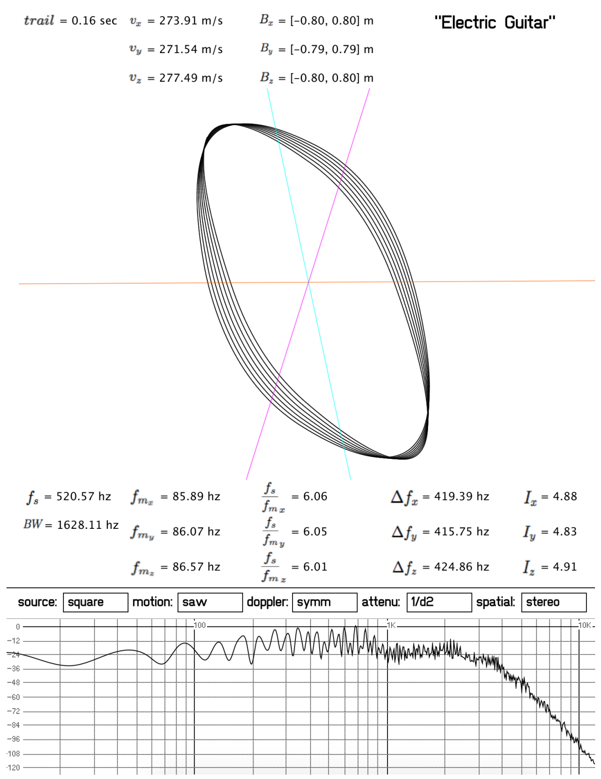
"Scan Bass 1"

$trail$ = 0.16 sec v_x = 138.35 m/s B_x = [−0.81, 0.81] m
v_y = 207.63 m/s B_y = [−0.55, 0.67] m
v_z = 27.52 m/s B_z = [−0.00, 0.00] m

f_s = 1001.00 hz f_{m_x} = 42.52 hz $\frac{f_s}{f_{m\,x}}$ = 23.54 Δf_x = 407.33 hz I_x = 9.58
BW = 1647.41 hz f_{m_y} = 84.68 hz $\frac{f_s}{f_{m\,y}}$ = 11.82 Δf_y = 611.29 hz I_y = 7.22
f_{m_z} = 68.80 hz $\frac{f_s}{f_{m\,z}}$ = 14.55 Δf_z = 81.03 hz I_z = 1.18

source: scan motion: saw inv doppler: symm attenu: off spatial: stereo

B.2 Tonal



$trail$ = 0.16 sec v_x = 273.91 m/s B_x = [−0.80, 0.80] m "Electric Guitar"

v_y = 271.54 m/s B_y = [−0.79, 0.79] m

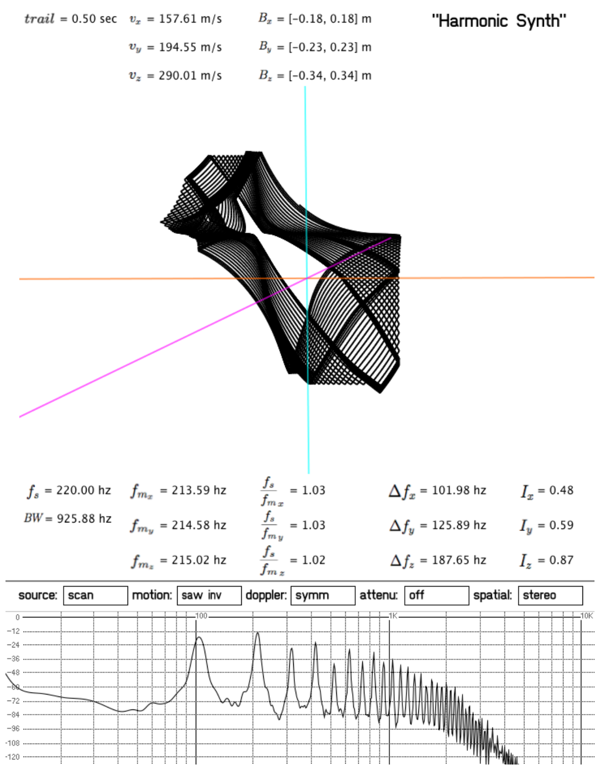
v_z = 277.49 m/s B_z = [−0.80, 0.80] m

f_s = 520.57 hz f_{m_x} = 85.89 hz $\frac{f_s}{f_{m\,x}}$ = 6.06 Δf_x = 419.39 hz I_x = 4.88

BW = 1628.11 hz f_{m_y} = 86.07 hz $\frac{f_s}{f_{m\,y}}$ = 6.05 Δf_y = 415.75 hz I_y = 4.83

f_{m_z} = 86.57 hz $\frac{f_s}{f_{m\,z}}$ = 6.01 Δf_z = 424.86 hz I_z = 4.91

source: square motion: saw doppler: symm attenu: 1/d2 spatial: stereo



$trail$ = 0.50 sec v_x = 157.61 m/s B_x = [−0.18, 0.18] m "Harmonic Synth"

v_y = 194.55 m/s B_y = [−0.23, 0.23] m

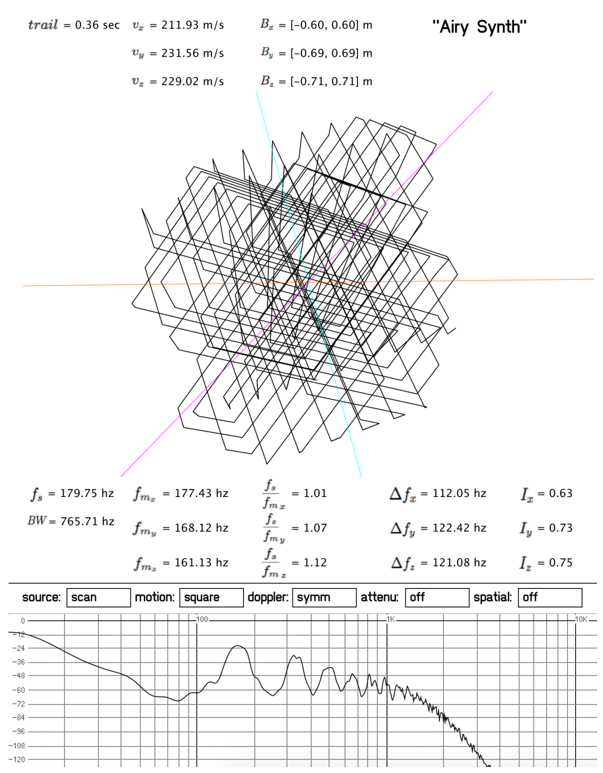
v_z = 290.01 m/s B_z = [−0.34, 0.34] m

f_s = 220.00 hz f_{m_x} = 213.59 hz $\frac{f_s}{f_{m\,x}}$ = 1.03 Δf_x = 101.98 hz I_x = 0.48

