Acronyms

VAE Virtual Acoustic Environment . . . 6
VSS Virtual Singing Studio . . . 4
RIR Room Impulse Response . . . 6
ISM Image Source Method . . . 8
TO Transition Order . . . 10

first: Click here

Second: Click here

Contents

INTRODUCTION

No prior understanding of room acoustics is required to appreciate the difference in sound experienced when talking, singing or playing an instrument in rooms of different size and interior. For those who listen even more closely, the effect of standing in different positions of a room can also be appreciated, such as the deep booming sound of standing in a corner or the fluttering echo heard when clapping in the centre of two symmetrical walls. However, the benefits of the hard work that has gone into studying room acoustics and sound propagation are apparent across multiple platforms, from understanding how to acoustically treat a room to make it sound the way we want, to creating tools and defining methods that allow us to recreate or synthesis a rooms acoustics, leading to the ability to develop realistic soundscapes that can bring virtual worlds to life, such as in video games and films.

The ability to do such things has been utilised in other areas of research, such as studying how musicians perform differently when playing in rooms with different room acoustics [1]. This has been done at the University of York, where the benefit of being able to simulate the acoustics of multiple rooms within moments has been used to effectively transport musicians and researchers to different venues without having to travel anywhere. The system used to do this is the Virtual Singing Studio (VSS), upon which this project is based. In addition to allowing a user to simply hear themselves in different venues, this project aims to allow the user to also move themselves around that venue, allowing them to experiment with performing in different positions.

The rest of this document will be structured as follows:

Background
An overview of all the material that form the basis of this project, including an overview of the software used to achieve the end result. Previous work is discussed, leading to a full description of the project including the motivation behind it. Finally, the project aims and objectives are stated, including how the system was to be produced and tested to access plausibility, providing a set a statements that are used to measure the success of the project upon completion.

Implementation
This section presents the practical work that was carried out throughout the project, describing the decisions made, issues encountered and how they were overcome.

User Testing
The procedure that was carried out to test the implemented system is described and the results are presented and discussed, followed by a conclusion, highlighting potential reasons for the obtained results.

Project Management
A brief overview of how the project was planned and time managed.

Conclusion

The project as a whole is summarised and measured against the initial metrics set, evaluating the overall success of the project. The issues raised throughout previous sections are reviewed and potential solutions are discussed. Further improvements to the system are suggested, including some of the work that was initially intended to be carried out.

BACKGROUND

The following section covers material that forms the basis of the system produced as a result of this project.

3.1 - VIRTUAL ACOUSTIC ENVIRONMENTS

Virtual acoustics has been previously described [2] as follows:

> *"Virtual acoustics is a general term for the modelling of acoustical phenomena and systems with the aid of a computer"*

By this definition, a Virtual Acoustic Environment (VAE) can be thought of as an environment (such as a room) for which the acoustical phenomena have been either recreated or synthesised. To produce a VAE, prior knowledge regarding the room which is to be acoustically recreated must be known; how do all audible frequencies propagate around the room for a set sound source location and receiver location?

This information can be gathered by taking a Room Impulse Response (RIR) and used to recreate the acoustics of a room for the set sound source and receiver location.

3.2 - ROOM IMPULSE RESPONSES

In order to reproduce the acoustical phenomena of a room, an RIR must be obtained. This is done by exciting all audible frequencies within the room by using a sound source such as a loudspeaker, and recording the result using a receiver microphone.

There are a number of techniques used for exciting all audible frequencies. These include an **impulse** (such as a starter pistol) or an **exponentially swept sine** for which a sine wave is exponentially increased in frequency over a fixed period of time. The difference between these techniques is that an impulse emits all of the audible frequencies into the room at the same time, whereas the sine sweep does it gradually. As the idea is to obtain an **impulse**, using the impulse technique is much more simple as no post processing is required. Using an exponentially swept sine requires post processing in order to time align all of the frequency dependent room reflections through the use of a deconvolution algorithm and thus producing an impulse response, making this method a little less simple. However, this method produces a greater signal to noise ratio thus is the desired method [3]. Once the impulse is obtained, it can be convolved with an audio signal, making that audio signal sound as though it is being produced within the room that the RIR was taken from. Convolution simply takes an input signal and multiplies each of the discrete samples with

the impulse response, thus simulating the effect the room would have on each of these samples if they were actually emitted in the room itself [4].

3.3 - AMBISONICS

Though using an omni-directional microphone to record an RIR is a set standard [5], it is also possible to use techniques such as Ambisonics to record RIR's that can be used to reproduce a soundfield in three dimensions.

Ambisonics is a technique used to encode and decode three dimensional spatial audio information using just four audio channels. A three dimensional sound field can be recorded using a microphone known as a Soundfield microphone shown on the left in figure 1. These microphones contain four coincident capsules, one of which is an omni-directional capsule (W) and the rest of which are figure of 8 capsules used to record sound in the X (front and back), Y (left and right) and Z (up and down) direction illustrated on the right in figure 1. In theory, the aim is to record the sound field at a single point, however this will never be possible given that the microphones take up physical space. Therefore, the sound captured by the microphones are first recorded into A-format (raw unedited signals) and then converted to B-format which is the same signal with an applied inter-capsule time correction to compensate for the physical separation of the microphones [6] and the fact that they are not absolutely coincident [7]. Once the sound field has been captured, a system specific decoder can be used to replay the captured signal over an arbitrary number of loudspeakers.

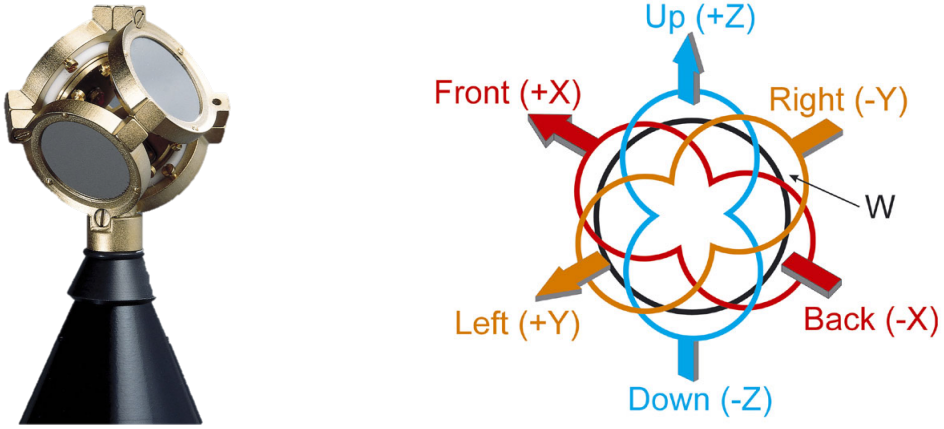


Figure 1: **Left**: Picture of a Soundfield microphone with coincident capsules exposed. **Right**: Soundfield microphone polar pattern. Images sourced from [8] (background removed from left image)

3.4 - Software Overview

3.4.1) ODEON: Simulating Room Acoustics

Methods for physically measuring RIR's within rooms has been discussed, however there is also a way to synthesis RIR's by mathematical modelling the way in which sound interacts with a room. The two methods of doing so are described as **Wave-based Methods** and **Geometrical Acoustic Methods**.

Note: The next paragraph on wave-based methods is adapted from the initially submitted literature review and is not required for the understanding of this project. However, it is provided here as the initial project plan included the use of such methods and is therefore mention later in this section

Wave-based methods are based on solving the wave equation, producing a solution that is physically correct and can be used to accurately model how a sound would propagate from a source in a room and reflect [9]. However, accurately calculating the way in which a wave moves through and interacts with a room is computationally expensive making them impractical for real time simulations.

Instead, it is possible synthesis RIR's by using room acoustic simulation software such as Odeon [10] that utilises the much quicker geometrical methods. Odeon was designed to provide reliable predictions of room acoustics by using a hybrid of two geometric acoustic modelling methods: **ray-tracing** and the **Image Source Method (ISM)** to synthesis RIR's. These methods model how sound would propagate and interact with a room as though it were a straight line (ray). This inherently neglects wave phenomena such as phase and diffraction, properties that are negligible at high frequencies, however they are fundamental in describing low frequency wave behaviour [9]. Therefore geometrical methods are not accurate at modelling sound propagation for low frequency waves, however they provide a good approximation for most applications and are much quicker to compute.

A description of both of the geometrical methods are described below, leading to a description of the hybrid method used in Odeon.

3.4.1.1 Ray-Tracing

The ray-tracing method imitates a sound source by emitting a large number of particles in various directions from a single point [11]. These particles are then traced around the room, losing energy each time the particle encounters a surface by a factor of $1 - \alpha$, where α is the absorption coefficient assigned to that surface, illustrated in figure 2. The angle at which the particle is then reflected is determined by the scattering coefficient assigned to the surface of contact, ranging from a specular reflection to a completely random reflection [12] (described in section 3.4.1.4 Scattering). For a

specific receiver position, an area around said point is defined in which rays are collected and used to calculate the results.

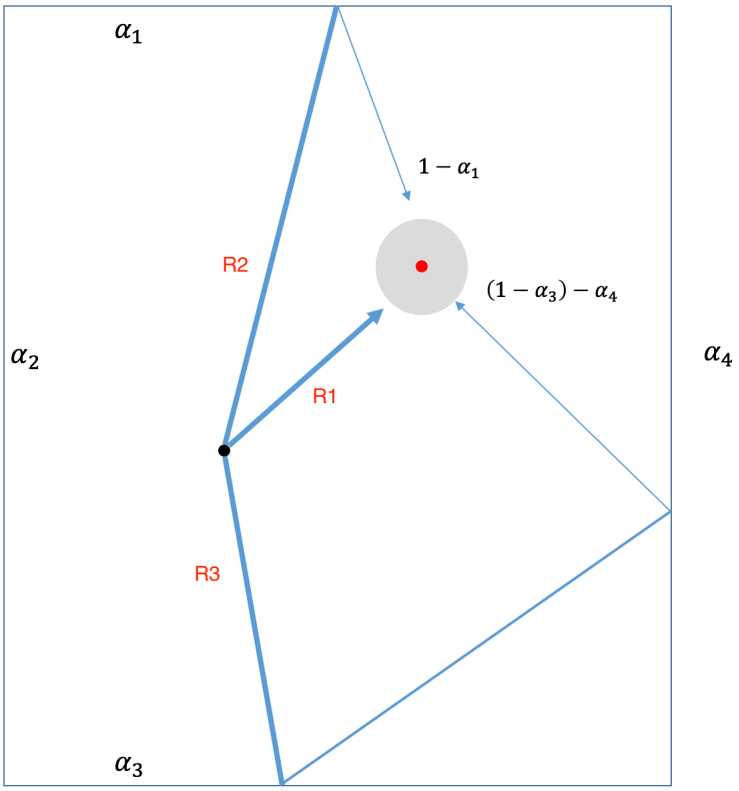


Figure 2: Illustration of the ray-tracing method (by the author) showing a sound source (black) emitting three rays (R1, R2 and R3) and shows how each ray is attenuated by a factor of $(1-\alpha$ each time it is reflected from a wall. The red dot represents the receiver positions and the outlining gray circle represents the area in which rays need to pass through to be used in the RIR calculation.

Ray-tracing does not provide a completely accurate result as it is a risk that some rays may not pass close enough to the receiver to contribute to the final result. The outcome of the ray-tracing method is a statistical result rather than a complete one.

3.4.1.2 Image Source Method

The ISM can be used to find all possible specular reflections paths from the sound source to the receiver position. This is done by representing each specular reflection (a reflection with angle equal to the angle of incidence) from a surface as a secondary source known as an "image source" [11], illustrated in figure 3. Once these image sources have been created, a visibility test is run which checks to see which image sources are in sight of the receiver thus calculating which image sources should be used in the final calculation of the RIR. If the receiver is then moved, the images sources to not need to be recalculated, only the visibility check needs to be run again.

It is due to this that the ISM is much more accurate than ray- tracing which relies on random reflections to calculate an RIR. However, as each new image source emits a number of rays of its own, there is a potential for it to produce a number of image sources. Therefore, with each new image source there is an exponential growth in the total number of image sources, making this method much more computationally expensive than ray-tracing. This problem can be avoided by setting a *reflection order* which determines how many times a ray can reflect around a room before calculations are stopped, thus preventing the creation of more image sources. This however will cause the prediction of the rooms acoustics to be incomplete. This obviously then makes the ISM less accurate with a smaller reflection order.

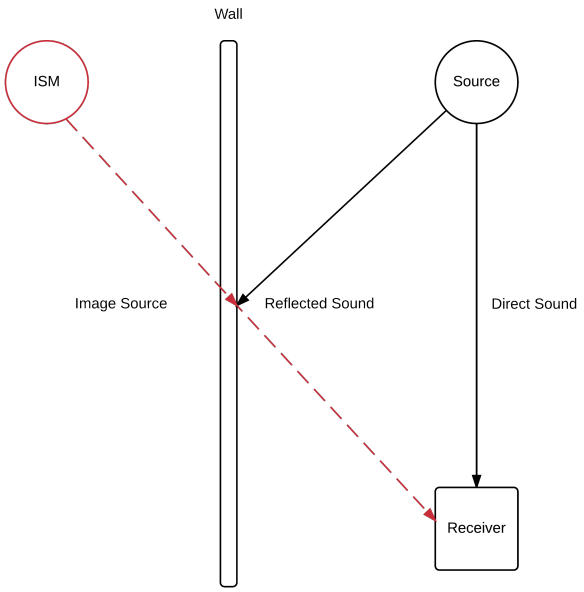


Figure 3: Illustration of the image source method (by the author) showing how the reflection from the sound source to the receiver is modelled by a secondary sound source.

3.4.1.3 Hybrid Method

Ideally, the ISM would be used to calculate all sound rays due to its accurate results, however due to the computational limitations, Odeon uses a hybrid method to provide a reasonable compromise between calculation time and accuracy.

The hybrid method first uses the ISM to calculate a number of image sources up until a specified reflection order determined by a Transition Order (TO). For example, if the TO = 2, the image source method will allow a ray to reflect twice which will produce a number of image sources at which point it will then switch to using the ray tracing method to calculate a statistical model of how the rest of the rays might interact with the room. This gives the user the choice between computation time and accuracy.

3.4.1.4 Scattering

Unless a surface is infinitely smooth (which almost all surfaces are not), when sound comes into contact with that surface it is scattered at a angle that deviates for that at which it hit the wall. Odeon takes this phenomena into account by using 'Vector Based Scattering' [12]. This is done by adding the specular vector to a randomly reflecting vector that has been scaled by a value of 1-s, where s is the scattering coefficient applied to a surface (ranging from 0 - 1). This essentially calculates how much a specular reflection should deviate based on how 'rough' a surface is, where the rougher the surface, the higher the scattering coefficient. Figure 4 shows an annotated image taken from the Odeon manual illustrating this concept.

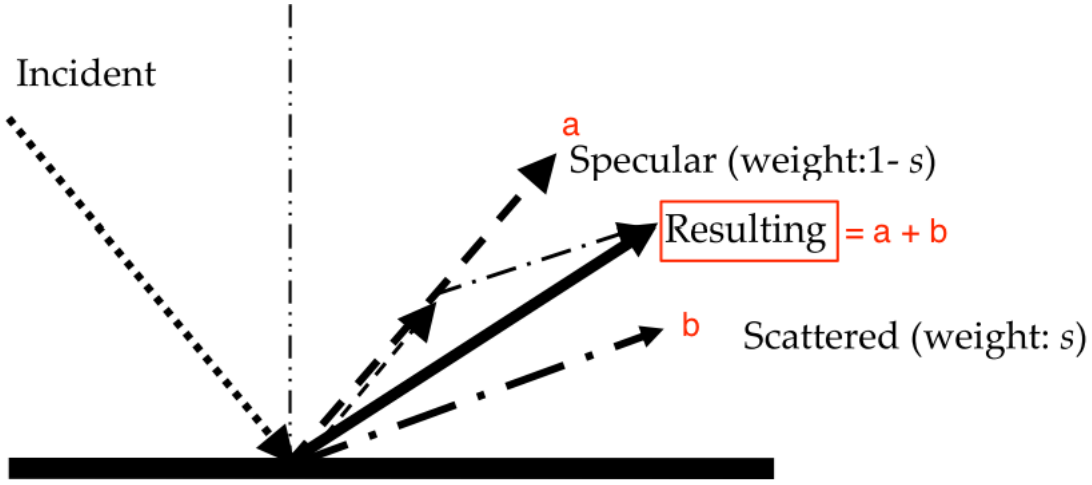


Figure 4: Annotated image from the Odeon manual [12] showing how scattered reflections are calculated, where a = the specular reflection, b = randomly scattered vector.

