Open-Source 3D Audio Sequencer for Sound and Musical Recognition in MaxMSP using HOA Library and Jitter

Justin Kerobo
School of Computer Science and Music
Earlham College
Richmond, Indiana 47374
Email: jakerobo15@earlham.edu

***Abstract*—A musician, artist, composer, or even scientist has many needs when it comes to the sound space, which is one of the first dimensions of the contemporary musical and sonal thought. There are many ways to affect the virtual sonal environment with hardware and software. Of the various tools in the electroacoustic domain, this focuses on spatialization. Using this, an artist can create new music, a composer can change sound directly, and a musician can edit their sound in real-time. HOA Library is one of the tools that has allowed for musicians and composers to synthesize, transform and render sound spaces in a creative way. However, due to conventional means of spatialization, HOA Library is widely overlooked. Therefore, I developed "SonoSpatial Walk," a 3D audio sequencer in Max that uses MIDI, Jitter, and HOA Library. It is dependent on nothing but open-source, cross-platform libraries. I use HOA Library for the sound spatialization by getting the values of the environment from the Jitter and OpenGL matrix that create the objects, space, and scene by values. The program also allows for the saving of sounds and objects in the space, along with reverberation values of the room that are editable due to HOA Library. A user can control the room reverberation, enable keyboard control, full-screen, non-movement commands, and navigation commands in the sound space with MIDI control. Finally, I add an interface by using Java which allowed for the traversal of the sound space with a keyboard and mouse to be able to trigger objects.**

***Keywords*—MIDI, OpenGL, Graphics, MaxMSP, HOA Library, Music Technology, Jitter, Computer Music.**

I. INTRODUCTION

The idea of a 3D audio sequencer in Max in a interactive virtual environment made Jitter and OpenGL for sound manipulation is nothing new, but rare. In addition, it relies on the idea of a spatial processor or spatializer. Veritably, as early as the launch of Spat [4] in 1995, the solution has been present, but the libraries are not open-source which makes it difficult to access. Regardless, the area of audio sequence can be useful for computer music research and still can be developed. While there have been many examples using other graphics engines and spatialization toolkits, there has yet to be a 3D audio sequencer that is powered by a spatial processor powerful enough to allow for rendering sound fields, and available as an open-source library. HOA Library, which launched in 2012, is a spatial processor that addresses this problem, with the CICM at the University of Paris 8[1] community making open-source development for spatialization possible. The spatialization techniques of HOA Library, in particular, provide a direct way to connect applications for the 3D audio sequencer to function. Additionally, with a later update, I believe the software, HOA Library (v2.2), paves the way forward for development into this area of research. HOA Library (v.2.2) brings spatial processing power through Ambisonics that had previously been not available as an open-source library. It allows for improved spatialization, within a space, virtual or physical, and is easily accessible.

HOA Library[2] is available for three languages/mediums; MaxMSP[3], PureData[4], and VST[5]. With PureData, it is open-source and has a broad user base. The documentation in PureData is difficult to understand for the beginner computer music developer.[6] With VST, it is available in a lot of Digital Audio Workstations (DAW), but implementation within that environment would not allow visualization, and therefore would not be a 3D audio sequencer. With MaxMSP, video tutorials[7] exist that ease the learning curve, which is why it was chosen.

Despite the existence of these resources, however, the learning curve is still often an issue for new developers.[8] In addition, the rarity of 3D audio sequencers and the lack of open-source offerings make the computer music research in this area quite limited, and could also be improved upon by this application. Therefore, in the next section, I will highlight and compare two 3D audio sequencers made in Max, and detail the benefits and drawbacks of each system. Afterward, I will propose and create a new 3D audio sequencer in Max using HOA Library (v2.2) for sound spatialization by getting the values of the environment from the Jitter and OpenGL matrix that create the objects, space, and scene by values. Furthermore, I use MIDI and Java to control reverberation, enable keyboard control, full-screen, non-movement commands, and navigation commands. Finally, I test the performance of the system by measuring the sound spatialization to get the finite impulse

[1]http://cicm.mshparisnord.org/

[2]http://hoalibrary.mshparisnord.fr/en

[3]https://cycling74.com/products/max/

[4]http://puredata.info

[5]Virtual Studio Technology

[6]http://puredata.info/docs/manuals/pd/x2.htm

[7]https://cycling74.com/tutorials/page/1

[8]http://www.paulschuette.com/wp-content/uploads/2013/01/DEMYSTIFYING-MAXMSP.pdf

response (FIR) frequency response by using the buffir∼ object in Max to convolve an input signal with samples from an input buffer.

