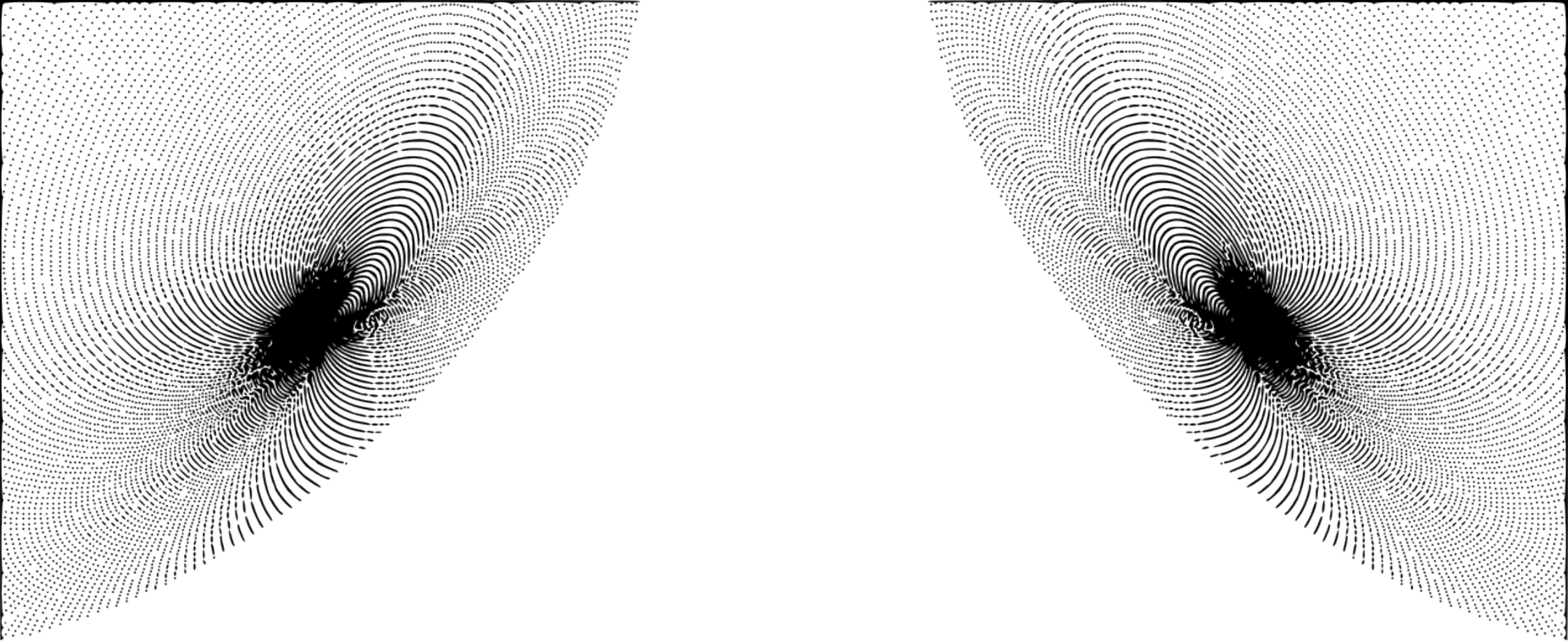
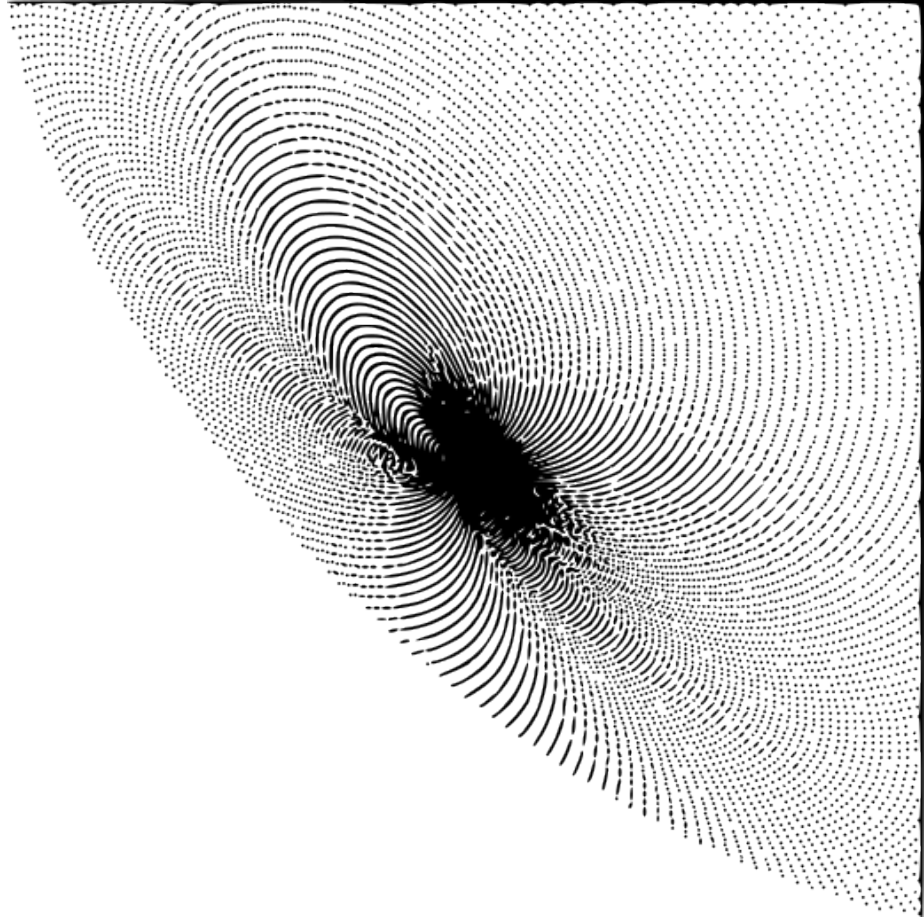


SOUND SPECTACLE

Petros Tatsiopoulos, LibreOffice, SuperCollider, 2024



Special Thanks

My relatives and friends
Anastasios Nyfadopoulos
Giannis Karytinos
Viktoria Elositou
Paul Modler
Jonathan Bepler
Anja Dorn
Georgia Kotretsos
Jennifer Nelson
Timothy Ward
Sotirios Bahtsetzis
Panos Kiousis
Eli Fieldsteel
scsynth forum
3blue1brown and Ben Eater
numberphile
Mike Hockney
Eleutheria Papagergiou
Peter J. Caroll
Eric L'Homme
Giorgos Stamatelopoulos
Elisavet Fakatseli
Su Real
Hertz Lab Institute Karlsruhe

Introduction

The following essay is about a sound artwork realized as a multichannel electroacoustic installation. Sound art is commonly understood as an interdisciplinary form that raises questions on the use of public space, the composer's status and the very nature of an art object or musical composition. This artistic category is situated, without clear borders, between visual arts, music and performance. The practice-based research, design and realization of the further on described installation aim to tackle each of the above mentioned elements.

A sound spectacle has as its medium sound, which in this case, refers both to electrical signals inside a circuit and acoustic waves. Thus by definition the medium, being present in the air that surrounds us and we breath, is not only time-based but also immersive and propagative in real space, meaning the viewer exists inside the frame of the artwork. This way, the public space, the exhibition space, the positive and negative space inside the artwork, all come into question and are redefined. In these spaces, the virtuality of a sound installation is opposed not to the real but to the actual. The virtual of the artwork of extended frame, is real without being actual, ideal without being an abstract and symbolic without being fictional.

A sound artwork defines the medium of sound in an abstract form, somewhat devoid of intent, category and predefined use, existing in the realm where the medium is the message and the message is of mental and not practical nature. Thus, sound being the main focus, the nature of the art object, contains elements of categorized methods like music, and works as a space of invention and exploration of new methods for such elements. The composition can extend in further microcosms or macrocosms incorporate new ideas, variables and means of production. Such an example would be electroacoustic sound where the composition can become algorithmic, the production mechanical and the performance automated, changing the role of the composer from eg the dictator of events to an architect, an engineer, an explorer of patterns and acousmatic spaces, a performer.

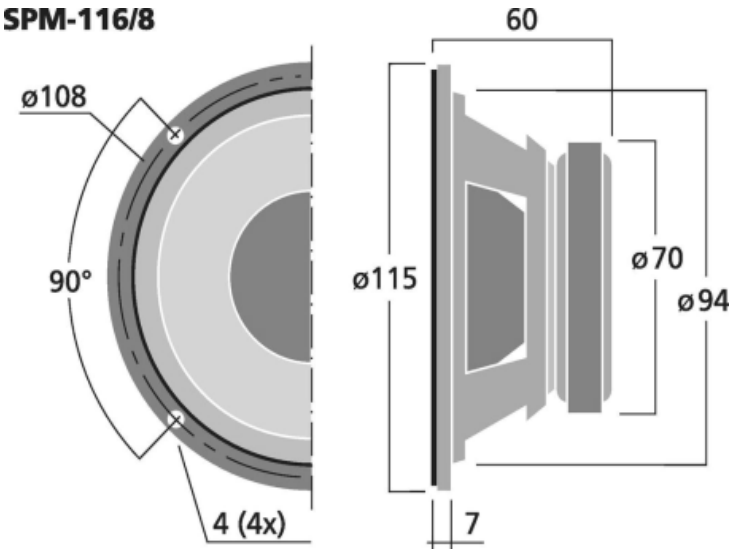
Although pictures and visual art in general can exist without inherent acoustic stimuli, sound art has a high dependence on visual stimuli. They affect and define the perception of a sound source, either that is direction, or the timbre of an instrument. Visual and sculptural elements help define the space of a sound artwork, they are the machines that excite the air space acoustically and form the building blocks of a sound installation. Vision cannot exist without light and sound cannot without propagating air waves and molecules.

