neThing: An Open-Source Web-based Graphical User Interface for Controlling Ambisonic and Wave Field Speaker Arrays

Xxxxxxxxx Xxxxxx
Yyyyyy yyyy yyyyyyyyyy
zzz@zzz.edu

Xxxxxxxxx Xxxxxx
Yyyyyy yyyy yyyyyyyyyy
zzz@zzz.edu

ABSTRACT

As multi-channel speaker arrays become commonplace audio environemnts within academic and research-oriented institutions, opportunities to expose casual listeners to the complex offerings of ambisonic and wave field systems are more prevalent. To standardize user interfaces for such systems, the neThing project introduces a graphical, easy-to-use, free, open source, and educational interface for ambisonic and wave field arrays. By creating a simple and enjoyable user experience for the server and user side, we attempt to remove technical and conceptual barriers that potentially hinder access to such systems, while at the same time increasing interest to use and maintain large-scale systems. This web application aids in making already structured ambisonic and wave field arrays more accessible for education, performance, and research. By creating a user friendly experience, people can begin to ask more questions and experiment with this format of audio. This method effectively lowers the learning curve required to begin work with ambisonics. Finally, by using a web application, we reduce the amount of software needed to be distributed across users, allowing for mobile use as well. The software will be downloadable for users to run on a closed network if so desired.

1. INTRODUCTION

Though hardware and software systems used to design, configure, and drive multi-speaker ambsonic and wave field speaker arrays are technologically complex, there is still development that can be done for presenting demonstrations or concert playback of rendered material or dynamic sound sources. This project was launched by a need to quickly and easily demonstrate and reconfigure speaker arrays in multiple venues in the Department of Arts and Experimental Media and Performing Arts Center (EMPAC) at Rensselaer Polytechnic Institute(RPI). In particular, the launch of the Department of Arts' new recording studio and multi-channel audio space (Hamilton, ICMC 2017) brought a new 496-channel wave field speaker array designed and built at EMPAC which spurred the need for an easy to use interface.

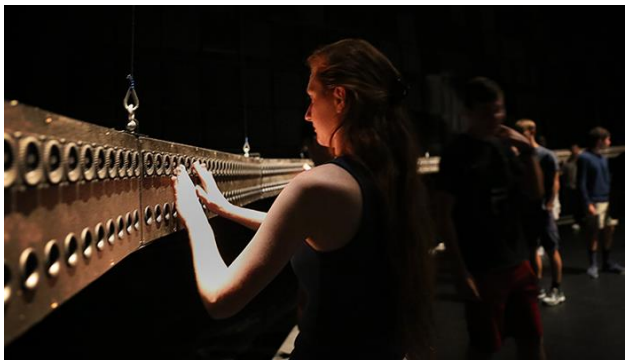

Figure 1. The 496-channel wave field speaker array built and designed at Rensselaer's EMPAC.

2. SYSTEM OVERVIEW

This project incorporates an easy-to-use web-based Graphical User Interface(GUI) for mapping individual sound sources to locations in space with the development of a fully-fledged multi-speaker demonstration system application suitable for novice users. The intent is to allow all owners of an ambisonic or wave field speaker array to be able to start and reconfigure it with a goal of demonstration and education. The web server itself contains sample sounds and spatialization examples suitable for demonstrating the capabilities of such speaker systems. Users can expand the sonic offerings of the system by downloading a client version of the software to run on their own custom web server, users can expand the sonic offerings of the system. Furthermore, because this piece of software runs as a web application, it can be used on any major web browser, as well as on mobile devices. This program only provides graphical interfacing and no actual sonic control besides visualization of where a sound is placed, so the program assumes that the speaker array itself be already physically configured and attached to an encoder and decoder as necessary. Therefore, the more precise the ambisonic array is arranged, the more accurate the estimated node placed on screen will appear to sound in space. The software uses Ruby on Rails for the backend server code to communicate with the user over a network. Bootstrap and JavaScript are both used for front end dynamic interfacing. The stock audio comes with attack, decay, sustain, and release control for several sample sounds, created using Max MSP.

