Capstone Project Report

University of Washington

Department of Electrical Engineering

Digital Signal Processing

Yi-Ching (Cathy) Oun

Jeffrey Alcock

June 8, 2015

Contents

[Problem Formulation 3](#_Toc421404892)

[Motivation 3](#_Toc421404893)

[Virtual Reality 3](#_Toc421404894)

[Surround Sound Technology 3](#_Toc421404895)

[Delay 4](#_Toc421404896)

[Volume Manipulation 4](#_Toc421404897)

[Frequency Filtering 4](#_Toc421404898)

[A Maze Solving Game using Auditory Cues 5](#_Toc421404899)

[Concept 5](#_Toc421404900)

[Controls 6](#_Toc421404901)

[Algorithms 6](#_Toc421404902)

[Distance and Intensity 6](#_Toc421404903)

[Direction 7](#_Toc421404904)

[Low Pass Index 8](#_Toc421404905)

[Implementation 8](#_Toc421404906)

[High Level 8](#_Toc421404907)

[MATLAB 8](#_Toc421404908)

[Back-end 9](#_Toc421404909)

[Music 9](#_Toc421404910)

[GUI 9](#_Toc421404911)

[C 10](#_Toc421404912)

[Results/Discussion 10](#_Toc421404913)

[Surround Sound Concepts 10](#_Toc421404914)

[Success of Real-Time Audio Processing 10](#_Toc421404915)

[Future Work 10](#_Toc421404916)

[References 11](#_Toc421404917)

# Problem Formulation

## Motivation

This project aims to use Altera’s DE2 board in a real-time signal processing application. The course affiliated with this project extensively covered audio processing; consequently, it was decided to focus on processing and analyzing audio in real-time in order to create an immersive virtual reality experience game that exercises concepts learned in class, and it is also fun to play.

## Virtual Reality

Virtual reality is defined in the *Merriam-Webster Dictionary* as, “an artificial world that consists of images and sounds created by a computer and that is affected by the actions of a person who is experiencing it.”[1]. The goal is to replace our innate human reliance on visual feedback with audio feedback. The specific technology utilized will be the idea of surround sound.

## Surround Sound Technology

Surround sound gives a sense of direction to an observer listening to an audio source. This technology is commonly utilized in high-end media modalities (such as movie theaters) to provide observers with a more immersive experience. In this specific application, surround sound is implemented on stereo audio hardware (ear buds) instead of a more traditional model where physically distinct speakers produce different sounds. In this application, the granularity of the sound field surrounding the user was quantized to eight different regions: front, front-right, right, back-right, back, back-left, left, front-left as shown in Fig. 1. This simplifies computations and strives to make the effects more distinct. And we are able to use the difference of distance between each region to implement the delay in order to create surround sound effect.

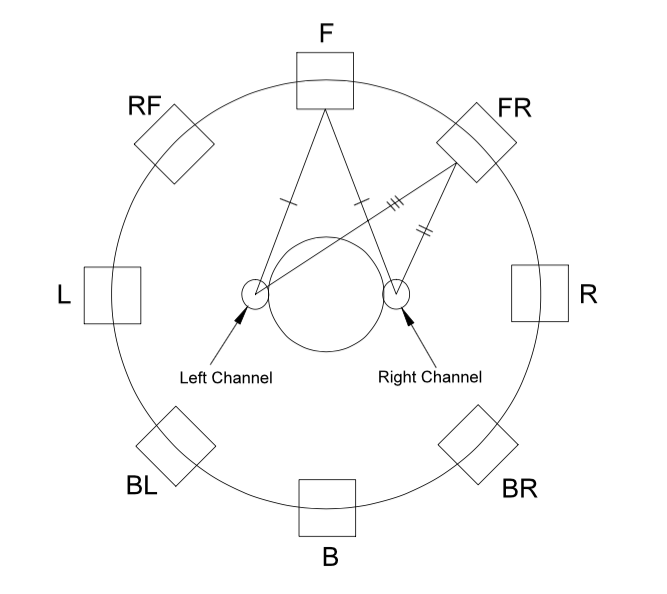


Fig. 1 Sound Field Diagram

Surround sound can be achieved through a compilation of basic processing effects, as described in the three steps below.