3.4.1.5 Room Modelling

In order to predict the acoustics of a space, Odeon requires a geometry file in the form of a .par file. This file contains information regarding the room dimensions, object dimensions and positions. This can be produced within Odeon itself by using the built in 'Extrusion Modeller' which allows a user to use a script like language to describe the rooms geometry. It is also possible to use 3rd party applications such as Google SketchUp [13], a software which is described in section 3.4.2 Google SketchUp.

3.4.1.6 Material Selection

Selecting the appropriate absorption coefficients of the surfaces within a VAE is crucial to determining how sound will attenuate as it reflects around the room (as described in 3.4.1.1 Ray-Tracing), thus the accuracy of the generated RIR. Odeon provides a material list with common

materials with pre-determined absorption coefficients that can be assigned to the surfaces of a model read from a geometry file. This material list can be extended by creating new materials and assigning absorption coefficients to the appropriate frequency bands.

3.4.2) Google SketchUp

Google SketchUp [13] is an easy to use 3-D modelling software that can be used to produce room models. Unlike Odeon extrusion modeller, it allows the user the draw surfaces with a mouse, easily duplicate structures and provides simple measuring tools and markers to enable accurate modelling. Plug-ins such as SU2Odeon ("SketchUp to Odeon") [14] enables the user to convert the model into a .par file for Odeon to use as a geometry file.

3.4.3) Max/MSP

Max/MSP (Max) is a visual programming language that essentially provides pre-written blocks of code in the form of objects, providing the ability to chain them together in order to route and manipulate audio signals [15]. This allows user to quickly and easily produce audio applications (called patches) without having to deal with the complex side of audio programming. This is beneficial for students and researchers as it allows them to concentrate of the results rather than the application programming itself. Additionally, third-party plug-ins can be incorporated into a Max patch as libraries would in any other programming language. One third-party application utilised in this project is Spat. Developed at the research institute IRCAM [16], Spat is designed for real-time spatialisation of sound signals in Max without having to touch any code. It provides an simple way to convolve B-format audio signals with impulse responses as well as decode the B-format signals for any given number of speakers in any arrangement by simply providing the number of speakers and angles at which they are placed.

3.5 - Project Description

This section starts by explaining what the VSS is and how this project will build upon it. The motivation for doing so is covered and the project objectives and metrics are stated.

3.5.1) The Virtual Singing Studio

The VSS is a loudspeaker based room acoustics simulator used as a tool for analysing the correlation between room acoustic characteristics and vocal performance parameters as part of Dr Jude Brereton's PhD Thesis [1]. It is comprised of a head-mounted microphone used to capture a real-time audio input from a singer, a software patch written in Max/MSP that convolves the audio signal with a number of Ambisonic B-Format RIR's using Spat and finally a spherical array of 16 loudspeakers for which the convolved audio signal is decoded and fed to allowing the performer stood within the speaker array to hear themselves as though they are stood in the VAE.

In addition, a head-tracking device (an Oculus Rift [17]) is also used to track which direction the user is facing in the virtual space. A flow diagram of the system can is shown in figure 5.

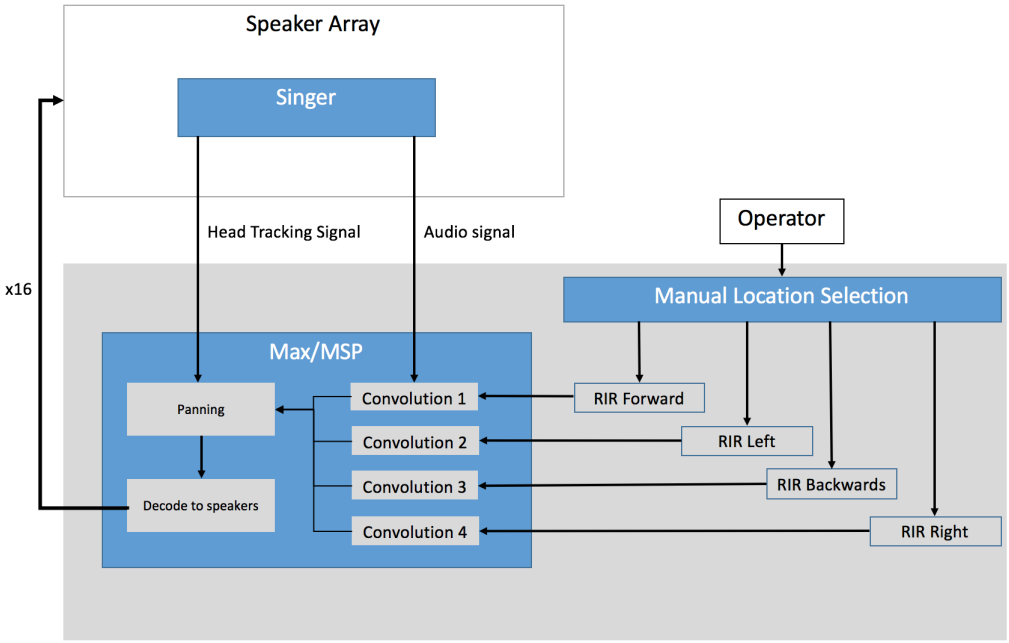


Figure 5: Flow diagram of the VSS showing how the audio signal and head tracking signal form the performer are used in Max before the convolved audio signal is sent back out to the speaker array.

The VAE used in the VSS was initially the National Centre for Early Music, a space in York frequently used for musical performance. Using a Soundfield microphone, Ambisonic B-Format RIR's in each direction (front, left, back, right) were captured in 4 locations. The four directional RIR's are used to approximate the room acoustic phenomena that would occur if the user were projecting whatever sound they are producing in that direction. This is done by using the data from the head-tracking device to amplitude pan the convolved signals before sending them to the spherical speaker array.

3.5.2) Project Aims and Motivation

The VSS addressed a problem faced when trying to research how musicians perform in different acoustic environments: having to travel to each performance space with musicians and researchers. This also indirectly provides a solution for performers wanting to rehearse in spaces that are often inaccessible and would otherwise be expensive to book and travel to. By obtaining an RIR of the desired location, the only time travelling will be necessary is to initially obtain said RIR. However, one limitation of using the VSS is the restriction of position. If a performer wanted to try and sing at another point in the room, an RIR would have to be taken in that position too.

This could be done initially, taking a range of RIR's in a number of positions, however it cannot be guaranteed that all positions desirable to the performer will be available. For this to be certain, technically an infinite number of RIR's would have to be taken. As this is obviously not possible, the next best thing would be to take a large grid of RIR's and simulate movement by interpolation between a number of neighbouring RIR's. This raises the question:

> **"How many RIR's are required to convince a user that they are able to move to any position they like within the virtual space?"**

As investigating this question would require producing a large grid of RIR's, measuring them in a real space would be impractical due to the sheer number that need to be taken for experimentation. Instead, Odeon can be used to produce a bulk of RIR's much quicker.

This technique has been considered previously in [18], referring to it as the *direct room impulse response rendering method*. However another method called *parametric room impulse response rendering* was favoured instead. This method actually synthesises RIR's in real time for a given position of the user within a VAE using geometric acoustic modelling methods. The reason this was chosen over the dynamic method was the fact that rendering the RIR for the users virtual position would produce a much more accurate RIR, as opposed to having to interpolate between a number of RIR positions. In addition to this, it is noted that the direct rendering method requires a lot of storage space. However, this method is much more complicated and limits itself to the accuracy of geometrical methods as wave-based methods are currently not able to run in real time (not even with using GPU technology, still taking approximately 10 minutes to render a single RIR [19]).

There are three reasons why using the direct rendering method in this project was initially considered:

1. Simplicity, as a real time RIR rendering system would not have to be produced.
2. The existing VSS system could be used as a starting point (thus adding to the simplicity).
3. The potential to use wave-based methods.

As using the parametric method required the RIR's to be calculated in real time, it would not be possible with current technology to produce a more accurate RIR than can be obtained using the geometrical acoustic modelling methods. However, as the direct RIR method produced the RIR's offline, it could be possible be produce more accurate RIR's in terms of frequency content. However, early in the project it was decided that using wave-based modelling methods would be much too time consuming to implement along with carrying out the primary project aims given 1) the time scale of the project and 2) the time it takes to produce such RIR's compared to the ones produced in Odeon. It was therefore decided that a system would be built using only RIR's produced in Odeon which would still allow the question of how many RIR's are required to produce such a system to be investigated.

In order for the user to move themselves around the VAE, the Max patch that the VSS was built on had to be extended in the following ways:

1. The production of a user interface that can be used remotely from within the spherical speaker array.

2. The extension of the Max patch that could accommodate the user interface and load the appropriate RIR files required to place the user in the desired location within the VAE.

3. The production of a system that can interpolate between the appropriate RIR positions.

Figure 6 shows an adapted flow diagram from figure 5 with the added user interface and automatic file searching functionality.

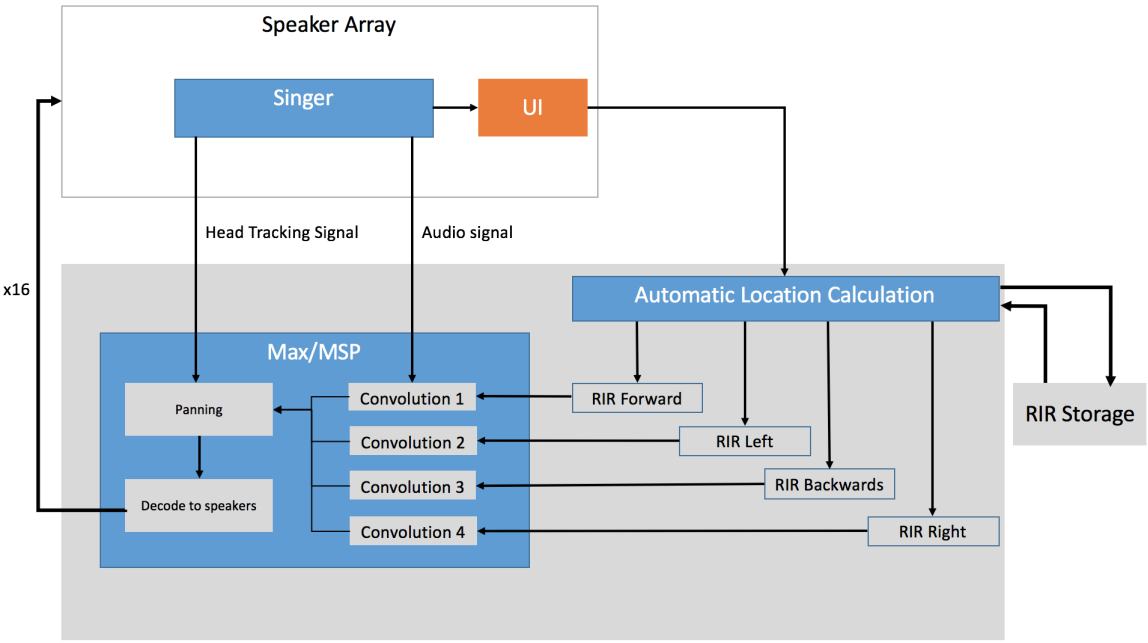


Figure 6: Flow diagram of the way in which the desired extended version VSS works

As the original implementation of the VSS uses real RIR's to simulate a VAE and the newly proposed implementation was to use synthetic RIR's, the difference in perception regarding how a user feels they are moving around the space was also to be investigated. This would provide an insight as to whether the newly implemented functionality of movement was worth sacrificing the accuracy of the real RIR's. For this to be investigated, real RIR measurements must be taken, therefore, the VAE to be modelled must be a building that was accessible for real RIR measurements.

To conclude, the project aims were:

1. To implement a direct RIR rendering system as an extension of the VSS
2. To investigate the number of RIR's that are required to convince a user that they can freely move themselves around a virtual space without limitation

3. To investigate the difference in the perception of mobility in the virtual space when using real RIR's and synthetic RIR's

3.5.3) Project Objectives

Given the project aims and the background provided, the following is a list of objectives that were set in order to complete this project:

1. Find an appropriate room to be modelled as a VAE
2. Digitally model the room using in Google SketchUp
3. Import room model into Odeon to finish material selection and RIR settings
4. Produce a grid of RIR's that can be used to interpolate between to simulate user positions
5. Record real RIR's in locations that can be compared to synthetic RIR's
6. Extend upon existing software patch to accommodate new functionality
7. Preform user tests:
    - Test #1: Does the perception of distance change when using real or synthetic RIR's?
    - Test #2: How far does the user have to move in the given VAE before they notice they have moved
    - Test #3: How many RIR's are required for interpolation for the user to feel they are moving around the space freely

To assess the success of this project, the following list of statements are defined with the intent of evaluating the extent of their correctness:

- Produce a system that:
    - Allows the user to move themselves around the VAE freely
    - Is beneficial to the user
    - Is simple to implement compared to other methods
- Obtain information regarding:
    - How many RIR's are required for convincing mobility
    - The effect of using synthetic RIR's on the perception of movement

3.6 - SOFTWARE

As a software patch was already in use with the VSS, the idea was to extend this software patch to accommodate the newly proposed functionality. This section will first give a quick overview of the original max patch and then an overview of the newly produced software with a simple explanation of how it works, followed by a more detailed explanation of how the two main parts of the software work.

The software described was run on a 2012 Mac mini with 16GB of RAM, an IntelCore i7 processor running OSX 10.10.4.

3.6.1) Software Overview

3.6.1.1 The Original Patch

The original max patch was used to convolve a real time audio signal with a set of four RIR's simultaneously, allowing the user to turn their head in the VAE through the use of an Oculus Rift as a head tracking device. Four positions within the VAE were available. To select one, the user (or an operator) selected an 'open' button which prompted a file navigation window. The RIR files then had to be found (in the correct order) and opened one at a time, with a new file window opening after each file had been selected. A screen shot of this process is shown in figure 7. This was the primary aspect of the original patch that needed extending to make the process automatic.

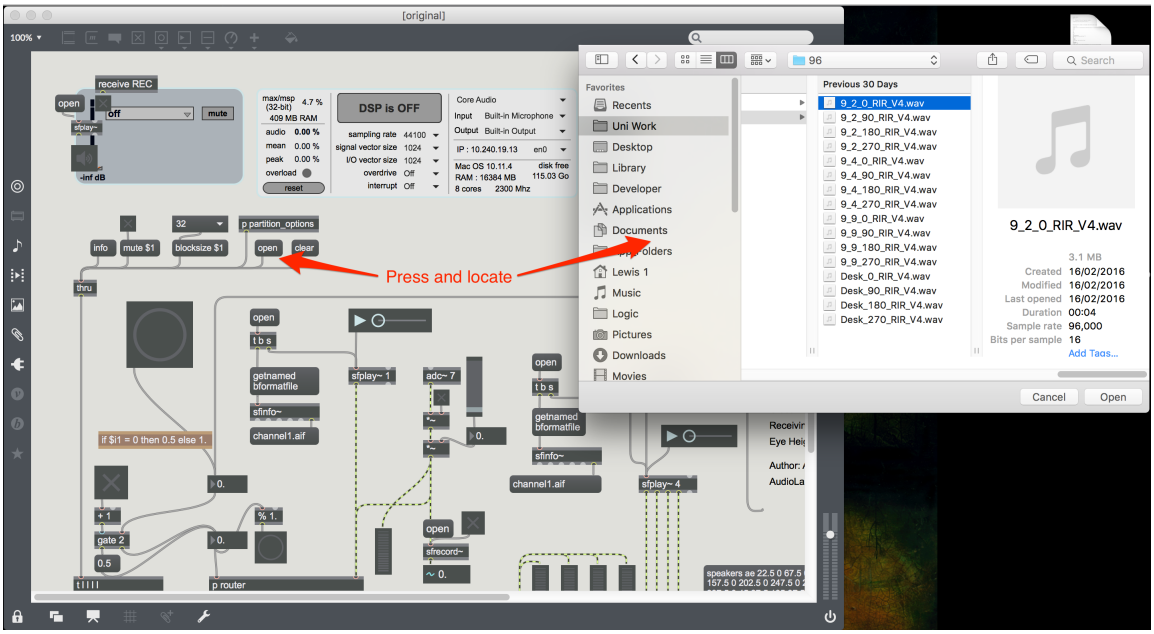


Figure 7: Original Max patch used in the VSS showing the file location window that pops up 4 times.

3.6.1.2 Extended Patch Overview

Figure 8 shows an annotated top level view of the Max patch produced to take a user input, load the appropriate RIR files and convolve with a real time audio input. The annotated sections can be described as follows:

1: Two buttons used to reset the system and start the timer used when moving around the space. The settings patch extends to provide a range of options including the density of

2: Here, an audio file can be loaded into the system and used instead of a real time audio input. This is used in user test #3.