3. PRIOR WORK

Graphical user interfaces for controlling ambisonic and wave field spatialization software packages are today as commonplace as they are useful, often integrating a set of GUI elements into the same software environments used to encapsulate the ambisonic encoding and decoding algorithms themselves such as Max MSP (Carpentier; Sacher 2010), Reaper/VST (Reaper plugin), SketchUp () and Unity().

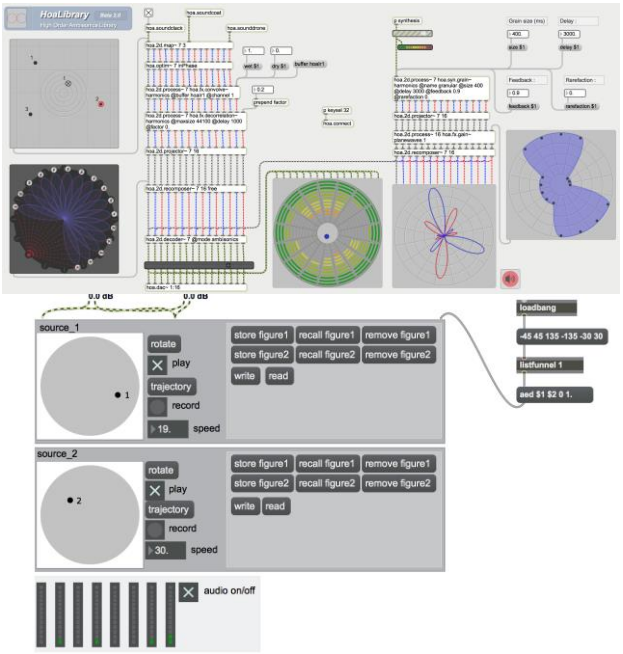

Figure 2. HoaLibrary v2, a Max MSP interface for spatializing sound, and a series of objects from Philippe Kotcher of the ICST (Institute for Computer Music and Sound Technology) of the Zurich University of the Arts. While useable, they lack basic features that are visually appealing, and are complicated to use without an in-depth understanding of their inner workings.

Powerful systems have been designed to run custom speaker arrays and configurations such as the OpenMixer project (). However, the interfaces for each of these toolkits and systems are disparate, disconnected, and need customization or custom control-data routings in order to make them interoperable. In the context of a high-turnover multi-user multi-system environment, switching from one system and interface to another can be a daunting challenge for experienced sound designers and engineers, let alone for novices in multi-channel speaker environments.

4. USER CONTROLS AND FORMAT

4.1 Spatial Sound Interface

The stock user interface is designed for educational purposes to engage an unexperienced user with new sonic abilities they have perhaps not yet experienced. This, of course, can be modified with a basic understanding of JavaScript, Max MSP, and Ruby on Rails.

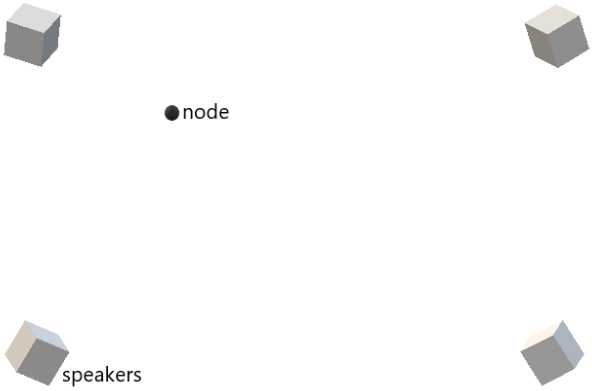

Figure 3. An early conception of the basic user interface, in this figure, for four speakers in an array, but can be modified for any number and placement of speakers.

For the user interface itself, it pulls from a minimal design displaying only what is important, as shown in Figure 3. In a configuration where multiple users can connect to the web server at the same time from multiple web-enabled device, each user can subsequently place and control sound objects within the space. As more users fill into the space and use the array, there are server setting that allow for a user to see where other users are placing their own nodes, shown in a different, grey scaled color. A user simply needs to click or tap the screen to trigger their sound generation. Holding the screen will sustain the sound depending on the specified attack, decay, sustain, and release settings the user defined. When holding the node down on the screen, users can slide their finger or move their curser to new areas of the screen to move the sound throughout space. Users can maintain their specific location by double clicking or tapping to make a node stationary on the page. Depending on the server settings, the sound can either have a constant sustain or only trigger when a user clicks or taps it.