II. Previous Research

A. Other Software

In 2014, Vic Hug used Jitter and OpenGL to create SoundStroll[9], a virtual landscape in Max while making IRCAM's Spat[10] and HOA Library (v2.0) the spatial driver. [3] The Spat implementation works, but the HOA Library (v2.0) implementation is "half broken or badly programmed," and as a result is not usable. [3] Additional control of the 3D audio sequencer comes from Java to map keys to control inside the world has been implemented as well. The loading and saving of sounds are also possible by the saving text and Jitter files. While object generation is created using Jitter, Spat is used within the world to control how the sounds are heard within that space, which it does by binaural panning. [5] While Spat is not open-source, SoundStroll is. Spat implementation will likely be outside the scope of the research. However, Spat has a model for binaural panning that is worth emulating and will add a sense of realism when rebuilt using HOA Library (v2.2) and what it provides for the 3D audio sequencer. However, since SoundStroll's creation in 2014, it has not been updated, and the creator wants to believe that "it is usable by someone else than me at this point!" [3]

In 2016, Timo Hoogland created Soularis[11], a 3D audio sequencer in Max with OpenGL intending to "create soundscapes in a live performing situation." [2] It is theme based on the solar system, and allows for the creation of planets in which make up the melodies, rhythms, and chords. [2] Concerning the theme, the center of the solar system is the sun. It works by giving a pulse on a "certain time interval" [2], which then plays the planet in a 3D environment. The spatial driver is something that Hoogland created himself at the time. The system "pans" the sound based on the angle to the user's position/view-direction (time-difference stereophonic), adjusts the volume based on the distance and lowpass-filtering based on the sound being in front or behind the user's view (to simulate the filtering of the head/ear). Spatialization created for reproducing and controlling the localization of sound sources, and the projection of that sound in real or virtual space is a problem in computer music research. It corresponds to difficulty of the creation of a 3D audio sequencer, as this helped with the building of the HOA Library implementation in the 3D audio sequencer. Additional control inside the world is mapped to a Sony PlayStation 3 controller. However, Hoogland, has never used Soularis for a live performance. [2] Currently, there is no information about saving scenes for performance. In addition, Hoogland notes that the project is not finished and that he planned to release the project as an open-source program, but has not released an update in 3 years. [2]

B. HOA Library

The highest use of HOA Library for sound and musical spatialization occurred between 2012 and 2015. This period was when the active development was worked on and fostered at the University of Paris 8. Since 2012, HOA Library has been used in the framework of musical creation projects in live electronic composition workshops at the music department of University of Paris 8 and for several demonstrations. Those demonstrations that have occurred used spatialization tools based on High Order Ambisonics. Artistically, the research works in two areas: "the development of new functions of sound and space manipulation relevant to composition and musical context, and the adaptation of the spatialization tools to many uses and various restitution systems such as quadaphonic, or binaural." [8] HOA Library (v2.2), which was developed in 2015, "has made multiple improvements upon previous versions with updates and new features. Those updates and features include: new binaural rendering, new tutorials, exchanger for Ambisonics data format, and optimizations." [7] HOA Library's greatest asset is its use of Ambisonics. It approaches Ambisonics as the "processing of circular harmonics involving gain operations that allow simulation of the position of point sources or the application of transformations across the sound field" [8], such as reverberation.

Both previous examples, Soularis [2] and SoundStroll [3], were never used in a performance setting and had great limiting factors which contributed to their disuse. However, this project aims to develop an application, a 3D audio sequencer, that uses HOA Library (v2.2) and Ambisonics to manipulate the harmonics to use the sound space as an expressive dimension of music and sonic design that can be used. [8] Moreover, a more significant limiting factor of this system is the latency of samples can change depending on spatialization. Nevertheless, this project aims to develop an application with latency that is controllable by the performer, as well as not noticeable by the audience.

III. Program Design

"SonoSpatial Walk" is a 3D audio sequencer in the form of a Max Collective, that allows for the creation and editing of soundscapes in a 3D space for sound and musical recognition. The collective uses MIDI, Jitter, and HOA Library. It is dependent on nothing but open-source, cross-platform libraries. It is capable of the creation the soundscapes through using HOA Library (v2.2) for sound spatialization by getting the values of the environment from the Jitter and OpenGL matrix that create the objects, space, and scene by values. Furthermore, I use MIDI and Java to control reverberation, enable keyboard control, full-screen, non-movement commands, and navigation commands. Finally, I test the performance of the system by measuring the sound spatialization to get the finite impulse response (FIR) frequency response by using the buffir∼ object in Max to convolve an input signal with samples from an input buffer. The overall flow of the data in the application is shown in Figure 1. You can find the source code of this application on GitHub[12], as well as on my website.[13]

To start making a proper scene, these patchers must be opened:

- 1. Graphics (shapes.maxpat),

[9]https://github.com/vichug/soundstroll
[10]http://forumnet.ircam.fr/product/spat-en/
[11]https://www.timohoogland.com/soularis-3d-sound-sequencer/
[12]https://github.com/JustinKerobo/justinkeroboprojects
[13]http://justinkerobo.xyz/

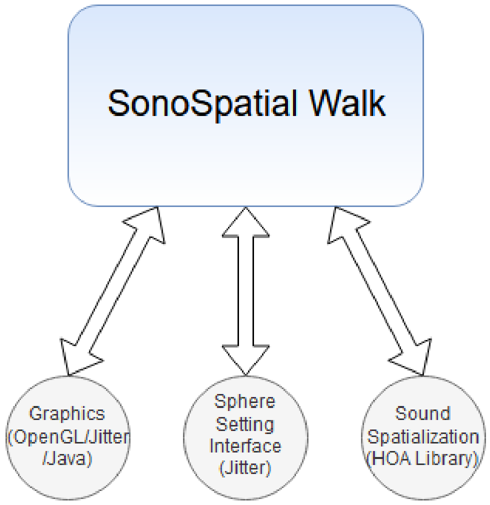

SonoSpatial Walk
Graphics (OpenGL/Jitter /Java)
Sphere Setting Interface (Jitter)
Sound Spatialization (HOA Library)

Fig. 1. Overview of the flow of data in the collective.

- 2. Sphere Setting Interface (sfcolorsgui.maxpat),
- 3. Sound Spatialization is happening (for example, HOA_soundhost_2.maxpat)

These 3 above-mentioned patchers should be launched on project load. Within this paper, the presentation modes of the patchers are shown, but are explained in the context of the full algorithm.

The spatialization patcher is linked to a specific spatialization tool and configuration: for another speaker setup or another spatialization engine (for example IRCAM's spat), another patcher will be needed. In this distribution, there are currently only two spatialization patchers which are working properly, and they are for a binaural headphones setup: HOA_soundhost_2.maxpat and IRCAMspat_soundhost_2.maxpat. The one using HOA is the default one. IRCAM's Spat is usable but is not free.

The 3D audio sequencer is based on sound spheres. There are two spheres, trigger and source spheres. A trigger sphere is a gridded sphere, much bigger than the source (opaque) sphere. When a trigger sphere is created, a matching source sphere is also linked to it. A source and its trigger will always have the same color. When one enters a trigger sphere, the matching source will be triggered, and, thus, spatialized and heard. When one leaves a trigger, the matching source will be stopped (immediately or not, depending on the trigger mode). Having those two spheres allows for distant sounds, which won't be triggered when the user is near the source, but only when they are inside the trigger sphere. This allows for more control in the sounds heard in the soundscape.

A. *Graphics using OpenGL and Jitter*

The graphics that make the 3D audio sequencer are shown in Figure 2, or shapes.maxpat. There are numbers within the Max patch put as labels, in yellow and blue. There are five number labels in yellow corresponding with functions inside the 3D world. The first enables graphics in the 3D world by allowing access to stop the rendering of the world on command. The second allows for the traversal of the 3D world with the ability to use the mouse and choose keyboard keys that map to movement. The third creates the starting plane that is shown in Figure 3, it also gives it a texture and transparency so that a user can see through it. The fourth allows for the control acceleration and speed of navigation, in addition to the ability to enable navigation commands using keyboard controls. The fifth allows for the placement and movement of the sound sources (spheres) within the 3D space. There are two blue number labels, the first corresponds to the trigger spheres and its properties within the 3D world, and the second is the same, but to source spheres. The perception of endless space is created by the 3D space enclosed by a skybox of six randomly-colored "walls" created by OpenGL and Jitter. Since it is a skybox, a user can never get closer to the walls. [1] This choice in the style of the 3D space was made to allow for an artist to have an unlimited canvas in which to place sound sources. The sound sources are figured by spheres of color. The spheres are also created by Jitter and the OpenGL matrix format. [1] To each source, there is a trigger related: a trigger consists of a gridded sphere, that is tied to its sound source sphere. Each time a user creates a new source, there will be a related trigger created. The "starting plane" is a beacon to always find the scene's origin at a glance. A skybox corner, spheres, triggers, and starting plane are in Figure 3.

All of these components and functions are connected together inside shapes.maxpat to allowed for creation of the 3D space, the traversal of the 3D space, the ability to save and read a scene within that space, and the ability to enable and use technical and navigation commands. Because of the Jitter and OpenGL matrix, the rendering of soundscapes are possible. Another major component that allows for a seamless user interface is Java and MIDI. Both connect within Max to allow for the ability to enable and use technical and navigation commands.

