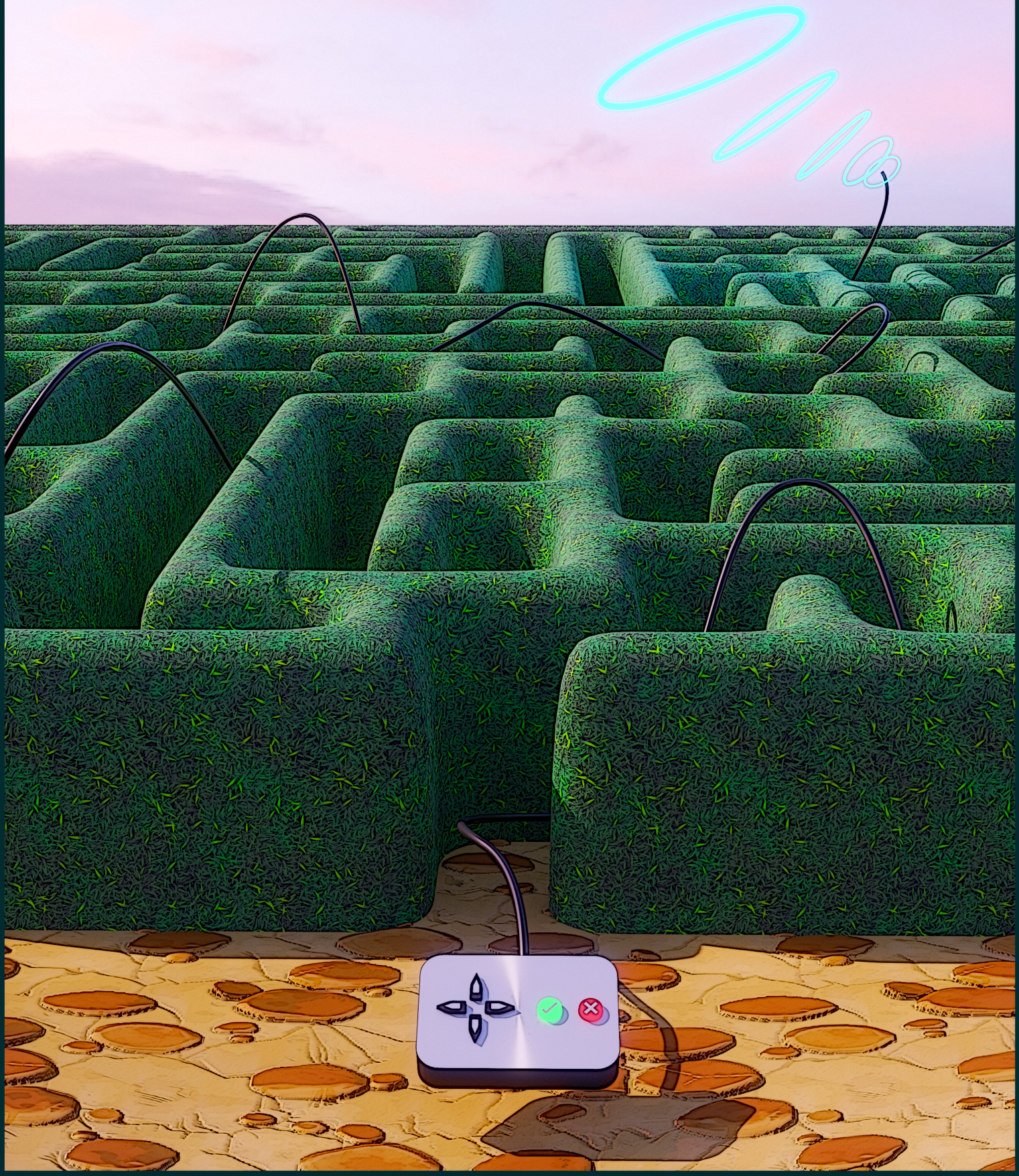
THE MAZE



The Maze

An audio guided computer game

Asmus Eg Nielson Andresen, Daniel Claes Thiesen, Elizabete Avotina,
Emil Linddahl Christensen, Marc Bisgaard Petersen, Scott James Naylor
Supervisor: Sofia Dahl

Medialogy group 406, Copenhagen, 2020



Copyright © Aalborg University Copenhagen 2020



The Technical Faculty of IT and Design
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
The Maze

Theme:
Sound Computing and
Sensor Technology

Project Period:
Spring Semester 2020

Project Group:
406

Participant(s):
Asmus Eg Nielson Andresen
Daniel Claes Thiesen
Elizabete Avotina
Emil Linddahl Christensen
Marc Bisgaard Petersen
Scott James Naylor

Supervisor(s):
Sofia Dahl

Copies: 1

Page Numbers: 149

Date of Completion:
May 28, 2020

Abstract:

This project focuses on how elements of audio processing in combination with nudging can create audio cues that assist a user navigating a maze game. After analysing maze design, performance, nudging, audio processing and state of the art, a maze game was implemented. It was done using Unity in combination with Wwise. The final implementation was tested online in an A/B test. The results were that the null hypothesis could be rejected in favour of the alternative hypothesis, which stated that the audio cues improved the players performance in the game. The final problem statement was therefore answered to a satisfying extent.

The content of this report is freely available, but publication (with reference) may only be pursued due to agreement with the author.

Contents

1 Introduction **3**

2 Analysis **5**
2.1 Labyrinths and mazes 5
2.2 User Performance 8
2.2.1 Problem solving 8
2.2.2 Navigation, Path Length & Time 9
2.3 Nudging 9
2.4 Physical Interface Design 12
2.4.1 Game Controllers 12
2.4.2 Interaction with a personal Computer 14
2.4.3 Micro-controller 14
2.4.4 Electric Circuits 15
2.5 State of the art - audio games 16
2.5.1 A blind legend 17
2.5.2 Maze Sounds 18
2.5.3 Sound Labyrinth 18
2.5.4 Conclusion of SOTA 19
2.6 Audio Processing 20
2.6.1 Audio 20
2.6.2 Earcons 34
2.7 Delimitation 36
2.8 Design requirements 37

3 Methods **39**
3.1 Agile 39
3.2 Hypothesis 39
3.3 Usability 40
3.4 Descriptive Statistics 40
3.4.1 Normal Distribution 40
3.4.2 Standard Error of the Mean 40

3.4.3 Variance . . . 40
3.4.4 Independent and Dependent Variables . . . 41
3.5 Parametric Statistics . . . 41
3.5.1 Shapiro-Wilk Test . . . 41
3.5.2 Levene's test . . . 41
3.5.3 T-Test . . . 41
3.5.4 Non-Parametric tests . . . 42
3.6 A/B Testing . . . 42
3.7 Likert scale . . . 42
3.8 Data Collection Methods . . . 42
3.9 Don Norman's design Principles . . . 43
3.10 Gestalt Principles . . . 44

4 Production **47**
4.1 First iteration . . . 47
4.1.1 Design . . . 47
4.1.2 Implementation . . . 60
4.1.3 Panning with Unity . . . 61
4.1.4 Evaluation - first test . . . 69
4.2 Second Iteration . . . 72
4.2.1 Design . . . 72
4.2.2 Implementation . . . 77
4.2.3 Evaluation - Usability test . . . 88
4.3 Third Iteration . . . 91
4.3.1 Design . . . 92
4.3.2 Implementation . . . 94
4.3.3 Evaluation - Final test . . . 96

5 Discussion **111**
5.1 The final test . . . 111
5.2 Physical user interface, controller . . . 113
5.3 Future perspective . . . 114
5.3.1 Considerations regarding the audio design . . . 114
5.3.2 Considerations for expanding the game to other platforms . . . 114
5.3.3 Considerations regarding the maze design . . . 114
5.3.4 Considerations regarding the User Interface . . . 115
5.3.5 Final considerations . . . 115

6 Conclusion **117**

7 Appendices **119**
7.1 Appendix 1 119
7.2 Appendix 2 124
7.3 Appendix 3 131

Bibliography **145**

Chapter 1

Introduction

Audio is essential for humans as it is one of the main senses humans experience the world with, and a huge part of the attention is devoted to it. For example, sudden sounds make people react differently than ambient ones. Humans are used to having sounds around them every day and becoming familiar with certain sounds. Some industries use sound or audio to control the attention or nudge the behaviour in a certain direction. Industries such as: film, music, marketing, video games, even board games and the list goes on, include audio and sound. Audio can indeed be a huge factor in the way humans make decisions [33].

In the video gaming industry for example, audio is often used to improve the performance of a player. Can all aspects of the performance be equally improved? This is one of the many questions that arise when considering how audio affects performance.

This paper aims to use audio to guide the user in an effective manner while performing a specific task in a computer game. Humans navigate certain situations by locating the source of a sound. They do this by comparing the characteristics of sounds arriving in their hearing mechanisms [60]. This means, that humans are capable of walking a path between two points, solely relying on their auditory perception. However, if faced with the challenges between the two points, which are physically constraining their movement, such as walls and paths, the level of difficulty to complete the task becomes much harder. An example here could be as follows - if one would be asked to solve a labyrinth or a maze in a complete darkness, where visuals could no longer be used to navigate, person might pay a bigger attention to the audio available to navigate themselves. In games that focus on solving labyrinths and mazes, players are often provided with the possibility to see an overview of the environment. That gives the player visual guidance on how to solve it. The performance of solving this specific task greatly depends

on the visuals available. If the overview was restricted to the player's immediate surroundings, it would be more difficult to solve relying solely on visuals. An environment such as this could be used as a way to test the effectiveness of audio guidance. In such a scenario, could audio cues increase the player's performance? Based on this question, the following initial problem statement has been proposed: *"How can audio cues be used to improve the performance of a player navigating through a labyrinth or maze game?"*

Chapter 2

Analysis

The analysis will look into subjects related to how a user can be guided through the process of solving a labyrinth, and how the user performance can be improved. This chapter will look into labyrinths, performance and nudging elements. Then physical interface design will be analysed and state of the art solutions for audio based games will be looked at. A major part of it will be devoted to audio processing, filters and techniques of it. This will all be used to gain the necessary knowledge for proposing design requirements for later design and implementation of a game, including the evaluation. This chapter will be concluded with a delimitation leading to the final problem statement (FPS) and the design requirements for the first iteration of the design.

2.1 Labyrinths and mazes

To understand how labyrinths and mazes are designed and how they can be solved by a user, this section will cover the comparison of labyrinths and mazes. It will furthermore list the properties of mazes and labyrinths and the possible strategies of solving them.

The main purpose of labyrinths and mazes are to delay the person solving these from a starting point A, to end point B. This is achieved by stimulating their curiosity and providing the person with surprises and/or puzzles along the paths. The delay is based on the overall length of the path from the starting point to the end point. In mazes and labyrinths the person walking is restricted by walls or hedges which only allows movement in certain directions [21].

The classic labyrinth consists of one single winding path that folds within itself [29]. Meanwhile, mazes are more complicated compared to labyrinths. They include many ways in which they can be traversed. Paths are branching and dead ends are included, thereby making it harder to choose the right direction and complete the maze [29]. A comparison between the layout of labyrinths and mazes can

bee see in the Figure below 2.1.

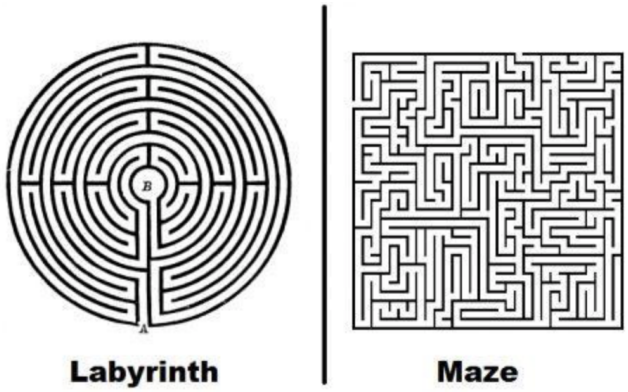


Figure 2.1: Labyrinth and maze - top down view

Based on these observations, mazes are more of a challenge to solve than labyrinths, and therefore the following analysis will focus on mazes.

The maze designer, Adrian Fisher, has been designing mazes all over the world since 1979. The mazes he designed did not only include regular hedge and wooden fences but also consisted of mirrors, light, water and more. Which means that mazes have virtually no limitations in how they can be constructed [23].

Properties of Mazes

The main properties and characteristics by which the difficulty of the mazes is measured will be listed below [36].

- **Simply connected/Complex** - a simple maze would be one in which it is possible to visit every point of the maze by simply following one wall. Which simultaneously means that this type of maze does not include any loops.
- **Number of dead-ends** is a measure by which one can tell how many locations or paths branching there are with only one way in.
- **Length of longest path** is measured as percentage. The optimal path of solving a maze is found first and then the ratio is calculated between it and the longest path.
- **Convolution** is measured as "twistiness" or straightness. It is a measure that counts how many times one has to make a turn in order to solve the maze. It is easier to find the correct path if fewer turns have to be made.

- **Distribution of Valency** - for instance, the dead ends have a valency of one - one way in, the same way out. Crossroads have a valency of 4, unbranched corridors - 2, T-junctions - 3. The more options to turn, the more complicated the maze is.
- **Complexity** measures the average number of decisions a solver has to make in order to go from the point A to B in the maze.

Solving Mazes

While there are many algorithms that are capable of solving mazes, the main and most relevant methods on how different types of mazes can be solved by humans (either with pen and paper or actually being in the maze) will be covered in the following paragraphs.

- **Eliminating Dead Ends** is a method which can be used if the maze can be seen from above. This is possible if one for example, solves the maze with pen and paper [37]. It is done by eliminating the paths which have dead ends.
- **Right Hand Rule** also called **Wall follower** is a solving method that can be used if all walls are connected together or to the outer boundary. This makes it possible to solve the maze by turning right every time there is a branching path. 2 A simple illustration can be seen below 2.2.

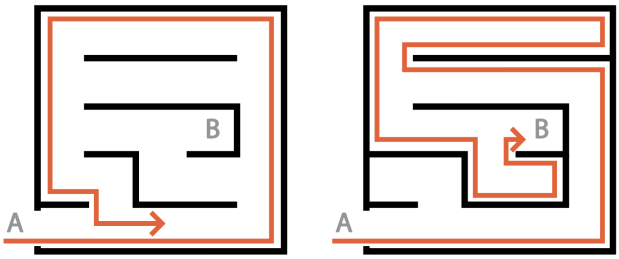


Figure 2.2: Comparison between how a simple maze can be designed in 2 ways - first without and second with the possibility for the user to use the right hand rule [38]

To conclude, mazes are more challenging for the user to complete when they include branching paths, dead ends and different design strategies to prolong the path from point A to point B. Different properties are used to describe different types of mazes, overall, they are used for determining how difficult it is to complete a maze. As for solving a maze, there seems to be some strategies which works for some mazes but not all types of mazes. Next performance will be analysed to gain an understanding of how it can be measured.

2.2 User Performance

The following section and its subsections will explore how user performance can be measured when asked to perform a problem solving task.

2.2.1 Problem solving

Problem solving is a concept that is explored in many branches of sciences such as psychology, cognitive science, computer science, logic, engineering and many more. The concept refers to a situation in which a person, for example, is faced with a problem or task that the person is asked to solve in order to gather some kind of data about how the person achieved the solution [49]. According to Wang et al. in their paper *On the cognitive process of human problem solving* [57] there are various approaches one can take in a problem solving situation, these include:

- *Direct facts – finding a direct solution path based on known solutions.*
- *Heuristic – adopting rule of thumb of the most possible solutions.*
- *Analogy – reducing a new problem to an existing or similar one for which solutions have already been known.*
- *Hill climbing – making any move that approaches closer to the problem goal step by step.*
- *Algorithmic deduction – applying a known and well defined solution for a problem.*
- *Exhaustive search – using a systematic search for all possible solutions.*
- *Divide-and-conquer – solving a whole problem by decomposing it into a set of sub-problems.*
- *Analysis and synthesis – reducing a given problem to a known category and then finding particular solutions* [57]

Through the research of problem solving, an exciting task popularized in the 1800s and 1900s field of psychology came to light. The use of mazes to test spatial awareness, memory, reasoning and more in humans [26]. It therefore makes sense to combine some aspects of the approaches above with maze design in an effort to present the user with a task which has variables that can be measured in the form of performance of the user. Mainly the task should focus on the first approach, direct facts, which is presenting a path for the user, and most importantly the hill climbing approach, which should give participants the possibility to move closer to the goal as they move along [57].

2.2.2 Navigation, Path Length & Time

Navigation is an inherent part of mazes and relates to the concept of moving from a starting point - A to a finishing point - B. Depending on the size of a maze and the amount of errors made by a user, the path length and time consumption to complete a maze can vary. The error-score can be said to be the amount of errors, or wrong turns a person makes from point A to point B, a reason not to measure these is that they depend on the participants exploratory nature which isn't part of the problem solving approaches that was chosen above. The time and path length variables are interesting variables to test because they make it easy to compare test participants against each other [46]. Furthermore, if test participants have to rely on various audio cues to solve a maze, it will be relevant to test these variables in an effort to measure the effectiveness of the cues in groups that have them available and groups that do not.

2.3 Nudging

Earlier in the analysis, performance was explore. Not only is it important to know how performance may be improved, it is also crucial to know how a desired action by the user can be achieved. Therefore, this section will explore nudging and its mechanisms.

Thaler and Sunstein [52] introduced the term Nudging in order to emphasize the idea that a *systemically biased decision making process* can be leveraged, which means that human behavior can be altered in predictable ways. The definition of nudges is as follows *"any aspect of the choice architecture that alters people's behavior in a predictable way without forbidding any option or significantly changing their economic incentive" [52].*

Mechanisms of nudging

In the following section, mechanisms of nudging by Caraban et al. [11] will be discussed. Based on their research paper *"23 Ways to Nudge: A Review of Technology-Mediated Nudging in Human-Computer Interaction."* six different categories of nudges can be considered: facilitate, confront, deceive, social influence, fear, and reinforce as seen in figure 2.3.

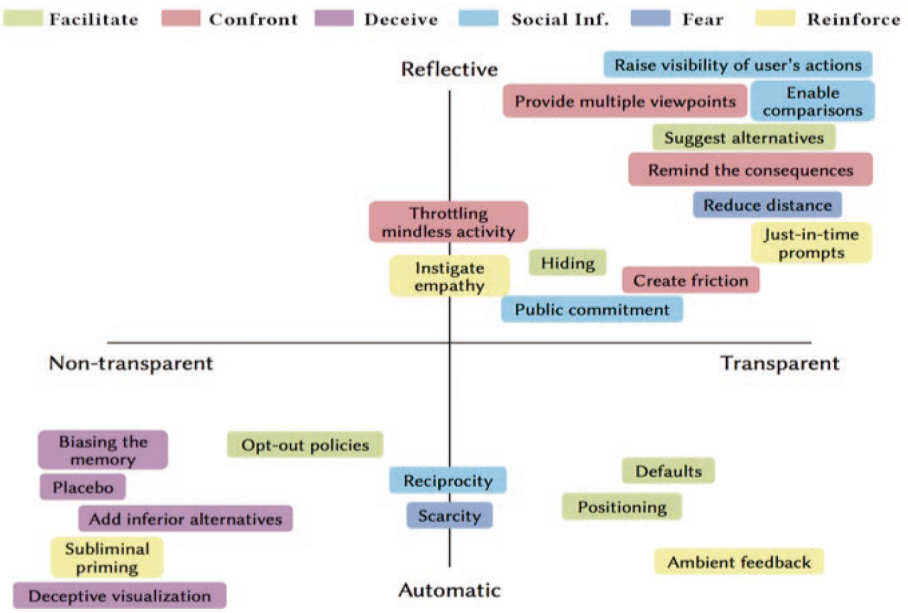


Figure 2.3: Nudges positioned along the transparency and reflective-automatic axes [11]

Facilitate nudges alters decision making in a way which diminishes physical and mental effort to their possible extents. These nudges can be broken down into smaller parts:

- *Positioning* is about the visuals and placement of the information;
- *Hiding* is the opposite of Positioning. It tries to hide the unwanted action and makes it harder to reach the option or becomes completely unreachable;
- *Suggesting alternatives* is a tactic in which people get offered alternatives to whatever they have chosen beforehand [11].