### Delay

The distance from a source to each ear can vary by direction based on simple geometry. For example, in Fig. 1, if a user is facing forward, sound emitted from the front (F) will reach both user’s left and right ear at same time. However, if the sound source is at their front-right (FR), the sound will reach the user’s right ear before it reaches the left ear. This equates to a simple delay effect in one channel and helps the user localize the direction of the sound source.

### Volume Manipulation

Two phenomena contribute to the need for volume manipulation to emulate surround sound. Firstly, sound shadow refers to the absorption of sound by an object in its path [1]. This concept is easily demonstrated in nature by considering the effect that standing behind a tree has to listening a sound source behind the tree. The sound will be more muffled and sound more distant. In surround sound, the head is the object that produces a sound shadow, as demonstrated in Fig. 2 below.

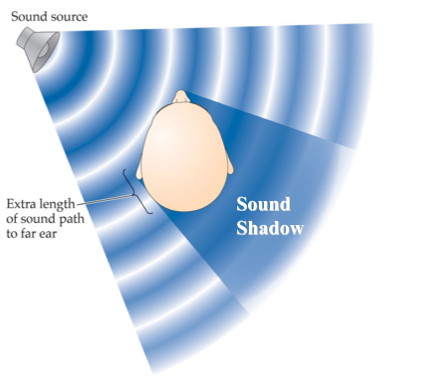


Fig. 2 Sound Path Diagram [1]

Secondly, interaural intensity difference describes the differences in the sound perceived by both ears [1]. These differences allow the human brain to localize sound and determine information about its origin. This concept is expressed in surround sound by the intuitive idea that sound from a specific direction may be heard more clearly in one ear over the other. For example, if a sound is emitted from the right of a user, the right ear will perceive the sound as louder than the left ear’s perception. This model was simplistically implemented by modifying the magnitude of the output produced in the left or right channel.

### Frequency Filtering

Human ears are oriented forward in order to hear in front of them. The outer structure of the ear, called the Pinna, allows sound to bounce off in various ways to orient the listener. The structure of the ear and some example sound paths are illustrated in Fig. 3.

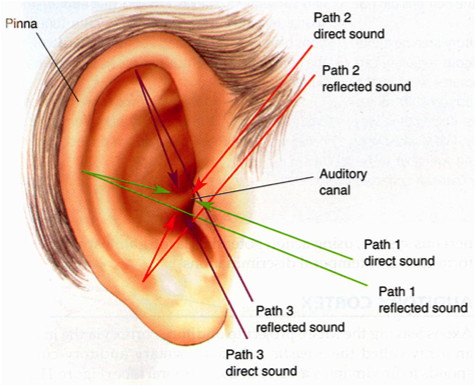


Fig. 3 Ear Anatomy [1]

Modeling sound emitted from behind the user presented a unique challenge. There is existing research about the HRTF (head-related transfer function); however, for simplicity, it was decided that the Pinna should be modeled as an exaggerated head shadow effect, meaning the user cannot hear sound emitted behind them as clearly. This effect was implemented as a low pass filter in this specific instance in order to emulate the loss of information caused by sound being absorbed by the Pinna.

## A Maze Solving Game using Auditory Cues

A natural application of surround sound is a game that requires extensive spatial knowledge, such as a maze. In two dimensions, the solver of a maze relies on visual inspection of paths and wall structures in order to solve the puzzle. In this application, these visual requirements are replaced by audio.

### Concept

The objective of the game is to find a hidden sound source located at an unknown location in the maze. When the user is very far from the sound source, all that may be heard is an ambient jungle noise, indicating that the game is working, but the user is not near the sound source. Upon moving closer, the user should hear a trumpet fanfare (this is the sound source) [2]. The fanfare will become louder as the user approaches and will filter differently based on proximity and the composition of the maze. The GUI (graphical user interface) of the game is shown in Fig. 4.