BW = 925.88 hz f_{m_y} = 214.58 hz $\frac{f_s}{f_{m\,y}}$ = 1.03 Δf_y = 125.89 hz I_y = 0.59

f_{m_z} = 215.02 hz $\frac{f_s}{f_{m\,z}}$ = 1.02 Δf_z = 187.65 hz I_z = 0.87

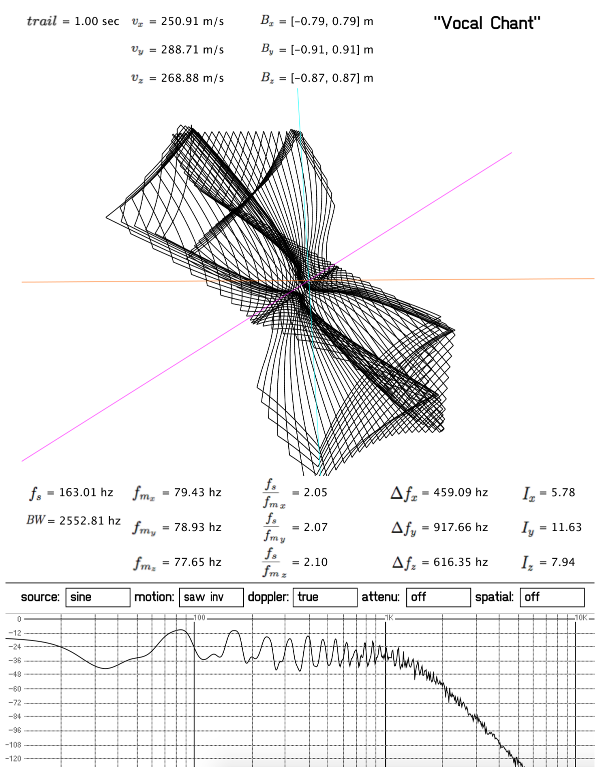
source: scan motion: saw inv doppler: symm attenu: off spatial: stereo


"Airy Synth"

$trail$ = 0.36 sec v_x = 211.93 m/s B_x = [−0.60, 0.60] m
v_y = 231.56 m/s B_y = [−0.69, 0.69] m
v_z = 229.02 m/s B_z = [−0.71, 0.71] m

f_s = 179.75 hz f_{m_x} = 177.43 hz $\frac{f_s}{f_{m\,x}}$ = 1.01 Δf_x = 112.05 hz I_x = 0.63
BW = 765.71 hz f_{m_y} = 168.12 hz $\frac{f_s}{f_{m\,y}}$ = 1.07 Δf_y = 122.42 hz I_y = 0.73
f_{m_z} = 161.13 hz $\frac{f_s}{f_{m\,z}}$ = 1.12 Δf_z = 121.08 hz I_z = 0.75

source: scan motion: square doppler: symm attenu: off spatial: off

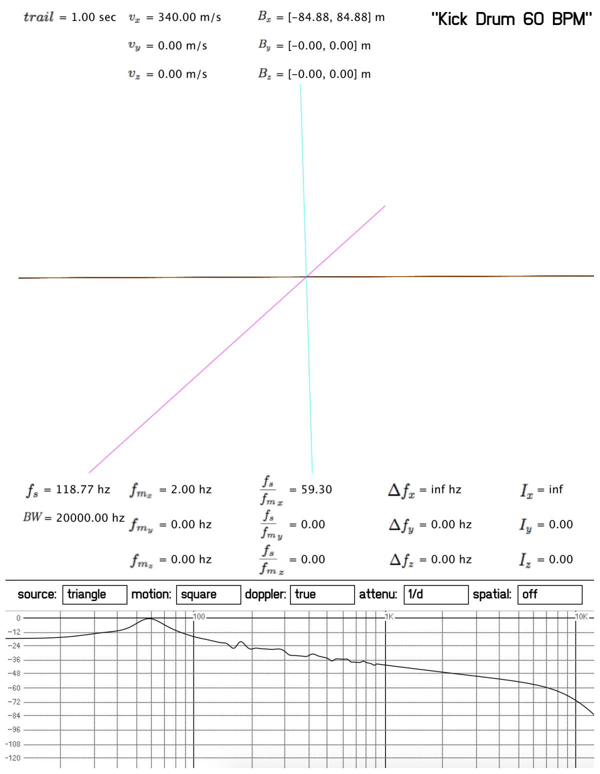

"Vocal Chant"

$trail$ = 1.00 sec v_x = 250.91 m/s B_x = [−0.79, 0.79] m
v_y = 288.71 m/s B_y = [−0.91, 0.91] m
v_z = 268.88 m/s B_z = [−0.87, 0.87] m

f_s = 163.01 hz f_{m_x} = 79.43 hz $\frac{f_s}{f_{m\,x}}$ = 2.05 Δf_x = 459.09 hz I_x = 5.78
BW = 2552.81 hz f_{m_y} = 78.93 hz $\frac{f_s}{f_{m\,y}}$ = 2.07 Δf_y = 917.66 hz I_y = 11.63
f_{m_z} = 77.65 hz $\frac{f_s}{f_{m\,z}}$ = 2.10 Δf_z = 616.35 hz I_z = 7.94

source: sine motion: saw inv doppler: true attenu: off spatial: off

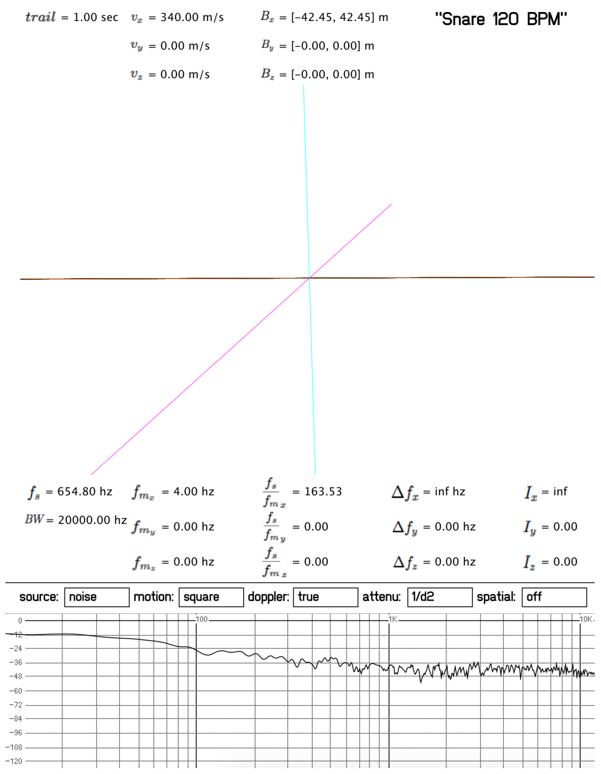
B.3 Percussive


"Kick Drum 60 BPM"