3: A timer system used to synchronise the output of data transfer (from **4.2** to **4.3**), panning between convolved audio and the movement of the user interface feedback system (see section Location Tracking).

4.1: Patches that send a user interface to an iPad which allows a user to select a location within the VAE. User interaction is monitored (in the form of screen coordinates) and sent to **4.2**.

4.2: Takes the coordinates of the user input and calculates which (if any) RIR file should be loaded into the system, in section **4.3**.

4.3: Three patches (extended versions of the max explained in section The Original Patch) used to simultaneously convolve an audio signal (real time or audio file) with four directional RIR files. While one loads the next necessary RIR file the other two are used to simulate the movement of the user by panning the real time audio between the two currently running convolutions. This is how the user is moved along a path, explained in section Iteration 3 (Final).

5: An extended version of the real time head-tracking system used in the original patch, used to pan between the four directional RIR's to simulate head movement in the VAE.

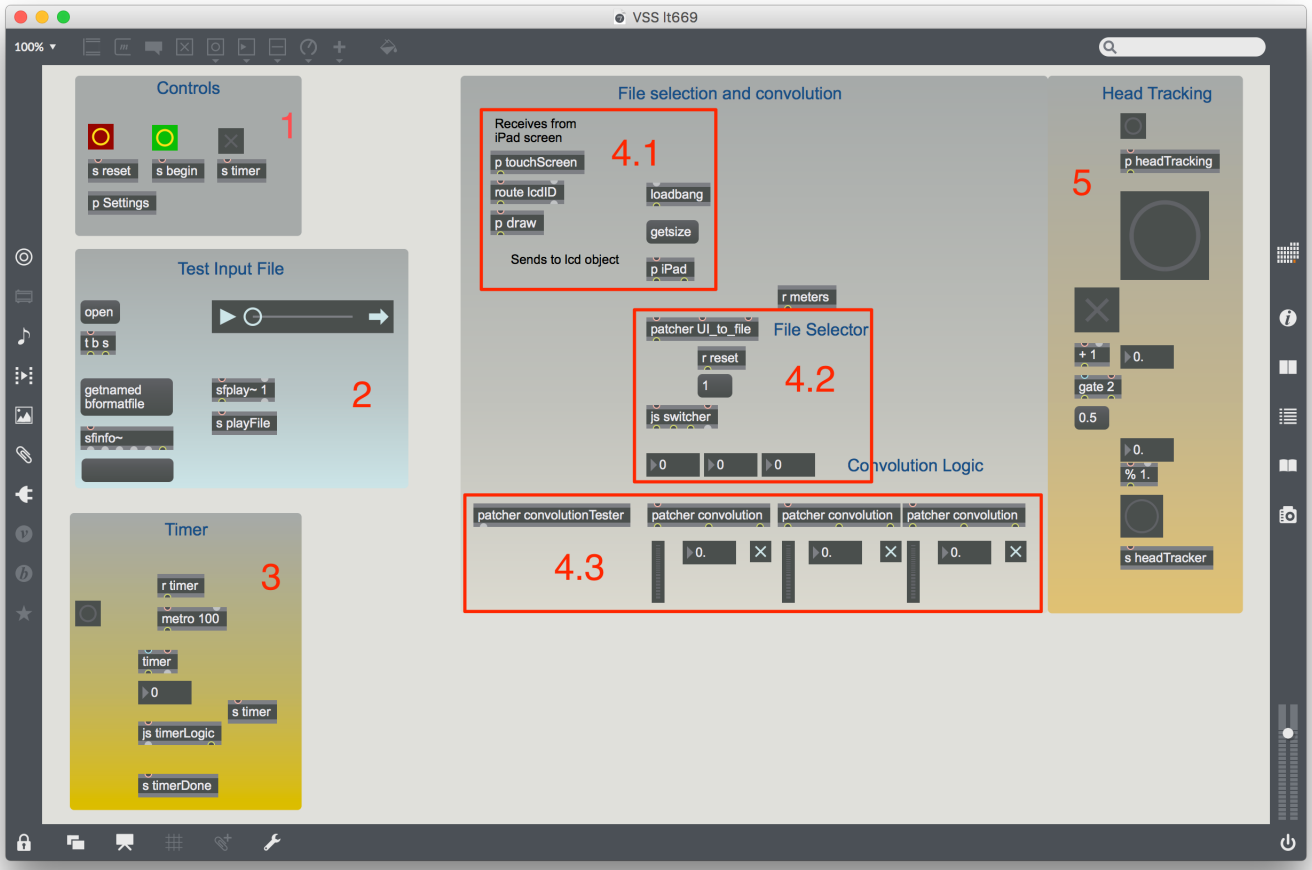


Figure 8: Top level Max patch

The software implementation was split into two main sections: **Location Selection** consisting of section **4.2** and **Mobility** mainly consisting of section **3 and 4.3**.

3.6.2) Location Selection

In order to allow the user to move themselves around the room, they had to be presented with a means of doing so. This involved presenting the user with an interface that resembled the space available to them on which they could select their location. This then had to output the coordinates selected by the user and convert them into a format which can be interpreted as a file name indicating which RIR in the available grid to use. This could then be passed to the rest of the system to load the appropriate files. A simple block diagram of the user interface part of the system is shown in figure 9.

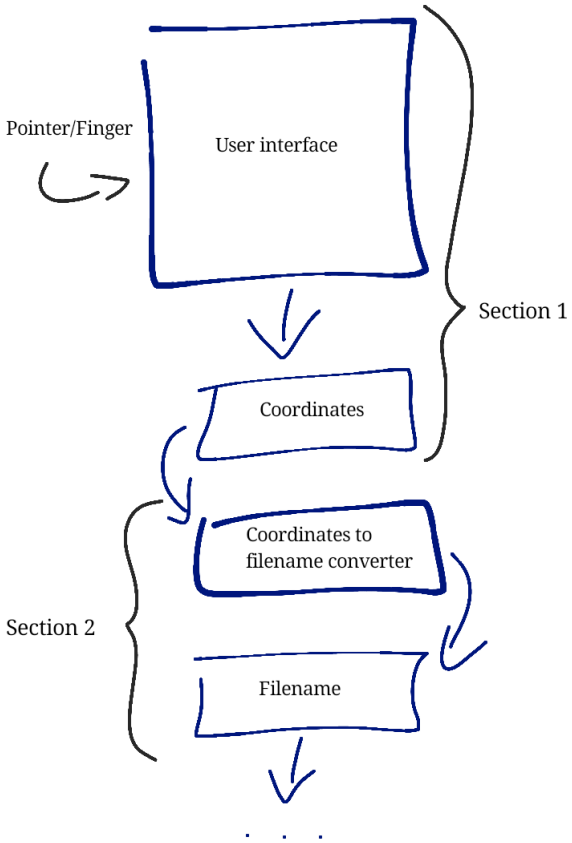


Figure 9: Flow diagram of the location selection software design. **Section 1** indicates the user interface section where coordinates are recorded and **Section 2** takes these coordinates and finds the appropriate RIR file.

In Max, the 'lcd' object is used for this function. This object presents a quadrilateral of variable length and height with the ability to output its dimensional information by sending a 'getSize' message to its input, as well as output the coordinates of a mouse click/drag. Figure 10 shows the lcd object with its inputs and outputs represented by section 1 in figure 9.

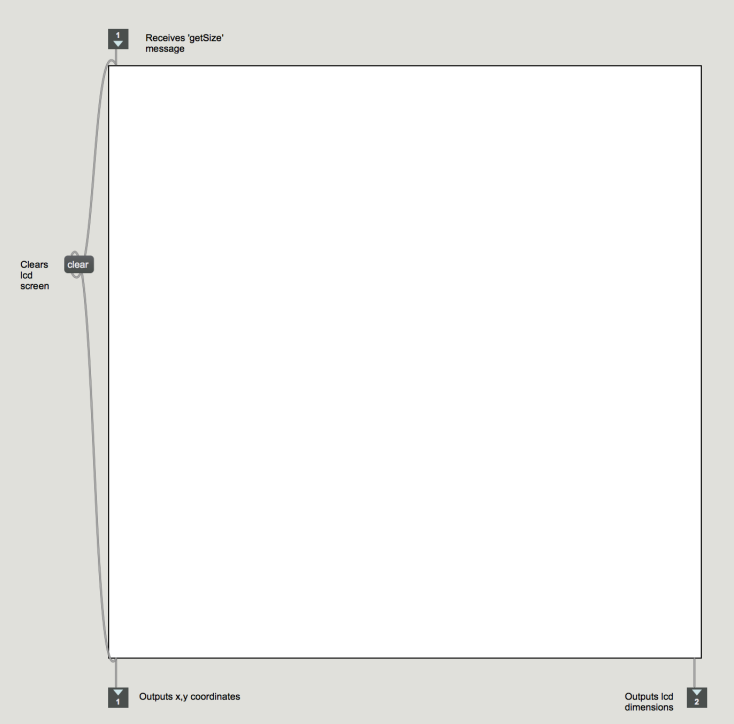


Figure 10: Flow diagram of the location selection software design

The outputs from the lcd object are sent to the patch 'UI_to_file' which contains a JavaScript file called 'loadFilesLogic'. This JavaScript file converts the (x,y) coordinates into an appropriate file name, by taking into account the size of the lcd screen and how many RIRs there are per meter.

This lcd screen could also be displayed on an iPad which was used as a remote user interaction. Initially, the more popular and supported application **Mira** [20] was going to be used. This allows for the mirroring of a selected portion of a Max patch to be displayed and interacted with on an iPad. However, the lcd object used to track coordinates was not supported by this application and thus did not appear on the iPad. Instead an alternative was found. Cycling74, the same company that produces Max/MSP and Mira also created an app for the iPhone and iPad called c74 [21] and allows for the creation of an independent interface that can control objects within the max patch. The operation of this section is explained further in section Location Tracking

The javascript takes 5 inputs: UI (x,y) coordinates , (x,y) lcd dimensions and a number representing how many rows and colums of RIR locations there should be available. This last value is calculated by sending a number from 1-5 (distance per RIR) to the third inlet of the UI_to_file patch (on the far right). As the maximum number of RIRs that will be available per length of the room is 15, 15 is divided by the input and rounded to the nearest integer, giving the number of rows and columns the lcd screen should be split into (the exact number of rows and columns is calculated later in the javascript file). This information can then be used to determine in which *'section'* (row and column) the user is currently located based on their coordinates, allowing it to load the appropriate RIRs that are available in that section. Figure 11 shows the section of

'UI_to_file' highlighting each section.

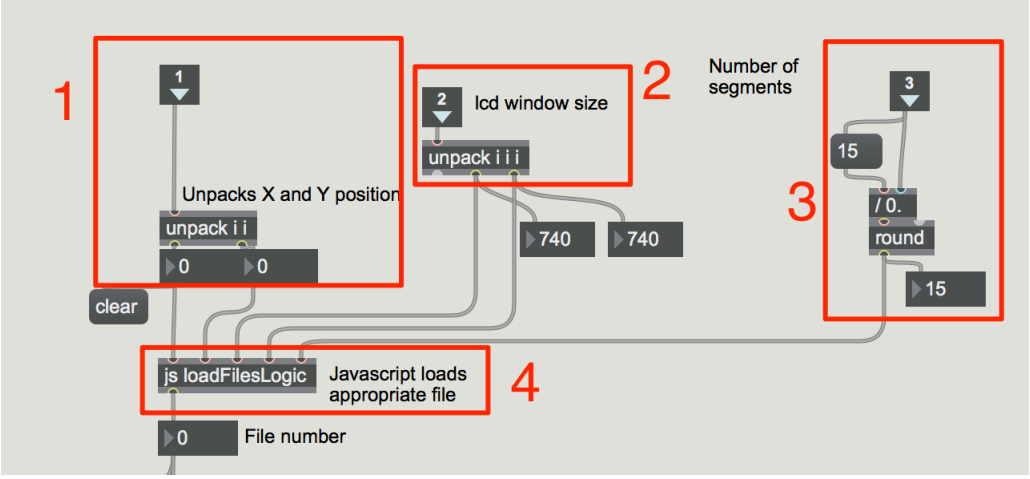


Figure 11: Screen shot from Max showing a section that converts the user interface coordinates into the appropriate file name. 1) Coordinates from lcd object 2) Dimensions of lcd object 3) Number of segments the lcd object should be split into due to the number of RIR's per meter 4) JavaScript file that produces an appropriate file name given the input data.

3.6.2.1 File name JavaScript: 'loadFilesLogic.js'

```
function msg_int(input){
 if(inlet == 0){
  xPos = input;
 } else if (inlet == 1){
  yPos = input;//Add off set to start at (0,1)
 } else if (inlet == 2){
  windowSize [0] = input;
 } else if (inlet == 3){
  windowSize [1] = input;
 } else if(inlet==4){
  numberOfMeters = input;
 }

```

The first section in the JavaScript simply stores the data from different inputs to different variables that are used throughout the rest of the code, where:

xPos	= user x coordinate
yPos	= user y coordinate
windowSize[0]	= lcd length
windowSize[1]	= lcd height
numberOfMeters	= Integer to calculate rows/columns

As can be seen in figures **??**,**??**,**??** and **??** in **??**, there are not always the same number of rows as there are columns in each of the grids. The if else statements in figure 12 calculate the users positions by taking the users x and y coordinates and dividing by the x and y dimensions of the lcd screen, giving a percentage of how far across the room they are. This allows the for lcd screen to be resized allowing it to be used of different sized screens in the future (see section **??**). This value is then multiplies by either the number of rows (x axis) or columns (y axis) there are, which is calculated by adding or subtracting 1, or using the value of numberOfRows which is calculated

before being input into the js object. The following table indicates how many rows and columns there are given the inputs to the js object:

Distance between RIR's (m)	Calculation	numberOfMeters	Rows	Columns
1	15/1	15	15	16
2	15/2	8	7	8
3	15/3	5	5	5
4	15/4	4	3	4
5	15/5	3	3	3

Lines 17 and 18, the calculated positions are rounded to the nearest integer in order to determine which RIR location the user is closest too. Lines 21 to 26, create an initial offset to prevent the lcd screen from starting at location (0,0), as this is not a physical point in the room itself thus preventing errors from occurring when searching for the appropriate RIR file.

```
1   //Split into sections
2   if(numberOfMeters == 3 || numberOfMeters == 5){
3    //Even grid for 3m and 5m
4    xPosition = (xPos/windowSize[0])*(numberOfMeters);
5    yPosition = (yPos/windowSize[1])*(numberOfMeters);
6   } else if (numberOfMeters == 4 || numberOfMeters == 8){
7    //4m separation requires different x,y coordinate scaling
8    xPosition = (xPos/windowSize[0])*(numberOfMeters-1);
9    yPosition = (yPos/windowSize[1])*(numberOfMeters);
10  } else{
11   //Extra row for others
12   xPosition = (xPos/windowSize[0])*(numberOfMeters);
13   yPosition = (yPos/windowSize[1])*(numberOfMeters+1);
14  }
15
16  //Round to nearest value
17  xSection = Math.round(xPosition);
18  ySection = Math.round(yPosition);
19
20  //Start the lcd grid sections from column 1 row 1 instead of column 0 row 0
21  if(xSection == 0){
22   xSection = 1;
23  }
24  if(ySection == 0){
25   ySection = 1;
26  }
27
```

Figure 12: Code used to calculate user location

The section the user is currently located in is then saved to the first address in a array and can be used to compare against their previous position to check whether a new file needs to be loaded.

This is necessary because each time the user touched the user interface, thus changing the x and y coordinates, javascript file is run. However, if there new location is still within the same 'section' then a new RIR file does not need to be loaded.

In figure 13, lines 6 to 26 contain one large if statement. This essentially prevents the program from calculating, searching for and trying to load a file if the same one is still in use, thus saving computation time. This is done by comparing the section of the grid the user is currently in with the previous section they were currently in and if it has changed the file name for the new location should be calculated and searched for, otherwise no new file should be calculated.

If the location of the user has changed, two more conditions are checked in lines 8 and 14. This ensures that only the section that has changed is updated, ie, if the user has moved to the left, only the X axis section is updated.

Finally, the appropriate file name is calculated. Two different algorithms are used depending on which RIR grid is being used. Both essentially take the section the user is located on the X axis, and add the number of locations there are due to how many rows down the room they are located.