Confront nudges are attempting to delay an undesired process by leaving space for doubt. People tend to become more thoughtful of their decisions if there is a certain risk involved in their decision making [18].

- *Throttling mindless activity* can be used by introducing a time buffer before making a decision. It can delay a process or reverse the initially desired action [11].
- *Reminding of the consequences* are nudges which prompt reflections on possible consequences after performing an action. As an example, users can be shown a pop up message before they try to exit a game;

- *Providing multiple viewpoints* can be used to avoid a biased situation, as humans tend to pay less attention or even reject newly introduced information if it does not fit with their own reasoning or beliefs. Therefore, providing multiple viewpoints can be used to nudge people into considering other points of view.

Deceive This category's nudges are using deceptive mechanisms for influencing how alternative choices are perceived, while aiming to promote specific results [11].

- *Adding inferior alternatives*. People have a tendency of preferring the choice when an alternative is added to the original set. For example, if the paths of a maze are branching, the user has to compare them with the alternative choices;
- *Biasing the memory of past experiences* choices might be made depending on the person's previous experience [11]. For example, if a player in a video game notices that a positive sound cue is related to making a choice, they will continue doing that, based on the previous experience.
- *Placebos* The placebo effect suggests that providing an element which has no effect upon one's environment [11] might still influence the user's decision.
- *Deceptive visualizations* People tend to keep a focus on items which they believe are more important rather than the ones which seem to be less important [55]. In a maze for example, small shrubberies may be placed in random locations throughout, these would serve no purpose at all other than making the user question their presence.

Social Influence are nudges which makes people conform their behaviour to fit what is believed to be expected [11].

- *Invoking feelings of reciprocity*. Reciprocity bias can be used to make people reciprocate actions they have just received [15].
- *Leveraging public commitment*. The commitment bias is a tendency for humans to uphold commitments they have publicly made [51]. If the user were to play the game in a group, competing with other players in real time, the user might feel the urge to finish the game.

Fear nudges are defined as *"nudges that invoke feelings of fear, loss and uncertainty to make the user pursue an activity" [11].*

- *Making resources scarce* is a possible way to make people feel like the availability of a certain item or resource seem limited. This puts a higher value on things in the present, for instance a time limit in a game [15].

- *Reducing the distance* It is difficult for people to stay committed to self-beneficial activities if the results are in a the future, especially in a far future. Nudges in this category aim to make the gap between the action and the results seem smaller [54]. For instance, if the user can see a progression bar, they might feel that they are approaching the end.

Reinforce nudges are used to *"reinforce behaviour through increasing their presence in individuals' thinking" [11].*

- *Ambient feedback* tries to reinforce behaviour in an ambient manner, reducing disruption while the person is performing an activity [11]. Here, for example, audio cues could be used to nudge in an ambient manner enough for users not to get distracted but still notice them.
- *Instigating empathy* allows people to learn by discovering and reflecting upon their own personal experiences, which might influence their decision making [11].

To sum up, nudges are effectively used for altering users attention, decision making process or shifting their behavior. The specific nudges which could be useful for later design are as follows. From the *Facilitate* nudges, *Positioning*, *Hiding* and *Suggesting alternatives* can be useful for designing the visual environment. Which might minimise how the visual environment could nudge users. *Confront* nudges - implementing audio cues which alters the users attention, nudging them in the desired direction. *Deceive* nudges - using what has happened to the user in the game already, could have an effect on how users make their next choice.

2.4 Physical Interface Design

For humans to be able to communicate with any digital device, some sort of an interface is needed. The following section will explore various physical interfaces and how intuitiveness can be achieved. In the context of this report, which is to increase the performance/navigation in a maze game, a controller of some sort should be used to control the user's movement throughout it.

2.4.1 Game Controllers

When considering design principles for game controllers, there are three main companies which are the front-runners for the design of game controllers. These companies are Sony with their PlayStation controller also known as the DualShock controller [50], Microsoft with their Xbox controller [39] and Nintendo with their Nintendo switch controller [40]. All of their controllers have the inputs listed below as part of their design in common:

- Directional pad;
- Two analog sticks;
- Two triggers and bumpers;
- Variations of digital buttons for action buttons;
- Pairing system through Bluetooth.

Directional pad or D-pad is normally being represented by arrows on a plus-sign layout where arrows are pointing in four directions (forward, backward, left, right). This is used for interacting with games and apps.

Analog stick is a small joystick that can move in any direction except up and down. This can have the same functionality as the D-pad to interact with games and apps on the game consoles. They can also be pressed down and function as a button.

Bumpers are placed on the top of the controller where the index fingers can press them. These can be used to jump between menus in games or apps, otherwise they have varied functionality depending on the game or app.

Triggers are placed below the bumpers and are most commonly used in games.

Action buttons are four buttons that interact with games and apps. They are being represented as Y,X,B,A for the Xbox controller and the Nintendo switch controller, while the DualShock uses triangle, circle, cross and square for theirs.

The two controllers that have other inputs are the DualShock [50] and the Nintendo Switch controller [40]. These two controllers both implement an accelerometer and a gyroscope.

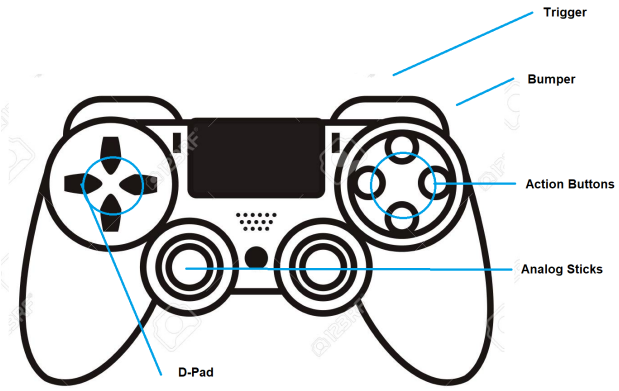


Figure 2.4: A Typical Button Layout [59]

Accelerometer is a sensor that can be found, for example, in the DualShock, Nintendo Switch controller and various smartphones, where it detects forces like

gravity, movement and vibrations. Additionally, change of velocity or speed over time can be detected [25]. This would be an excellent choice if the project, for example, needed to detect similar movement to swinging a bat, but using this component would not be intuitive for moving a character around for a user.

Gyroscope is a sensor that is measuring or maintaining orientation. This can be used to detect whether the controller is tilted in a given direction or if it is upside down, for example [24]. This could be used for movement of the character by tilting it in the direction the user wants to go in.

2.4.2 Interaction with a personal Computer

For interaction with a personal computer a keyboard and mouse is commonly used since the keyboard was made for writing and the mouse for navigation on the computer. On the keyboard, the movement buttons that are commonly used in games are "WASD" and the arrows keys while the mouse enables actions such as camera control and performing various actions in games and apps. With the number of buttons on the keyboard and the smooth movement of the mouse input, these can provide functionality for a wide range of games and apps. [16].

Relevant outcomes from reviewing popular game controllers

Since this project will be focusing on a maze game, the preferred input components would be the D-pad and action buttons. The gyroscope and accelerometer is still fairly new when it comes to input components for games. They have seen success with the switch controllers in certain games where the game mechanics were built around moving the controller. Triggers and bumpers are equivalent to action buttons but for the index and middle fingers, since the function of these change depending on the media. The D-pad and default action buttons share similar functionality throughout many games. The analog stick could be considered if movement in every direction is needed, since the D-pad is limited to a four-directional buttons. Micro-controllers, explained below, are used to read the input of these components.

2.4.3 Micro-controller

A device is required to communicate with these components. Such a device would be a micro-controller. The one micro-controller the group was taught through the physical interface design course was the Arduino Uno. The Uno is a versatile micro-controller with 20 digital and 6 analog pins that can be connected with various components and operates on 5v with 20 milliamps (mA)[4]. To implement using an Arduino Uno knowledge of electrical circuits along with resistors, voltage and current is required.

2.4.4 Electric Circuits

An electric circuit includes components connected together which supply a route for the flow of electricity [58]. For an electric circuit, a power supply is required to continue the flow of electricity through the circuit. To get a better understanding of what is in the circuit, it is needed to understand what voltage, current and resistance is.

Voltage is the measurement of potential energy between two points and is measured in volts(V);

Current is the continuous movement of flow through the circuit and is measured in amperes(A);

Resistance is a measure of the opposition to the flow of current in a circuit and is measured in ohms.

Voltage, current and resistance are used in Ohm's law which will be explained next.

Ohm's Law Ohm's law is used to describe the relationship between voltage, current and resistance, it is possible to see their mathematical relationship, which can be seen below in figure 2.5.

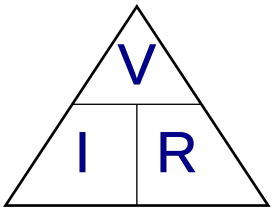


Figure 2.5: Ohm's law triangle representation [42]

if the voltage needs to be isolated, the current and resistance need to be multiplied like this:

$$V = I * R \tag{2.1}$$

In the equation(2.1) for Ohm's law, E is the voltage while I is the current and R is the resistance. Then depending on which of the three (volts, amperes or ohms) need to be found, it can be done with this law, if two of the values are known.

Kirchhoff's Law Kirchhoff's law is split into two laws, one for voltage[32] and one for current[31]. The law for voltage states that the sum of all voltage in a loop must be equal to zero. What this law is proving, is that when all of the voltage is

equal to zero, there is not any unnecessary energy going to be wasted. The law for current is that the sum of all current entering and exiting a node (junctions) should remain the same. Meaning the current entering a component, would come out in the same amount.

This is illustrated with the images below:

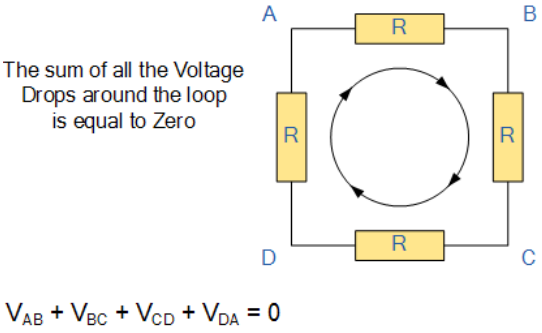


Figure 2.6: Kirchhoff's law of voltage [30]

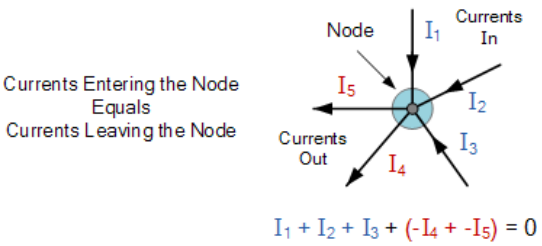


Figure 2.7: Kirchhoff's law of current [30]

After reviewing and analysing the electrical circuits and the laws of Ohm and Kirchhoff, the knowledge gathered should prove valuable for later implementation.

2.5 State of the art - audio games

The following section will explore the state of the art for games in the field of audio only and/or maze genres. Furthermore, how are the games designed and how do they guide the player. These observations will results in some considerations in regards to how the project can learn from these ideas and implement similar strategies into the game. Finally, based on this section design requirements should be able to be formed and outcomes taken for the later design iterations.

2.5.1 A blind legend

This is a collaborative project of an audio-only action/adventure game for mobile phones. Binaural sound is used for the player to understand what is happening in the game and how to navigate further. The only guide provided is the 3D sound, the control over the character in the game is by multi-point tactile gestures. The game is aimed not only for visually impaired people but anyone who want to enjoy an original sensory experience [17].

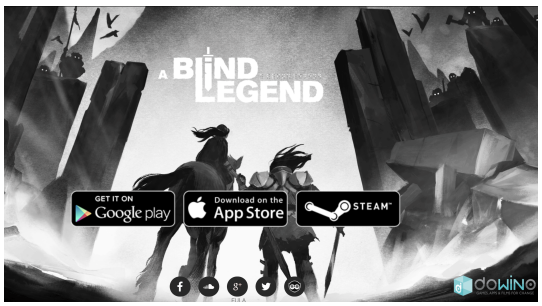


Figure 2.8: Cover photo for the game "A blind legend"



Figure 2.9: In-game screenshot

The game itself is using a visually simple, dark screen, with some minor graphics. Whenever the player is in control of the character, a circle with 4 arrows appear on the screen, which gives some visual feedback, besides the audio itself, indicating that the player is moving in the game environment. Different adventurous happenings can be heard while interacting with the game. Interaction is mainly

done by sliding fingers in the desired direction of which the character should be moving towards. Faster and slower sliding has an impact on the action performed within the game.

Despite of the music, noise and background conversations in the game, it is still possible to distinguish the guiding sounds. These sounds combined lets the user understand in which direction they are moving. The voice of a girl is used to distinguish between other audio and guide the player in the right direction. A deeper voice is used for instructions and basic information to understand the rules of the game and the options available.

2.5.2 Maze Sounds

Maze Sounds is a first-person puzzle game where the player is forced to focus on the sound in the game. When playing, the player is presented with a floor that is filled with tiles with distinct edges. It is up to the player, to cross the field of tiles to reach the opposite side without loosing the game. Each tile has an invisible electrical fence on two of the edges and the fences are lethal to the player. It is up to the player to listen for the noise the fences are producing to avoid them and progress in a maze filled with these "lethal walls". The noise can be heard in four directions (in-front, behind, left and right) this guides the player while listening for the noise.

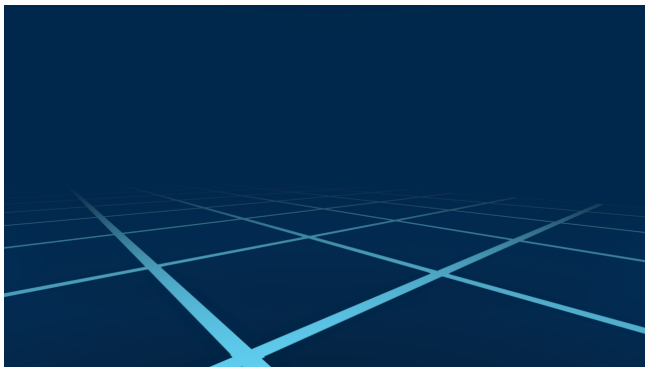


Figure 2.10: In-game screenshot of Maze Sounds

2.5.3 Sound Labyrinth

Audioversum, an interactive museum in Innsbruck, Austria, is provides an interactive sound experience. They have developed a 3D mixed-reality labyrinth which is solved by wearing mixed reality glasses. To outsiders, it is an invisible labyrinth, guided only by sounds [8]



Figure 2.11: Press photo of the labyrinth, developed by Audioversum

People interacting with the labyrinth must be wearing the Microsoft HoloLens headset [27]. When the user makes noise, such as speaking or clapping their hands, they create holographic sound waves which bounce off the virtual walls and makes them visible momentarily. The aim is to complete the labyrinth within 3 minutes, as that is the maximum time for one game. There are points which can be collected along the way, however, colliding with virtual walls sets off an alarm and causes the player to lose points.

Overall, the approach of showing the walls only for a moment and only when the user creates sound, is another way of tricking the user and prolonging the process with another type of restriction.

2.5.4 Conclusion of SOTA

An important takeaway from the game "A blind legend" is the use of binaural sounds and the simplistic use of a graphical user interface to visualise what is happening in the game. It is a experimental and different approach to audio focused games. It makes the user pay more attention to the audio than what is visualised 2.5.1.

"Maze Sounds" is a game which limits the amount of visuals and relies on having distinguishable guiding sounds used to convey information about the environment to the player 2.5.2.

"Sound Labyrinth" is a real life holographic labyrinth in which people have to move around and create their own sound in order to complete the game. The game implements the use of a physical interface, namely the Microsoft HoloLens

2.5.3.

To gain an understanding how audio can be processed, next section will be devoted for analysing topics related to the sound design, audio cues and filters.

2.6 Audio Processing

The following section will explore how various audio techniques can be used to manipulate the audio in the game. The main purpose of this section is to figure out which audio filtering and processing techniques that can be used. This will be done in order to gain knowledge on how to guide a player through a maze in a proper manner. First the definition of audio will be discussed, as well as its fundamental and physical properties which will help to understand how filtering and manipulation of audio works.

2.6.1 Audio

In order to create and manipulate sounds, an understanding of basic audio processing is needed. This section will therefore explain the fundamentals of sounds and how it can be used in the context of the initial problem statement. Audio is waves of pressure, molecules being pushed and pulled, oscillating, changing in various mediums like air and propagating outwards from a source like a speaker or an instrument for example. This makes the air molecules move quickly back and fourth, vibrating at certain speeds, often repeating these pressure changes, described as cycles. The sound waves all have physical and perceptual elements. The amount of cycles completed per second is known as frequency and is heard as the pitch of a tone. The distance the molecules move from its original position is called the amplitude and is the loudness of the sound [60]. The frequency or pitch is measured in hertz (Hz) while the amplitude or loudness is measured in decibels (dB). Sound waves that have these repeated vibrations in a regular way are called periodic sound waves which are often produced by instruments. Apposed to aperiodic sound waves that are produced by quick events like thunder or ocean waves breaking. These sound can also be described as noise. A *sinusoid* however, can be described by a mathematical formula and is a pure tone which vibrates periodically. It is this mathematical formula that makes it possible to manipulate or change sounds. For example, increasing the volume of the sound by increasing the amplitude in the formula [12]. The standard sinusoid looks as follow:

$$x(t) = Acos(\Omega t + \Phi) \tag{2.2}$$

In this formula, A is the amplitude of the sinusoid, the omega Ω is the frequency, while t is the time, and Φ is the phase which enables the sinusoid to be

time shifted.

Sampling

To store sounds digitally, sampling the original continuous signal into a discrete signal is required. Sampling is a process of recording the amplitude of a wave at a given time and the amount of times this happens per second is called the sampling rate. The Nyquist Theorem states that in order to correctly plot a wave, the wave must be sampled with more than twice the frequency of the original signal. If the sampling rate is lower than twice the frequency this can result in aliasing. Aliasing means that the recreation of the signal will be different from the original signal. When the sound has been stored digitally manipulation of the audio via filters and effects can be performed, before recreating it into a new signal [12].

Static and Procedural Audio

This section explores the similarities, disadvantages and advantages of both static and procedural audio. As the project is planned to be related with audio in computer games, games will be used as a medium for explanation. However, it should be noted that static and procedural audio can both be implemented into a vast range of multimedia.

Static Audio

Static audio is a simple type of audio that is implemented in a lot of games. It refers to recorded and possibly pre-manipulated audio. The word static describes this as it is not altered at the run time of the game. The main advantage of implementing static audio is the ease of implementation. Also, static audio can easily allow for stereo panning. It only requires one or sometimes an array of recordings for an event that happens multiple times throughout a game. The major disadvantage of using this type of audio is the lack of individuality of the audio it can become repetitive if played often. This lack of idiosyncrasy is not a true reflection of what humans are used to in everyday life [56].

Procedural Audio

Unlike static audio, procedural audio is synthesised and altered at the run time. For example, in figure 2.12 an Unreal Engine blueprint of an explosion can be seen. The individual sounds that make up the explosion are separated, and combined at run time randomly. This allows for much more variety in the audio without having to import many different audio files. The preparation and implementation of procedural audio is more time consuming than static one. However, for long

running games where a specific event occurs multiple times, implementing this type of audio can result in far superior results than static audio [3].