$trail$ = 1.00 sec v_x = 340.00 m/s B_x = [−84.88, 84.88] m
v_y = 0.00 m/s B_y = [−0.00, 0.00] m
v_z = 0.00 m/s B_z = [−0.00, 0.00] m

f_s = 118.77 hz f_{m_x} = 2.00 hz $\frac{f_s}{f_{m\,x}}$ = 59.30 Δf_x = inf hz I_x = inf
BW = 20000.00 hz f_{m_y} = 0.00 hz $\frac{f_s}{f_{m\,y}}$ = 0.00 Δf_y = 0.00 hz I_y = 0.00
f_{m_z} = 0.00 hz $\frac{f_s}{f_{m\,z}}$ = 0.00 Δf_z = 0.00 hz I_z = 0.00

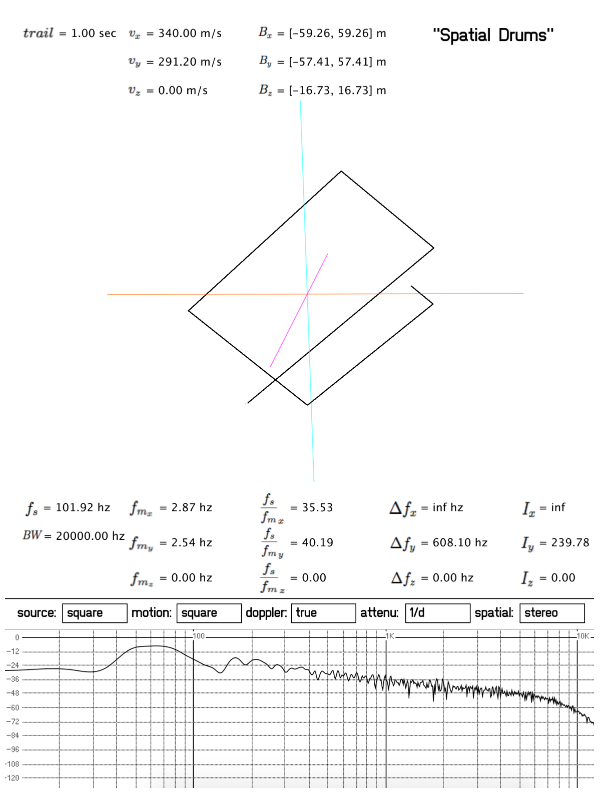
source: triangle motion: square doppler: true attenu: 1/d spatial: off


"Snare 120 BPM"

$trail$ = 1.00 sec v_x = 340.00 m/s B_x = [−42.45, 42.45] m
v_y = 0.00 m/s B_y = [−0.00, 0.00] m
v_z = 0.00 m/s B_z = [−0.00, 0.00] m

f_s = 654.80 hz f_{m_x} = 4.00 hz $\frac{f_s}{f_{m\,x}}$ = 163.53 Δf_x = inf hz I_x = inf
BW = 20000.00 hz f_{m_y} = 0.00 hz $\frac{f_s}{f_{m\,y}}$ = 0.00 Δf_y = 0.00 hz I_y = 0.00
f_{m_z} = 0.00 hz $\frac{f_s}{f_{m\,z}}$ = 0.00 Δf_z = 0.00 hz I_z = 0.00

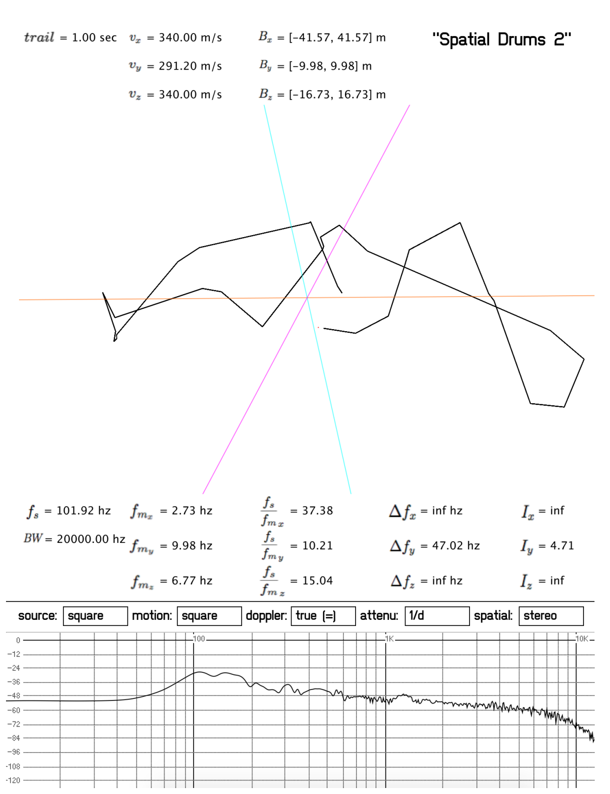
source: noise motion: square doppler: true attenu: 1/d2 spatial: off


"Spatial Drums"

trail = 1.00 sec v_x = 340.00 m/s B_x = [−59.26, 59.26] m
v_y = 291.20 m/s B_y = [−57.41, 57.41] m
v_z = 0.00 m/s B_z = [−16.73, 16.73] m

f_s = 101.92 hz f_{m_x} = 2.87 hz $\frac{f_s}{f_{m\,x}}$ = 35.53 Δf_x = inf hz I_x = inf
BW = 20000.00 hz f_{m_y} = 2.54 hz $\frac{f_s}{f_{m\,y}}$ = 40.19 Δf_y = 608.10 hz I_y = 239.78
f_{m_z} = 0.00 hz $\frac{f_s}{f_{m\,z}}$ = 0.00 Δf_z = 0.00 hz I_z = 0.00