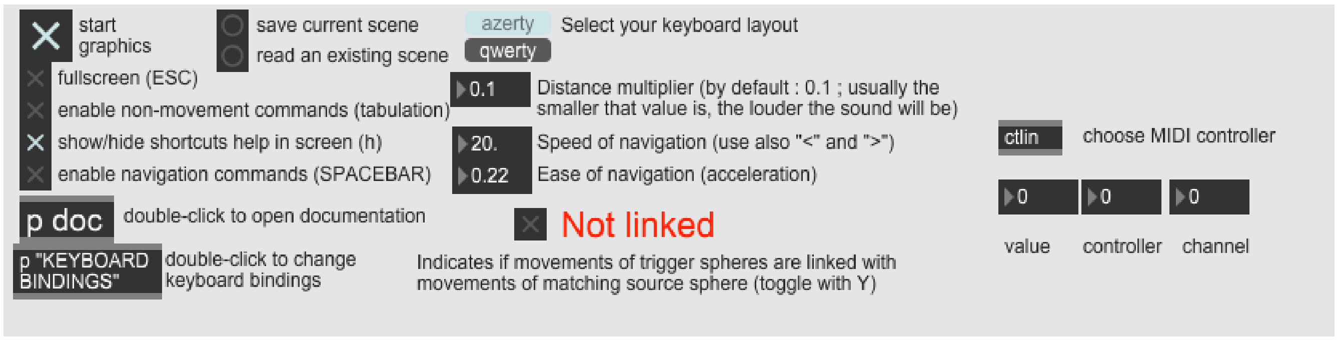
B. *Controls and Navigation through Java and MIDI*

Jitter is an excellent tool when used with OpenGL to make objects. In combination with Jitter, Java in shapes.maxpat is used to map keyboard controls within the 3D space created by OpenGL and Jitter. Within shapes.maxpat, it maps .json[14] files to Jitter by using a user interface mapper within the OpenGL and Jitter matrix format. To navigate, it works like a classic, first-person video game-like control scheme. The mouse is used to look around. The keyboard is used to move. In shapes.maxpat, the user can toggle between AZERTY and QWERTY keyboard with MIDI and keyboard controls.

In Figure 2, there is an object in Max that allows for MIDI to be mapped to control reverberation (Figure 4), enable keyboard control, fullscreen, non-movement commands, and navigation commands in the soundscape. The ctlin∼ object in Max allows the ability to receive the output received MIDI control values. The object directly outputs the value from a specific controller number and MIDI channel.

When moving the knob or pressing a pressure pad on a MIDI controller, the user can see the channel that the controller is on. The specific knob or pad is the controller on the MIDI device. When using it, the values change, and

[14]JSON (JavaScript Object Notation) is a lightweight data-interchange format.


start graphics

fullscreen (ESC)

enable non-movement commands (tabulation)

show/hide shortcuts help in screen (h)

enable navigation commands (SPACEBAR)

p doc double-click to open documentation

p "KEYBOARD BINDINGS" double-click to change keyboard bindings

save current scene

read an existing scene

azerty

qwerty

Select your keyboard layout

0.1 Distance multiplier (by default : 0.1 ; usually the smaller that value is, the louder the sound will be)

20. Speed of navigation (use also "<" and ">")

0.22 Ease of navigation (acceleration)

Not linked

Indicates if movements of trigger spheres are linked with movements of matching source sphere (toggle with Y)

ctlin choose MIDI controller

value	controller	channel
0	0	0

Fig. 2. Graphics (Presentation Mode): shapes.maxpat

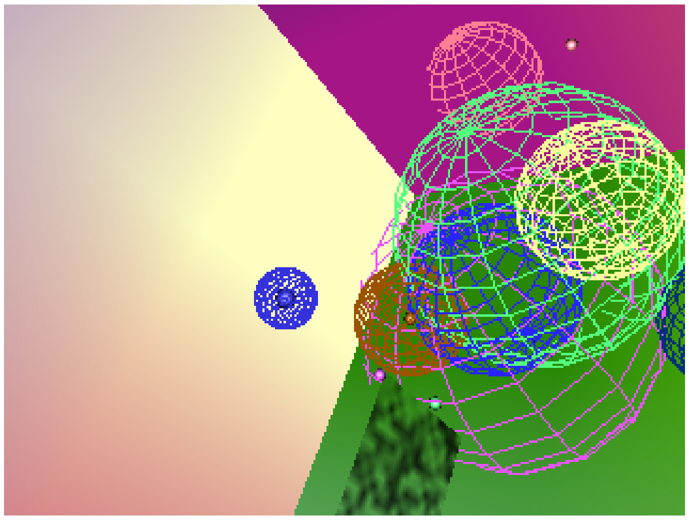
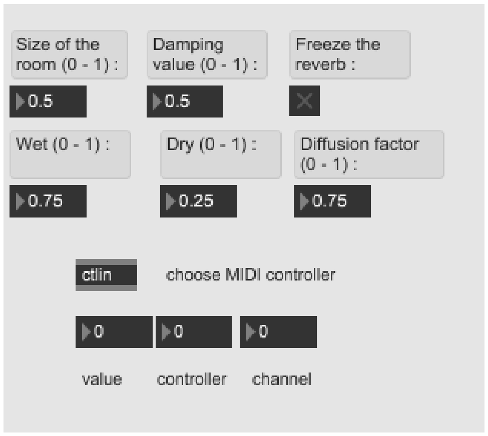