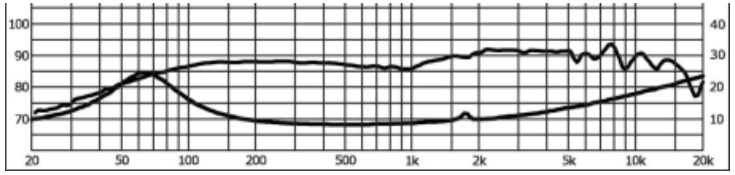
Why spectacle? The main coping mechanism of a person in the modern world, that is until this day, is the spectacle. Everything about and in life, information, culture, religion, wealth, products, events have to be made into a spectacle in order to be copied with, we live in a society of spectacle. Walter Benjamin's so called aura of the artwork was and is not lost through the means of mechanical reproduction. In contrast, modernity and its industrial revolutions have made possible to expand the amount of available auras, the amount in quantity, quality, identity, location all over the earth and presence in time. These expansions in turn have made possible the expansion of the spectacle as a main means of coping with the actual, taking territory out of other methods like religion, politics even science and technology. Thus everything becomes a spectacle, has to gain a virtuality in order to be copied with in the development of Consciousness. Nevertheless and in any case a spectacle has to be something reasonable and statically stable, where all levering forces are balanced and in place.

The following chapters focus on the description of a sound spectacle, with its specific parts, its synthesis as an artwork. This sound installation is an artwork whose essence is to make a spectacle out of sound whose aim is to visualize the actual. The only thing mediated is the medium itself, through electrical signals, gates and refractors between the medium, air, and the mediation. This electroacoustic, sculptural and architectural artwork is presented and justified in three chapters, consisting of the software used for the composition of its signal, the hardware and circuitry used where the electronic signal flows until it is transformed to acoustic energy in air, and the incorporation of this system into an installation with specific format.

Works Consulted

- [Oliver_Grau]_Virtual_Art_From_Illusion_to_Immers(BookFi)
- Deleuze on Virtuality
- (The MIT Press) Scott Wilson (editor), David Cottle (editor), Nick Collins (editor) - The SuperCollider Book (The MIT Press)-The MIT Press (2011)
- benjamin
- Bernhard Leitner/ Sound Spaces | ArchDaily
- friedberg-the-virtual-window-from-alberti-to-microsoft-2006
- Helguera-Pablo_Socially-Engaged-Art
- Mario Livio - The Golden Ratio_ The Story of PHI, the World's Most Astonishing Number-Broadway Books (2003)
- Michael Gerzon
- nicola bouriad postproduction
- Nominalism in Metaphysics
- philolaus
- the noise instruments of luigi roussolo
- Wardrip-Fruin_Noah_Montfort_Nick_eds_The_New_Media_Reader
- Xenakis_Iannis_Formalized_Music_Thought_and_Mathematics_in_Composition
- Nicomachus
- Gilles Deleuze, Diffierence and Repetition
- Berkeley, Principles of Human Knowledge
- Simon Emmerson, The Language of Electroacoustic Music
- Robert Sekuler, Randolph Blake, Perception





On SuperCollider

SuperCollider is an audio server, programming language, and IDE for sound synthesis and algorithmic composition. Originally released in 1996 by James McCartney for real-time audio synthesis and algorithmic composition, since then it has been evolving into a system used and further developed by both scientists and artists working with sound. It is a dynamic programming language providing a framework for acoustic research, algorithmic music, interactive programming and live coding. First released under the terms of the GPL-2.0-or-later in 2002, and from version 3.4 under GPL-3.0-or-later, SuperCollider is free and open-source software.

On thing to consider before running any program in SuperCollider is, apart from a general understanding of the Server Architecture, some important options and settings for the audio server. In the following example options are going to be given for a specific algorithmic and multichannel composition.

```
(
s.options.numWireBufs = 1024*16;
s.options.numOutputBusChannels=4;
s.options.numInputBusChannels=0;
TempoClock.default.tempo_(120/120);
s.options.memSize = 8192*64;
s.options.blockSize = 16;
s.options.maxLogins = 32;
s.waitForBoot
)
```

In this case the numWireBufs option sets the maximum number of buffers that are allocated to interconnect unit generators. This sets the limit of complexity of SynthDefs that can be loaded at runtime. The default is 64 and it has to be a power of two. NumOutputBusChannels and numInputBusChannels sets the amount of output and input channels respectively. Then the overall tempo is set e.g. for patterns and sequences that contain events with durations, memSize which is the number of kilobytes of real time memory allocated to the server. This memory is used to allocate synths and any memory that unit generators themselves allocate and the default is 8192 again being always a power of two. Finally blockSize and maxLogins options set the number of samples in one control period and the Integer indicating the maximum number of clients which can simultaneously receive notifications from the server, respectively. After all options are correct and set the Server can be booted so it can produce sound. Note that SuperCollider can host many different Servers apart from the default. Each Server has a limited processing power so if more of that is needed, different tasks, that is bunches of code, can run in different Servers simultaneously. Alternatively if an audio file is to be produced, different tasks can run and be recorded one at a time on the same Server, and the various component audio files of the composition can be mixed at a later point, considering all time signatures are correct. The below methods describe how to set different servers and make added SynthDefs readable to them.

```
(
~s1=Server.new(\server1,NetAddr("localhost", 57101),Server.local.options);
~s2=Server.new(\server2,NetAddr("localhost", 57102),Server.local.options);
~s3=Server.new(\server3,NetAddr("localhost", 57103),Server.local.options);
);
```

```
(~s1.boot;~s2.boot;~s3.boot;);

(
~s1.makeGui(w);
~s2.makeGui(w);
~s3.makeGui(w);
);

(
SynthDescLib.global.addServer(~s1);
SynthDescLib.global.addServer(~s2);
SynthDescLib.global.addServer(~s3);
);
```

Then an allocated SynthDef can be played on a specific Server by sending the appropriate message to that Server as below,

```
~s1.sendMsg("/s_new", "synth", x = s.nextNodeID, 1, 1);
```

or via a Pattern by adding

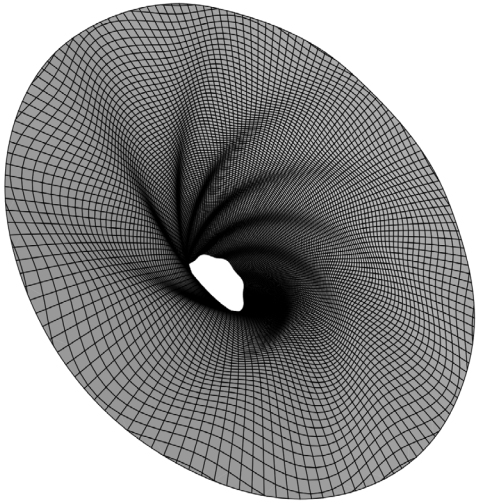
```
 \server,~s1
```

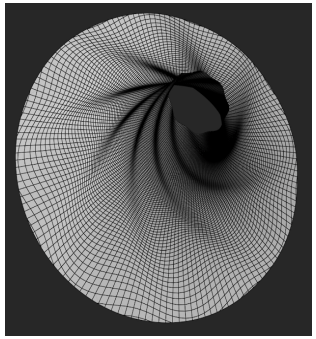
to its Pbind. SynthDefs and Patterns will be made clearer through further given examples corresponding to the composition presented in this chapter. The SuperCollider help-files are always helpful.

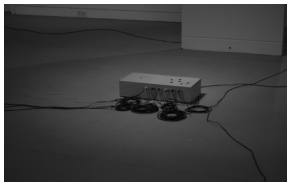
The server application uses synth definitions as templates for creating Synth nodes. In essence SynthDefs are used in this composition as the part of the code responsible for representing the synthesizer responsible for the production of sounds. All elements and synthesis of the sound, controlled arguments, default variables and iterations happen inside the SynthDefs which are then triggered and played according to patterns in their corresponding Pdefs, which in turn are a class that provides an interface to its superclass EventPatternProxy, keeping a reference to a stream that can be replaced while playing. They can be thought of as algorithmic sequencers.

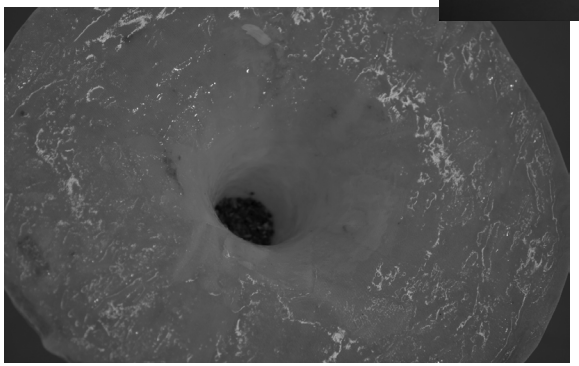
In the realm of this particular composition three SynthDefs are going to be used and because all are same, working as different instances or voices of the same code, one example will be given and explained. As you will later notice, each SynthDef uses VBAP to accomplish multichannel expansion in three dimensional coordinates. VBAP is an implementation of Vector Base Amplitude Panning. It allows for equal power panning of a source over an arbitrary array of equidistant speakers. Normally this would be a ring, a dome, or partial dome. VBAP was created by Ville Pulkki.