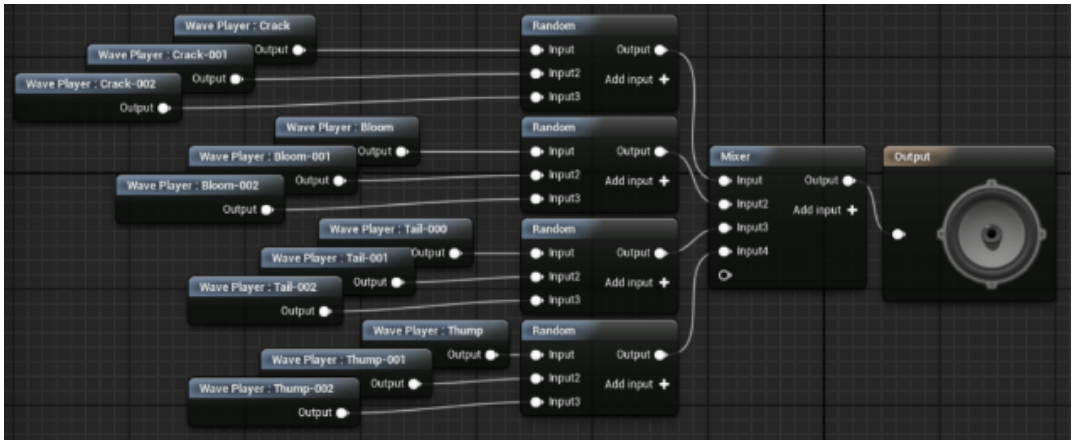


Figure 2.12: Procedural Audio System

Stereo Panning

Stereo panning is used to give directional sound in various applications. When using binaural audio, which means the audio is relating to two ears, a common practice is to use two speakers. For headphones, one speaker is on the left ear and one on the right. A commonly used method of panning is amplitude panning. This technique relies on changing the amplitude of a mono signal in each of the speakers to give the listener a perceived direction of the audio, illustrated in the figure 2.13 [13].

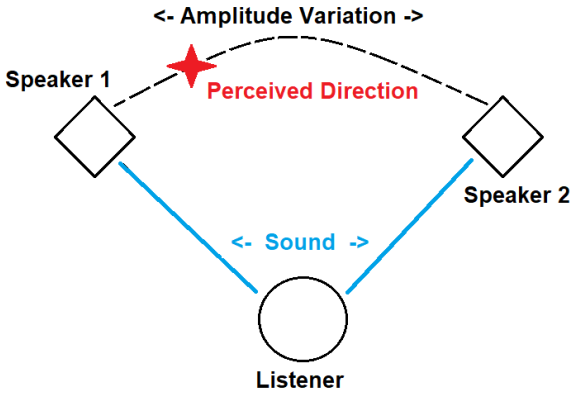


Figure 2.13: Amplitude Panning in a stereo setup.

Filters

Manipulation of audio frequently happens through filters. Once a signal is sampled it can be used as an input, mostly known as x_n, which when passed through a filter becomes a new signal called y_n, also known as the output signal, see the figure 2.14 [13].



Figure 2.14: Filter diagram.

This y_n signal can then be reconstructed and some change, depending on the filter, will have occurred.

Digital filters can be described by *difference equations*. These equations show the mathematical operation applied to the input signal, in order to change it to the output signal. This happens through addition and multiplication upon each sample of the input as well as the output signal. A simple difference equation can be seen in figure 2.15.

$$y_n = x_n + ax_{n-1}$$

Figure 2.15: Difference equation for a filter.

A difference equation can also be visualised through a block diagram. Below is a block diagram 2.16 shows a filter based on the difference equation above.

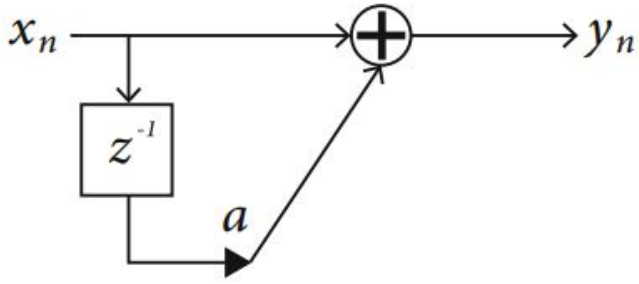


Figure 2.16: Block diagram for a system.

The block diagram shows the signal x_n which has its own signal delayed by one sample, x_{n-1}. Then, the delayed sample gets scaled by a so called *filter coefficient a*. The delayed version of the signal is then added back to the input sample

creating the output signal y_n. The block z^{-1} explains that the signal is delayed by one sample which is known as a unit delay.

The difference equation can also be converted into a *transfer function*. This function is the phasor representation of the input and output signal. This representation carries information about the amplitude and frequency of these signals. The transfer function can be used to explain what a filter will do to an input signal without knowing what signal is passed through it. By passing a phasor through the input of the filter and measuring the output, it is possible to divide the output by the input and the transfer function which has been calculated. Below in the figure 2.17 showing the transfer function, H(z) is systems transfer function, while Y(z) is the output and X(z) is the input [13].

$$H(z) = \frac{Y(z)}{X(z)}.$$

Figure 2.17: The transfer function.

If one replaces z in the function with $e^{j\omega}$ the frequency response of the system $H(\omega)$ is calculated instead. The frequency response describes how the magnitude and the angle of the phasor is changed by the filter. This change relates to how the response can be described on a unit circle. This description can show if the filter will attenuate or cancel out the signal depending on the frequencies applied to the system/filter [13].

Filters that only uses the input signal to create an output signal are called *feed-forward filters*, or FIR (Finite Impulse Response) filters. Filters that uses the output signal to compute a change, are called *feedback filters* or IIR (Infinite Impulse Response) filters. This relates to the fact that, for FIR filters, at some point, it will run out of samples to change, and so it is finite. IIR filters however, because it also uses the output signal, can in principle run infinitely.

Amplification and Attenuation

Attenuation relates to the dampening of the amplitude of an input signal while amplification has the opposite effect. In the real world, sound attenuates with distance. This happens due to the friction causing an amount of the sound waves' energy to be turned into heat. The degree to which this occurs depends on the

medium in which the sound wave is travelling [5]. Below, the digital implementation of this will be discussed.

Implementing Distance Based Digital Attenuation

Distance based attenuation is often implemented in games to add realism to the audio. This would be the systematic attenuation and amplification of the volume of a sound according to the distance the listener is from the source, see figure 2.18 [28].

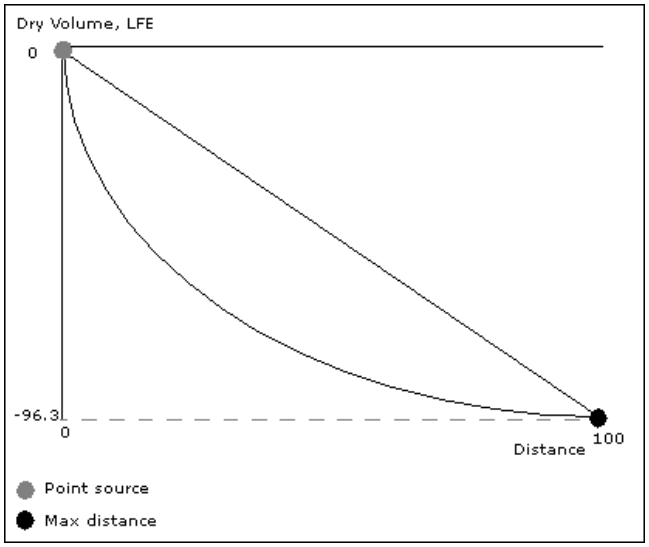


Figure 2.18: Example of an attenuation curve [28]

Attenuation is usually implemented as an inverse square relationship between volume and distance as seen in figure 2.18. Meaning that audio is attenuated exponentially with distance. To implement this from scratch, one could use the formula in figure 2.19 below.

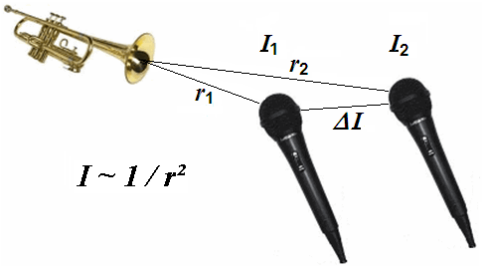


Figure 2.19: Intensity of sound in relation to distance, where I is the intensity of the sound and r is the distance from the source [47]

This is simple to implement and satisfies the purpose of attenuation for realism. However, the functional relationship between distance and attenuation/amplification does not always have to replicate the real world. The relationship can be taken advantage of by, for example, creating a custom attenuation curve, which makes the sound audible but quiet at large distances, which with an exponential roll off might have been be inaudible. Considering what has been discussed above, to create a distance based attenuation/amplification filter, one could use the filter seen as a illustration in figure 2.20 below.

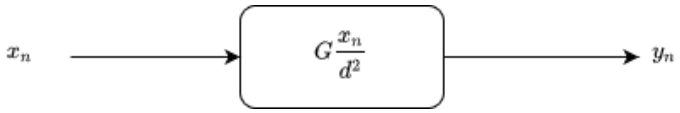


Figure 2.20: Filter diagram of attenuation filter

Where x_n is the input signal, d is the distance from the source, G is the gain used to increase or decrease the intensity of the filter and y_n is the output signal.

Reverberation

Reverberation, or reverb for short, is the persistence of audio in an environment after the production of audio from a source has stopped. An example of natural reverberation is singing inside a church, even when the choir stops singing, the sound bounces around the room until they have been absorbed by the material or been fully dissipated into heat. To understand how a signal can be artificially manipulated to mimic this effect, one must first realise that sound in a normal listening space, like a living room or church has an impulse response. That means the room responds to the impulse or sound propagating from the speakers within it. This impulse response can be divided into three categories - direct sound, early reflections and late reflections [13]. In figure 2.21 below, the audio reverberating over time is seen.

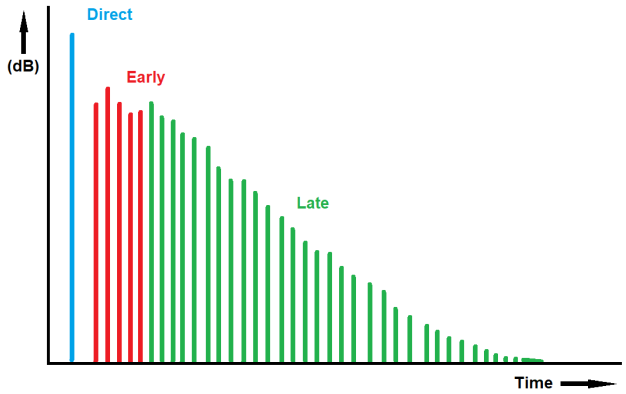


Figure 2.21: Impulse response - Audio decay.

As the names indicate, direct sound is coming straight from the sound source and into the listeners ear. Early reflections only bounce off a few objects, like walls before reaching the listener. The later reflections bounce around the room for a longer period of time before reaching the listener which results in these reflections being more noisy 2.22 [13].

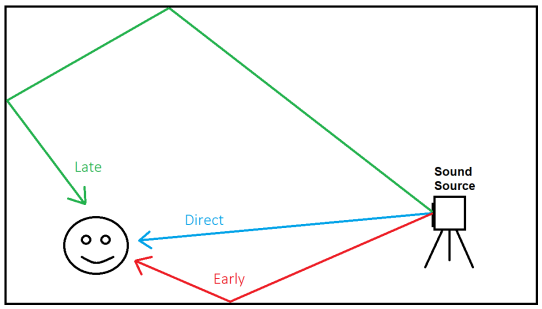


Figure 2.22: Direct, Early & Late reflections.

Artificial reverb is used to make dry audio sound more realistic in respect to the spatial acoustics that normally accompany sound. Usually when audio is recorded it will be recorded with a single microphone close to the object being recorded, with as little of the natural acoustics interfering with the recording as possible. This makes the audio sound dry and unnatural. Adding artificial reverb i.e. manipulating the digital audio signal with algorithms, will make recordings sound like they were recorded in a different location, a church for example, a concert hall or a professional studio. In other words, the spatial characteristics of the sound are being manipulated. One way of adding reverb to a signal is through convolution. To do this, one must know the impulse response of the room in which the recording was made, or the impulse response of a room in which one wishes the audio sounds like it was recorded in. The impulse response can be added to the origi-

nal signal to make it sounds like it was recorded in another location. To measure the impulse response of a certain space an impulse can be added to the input of a system and the output is then measured. In layman's terms one would play a short and loud sound in a room, record, then measure the sound propagating in the room and how it returns to a specific point [13]. Another way of adding reverb to a signal is through artificial reverberation algorithms. These algorithms mainly applies an echo effect on the signal in which each echo is attenuated to mimic reverb in various spaces. In other words the effect of reverberation is synthesized using feedback filters like a comb filter. There are two types of comb filters - feed forward and feedback filters .

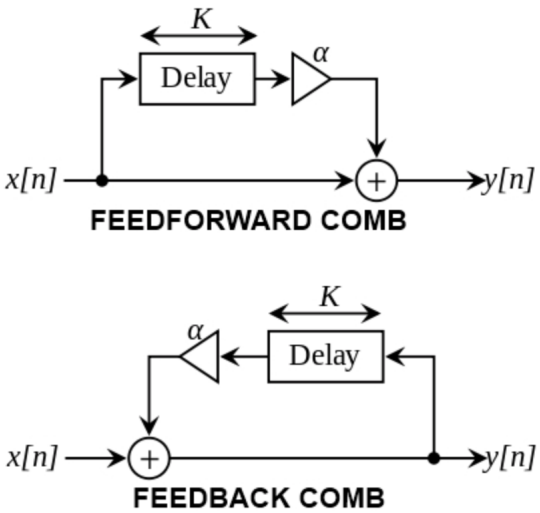


Figure 2.23: Graph illustrating processing in comb filters - feed forward and feedback [20]

To distinguish between them, not only the process of how they are applied to signals is important, but also the amplitude response. The amplitude response in a feedback comb filter has sharp peaks but soft dips, see figure 2.24.

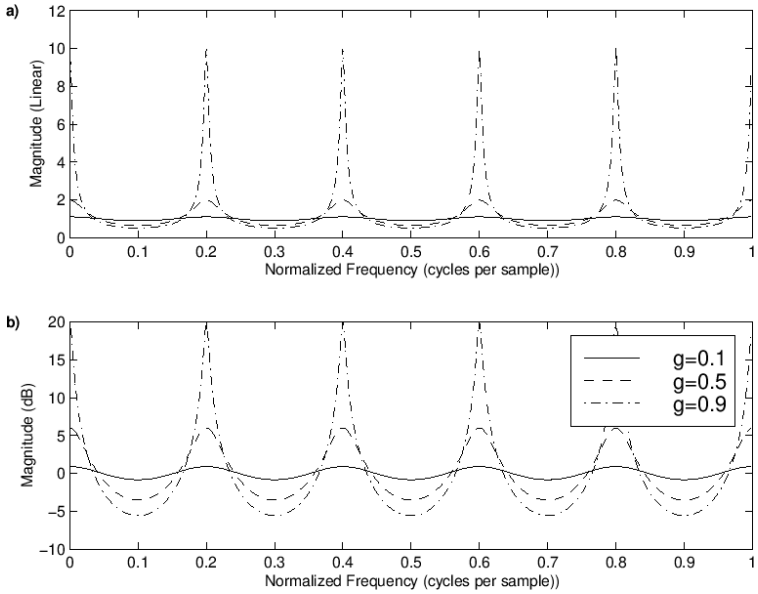


Figure 2.24: Amplitude response of the feedback comb-filter, a) Linear amplitude scale. b) Decibel scale

The amplitude response in a feed-forward comb filter has soft peaks but sharp dips, see figure 2.25.

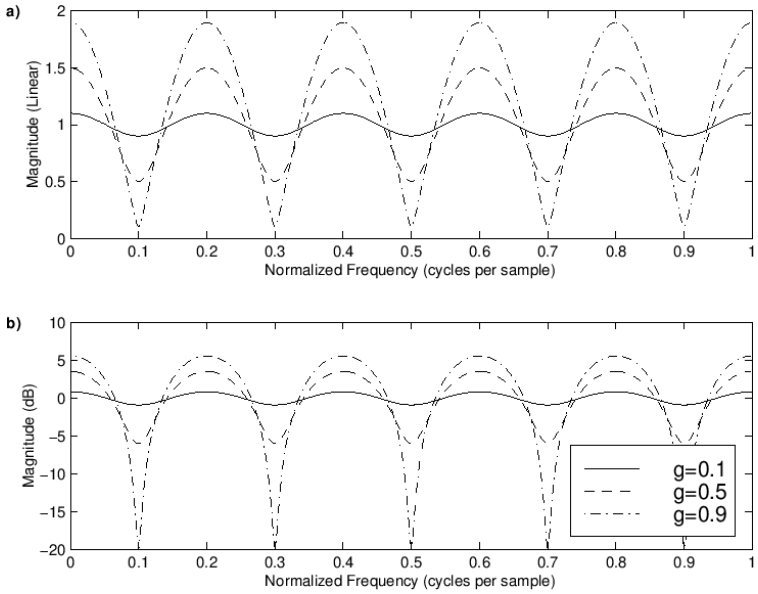


Figure 2.25: Amplitude response of the feed forward comb-filter, a) Linear amplitude scale. b) Decibel scale

A comb filter can manipulate the signal in such a way that the impulse response

decays over a set time. The comb filter or the plain reverberator is one of the fundamental building blocks of the Schroeder's reverb [13] which will be discussed in the following section.

Schroeder's reverb

The Schroeder's reverb is based on the idea of artificial reverberation by Manfred Schroeder. It is a combination of comb and all-pass filters, the figure 2.26 shows the overview of it. Using comb filters in parallel with various gains and delays and then adding up the output of these filters will generate a reverberation effect [14]. To mitigate any high frequencies produced by the comb filters, a series of all-pass filters can be used in conjunction with the comb filters to produce a more natural sound image. The reason for the popularity of the Schroeder's reverb is that it is computationally effective and conceptually simple. The filter diagram for Schroeder's reverb can be seen below, in figure 2.26.

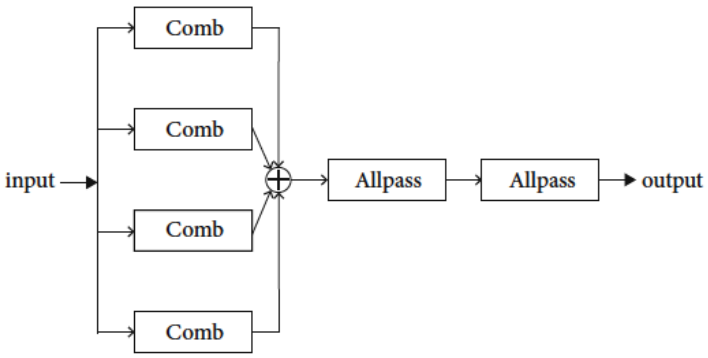


Figure 2.26: Filter diagram of Schroeder's classical reverb, comprised of four comb filters in parallel followed by two all-pass filters in series [14]

The four comb filters can be named H1(z), H2(z), H3(z), H4(z) which corresponds to each of the comb filters. The transfer functions of the four comb filters are as seen in the figure 2.27.

$$H_k(z) = \frac{b_k}{1 - a_k z^{-D_k}}, \text{for } k = 1, 2, \ldots, 4,$$

Figure 2.27: The transfer function of the comb filters used in the Schroeder's reverb [14]

The D in the transfer function is the delay, a - feedback coefficient, and b - mixing parameter [14]. The reason for using the mixing parameters is that when

a number of filters are connected in parallel, this parameter can be used to control how much of each comb filters output will be used in the total production of the output.