4.2 ADSR and Sound Controls

The ADSR controls are currently modeled on Max MSP's function object, compiled down to a web executable and placed in a tab on the application for sound control. A snapshot of the wave that is being played by the user and the shape its amplitude makes over time is shown. This allows the user to visualize the sound they are hearing and experiment with creating new sounds. Each user has fine control over the attack, decay, sustain, and release of their sound, plus the ability to edit the curves of the function itself, for rounder and smoother sounds.

4.3 Backend Server Controls

The server side has extra controls for maintenance and CPU usage to help with any errors that might occur. In the event of too many processes created, the server controller can kill one, many, or all processes running. Statistics on the speaker arrays are available including gain staging of sound and number of processes currently creating sound on the array. In the event the server owner does not want someone to access the server without proper

authorization, each server can be assigned a unique server key that can allow users to log into a specific server. The server controller can give users the ability to edit the sound by dragging a waveform into the snapshot. Several other options are available, though disabled by default, are fine tuning of pitch, amplitude shaping, multiple node creation, saving of particular sounds to a user profile, and a limit of sounds that can be saved. The maximum nodes that can be created, both by users and in total for a particular server also can be detailed in the server specifications.

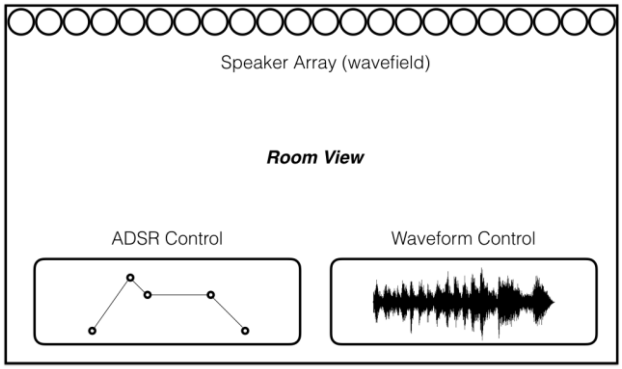

Figure 4. Tablet interface displays current selected speaker configuration, ADSR controller and currently selected waveform.

4.4 Main Web Application Features

The main web application hosted on the static website is unique as it allows users to create an account to be able to log into any server that is created and set up on the internet. If a client decided to save the settings of their speaker array on the server instead of running their own server instance, they can do so and it will be saved in the main database. This includes the information detailed in section 4.3 of this paper – *Backend Server Controls.* The usefulness of this is that to create and modify backend server settings, a user simply needs to create a user account as a server controller instead of a basic user. A user can also be able to log into different servers and have their settings for each one saved, and since there are universal settings that are defined on the main web application, a user's personal sound library and recently used servers can be retrieved once they log into the application. A user does not need to create an account, as long as they are able to connect to a specific server, based on the network they are on, and sever key they have.

5. AUDIO GENERATION

5.1 Max MSP

In the initial implementation of this system sounds are created using Max MSP, and run entirely on the server which the speaker array is connected to. In a future release, the users would have the opportunity to generate sound from their own devices as well, but currently it is easier to create sound from one audio generation source.

5.2 Sound Design

The sounds for the initial sample sets that are used are developed with easily distinguishable acoustic features. In this case sounds were generally midrange to high end pitched sounds, as well as complex waveforms created from simple shapes that change over time, usually with added noise and in some cases generated purely from noise. Resonant filters were used for "clicky" sounds, which are helpful in being able to discriminate them spatially from other sounds in the room. Random variables are introduced when multiple people are in a room using the array, as to avoid confusion of who's sound belongs to who. One set of sounds used were directly inspired by Andy Farnell's book, "Designing Sound" . As an artistic choice sounds found in nature, and even more specifically, insect sounds, as while they are very easy to pinpoint, such a fly's wings, or a mosquito's tonal buzz were explored in our initial sample sets.