So before the allocation of any SynthDef, first an array of speaker coordinates has to be specified inside a Buffer which can later be used by the VBAP object inside the SynthDef, in order to pan the signals accordingly. In the below example two arrays are given. One gives symmetric coordinates forming a tetrahedron, while the other uses random ones on a sphere. While randomness can have interesting results, as will later be described, the second setup can happen to yield coordinates, very close to each other, which might be an unwanted coordinate space for the corresponding to it composition and speaker arrangement.









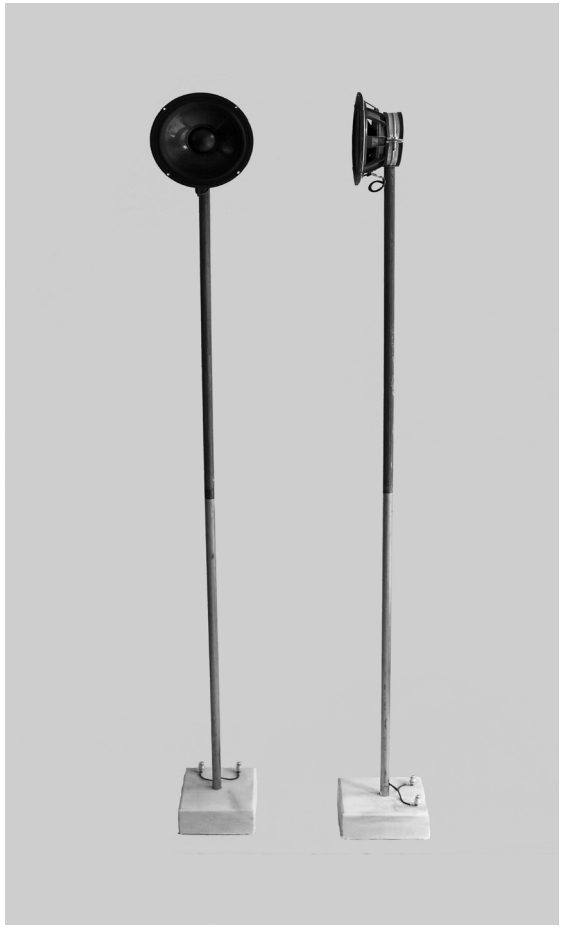
```
(
// 3D
~a = VBAPSpeakerArray.new(3, [[0, 45], [180, 45], [90, 45.neg], [90.neg, 45.neg]]);
~b = Buffer.loadCollection(s, ~a.getSetsAndMatrices);
);

(
// 3D random
~a = VBAPSpeakerArray.new(3, [[180.rand2, 90.rand2], [180.rand2, 90.rand2], [180.rand2,
90.rand2], [180.rand2, 90.rand2]]);
~b = Buffer.loadCollection(s, ~a.getSetsAndMatrices);
);
```

As such, with the speaker array formed, below you will find a SynthDef example.

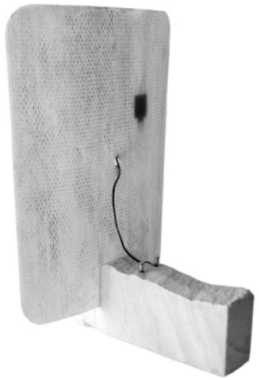
```
(
SynthDef.new(\oa, {

	arg  n1=1, cutt=440,decay=6,attack=0.05,x=1,y=0.5,z=0.25,rx=1,ry=1,rz=1,rx1=1,
ry1=1,rz1=1,ng1=1,ng2=1,ng3=1,ng4=1,ng5=1,ng6=1;

	var q,q1,i,j,k,i1,j1,k1,va,vb,cva,cvb,phx,phy;
	var freq, env1,env2,m=1,bank, pitch, fund, angle=0,angle1=90,r1,r2,p;
	var osc1_x,osc1_y,ang=(360-(360/1.6180339887499));

	var wrappedOut1 = { |busArray, signalArray|
	[busArray, signalArray].flop.do { |pair|
		Out.ar(pair[0], pair[1])}};
	var wrappedOut2 = { |busArray, signalArray|
	[busArray, signalArray].flop.do { |pair|
		Out.ar(pair[0], pair[1])}};

	var out = NamedControl.kr(\out, [0, 1, 2, 3]);  // multiple buses!

	r1=[rx,ry,rz].normalizeSum;
	r2=[rx1,ry1,rz1].normalizeSum;

	i = Quaternion(0, r1.[0].sqrt, 0, 0);
	j = Quaternion(0, 0,r1.[1].sqrt, 0);
	k = Quaternion(0, 0, 0, r1.[2].sqrt);

	i1 = Quaternion(0, r2.[0].sqrt, 0, 0);
	j1 = Quaternion(0, 0,r2.[1].sqrt, 0);
	k1 = Quaternion(0, 0, 0, r2.[2].sqrt);

	p=[x,y,z].normalizeSum;

	va=Quaternion(0,p.[0].sqrt*ng1,p.[1].sqrt*ng2,p.[2].sqrt*ng3);
	vb=Quaternion(0,p.[0].sqrt*ng4,p.[1].sqrt*ng5,p.[2].sqrt*ng6);

	bank=8;
```

```
        pitch=2.pow(n1/(2*6));
        fund=28;

        freq = (fund*6)*pitch;

        bank.do{

angle=(angle+(360-(360/1.6180339887499))).wrap(0,360);
angle1=(angle1+(360-(360/1.6180339887499))).wrap(0,360);

        q=(cos(ang.degrad/2)+(sin(ang.degrad/2)*(i+j+k)));
        q1=(cos(ang.degrad/2)+(sin(ang.degrad/2)*(i1+j1+k1)));

        va=q*va*q.conjugate;

        vb=q1*vb*q1.conjugate;

        cva=Cartesian(va.b,va.c,va.d);
        cvb=Cartesian(vb.b,vb.c,vb.d);

        phx=exp(Complex(0,1)*(angle.degrad)).phase;
        phy=exp(Complex(0,1)*(angle1.degrad)).phase;


env1=EnvGen.ar(Env.perc(attack,decay,pitch.reciprocal),doneAction:2,levelScale:phx.real.
round(0.0001));
env2=EnvGen.ar(Env.perc(attack,decay,pitch.reciprocal),doneAction:2,levelScale:phx.ima
g.round(0.0001));

osc1_x=VBAP.ar(4,SinOsc.ar((freq*m),mul:m.reciprocal,phase:SinOsc.ar(0.1,phx,2pi)),~b.
bufnum,cva.theta.raddeg,cva.phi.raddeg);
osc1_y=VBAP.ar(4,SinOsc.ar((freq*m),mul:m.reciprocal,phase:SinOsc.ar(0.1,phy,2pi)),~b.
bufnum,cvb.theta.raddeg,cvb.phi.raddeg);

m=m+1;

        wrappedOut1.value(out,LPF.ar(LeakDC.ar((osc1_x)*env1),cutt));
        wrappedOut2.value(out, LPF.ar(LeakDC.ar((osc1_y)*env2),cutt));

        //Out.ar(0,LPF.ar(LeakDC.ar((osc1_x)*env),cutt));
        //Out.ar(0, LPF.ar(LeakDC.ar((osc1_y)*env),cutt));

        };
}).add;
);
```

In this example there are several fixed variables as var and controlled variables, arguments, as arg. These form the different calculation components for the SynthDef as an algorithm. For this composition the SynthDef utilizes an additive synthesis for spatial sound. There are two basic waveforms formed, using the SinOsc Unit Generator, across the output channel field, osc1_x and osc1_y, which translates to two oscillators being present, These two always have a phase difference of 90 degrees, regardless of the



Appendix



different phases present in the partial harmonics of the waveforms. This is expressed by phx and phy, and their corresponding functions. SinOsc is iterated over a number of times defined by the variable ‘bank’. In this case a value of 8 harmonics is enough for three SynthDefs in one Server simultaneously over a 1/1 tempo, this value could be up to 32 if one SynthDef is used. In these iterations the different harmonics form, in a sawtooth wave fashion, that is with harmonics 1,2,3,4,5… with their corresponding amplitude compensation and an evolving, modulated phase difference, across the three dimensional sound-field VBAP provides. The phase differences along with the panning locations are defined in two dimensions and three dimensions respectively. In two dimensions the rotations are enabled through the Euler identity function as,

```
angle=0,angle1=90

angle=(angle+(360-(360/1.6180339887499))).wrap(0,360);
angle1=(angle1+(360-(360/1.6180339887499))).wrap(0,360);

phx=exp(Complex(0,1)*(angle.degrad)).phase;
phy=exp(Complex(0,1)*(angle1.degrad)).phase;
```

where ‘angle’ and ‘angle1’ represent two angles of rotation with a phase difference of 90 degrees. In three dimensions the same angle of rotation applies and though the use of a quaternion rotations are calculated. The new points are then passed to cartesian coordinates and then converted in azimuth and elevation values for VBAP. Each time the SynthDef is played, a different starting point of rotation in the 3d space of the quaternion is set, and through iteration every harmonic is panned to a different point in 3d space, formed through rotation along different axis of rotation, on every rotation of the iteration. The angle used in both cases is the so called golden angle.

```
ang=(360-(360/1.6180339887499))
```

This angle has the property that no new point overlaps a previous or next one over space and time, through iteration of the rotations. This essentially means that as iteration tends to infinity, if the rotations happen over the same radius and on the same circle, when in 3d that is, the points off these rotations tend to form a circle whose points are all unique coordinates. This series of values are used in the SynthDef to define unique phase stamps and unique panning locations for each harmonic under the same pattern, that of the golden ratio. The uniqueness of the space and time stamps are very important for this additive synthesis, because they allow a most even distribution, where no simple sine wave overlaps and completely interferes with another across panning location or phase. This method allows the creation of multiple phantom sources across a minimal 3d output system such as the vertices of a tetrahedron. In essence the amount of phantom sources and ambience tends to be directly reciprocal to the amount of speakers used, with a minimum of four, since the sound-scape composed is three dimensional.