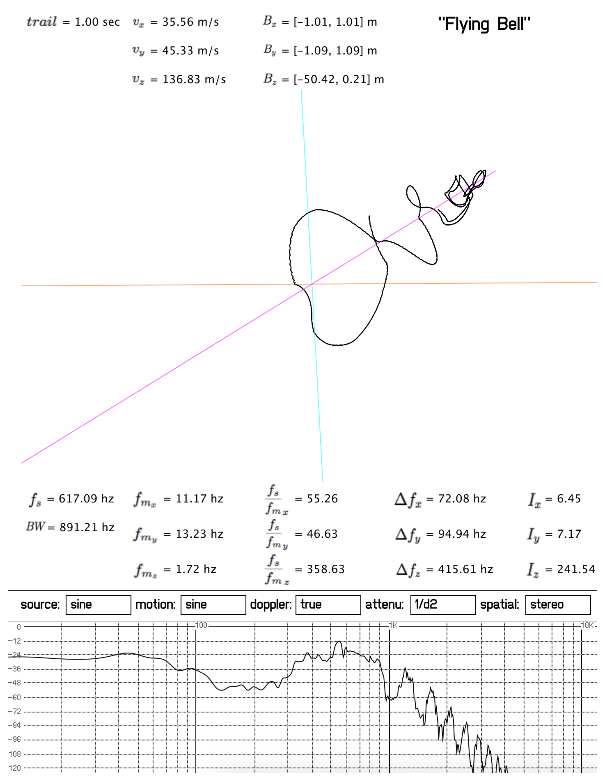
source: square motion: square doppler: true attenu: 1/d spatial: stereo


"Spatial Drums 2"

trail = 1.00 sec v_x = 340.00 m/s B_x = [−41.57, 41.57] m
v_y = 291.20 m/s B_y = [−9.98, 9.98] m
v_z = 340.00 m/s B_z = [−16.73, 16.73] m

f_s = 101.92 hz f_{m_x} = 2.73 hz $\frac{f_s}{f_{m\,x}}$ = 37.38 Δf_x = inf hz I_x = inf
BW = 20000.00 hz f_{m_y} = 9.98 hz $\frac{f_s}{f_{m\,y}}$ = 10.21 Δf_y = 47.02 hz I_y = 4.71
f_{m_z} = 6.77 hz $\frac{f_s}{f_{m\,z}}$ = 15.04 Δf_z = inf hz I_z = inf

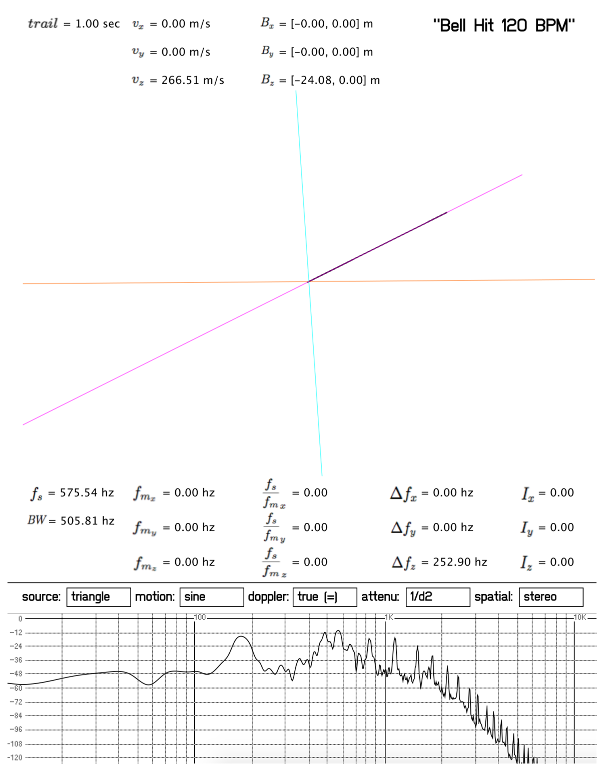
source: square motion: square doppler: true (=) attenu: 1/d spatial: stereo


"Flying Bell"

trail = 1.00 sec v_x = 35.56 m/s B_x = [−1.01, 1.01] m
v_y = 45.33 m/s B_y = [−1.09, 1.09] m
v_z = 136.83 m/s B_z = [−50.42, 0.21] m

f_s = 617.09 hz f_{m_x} = 11.17 hz $\frac{f_s}{f_{m\,x}}$ = 55.26 Δf_x = 72.08 hz I_x = 6.45
BW = 891.21 hz f_{m_y} = 13.23 hz $\frac{f_s}{f_{m\,y}}$ = 46.63 Δf_y = 94.94 hz I_y = 7.17
f_{m_z} = 1.72 hz $\frac{f_s}{f_{m\,z}}$ = 358.63 Δf_z = 415.61 hz I_z = 241.54

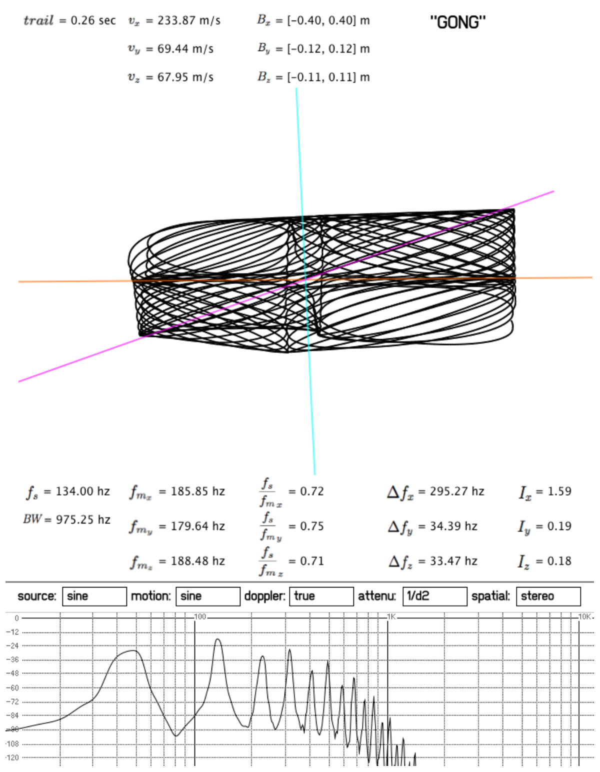
source: sine motion: sine doppler: true attenu: 1/d2 spatial: stereo