Fig. 3. A skybox corner, spheres, and triggers; plus the starting plane.

those instances are mapped to functions of the 3D audio sequencer in Max. In the reverb Max patch, or Figure 4, MIDI is mapped to the size of the room, dampening value, freeze of reverb, wet and dry audio effects, and the diffusion factor. This patch is also connected to HOA_soundhost_2.maxpat. In HOA_soundhost_2.maxpat it tracks sound information based on Jitter and OpenGL matrix values, which connect to HOA Library for sound spatialization. Because reverb is connected to HOA Library, it can change the values in the 3D space, making it larger or smaller, and also affecting what tones are heard.

C. Sphere Setting Interface with Jitter

The creation and placement sound sources (spheres) are determined by Figure 5, or sfcolorsgui.maxpat. This Max patch also allows for the saving and reading of scenes that one has created, along with shapes.maxpat. Other features include the ability to toggle "loop" or "trigger mode." The "loop" toggle allows one to loop one sound (on by default), and "trigger mode" changes the way it's triggered. If activated (by default), then the sound will be read until the end of the sound file when going out of the trigger sphere. If it's set on 0, going out of the trigger sphere will stop the sound as is, without going to the end of it first. In combination with spheres.maxpat, Java and MIDI controls are mapped to sfcolorgui.maxpat through the



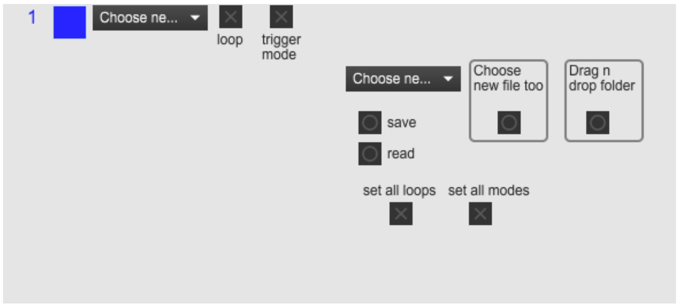
Size of the room (0 - 1) :	Damping value (0 - 1) :	Freeze the reverb :
0.5	0.5	
Wet (0 - 1) :	Dry (0 - 1) :	Diffusion factor (0 - 1) :
0.75	0.25	0.75

ctlin choose MIDI controller

value	controller	channel
0	0	0

Fig. 4. Max Patch (Presentation Mode): reverb

3D space. When a new sphere is created in the space, it will show in the patch. If the user moves away from the sphere, the sphere should be should be blinking. To stop the blinking the user needs to give the sphere a custom color. This can be done through a contextual or mini-menu. Afterwards, a user can click the colored area to choose a new color. The mini-menu is the umenu[15] object in Max. The umenu object allows text to be displayed by a pop-up menu. The mini-menu is used to manage sound files: there are different options to populate this menu, after what one can choose an available sound to assign to that sphere. The user may close the contextual menu by right-clicking anywhere. All spheres settings may also be changed from the patcher "sfcolorsgui."


1 Choose ne... loop trigger mode

Choose ne... Choose new file too Drag n drop folder

save

read

set all loops set all modes

Fig. 5. Sphere Setting Interface (Presentation Mode): sfcolorsgui.maxpat

[15]https://docs.cycling74.com/max7/maxobject/umenu

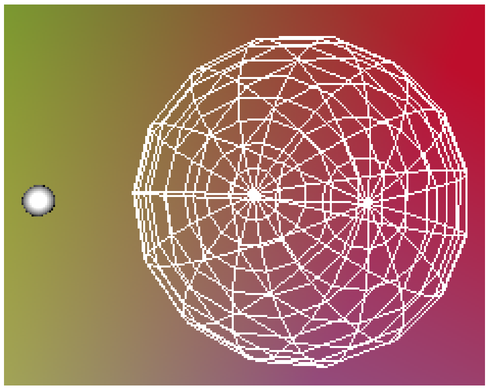


Fig. 6. A sound source (left) and its matching trigger sphere (right)

It is possible to move both sound sources and triggers, in both a linked mode - where the trigger stays centered around the source - and a not-linked mode - where the user can move a trigger sphere away from its source. To toggle one mode or the other, is mapped to keyboard controls. It is also a virtual button in shapes.maxpat. Users can move a source in linked mode, and its trigger will immediately position around it, even if it had been separated previously in not-linked mode. It is also possible to hide/show all the triggers spheres. Examples of source and trigger spheres are shown in Figures 6 and 7.