Each waveform additive synthesis is based on a fundamental frequency. This frequency is calculated by multiplication using an arbitrary value that essential tunes the overall pitch of the instrument, in this case a value of 28 Hertz is used, and a tone value ‘pitch’ which forms the notes and tonality of the instrument in a 12-tonic scale. Also in this case the fundamental is scaled to a higher frequency by a multiplication with six.

```
pitch=2.pow(n1/(2*6));
        fund=28;
```

```
freq = (fund*6)*pitch;
```

Final components of the synthesis is a low-pass filter for each harmonic and an envelope. The envelope has short attack and long variable decay and it compensates the amplitude of each tone. The output of the SynthDef can run across four channel in two possible ways. One is by setting a constant series of these channels where eg. channel 0 corresponds always to the same coordinate,

```
//Out.ar(0,LPF.ar(LeakDC.ar((osc1_x)*env),cutt));
//Out.ar(0, LPF.ar(LeakDC.ar((osc1_y)*env),cutt));
```

or by randomizing the series of channels [0,1,2,3] each time eg. [1,3,0,2].

```
wrappedOut1.value(out,LPF.ar(LeakDC.ar((osc1_x)*env1),cutt));
wrappedOut2.value(out, LPF.ar(LeakDC.ar((osc1_y)*env2),cutt));
```

Following, after the three SynthDefs have been added, Patterns can control them. Before that an array containing two series of an arbitrary length, which defines the length of the composition in time, has to be calculated and collected. The one series defines the sequence of durations for each tone and the other the series of tones played in a scale of the Scale.derictory.

```
(
var
angle1=0,angle2=0,angle3=0,angle4=0,q1,q2,q3,q4,v1,v2,v3,v4,ph1,ph2,ph3,ph4,r,m=2;
var i = Quaternion(0, (1/3).sqrt, 0, 0),j = Quaternion(0, 0, (1/3).sqrt, 0),k = Quaternion(0, 0,
0, (1/3).sqrt);
var i1=0,i2=0,i3=0,i4=0;

r=[1/2,2/3,3/4].normalizeSum;

v1=Quaternion(0,r.[0].sqrt.neg,r.[1].sqrt,r.[2].sqrt);
v2=Quaternion(0,r.[0].sqrt.neg,r.[1].sqrt,r.[2].sqrt);
v3=Quaternion(0,r.[0].sqrt.neg,r.[1].sqrt,r.[2].sqrt);
v4=Quaternion(0,r.[0].sqrt.neg,r.[1].sqrt,r.[2].sqrt);

~axis=10000.collect{[0.999999.rand,0.999999.rand,0.999999.rand].normalizeSum;};

        ~phase1 = 10000.collect{
                angle1=((360-(360/1.6180339887499)));
                q1=(cos(angle1.degrad/2)+(sin(angle1.degrad/2)*(i+j+k)));
        i=Quaternion(0,~axis.[i1].[0].sqrt,0,0);
        j=Quaternion(0,0,~axis.[i1].[1].sqrt,0);
        k=Quaternion(0,0,0,~axis.[i1].[2].sqrt);
        i1=i1+1;
        v1=q1*v1*q1.conjugate;
        ph1=(v1.a*m).round(0.25);
        };
```

115cm, and have a small crease inward across the periphery of the cycle so that the speaker chassis can be bolted on the fiberglass horn. Fiberglass and resin have been chosen because fiberglass offers a unique and uniform transparency, is strong and light.

The same bolts can be used to hang the speakers from the ceiling with nylon strings, from differing heights. Alternatively the speakers can be laid down on the floor. This gives the impression of a space with scattered translucent bells, the reminiscence of a church tower graveyard of hyper objects.

This sculpture, although simple, it might bring some unwanted effects in the installation. The fact that the back of the speaker chassis is open to the environment is certainly to cause more random refractions and ambience, as sound pressure also propagates from the back of the speaker. To solve this, the speaker sculpture can be further converted to a closed cabinet. This can be achieved by adding a spherical shape out of fiberglass to the form, by having it designed to have a hole as wide as the speaker horn. Then the speaker horn system can be inserted into the sphere and before that, in the spheric cabinet, damming material is to be placed. This construction has three advantages, one is that it kinda looks like a fruit, or a 3D heart shape eg. using the Mandelbrot Set cardioid. Second is that that unwanted sound pressure can be dammed, and thirdly that because of the shape's center of gravity, being the heavier than fiberglass, speaker chassis, the sculpture when placed on the ground naturally rotates and leans to a position where the speaker horn faces upwards. This makes the speakers ideally placeable on the ground in an installation setup and any other placement, like hanging at different heights, would require methods to counter-rotate the sculpture's center of gravity, like strings with weights or static lifting abilities.

Installation

A sound installation can be imagined to exist in a pitch black anechoic chamber. Such an environment would enhance the presence of the sound composition as a thing in its own right, but such an environment would also be highly controlled, unrealistic and unfriendly for an audience. Of course this does not mean that in an exhibition space lights cannot be dimmed and appropriate acoustic enhancement measures cannot be taken. In an outdoor environment the setting changes dramatically, as sound propagates freely in the atmosphere and onto objects and buildings, natural sounds are blended with the installation and light is both dependent from the sun and artificial means. Also temperature is uncontrolled which also affects the air as a medium.

With this in mind the design of the installation that works as the realization and final product, containing the subjects of the previous chapters, tries to be in balance between different settings. Its aesthetic characteristics are an emphasis on the technical means used, electronic, architectural and acoustic. Its presence takes advantage of different lighting and acoustic settings with an emphasis on engaging the viewer in a non dictative way, allowing freedom of movement in both space and time. For the experience of sound this means that the installation can be viewed from any angle or radius from its center, and although its composition has a beginning and an end, the duration is so long that from the viewer's perspective there is no clear time spent being an audience, therefore this time interval becomes a free choice, along with when the installation is to be visited.

Thereof further design features are the choice of cable and its colors as they hang and lay freely in space, the location of the box containing the electronics as well as the choice between AC power socket or power bank, an attachment of fiberglass horn cones on the speaker drivers which amplifies their presence both acoustically and visually, and the way the speakers are set in the space, that is wether on a base, on the floor or hanging. Note that in this case four speakers are used, also if the speakers correspond to the symmetrical nature of tetrahedron vertices, the have to be set on such locations, otherwise a random placement is possible. This can be decided in situ.

Speaker cables should be long enough so that each can cover the needs of the root to the further speaker located to the proximity of the box. This way the cables are normalized to equal and standard length. The root they take has to be able to cover the distance of certain forming. First they have to be able to loosely hang from the box, creating a small spiral in front this hang. Then they have to be laid loosely on the ground without being an obstacle, not making sudden diagonal paths that is, until they reach the proximity of the speaker. Then depending on the remaining length of the cable they make a spiral and then connect to the speaker, either directly if this is on the ground, or forming a hang if the speaker is located further above. Be aware that the copper wire inside the cables is not a rope so it has to also not be treated as such.

The whole system can be powered either to a close to the box socket, or through a power cable extension to a further one. This cable is to have a similar form and dynamic through space as the speaker cables. In case of an outdoor installation, or in the design of an exhibition the use of a battery has to be highlighted or necessary, the use of a power power station, a big battery with an inverter that can provide 230V AC, can be used. These power stations commonly can be powered by solar panels, in case the installation is totally off grid.