This then outputs a number from 1 to the maximum number of locations in the selected grid.

```
1     //Store current location
2     xArray[0] = xSection;
3     yArray[0] = ySection;
4
5     //If either coordinate is changed search for new files
6     if(xArray[0] != xArray[1] || yArray[0] != yArray[1]){
7
8      if(xArray[0] != xArray[1]){
9       //Store previous value
10      xArray[1] = xArray[0];
11      X = xArray[0];
12     }
13
14     if(yArray[0] != yArray[1]){
15      yArray[1] = yArray[0];
16      Y = yArray[0]
17     }
18
19     //Output user location within grid
20     if(numberOfMeters == 4 || numberOfMeters == 8){
21      fileNumber = X + ((numberOfMeters-1)*(Y-1)); //Requires different algorithm for 2
    m and 4m due to different grid shapes
22     } else {
23      fileNumber = X + ((numberOfMeters)*(Y-1));
24     }
25     outlet(1,fileNumber);
26    }
27
```

Figure 13: Code used to search for appropriate file name

3.6.3) Mobility Implementation

The following sections describe two earlier iterations of the systems that were attempted and the issues faced, eventually leading to the third and final iteration which can been seen as a compromise between system speed and desired functionality. All three iterations are extensions of the original patch used in the VSS which use spat to load RIR files into the system, and to convolve a real time audio input with them.

3.6.3.1 Iteration 1

Initially, the idea was to pre-load a grid of the closest RIR locations that surrounded the user and simultaneously convolve the real-time audio with all of the RIR files. This is illustrated on the left in figure 14 which shows annotated grid positions on the lcd screen labeled 1 - 9. The patch on the right shows the corresponding volume bars (also labelled 1 - 9) used to automatically vary the

output level of each of the convolved signals. This required a panning algorithm for each of the positions in the defined grid, an overview of which is shown in figure **??** in **??**, based on where the user has moved relative to the centre position essentially allowing the user to move freely.

When the user reaches on of the other locations in the grid, that location becomes the centre of a new grid, and the appropriate files are loaded around that centre positions through the use of a javascript file. This requires the system to simultaneously load 4 RIR files per location, of which there are 9, meaning 36 files are loaded into the system at once. Due to the time taken to load all of the files, the system ran too slow and could not be used for real time movement.

A video demonstrating the functionality of the proposed system can be found link **HERE LINK!: Writeup/videios/iteration1Panning**.

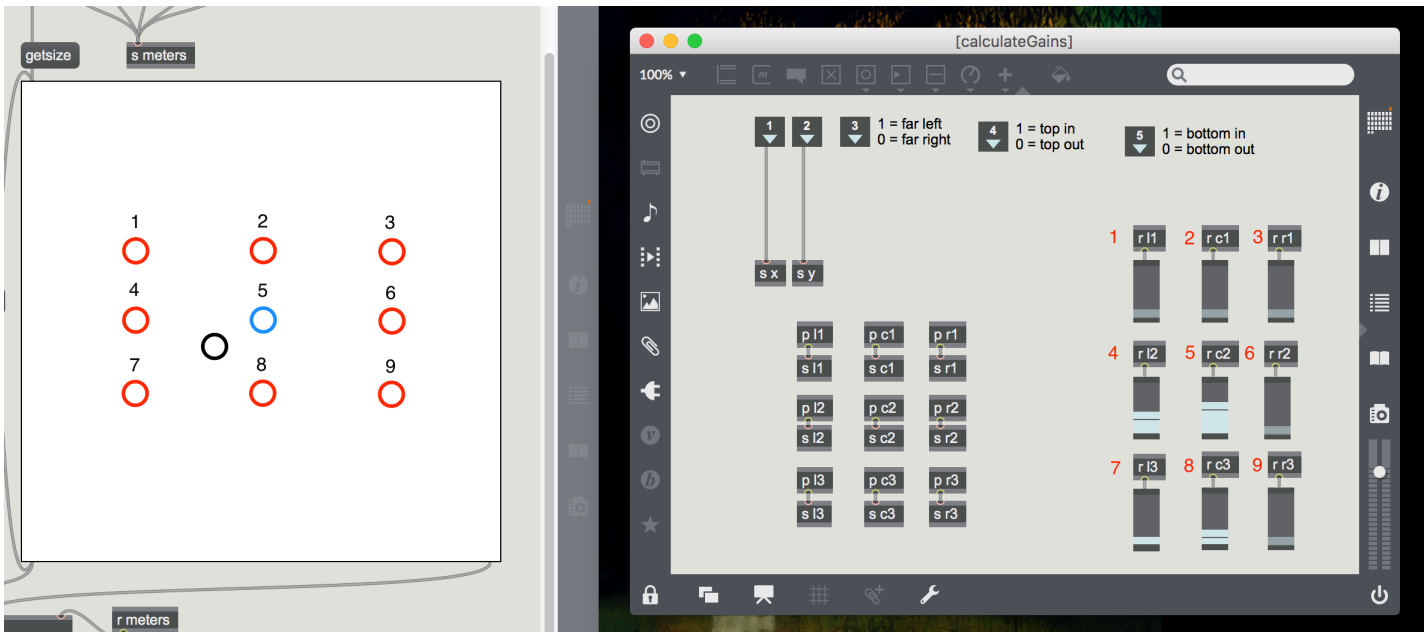


Figure 14

3.6.3.2 Iteration 2

In an attempt to maintain the current implementation and solve the file loading time issue, a patch was built to pre-load all the RIR files, meaning the files only had to be associated with a grid position as opposed to loaded every time a new position was used. This involved running a convolution patch for each file loaded into the system. As the files would no longer need to be loaded into the system, the user interface section described in section Location Selection is used to decide which of the convolution patched to bypass and mute, and which ones to run. This means that only the grid of RIR's being used would be convolved with the audio signal, saving computational resources. Figure 15 shows part of the patch used to pre-load all the RIR files.

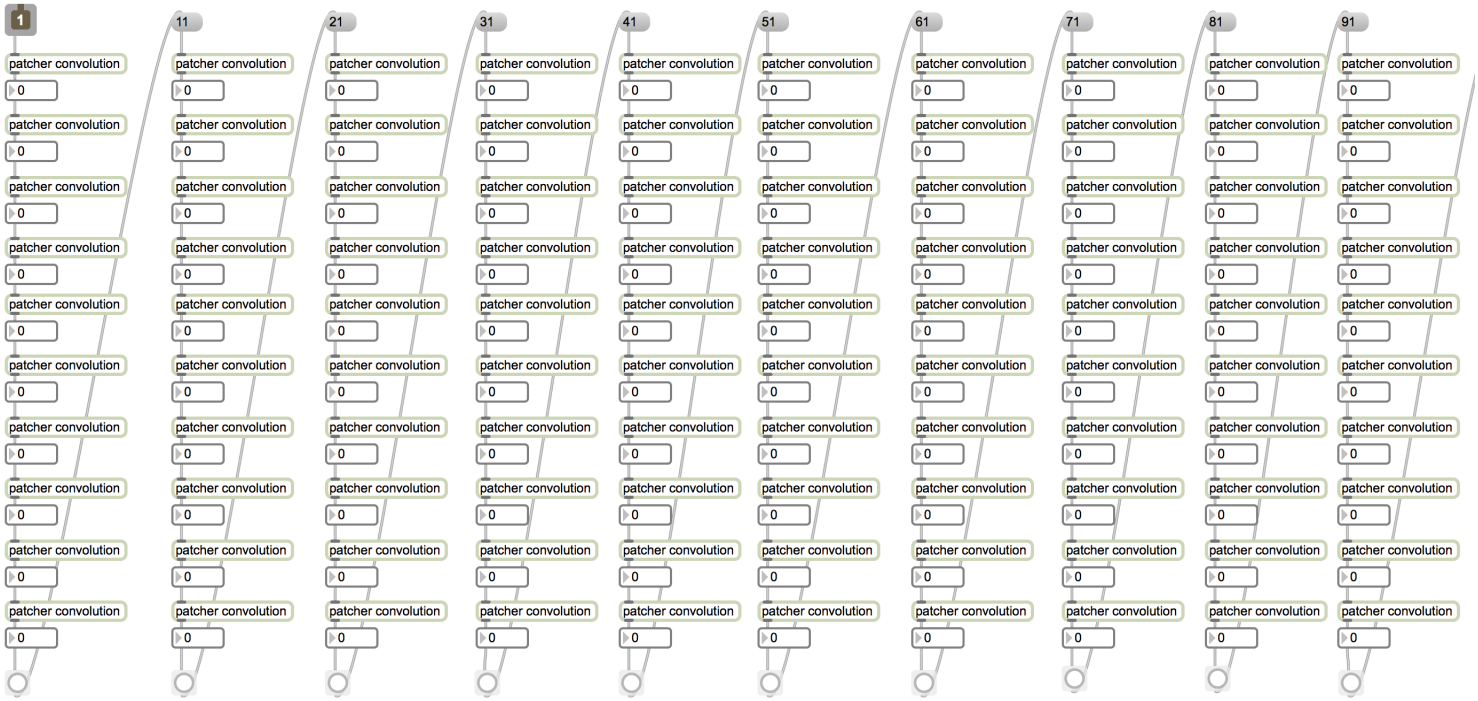


Figure 15: Chained convolution patches used to pre-load each RIR file. *Image taken from Max 6 as the version of this patch would not run in Max 7*

Though it was possible to load smaller grids, such as a 9x9 grid used when locations are separated by 5m, the larger grids caused the system to run too slowly due to the amount of RAM that was used.

3.6.3.3 Iteration 3 (Final)

The previous iterations attempted to allow the user to move anywhere within the VAE in real time as they moved their finger around the screen, however due to technological implications, this was not possible. It was therefore decided to find a compromise, where the user could draw themselves a path around the space and the system would move them along the path, one RIR location at a time at a rate determined by a timer, allowing the system the load all appropriate files with ample time to prevent the system from lagging.

The top half of figure 16 has been seen already in figure 11, however the bottom of figure 16 shows how the resulting file number is used. It gets sent to a javascript file that adds the file number to the end of a mutable array (an array that can be extended or truncated once created). As the file names are being stored, the corresponding files can be loaded when the system is ready, instead of trying to load them instantly as in Iteration 1. Once the js object receivers a message from the timer (highlighted as number 3 in figure 8), the number at the start of the array is sent to the output. The array is then truncated by 1, moving the next file number to the front of the array waiting for the next timer message. If a reset message is sent to the third inlet of this js object (by

pressing the first button in section 1 shown in figure 8), the arrays are cleared enabling the user to start drawing another path if the previous path had not been completely travelled.

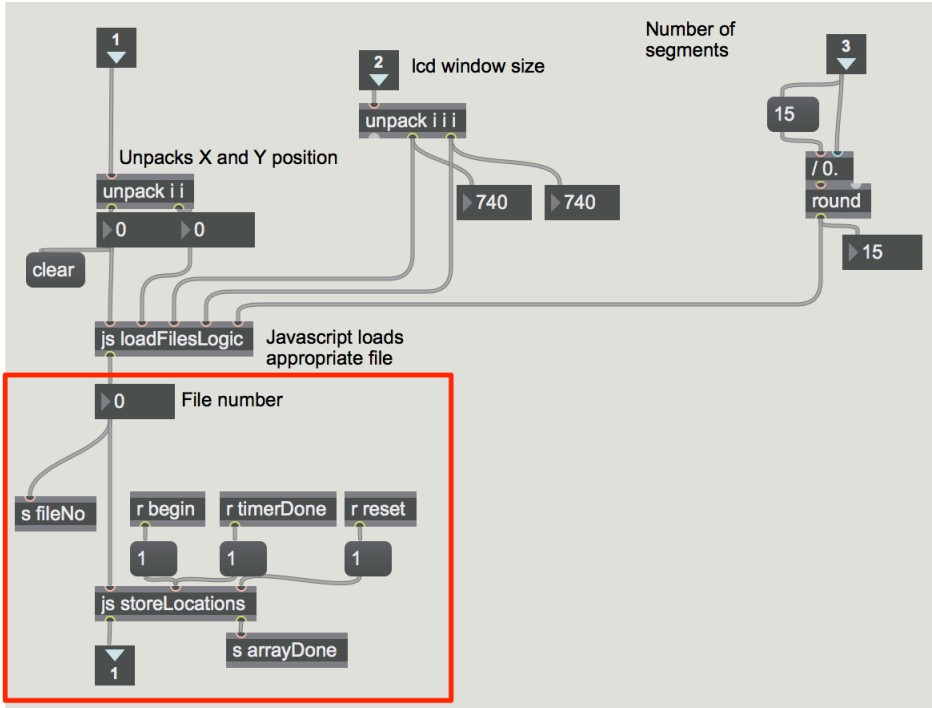


Figure 16: Highlighting the section of 'UI_to_file' that stores the file numbers and outputs them when a timer is done.

Figure 17 shows the output from the 'UI_to_file' patch in the red rectangle (the patch that contains the contents of figure 16) running into a new javascript called 'js switcher' in the blue rectangle. This simple script ensures that each new file number is sent to a different convolution patch (shown in the yellow rectangle). This means that while a new file is being loaded into one of the convolution patches, the other two can be used to pan between two pre-loaded positions. This is illustrated in figure 18

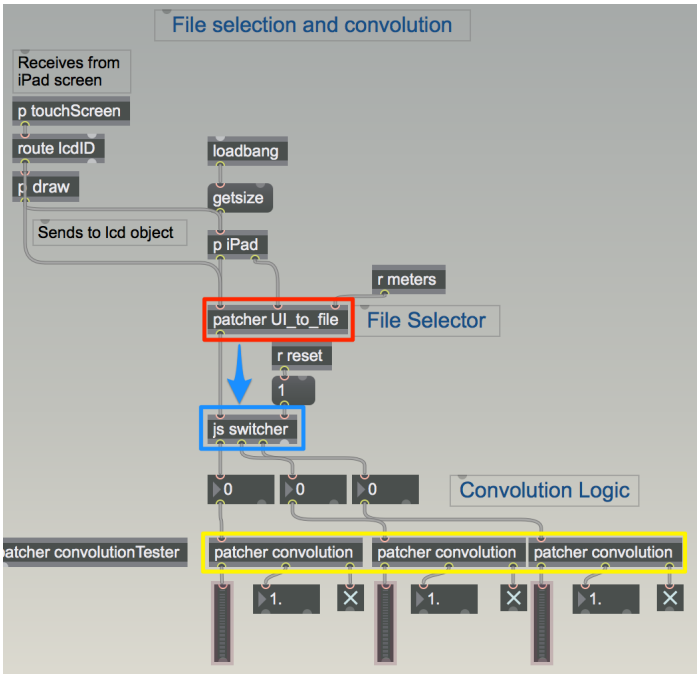


Figure 17: **Red:** Outputs file stored file number. **Blue:** Distributed file number to a different convolution patch each time. **Yellow:** Receives file number and loades appropriate file into the system.

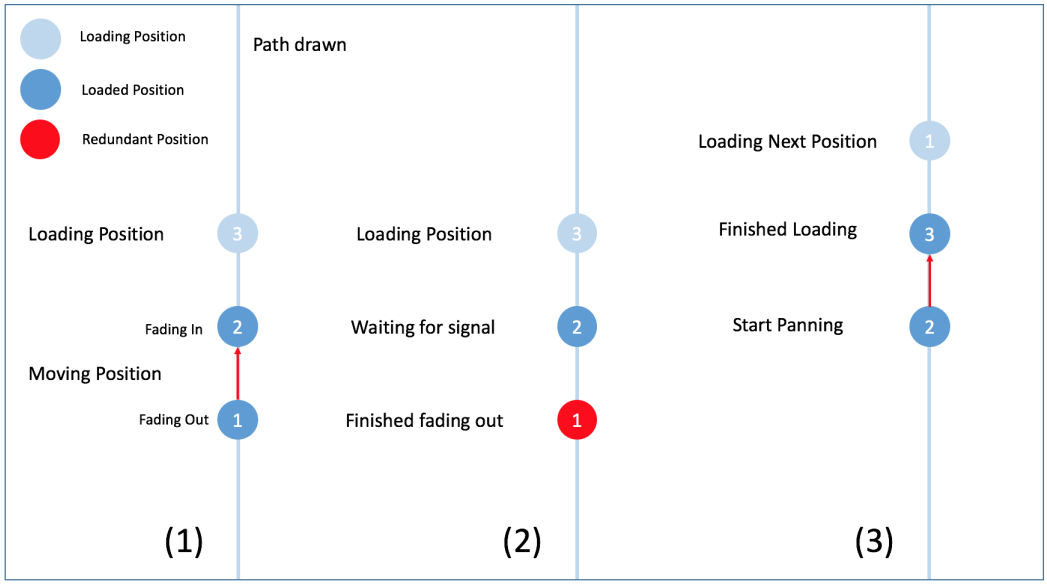


Figure 18: Illustration of how the three convolution patches work together to simulate movement, where the circles represent locations within the VAE and are numbered to indicate which convolution patch they are loaded in. **(1)** User is moved between two loaded positions while the third patch loads a new RIR fie. **(2)** The user is held in the new position until the next position has been loaded into the system. **(3)** User is moved between the two available positions while the next position is loaded into the system.