Similarly the H5(z) and H6(z) denotes the transfer functions of all-pass filters. Their transfer function can be seen in the Figure 2.28.

$$H_k(z) = \frac{-a_k + z^{-D_k}}{1 - a_k z^{-D_k}}, \text{for } k = 5, 6.$$

Figure 2.28: The transfer function for the all pass filters used in the Schroeder's reverb [14]

The z used in the equations corresponds to the pole and its radius [2]. To understand the poles, it is crucial to be familiar with the unit circle. Here is is called a Pole-Zero plot. Zeros and poles are fixed for each system and can vary between each [61]. The position of poles and zeros influence the frequency response [61]. Poles would increase the magnitude (dB) while the zeros will decrease the value of Magnitude (dB). The closer the poles and zeros are located to the unit circle aka away from the middle, the bigger the effect of the filter. See figure 2.29 for illustration.

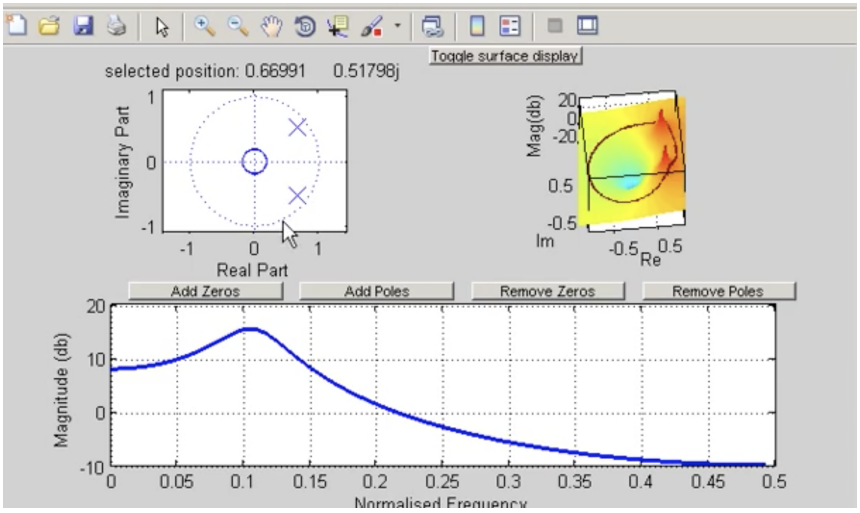


Figure 2.29: Frequency response depending on the pole and zeros position in unit circle [61]

A popular modification to the classical reverb is called the Moorer's reverb in which two more comb filters are added to the system in addition to having a low-pass filter implemented into each of the comb filters. This results in the echoes being more spread out, softer and more natural sounding like the plain

reverberator [13]. Below is the block diagram for the Moorer implementation of a low-pass filter in the comb filter loop.

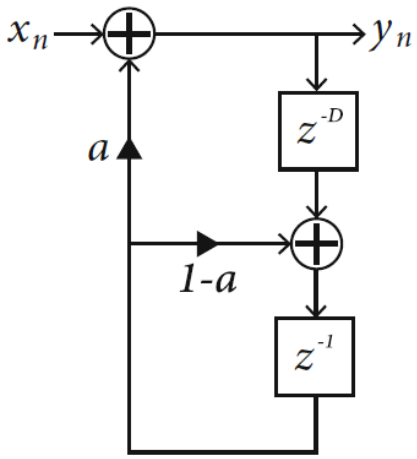


Figure 2.30: Block Diagram of the Moorer implementation

Equalisation

Equalisers are used in a vast variety of industries and have many use cases. An equaliser, in its simplest form is able to alter the amplitude of a signal at a given frequency. It operates with three variables

- the centre frequency, which can be considered the target frequency;
- the bandwidth which determines the spread of the effect from the centre frequency;
- the gain which controls the amplification or attenuation of the frequency in accordance to the aforementioned variables. [12].

There are many ways to implement equalisation for audio. There are also different types of equalisers such as the parametric equaliser, the graphical equaliser and the semi-parametric equalisers. The parametric equaliser will be explained below as that is being found to be most useful regarding the scope of the project.

Parametric equalisers are built from multiple equaliser filters connected in series. Equaliser filters are a combination of notch and peak filters. Where the peak filter amplifies and the notch attenuates depending on the variables input from the user [13].

$$H_{eq} = G_0 H_{notch}(z) + G H_{peak}(z) \tag{2.3}$$

Where G and G_0 are the gain values of the filters and H_{notch} and H_{peak} are the transfer functions of the notch and peak filters, respectively. Below, the notch and peak filters will be explained with their respective transfer functions. First, the two filter's common variables will be discussed. Both the notch and peak filter's take the parameters:

$$Bandwidth(\Delta\omega) \tag{2.4}$$

$$CenterFrequency(\omega_0) \tag{2.5}$$

$$Gain(G) \tag{2.6}$$

The bandwidth can be calculated using the following formula:

$$\Delta\omega = \frac{2\pi f}{f_s} \tag{2.7}$$

The center frequency can be found using the formula:

$$\omega_0 = \frac{2\pi f_0}{f_s} \tag{2.8}$$

Notch Filter The notch filter, as mentioned above, is responsible for attenuating frequencies. Its transfer function is as follows:

$$H_{notch}(z) = \frac{1 - 2cos\omega z^{-1} + z^{-2}}{1 - 2bcos\omega_0 z^{-1} + 2b(2b-1)z^{-2}} \tag{2.9}$$

Where b is the gain coefficient and is calculated using:

$$b = \frac{1}{1+\beta} \tag{2.10}$$

$$\beta = \frac{\sqrt{1 - G_B{}^2}}{G_B} tan\frac{\Delta\omega}{2} \tag{2.11}$$

Where G_B is the gain at the cut off frequencies, or the gain at the center frequency +/- half of the bandwidth.

Peak Filter

The peak filter is very similar to the notch filter. However, as the name suggests it is responsible for amplifying frequencies. Its transfer function is:

$$H_{peak}(z) = (1-b)\frac{1 - z^{-2}}{1 - 2bcos\omega_0 z^{-1} + (2b-1)z^{-2}} \tag{2.12}$$

where b is calculated from:

$$b = \frac{1}{1+\beta} \tag{2.13}$$

$$\beta = \frac{G_B}{\sqrt{1-{G_B}^2}} tan\frac{\Delta\omega}{2} \tag{2.14}$$

The Parametric Equaliser Filter

As mentioned above, the parametric equaliser filter is the linear combination of the notch and peak filters in series. Therefore, the transfer functions above need to be combined into one [13].

Combining the two transform functions gives us:

$$H_{eq}(z) = \frac{\frac{G_0+G\beta}{1+\beta} - 2\frac{G_0 cos\omega_0}{1+\beta}z^{-1} + \frac{G_0-G\beta}{1+\beta}z^{-2}}{1 - 2\frac{cos\omega_0}{1+\beta}z^{-1} + \frac{1-\beta}{1+\beta}z^{-2}} \tag{2.15}$$

Where the combined β is defined as:

$$\beta = \sqrt{\frac{{G_B}^2 - {G_0}^2}{G^2 - {G_B}^2}} tan\frac{\Delta\omega}{2} \tag{2.16}$$

Finally, the difference equation can be formulated which will return the output signal given an input signal and the parameters:

$$y_n = \frac{G_0+G\beta}{1+\beta}x_n - 2\frac{G_0 cos\omega_0}{1+\beta}x_{n-1} + \frac{G_0-G\beta}{1+\beta}x_{n-2} + 2\frac{cos\omega_0}{1+\beta}y_{n-1} - \frac{1-\beta}{1+\beta}y_{n-2} \tag{2.17}$$

And this is parametric equaliser filter.

2.6.2 Earcons

There are many ways in which a user can be guided via audio. One such way is through the concepts of earcons, which are short sound bites made to inform users of something. In this section, the definition of earcons will be explained and its use elaborated upon, as a means to influence or guide a user, of an interface. Sounds which are structured with similar principles as icons, are called earcons. These earcons are described by Blattner et al. as *nonverbal audio messages used in the user computer interface to provide information to the user about some computer object, operation, or interaction*. Earcons are brief and distinctive sounds representing a special event or signal information [9]. They are used for grabbing the attention of the user and can be used in many situations such as prompts or execution. As an example, they could be a notification sound for a text message and an interface giving an error or a reminder [9]. Studies have shown that high scores from arcade video games drops when the accompanying sound is turned off [9]. Earcons can

be grouped together as *compound earcons*. This is a hierarchy of earcons that sound familiar to one another. For example, error sounds should all be under one earcon category, and all the different kinds of errors should sound similar to one another [10]. This could be relevant as a way to signal to the user that they are moving the right or wrong direction by playing either an error earcon or correct earcon.

Guidelines for creation of earcons

There are various aspects to consider when designing and creating earcons. A few of the most prominent and important ones will be listed below [10].

When creating earcons, one should consider:

- Timbre;
- Register;
- Rhythm;
- Intensity [10].

Timbre can be recorded by musical instruments with multiple harmonics. Timbre makes it easier to differentiate between earcons, and reduces the chance of the earcon being masked amongst other sounds. Also, using more than one timbre per earcon improves the use of compound earcons. **Register** is related to timbre which should vary two or three octaves in its register. The **Rhythm** can vary, and should consist of different notes in each section of the rhythm. The **Intensity** or loudness of the earcon should not be used to differentiate between earcons and should not annoy the user [10]. Humans are capable of hearing sounds that are between 20 and 20000 hertz, but sounds are most noticeable at around 1000 Hz. Generally speech is heard as between 85 and 300 hz, and is between 25 and 65 db [60].

In conclusion, earcons is a widely used way of passing information to the user in a fast and simple manner, that if done correctly can help remind or tell a person about something and could be a navigational tool for people trying to solve a maze.

The Shepard's tone

This section explores an auditory illusion which produces two different tones which are still equal in their characteristic. This is achieved by a specific combination of octaves in a tone. The first octave in the tone is increasing or decreasing its volume, the middle part of the tone is in a steady volume, while the last part of the tone increases or decreases, in opposition with the first octave. An illustrations of that can be seen in the figure 2.31.

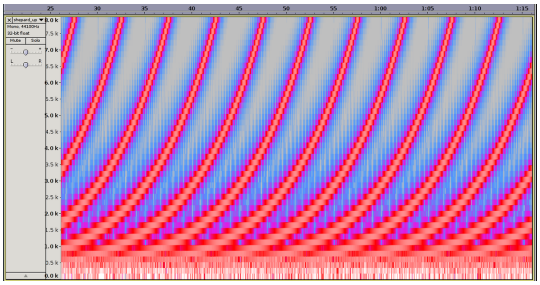


Figure 2.31: Spectogram of Shepard's tone [48]

When the first octave reaches a certain threshold it drops down to where it started, so does the other two octaves. This produces the illusion of an ever increasing volume if the tone in octaves increase, and ever decreasing volume if the tones in octaves decrease, yet the volume stays the same. This specific version of the tone is known as the *Shepard-Risset Glissando* [53].

2.7 Delimitation

This report started with an introduction leading up to a simple question relating to audio cues, performance and navigation as noted in the Initial Problem Statement - *"How can audio cues be used to improve the performance of a player navigating through a labyrinth or maze game?* (1).

Through the analysis, a plethora of various research areas was brought forth. First and foremost, it was found that the project should concern itself with mazes and not labyrinths. The reason for choosing to work with mazes, is that they have branching paths, while labyrinths do not (2.1). Furthermore, it was found that the performance of a player, solving a maze, could be measured by two meaningful variables. The length of the path it takes one to move from start to finish and the time it takes (2.2). Moreover, when navigating through a maze, it was found that the user can be nudged using various techniques, explained previously (2.3). Interaction with the game, could be done using a physical interface. Physical interfaces are an integral part of playing games, the design of game controllers was analysed in the physical interface design section. The main takeaway was to use buttons that allow the player to interact with all aspects of the game (2.4). Additionally, through a state of the art analysis, requirements for the audio design were found such as using binaural sound to guide players, using various audio cues and limiting visual details (2.5). The aforementioned audio cues, sound design and utilisation of audio processing was explored in the audio processing section. Most notably panning, reverb, attenuation and equalisation will be used for the overall audio design of the game (2.6). With the analysis and delimitation in mind, the final problem

statement can be formulated as:

Final Problem Statement

"How can audio processing effects be used to make audio cues that assist a player navigating a digital 3D maze, thereby decreasing the path length and time taken to complete it?"

2.8 Design requirements

In this section, design requirements based on the preceding analysis and delimitation will be formed in a list. This will help in later design processes.

- The player should be able to walk on paths included in the maze but should not be able to cross the walls or hedges used to separate paths, nor walk outside the parameters of the game (2.1);
- It must be possible to time the progression through a maze from the starting point to the finish point (2.2);
- It must be possible to measure the path length of individual users through a maze from the starting point to the finish point (2.2);
- The maze should include dead-ends, and be designed in a complex way (2.1);
- Panning should be used to provide the user with directional cues (2.6.1);
- Artificial reverberation should be used to convey information about the spatial characteristics of the maze (2.6.1);
- The maze should be designed to eliminate the use of the Right Hand Rule strategy of solving mazes (2.1);
- The camera angle should only enable a partial view of the full maze (2.1);
- Audio cues should assist the player by notifying whether they are moving in the right or wrong direction (2.3, 2.6.2);
- The game should be able to be controlled using a physical interface in the form of a game controller (2.4);
- The game controller should include a D-Pad and Action Buttons (2.4);

Chapter 3

Methods

This section will take a look at the methods, primarily, the testing and statistical methods that will be used to reach the goal of this project and answer the final problem statement (2.7). The section also includes methods used for design and implementation.

3.1 Agile

For this project an iterative and agile approach will be taken. This will be implemented using the Scrum system as the outline for the work flow. The agile work flow is an iterative method that enables fast improvement of the product by simultaneously evaluating and implementing the product based on previous evaluations. When compared to another style of work flow such as the waterfall method, the agile method allows for the design and development of a product to change direction from the initial idea, if evaluation favours a change in design, whereas waterfall does not. The agile method allows for the product developers to understand which changes made a specific impact and what needs to be changed for the next sprint. Sprints are the process of fulfilling goals set by the previous evaluation. These changes are then evaluated upon before a new sprint can commence. This is repeated and thereby creating a cycle [43].

3.2 Hypothesis

A hypothesis is a presumption of the relationship between two or more variables made based on limited evidence and serves as a starting point for further research. When performing said research, an experiment is made that should aim to either accept or reject the hypothesis based on statistical analysis. The experiment tests whether an independent variable influences a dependent variable. The former variable is in the experimenters control while the latter variable is the measurement

of a change caused by the independent variable. A null hypothesis is a statement of the absence of a relationship between these variables. An alternative hypothesis is a statement of a relationship between the variables [22].

3.3 Usability

Usability testing is a testing method, which can be used in both the early and later stages of the development process. Usability testing is used to figure out different things about the product in development, such as whether or not product is intuitive to use, if the product works as intended, but also if any mechanisms are missing. Furthermore, it tests whether a system is easy to learn, satisfying and efficient to use, easy to remember and whether it reduces errors in the user's input [44].

3.4 Descriptive Statistics

Descriptive statistics is as the name suggests, statistics that describes data in various terms. It can represent a sample or population in various ways, like the mean and median. For example, the mean, which is a measure of central tendency that can be calculated by adding up all data points and then dividing by the total number of data points. Another example is the median which is the middle number if all data points were sorted from the lowest value to their highest [22].

3.4.1 Normal Distribution

Normal distribution is a probability distribution which is symmetrical around the mean. When visualised it resembles a bell, hence the name bell curve. This simultaneously shows that data occurs more frequent around the mean [22].

3.4.2 Standard Error of the Mean

The standard error of the mean is an individual estimate of a data points difference to its mean [22].

3.4.3 Variance

Variance is a measure of how far the data points are estimated to differ from the sample mean. In other words, it is a measure of how far data points are spread out in relation to the mean [22].

3.4.4 Independent and Dependent Variables

When doing experimental tests, independent variables are controlled by the experimenters. The dependent variables are then recorded as a result of changing the independent variables [22].

3.5 Parametric Statistics

Parametric statistics can be applied to a set of data if the data is measured at an interval or ratio level. That however is not the only requirement, parametric statistics also rely on variances between conditions or groups being the same. Variance is in other words the accuracy of the mean, described above. The final requirement is that the data comes from a normal distributed sample, also explained above [22].

3.5.1 Shapiro-Wilk Test

The Shapiro-Wilk test compares the scores in a data set with that of a normally distributed data set with the same standard deviation and mean. If this test is insignificant (i.e p-value < 5%) one can assume that the data is normally distributed, however if the test score is significant (i.e p-value > 5%) the data can be assumed abnormally distributed [22].

3.5.2 Levene's test

Levene's test aims to compare variance between two sets of data by testing the hypothesis that they have equal variance. If the outcome is significant (i.e p-value < 5%) one can conclude that the variances are unequal. However if the outcome is insignificant (i.e p-value > 5%) one can conclude that the variances are equal and further statistical analysis can be conducted whilst assuming equal variances. The 5% significance level might change according to the sample size and data type [22].

3.5.3 T-Test

Once a Levene's test has assumed that the data sets have equal variance, a T-test can be performed. The T-value is calculated by dividing the mean difference and the standard error, also explained above. This value is then compared to that of which one might get by chance using various degrees of freedom. Again this value will be deemed significant or insignificant. If the value is lower than the chosen significance level, 5% for example, one can conclude that there is a significant difference in the sample means and further conclude any results from this with a 95% confidence interval [22].

3.5.4 Non-Parametric tests

If the data shows to be non-parametric, the Welch T-test will be used to compare sample means. This T-test does not assume equal variances, however, it still assumes normal distribution [45].

3.6 A/B Testing

A/B testing is a testing format that splits the people testing into two groups, also known as a between-group test. One group tries one version of the program or product while the other group tries a different version. The idea is that there should be a significant difference in the results between the group in relation to the hypothesis [22].

3.7 Likert scale

When testing the hypothesis and executing the test, some tools are needed in order to receive feedback which can be used to answer the hypothesis or support a discussion. Such a tool is likert scales. Likert scales, in their classic form provides a statement in which the participant can express their level of agreement, based on several likert items. An example of two likert items could be, "strongly agree" or "strongly disagree" [22].

3.8 Data Collection Methods

When collecting data there are typically two methods of doing so, qualitative and quantitative. Quantitative data collection is based around receiving feedback that can be analysed with numbers and statistical methods, while qualitative data goes more in depth and allows the person responding to go into more detail about their answer. For the final test two sources of data will be collected, one directly from the game and another from the questionnaire. The questionnaire will include both qualitative and quantitative questions and the game will record the participants time spent in the game and their path length playing the game. In the questionnaire the qualitative questions will be in relation to possible improvements for the next iterations and overall impressions and comments. Here they are allowed to respond in detail and in their own words. The quantitative questions will be in relation to rating the game experience and quality of the game and sound. This kind of data is called ordinal data. In the case of this project the participants will be asked to rate the questions between 1 and 7 where 1 is completely disagree and 7 is completely agree. The other data that is collected is the game data, this data

is classified as ratio data, this type of data includes a true zero, which makes them favorable to use for statistical analysis [22] [44].

After exploring the methods above, an overview of design principles and methods relevant to this project will be presented below.

3.9 Don Norman's design Principles

In the book by Don Norman, *The Design of Everyday Things[41]*, Don talks about six concepts that a designer should use when designing a product. These are affordance, consistency, mapping, visibility, constraints and feedback and each of them have an effect on the user of the product. Each concept will be further explained below.

- **Affordance** is about how the user is using the object. When the user is being presented with a physical object or a digital element, it should be intuitive how it should be used, from just looking at it. This concept is crucial to get right because if it is too confusing or difficult for the user to handle, they might not use the product. According to Don, the user tends to use simple and easy to handle products [41].
- **Consistency** is for example when a user is being presented to a design they have seen before, they will expect a similar design. This can be seen for instance in the Playstation and Xbox controller 2.4.1 where the layout are similar enough for a new player to use them without any trouble learning them[41].
- **Mapping** is important when creating an interface since the user expects the functionality to correspond to the layout of the design. For example, pressing the up arrow on a directional pad, the user would expect forward movement [41].
- **Visibility** is when an element is more visible to the user than other elements, this can lead to a more frequent use of the given element's functionality. An example of this could be a big button on a website that says 'Home' [41].
- **Constraints** is to limit the user of what they can do. This is done to limit the possibilities of what the user can do in a more controlled environment. An example of this could be a game in which the mechanics limit what a user can do[41].
- **Feedback** is something that happens after the user has taken an action and an interface somehow confirms the action to the user. For example, when using game controllers 2.4.1, the user is given feedback in the form of vibrations when an action is happening in the game[41].