A sphere can also be moved in the space in all directions. Controls are also mapped to move a sphere towards the user or away. In linked mode, only source spheres can be moved. The trigger spheres move with it by default. A user can also change increase or decrease the size of a trigger sphere.

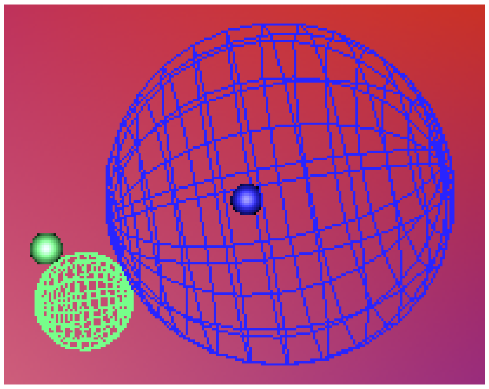


Fig. 7. Source and trigger apart (left) and trigger still linked around source (right)

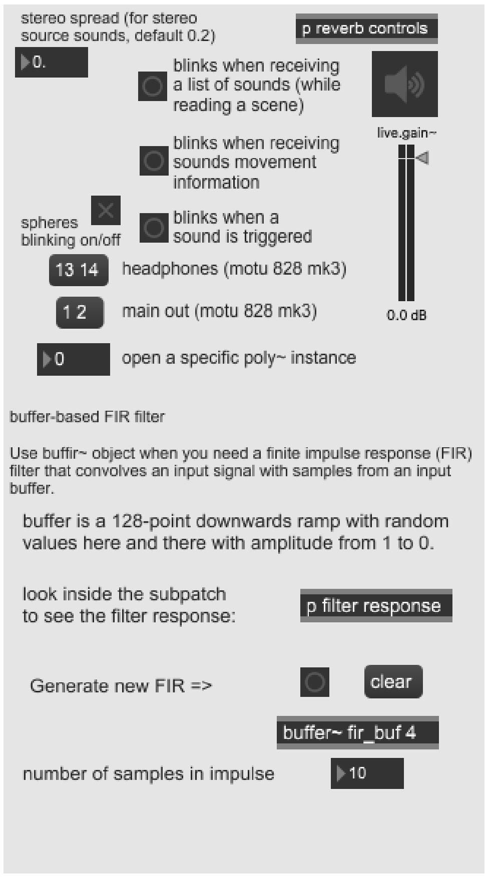
Once a user creates a scene in the 3D audio sequencer that they would like to use again, one can save it. To save, click the button labeled "save current scene" in shapes.maxpat or "save" in sfcolorsgui.maxpat. A save consists of several files: Four matrix files:

- Contains the color of each sphere/trigger duo (colors.jxf.jit),
- The size of each trigger sphere (scaleoftriggers.jxf.jit),
- For the coordinates of each source sphere (sources.jxf.jit),
- Gives the position of each trigger sphere (posoftriggers.jxf.jit).

There are also five text files, read by call objects:

- The sounds currently assigned to each source sphere (currentfiles.txt),
- Contains all the sounds available in this scene (soundscoll.txt),
- Remembers the file paths of sound folders (filepaths.txt),
- Saves the loop state for each sphere (loops.txt),
- Saves the mode state for each sphere (trigmodes.txt).

Saving will create all those files in the selected folder, so one may want to create a new folder and label it depending on scene. To open a scene, click "read" (sfcolorsgui.maxpat) or "read an existing scene" (shapes.maxpat) then select the folder where all those files are saved. The creation of the ability to save and read was made like this to retain creativity. By editing the values within the files, a user can make a soundscape without even being inside the 3D audio sequencer.



stereo spread (for stereo source sounds, default 0.2)

p reverb controls

0.

blinks when receiving a list of sounds (while reading a scene)

live.gain~

blinks when receiving sounds movement information

spheres blinking on/off

blinks when a sound is triggered

13 14 headphones (motu 828 mk3)

1 2 main out (motu 828 mk3)

0.0 dB

0 open a specific poly~ instance

buffer-based FIR filter

Use buffir~ object when you need a finite impulse response (FIR) filter that convolves an input signal with samples from an input buffer.

buffer is a 128-point downwards ramp with random values here and there with amplitude from 1 to 0.