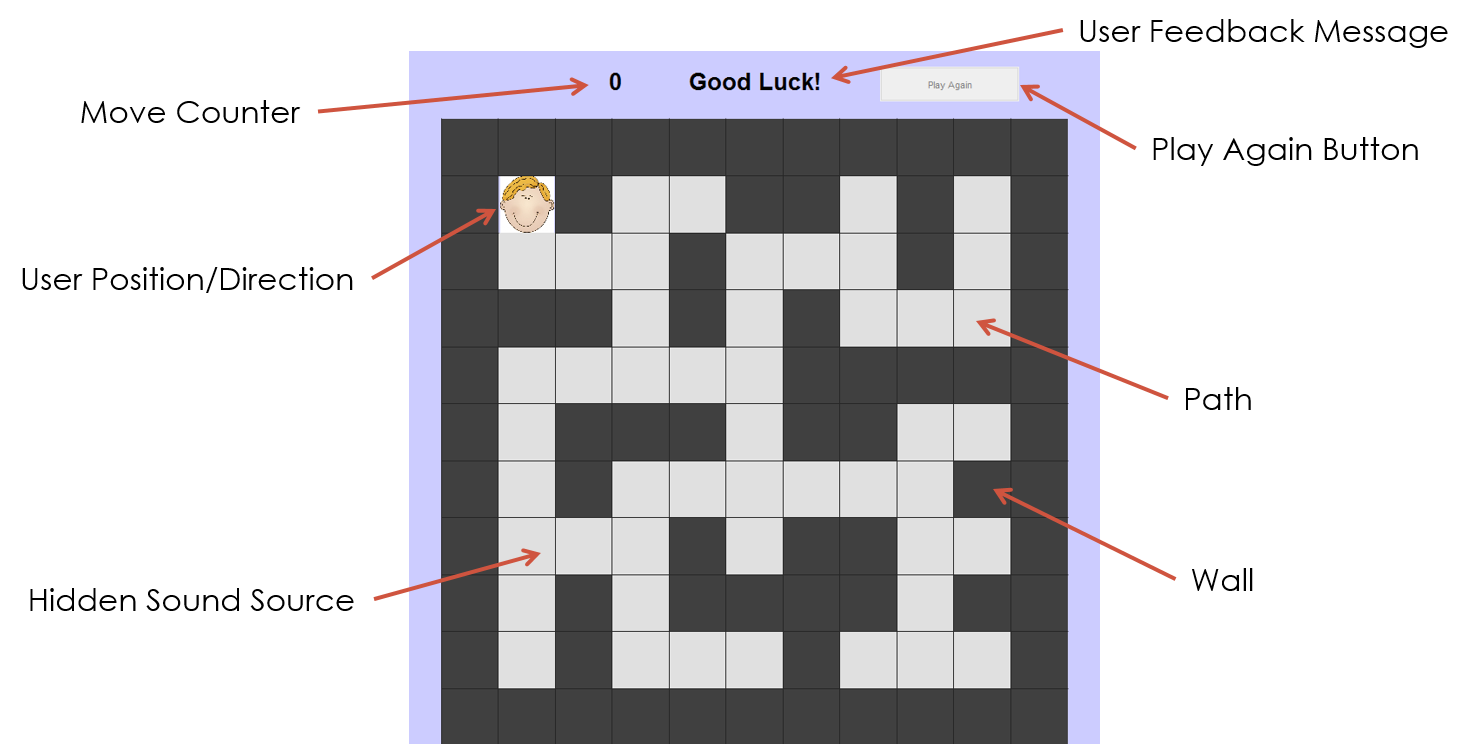


Fig. 4 Graphical user interface of the maze

The GUI through which the user will interact with this game will be created in MATLAB and displayed on a computer monitor [3]. The main GUI area is occupied by the maze itself, composed of dark tiles representing walls, and light tiles representing path. The user’s position and direction are both displayed using an animated face that moves and rotates based on user input. At the top of the GUI is a move counter that starts at 0 and increments every time the user takes a step in order to track how quickly and efficiently the maze is navigated. The user feedback message area begins with the message “Good Luck!” and changes to “You Win!” upon completion of the game. Winning also plays a celebratory sound and enables the Play Again button, to allow the user to play another instance of the maze with a newly generated sound source location.