The fiber glass horns are to be manufactured as follows. First a mold has to be made, preferably with a 3D foam cutting machine based on a symmetrical drawing of an octagonal Bezier Curve horn. Then the mold is to be made more round with a layer of gypsum. Following the mold can be used to form fiber glass sheets with epoxy resin. The thin part of the horn has to end in a diameter equal to the speaker chassis diameter, argument, durations and tones. Since the algorithm uses a quaternion to calculate the values according to ration in 3d space, the values used are only the three last arrays, corresponding to three axis x, y and z. The first array collected is always 0 because of the unit quaternion. Below you will find the Patterns used, to play them and record them.

```
(

b=Pdef(\2, Pbind(\instrument, \ob,
        \dur,Pseq(~phase2.abs,inf),
\n1,PdegreeToKey(Pseq(~n2,1),Scale.phrygian),
        \cutt,Pbrown(220*2,220*8,100,inf),
                \out,Pn(Pshuf([0, 1, 2, 3], 1), inf).clump(4).collect([_]),
        \x,Pbrown(0,1,0.1,inf),
        \y,Pbrown(0,1,0.1,inf),
        \z,Pbrown(0,1,0.1,inf),
        \rx,Pbrown(0,1,0.1,inf),
        \ry,Pbrown(0,1,0.1,inf),
        \rz,Pbrown(0,1,0.1,inf),
                \rx1,Pbrown(0,1,0.1,inf),
        \ry1,Pbrown(0,1,0.1,inf),
        \rz1,Pbrown(0,1,0.1,inf),
                \ng1,Prand([1.neg,1,1.neg,1],inf),
        \ng2,Prand([1.neg,1,1.neg,1],inf),
        \ng3,Prand([1.neg,1,1.neg,1],inf),
                \ng4,Prand([1.neg,1,1.neg,1],inf),
        \ng5,Prand([1.neg,1,1.neg,1],inf),
        \ng6,Prand([1.neg,1,1.neg,1],inf),
        \decay,Pbrown(2,8,1,inf);
));

c=Pdef(\3, Pbind(\instrument, \oc,
        \dur,Pseq(~phase3.abs,inf),
\n1,PdegreeToKey(Pseq(~n3,1),Scale.phrygian),
        \cutt,Pbrown(220*2,220*8,100,inf),
                \out,Pn(Pshuf([0, 1, 2, 3], 1), inf).clump(4).collect([_]),
        \x,Pbrown(0,1,0.1,inf),
        \y,Pbrown(0,1,0.1,inf),
        \z,Pbrown(0,1,0.1,inf),
        \rx,Pbrown(0,1,0.1,inf),
        \ry,Pbrown(0,1,0.1,inf),
        \rz,Pbrown(0,1,0.1,inf),
                \rx1,Pbrown(0,1,0.1,inf),
        \ry1,Pbrown(0,1,0.1,inf),
        \rz1,Pbrown(0,1,0.1,inf),
                \ng1,Prand([1.neg,1,1.neg,1],inf),
        \ng2,Prand([1.neg,1,1.neg,1],inf),
        \ng3,Prand([1.neg,1,1.neg,1],inf),
                \ng4,Prand([1.neg,1,1.neg,1],inf),
        \ng5,Prand([1.neg,1,1.neg,1],inf),
        \ng6,Prand([1.neg,1,1.neg,1],inf),
        \decay,Pbrown(2,8,1,inf);
```

```
));

d=Pdef(\4, Pbind(\instrument, \od,
        \dur,Pseq(~phase4.abs,inf),
\n1,PdegreeToKey(Pseq(~n4,1),Scale.phrygian),
        \cutt,Pbrown(220*2,220*8,100,inf),
        \out,Pn(Pshuf([0, 1, 2, 3], 1), inf).clump(4).collect([_]),
        \x,Pbrown(0,1,0.1,inf),
        \y,Pbrown(0,1,0.1,inf),
        \z,Pbrown(0,1,0.1,inf),
        \rx,Pbrown(0,1,0.1,inf),
        \ry,Pbrown(0,1,0.1,inf),
        \rz,Pbrown(0,1,0.1,inf),
        \rx1,Pbrown(0,1,0.1,inf),
        \ry1,Pbrown(0,1,0.1,inf),
        \rz1,Pbrown(0,1,0.1,inf),
                \ng1,Prand([1.neg,1,1.neg,1],inf),
        \ng2,Prand([1.neg,1,1.neg,1],inf),
        \ng3,Prand([1.neg,1,1.neg,1],inf),
        \ng4,Prand([1.neg,1,1.neg,1],inf),
        \ng5,Prand([1.neg,1,1.neg,1],inf),
        \ng6,Prand([1.neg,1,1.neg,1],inf),
        \decay,Pbrown(2,8,1,inf);
));
);

(
b.play;
c.play;
d.play;
s.record(numChannels:4)
)
```

This concludes the synthesized sound section of the composition. To further develop the acousmatic space of the composition, recorded sound can be used. After audio files are loaded onto Buffers inside SuperCollider, a SynthDef can take over. These SynthDefs use the same techniques mentioned above for the synthesized sound by running the audio files through an FFT analysis of partials. This way different events and auditory components of eg. a sound walk, can be panned on different coordinates, both for the overall acoustic event and its partials, across a three dimensional field. These SynthDefs form pieces of a recorded audio composition with their corresponding timestamps, and are played once or looped over longer periods of time. The recorded technique required for the composition initial Buffers can be made with a stereo recorded, using later the two stereo channels, as two different locations in the playback three-dimensional field. Alternatively if a mono recording is made, this can be used to generate two Buffers with phase difference of 90 degrees through an FFT phase shift. Below you will find the code for the before mentioned points.

On Electroacoustic Circuit

There are multiple products that can be combined to create a multichannel audio setup and multiple systems. Some examples, are active monitors on an audio interface, PA systems, Bluetooth 5 stereo amplifiers with speakers, Arduino mp3 triggers, WAV trigger DACs, 7.1 sound cards with stereo amplifiers or amplifier stacks. For the purposes of this composition a specific setup is chosen which can support up to 8 output channels in 4 stereo channel format. For this a Raspberry Pi 5 is used along with a Hifiberry DAC8X is used which allows the Raspberry to output four stereo signals up to 192kHz. After the composition audio file is created, wether that is in four channels as in this case or more (eg. 8 for a cube), the file can then be transferred in the Raspberry, which can be then programmed to play and loop the file when booted. In case of using VLC for file playback, the following commands in the Raspberry Pi terminal should enable playback on boot if the file is located in the /home/pi folder.

```
mkdir /home/pi/.config/autostart
nano /home/pi/.config/autostart/autovlc.desktop

[Desktop Entry]
Type=Application
Exec=vlc /home/pi/file.wav
```

This is especially helpful for a standalone sound installation in an exhibition space, or any space, because the audio can be in high quality and be played or stopped by simply plugging in or out the Raspberry from power. This power source can be either a mains socket or a long lasting power bank with an inverter.

For the amplification of the signals two 40 or 30 Watt per channel stereo amplifiers or amplifier boards with their corresponding power supply are used. These amplifiers are connected to the DAC8X via a jack 3.5 connection cable to whatever input the amplifier have, usually that is RCA. Then speaker cable connect to the four speakers which are 8 Ohm, 40 Watt, full-range speakers, specifically the Monacor SPM-116/8 model. Important here is to note that the longer the distance of the cable the thicker it needs to be because of resistance build up which can affect the signal. For the purposes of this installation any speaker cable not too thin should be enough.

For this setup, speaker drivers are chosen and not exciters/ transducers because for the purposes of the composition a clearer and stronger sound is needed with the SPL a speaker driver can provide. Although exciters can be very useful when trying to design versatile setups for sound installations, sculptural speakers and experiment with the resonance os different materials, they cannot provide the sound pressure and fidelity needed for the purposes of this composition and installation. Imagery concerning the audio setup and examples of the use of exciters/transducers in artworks can be found in the appendix. All electronics can be enclosed in a box, preferably water-tight to enable outdoor use. The box can have one input being the AC power cable and four outputs for the speaker cables and speakers. These latter connections can be made with PA speaker twist plugs and sockets, for ease of installation and outdoor use.

```
	\ng5,Prand([1.neg,1,1.neg,1],inf),

\ng6,Prand([1.neg,1,1.neg,1],inf),
	\decay,Pbrown(2,8,1,inf);
));

d=Pdef(\4, Pbind(\instrument, \o3,

	\dur,Pseq(~phase2.abs,inf),
\n,PdegreeToKey(Pseq(~n2,1),Scale.phrygian),
	\x,Pbrown(0,1,0.1,inf),
	\y,Pbrown(0,1,0.1,inf),
	\z,Pbrown(0,1,0.1,inf),
		\ng1,Prand([1.neg,1,1.neg,1],inf),
	\ng2,Prand([1.neg,1,1.neg,1],inf),
	\ng3,Prand([1.neg,1,1.neg,1],inf),
			\ng4,Prand([1.neg,1,1.neg,1],inf),
	\ng5,Prand([1.neg,1,1.neg,1],inf),
	\ng6,Prand([1.neg,1,1.neg,1],inf),
	\decay,Pbrown(2,8,1,inf);
));
);

(
b.play;
c.play;
d.play;
s.record(numChannels:4)
)
```

If PitchShift is not the tool of choice, a band-pass filter could prove as useful.