look inside the subpatch to see the filter response:

p filter response

Generate new FIR =>

clear

buffer~ fir_buf 4

number of samples in impulse 10

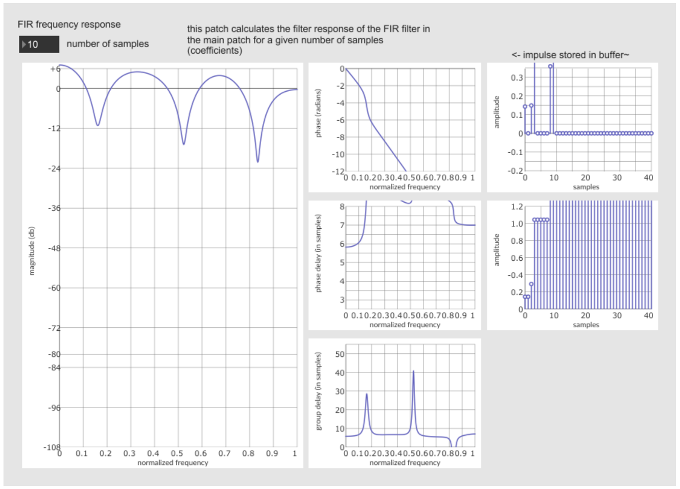
Fig. 8. Sound Spatialization (Presentation Mode): HOA_soundhost_2.maxpat

D. Sound Spatialization with HOA Library

The spatial and sound processor was made with HOA Library, which is shown in HOA_soundhost_2.maxpat, or Figure 8. It gets binaural audio and sense of space within the sound space by tracking information in HOA_twosounds.maxpat. It tracks when it receives sound movement information, when a sound is triggered, and when it receives a list of sounds while reading a scene. It uses that information and unpacks it into the pure data that HOA Library needs to play the sound binaurally. Within Max, it inputs that data into HOA_twosounds.maxpat. Finally, the data mapped through the 3D map of HOA Library and gives the sense of a 3D space sonally.

After that, it is put through hoa_processing, which allows it to be played out in the headphones or speakers. The way that hoa_processing allows for that to happen is through a dual output, and that is processed out directly into the live gain to be heard for playback.

1) Buffer-Based FIR Filter: Before directly processing the sound out to live.gain∼, I put it through the buffir∼ object first. The buffir∼ is used for the finite impulse response (FIR) filter that convolves an input signal with samples from an input buffer. [6] In a side Max patch, it can calculate the filter response of the FIR filter in the main patch for a given number of samples (coefficients). By using the samples, it change the contents of the signal. [6] When the input signal is convolved with samples from an input buffer, the signal travels horizontally, against the samples. [6] Depending on how many samples the input signal is convolved by, values are gained and are known as products. Products are the input signal and the samples that the buffer is convolved by, multiplied in pairs. [6] The sum of those products is the output sample, and will change what is heard inside the 3D audio sequencer.


FIR frequency response

10 number of samples

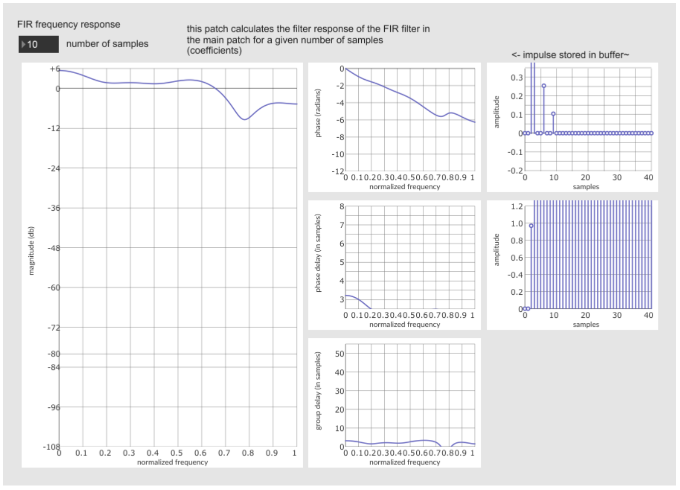
this patch calculates the filter response of the FIR filter in the main patch for a given number of samples (coefficients)

<- impulse stored in buffer~

Fig. 9. FIR Filter: First Convolve

It can also show the normalized frequency, the amplitude of samples, the magnitude of frequency, phase delay, phase, and the group delay all in samples as well. The graphs are made by using the sum of the products, which take information and produce a finite number of non-zero values, which also changes the sound in the 3D audio sequencer. [6]

It changes every time a user presses the button and convolves the samples with input samples, which indicates that 3D sound spatialization is working. It affects the response from the filter, as well as the sound heard in the 3D space. The change is shown in Figures 9 and 10. This was added for the ability to test the sound spatialization in conjunction with HOA_soundhost_2.maxpat. It registers as a sound being triggered in HOA_soundhost_2.maxpat, and the sound changes because the input signal is being edited with use of the filter.