### Controls

The user can interact with the game through the keyboard and mouse of the computer. The four arrow keys move the user along the path of the maze (the down arrow moves the user one tile down and so on). Similarly, the W, A, S, D keys allow the user to change the direction they are facing. W makes the user face up, A faces left, S faces down, and D faces right. It is not necessary for the user to turn, but it allows them to observe the auditory cues from a different angle. By rotating, one can stay in a single tile and hear the sound source move around their head by rotating to each different direction.

# Algorithms

A number of algorithms were used to calculate the various parameters needed to generate the surround sound as described below.

## Distance and Intensity

Sound intensity is the first thing modified to implement surround sound. Because intensity is proportional to the inverse of the square of distance, a simple distance algorithm was implemented [4]. Each tile in the maze is treated as a simple coordinate system with the top left path tile being the coordinate pair (2,2). The distance formula used is shown below, where *currX* and *currY* represent the current column and row, respectively, and *destX* and *destY* represent the same information about the sound source.

This distance is then rounded and quantized to fit within the integers between 0 and 9 (inclusive).

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

Once modeled in this range, the inverse square law is applied using the following formula [4] [5]**.**

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

In practice, this formula means that the fanfare it is applied to will sound very distant when far from it, and then get louder at the rate that the second order polynomial grows.

## Direction

The goal of the direction algorithm is to take the current position and sound source position and decide which of the eight directions the user should perceive the sound from. A rough idea of the regions is shown in Fig. 5 below.

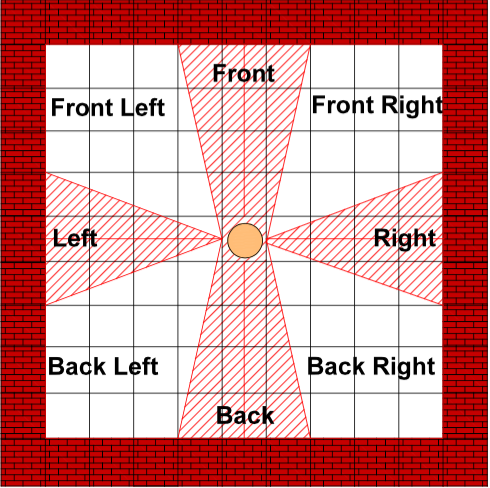


Fig. 5 Direction Mapping

Just as in the distance calculations, the tiles are interpreted as a coordinate system and the slope between the source and current position is calculating using the simple slope equation given below.

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

For an arbitrarily sized maze, the slope calculation computed at every point would result in a vector field across the entire maze; certain ranges of values apply to specific directions. Approximate thresholds for where the boundary lines should be were decided as 0.25 and 4. For example, to be considered in the right region, the slope from destination to current position must be between -0.25 and 0 or 0 and 0.25. The left region has a similar computation but can be distinguished by considering the relative magnitudes of the destination and current position (the destination X is smaller than the current X when in the left region, while the opposite it true in the right region). These thresholds are the one computation that doesn’t scale for larger mazes as one might expect. This is because at far distances, the “diagonal” direction such as back-left or front-right will expand much wider than the four “normal” distances in order to exaggerate directional perception for the user.

## Low Pass Index

A variable called lowPassIndex is created to represent exactly what sort of low pass filter (if any) should be applied to the fanfare sound. This index will be an integer between 0 and 9 (inclusive) representing increasingly small cutoff frequencies. This means that a higher number will be more low passed (filter out more frequencies) than a low number. Because of time constraints, this algorithm was greatly simplified by making the naïve assumption that this index is proportional to the distance from the sound source. In reality, the composition of the maze has a lot of influence and should be accounted for. However, the simple assumption that a faraway sound source should be low pass filtered works well enough to convey the idea. More of these ideas are discussed in the future work section.