The convolution patches have three outputs connecting to:
1: A level meter used to see which patch is being used.
2: A number box to check the level of each patch.
3: A toggle box used to see which spat convolution algorithms have been bypassed (a function that was later removed, explained in section Software Issues) These were used to monitor the functionality of the patch while testing.

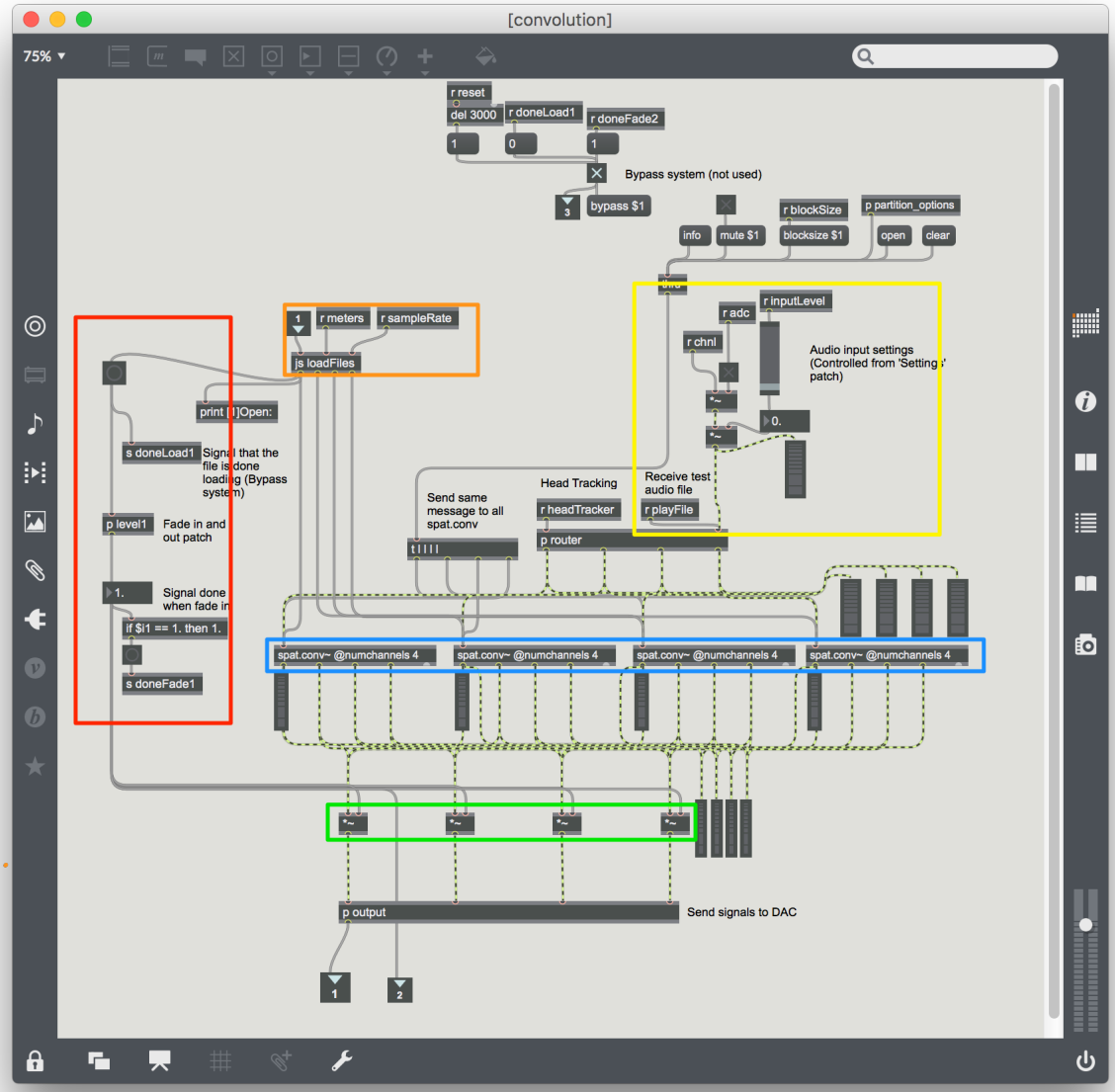


Figure 19

To achieve the path following functionality, each convolution patch contains its own level control algorithm. Figure 19 shows an overview of the first convolution patch which is a modified version of the original pathch used in the VSS. The input to the convolution patch (orange) feeds the file number directly into a 'loadFiles.js' javascript object. This simply prepends the number with the appropriate number of 0's (eg 1 becomes 001 and 28 becomes 028), then this number is concatenated with a file path pointing to where the audio files are located (see js file: loadFiles.js **LINK**). This outputs four 'open' messages (one for each directional RIR file) followed by a file paths to the spat.conv objects (blue). These are the objects that actually search for the file and load it into the system. By sending the prepending 'open' message, manually searching for a file is

not required, thus automating the process. These objects convolve the loaded RIR file with whatever audio input is given at its inlet, in this case the real time audio input or audio file (yellow). These outputs are then sent through multipliers (green), used to fade the signal in and out with an automated volume control (red).

The automated volume controls receives a 'bang' (signalling something has happened) when the 'open' message is sent form the 'loadFiles' js object (orange). This bang is also sent to the convolution patch that is currently at full volume. This prompts the volume of the current patch to increase while the volume of the previous patch decreases, thus panning between the two signal.

3.6.4) Location Tracking

Figure 20 shows the patch 'touchScreen' which contains two main elements. The left side is used to draw a dot on the user interface that indicated to the user where they currently are in the virtual space and the right side is used to draw the user interface. Both are described in the following sections.

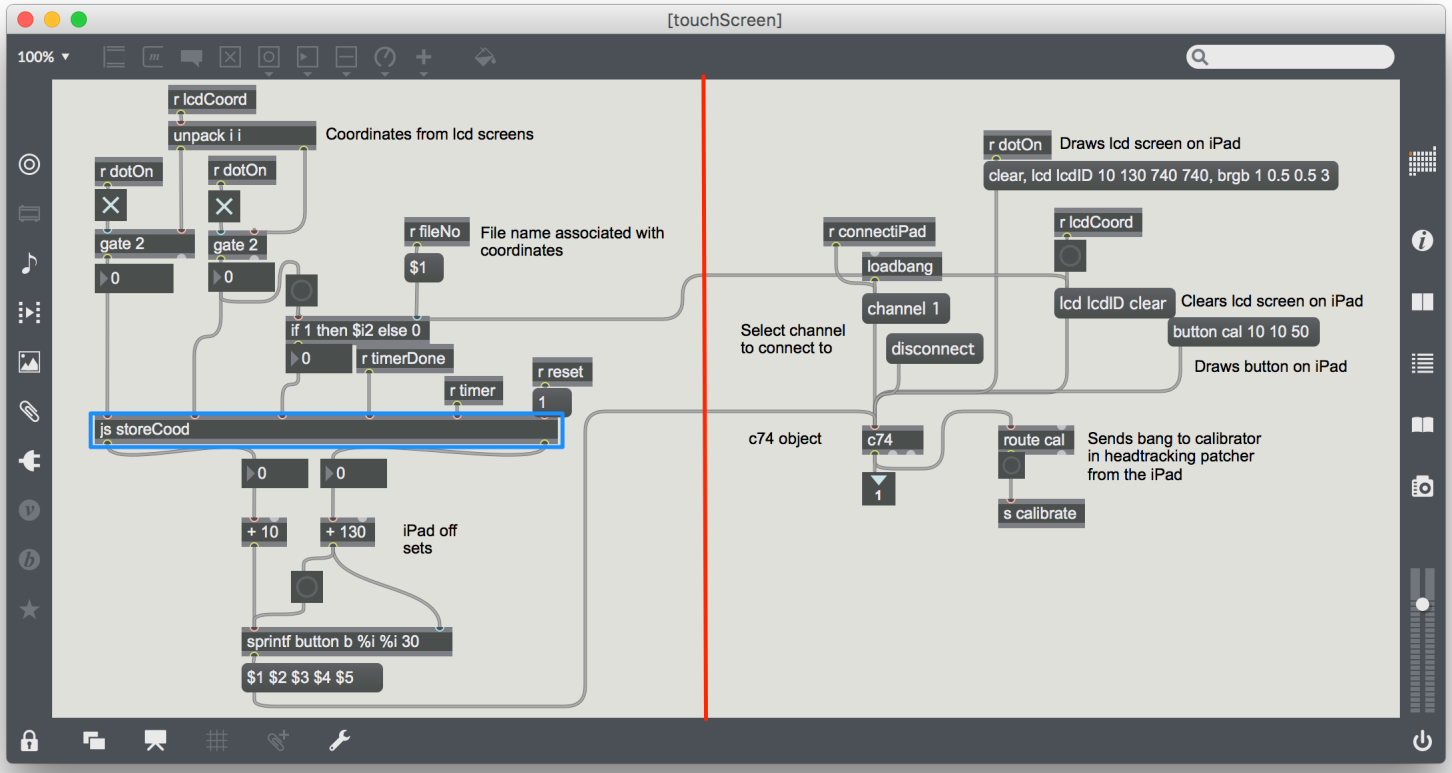


Figure 20: Screen shot of 'touchScreen', the patch used to produce the user interface on the iPad (right side) and to draw a dot on the iPad showing the user where they are located in the virtual space (left side). 'storeCood' javascript object highlighted in blue.

3.6.4.1 User Interface

As previously mentioned, the c74 application was used to display a user interface on an iPad, allowing the user to select their location within the space. This is created on the right side of figure 20, by sending the 'c74' object a message that contains the type of object that should be presented, its coordinates in the space as well as size and colour, the results of which are shown in figure 21. The output of the 'c74' object returns information regarding the objects created in the user interface. In this case, the positions touched on the lcd screen will be sent from the iPad back to this 'c74' object. This information is sent to the input of the 'UI_to_file' patch. This allows the iPad screen to be used the same way as the lcd screen in Max is used (show previously in figure 10).

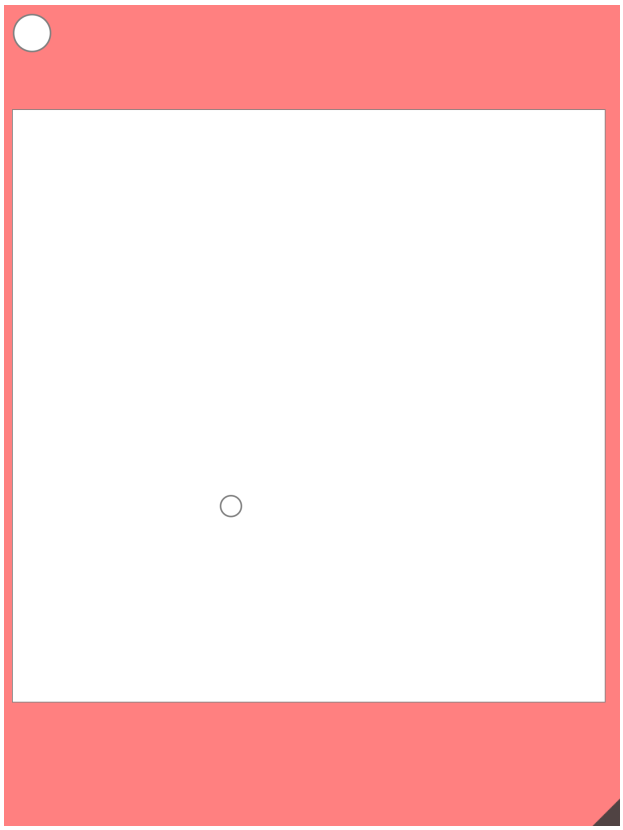
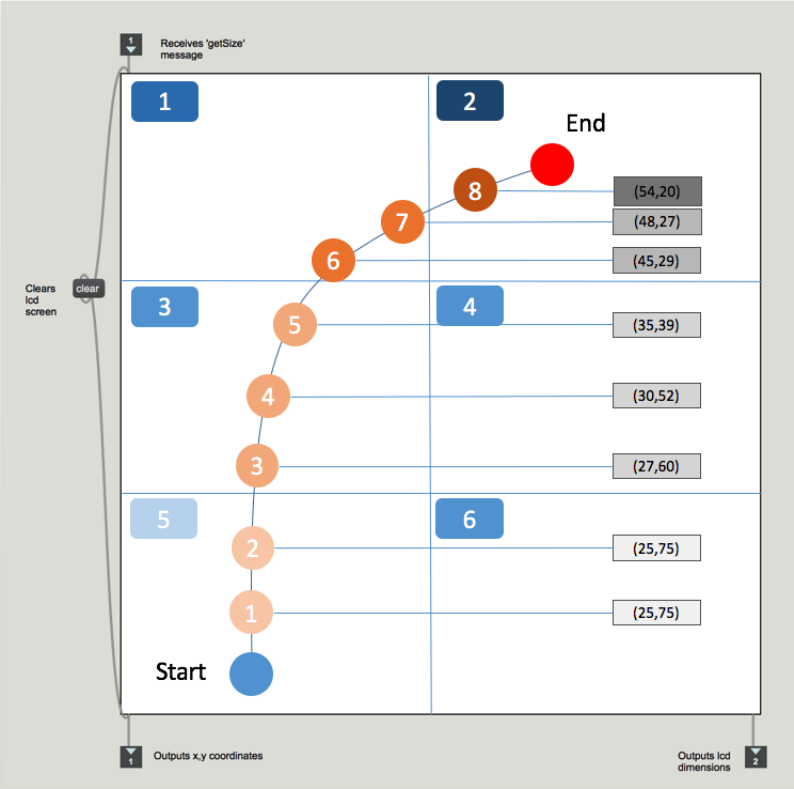


Figure 21: Screen shot of the user interface produced on the iPad. The large white rectangle represents the virtual space in which the user can place themselves, where the top of the rectangle is the front of the room (where the blackboards are in Hendrix Hall). The dot in the white rectangle is used to show the user where they are located in the space and the button in the top left corner is used for calibrating the head-tracking device.

3.6.4.2 Position Feedback

The left side of figure 20 contains a javascript file called 'storeCood' **Available here: LINK** (highlighted in blue) with a number of inputs. As mentioned above, the 'c74' objects can retrieve data from the lcd screen on the iPad. When the user selects a location, the (x,y) coordinates are sent to the first two inputs of the javascript file respectively. These coordinates are stored in a list along with the corresponding section that they are located in (third input), calculated using the method described in section 3.6.2 Location Selection. Figure 22 shows a simple example, where the coordinates of a path spanning across 4 sections are stored in groups.

Input 4 and 5 receive two different signals routed from the main timer (labelled 3 in figure 8). Input 4 receives a signal every time the timer is done (ever 2.5 seconds) and input 5 receives a message every 100ms (the rate at which the timer increments). These are used to call the two main functions that are used to output the coordinates at the correct times, `findLength()` and `outputCoords()`. Each time the timer finishes (after 2.5 seconds), `findLength()` starts from the beginning of the list of stored coordinates and scans through them, calculating how many coordinates are located in the same section. For the example in figure 22, this would find a length of 2 for positions 1 and 2 located in section 5. As the user is moved to a new RIR location every 2.5 seconds, `outputCoords()` will output the coordinates that were travelled in that section evenly within that amount of time. This is done by dividing 2500(ms) by the number of points within that section. For the example in figure 22, this would be $2500/2 = 1250$. Therefore, every 100ms the function checks to see whether it is time to output the stored coordinates, meaning the coordinates of position 1 and 2 will be output at times 1250ms and 2500ms. If we were to continue with the example, the next three positions, 3, 4 and 5 would be output at times 833ms, 1666ms and 2500ms, thus evenly spreading out the movement of the dot on the user interface screen *meaning the dot should travel around the path at a continuous rate instead of in staggered increments*. An example of the dot movement can be found **LINK TO DOT MOVEMENT VIDEO**.



Point	Section	(x,y)
1	5	(25,80)
2	5	(25,75)
3	3	(27,60)
4	3	(30,52)
5	3	(35,39)
6	1	(45,29)
7	1	(48,27)
8	2	(54,20)

Figure 22: Simple example of the coordinates along a path being stored in groups depending on which section of the lcd screen they are located. *(The smallest number of sections possible is actually 9, but 6 are used here for simplicity)*

3.6.5) Head-Tracking

The original VSS patch used an Oculus Rift as a head-tracking device which was also used to provide visuals of the VAE. As this project does not provide visuals, a smaller, less obtrusive device was sought. Initially the YEI 3-Space Sensor [22] was investigated, a small device that can track angular rotation and output results into Max. It was found however that the software used to interface between the device and the computer (a Mac) was only in a Beta stage. As a consequence, the output from the sensor did not function correctly or provide useful information.