"Bell Hit 120 BPM"

trail = 1.00 sec v_x = 0.00 m/s B_x = [−0.00, 0.00] m
v_y = 0.00 m/s B_y = [−0.00, 0.00] m
v_z = 266.51 m/s B_z = [−24.08, 0.00] m

f_s = 575.54 hz f_{m_x} = 0.00 hz $\frac{f_s}{f_{m\,x}}$ = 0.00 Δf_x = 0.00 hz I_x = 0.00
BW = 505.81 hz f_{m_y} = 0.00 hz $\frac{f_s}{f_{m\,y}}$ = 0.00 Δf_y = 0.00 hz I_y = 0.00
f_{m_z} = 0.00 hz $\frac{f_s}{f_{m\,z}}$ = 0.00 Δf_z = 252.90 hz I_z = 0.00

source: triangle motion: sine doppler: true (=) attenu: 1/d2 spatial: stereo



"GONG"

$trail$ = 0.26 sec v_x = 233.87 m/s B_x = [−0.40, 0.40] m

v_y = 69.44 m/s B_y = [−0.12, 0.12] m

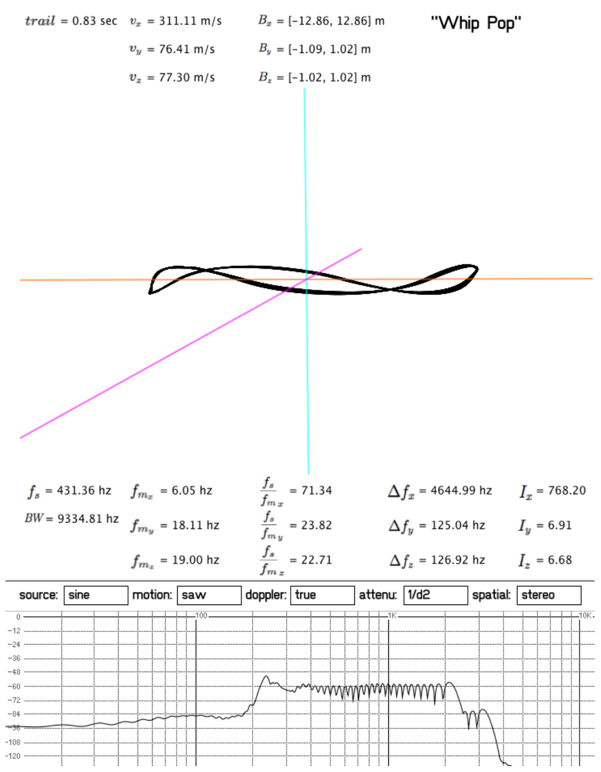
v_z = 67.95 m/s B_z = [−0.11, 0.11] m

f_s = 134.00 hz f_{m_x} = 185.85 hz $\frac{f_s}{f_{m\,x}}$ = 0.72 Δf_x = 295.27 hz I_x = 1.59

BW = 975.25 hz f_{m_y} = 179.64 hz $\frac{f_s}{f_{m\,y}}$ = 0.75 Δf_y = 34.39 hz I_y = 0.19

f_{m_z} = 188.48 hz $\frac{f_s}{f_{m\,z}}$ = 0.71 Δf_z = 33.47 hz I_z = 0.18

source: sine motion: sine doppler: true attenu: 1/d2 spatial: stereo



"Whip Pop"

$trail$ = 0.83 sec v_x = 311.11 m/s B_x = [−12.86, 12.86] m

v_y = 76.41 m/s B_y = [−1.09, 1.02] m

v_z = 77.30 m/s B_z = [−1.02, 1.02] m

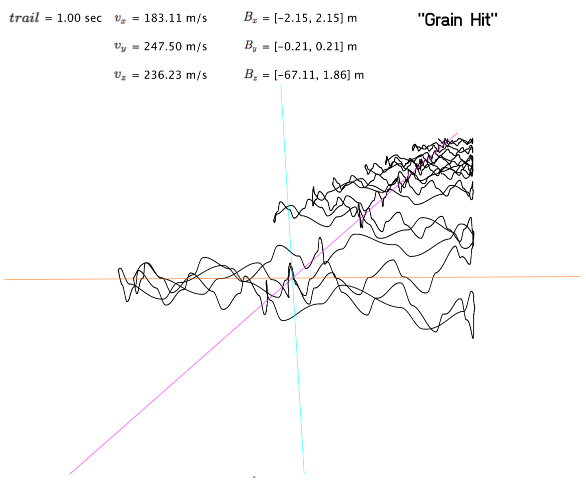
f_s = 431.36 hz f_{m_x} = 6.05 hz $\frac{f_s}{f_{m\,x}}$ = 71.34 Δf_x = 4644.99 hz I_x = 768.20

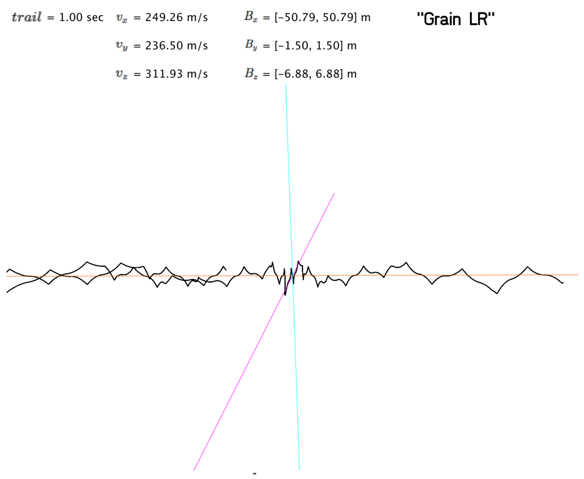
BW = 9334.81 hz f_{m_y} = 18.11 hz $\frac{f_s}{f_{m\,y}}$ = 23.82 Δf_y = 125.04 hz I_y = 6.91