# Implementation

This project was implemented using a standard PC equipped with MATLAB, the NiosII Eclipse IDE, and Quartus. The other necessary hardware is the Altera DE2 board equipped with a HCC-HSMC audio processing AD/DA daughter card.

## High Level

Fig. 6 illustrates the basic flow of data, beginning with keyboard input that affects the GUI and the sound produced in the ear buds. The bubble labeled Back-End represents that MATLAB code running on the PC, and the bubble labeled Audio Processing represents the C code written in the Eclipse environment that runs on the DE2 board.

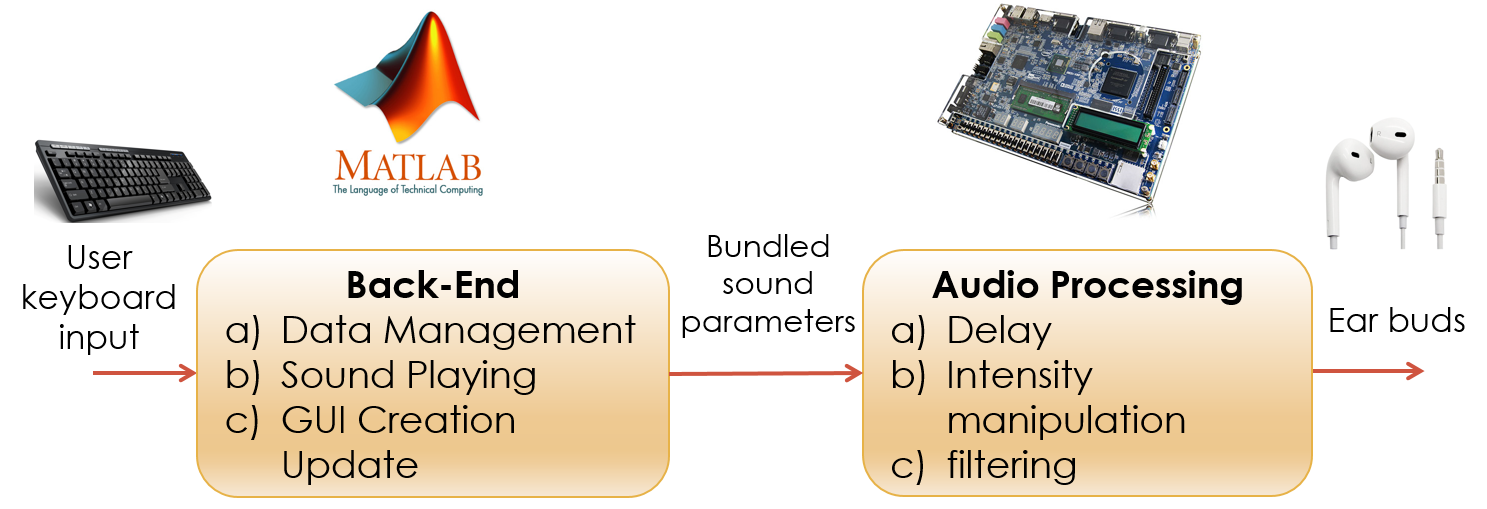


Fig. 6 System Block Diagram

## 

## MATLAB

The MATLAB code is responsible for three main functions: The back-end, music, and GUI.

### Back-end

The back-end refers to all the data management necessary to display the maze and run various algorithms. This includes variables called currentRow, currentCol to denote the user’s current position, destinationRow, destinationCol to denote the sound source’s location. Additionally, distance, direction, and lowPassIndex represent the three effects needed to make surround sound. These variables are calculated using the previously discussed algorithms and are later sent to see for real-time processing. A SerialChannel is also opened to send data to C using the UART interface.