3.10 Gestalt Principles

When making an intuitive design, it should be clear to the user what belongs together and what does what. A way to communicate this information, is through knowledge of how the brain is interpreting information about an interface. These innate human characteristics can be taken advantage of by using seven principles named the Gestalt Principles [34]. These principles are commonly used when designing various types of user interfaces. There are seven principles to consider and they are as follows:

- Similarity;
- Continuity;
- Closure;
- Proximity;
- Figure-Ground;
- Focal point;
- Common region.

The relevant design principles for the scope of the project will be described further below.

Similarity is when objects share similar characteristics such as shape, size or color. When they share similarities they are perceived as one group [34].



Figure 3.1: Example of Similarity [19]

Proximity is when elements are close together, the brain will perceive those elements to be a group separate from elements that are further apart [34].

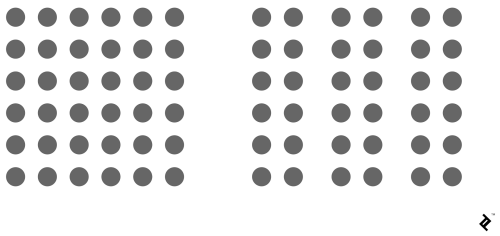


Figure 3.2: Example of Proximity [19]

Figure-Ground is describing how the eye has a tendency to separate objects from the foreground to the background of an image [34].

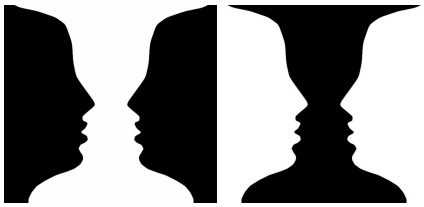


Figure 3.3: Example of Figure/Ground[19]

Common region - is closely related to proximity. Objects located within a small region are perceived as being grouped together. A visual example is illustrated in the figure 3.4 [34].

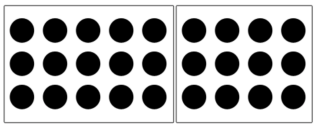


Figure 3.4: Example of Common region [1]

Chapter 4

Production

This chapter will cover the process of the production, divided in 3 iterations. Each of which consists of design, implementation and evaluation phases. During each iteration, considerations will be noted and conclusions made accordingly to improve the upcoming iterations. The final iteration will include statistical analysis, and will be a baseline for forming a conclusion of the project.

4.1 First iteration

This section will cover initial design considerations. After that implementation will be described, followed by the initial user test. Feedback collected will be used to improve the product.

4.1.1 Design

This section will cover design ideas and considerations based on the analysis, see (2) and design requirements, see (2.8). It will include maze design, initial design of audio and design ideas for a game controller.

Design of a Maze

After analysing the labyrinths and mazes, it was decided to keep the main focus on the mazes, as that is where audio guidance would be more useful to a player. Presented below are the decisions of what the maze will include and how it will be designed in the first iteration.

Based on the design requirements (2.8) it is decided that users will be able to walk freely on the paths included in the maze, but will not be able to cross hedges nor walk outside the boundaries of the maze. To ensure that, there will be a visual difference between how the background and foreground is going to be displayed

and perceived by the user, see the Gestalt Principle "Figure/Ground" in section (3.10). One must be aware of how strong an impact the visual information has when solving a maze, therefore, based on the *"facilitate nudges"* from section (2.3), the visual information provided to the users will be limited.

The navigation through the maze was initially planned to be done with a physical controller designed for the game, which will later be discussed in the section (4.1.1). For later testing purposes it is planned to implement a feature to time the game play and calculate the path length from the user's input (2.8). However, these details would not be shown to the users. These two variables would be used to compare the data between the two test groups and not to encourage users to complete the maze faster or with a better performance based on the data visible.

The design of the maze (how the paths are branching and the walls are used to divide the paths) will be based on the properties which were earlier looked upon in the section (2.1). In order to make the audio cues work, the maze is planned to be challenging enough for the user to rely on the audio and not the visuals. The solving strategies mentioned in the maze section (2.1) will be eliminated as much as possible, as stated in the design requirements (2.8). Therefore, the maze will be designed in a complex way, with dead-ends, branching paths, multiple major choices, which lead to longer paths. The view of the maze will not be fully visible to the user. This is another way in which the maze is made more difficult to complete using visuals (2.8). To eliminate the strategy of "Right Hand Rule", the outer walls will not always be connected with the inner walls, which means that the player will walk around in loops if they follow the right hand rule and not the audio cues, these cues will be discussed shortly in audio design.

While it is not yet clear how many dead-ends should be included in the maze, it is agreed that there will be enough dead-ends so that several audio cues can be implemented at the junctions. This will increase the amount of variations in which the player can walk throughout the maze and how audio cues can be designed.

The overall complexity of the maze can be determined by how many decisions in total the player must make in order to complete the maze as analysed in the section (2.1). As the project is not focusing on designing a maze purely from scratch, and the maze is just a way to test the use of audio cues, the tool used for creating an initial maze was a maze generator from the web page *http://www.mazegenerator.net*. However, the maze's difficulty might be increased in later iterations if needed. Once the maze had been generated, the group made multiple changes to make it fit the requirements. These changes included dead ends, right hand rule prevention and loops for the outer walls of the maze.

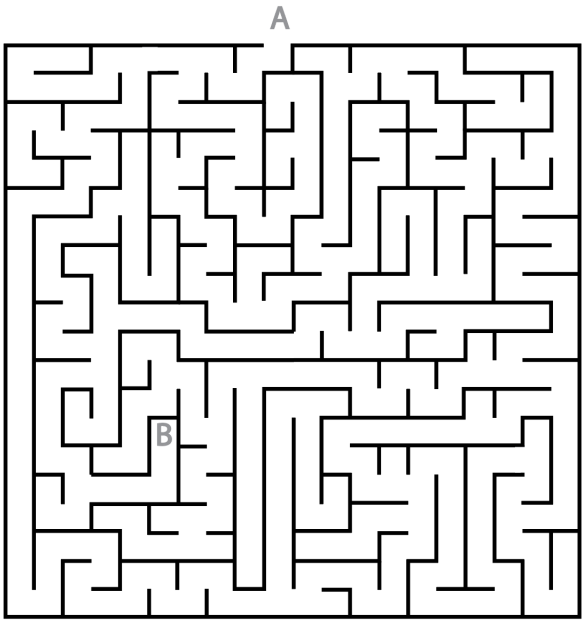


Figure 4.1: A top view of a full maze

The next figure 4.2 shows how the maze can be solved (green line), how it will be possible to use the right hand rule unsuccessfully (orange line). There are 3 locations in which the user is forced to make a major choice about which path to choose next. The reason for choosing to place the end point in the middle of the maze was to eliminate the use of the right hand rule. It was further decided to use a hidden object as the finish line to make the user focused on finding that object.

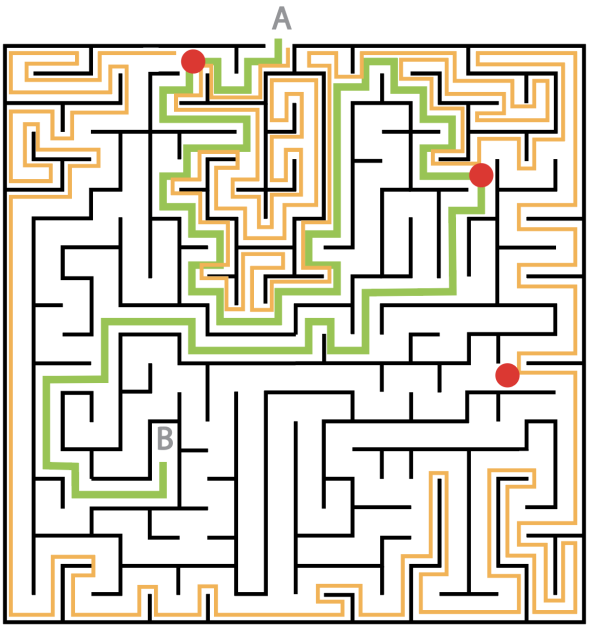


Figure 4.2: Solution in green, possible path using right hand rule in orange, major choices to be made while solving the maze in red

In order to eliminate other maze solving strategies covered earlier in the labyrinths and mazes section (2.1), only a part of the maze is going to be visible to the user as set in the design requirements (2.8). The level of complexity is going to increase if one would be asked to solve the maze without being able to see the overview of it. This will give a placebo effect, reviewed in *"deceive nudges"* (2.3), of being in a relatively big maze. This forces the user to pay more attention to the audio rather than the visuals. For instance, if the user is standing on a 1 by 1 square, then the view would be covering the space around the user equivalent to approximately 3 by 3 squares as seen in the figure 4.3.

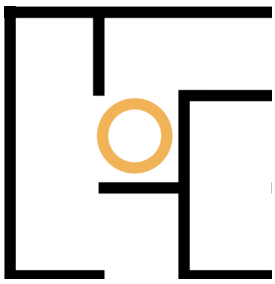


Figure 4.3: The view visible for the user who tries to complete the maze, orange circle to be perceive as the location of the user

A more precise description on how it is initially planned to use audio for guiding the user will be looked upon in the next section.

Audio Design

The design of the audio and the techniques used to guide a user through the maze will be explored next. These decisions are based on techniques covered in the audio processing section of the analysis (2.6) and the design requirements presented earlier (2.8).

It should be noted, that for the audio design section, the term *audio cues* can and will refer to various audio related concepts that aid the player in any way. Concepts of audio that can be used to create audio cues include but are not limited to: *Stereo Panning, Attenuation, Amplification, Reverb, Earcons and Shepard's Tones*. For example, stereo panning can aid the player by revealing the direction of a source.

The group got inspired by the SOTA (2.5) and decided that the binaural sounds and stereo panning techniques will be used which would require future test participants to use a pair of headphones (2.6.1). The audio of the object hidden inside the maze will interactively change depending on where the player is positioned in the maze and which direction they are moving towards. Audio will be changed according to what was written in the attenuation and amplification section of the audio processing section (2.6.1). This will provide an overall spatial feeling of the maze, without the need to fully visualise it. It will be an addition to the audio cues which will work as smaller hints to guide the user once they make a decision by turning down a path. In the game, it will not be written as a rule to follow the audio cues and pay attention to the sounds, users should be investigating themselves to gain more out of it based on the *reinforce nudges - "Instigating empathy"*, reviewed in the nudging section of the analysis (2.3).

Sounds used in the game will not be above the limits of frequencies which are comfortable to be heard by human hearing (2.6.2). In this iteration a change of pitch will be implemented at every major junction of the maze. That is to say, when the user is proceeding through the maze correctly, the audio will play at its native pitch. When the user makes an incorrect choice, the audio will shift in pitch until it sounds incorrect. This aspect of the audio will be influenced in an ambient manner to not disturb the player, based on the *Reinforce nudge - "Ambient feedback"*, see the nudging section of the analysis (2.3).

In future iterations the audio is planned to be implemented within the program Pure Data. This should then communicate with the Unity game engine, where the game will be built. However, for this first iteration advanced audio manipulation

is not required and will thus be implemented in Unity.

Audio Cues

Audio cues described above will be used to test if and how users interpret the sounds and if the audio cues serve their purpose. Based on the section about nudging (2.3), multiple nudges were used when designing the audio. *Reminding of the consequences*, a part of the category *confronting nudges*, see (2.3), can be used when the user makes a decision to turn in the game, by playing a sound, thereby reminding the player of the decision they made.

Biasing the memory of past experiences from the category *deceive nudges*, see (2.3) will be used as they reflect on ones previous experience. Meaning, the users should be nudged to rely on what they learned the previous time they heard an audio cue.

It should be noted that based on the analysis, see (2.6.2), Shepard's tones were explored as audio cues. One of which will be played forwards, the other one backwards, these tones can create a positive uplifting and a negative lowering effect. It should make the player think that they are being guided in the positive or negative direction depending on which version is being played. However, these could not be made to sound distinguishable enough.

Reverb

Reverb will be used to create an immersive and realistic element in the sound of the game. One of the sounds that will be present in the game, will be the sound of the player's footsteps. This sound will be reverberated in order to create a sense of the size of the maze. The sound for the footsteps will be found from available sound effects online, then, depending on the variables, filtered and changed according to research into reverb in the audio processing section of the analysis (2.6.1).

To sum up, audio cues will be present at every major junction in the maze. At these junctions the player will receive positive or negative feedback in the form of an audio cue based on, if the path they have chosen is leading towards the end of the maze or not. Simultaneously an audio pulse will serve as a panning audio cue to give the direction of the end of the maze.

Design of an In-Game User Interface

Due to the nature of this project, a user interface was required for the user to control the game. This was not of substantial importance to the project or tests but

was required. Therefore, a simple interface was designed accordingly. Initially, a sketch was made to visualise the group's ideas, see figure 4.4.

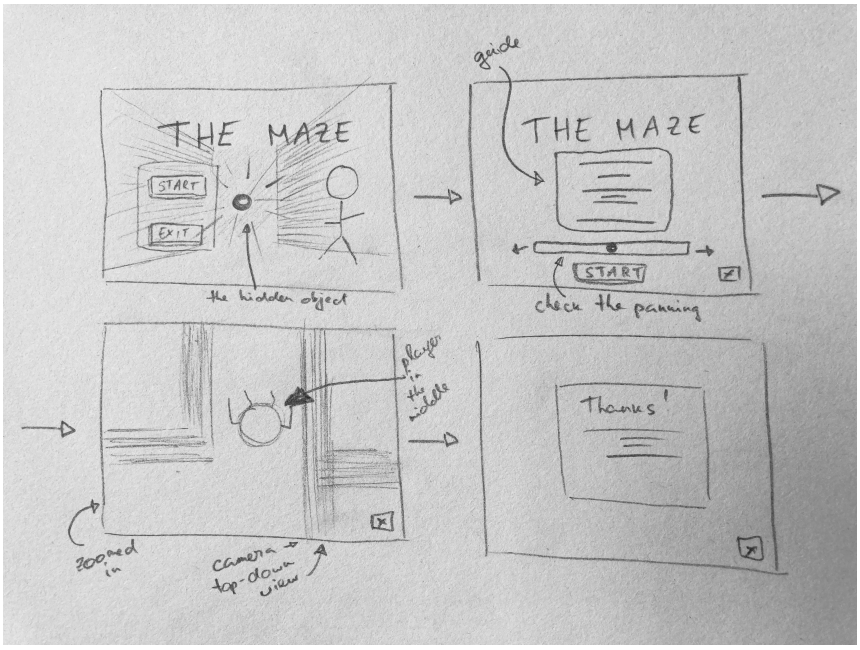


Figure 4.4: Sketch of the visualisation of the group's ideas for the user interface

For this first iteration, it was decided that only the essential functions should be implemented in the form of buttons. As stereo headphones are required for the in-game audio panning, the first screen that is shown to the user should be designed to test if the headphones are stereo. A simple slider should be able to control the panning, allowing the user to test their headphones. After the headphones have been checked by the user, they can navigate to the main menu by pressing a confirm button as seen in figure 4.5.



Figure 4.5: Panning test

The only essential functions on the main menu for this first iteration will be a “play” button and an “exit” button. These would use grouping and figure-ground from the Gestalt principles (3.10) to clearly convey they are the two interactable elements on the screen. They will also be coloured according to their function. For example, the “play” button will be green to signify moving forward, and the "exit" button will be red. The buttons will be placed on a dark blurred background based on the Gestalt design principle common region (3.10). An illustration can be seen in the figure 4.6.



Figure 4.6: The main menu in iteration one

When the user clicks "play" the game will begin. A pop-up menu will appear to briefly explain the controls and the aim of the game in general as seen in figure 4.7.



Figure 4.7: Help pop-up menu

When the user has completed the game successfully a second pop up menu

will appear to thank the player for testing and house an "exit" button to quit the application as seen in figure 4.8.



Figure 4.8: In game interface of iteration one

Controller Design

For this project, a custom controller is going to be made. The design of the controller is going to be based on the research made in the analysis (2.4.1). Initially, a group brainstorm was made. This brainstorms purpose, seen in figure 4.9, was to form a design of a controller which followed the gestalt principles (3.10) and Don Norman's principles (3.9) and have the necessary functionality. The input that controls the character was decided to be in the style of a traditional D-pad. Many state of the art controllers (2.4.1) are using a D-pad and the maze only requires four buttons for the direction (forward, backward, left and right) to move the player and two action buttons for confirming and declining menu actions.

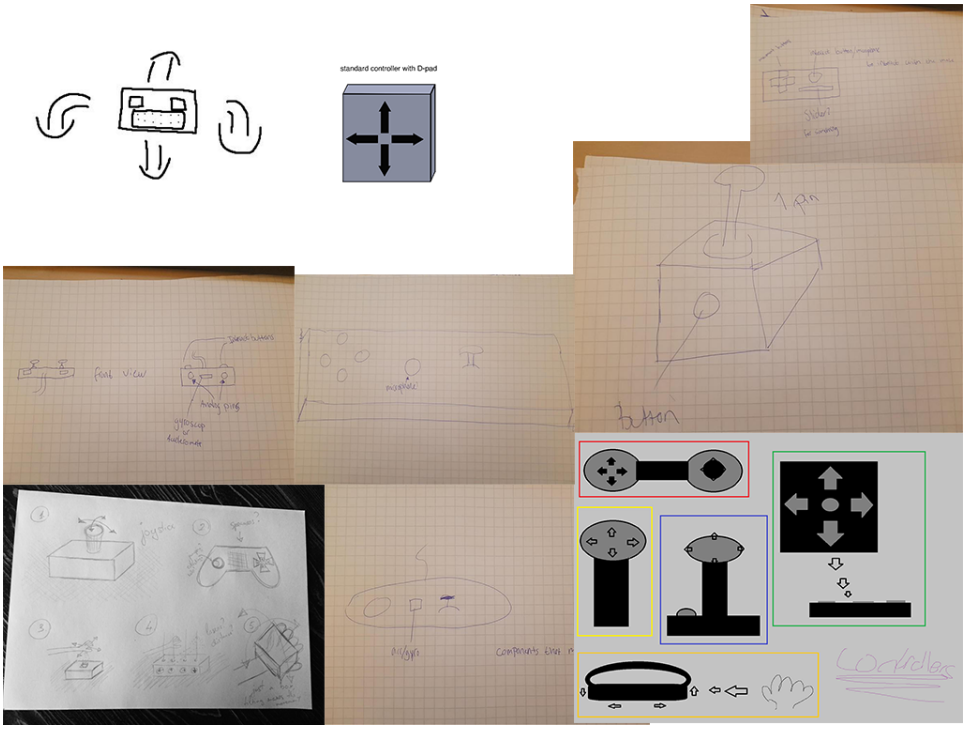


Figure 4.9: Collage of controller brainstorm

To make sure the user understands that they are different buttons, the gestalt principle of similarity can be utilised for the buttons, see section (3.10). The buttons for player movement will be in shape of arrows and be of equal size, color and shape based on the design principles *Consistency* and *Mapping* by Don Norman (3.9). The design and position of the buttons is using well known and intuitive principles for the user to understand that these buttons correspond to movement in certain directions inside the game. The two action buttons will be a green tick and a red cross as seen in the figure 4.10.