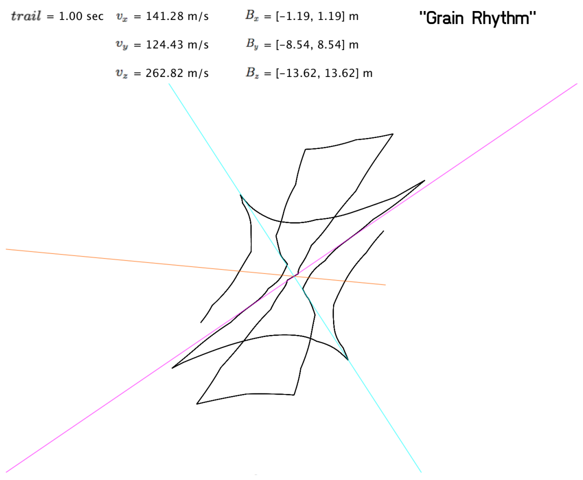
f_{m_z} = 19.00 hz $\frac{f_s}{f_{m\,z}}$ = 22.71 Δf_z = 126.92 hz I_z = 6.68

source: sine motion: saw doppler: true attenu: 1/d2 spatial: stereo

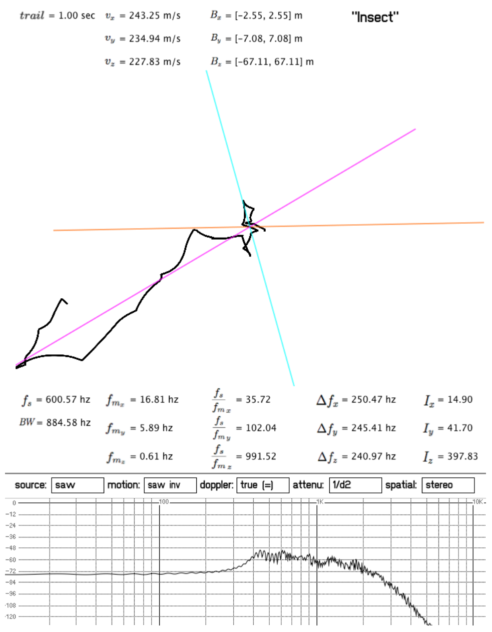
B.4 Granular


$trail$ = 1.00 sec
v_x = 183.11 m/s
v_y = 247.50 m/s
v_z = 236.23 m/s
B_x = [-2.15, 2.15] m
B_y = [-0.21, 0.21] m
B_z = [-67.11, 1.86] m
"Grain Hit"


$trail$ = 1.00 sec
v_x = 249.26 m/s
v_y = 236.50 m/s
v_z = 311.93 m/s
B_x = [-50.79, 50.79] m
B_y = [-1.50, 1.50] m
B_z = [-6.88, 6.88] m
"Grain LR"


$trail$ = 1.00 sec
v_x = 141.28 m/s
v_y = 124.43 m/s
v_z = 262.82 m/s
B_x = [-1.19, 1.19] m
B_y = [-8.54, 8.54] m
B_z = [-13.62, 13.62] m
"Grain Rhythm"

B.5 Sound Effects



"Insect"

trail = 1.00 sec v_x = 243.25 m/s v_y = 234.94 m/s v_z = 227.83 m/s

B_x = [-2.55, 2.55] m B_y = [-7.08, 7.08] m B_z = [-67.11, 67.11] m

f_s = 600.57 hz *BW* = 884.58 hz

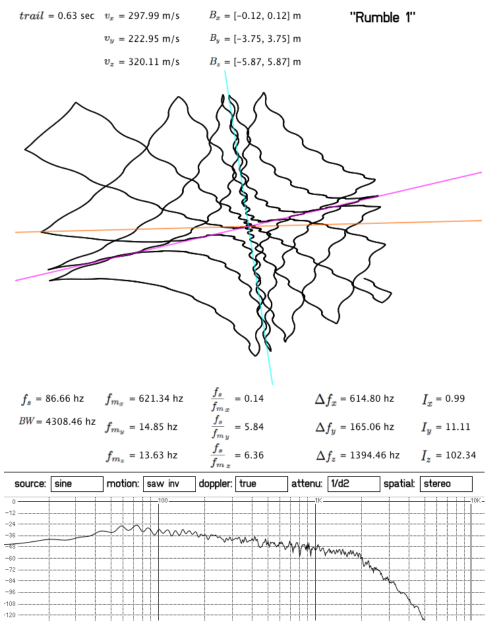
f_{m_x} = 16.81 hz f_{m_y} = 5.89 hz f_{m_z} = 0.61 hz

$\frac{f_s}{f_{m_x}}$ = 35.72 $\frac{f_s}{f_{m_y}}$ = 102.04 $\frac{f_s}{f_{m_z}}$ = 991.52

Δf_x = 250.47 hz Δf_y = 245.41 hz Δf_z = 240.97 hz

I_x = 14.90 I_y = 41.70 I_z = 397.83

source: saw motion: saw inv doppler: true (=) attenu: 1/d2 spatial: stereo



"Rumble 1"

trail = 0.63 sec v_x = 297.99 m/s v_y = 222.95 m/s v_z = 320.11 m/s

B_x = [-0.12, 0.12] m B_y = [-3.75, 3.75] m B_z = [-5.87, 5.87] m

f_s = 86.66 hz *BW* = 4308.46 hz

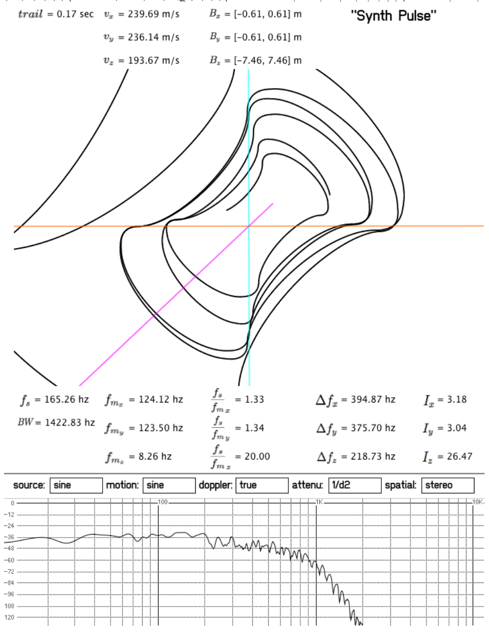
f_{m_x} = 621.34 hz f_{m_y} = 14.85 hz f_{m_z} = 13.63 hz

$\frac{f_s}{f_{m_x}}$ = 0.14 $\frac{f_s}{f_{m_y}}$ = 5.84 $\frac{f_s}{f_{m_z}}$ = 6.36