### Music

MATLAB imports and plays the audio used in the game. The sound files are loaded and an audioplayer is created for each sound. The jungle and fanfare are combined to form a stereo sound to be sent to C, while the celebratory sound is left in a separate audioplayer. The jungle and fanfare are sent together so that each channel can be processed separately (the jungle remains unprocessed, while the fanfare undergoes all the surround sound effects described above). The dynamically repeated or stopped as needed in MATLAB. All audio is played through the computer’s headphone jack and input into the LINE IN of the daughter card.

### GUI

The interface described above is created in MATLAB. It was made from scratch (not using GUIDE). When user input is received it changes accordingly. As shown in Fig. 7, the diagram below describes the flow of functional calls.

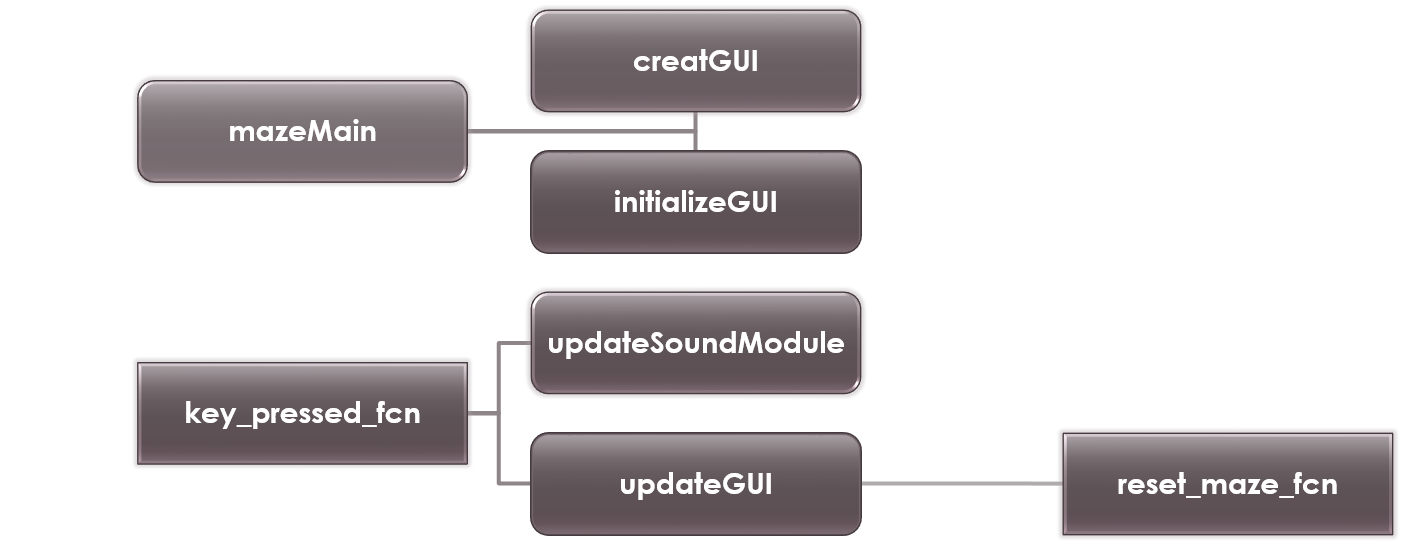


Fig. 7 MATLAB Functional Flow Diagram

* mazeMain: initializes all data and declares all the required global variables.
* createGUI: creates the necessary UI elements.
* initializeGUI: manipulates them to show in the correct state for the beginning of the maze.
* key\_pressed\_fcn: it’s called anytime keyboard input is given. It also triggers the updateGUI method which updates the GUI to its new state. if the sound source is found on that move, It calls the reset\_maze\_fcn.
* updateSoundModule: package the three parameters (distance, direction, lowPassIndex) as characters to be sent to C via UART.
* reset\_maze\_fcn: resets the user feedback message, step counter, and generates a new random location for the sound source that is guaranteed to be a reasonable distance from the user’s starting position in the upper left corner of the maze.