This concludes the composition and the steps necessary to produce it in SuperCollider, either as real-time audio or for a quadraphonic sound file. This file could be a mix of the three recorded parts of the composition, fading in, fading out and looping with the use of sine envelopes. This composition apart from being spatial and ambient in 3d, aims to juxtapose synthesized sound with recorded, natural sound, in a common sound-field of long duration. Abstractions and realisms are synthesized together in a musical dramaturgy of time, redefining themselves, and offering an audio virtuality of thesis, antithesis and synthesis. Out of the three parts of the composition none have to be used altogether, but also only one or two. That is because, depending on where the composition is installed and what the final result needs to be, not all might be necessary. For example if we are talking about an outdoor installation with inherit natural sounds that blend in with the composition, only the first part, the synthesized sounds, might be necessary. The composition leans on a scenographic, space oriented and landscape portrayal oriented intend with open suggestion on what happens in its grounds. The story told is that of symmetry, randomness, proportion, reciprocality, difference, repetition, time and space.

argument, durations and tones. Since the algorithm uses a quaternion to calculate the values according to ration in 3d space, the values used are only the three last arrays, corresponding to three axis x, y and z. The first array collected is always 0 because of the unit quaternion. Below you will find the Patterns used, to play them and record them.

```
(

b=Pdef(\2, Pbind(\instrument, \ob,
	\dur,Pseq(~phase2.abs,inf),
\n1,PdegreeToKey(Pseq(~n2,1),Scale.phrygian),
	\cutt,Pbrown(220*2,220*8,100,inf),
		\out,Pn(Pshuf([0, 1, 2, 3], 1), inf).clump(4).collect([_]),
	\x,Pbrown(0,1,0.1,inf),
	\y,Pbrown(0,1,0.1,inf),
	\z,Pbrown(0,1,0.1,inf),
	\rx,Pbrown(0,1,0.1,inf),
	\ry,Pbrown(0,1,0.1,inf),
	\rz,Pbrown(0,1,0.1,inf),
		\rx1,Pbrown(0,1,0.1,inf),
	\ry1,Pbrown(0,1,0.1,inf),
	\rz1,Pbrown(0,1,0.1,inf),
		\ng1,Prand([1.neg,1,1.neg,1],inf),
	\ng2,Prand([1.neg,1,1.neg,1],inf),
	\ng3,Prand([1.neg,1,1.neg,1],inf),
		\ng4,Prand([1.neg,1,1.neg,1],inf),
	\ng5,Prand([1.neg,1,1.neg,1],inf),
	\ng6,Prand([1.neg,1,1.neg,1],inf),
	\decay,Pbrown(2,8,1,inf);
));

c=Pdef(\3, Pbind(\instrument, \oc,
	\dur,Pseq(~phase3.abs,inf),
\n1,PdegreeToKey(Pseq(~n3,1),Scale.phrygian),
	\cutt,Pbrown(220*2,220*8,100,inf),
		\out,Pn(Pshuf([0, 1, 2, 3], 1), inf).clump(4).collect([_]),
	\x,Pbrown(0,1,0.1,inf),
	\y,Pbrown(0,1,0.1,inf),
	\z,Pbrown(0,1,0.1,inf),
	\rx,Pbrown(0,1,0.1,inf),
	\ry,Pbrown(0,1,0.1,inf),
	\rz,Pbrown(0,1,0.1,inf),
		\rx1,Pbrown(0,1,0.1,inf),
	\ry1,Pbrown(0,1,0.1,inf),
	\rz1,Pbrown(0,1,0.1,inf),
		\ng1,Prand([1.neg,1,1.neg,1],inf),
	\ng2,Prand([1.neg,1,1.neg,1],inf),
	\ng3,Prand([1.neg,1,1.neg,1],inf),
		\ng4,Prand([1.neg,1,1.neg,1],inf),
	\ng5,Prand([1.neg,1,1.neg,1],inf),
	\ng6,Prand([1.neg,1,1.neg,1],inf),
	\decay,Pbrown(2,8,1,inf);
```

```
));

d=Pdef(\4, Pbind(\instrument, \od,
        \dur,Pseq(~phase4.abs,inf),
\n1,PdegreeToKey(Pseq(~n4,1),Scale.phrygian),
        \cutt,Pbrown(220*2,220*8,100,inf),
        \out,Pn(Pshuf([0, 1, 2, 3], 1), inf).clump(4).collect([_]),
        \x,Pbrown(0,1,0.1,inf),
        \y,Pbrown(0,1,0.1,inf),
        \z,Pbrown(0,1,0.1,inf),
        \rx,Pbrown(0,1,0.1,inf),
        \ry,Pbrown(0,1,0.1,inf),
        \rz,Pbrown(0,1,0.1,inf),
        \rx1,Pbrown(0,1,0.1,inf),
        \ry1,Pbrown(0,1,0.1,inf),
        \rz1,Pbrown(0,1,0.1,inf),
                \ng1,Prand([1.neg,1,1.neg,1],inf),
        \ng2,Prand([1.neg,1,1.neg,1],inf),
        \ng3,Prand([1.neg,1,1.neg,1],inf),
        \ng4,Prand([1.neg,1,1.neg,1],inf),
        \ng5,Prand([1.neg,1,1.neg,1],inf),
        \ng6,Prand([1.neg,1,1.neg,1],inf),
        \decay,Pbrown(2,8,1,inf);
));
);

(
b.play;
c.play;
d.play;
s.record(numChannels:4)
)
```