Δf_x = 614.80 hz Δf_y = 165.06 hz Δf_z = 1394.46 hz

I_x = 0.99 I_y = 11.11 I_z = 102.34

source: sine motion: saw inv doppler: true attenu: 1/d2 spatial: stereo



"Synth Pulse"

trail = 0.17 sec v_x = 239.69 m/s v_y = 236.14 m/s v_z = 193.67 m/s

B_x = [-0.61, 0.61] m B_y = [-0.61, 0.61] m B_z = [-7.46, 7.46] m

f_s = 165.26 hz *BW* = 1422.83 hz

f_{m_x} = 124.12 hz f_{m_y} = 123.50 hz f_{m_z} = 8.26 hz

$\frac{f_s}{f_{m_x}}$ = 1.33 $\frac{f_s}{f_{m_y}}$ = 1.34 $\frac{f_s}{f_{m_z}}$ = 20.00

Δf_x = 394.87 hz Δf_y = 375.70 hz Δf_z = 218.73 hz

I_x = 3.18 I_y = 3.04 I_z = 26.47

source: sine motion: sine doppler: true attenu: 1/d2 spatial: stereo

Appendix C

Code Excerpts

C.1 *W.A.N.T.S.* MATLAB Spatialization Script

Listing C.1: MATLAB Spatialization Script for W.A.N.T.S.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Octaphonic Panner
% Ryan McGee 2009

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Input Sound File
snd = wavread('whisper1.wav'); %input sound file
sndsize = size(snd); % size of sound mactrix [samples, channels]
ss = sndsize(1); %number of samples (MATLAB is 1 indexed)
FS = 44100; % sampling rate for input sound files

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Speaker Layout
startdeg = 90; % starting position in degrees (90 is in front of listener)
C1=112.5; % octaphonic speaker positions in degrees
```

```
C2=67.5;
C3=247.5;
C4=292.5;
C5=157.5;
C6=22.5;
C7=202.5;
C8=337.5;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Must run 8 times per sound, changing Cdeg for each speaker/channel)
Cdeg = C1; % position of speaker for channel to be rendered
direction = 1; % 1 for CCW, -1 for CW
numCircles = 8; % number of circles or spiral rotations
wavname = 'whisper1_fib8CCW_Ch1.wav'; %name of output file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Required length of Fibonacci seq based on number of rotations
fibNumIndex = 4*numCircles + 1;
samplesPerCircle = round(ss/numCircles);
samplesPerFibArc = round(samplesPerCircle/4);
lastFibArcStartIndex = 1; %sample index for beginning of last arc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Matrix allocation
fib = zeros(fibNumIndex, 1); %array of fibonacci numbers
theta = zeros(ss, 1); %angle of sound
R = zeros(ss, 1); % radius
unitDistance = zeros(ss, 1); % distance multiplier
sndC = zeros(ss, 1); %output sound array
time = 1:ss;
time = time./FS;
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Fibonacci sequence generation
fib(1) = 1;
fib(2) = 1;
for i = 3:fibNumIndex;
    fib(i) = fib(i-1) + fib(i-2);
end
R(1) = fib(fibNumIndex);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute theta and the radius, R, for each sample
for i=1:ss;
    theta(i) = direction*round(((360*numCircles/ss)*i)) + startdeg;
    %if theta is a multiple of 90 degrees
    if(mod(i, samplesPerFibArc) == 0 && i ~= ss)
       fibNumIndex = fibNumIndex - 1;
       lastFibArcStartIndex = i;
       R(i) = fib(fibNumIndex); % R is a fibonacci number
       R(i)
    else
        if(i == 1)
            R(i) = fib(fibNumIndex); %initial R
        else
            if (fibNumIndex > 2)
                R(i) = fib(fibNumIndex) - (fib(fibNumIndex-2) *
                          (i-lastFibArcStartIndex)/(samplesPerFibArc));
            else
                R(i) = 1;
            end
        end
```

```matlab
    end

    %output samples processed to monitor processing time
    if(mod(i, 1000) == 0)
        i
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute the distance for each sample
unitDistance = sqrt((cosd(theta) - cosd(Cdeg)).^2 +
                    (sind(theta) - sind(Cdeg)).^2);

% Apply gain multiplier for spiral:
sndC(:, 1) = (1./sqrt(R)).*(1 - 0.5.*unitDistance).*snd(:, 1);
% or
% Apply gain multiplier for circle:
%sndC(:, 1) = (1 - 0.5.*unitDistance).*snd(:, 1);