FIR frequency response

10 number of samples

this patch calculates the filter response of the FIR filter in the main patch for a given number of samples (coefficients)

<- impulse stored in buffer~

Fig. 10. FIR Filter: Second Convolve

IV. Conclusion and Future Work

The collective performs better in terms musical and sonal latency (on lower latency systems). This is also shown through Figures 9 and 10. The normalized frequency of Figure 9 was done on a system that had half of the system memory committed to the program, or 16GB. The magnitude of the normalized frequency fluctuated more, which is shown in three dips that go from -10db (decibels) to -15db to -23db, and that distorts the sound because of the lack of audio processing power. The normalized frequency of Figure 10 was done on a system that had the full system memory committed to the program, or 32GB. The magnitude of the normalized frequency in Figure 11 generally stays the same, and only dips to -10db once. This shows better sonal latency on lower latency systems, in addition to sound spatialization working. A number of improvements can still be made for the sample latency of samples heard when spatialized inside of the spheres and the improvement of the spatial processor to allow for more forms of sound spatialization. As previously mentioned with HOA Library and Ambisonics, version 2.2 made multiple improvements upon the previous one, but still can improve in two major ways. The first is the latency and spatialization algorithm, as there are times where other objects in the sequencer can be heard with a delay from the reverberation even after the stoppage by the reverberation, but this usually occurs on higher-latency systems. The second is that with the 3D adaptations for a space that is not spherical in its design or a platonic solid[16] is hard to get optimal projection in the space. [8] With the skybox in Jitter that the 3D audio sequencer is spaced in, the acoustics are difficult to process with binaural [8], which the collective uses.

Besides HOA Library, another improvement that could be made in the collective is a completely other spatial processor. While HOA Library is open-source, IRCAM's Spat is not, but

[16]Only five platonic solids (tetrahedron, cube, octahedron, dodecahedron, icosahedron).

they have more active updates and features that a paid service provides. With it comes "an efficient signal-processing library" and over 250 external objects for Max. [5] The last main improvement that could have been made deals with MIDI. While the collective uses MIDI in a functional way, it does not use MIDI for direct musical assignment to the sound sources within the 3D audio sequencer. In addition, the collective was compiled and tested under Microsoft Windows 10.0.18362, but because it is a Max Collective, it is compatible for Mac computers as well.

In the end, the open-source collective was developed that allows the user to interact with their sounds in a 3D environment, and also hear how the spatialization would affect it based on reverberation values of the room. All things considered, any sample is playable and assignable to a sound source; one can compose and playback a scene using this software. Although HOA Library may no longer be actively developed by the CICM, the research using the spatial processor for a wide variety of applications has not ceased. With the emergence of better software and HRTF[17] databases, the idea of a 3D audio sequencer is closer than ever.

Acknowledgment

The author would like to thank David Barbella, Marc Benamou, Xunfei Jiang, Charlie Peck, and Forrest Tobey.

References

[1] Cycling '74. *Appendix B: The OpenGL Matrix Format*. June 2009. URL: https://docs.cycling74.com/max5/tutorials/jit-tut/jitterappendixb.html.

[2] Timo Hoogland. *soularis*. Jan. 2016. URL: https://www.timohoogland.com/soularis-3d-sound-sequencer/.

[3] Vic Hug. *Tool: SoundStroll*. 2014. URL: https://cycling74.com/tools/soundstroll.

[4] Jean-Marc Jot and Olivier Warusfel. "A Real-Time Spatial Sound Processor for Music and Virtual Reality Applications". In: *ICMC: International Computer Music Conference*. cote interne IRCAM: Jot95c. Banff, Canada, Sept. 1995, pp. 294–295. URL: https://hal.archives-ouvertes.fr/hal-01106971.

[5] Jean-Marc Jot and Olivier Warusfel. *Spat*. 2012. URL: http://forumnet.ircam.fr/product/spat-en/.

[6] Dan Lavry. "Understanding FIR (Finite Impulse Response) Filters-An Intuitive Approach". In: (1997).

[7] Anne Sedes, Paris Guillot, and Julian Colafrancesco. *High Order Ambisonics Library*. 2012. URL: http://hoalibrary.mshparisnord.fr/en.

[8] Anne Sedes, Pierre Guillot, and Elliot Paris. *The HOA library, review and prospects*. Sept. 2014. URL: http://speech.di.uoa.gr/ICMC-SMC-2014/images/VOL_1/0855.pdf.

[17] Head-related transfer function.