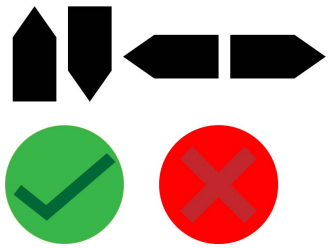


Figure 4.10: Design of buttons

The placement is important since it can change the perception of what the button's functions are. From what was found in the analysis (2.4.1), the directional

buttons are using Gestalt principle of proximity (3.10). The user will get an idea that the buttons have a similar purpose to each other as they are grouped, close to each other. Action buttons will also be grouped together, while leaving a space between them and the directional buttons. The final design of the controller itself could be in the shape of a box where the different components can be hidden inside, see figure 4.11 for the design of the controller.

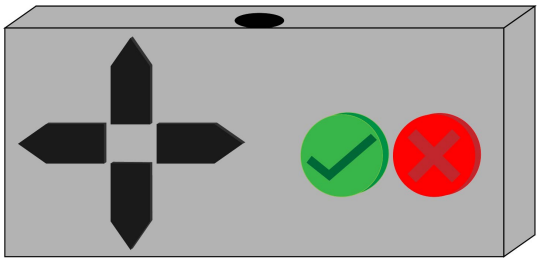


Figure 4.11: Design of Controller

The buttons can be placed on the front facing surface of the controller while a wire would be plugged at the top of the game controller as seen in the 3D rendering in figure 4.12. To use the principle of *Constraints* by Don Norman.



Figure 4.12: Render of Controller

The overall design uses the Don Norman's design principle *Affordance* described in the section (3.9). The design is kept simple with no unnecessary functions.

The following section will take a look at the components that will be used, as well as their placement in a circuit diagram.

Circuit Design

All of the buttons are going to be connected to a micro-controller in order for them to fulfill their interactivity purposes. In this case an Arduino Uno is used (2.4.3). The Uno is a micro-controller board that can connect to a computer to send and receive information. The buttons are going to be connected in parallel to each other and to the digital input pins and will need resistors to minimise the possibility of sending wrong information which would result in the player moving in an undesired direction.

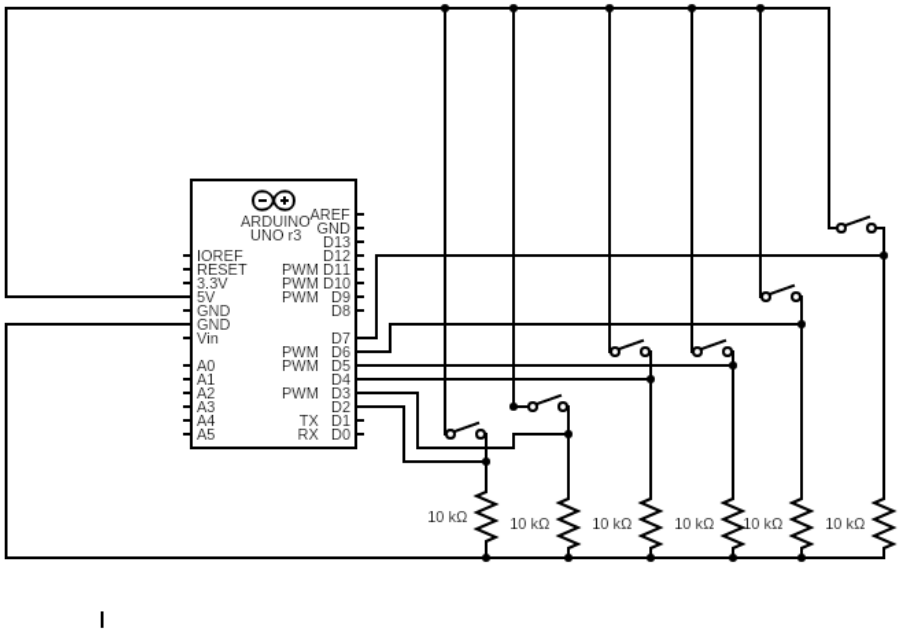


Figure 4.13: Circuit diagram for an Arduino Uno with buttons connected to it.

In figure 4.13, a circuit simulation of the Arduino Uno with the buttons and resistors can be seen. First the 5V and ground are connected to the breadboard. Then a 10kΩ resistor is connected between the ground and a button. Buttons are then further connected with the 5V power supply, if pressed down this connects the 5V to the digital pin through the button. The completion of the circuit sends information to the computer that the button has been activated. The same technique is repeated until all six buttons and resistors are connected. The simulation of the controller can be seen in figure 4.14.

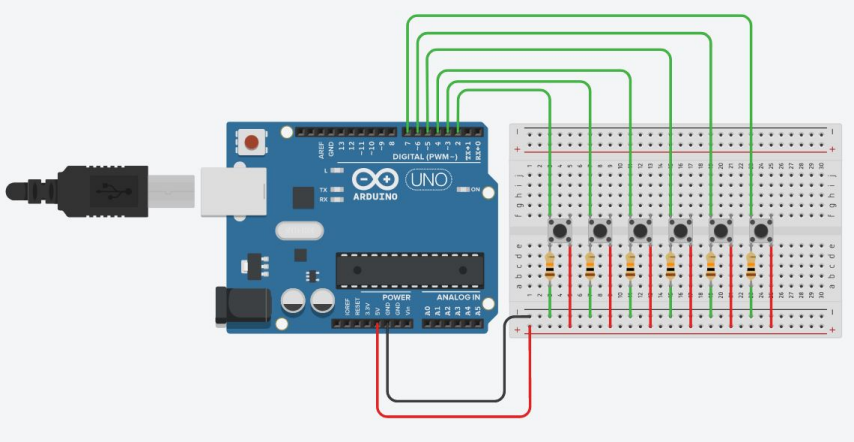


Figure 4.14: Circuit simulation of the controller.

4.1.2 Implementation

This section will describe the implementation of the first iteration. General programming for the game itself, programming of the audio design and code for the micro-controller will be presented. An overview of how the communication between the controller, computer, and game can be seen in figure 4.15.

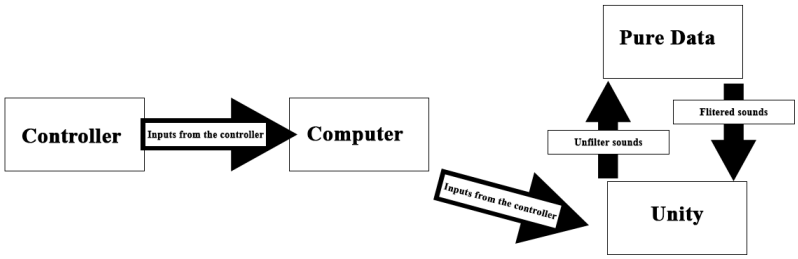


Figure 4.15: Overview of implementation of first iteration

Implementation of a game in Unity

To simplify the implementation process, many of the assets in the game were outsourced from various places. The exception being the maze itself and the logic scripts that govern the audio in the game. The character was found on the unity asset store and the controller that moves the character was included. Slight alterations were made to restrict certain functions, such as the sprint and jump functions.

The maze was modelled as one object using Blender, this made it simple to apply a mesh collider in Unity, stopping the player from moving through the walls

of the maze. In order to track the player's time and distance in the game, a simple script was attached, see figure 4.16.

```
private void Update()
{

    if (distanceCounter > 0 && ended != true)
        timeCounter += Time.deltaTime;

    if (controller.m_currentV > .01 || controller.m_currentH > 0.01 || controller.m_currentV < -.01 || controller.m_currentH <- 0.01)
        footstepsSource.volume = Mathf.Lerp(footstepsSource.volume, 0.5f, 10* Time.deltaTime);

    else footstepsSource.volume = Mathf.Lerp(footstepsSource.volume, 0,10* Time.deltaTime);
}

0 references
void FixedUpdate()
{
        distanceCounter += Mathf.Abs(controller.m_currentH + controller.m_currentV)*Time.deltaTime;
}

0 references
void OnTriggerEnter(Collider other)
{
    if (other.tag == "End")
    {
        ended = true;
        manager.GetComponent<Help>().End();
    }
}
```

Figure 4.16: Script attached to player

The "distanceCounter" variable, is tracking the distance moved by the player. It does this in the "FixedUpdate" method which is called at a fixed frame rate. The "Mathf.Abs" function is getting the absolute value of the current horizontal movement of the player added to the current vertical movement of the player. This is then multiplied by the the delta of the time since the last frame. This was found to get reliable recordings of the distance that can be compared.

Now that the game mechanics have been implemented the group will implement audio cues for the first iteration of the game.

4.1.3 Panning with Unity

The game engine Unity comes with its own sound system. This system can be used for the first iteration since it can play the background audio and the sound from the end of the maze simultaneously. The background music is continuously playing for the player, so the sound source component as seen in figure 4.17 will be added to the "Camera" object. This sound source component will have the background sound in it and the loop function will be activated, so it will keep playing as long the player is playing the game.

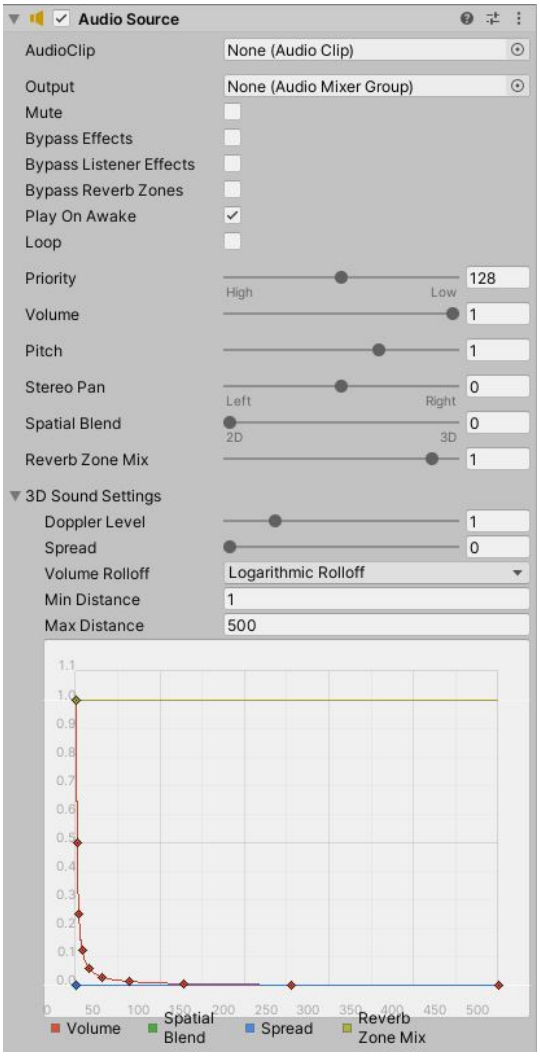


Figure 4.17: unedited sound source in unity

The sound for the end of the maze will also have a sound source component attached to it. It is going the have the same loop effect as the background music, but to get the panning effect, the sound source component will need some changes. By changing the spatial blend in the sound source component to 3D, it will affect the given sound by applying a 3D spatialisation calculation to it. This will result in the player being able to hear in which direction the end of the maze is located. With the spatial blend set to 3D, the 3D sound settings will become available. These settings can change how the volume is perceived based on how far away the object is from the player. The volume change will be affected by the option called "volume rolloff" where the volume is affected by a logarithmic, linear or a custom rolloff. Linear rolloff was found to be the most effective. Since the group wanted to test

the concept of the final problem statement as early as possible, the sound system from Unity will suffice for this iteration. Unity has many features built in to control audio in the game. Such as, attenuation, spatialization, panning and more. All of these are useful tools to prototype the first iteration of the maze. The pitch of the background sound was changed in the major junctions of the maze.

Implementation of the audio filters in Pure Data

For implementing audio cues, it was agreed to work with Pure data as this tool was used in the Audio Processing course. Reverb and a Shepard tone will be implemented with Pure Data and later used to connect with Unity. An explanation of how this was done will be presented next.

Shepard's tone for positive and negative feedback A simple implementation of Shepard's tone was done in Pure Data. In the Figure 4.18, the overview of the implementation can be seen.

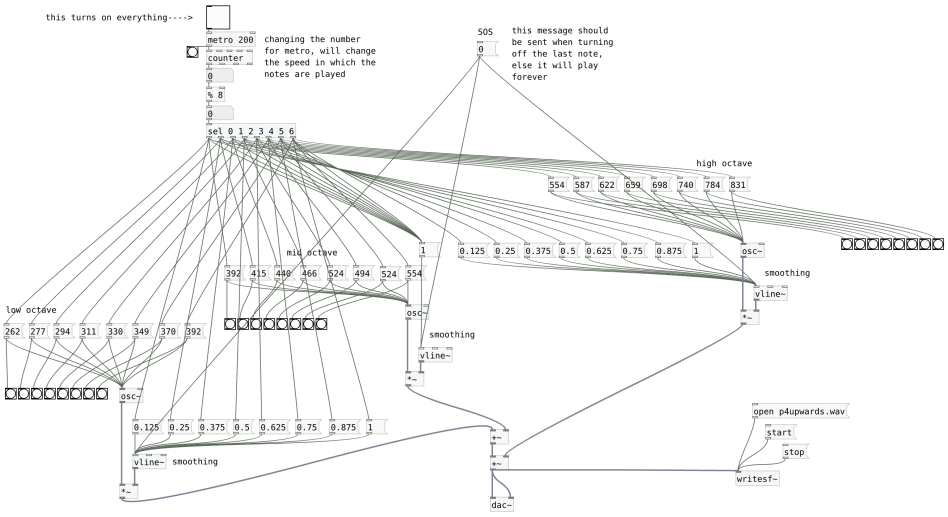


Figure 4.18: Shepard's tone implemented in Pure Data

It begins with a metronome object set to 200 milliseconds. A counter takes in the input and starts the counting. By modulus 8 it resets the counter at the end of an octave.

Frequencies of specific tones are used as input values for open sound controller (osc) objects. A multiplication of the signals is done to achieve the change in the volume. "vline" object is used for smoothing the sound being played at the digital analog converter (dac) output. The "bang" object governs the flow of the system. "Writesf" is used to save the file generated by Pure Data.

Reverb The reverb patch uses four comb filters and two all pass filters. The pure data patches for these, can be seen below. The implementation is based on the Schroeder's Reverb as earlier described in the analysis (2.6.1).

Each of the comb filters are implemented as seen in the figure 4.19. Both parameters - "delay D" and "coeff a" are used to set the desired reverberation effect.

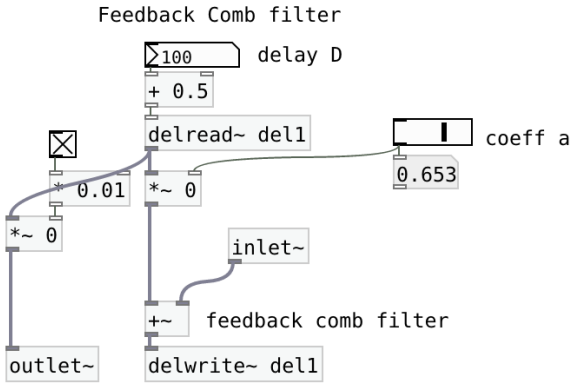


Figure 4.19: One of the four IIR feedback comb filters implemented in PD

The delay is self explanatory - the higher the value, the bigger the delay. *a* is the filter coefficient and is set to be a positive value based on section (2.6.1).

The implementation of all-pass filters can be seen in the Figure 4.20. The equation used in the all-pass filter under the *fexpr* , takes a sample and one before it, adds it together and uses the output value as an input as well which makes it an infinite impulse response filter.

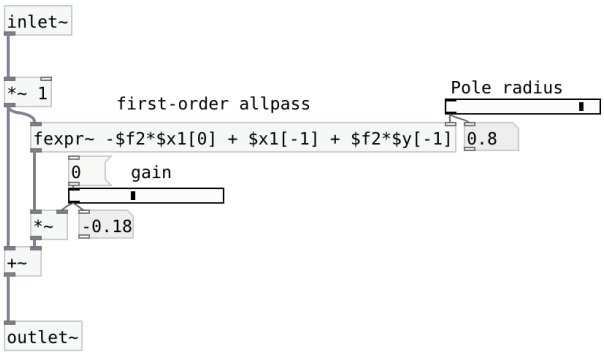


Figure 4.20: One of the two all pass filters implemented in PD

If the pole radius were to be set to the value "1", the signal would become unstable, therefore limitations are used. Gain is used to alter the strength of the filter, as it is multiplied with it. When the above mentioned filter is used, the reverberation is added to a sound file. However, this implementation did not prove to be working with Unity. Therefore, another solution was tried to be found as described next.

Moorer's Reverb The footstep audio file was passed through a Moorer reverb implemented in Python (2.6.1) to improve the spatial characteristics of the footstep audio. Below is a snapshot of the python code 4.21.

```
def moorersReverb(inputSignal, mixingParams, plainDelays, plainGains, allpassDelays, apParams):
    nData = np.size(inputSignal)
    tmpSignal = np.zeros(nData)
    # run the plain reverberators in parallel
    nPlainReverberators = np.size(plainDelays)
    for ii in np.arange(nPlainReverberators):
        tmpSignal = tmpSignal + mixingParams[ii]*plainReverberator(inputSignal, plainDelays[ii], plainGains[ii])
    # run the allpass reverberators in series
    nAllpassReverberators = np.size(allpassDelays)
    for ii in np.arange(nAllpassReverberators):
        tmpSignal = allpassReverberator(tmpSignal, allpassDelays[ii], apParams[ii])
    return tmpSignal
```

Figure 4.21: Moorers reverb implementation in python.

The sound with the reverb however, sounded more like a person walking around indoors, and since the maze is in an outdoor environment, the reverb was deemed unusable for the requirements of the game.

Implementing the code for Arduino

For this project, a custom controller was designed 4.1.1. For its communication with the computer, a micro-controller was used. An Arduino can handle that aspect by sending the input information from components connected in a circuit through a USB port back to the computer. The code used on an Arduino is written in C++ and has an integrated development environment (IDE), which can help compile the code and upload it to the Arduino. In the code there will be six commands sent through the USB port depending on which of the six buttons have been pressed. From the design (4.1.1), it is required to send four commands for movement and two commands to accept and deny actions.

In the arduino IDE, three arrays and two variables are declared. The two integer variables are called "btnStateLength" and "delayTime". The "btnStateLength" is used to tell one of the arrays it's length and "delayTime" is used to store how long each input should wait before sending a new input. The integer array that uses "btnStateLength" is called "btnState" which stores the state of each button which can be either 0 or 1. To read the input of the buttons, an array is constructed to

tell which pin is in use on the Arduino and it is called "digitalPin". Lastly, there is a string array called "commands", which stores the string commands that will be sent through the USB port to the application.

From there, there are two pre-made functions that are called "setup" and "loop". These were created when the script was made and they have different use cases. The "setup" function is called once every time the Arduino is powered up. Where on the other hand the loop function is continuously called as long as it is powered. In the "setup" function, there are two functions called "pinMode" and "Serial.begin". The "pinMode" function reads the input of the digital pin on the Arduino where the "Serial.begin" tells the arduino what data rate the data should be sent in and initialises the serial port 4.22.

```
const int btnStateLength = 6;
int delayTime = 20;

int digitalPin[] ={2,3,4,5,6,7};
int btnState[btnStateLength];

String commands[] = {"W","S","A","D","O","x"};


void setup() {
  // put your setup code here, to run once:
pinMode(digitalPin,INPUT);
Serial.begin(9600);
}
```

Figure 4.22: Initial variables and "Setup" function in Arduino IDE

In the "loop" function, there are two for-loops. The first for-loop sets each "btnState" to read the port that it is associated within the "digitalPin" array. This returns the state of the pin, which can be either connected or not and it does that by printing "HIGH" for connected or "LOW" for not connected. The second for-loop has a check to see if any "btnState"'s input is "HIGH", then it will print a command from the "command" array which will be sent to the serial port. Afterwards, the serial port is flushed, which means it will wait until the outgoing data has been sent. At the end, it will wait 20 milliseconds with the delay function, before taking a new input from the Arduino 4.23.

```
void loop() {
  for(int i =0; i<btnStateLength;i++)
  {
    btnState[i] = digitalRead(digitalPin[i]);
  }

  for(int i =0; i<btnStateLength;i++)
  {
    if(btnState[i] == HIGH)
    {
      Serial.println(commands[i]);
      Serial.flush();
      delay(delayTime);
    }
  }
}
```

Figure 4.23: The loop function in Arduino IDE

From here, the commands can be read from the serial port by the game program where a certain action will happen depending on the context of the state in the game. The next section will cover the game receiving the commands.