% Write the final wav file for this channel
wavwrite(sndC, FS, wavname);
```

Bibliography

[1] J. Chowning, *The simulation of moving sound sources*, *Computer Music Journal* **1** (1977), no. 3 48–52.

[2] J.-C. Risset, *Max mathews's influence on (my) music*, *Computer Music Journal* **33** (2009), no. 3 26–36.

[3] H. Brant, *Space as an essential aspect of musical composition*, in *Contemporary Composers on Contemporary Music* (B. C. Elliott Schwartz, ed.), pp. 221–242. Da Capo Press, 1967.

[4] C. Roads, *Sound composition with pulsars*, *Journal of the Audio Engineering Society* **49** (2001), no. 3 134–146.

[5] S. Wilson, *Spatial swarm granulation*, in *Proc. Int. Computer Music Conf.*, (Belfast), 2008.

[6] J. B. Marlon Schumacher, *Spatial sound synthesis in computer-aided composition*, *Organized Sound* **15** (2010), no. 1 271–278.

[7] B. Verplank, M. Mathews, and R. Shaw, *Scanned Synthesis*, in *International Computer Music Conference*, 2000.

[8] C. Roads, *Wave terrain synthesis*, in *The Computer Music Tutorial*, pp. 163–167. MIT Press, 1996.

[9] C. Roads, *Graphic Sound Synthesis*, *The Computer Music Tutorial* (1996) 329–334.

[10] B. Evans, *Foundations of a Visual Music*, *Computer Music Journal* **29** (2005), no. 4 11–24.

[11] A. Abbado, *Perceptual correspondences of abstract animation and synthetic sound*, Master's thesis, Massachusetts Institute of Technology, 1988.

[12] T. Hermann, A. Hunt, and J. G. Neuhoff, *The Sonification Handbook*. Logos Verlag, 2011.

[13] V. Pulkki, *Virtual sound source positioning using vector base amplitude panning*, *Journal of the Audio Engineering Society* **45** (1997), no. 6 456–466.

[14] 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.

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

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

[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*, pp. 314–317, 2001.

[18] R. McGee, *Sound element spatializer*, in *Proceedings of 2011 International Computer Music Conference*, 2011.

[19] 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.

[20] S. Wilson and J. Harrison, *Rethinking the BEAST: Recent developments in multichannel composition at Birmingham ElectroAcoustic Sound Theatre*, *Organised Sound* **15** (Oct., 2010) 239–250.

[21] C. Ramakrishnan, "Zirkonium."

[22] 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*, pp. 294–295, 1995.

[23] M. Schumacher and J. Bresson, *Spatial Sound Synthesis in Computer-Aided Composition*, *Organised Sound* **15** (Oct., 2010) 271–289.

[24] N. Peters, S. Ferguson, and S. McAdams, *Towards a Spatial Sound Description Interchange Format (SpatDIF)*, *Canadian Acoustics* **35** (2007), no. 3 64–65.

[25] C. Agon, *OpenMusic: un langage de programmation visuelle pour la composition musicale.* PhD thesis, Université Pierre et Marie Curie, Paris 6, France., 1998.

[26] G. S. Kendall, *The Decorrelation of Audio Signals and Its Impact on Spatial Imagery*, *Computer Music Journal* **19** (Jan., 1995) 71–87.

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

[28] 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.

[29] R. McGee, *Sensynth: a mobile application for dynamic sensor to sound mapping*, in *Proceedings of 2012 New Interfaces for Musical Expression Conference*, 2012.

[30] A. Hunt, M. M. Wanderley, S. S. West, and M. Paradis, *The importance of parameter mapping in electronic instrument design*, in *New Interfaces for Musical Expression*, pp. 1–6, 2002.

[31] D. Wessel, M. Wright, and J. Schott, *Intimate Musical Control of Computers with a Variety of Controllers and Gesture Mapping Metaphors*, in *New Interfaces for Musical Expression*, pp. 192–194, 2002.

[32] D. Overholt, C. Roads, and J. Thompson, *On Musical Gestures and New Performance Interfaces for Electronic Music*, in *International Gesture Workshop*, April, 2003.

[33] L. Gaye, R. Maze, and L. E. Holmquist, *Sonic City : The Urban Environment as a Musical Interface*, in *New Interfaces for Musical Expression*, pp. 109–115, 2003.

[34] R. Jacobs, M. Feldmeier, and J. Paradiso, *A Mobile Music Environment Using a PD Compiler and Wireless Sensors*, in *New Interfaces for Musical Expression*, pp. 193–196, 2008.

[35] G. Essl, *SpeedDial : Rapid and On-The-Fly Mapping of Mobile Phone Instruments*, in *New Interfaces for Musical Expression*, pp. 270–273, 2009.

[36] G. Essl and A. Müller, *Designing Mobile Musical Instruments and Environments with urMus*, in *New Interfaces for Musical Expression*, pp. 76–81, 2010.

[37] N. J. Bryan, J. Herrera, J. Oh, and G. Wang, *MoMu : A Mobile Music Toolkit*, in *New Interfaces for Musical Expression*, pp. 174–177, 2010.

[38] G. Kramer, *Auditory Display: Sonification, Audification, and Auditory Interfaces.* Addison-Wesley, 1994.

[39] C. Roads, *The Computer Music Tutorial.* MIT Press, 1996.

[40] M. S. OModhrain, *Playing by Feel: Incorporating Haptic Feedback into Computer-Based Musical Instruments.* PhD thesis, Stanford University, 2000.

[41] D. Ashbrook, P. Baudisch, and S. White, *Nenya: Subtle and Eyes-Free Mobile Input with a Magnetically-Tracked Finger Ring*, in *Annual Conference on Human Factors in Computing Systems*, pp. 2043–2046, 2011.

[42] R. McGee, *Voice of sisyphus: An image sonification multimedia installation*, in *Proceedings of International Conference on Auditory Display*, 2012.

[43] R. McGee, *Vosis: a multi-touch image sonification interface*, in *Proceedings of 2013 New Interfaces for Musical Expression Conference*, 2013.

[44] B. Schneider, *On Hearing Eyes and Seeing Ears: A Media Aesthetics of Relationships Between Sound and Image*, *See this Sound: Audiovisuology 2* (2011) 174–199.

[45] W. S. Yeo and J. Berger, *A Framework for Designing Image Sonification Methods*, in *Proceedings of International Conference on Auditory Display*, 2005.

[46] W. S. Yeo and J. Berger, *Raster Scanning : A New Approach to Image Sonification, Sound Visualization, Sound Analysis And Synthesis*, in *Proceedings of the International Computer Music Conference*, 2006.

[47] W. Jones, *Sight for Sore Ears*, *IEEE Spectrum* (February, 2004).

[48] M. Chion, W. Murch, and C. Gorbman, *Audio-Vision: Sound on Screen.* Columbia University Press, 1994.

[49] C. Hayward, *Listening to the earth sing*, in *Auditory Display: Sonification, Audification, and Auditory Interfaces.* Addison-Wesley, 1994.

[50] R. McGee, *Spatial modulation synthesis*, in *Proceedings of 2015 International Computer Music Conference*, 2015.

[51] V. Pulkki, *Virtual sound source positioning using vector base amplitude panning*, *Journal of the Audio Engineering Society* **45** (1997), no. 6 456–466.

[52] M. Gerzon, *With-height sound reproduction*, *Journal of the Audio Engineering Society* **21** (1973), no. 1 2–10.

[53] J. A. Julius Smith, Stefania Serafin, *Doppler simulation and the leslie*, in *Proc. Int. Conf. on Digital Audio Effects*, (Hamburg), 2002.

[54] C. Roads, *Modulation synthesis*, in *The Computer Music Tutorial*, ch. 6, pp. 239–240. MIT Press, 1996.

[55] J. Chowning, *The synthesis of complex audio spectra by means of frequency modulation*, *Journal of the Audio Engineering Society* **21** (1973), no. 7 526–534.

[56] J. T. Victor Lazzarini and T. Lysaght, *Adaptive fm synthesis*, in *Proceedings of the 10th International Conference on Digital Audio Effects*, 2007.

[57] J. Dattorro, *Effect design: Part 2 delay-line modulation and chorus*, 1997.

[58] J. R. Carson, *Notes on the theory of modulation*, in *Proceedings of the Institute of Radio Engineers*, vol. 10, pp. 57–64, 1922.

[59] C. Roads, *Varieties of particle synthesis*, in *Microsound*, ch. 4. MIT Press, 2001.

[60] G. Wakefield and W. Smith, *Cosm: A toolkit for composing immersive audio-visual worlds of agency and autonomy*, in *Proceedings of the International Computer Music Conference*, 2011.