Instead, the c74 application already being used for the iPad user interface (see section 3.6.2 Location Selection) was used to extract rotational data from an iPhone. The iPhone data retrieved was accurate and provided a steady wireless connection, as opposed to the USB connection required for the YEI sensor. For user testing, a low-tech head-mounting solution was found by wrapping the iPhone in the front of a cotton hat. This was integrated with the existing head-tracking by creating a new patch that reads data from the iPhone (much like the patch used for the iPad) and maps the data to an angle from 0° to 360°, including a calibration system ensuring that each user is facing the same way in the VAE. The output of this was sent to the input of the existing head-tracking system where the Oculus rift data would have been sent. This data is used to pan

between the four direction RIR files simulating the effect of turning the the VAE.

3.6.6) Settings Menu

Figure 23 shows the settings patch 'settings'. This can be used to determine which RIR grid to use by selecting the number 1-5 on the top left of the patch (green buttons) as well as the sample rate of the RIR files. The audio input settings can be used to change which audio input is used. The panning settings determine the speed at which the user is moved between RIR locations.

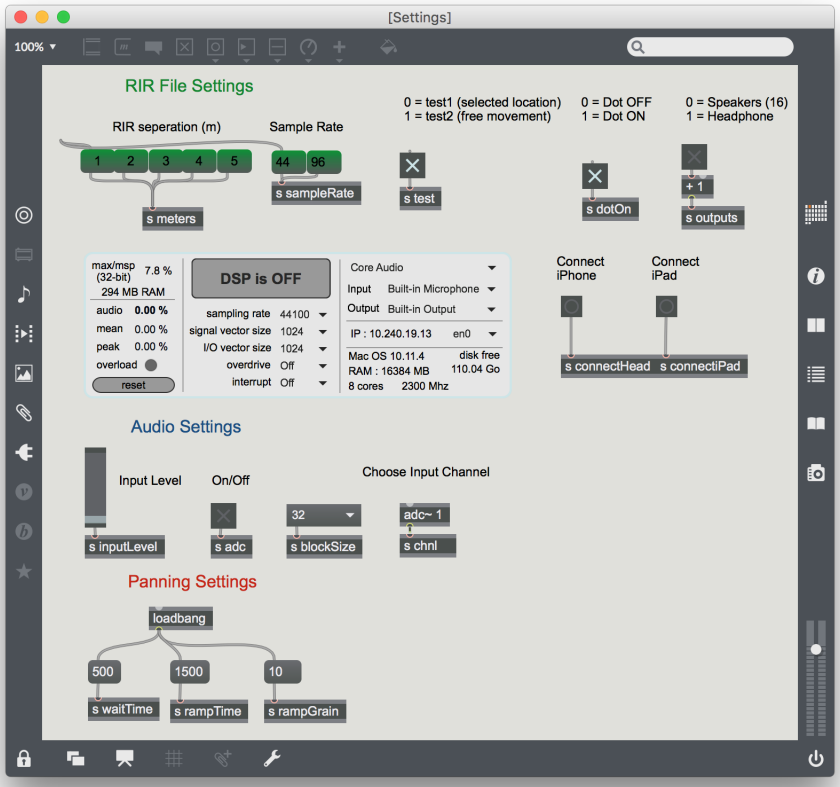


Figure 23: Settings patch used to control parts of the overall patch from one location

3.6.7) Software Issues

One issue raised when implementing iteration 3 was the way in which the user is moved through the virtual space. As only two RIR locations are used at a time as opposed to using 4 RIR locations for interpolation, the user is moved in a 'zigzag' pattern. Figure 24 shows two images illustrating how the user is moved through the virtual space using the final iteration of the software against the initial two iterations, where the curved blue line represents the path drawn by the user. The image on the right shows how the 4 RIR locations are used together to make it sound like the user

is placed between them. The image on the right shows how the user is passed between locations one at a time, causing the 'zigzag' pattern.

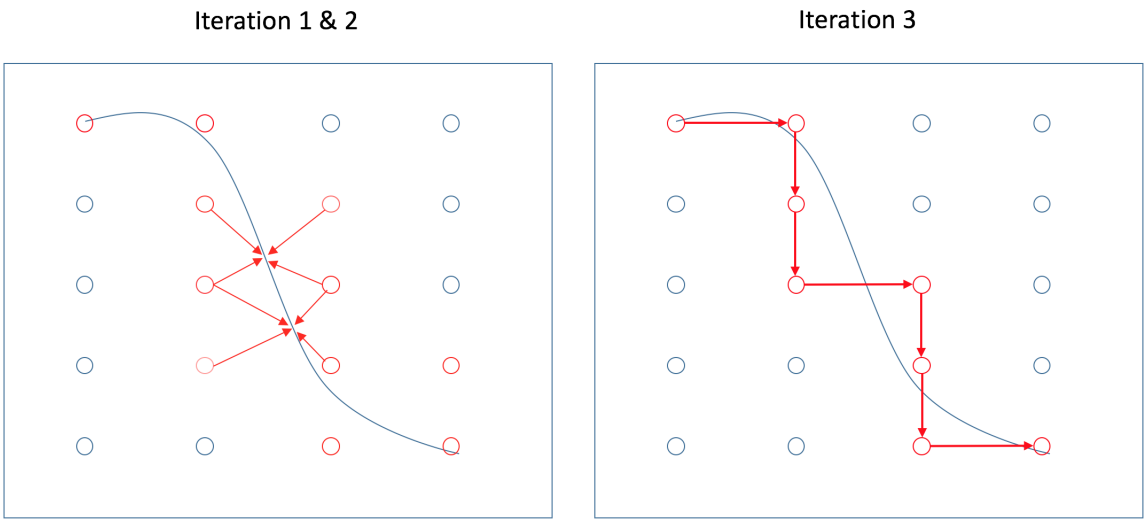


Figure 24: Illustration of how the user is moved through the virtual space using different numbers of RIR locations where the image on the **left** shows how 4 RIR's are used at a time and the image on the **right** shows the user being passed between two RIR locations at a time. The blue line represents the paths drawn by the user and the circles represent RIR locations.

As previously explained in section 3.6.3.3 Iteration 3 (Final), three convolution patches were used to move the user between two RIR locations while the third loads the next location. In an attempt to make the system less computationally expensive, if any of the convolution patched were loading a file as opposed to convolving an audio signal with one, the convolution algorithm was bypassed. However it was found that any delay in turning the bypass on or off produced a 'popping' noise. Small delays in turning the bypass on and off often occurred, therefore this bypass functionality was removed.

3.7 - Latency

The time taken for the input signal to run through the system causes the room reflection to arrive at the user at the incorrect times. To determine the length of the delay, a latency test was conducted. The test procedure was similar to that in the production of the original VSS [1] with some slight differences. Two Earthwork M30 reference microphones [23] were placed in the centre of the speaker array. One was connected to the input of the max patch and the other was connected to a Mac running reaper through a MOTU audio interface [24] to record a reference input. An output channel running to one of the Genelec speakers was routed to the second input of the MOTU interface. This was used to record the signal after it had run through the Max patch, thus allowing the two signals to be compared to find the latency time. Both systems were running at a

sample rate of 44.1kHz. To identify the impulse clearly, a pair of drumsticks were used to produce an impulse. This was then convolved with a 4-channel Dirac impulse (1 sample of amplitude 1) of approximately 3 seconds long. Figure 25 shows a diagram of the latency measurement process.

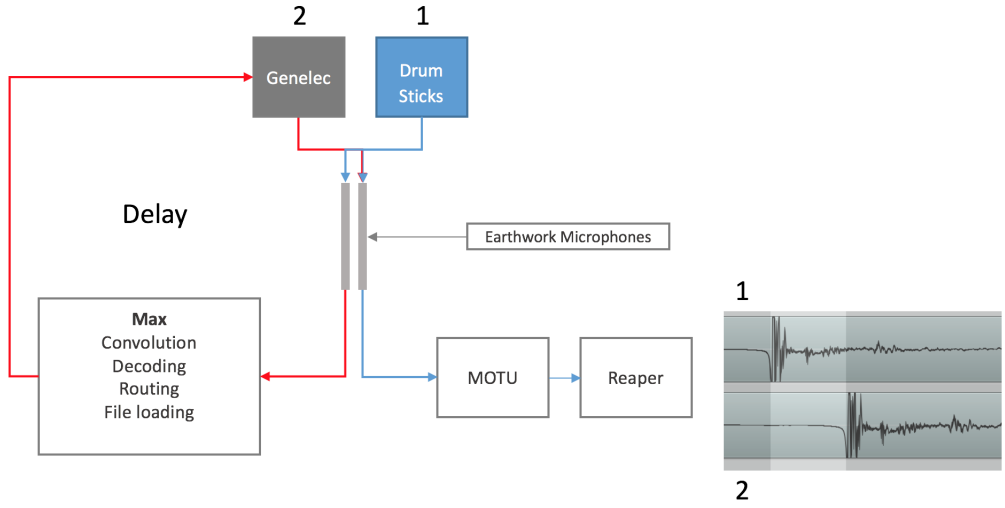


Figure 25: Diagram showing the latency measurement process. Red arrow shows the signal being delayed due to running through the max patch, eventually being output from the Genelec and running into reaper. Blue arrow indicates the direct signal to Reaper. The numbers indicate which of the audio waves comes from which source, 1 being the direct sound and 2 being the delayed sound.

Previous testing of the system had shown that the system ran smoothly when using an input block size of 32 in Max, however for future records (and in case the system speed were to slow down due to potential future changes) the latency was measured for varying block sizes. The following table shows the resulting latency time due to the different block sizes used in Max. The total latency is calculated by adding 5ms to the system latency. This accounts for the time taken to travel from the loudspeaker to the centre of the speaker array where the user will be present.

Sample Rate (kHz)	Block Size	System Latency (ms)	Total Latency (ms)
44.1	512	27	32
44.1	256	20	25
44.1	128	18	23
44.1	64	17	22
44.1	32	17	22

3.8 - RIR Trimming

Due to the latency present in the system, the first 22ms of the RIR's had to be trimmed. Figure 26 shows two of the synthetic RIR's, the top shows one from the centre of the room (8.85m from the left wall) and the bottom shows an RIR taken 1.85m away from the left wall which are two of the

locations that were also taken in Hendrix Hall for user test #1. Both of the RIR's being analysed were produced with the sound source facing the left wall to emphasis the early wall reflection making the following point more obvious.

Points (1), (2) and (3) are identical in both RIR's showing the start of the impulse, the direct sound and the floor reflection respectfully. The main difference in the RIR's are the direct reflection times from the left wall. (4) shows the wall reflection occurring at 0.04084s, indicating a travel distance of 3.75m (following the same patch described in figure **??** in section **?? ??**). (5) shows the reflection from the same wall, however occurring much later due to the greater distance from the wall. The point at which the RIR's will be trimmed is indicated by the red intersecting line, occurring at approximately 0.052s. It can be seen that the direct reflection from the left wall (4) is removed from the impulse, whereas in the RIR in the centre of the room, the direct wall reflection (5) remains in the trimmed version of the RIR. Due to the 22ms delay, any sound that travels less than 7.568m (if the source and receiver were placed 3.784 away from a surface) before reaching the receiver will be removed from the RIR.

The necessity for direct sound has been previously discussed in [25], stating that as reverb builds up over time eventually masking reflections, the direct sounds are used for sound source localisation. As the user in the VAE is both the sound source and the receiver, they will rely on direct reflections from the walls to determine their location within the space, however, the direct reflections from walls have had to be removed in some cases. Therefore, when moving closer to a wall, it may become difficult for the user to determine their locations.

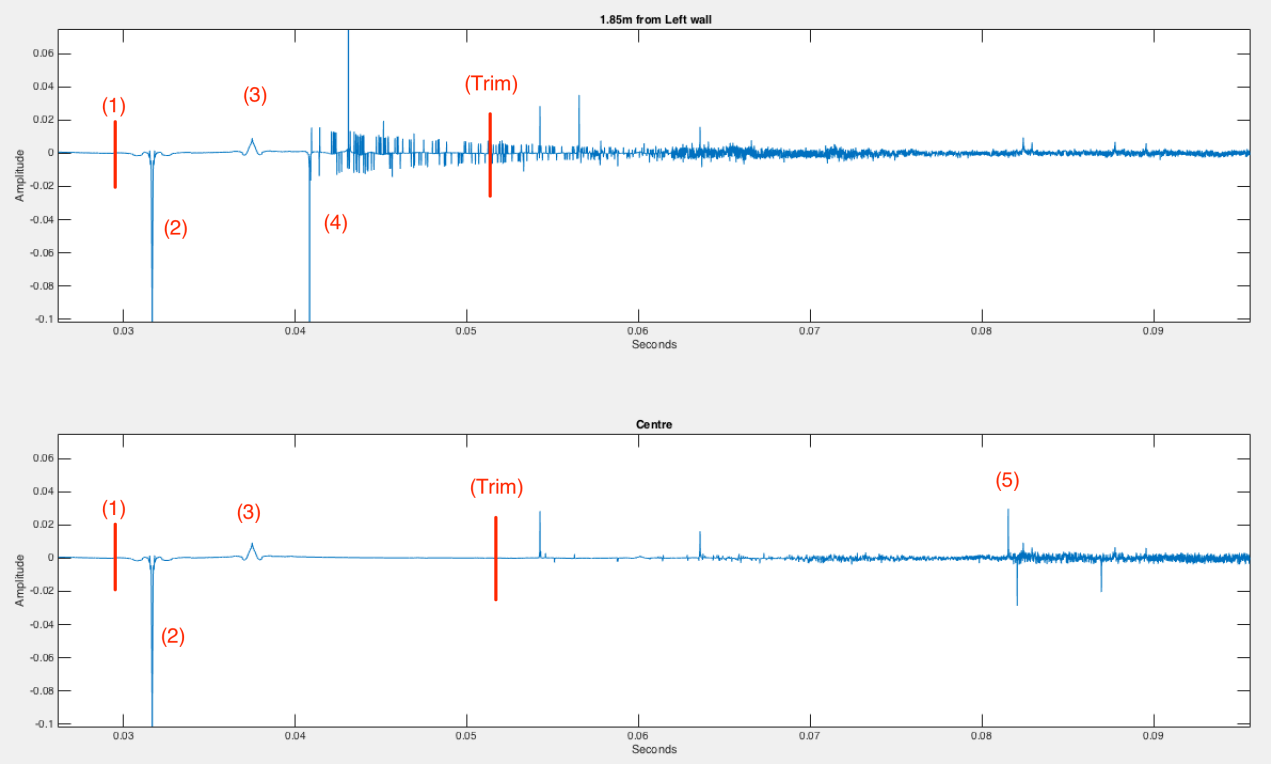


Figure 26

The Matlab script used from audio file trimming can be found: RIRtrim.m

3.9 - Software Implementation Summary

Revisiting the aims regarding software implementation set in 3.5.2 Project Aims and Motivation, it can be said that two out of the three accomplished:

1) The production of a user interface that can be used remotely from within the spherical speaker array.

2) The extension of the Max patch that could accommodate the user interface and load the appropriate RIR files required to place the user in the desired location within the VAE.

These has been achieved by producing the user interface using the 'c74' object, allowing the iPad to communicate with the Max patch and visa versa. The interaction with the user interface is interpreted by the 'UI_to_file' patch and the 'loadFilesLogic' javascript object. These successfully interpret the users intention to move to a position within the VAE and calculate the appropriate files to load.

3) The production of a system that can interpolate between the appropriate RIR positions.

For the third aim set, it must be said that this was only partially fulfilled. Though the final iteration of the software (3.6.3.3 Iteration 3 (Final)) interpolates between the appropriate RIR's for the given design, it does not work the way originally intended due to the technological restrictions previously stated. Instead of allowing the user to 'freely' move around the space, they are instead restricted the the available RIR locations and are only able to move between them when moving from one set location to another, as opposed to being able to hover between them as would have been able using iteration 1 (3.6.3.1 Iteration 1).

However, as the user interface interpretation section of the software simple outputs the required file names, it can be integrated with an improved version of the mobility section in the future. Therefore with simplifications to the first iteration of the software, such as providing the user with a grid of four RIR locations instead of a full grid of 9, though less accurate, would be much quicker and would provide more freedom than the currently implemented version.