Unity receiving input from the Arduino To receive the inputs from the Arduino, Unity needs a script that can understand what is being sent from the Arduino. To read the input, the script needs a library which can read the ports that are available for the computer and it includes the "system.IO.Ports" library. This allows the script to see the data being sent, in any of the USB ports.

In the beginning of the "arduinoController" class, a float, a vector and a char variable is made with a RigidBody and serialPort object as well. The float controls the movement speed of the character and the vector named "movement" tells the game what direction the character should move in. The char variable is called "direction" and it will hold the input that is sent from the Arduino. The first object is a "rigidbody" component that is used to move the character in-game. The other object is called "sp" which is the serial port that is being used. The "sp" object is told which USB port it should communicate with and what data rate it should match with. In the start function, the "rigidbody" object called "rb" is assigned to the "rigidbody" component. In the start function, "sp" calls the function "Open" from the port library, which opens a connection with the given port and data rate. Afterwards, "ReadTimeout" from "sp" is set to "1" and that defines a limit to the amount of time taken to read an input, an exception is thrown if the time exceeds this 4.24.

```
public class arduinoController : MonoBehaviour
{

    public float movementSpeed;
    public Vector3 movement;
    Rigidbody rb;
    public char direction;
    SerialPort sp = new SerialPort("COM3",9600);

    // Start is called before the first frame update
    void Start()
    {
        rb = this.gameObject.GetComponent<Rigidbody>();
        sp.Open();
        sp.ReadTimeout = 1;
    }
```

Figure 4.24: The start function in "arduinoController" script

After the start function, there is an update function that is called every frame which contains several if-statements. The first statement checks to see if the port is open, from there it will try to get the input as a char variable from the port, otherwise it will throw an error. When it has found a char input from the port, it will be saved in the "direction" variable. After the "direction" variable has been set, it will check if the "direction" is one of four letters (w,a,s or d). If the char in "direction" is one of the following letters, it will set a new value for the "movement" variable. This is then used in the function called "moveCharacter" which uses the "MovePosition" method from the "rigidbody" component 4.25.

```
private void Update()
{
    if (sp.IsOpen)
    {
        try
        {
            direction = System.Convert.ToChar(sp.ReadByte());
        }
        catch (System.Exception)
        {
        }
    }
    if (direction =='W')
    {
        movement = new Vector3(0, 0, 1);
    }
    if (direction == 'S')
    {
        movement = new Vector3(0, 0, -1);
    }
    if (direction == 'A')
    {
        movement = new Vector3(-1, 0, 0);
    }
    if (direction == 'D')
    {
        movement = new Vector3(1, 0, 0);
    }
}
void moveCharacter(Vector3 direction)
{
    rb.MovePosition((Vector3)transform.position + (direction * movementSpeed * Time.deltaTime));
}
```

Figure 4.25: The update and moveCharacter function in "arduinoController" script

This will result in moving the character in one of the four directions with a custom controller as seen in figure 4.14. At the time of implementation, the group did not have access to an Arduino and the components to build it with. Therefore, it was dropped for a new movement script that could be used with a standard computer keyboard.

4.1.4 Evaluation - first test

The goal of the first test is to serve as a pre-test for receiving initial feedback on the concept, sound and game mechanics for the first iteration of the game.

Setup

This section will briefly go over the measures that will be taken to ensure that the test participants are able to properly test the game from their own home.

Before the participants tested the game, they were asked to wear headphones. This will be done to ensure that the participants can hear the sound cues and the direction which they are coming from. They will also be asked if they are alone or otherwise without distractions, to ensure that they are focused on completing the maze and answering the questionnaire.

Procedure

For the first iteration a pre-test questionnaire was prepared. In this test the participants were asked to sign a consent form. The consent form states that the participant agrees to let their responses be used in the report. It also states that their personal information such as name, email or similar information will not be recorded or used. To see the full test go to the appendix 1 7.1.

Then the participants were asked to download the game. Along with the download links to both Windows and Mac versions there was a short tutorial on how to download it and make it run so that they were able to test the game.

When the participants finished the game or gave up they were asked to go and answer the questions for the post-test questionnaire. This concluded the test.

Participants

The tests took place in an online format. Therefore, the recruitment of participants was based around online outreach. This was accomplished by doing network outreach, posting the test in different forums and reaching out to other students at AAU. For the first test, friends and family were asked to test the game. The group also used the Medialogy 4th semester midterm presentation as a forum to get responses from fellow students and course teachers.

Results of first test

For the first test a total of nine responses were received. The participants were asked seven questions about the game and the audio in the game. To find all the results gathered in this pre-test go to appendix 1 7.1.

The feedback received from the test as well as the feedback from the course teachers during the midterm presentation will be the main motivation for the changes in the next iteration of the game. The results that will be presented are the most notable results from this test, and the results that will stand as the basis of some of the changes.

First off the participants were asked *"In your own words, what did you just experience?"*, four participants answered that they were playing a maze, three of those four mentioned that sound, in some form, was part of the game.

The second question was *"What was the purpose of the sound in the game?"*, here five participants answered that the sound was either guiding them to the finish or that it was telling them when they were going in the wrong direction. Although,

two of the participants seemed unsure that they were correct and posed their answer as a question. One participant wrote *"closer to the object the nicer sound?"* and another said *"I guess it should have helped to locate the object"*, this could indicate that the participants were not directly aware of the cues while playing the maze.

The third question asked *"How would you rate the quality of the sound in the game"* these results can be seen in figure 4.26. In the figure the x-axis represents how much the participants agrees with the statement from 1 to 10, where 1 is "bad" and 10 is "good". The y-axis represents the number of people that responded.

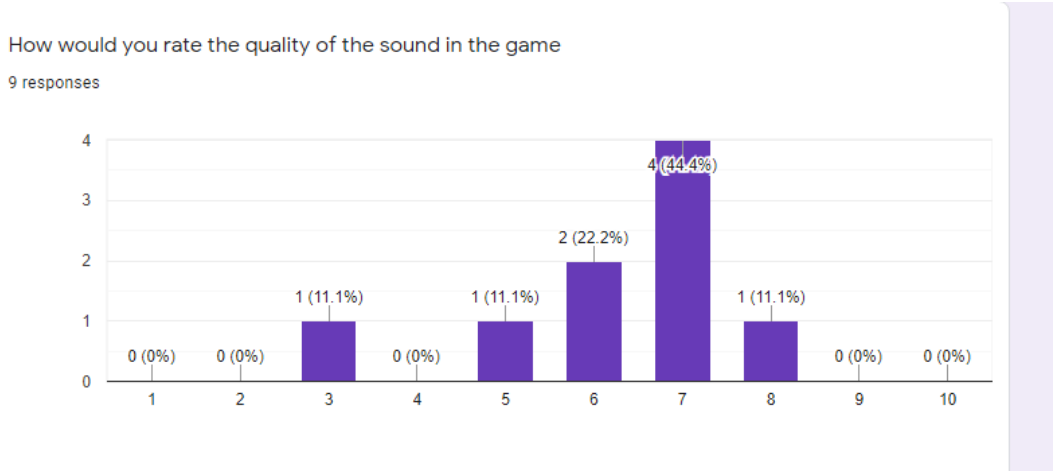


Figure 4.26: The participants rating of the sound in the game

The results in the graph points towards the fact that there should be some improvements made to the sound design 4.26.

Additional feedback from the Audio Processing course teacher was that it might be a good idea to make the finish line a pulsating sound with a frequency in the 1000Hz to 1500Hz range as human hearing is better at locating signals within this range.

Question four asked the participants *"Was the maze too easy?"*. Here three participants said "No" and four said "Yes". This question was supplemented with two sub questions *"What made it easy?"* and *"What made it difficult?"*. Three participants mentioned that the maze was too difficult because they could not see the whole maze or at least too little of the maze. This of course was the purpose of the game so that they would rely on the audio cues. Other participants said it was too easy, and mentioned that the maze was too simple and that the audio cues made it easy to complete. While some participants said that the audio cues did not help at all and some even misunderstood the cues.

Finally, the participants where asked *"Do you have any other comments to add?"*. Here the participants gave suggestions for improvements and constructive feedback. Some of the key elements were that the participants wanted to move faster in the game, that the sound should be more clear when they are moving in the wrong direction and that there was some user interface and sound bugs which should be fixed.

The key elements that will be extracted from this test and implemented into the next iteration is an adjustment of the audio cues and improvements of the in-game interface. There will also be some extra interface buttons in the game, based on the comments made by participants. The view angle in the game will also be altered to give a more limited view of the maze overall.

4.2 Second Iteration

After conducting the test of the first iteration, several points were taken from the feedback and these will be considered for this iteration. The design of the maze, audio design, UI improvements and general implementation of them alongside evaluation will be discussed here.

4.2.1 Design

This section will cover the design decisions based on evaluations of the first test, feedback from the Midterm presentations and group discussions.

Design of the Maze

Concerning the level of difficulty of the maze, some participants from the first test conducted, claimed it was too hard to complete the maze because too little of the maze was visible. Some participants pointed out that it was too easy with the audio cues being in use. However, the purpose was to make the gameplay easy if the audio cues are present.

After reviewing the feedback from the first iteration test, the maze design itself was considered not responsible for the bolstered difficulty. Therefore the design of the maze was not altered.

Instead, the viewing angle was thought to be the problem.

Redesigning the In-Game User Interface

Various viewing angles were tested internally, such as an over the shoulder view and a first person view. It was considered that a 1st person view could prove to be more intuitive, as it would give the user a more familiar experience, specifically with the stereo panning. In figure 4.27 it is showed how the game view appeared from a first person perspective.

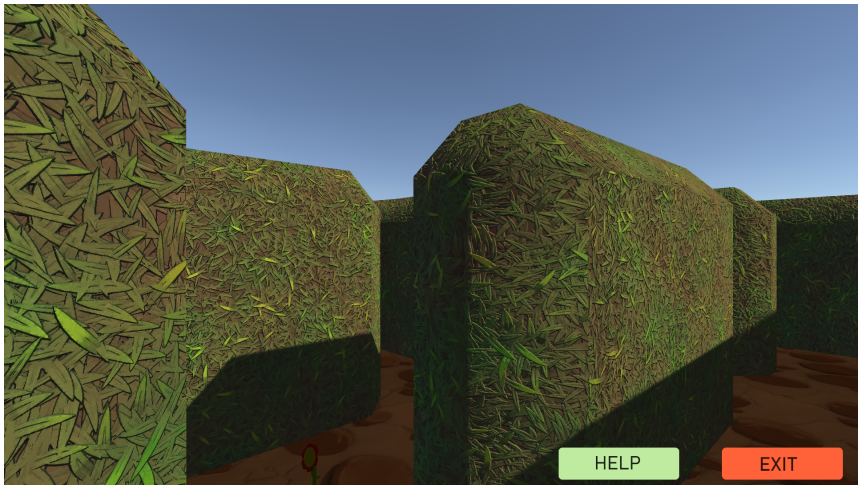


Figure 4.27: First person view in game

After internal group testing, it was decided to return back to the previous version of camera angle. This was due to repetitive textures which was disorientating making the maze harder to navigate. A decision was made to show even less of the maze than in the previous design iteration 4.1. A square vignette should be implemented, allowing for only the immediate surroundings of the player to be visible as seen in figure 4.28.

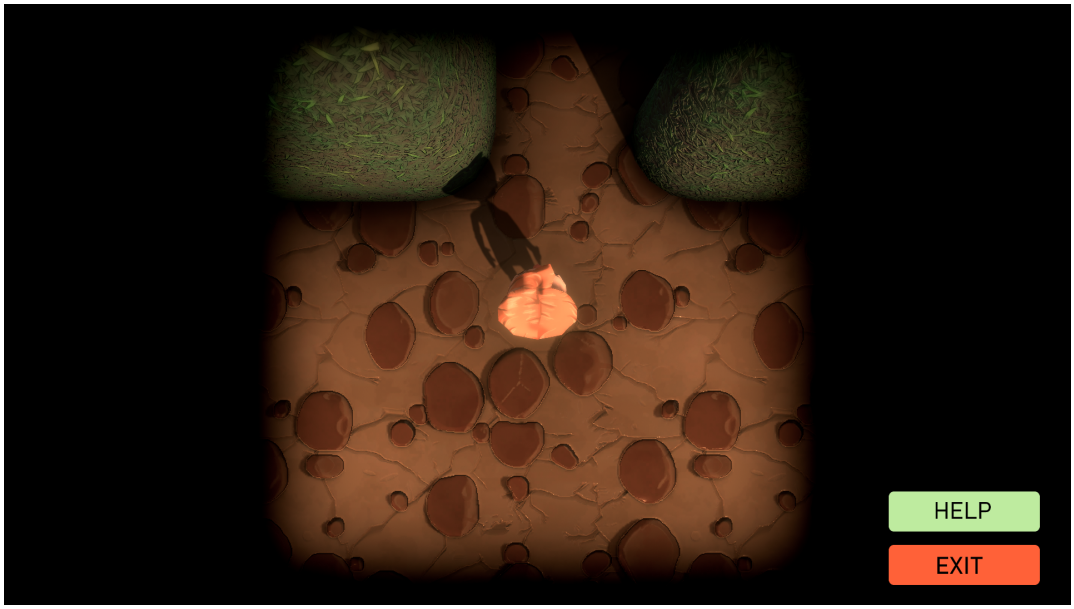


Figure 4.28: In game interface of second iteration

In addition, the speed in which the player was able to move was increased, as that was mentioned in the first evaluation.

The user interface should be updated to accommodate for features that are required for further testing. This includes features such as the code on the main menu that that will be used to match the player's performance data collected by Unity, as explained in the first iteration implementation (4.1.2), to the questionnaire response as seen in the figure 4.29.



Figure 4.29: Main menu of iteration two

Due to the questionnaire response from the first iteration *"It would be nice to have like a question mark button to check what was the purpose of the maze"* a "help" button should be implemented in the game. This will make the instructions, which previously was only available at the very beginning, visible again as seen in figure 4.28.

In case a player is stuck and walking aimlessly around the maze with no ending, or there is a bug in the game. It was decided that there should be an option to quit or give up on the maze, which can be seen in figure 4.30. This would ensure that the participant would be shown their code to input into the questionnaire if they quit early.

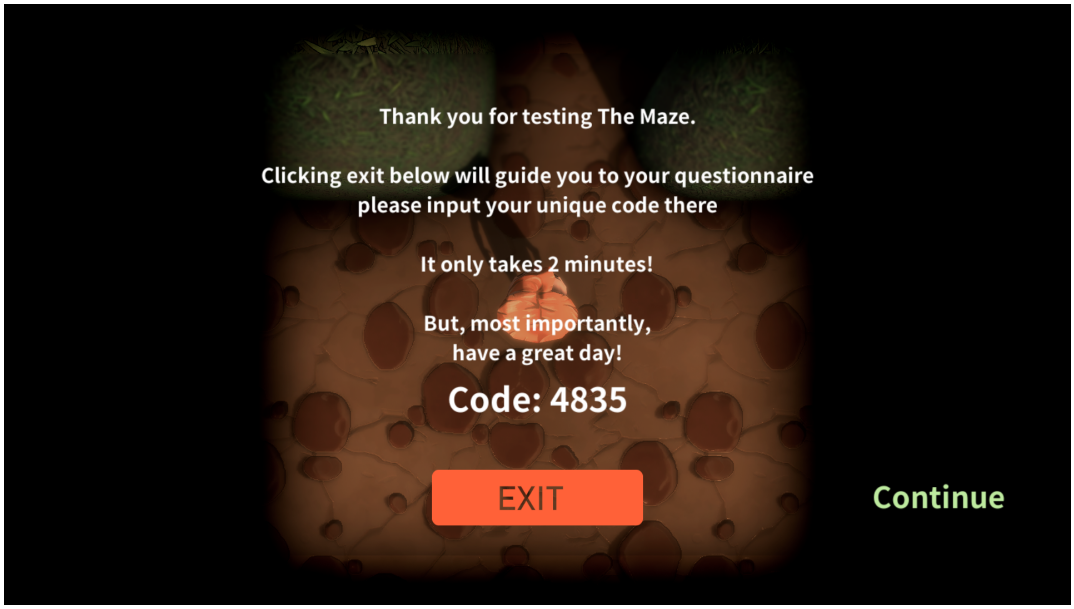


Figure 4.30: If the user quits before completing the maze

Audio design

In the first iteration the pitch of the background sound was being changed when the user took a new path at one of the junctions 4.1. Due to test participants not being able to distinguish between the change of pitch, a decision was made to implement separate sound effects in the form of a positive and negative audio cues which will be outsourced from freesound.org" [6] & [7]. The hidden object at the end of the maze will continue to implement stereo panning but will no longer change in pitch and it will implement a pulsating sound in the range 1000Hz to 1500Hz, see feedback from the initial user test 4.1.4.

Furthermore, the background music should be unintrusive and suitable to play in the background at a low volume without distracting the player. At this point various considerations of public domain music were made and it was decided to use "Leaving Home" by Kevin Macleod [35].

Overall, the core features were agreed to be kept from the previous iteration. However, the design of the user interface has been changed to accommodate new features and has slightly changed in layout. Additionally, positive and negative sounds have been found and will be implemented into the game in the proceeding section. The pulsating orb sound will be kept, with its pitch changed to between 1000 Hz and 1500 Hz.

4.2.2 Implementation

The ideas and changes from the preceding design section will now be implemented along with a new audio program named Wwise. This will be done to meet the requirements for the 2nd test, which will be a usability test. An overview of how the communication between the controller, computer, game and web server can be seen in figure 4.31.

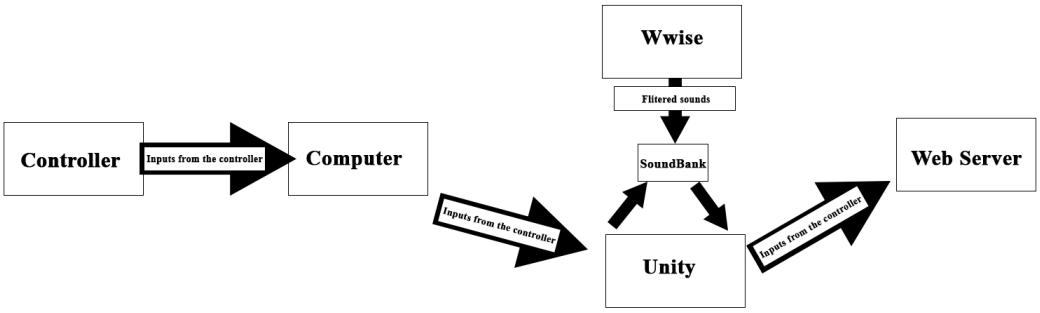


Figure 4.31: Overview of implementation of second iteration

Wwise

Due to complications with integrating Pure Data into Unity, it was suggested to make the switch to another program that could make similar sound effects as Pure Data. The suggested program by the audio processing teacher was Wwise. Wwise is a middleware for interactive media such as games and movies where sound is changing depending on what is happening in the given media. For this project, Wwise will be able to reproduce the audio effects that were made in Pure Data.

All of the sound files that are currently in unity and were planned to be changed by pure data, will be imported into Wwise. In Wwise, within the project explorer window, all the audio files will be placed as a sound SFX file in the actor-mixer hierarchy. From there, one of the audio clips can be selected and its properties can be changed. This includes its volume, pitch, panning and low/high-pass filters can be applied 4.32.