This concludes the synthesized sound section of the composition. To further develop the acousmatic space of the composition, recorded sound can be used. After audio files are loaded onto Buffers inside SuperCollider, a SynthDef can take over. These SynthDefs use the same techniques mentioned above for the synthesized sound by running the audio files through an FFT analysis of partials. This way different events and auditory components of eg. a sound walk, can be panned on different coordinates, both for the overall acoustic event and its partials, across a three dimensional field. These SynthDefs form pieces of a recorded audio composition with their corresponding timestamps, and are played once or looped over longer periods of time. The recorded technique required for the composition initial Buffers can be made with a stereo recorded, using later the two stereo channels, as two different locations in the playback three-dimensional field. Alternatively if a mono recording is made, this can be used to generate two Buffers with phase difference of 90 degrees through an FFT phase shift. Below you will find the code for the before mentioned points.

```
~n3 = 10000.collect{

        angle3=((360-(360/1.6180339887499)));

                q3=(cos(angle3.degrad/2)+(sin(angle3.degrad/2)*(i+j+k)));
                i=Quaternion(0,~axis.[i3].[0].sqrt,0,0);
        j=Quaternion(0,0,~axis.[i3].[1].sqrt,0);
        k=Quaternion(0,0,0,~axis.[i3].[2].sqrt);
        i3=i3+1;
        v3=q3*v3*q3.conjugate;
        ph3=(v3.c*m).round(1);
        };

~n4 = 10000.collect{
                angle4=((360-(360/1.6180339887499)));
                q4=(cos(angle4.degrad/2)+(sin(angle4.degrad/2)*(i+j+k)));
                i=Quaternion(0,~axis.[i4].[0].sqrt,0,0);
        j=Quaternion(0,0,~axis.[i4].[1].sqrt,0);
        k=Quaternion(0,0,0,~axis.[i4].[2].sqrt);
        i4=i4+1;
        v4=q4*v4*q4.conjugate;
        ph4=(v4.d*m).round(1);

        };
);

(
b=Pdef(\2, Pbind(\instrument, \o1,
        \dur,Pseq(~phase2.abs,inf),
\n,PdegreeToKey(Pseq(~n2,1),Scale.phrygian),
        \x,Pbrown(0,1,0.1,inf),
        \y,Pbrown(0,1,0.1,inf),
        \z,Pbrown(0,1,0.1,inf),
                \ng1,Prand([1.neg,1,1.neg,1],inf),
        \ng2,Prand([1.neg,1,1.neg,1],inf),
        \ng3,Prand([1.neg,1,1.neg,1],inf),
                        \ng4,Prand([1.neg,1,1.neg,1],inf),
        \ng5,Prand([1.neg,1,1.neg,1],inf),
        \ng6,Prand([1.neg,1,1.neg,1],inf),
        \decay,Pbrown(2,8,1,inf);
));
c=Pdef(\3, Pbind(\instrument, \o2,
        \dur,Pseq(~phase2.abs,inf),
\n,PdegreeToKey(Pseq(~n2,1),Scale.phrygian),
        \x,Pbrown(0,1,0.1,inf),
        \y,Pbrown(0,1,0.1,inf),
        \z,Pbrown(0,1,0.1,inf),
                \ng1,Prand([1.neg,1,1.neg,1],inf),
        \ng2,Prand([1.neg,1,1.neg,1],inf),
        \ng3,Prand([1.neg,1,1.neg,1],inf),
                        \ng4,Prand([1.neg,1,1.neg,1],inf),
```

```
~phase4 = 10000.collect{

	angle4=((360-(360/1.6180339887499)));

		q4=(cos(angle4.degrad/2)+(sin(angle4.degrad/2)*(i+j+k)));
	i=Quaternion(0,~axis.[i4].[0].sqrt,0,0);
	j=Quaternion(0,0,~axis.[i4].[1].sqrt,0);
	k=Quaternion(0,0,0,~axis.[i4].[2].sqrt);
	i4=i4+1;
	v4=q4*v4*q4.conjugate;
	ph4=(v4.d*m).round(0.25);
	};

);

(
var
angle1=0,angle2=0,angle3=0,angle4=0,q1,q2,q3,q4,v1,v2,v3,v4,ph1,ph2,ph3,ph4,r,m=12;
var i = Quaternion(0, (1/3).sqrt, 0, 0),j = Quaternion(0, 0, (1/3).sqrt, 0),k = Quaternion(0, 0,
0, (1/3).sqrt),n1,n2,n3,n4;
var i1=0,i2=0,i3=0,i4=0;

r=[1/2,2/3,3/4].normalizeSum;

v1=Quaternion(0,r.[0].sqrt.neg,r.[1].sqrt,r.[2].sqrt);
v2=Quaternion(0,r.[0].sqrt.neg,r.[1].sqrt,r.[2].sqrt);
v3=Quaternion(0,r.[0].sqrt.neg,r.[1].sqrt,r.[2].sqrt);
v4=Quaternion(0,r.[0].sqrt.neg,r.[1].sqrt,r.[2].sqrt);

~axis=10000.collect{[0.999999.rand,0.999999.rand,0.999999.rand].normalizeSum;};

	~n1 = 10000.collect{
		angle1=((360-(360/1.6180339887499)));
		q1=(cos(angle1.degrad/2)+(sin(angle1.degrad/2)*(i+j+k)));
		i=Quaternion(0,~axis.[i1].[0].sqrt,0,0);
	j=Quaternion(0,0,~axis.[i1].[1].sqrt,0);
	k=Quaternion(0,0,0,~axis.[i1].[2].sqrt);
	i1=i1+1;
	v1=q1*v1*q1.conjugate;
	ph1=(v1.a*m).round(1);
	};
~n2 = 10000.collect{
		angle2=((360-(360/1.6180339887499)));
		q2=(cos(angle2.degrad/2)+(sin(angle2.degrad/2)*(i+j+k)));
		i=Quaternion(0,~axis.[i2].[0].sqrt,0,0);
	j=Quaternion(0,0,~axis.[i2].[1].sqrt,0);
	k=Quaternion(0,0,0,~axis.[i2].[2].sqrt);
	i2=i2+1;
	v2=q2*v2*q2.conjugate;
	ph2=(v2.b*m).round(1);
	};
(
// 3D
~a = VBAPSpeakerArray.new(3, [[0, 45], [180, 45], [90, 45.neg], [90.neg, 45.neg]]);
~b = Buffer.loadCollection(s, ~a.getSetsAndMatrices);
)
(
// 3D random
~a = VBAPSpeakerArray.new(3, [[180.rand2, 90.rand2], [180.rand2, 90.rand2], [180.rand2,
90.rand2], [180.rand2, 90.rand2]]);
~b = Buffer.loadCollection(s, ~a.getSetsAndMatrices);
)

(
~c1 = Buffer.readChannel(s,"file",channels:0 );
~c2 = Buffer.readChannel(s,"file",channels:1 );
~c3 = Buffer.readChannel(s,"file",channels:0 );
~c4 = Buffer.readChannel(s,"file",channels:1 );
);

(
(
SynthDef.new(\o1, {
   var in1,in2, chainx,chainy,b=0,angle,r1,r2,i,j,k,i1,j1,k1;
	var v,q,q1,v1,vp,chainxx,chainyy,cva,cvb;

    in1 = PlayBuf.ar(1, ~c1.bufnum, BufRateScale.kr(~c1), loop: 0, doneAction:2);
	 in2 = PlayBuf.ar(1, ~c2.bufnum, BufRateScale.kr(~c2), loop: 0, doneAction:2);

	vp=[0.999999.rand,0.999999.rand,0.999999.rand].normalizeSum;
	v=Quaternion(0,vp.[0].sqrt.neg,vp.[1].sqrt,vp.[2].sqrt);
	v1=Quaternion(0,vp.[0].sqrt.neg,vp.[1].sqrt,vp.[2].sqrt);

	angle=((360-(360/1.6180339887499)));

	chainx = FFT(LocalBuf(1024), in1);
	chainy = FFT(LocalBuf(1024), in2);

  250.do{

	r1=[0.999999.rand,0.999999.rand,0.999999.rand].normalizeSum;
	r2=[0.999999.rand,0.999999.rand,0.999999.rand].normalizeSum;

	i = Quaternion(0, r1.[0].sqrt, 0, 0);
	j = Quaternion(0, 0,r1.[1].sqrt, 0);
	k = Quaternion(0, 0, 0, r1.[2].sqrt);
	i1 = Quaternion(0, r2.[0].sqrt, 0, 0);
	j1 = Quaternion(0, 0,r2.[1].sqrt, 0);
	k1 = Quaternion(0, 0, 0, r2.[2].sqrt);

	q=(cos(angle.degrad/2)+(sin(angle.degrad/2)*(i+j+k)));
	q1=(cos(angle.degrad/2)+(sin(angle.degrad/2)*(i1+j1+k1)));
```

```
v=q*v*q.conjugate;
        v1=q1*v1*q1.conjugate;

        cva=Cartesian(v.b,v.c,v.d);
        cvb=Cartesian(v.b,v.c,v.d);

   chainxx = chainx.pvcollect(1024, {|mag, phase, index| [mag, phase]; }, frombin: b, tobin:
b, zeroothers: 1);
   chainyy = chainy.pvcollect(1024, {|mag, phase, index| [mag, phase]; }, frombin: b, tobin:
b, zeroothers: 1);

        b=b+1;


        Out.ar(0,VBAP.ar(4,IFFT(chainxx),~b.bufnum,cva.theta.raddeg,cva.phi.raddeg));

        Out.ar(0,VBAP.ar(4,IFFT(chainyy),~b.bufnum,cvb.theta.raddeg,cvb.phi.raddeg));


        }
}).add;
);

(
SynthDef.new(\o2, {
    var in1,in2, chainx,chainy,b=0,angle,r1,r2,i,j,k,i1,j1,k1;
        var v,q,q1,v1,vp,chainxx,chainyy,cva,cvb;

    in1 = PlayBuf.ar(1, ~c3.bufnum, BufRateScale.kr(~c3), loop: 0, doneAction:2);
         in2 = PlayBuf.ar(1, ~c4.bufnum, BufRateScale.kr(~c4), loop: 0, doneAction:2);

        vp=[0.999999.rand,0.999999.rand,0.999999.rand].normalizeSum;
        v=Quaternion(0,vp.[0].sqrt.neg,vp.[1].sqrt,vp.[2].sqrt);
        v1=Quaternion(0,vp.[0].sqrt.neg,vp.[1].sqrt,vp.[2].sqrt);

        angle=((360-(360/1.6180339887499)));

        chainx = FFT(LocalBuf(1024), in1);
        chainy = FFT(LocalBuf(1024), in2);

  250.do{

        r1=[0.999999.rand,0.999999.rand,0.999999.rand].normalizeSum;
        r2=[0.999999.rand,0.999999.rand,0.999999.rand].normalizeSum;

        i = Quaternion(0, r1.[0].sqrt, 0, 0);
        j = Quaternion(0, 0,r1.[1].sqrt, 0);
        k = Quaternion(0, 0, 0, r1.[2].sqrt);
        i1 = Quaternion(0, r2.[0].sqrt, 0, 0);
        j1 = Quaternion(0, 0,r2.[1].sqrt, 0);
        k1 = Quaternion(0, 0, 0, r2.[2].sqrt);
}).add;
);

(
var
angle1=0,angle2=0,angle3=0,angle4=0,q1,q2,q3,q4,v1,v2,v3,v4,ph1,ph2,ph3,ph4,r,m=2;
var i = Quaternion(0, (1/3).sqrt, 0, 0),j = Quaternion(0, 0, (1/3).sqrt, 0),k = Quaternion(0, 0,
0, (1/3).sqrt);
var i1=0,i2=0,i3=0,i4=0;

r=[1/2,2/3,3/4].normalizeSum;

v1=Quaternion(0,r.[0].sqrt.neg,r.[1].sqrt,r.[2].sqrt);
v2=Quaternion(0,r.[0].sqrt.neg,r.[1].sqrt,r.[2].sqrt);
v3=Quaternion(0,r.[0].sqrt.neg,r.[1].sqrt,r.[2].sqrt);
v4=Quaternion(0,r.[0].sqrt.neg,r.[1].sqrt,r.[2].sqrt);

~axis=10000.collect{[0.999999.rand,0.999999.rand,0.999999.rand].normalizeSum;};

        ~phase1 = 10000.collect{
                angle1=((360-(360/1.6180339887499)));
                q1=(cos(angle1.degrad/2)+(sin(angle1.degrad/2)*(i+j+k)));
        i=Quaternion(0,~axis.[i1].[0].sqrt,0,0);
        j=Quaternion(0,0,~axis.[i1].[1].sqrt,0);
        k=Quaternion(0,0,0,~axis.[i1].[2].sqrt);
        i1=i1+1;
        v1=q1*v1*q1.conjugate;
        ph1=(v1.a*m).round(0.25);
        };

~phase2 = 10000.collect{
                angle2=((360-(360/1.6180339887499)));
                q2=(cos(angle2.degrad/2)+(sin(angle2.degrad/2)*(i+j+k)));
        i=Quaternion(0,~axis.[i2].[0].sqrt,0,0);
        j=Quaternion(0,0,~axis.[i2].[1].sqrt,0);
        k=Quaternion(0,0,0,~axis.[i2].[2].sqrt);
        i2=i2+1;
        v2=q2*v2*q2.conjugate;
        ph2=(v2.b*m).round(0.25);
        };

~phase3 = 10000.collect{
                angle3=((360-(360/1.6180339887499)));
                q3=(cos(angle3.degrad/2)+(sin(angle3.degrad/2)*(i+j+k)));
        i=Quaternion(0,~axis.[i3].[0].sqrt,0,0);
        j=Quaternion(0,0,~axis.[i3].[1].sqrt,0);
        k=Quaternion(0,0,0,~axis.[i3].[2].sqrt);
        i3=i3+1;
        v3=q3*v3*q3.conjugate;
        ph3=(v3.c*m).round(0.25);
};
```

```
		pitch=2.pow(n/12);

		env=EnvGen.ar(Env.perc(attack,decay,pitch.reciprocal),doneAction:2);



		in1 = PitchShift.ar(PlayBuf.ar(1, ~c1.bufnum, BufRateScale.kr(~c1), loop: 0,
doneAction:2),pitchRatio:pitch,mull:env);
		in2 = PitchShift.ar(PlayBuf.ar(1, ~c2.bufnum, BufRateScale.kr(~c2), loop: 0,
doneAction:2),pitchRatio:pitch,mull:env);

		vp=[x,y,z].normalizeSum;
		v=Quaternion(0,vp.[0].sqrt*ng1,vp.[1].sqrt*ng2,vp.[2].sqrt*ng3);
		v1=Quaternion(0,vp.[0].sqrt*ng4,vp.[1].sqrt*ng5,vp.[2].sqrt*ng6);

		angle=((360-(360/1.6180339887499)));

		chainx = FFT(LocalBuf(1024), in1);
		chainy = FFT(LocalBuf(1024), in2);

	250.do{

		r1=[0.999999.rand,0.999999.rand,0.999999.rand].normalizeSum;
		r2=[0.999999.rand,0.999999.rand,0.999999.rand].normalizeSum;

		i = Quaternion(0, r1.[0].sqrt, 0, 0);
		j = Quaternion(0, 0,r1.[1].sqrt, 0);
		k = Quaternion(0, 0, 0, r1.[2].sqrt);
		i1 = Quaternion(0, r2.[0].sqrt, 0, 0);
		j1 = Quaternion(0, 0,r2.[1].sqrt, 0);
		k1 = Quaternion(0, 0, 0, r2.[2].sqrt);

		q=(cos(angle.degrad/2)+(sin(angle.degrad/2)*(i+j+k)));
		q1=(cos(angle.degrad/2)+(sin(angle.degrad/2)*(i1+j1+k1)));
		v=q*v*q.conjugate;
		v1=q1*v1*q1.conjugate;

		cva=Cartesian(v.b,v.c,v.d);
		cvb=Cartesian(v1.b,v1.c,v1.d);

	chainxx = chainx.pvcollect(1024, {|mag, phase, index| [mag, phase]; }, frombin: b, tobin:
b, zeroothers: 1);
	chainyy = chainy.pvcollect(1024, {|mag, phase, index| [mag, phase]; }, frombin: b, tobin:
b, zeroothers: 1);

		b=b+1;

Out.ar(0,VBAP.ar(4,IFFT(chainxx),~b.bufnum,cva.theta.raddeg,cva.phi.raddeg)*env);
Out.ar(0,VBAP.ar(4,IFFT(chainyy),~b.bufnum,cvb.theta.raddeg,cvb.phi.raddeg)*env);


		}
```

```
		q=(cos(angle.degrad/2)+(sin(angle.degrad/2)*(i+j+k)));
		q1=(cos(angle.degrad/2)+(sin(angle.degrad/2)*(i1+j1+k1)));
		v=q*v*q.conjugate;
		v1=q1*v1*q1.conjugate;

		cva=Cartesian(v.b,v.c,v.d);
		cvb=Cartesian(v1.b,v1.c,v1.d);

	chainxx = chainx.pvcollect(1024, {|mag, phase, index| [mag, phase]; }, frombin: b, tobin:
b, zeroothers: 1);
	chainyy = chainy.pvcollect(1024, {|mag, phase, index| [mag, phase]; }, frombin: b, tobin:
b, zeroothers: 1);

		b=b+1;


		Out.ar(0,VBAP.ar(4,IFFT(chainxx),~b.bufnum,cva.theta.raddeg,cva.phi.raddeg));

		Out.ar(0,VBAP.ar(4,IFFT(chainyy),~b.bufnum,cvb.theta.raddeg,cvb.phi.raddeg));


		}
}).add;
);
);

(
Synth(\o1);
s.record(numChannels:4)
Synth(\o2);
s.record(numChannels:4)
)
```