Due to this being a real time system, the latency introduced as a result greatly effects a large number of the RIR files being used by having to cut 22ms of each one of them. This unfortunately removes some of the direct wall reflections for surfaces that are closer than 3.75m. However, this process also removed the direct sound and floor reflection as desired.

USER TESTING

Once the implementation of the system to allow the user to move themselves around a virtual space had been completed, the plausibility of the methods used were tested. The following sections present the aims of the user tests in full, the procedure required to fulfil the set objectives followed by the results and a discussion.

In total, seven students took part in the user tests, all of which were studying audio related subjects.

4.1 - USER TESTING AIMS

There were a total of 3 user tests, with the third being split into two parts with an identical procedure. The aims of each test were as follows:

Test #1: Investigate the effect of using the synthetic RIR's as opposed to the previously used real RIR's used.

Test #2: Investigate how far the user has to be moved before they notice that they have been moved using synthetic RIR's

Test #3.1: Investigate which RIR grid provides the user with the best sense of mobility

Test #3.2: Investigate whether the users opinion changes when using a position feedback system

Before the tests were conducted, each participant was presented with a 'Test Participant Form', informing them of the aims of the tests and the procedures that were to follow. As they were going to be stood inside the speaker array, the answers provided by each participant were taken down on their behalf. At the end of the experiment the answers were checked and signed by the participant assuring that their answers had been taken down accurately. The form provided to the participants can be found in **??**.

Before each test, the participants were allowed two 'dummy runs', allowing them to get used to the system without having their answers recorded, with the intention of removing the disadvantage of being unfamiliar with the system for the first few questions of each test. They were also informed that during each test they were allowed to turn their head in the space to obtain a more natural listening experience.

4.2 - Statement of Hypothesis

4.3 - Test #1

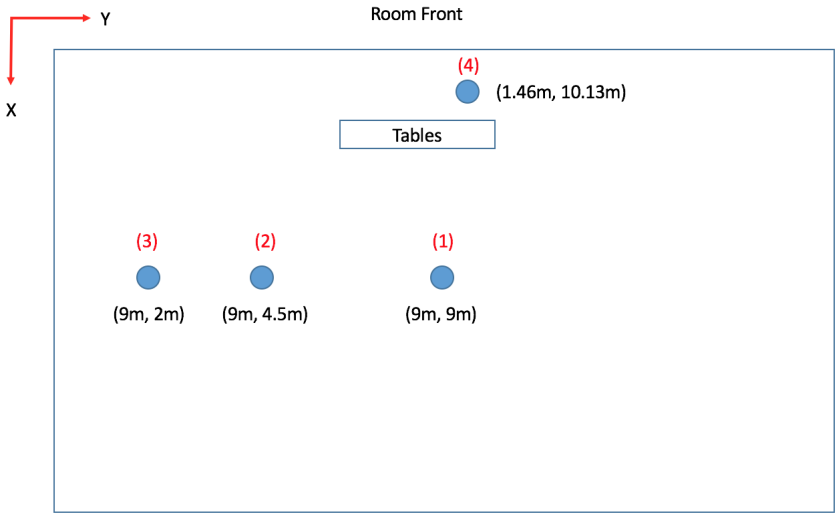
4.3.1) Procedure

To investigate the difference in the perception of distance moved when using either synthetic or real RIR's, the following procedure was carried out:

The participant was told that two methods were going to be investigated during this test, method **A** and method **B**. The participant was not told that in method **A**, the real RIR measurements from Hendrix Hall were going to be used to move them around the room and in method **B** the synthetic RIR's produced in Odeon were going to be used. The participant was told that using method **A**, they were going to be placed somewhere in the virtual space and that they would be asked to say the word 'Bob'. They would then be moved to another position in the space and asked to repeat themselves. This procedure would the be repeated, however this time using method **B**. They were then asked to state whether they felt they had:

1) Moved a **shorter** distance than they had in **A**
2) Moved the **same** distance they had in **A**
3) Moved a **further** distance than they had in **A**
4) I don't know

This was repeated three times where the distances used for mathes A and B were kept the same. For the final two trials, the distances between the two methods were changed. The left of figure 27 shows an illustration of the virtual space indicating where the four RIR locations used for the test are. The black numbers indicate the coordinates of the RIR's relative to the top left corner of the room (note that the x and y axis are opposite to convention due to the way the building was modelled in Google SketchUp), and the red numbers are used to indicate which location the user was moved to during each test, shown on the right in figure 27.



Trail	Real RIR **(A)**		Odeon RIR **(B)**	
	Start	End	Start	End
1	(1)	(2)	(1)	(2)
2	(1)	(3)	(1)	(3)
3	(1)	(4)	(1)	(4)
4	(1)	(3)	(1)	(4)
5	(4)	(1)	(1)	(3)

Figure 27: **Left:** Illustration of the RIR locations used in user test #1. **Right:** Table showing the pairs of positions the participant was moved to, corresponding to the positions shown in the diagram on the left.

4.3.2) Results

The participants answers for user test #1 were taken and averaged across all five trials. Figure 32 shows these results, showing the percentage of correct and incorrect answers given (left) and the percentage of each type of answer given (right). It can be seen that only 23% of the answers given across all trials and participants were correct, indicating there being difficult comparing the two distances moved.

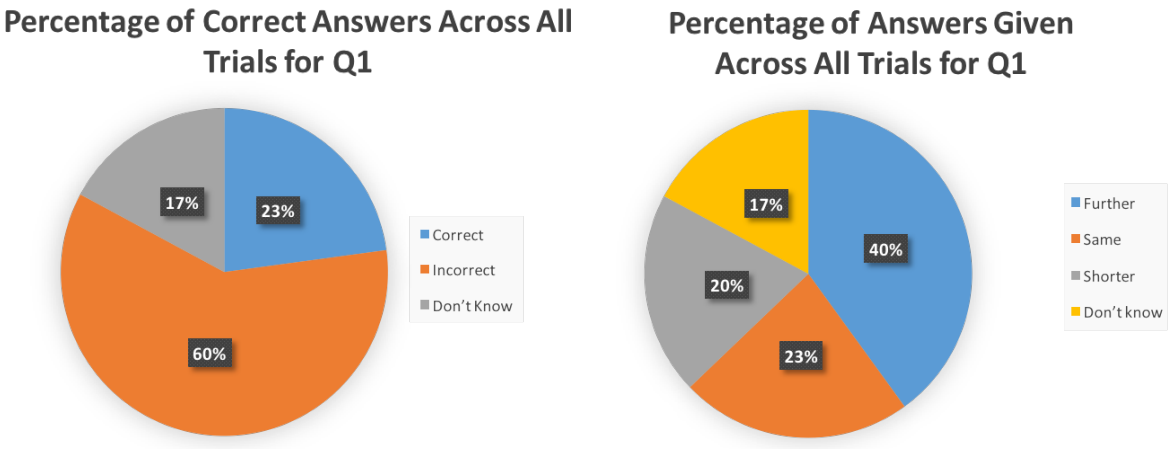


Figure 28: Pie charts showing the results of user test #1

The pie chart on the right shows that more commonly, people thought that they were moving a further distance when the synthetic RIR's were used. It can be seen in the table in figure 27 that the expected answers are as follows:

Trial	Correct Answer
1-3	Same
4	Shorter
5	Further

However, an interesting result can be seen when breaking down the answers given into their respective trials, as shown by the three pie charts in figure 29. When the distance moved was either the same (trials 1-3) or shorter (trial 4), more people tend to answer that they had moved further, more so when they had actually moved a shorter distance. When the distance moved was actually further, the answers are evenly split between 'Further', 'Same' and 'Don't Know'. This negative correlation between expected answer and given answer suggests that the pie chart on the right in figure 32, showing that the majority of people thought they were moving further throughout all trials, is not influence by the fact that people got the answer to trial 5 correct, indicating that it is the use of the synthetic RIR's themselves that cause this perception.

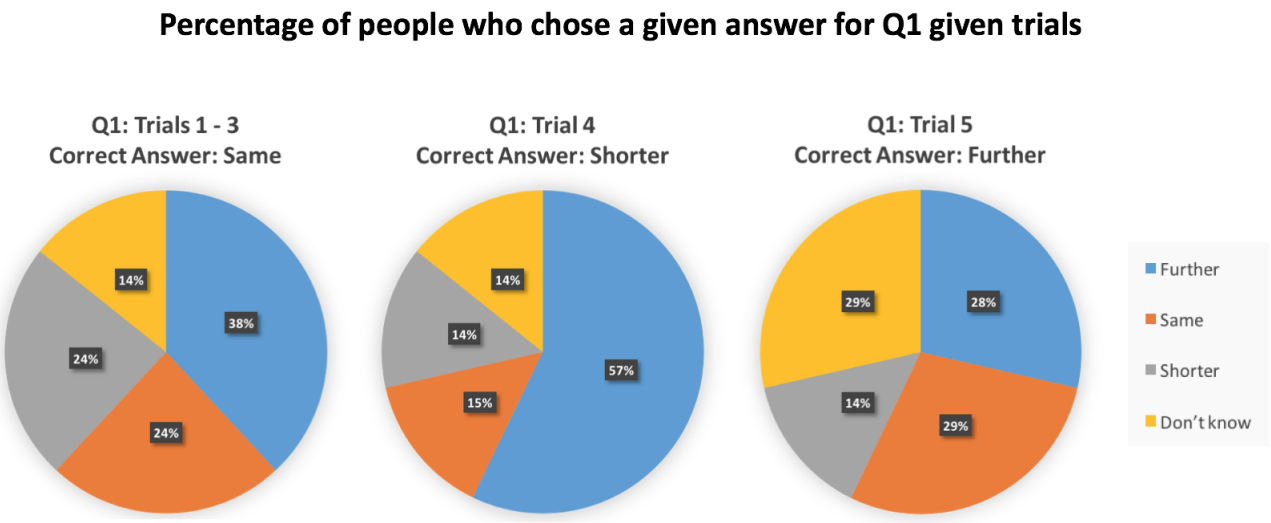


Figure 29: Three pie charts showing the percentage of answers given for: **Left:** Trials 1- 3, **Centre:** Trial 4, **Right:** Trial 5.

4.3.3) Discussion

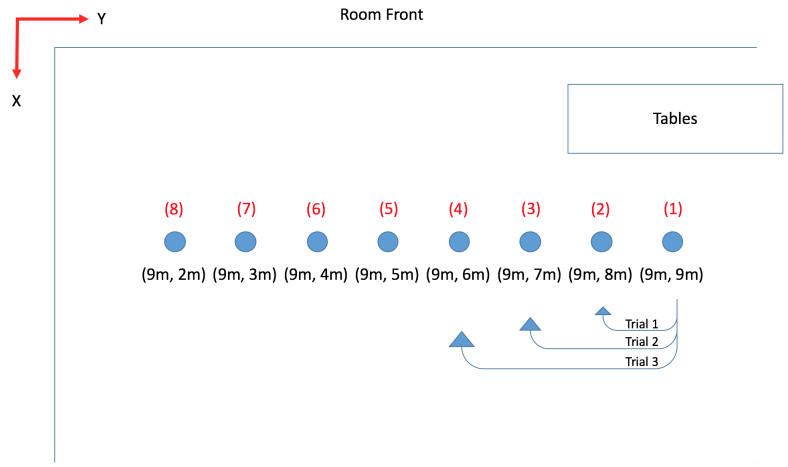
Due to the small percentage of correct answers given, it is apparent that comparing the difference in distance moved is not easy. For the system used, there may be 2 reasons for this:

1) **The test is too hard**
   Asking the participants to not only compare how two different locations sound to each other, but compare that difference to how another two sound compare to each other may have been too much to remember at once. This may have led to participants guessing answer as opposed to giving an answer based on opinion, even though the 'I Don't Know' option was available.
2) **Inaccurate RIR's**
   It has been mentioned in section 3.8 RIR Trimming that due the fact that the RIR's had to be trimmed due to system latency, much needed direct wall reflections are not present for surfaces less than approximately 3.78m away. Though this may effect the ability to accurately access ones location in the virtual space, it does not explain the perception of moving a further distance when using synthetic RIR's than when using real measured RIR's

4.4 - TEST #2

4.4.1) Procedure

The second user test is similar to that of the first test, though a little more simple. Figure 30 shows 8 RIR locations, where (1) is in the centre of the room. The participant was placed at position (1) and asked to say the word 'Bob'. They would then be moved closer to the left wall, asked to repeat themselves and asked whether they thought they had moved or not, simply answering yes, no or 'I Don't Know'. This was done 7 times, moving the participant a further distance each time. The table on the right in figure 30 shows the start and end position for each trial.



Trail	Position (x,y)	
	Start	End
1	(1)	(2)
2	(1)	(3)
3	(1)	(4)
4	(1)	(5)
5	(1)	(6)
6	(1)	(7)
7	(1)	(8)

Figure 30: **Left:** Illustration of the RIR locations used in user test #2. **Right:** Table showing the pairs of positions the participant was moved to, corresponding to the positions shown in the diagram on the left.

4.4.2) Results

Figure 31 shows a line graph of the results obtained from user test #2, showing on how many participants answered 'Yes', 'No' or 'Don't Know' for each of the trials.

The original hypothesis to this test was that the participants were unlikely to notice that they had moved for the first few trials as they were being moved such a short distance (1m, 2m), with the expectation that they would notice when they were being moved much larger distances, hence the reason there were moved 1m closer to the wall in each trial. However, it was found that all participants answered 'Yes' for the first trial (movement of 1m towards the left wall) and all other trials resulted in a number of participants giving an answer other than 'Yes'. Instead of a positive correlation between distance moved and 'Yes' answers given, there appears to be a dip from trials 2-4

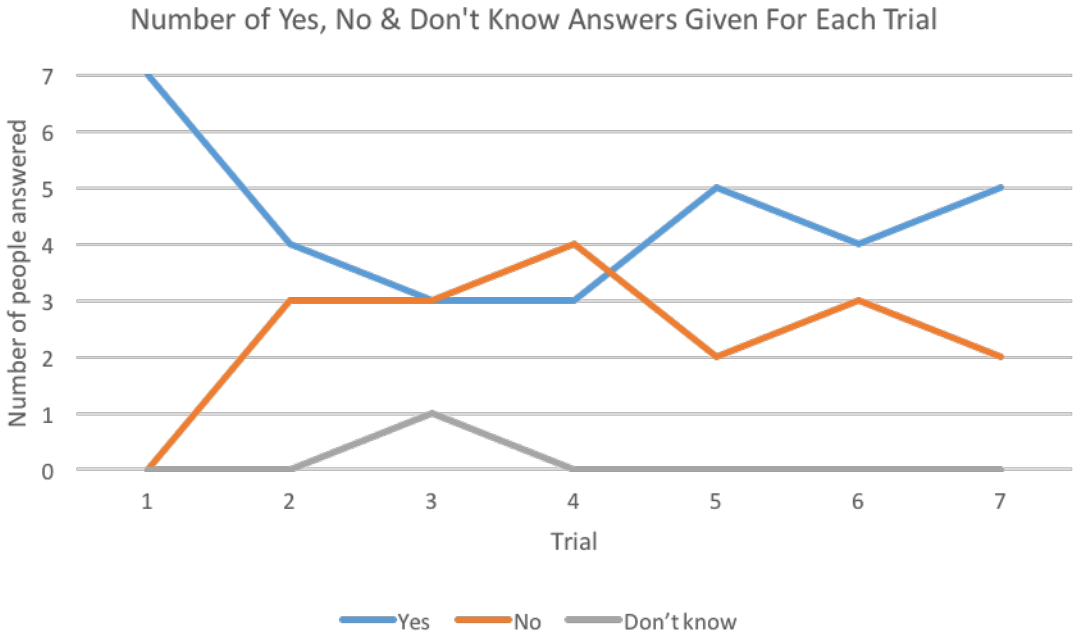


Figure 31: Line graph of the results from user test #2 showing the number of participants who answered 'Yes', 'No' or 'Don't Know' for each of the trials.

4.4.3) Discussion

Comparing the results to the original hypothesis, excluding the first two trails, there seems to be more people noticing movement as the distance moved increases. These first two trials may have been subject to a bias, where the user is expecting to be moved. This however can not be backed up and remains a speculation.

Other causes for these results could be one of the following:

1) That the RIR's used do not contain enough information regarding the users location.
2) The participants answering 'No' simply don't know or do not hear that they expect to hear when moving in a room.
3) The inconsistency in repeating themselves has more of an effect on their perception of location than the actually room acoustics.