6. BASIC REQUIREMENTS AND SETUP

The web server runs Rails 5.1.4, and Ruby 2.3.5. A user can download and run the server website locally on their own machine if they install Ruby and Rails[1]. Ensuring that their version of Ruby and Rails matches the one listed above, the user then launches the server by typing "rails s" to start the server within a terminal open to the repository.

7. HIGH LEVEL FUNCTION OF USER AND SERVER COMMUNICATION

The flow of the server is as follows:

1. The web server hosts user processes and is in charge of generating the sounds.
2. For every user connected to the server, the server created a new child process that maintains the preferences of that user.
3. Each user can create as many new child processes of nodes that generate sound as specified by the global maximum of the specific server.
4. Child sound nodes from the user are only created when the ADSR controls are toggled and are terminated after the sound has completed the decay.

Because the sound takes time from the point at which the user clicks or taps to start the attack, there will always be a time delay. Depending on the internet connection, it can take anywhere from 10 ms up to 100ms. These numbers are just an estimate; further testing will be required to determine optimal server and upload/download speeds, both from the main web application and also the downloadable server. The goal is to create a fast enough method for network communications to appear seamless. One

[1] www.installrails.com

workaround is using slow attacks, for pad and ambiance like sounds, to trick the user that the sound only appears to sound delayed because the attack is long (one second for example).

8. CONCLUSION AND FURTHUR STUDIES

The use of ubiquitous web technologies to interface powerful ambisonic and wave field encoding and decoding algorithms has allowed our students and less technical faculty to begin exploring the complexities and sonic advantages of large-scale speaker systems. The system is already in use for tours and demonstrations of the new recording studio space in the Department of Arts and by the time of this publication, the system will have been configured for use in a special demonstration room at EMPAC for components of the wave field speaker array. As further implementations continue, we plan on running a user study to gauge the successes and failures of our approach, allowing for future iteration and design improvement.

9. ACKNOWLEDGEMENTS

Thanks to Fernando Lopez-Lezcano for his guidance and support and to the technical and research staff of the EMPAC center and the Rensselaer Department of Arts.

10. REFERENCES

[1] L. Aspock, S. Pelzer, F. Wefers and M. Vorlander, A Real-Time Auralization Plugin for Architectural Design and Education, In Proceedings of the EAA Joint Symposium on Auralization and Ambisonics, Berlin, Germaby, pp. 156-160, April 3-5, 2014.

[2] T. Carpentier, M. Noisternig and O. Warusfel. Twenty Years of Ircam Spat: Looking Back, Looking Forward. 41st International Computer Music Conference (ICMC), Sep 2015, Denton, TX, United States. 41st International Computer Music Conference (ICMC), pp.270 - 277, 2015.

[3] A Farnell. Designing Sound. MIT Press, 2010.

[4] R. Hamilton and C. Bahn, "Interactive Music and Media @ Rensselaer: Studio Report 2017" In Proceedings of the International Computer Music Association Conference, Shanghai, China, 2017.

[5] F. Lopez-Lezcano and J. Sadural, "Openmixer: a routing mixer for multichannel studios," in Proceedings of the Linux Audio Conference 2010, 2010.

[6] T. Lossius & J. Anderson (2014): ATK Reaper: The Ambisonic Toolkit as JSFX plugins. Proceedings of the joint 40th International Computer Music Conference & 11th Sound and Music Computing Conference, Athens, pp. 1338-1345.

[7] D. Malham and A. Myatt. 3-D Sound Spatialization using Ambisonic Techniques. Computer Music Journal, 19(4):58–70, Winter 1995.

[8] J. Sacher, Seven Years of ICST Ambsionics Tools for MaxMSP – A Brief Report, In Proceedings of the 2nd International Symposium on Ambisonics and Spherical Acoustics May 6-7, 2010, Paris, France, 2010.

[9] Unity Documentation, "Ambisonic Audio", Unity User Manual (2017.3), accessed February 10, 2018: https://docs.unity3d.com/Manual/AmbisonicAudio.html