An FFT can also be used to change the pitch of a Buffer, the playback rate, or freeze a time stamp of audio in time to create a periodic signal. The latter method can yield a third part of the composition, a synthesis, where abstracted sounds from recordings can be pitched according to a series of tones and durations and panned in partials across the 3d soundscape. An example on how to derive these periodic abstractions can be found below.

```
~c1 = Buffer.readChannel(s,"file",channels:[0,1] );

(
~fftsize=8192;
~hop=0.25;
~win=0;
~f={Buffer.alloc(s,~c1.duration.calcPVRecSize(~fftsize,~hop))}!2;
);
```

```
(
{
	var sig,chain,localbuf;
	sig=PlayBuf.ar(2, ~c1.bufnum, BufRateScale.kr(~c1), loop: 0,doneAction:2);
	localbuf={LocalBuf.new(~fftsize)}!2;
	chain=FFT(localbuf,sig,~hop,~win);
	chain=PV_RecordBuf(chain,~f,run:1,hop:~hop,wintype:~win);
	0;
}.play;
)

(
x={
	var sig,chain,localbuf;
	localbuf={LocalBuf.new(~fftsize)}!2;
		chain=PV_PlayBuf(localbuf,~f,\rate.kr(1),loop:inf);
		sig=IFFT(chain,~win);
	}.play;
);

x.set(\rate,0);
x.set(\rate,1/1);
x.set(\rate,-2/3);
x.release(2);
```

Then this third composition part can be produced with the following code.

```
(
// 3D
~a = VBAPSpeakerArray.new(3, [[0, 45], [180, 45], [90, 45.neg], [90.neg, 45.neg]]);
~b = Buffer.loadCollection(s, ~a.getSetsAndMatrices);
)

(
// 3D random
~a = VBAPSpeakerArray.new(3, [[180.rand2, 90.rand2], [180.rand2, 90.rand2], [180.rand2,
90.rand2], [180.rand2, 90.rand2]]);
~b = Buffer.loadCollection(s, ~a.getSetsAndMatrices);
)

(
~c1 = Buffer.readChannel(s,"file",channels:0 );
~c2 = Buffer.readChannel(s,"file",channels:1 );
);

(
SynthDef.new(\o1, {
	arg n,decay,ng1=1,ng2=1,ng3=1,ng4=1,ng5=1,ng6=1;
   var in1,in2, chainx,chainy,b=0,angle,r1,r2,i,j,k,i1,j1,k1;
	var v,q,q1,v1,vp,chainxx,chainyy,cva,cvb,pitch;
```

Note: This composition and installation described aims at a real listener in a real space. In case of a virtual listener, eg. a stereo one-dimensional playback of the soundscape, the acousmatic space of the composition has to be seen as a virtual space with a virtual listener algorithm, with its interaction movements and echolocation acounted for the soundscape, azimuth as pan position, amplitude, reverb and echo compensation acoording to distance from virtual source and phase shift of sources as elevation. An example with tetrahedral space can be found below.

```
(
~c1 = Buffer.readChannel(s,"file",channels:0 );
~c2 = Buffer.readChannel(s,"file",channels:1 );
~c3 = Buffer.readChannel(s,"file",channels:2 );
~c4 = Buffer.readChannel(s,"file",channels:3 );
);
(
(
a={
		arg posx,posy,ampx,ampy,echox,echoy;
		var ch1,ch2,ch3,ch4,c,angle;

		ch1=FreeVerb.ar(PlayBuf.ar(1, ~c1.bufnum, BufRateScale.kr(~c1), loop: 0, doneAction:2));
		ch2=FreeVerb.ar(PlayBuf.ar(1, ~c2.bufnum, BufRateScale.kr(~c2), loop: 0, doneAction:2));
		ch3=FreeVerb.ar(PlayBuf.ar(1, ~c3.bufnum, BufRateScale.kr(~c3), loop: 0, doneAction:2));
		ch4=FreeVerb.ar(PlayBuf.ar(1, ~c4.bufnum, BufRateScale.kr(~c4), loop: 0, doneAction:2));

		angle=Complex(posx,posy).theta;
		c=exp(Complex(0,1)*angle);

		Out.ar(0,Pan2.ar(ch1,c.imag.neg.round(0.0001),ampy));
		Out.ar(0,Pan2.ar(ch2,c.imag.round(0.0001),ampy));
		Out.ar(0,Pan2.ar(ch3,c.real.round(0.0001).poll,ampx));
		Out.ar(0,Pan2.ar(ch4,c.real.neg.round(0.0001),ampx));
}.play;
);
Window.closeAll;
w = Window("Nest", Rect(100,10,300,300))
.front
.alwaysOnTop_(true);

Slider2D(w, Rect(10,80,200,200)).background_(Color.rand).x_(0).y_(0).action_({
				arg obj;
				var posx, posy, pl,echox, echoy, distx, disty;
		posy = obj.y.linlin(0,1,1.neg,1).round(0.0001);
		posx = obj.x.linlin(0,1,1.neg,1).round(0.0001);
		echox= obj.x.linlin(0,1,1.neg,1).abs.reciprocal.wrap(0,1).round(0.0001);
		echoy= obj.y.linlin(0,1,1.neg,1).abs.reciprocal.wrap(0,1).round(0.0001);
		distx= obj.x.linlin(0,1,1.neg,1).abs.linexp(0,1,0.01,1).round(0.0001);
		disty= obj.y.linlin(0,1,1.neg,1).abs.linexp(0,1,0.01,1).round(0.0001);
				pl = [posx, posy];
				pl.postln;
		a.set(\posx,posx,\posy,posy,\ampx,distx,\ampy,disty,\echox,echox,\echoy,echoy);
});
)
```

Alternatively, a binaural recording of the installation can be made to produce the virtual listener sound.
End of note