4.5 - TEST #3

4.5.1) Procedure

The participants were presented with an iPad and informed that it could be used to draw a path within the virtual space which they would then be moved around. Five trials were run, each using a different RIR grid, previously explained in section **??** **??** and displayed in section **??** **??** in figure **??** (1m separation) and in **??** **??** as figures **??**,**??**,**??**,**??** (2m - 5m separation respectively). The following table shows which trial used with RIR grid, stating the distance between RIR locations and the number of RIR locations within that grid:

Trial	RIR Location Separation	Number of Positions Available
1	1m	240
2	2m	112
3	3m	25
4	4m	12
5	5m	9

In each trial, the participant was asked to draw a path and listen to an audio sample [26] (**[REFERENCE AUDIO SAMPLE]** which was played in place of the participant speaking/singing to provide a constant sound source (as determining movement would be difficult with inconsistent speaking). They were then asked to rate on a scale of 1-10 the quality of mobility, where 1 = extremely "jumpy" movement and 10 = completely smooth movement, or to give the answer 'N/A' if they had difficulty telling whether they were moving or not.

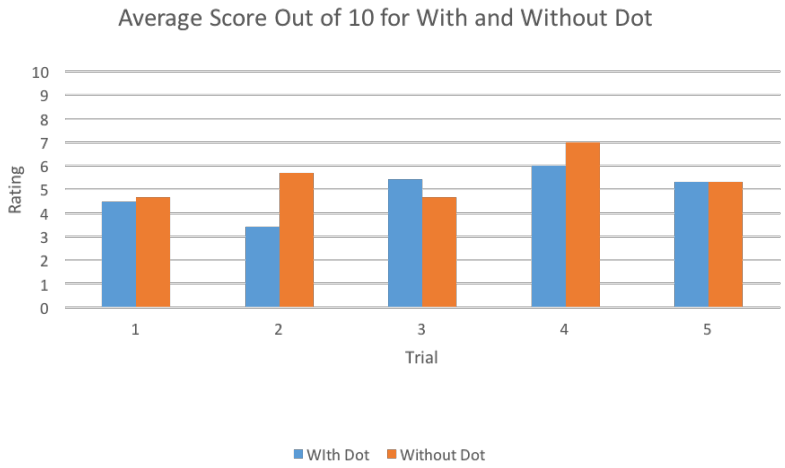
The five trials were run twice, once without a dot showing them where they are in the virtual space and once with a dot. This was to test whether their perception of how 'smoothly' they were moving around the space changed when they could see exactly where they were.

4.5.2) Results

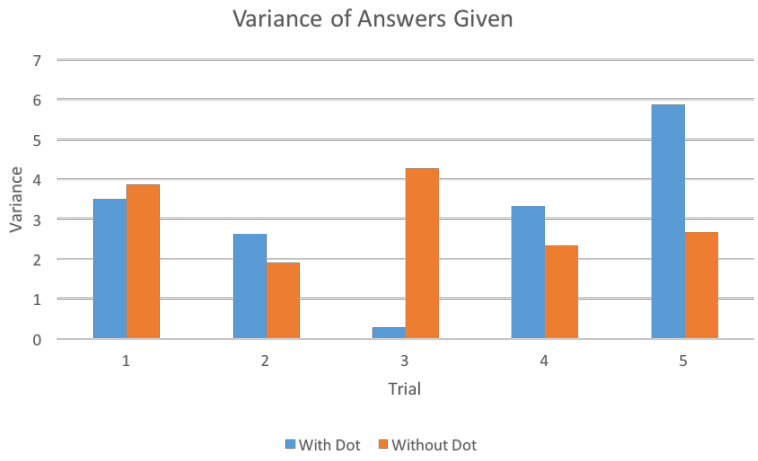
The results in figure 32a show that on average, using the grid with RIR locations separated by 4m when the location tracking dot was not used (blue bars), scored the highest. However it only scoring an average 0.5 higher than the grid using 3m of separation. By looking at that variance of the answers given in figure 32b, it can be seen that the grid with 3m separation was rated much more consistently that the grid with 4m separation.

When the location tracking dot was used, it was found the using the grid with 4m separation on average scored the highest with only a slightly higher variance than the grid of 2m separation, which scored second highest.

Participants were asked to provide comments on using the system both with and without the location tracking dot. Several participants commented on trial number 3, mentioning that it was easier to tell that they were moving than the prior trials. Comments on using the location tracking dot referred to the fact that it helped them determine what they should be listening for, one participant in particular referring to the speed the dot was moving at was important, given its variance across trials.



(a) Average score given from 1 - 10 (1 = jumpy movements, 10 = smooth movement) for each of the trials 1 - 5 (1 = 1m RIR separation, 5 = 5m RIR separation)



(b) Variance of the answers given in figure 32a

Figure 32: Pie charts showing the results of user test #1

4.5.3) Discussion

As discussed in section 3.6.3.3 Iteration 3 (Final), the rate at which the user is moved from one RIR location to another is determined by a timer which runs at a constant speed. This means that when the distance between RIR locations is greater, the speed at which the participant is moved across the room is greater. This inconsistency in movement speed is a variable that has had an effect on the perception of movement when using different sized RIR grids, where participants had noted that it felt like they were moving faster. This may have also had an effect on their scores of the different trials. For example, it had been noted by some participants that in trials 1 (1m RIR separation), it did not feel like they were moving. This is most likely because moving 1m every 2.5 seconds is too slow to notice. However, due to the compromise that had to be made due to system speed, they could not be moved any faster than this. This may have led to participants giving the trials that used more spacious RIR placement a higher score as the fact that they could heard themselves moving more obviously due to the greater distance travelled, thus thinking that they were moving more smoothly.

This is most likely the reason why trials 3 and 4 scored the highest whether the dot was used or not.

Due to the issues mentioned, the author concludes that the results of these tests are inconclusive, as they do not answer the original question asked due to the variable of speed incorporated into the system.

This test was more suited towards iteration 1.

4.6 - User Tests Discussion and Summary

4.6.1) Test Length

In total, all three user tests took around 45 - 60 minutes for each participant. Ideally, some of the test would have been longer allowing for a greater number of results to be gathered thus providing a more statistically reliable set of results. For example, the results for test #2 would benefit by making the user take the same test multiple times to check that variance in their answers in an attempt to minimise outliers, such as the unexpected in trial 1 (shown in figure 31). However, given that the user tests were already long, it was decided to not extend it any further. A possible solution to this would have been to do the three user tests separately at separate times, each one of them extended to obtain more reliable results.

4.6.2) The 'I Don't Know' Problem

Previous papers discussing the pros and cons of including an 'I Don't Know' (DK) answer, such as [27], suggest that including a DK response could leave holes in data by preventing a par-

ticipant from having to do any cognitive work thus not coming to a conclusion or formulating an opinion for an answer, even if they have one, or it could prevent noise in the data due to participants having to provide a guess for an answer if they truly have no answer to give. Due to the difficult of these listening tests, it was decided that it was necessary to include a DK response, running the risk that the difficulty of the test may result in participants taking the easy option. However, during the user tests it was noted by the author that participants seemed hesitant about giving a DK answer, even when encouraged to do so when they were clearly struggling to formulate an answer. A lot of the time it was noticed that the answers given by participants were not confident. Though the analysis of the confidence in their answers given are informal, the author feels it is important to take into account for two reasons;

1) The small number of people who took part in the user tests.
2) The difficulty of the tests.

As there were only 7 people who took part in the test, a guess answer from any of the participants may heavily influence the obtained results and as it is thought that most of the participants were not sure of their answers, the obtained results do in fact unreliable. Coupled with the fact that the test (especially test #1) were quite difficult, it is expected that guessing will have occurred. Again, these claims can not be backed up as the confidence in the participants answers are not formally part of this study, however it may explain some randomness in the results.

4.6.3) Results Summary

Test #1: When using the synthetic RIR's, participants were more likely to feel like they had moved further than when using the real RIR's regardless of the difference in distance moved.

Test #2: Mixed results show that there is no strict correlation between whether a participant can tell they have moved and the distance they were moved for the RIR files used in this system.

Test #3: For the currently implemented system, using an RIR grid with 3m separation provides the best experience for free mobility around the virtual space.

CONCLUSION

5.1 - PROJECT SUMMARY

A digital model of Hendrix Hall was deigned and used to produce a large grid of synthetic RIR's in order to implement a direct RIR rendering system with the aim of allowing a user to move themselves around a virtual space. The original Max patch designed for the VSS was built upon to accommodate this new functionality by implementing an automated file loading system and a RIR interpolation system that approximates the movement defined by the user. Three user tests were carried out to access the plausibility of the system as a whole and to uncover information regarding the perception of mobility with regards to the implemented system.

5.2 - PROJECT AIMS EVALUATION

Here, the project aims set in 3.5.2 Project Aims and Motivation are re-stated:

1. To implement a direct RIR rendering system as an extension of the VSS
2. To investigate the number of RIR's that are required to convince a user that they can freely move themselves around a virtual space without limitation
3. To investigate the difference in the perception of mobility in the virtual space when using real RIR's and synthetic RIR's

Upon completion of the project, it can be said that each of these aims had been achieved, however the quality

5.3 - PROJECT SUCCESS EVALUATION

The success of the project can be measured to evaluation the correctness of the statements first provided in section **?? ??**

> **"Produce a system that: allows the user to move themselves around the VAE freely"**

Here, the word 'freely' must be defined. Originally the intent was to allow the user to select any location in the virtual space and hear themselves as though they are present in the location. As it is not possible to provide and RIR in every possible location using the direct RIR rendering method, interpolation between neighbouring RIR's would provide an approximation of an RIR in that location. It was discovered however that software implementation that provided this functionality would not run smoothly in real time. Therefore a compromise was found where the

user can move themselves along a path in the virtual space, but could not technically experience even an approximation of an RIR location, instead only being able to place themselves on specific RIR locations. However, as the users were not informed that this was convinced, it was possible that their perception of location was changed, believeing that they were infact in the desired location.

Due to this compromise, this metric is deemed on partially successful

Originally, the intent was to allow the user to hear themselves as though they were placed anywhere within the virtual space. Though this would technically not be possible as RIR's would not exist at every point in the space, it is possible to convince the user that this is possible by interpolation between multiple RIR locations. Therefore, in this case the user must *feel* as though they can move freely.

5.4 - Metrics Review

Due to the fact that iteration 1 and 2 failed, the fact is that technically the user could not move themselves freely, but rather move themselves to the closest possible RIR.

5.5 - Overall System Implementation

REFERENCES

[1] J. S. Brereton and B. A. Hons, "Singing in space (s): singing performance in real and virtual acoustic environments — singers ' evaluation , performance analysis and listeners ' perception .," no. August, 2014.

[2] J. Huopaniemi, L. Savioja, T. Lokki, and R. Väänänen, "Virtual acoustics - applications and technology trends," *Proc. European Signal Proc. Conf. (Eusipco'2000)*, pp. 2201–2208, 2000.

[3] G. Stan, J. Embrechts, and D. Archambeau, "Comparison of different impulse response measurement techniques," *Journal of the Audio Engineering Society*, vol. 50, no. 4, pp. 249–262, 2002, ISSN: 1549-4950. [Online]. Available: http://orbi.ulg.ac.be/handle/2268/34825 (visited on Apr. 25, 2016).

[4] S. W. Smith, "The delta function and impulse response," *The Scientist and Engineer's Guide to Digital Signal Processing*, pp. 107–122, 2003.

[5] Iso 3382-1, "Acoustics - measurement of room acoustic parameters - part 1: performance spaces," vol. 3, 2009.

[6] H. Robjohns. (2001). You are surrounded, sos, [Online]. Available: http://www.soundonsound.com/sos/oct01/articles/surroundsound3.asp.

[7] P. J. Power, "Future spatial audio : subjective evaluation of 3d surround systems," PhD thesis, Acoustics Research Centre, University of Salford. [Online]. Available: http://usir.salford.ac.uk/34100/1/P%20Power_Thesis_Final.pdf (visited on May 7, 2016).

[8] P. White. (2016). Recording a live choral performance, sos, [Online]. Available: http://www.soundonsound.com/sos/jun04/articles/liveconcert.htm (visited on Apr. 25, 2016).

[9] S. Siltanen, T. Lokki, and L. Savioja, "Rays or waves? understanding the strengths and weaknesses of computational room acoustics modeling techniques," *Proceedings of the International Symposium on Room Acoustics, ISRA 2010*, no. August, pp. 1–6, 2010. [Online]. Available: http://www.acoustics.asn.au/conference%7B%5C_%7Dproceedings/ICA2010/cdrom-ISRA2010/Papers/O5a.pdf (visited on Apr. 25, 2016).

[10] (2016), Odeon, [Online]. Available: http://www.odeon.dk/pdf/ODEONManual.pdf (visited on Apr. 25, 2016).

[11] J. H. Rindel, "Computer simulation techniques for acoustical design of rooms," *Acoustics Australia*, vol. 23, pp. 81–86, 1995. [Online]. Available: http://www.odeon.dk/pdf/AustralAc_1995_Rindel.pdf (visited on Apr. 25, 2016).

[12] C. L. Christensen and G. Koutsouris, *Odeon user's manual*, Scion DTU Diplomvej Kgs. Lyngby Denmark, 2015. [Online]. Available: http://www.odeon.dk/pdf/ODEONManual12.pdf (visited on Apr. 25, 2016).

[13] Google. (2015). Google sketchup, [Online]. Available: http://www.sketchup.com/ (visited on Apr. 25, 2016).

[14] (2016). Su2odeon, Odeon, [Online]. Available: http://www.odeon.dk/su2odeon-plugin-trimble-sketchup (visited on Apr. 25, 2016).

[15] (2016). Max/msp, Cycling74, [Online]. Available: https://cycling74.com/products/max/ (visited on Apr. 25, 2016).

[16] (2016). Spat, IRCAM Centre Pompidou, [Online]. Available: http://forumnet.ircam.fr/product/spat-en/ (visited on Apr. 25, 2016).

[17] (2016), Oculus, [Online]. Available: https://www.oculus.com/en-us/ (visited on Apr. 25, 2016).

[18] L. Savioja, J. Huopaniemi, T. Lokki, and R. Väänänen, "Creating interactive virtual acoustic environments," *J. Audio Eng. Soc*, vol. 47, no. 9, pp. 675–705, 1999. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=12095 (visited on Apr. 25, 2016).

[19] R. Mehra, N. Raghuvanshi, M. Lin, and D. Manocha, "Efficient gpu-based solver for acoustic wave equation," pp. 1–10, 2010. [Online]. Available: http://mitran-lab.amath.unc.edu:8081/subversion/PetaFlopAcoustics/StochasticHelmholtz/biblio/Efficient%20GPU-Based%20Solver%20for%20Acoustic%20Wave%20Equation%20Nikunj%20Manocha.pdf.

[20] (2016). Mira, Cycling74, [Online]. Available: https://cycling74.com/products/mira/ (visited on Apr. 25, 2016).

[21] (2016). C74, Cycling74, [Online]. Available: https://cycling74.com/project/project43-c74-iphone-app/#.Vv1FhRIrJE4 (visited on Apr. 25, 2016).

[22] Y. Technology, *3-space sensor embedded*, 630 Second Street, Portsmouth, Ohio 45662. [Online]. Available: http://www.solarsystemexpress.com/uploads/5/0/6/0/5060129/3-space_sensor_users_manual_embedded_1.1_r13.pdf (visited on May 2, 2016).

[23] *M30 high definition measurement microphone*, Data Sheet, Earthworks, 2016. [Online]. Available: http://www.earthworksaudio.com/wp-content/uploads/2016/03/M30-Data-Sheet-2016.pdf (visited on May 3, 2016).

[24] (2016). Motu ultralite-mk3, MOTU, [Online]. Available: http://motu.com/products/motuaudio/ultralite-mk3 (visited on May 3, 2016).

[25] S. Devore, A. Ihlefeld, K. Hancock, B. Shinn-Cunningham, and B. Delgutte, "Accurate sound localization in reverberant environments is mediated by robust encoding of spatial cues in the auditory midbrain," *Neuron*, vol. 62, no. 1, pp. 123–134, 2009, ISSN: 08966273. DOI: 10.1016/j.neuron.2009.02.018.

[26] C. Steele-Perkins. (2016). 'the trumpet shall sound' anechoic trumpet sample, OpenAir, [Online]. Available: http://www.openairlib.net/sites/default/files/anechoic/data/helena-daffern/handel-trumpet/mono/tr-1888-piece2-t-32.wav (visited on May 6, 2016).

[27] Paul Beatty & Douglas Herrmann, "A framework for evaluating "don't know" responses in surveys," *Vasa*, pp. 1005–1010, 1989. [Online]. Available: http://medcontent.metapress.com/index/A65RM03P4874243N.pdf.