## C

The C portion of the code is fairly simple – it’s only purpose is to do the real-time audio processing. The following functionality is implemented using the source code given at the beginning of the class. Uart.c receives a string consisting of three characters sent from MATLAB in the uart\_byte\_recv(void) method. It stores the characters in the rx\_buffer after a brief grace period where random data is sent over from MATLAB to “initialize” the system. Odd behavior was encountered when trying to communicate via UART without first sending some “test” bytes. This buffer then keeps track of any data sent from MATLAB in real-time upon keyboard input. Next, main.c has a sound\_module(distance, lowPassIndex, direction) that is called any time the buffer changes and implements the changes necessary. These changes involve manipulating the leftChannelData, rightChannelData sent to the LINE OUT in YourISR.c. Delay is implemented using a basic buffer as in the preparatory labs. Frequency filtering is implemented using convolution defined in C as practiced in the preparatory labs. Intensity manipulation is achieved through simple scaling of the output.

# Results/Discussion

## Surround Sound Concepts

This project successfully demonstrated that surround sound can be achieved using stereo hardware. Some specific parameters could likely be tuned to improve the effectiveness, but overall the effects were noticeable and recognizable.

## Success of Real-Time Audio Processing

With limited hardware power available from the FPGA, the challenge becomes to keep real-time requirements without latency. By putting the bulk of the work on MATLAB to detect user input, computation was significantly reduced on the board. All the board needs, is the three parameters sent from MATLAB. It must then perform up to several convolutions and buffer effects, but it was ensured that these remain in fairly constant time. The finished product reacts very quickly to adapt to user input to create and immersive experience.

## Future Work

Given the limited time allotted to research and development of this project, there are a number of improvements that could have been incorporated, but were neglected due to time constraints.

One such improvement is to have better calibrated sound\_module on the board. User feedback indicated that the localization effects attempted by surround sound were not distinct enough. This is likely due to the variety of other distractions presented when in game, such as the jungle noise. However, it was duly noted that perhaps larger delays are needed to accurately convey directional information. Similarly, the naïve low pass filter approach to making a sound appear as if from behind the user is not ideal. More study of head-related transfer functions is needed to design a proper envelope to model this type of sound.

Another area that needs some attention is the lowPassIndex variable in MATLAB. Simply tying this index to distance was a greatly simplified assumption. In future work, this algorithm should take into account the number of walls between the user and the sound source and also the shortest path between the two. If there is a short, direct path between source and observer, sound in nature can easily propagate through this path (reflection off the walls, etc.) in a manner such that it should not be very low passed, and the observer still retains most of the emitted signal.

Feedback from users also indicated that the ability turn and face different directions if confusing. While this feature does not have to be utilized, it was intended to be useful for staying on one position and simply changing direction to localize sound. To be more accurate to real life, the maze should ideally be three dimensional and place the user as if they were on their feet, navigating the maze. Two dimensions greatly simplified the algorithms and GUI, but three dimensions would make it a more immersive experience.

A final notable improvement is automatic maze generation. Currently, the maze stays the same throughout instances of the game – only the sound source location changes. Algorithms to generate mazes of a specified size already exist and are well-defined. Incorporating such a feature would make the game of better replay value and test the robustness of the algorithms used and if they scale properly.

# References

[1] Brian Shannan, “Audiology-A Curriculum for Excellence”, 25 Nov 2010. Web. 30 May 2015.   
<http://www.ssc.education.ed.ac.uk/courses/deaf/dnov10i.html>

[2] Youtube Audio Library. Web. 30 May 2015   
<https://www.youtube.com/audiolibrary/music>

[3] MathWorks-Instrument Control Toolbox. Web. 30 May 2015.  
<http://www.mathworks.com/products/instrument/supported/serial.html>

[4] Inverse-square Law, Wikipedia. Web. 30 May 2015   
<http://en.wikipedia.org/wiki/Inverse-square_law>

[5] Conor O’Sullivan “A Formulation of Holophonic Display”, 5 July 2012. Web. 30 May 2015. <https://www.academia.edu/2931477/A_Formulation_of_Holophonic_Display>