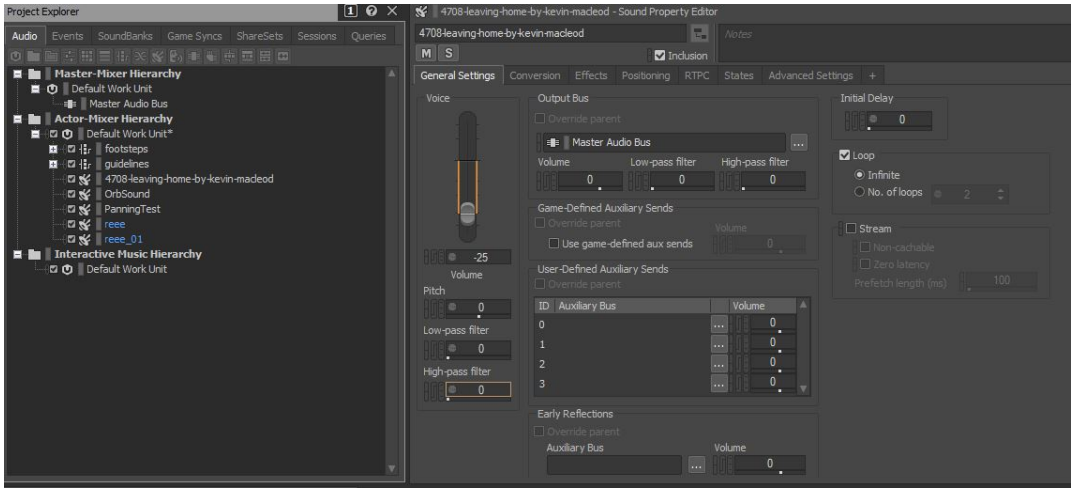


Figure 4.32: Project explorer and sound property editor in Wwise

By default, the background music was too loud, which would drown out the other sounds. As mentioned in the preceding design section (4.2.1), the background music should be unintrusive. Therefore, in the property editor, the volume of the music was attenuated enough until it was just audible.

The footsteps is a sound file with multiple step sounds, but through Wwise, it is possible to spilt the clip into single steps. This is done to have multiple step sounds which can be played in a random order to bolster immersiveness, as seen in figure 4.33.

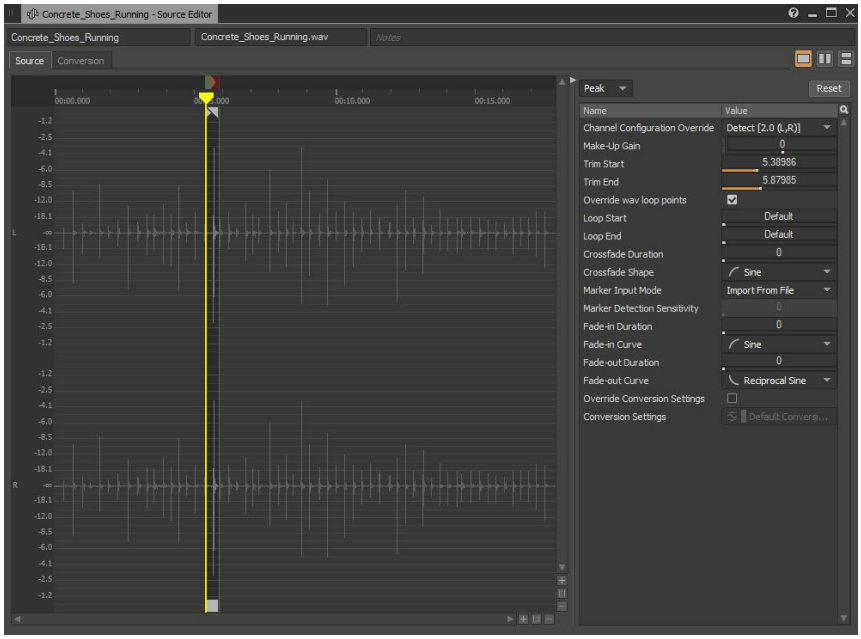


Figure 4.33: Editing indviual sound waves like the footstep clip in Wwise

The individual step sound files were placed under a parent that is called "random container". This "random container" has the same properties as a sound file, but any changes made to the container, will affect all the sound files that are contained inside of it. This is useful as you can simultaneously change properties of all the sound files by adjusting the parameters of the container. The random container has a setting called "play type" where options like "random" or "sequence" can be selected. By selecting sequence, it will play all of the sound clips in the order they are listed as the children to that container. The random option, play a sequence of the children in a random order.

A similar process for the audio cues sound clips can be used by assigning them under a switch container. The switch container works by switching between the positive and negative audio cues. To change between the sounds, a switch must be made in the game syncs section, see figure 4.34. Game syncs are used as parameters when changes are happening in real-time in the media and for this project it is used whether the player is going in the correct or incorrect direction.

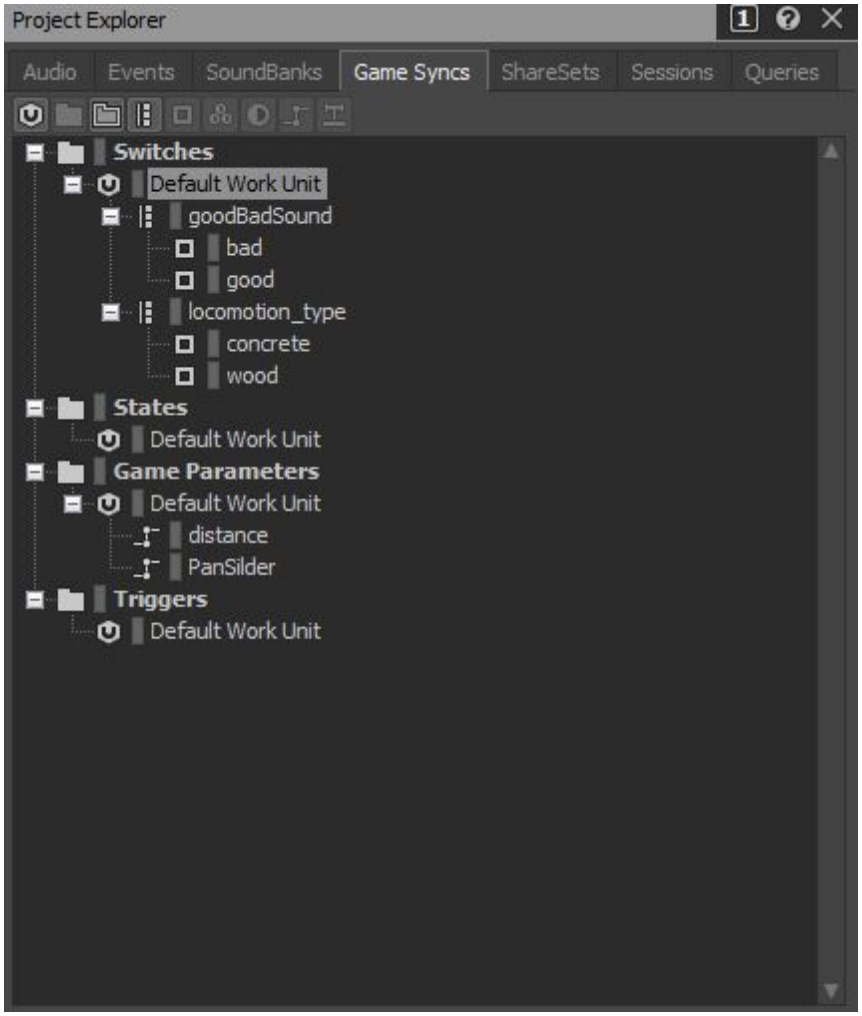


Figure 4.34: Game syncs tab within project explorer in Wwise

In the next section, the end of the maze audio was changed according to the feedback from first iteration, specifically, the panning was changed from being implemented in Unity's audio library to Wwise.

Panning with Wwise

The sound emitted from the hidden object at the end of the maze is important since it helps guide the player. This requires panning and attenuation/amplification, see section (2.6.1) & (2.6.1). In the "sound property editor", in a tab called "positioning", see figure 4.35, the "listener relative routing" was activated and the option "position + orientation for the 3D spatialization" was chosen. This was the basic setup to enable panning and attenuation/amplification based on the location of the "listener", which in this case is the in-game camera.

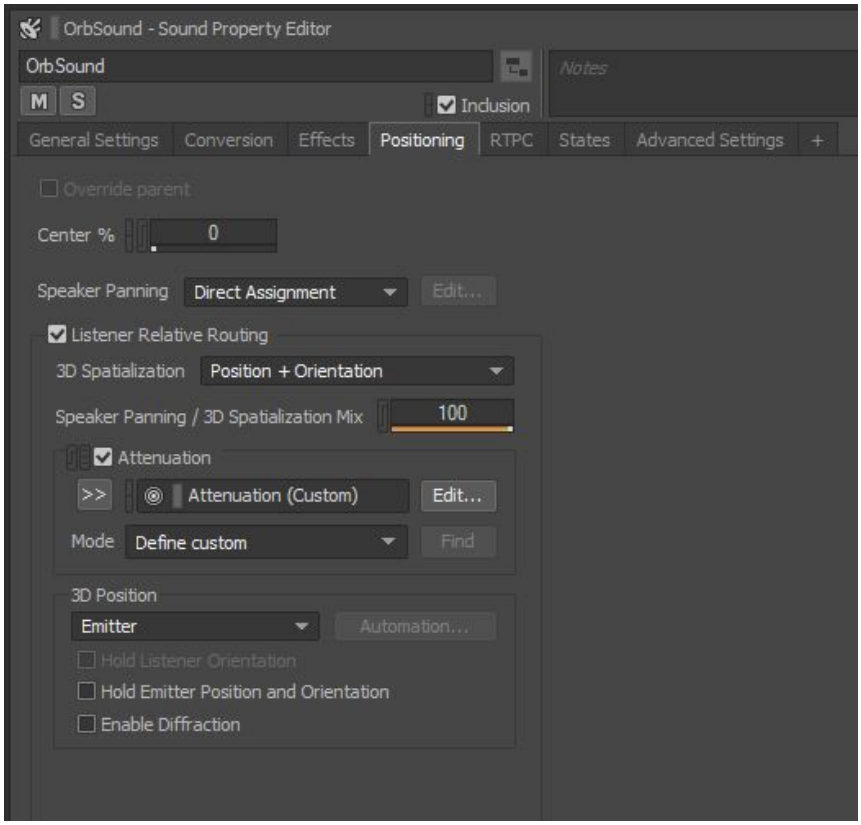


Figure 4.35: the positioning tab in Wwise

In Unity, sounds will either be played through discrete events (for the positive/negative audio cues) or the sounds will be changed on a continuous basis, depending on various variables (the hidden object's sound).

Playing the sounds in Unity

When all the sounds have been made as events, a soundbank must be generated that will contain the sound files, parameters, etc. that were set up in Wwise from the preceding section.

Wwise has a built in function to integrate itself into Unity. Once integrated, several new components replace Unity's native audio system. On the camera, there are two new components called "AkGameObj" and "AkAudioListener". The "AkAudioListener" has replaced the default listener, so it can only listen to audio from the from the Wwise soundbank. The "AkGameObj" is the component that keeps track of the player's position for the parameters that were given in Wwise. A new component will be added to the camera for playing the background music and it is called "AkAmbient".

The footstep sound will be triggered through the animation where a script with a function called "PlayFootStep" will play the appropriate event.

The hidden object's sound will be given two components, where one of them is the "AkGameObj" and the other is "AkEvent". The "AkEvent" will play the event for the hidden object's sound, while the "AkGameObj" will change its volume depending on the distance it is from the player.

The same will happen to the positive and negative sounds when they are triggered by scripts explained in the next section.

Additions to the implementation of the Game in Unity

The upcoming section will discuss the changes of the implementation regarding the game mechanics in Unity.

The tracking of where the player was located in the game was necessary. To implement effective audio cues, the player's location was needed to be known at all times. It was also checked at every junction if the player was on the correct path and if the player was moving the correct way towards the end of the maze. This was implemented as a grid of cubes, as seen in figure 4.36.

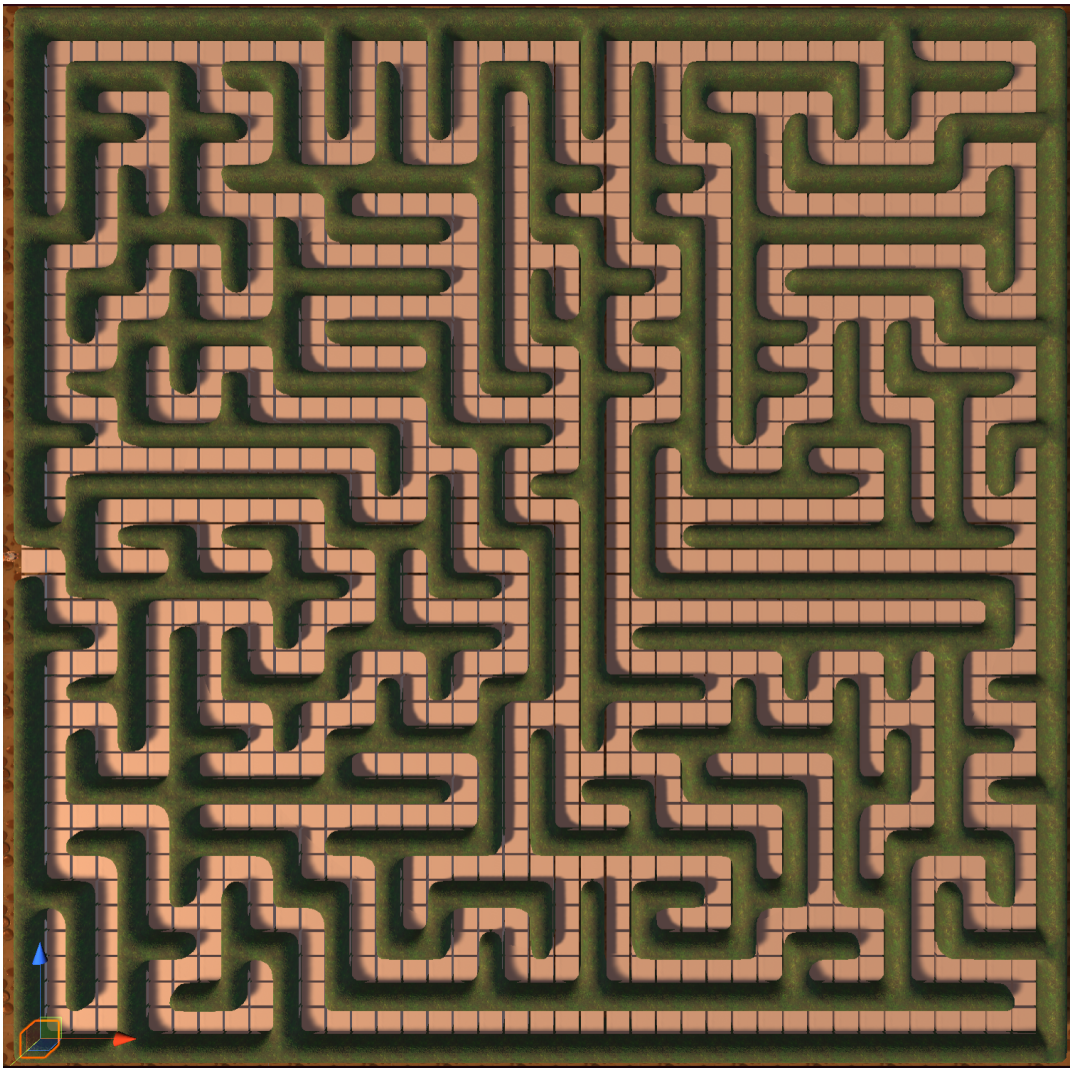


Figure 4.36: Trigger box grid for tracking player

Each cube, has a trigger collider attached and was set to not be visible in the game by not rendering them. These cubes are instantiated at run time and each has a unique ID. A list of correct IDs for the cubes was made from the start to the finish, representing the correct path. As long as the player moves from one cube to the next correct cube, no sound is played. However, as soon as the player deviates from the correct path, this will trigger the Wwise event associated with the negative audio, which will indicate to the player that they are moving in the wrong direction. When the player returns to the correct path, the positive audio cue is triggered.

In the figure 4.37 a start function of the logic script that is used to track the player's

movement can be seen.

```
void Start()
{
    start = boxCollider.transform.position;
    trans = start;

    for (int i = 0; i < 40; i++)
    {
        Instantiate(boxCollider, new Vector3(trans.x, trans.y, trans.z + i * offset), Quaternion.identity);
        for (int j = 0; j < 40; j++)
        {
            GameObject newCell = Instantiate(boxCollider, new Vector3(trans.x + j * offset, trans.y, trans.z + i * offset), Quaternion.identity);
            cells.Add(newCell);
            newCell.GetComponent<Cell>().id = i + j * 40;
        }
    }
    path_1_list.AddRange(path_1_ids.Split(',').Select(i => int.Parse(i)));
}
```

Figure 4.37: Start function of the logic script for tracking the player

A double for-loop is used to instantiate the cubes across the maze with equal distance. The unique ID is then assigned to each individual cube using a formula inspired by how one would identify a pixel using image processing. Outside of the for-loop the cubes that belong to the correct path are added to the "path_1_list", these IDs are assigned in the editor.

In the figure 4.38 below a function which tracks the player can be seen. The function is called when the player collides with a trigger box on one of the cubes. As seen, the function accepts a CellID which is passed from the collided cube's unique ID to check if it is the correct path. When the player exits a cube, that cube's ID is passed to the lastID variable. In this way, the current and previous cube the player is in, is known at all times.

```
public void PathCheck(int cellID, GameObject cellObject)
{
    var index = path_1_list.IndexOf(lastID);

    if (path_1_list.Contains(cellID) && path_1_list.Contains(lastID) && cellID != 19 && cellID != 0)
    {
        if (path_1_list[index + 1] == cellID)
        {

            if (!goodPlayed)
            {
                PlayGood();
            }
            goodPlayed = true;
            badPlayed = false;
        }

        else
        {

            if (!badPlayed)
            {
                PlayBad();
            }
            goodPlayed = false;
            badPlayed = true;
        }
    }

    else if (!path_1_list.Contains(cellID) || !path_1_list.Contains(lastID))
    {
        if (!badPlayed)
        {
            PlayBad();
        }
        goodPlayed = false;
        badPlayed = true;

        CheckMinorChoices(cellID);
    }
}
```

Figure 4.38: The function which tracks the player

Database

As mentioned in section (4.1.2), the game keeps track of the user's time spent in the game and how far they have walked. Due to physical restrictions when testing, participants had to test remotely using their personal computers. Therefore, a database was set up to accommodate the users' performance data. The specific data being uploaded will include a unique id for the user, time taken to complete or quit the game, distance traveled and if they have quit the game before reaching the end, as seen in figure 4.39. To store the data online, one of the group members had a web domain which was linked to a MySQL database.

Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
username	text	utf8_general_ci		No	None			Change Drop More
time	text	utf8_general_ci		No	None			Change Drop More
distance	varchar(128)	utf8_general_ci		No	None			Change Drop More
rageQuit	text	utf8_general_ci		No	None			Change Drop More

Figure 4.39: structure of the database

Send information to database Since Unity can not upload the participant's data directly into the database, it has to be send through another script that can connect to the database. The "middleman" script is a php file that can receive the information from Unity and uploads the given data to the database. In the game, when the user reaches the goal or exits the game before reaching the end, the game will run a script called "databaseSendData". This script starts a coroutine that calls an "IEnumerator" that takes the user's data as parameters as seen in the figure 4.40.

```
private void OnTriggerEnter(Collider other)
{
    if (other.tag == "Player" && sendData == false)
    {
        username = GameObject.Find("userNameHolder").GetComponent<generateUserName>().username;
        usernameHolder.text ="Code: "+ username;
        timeSpent =  other.GetComponent<movementScript>().timeCounter.ToString();
        distanceTraveled = other.GetComponent<movementScript>().distanceCounter.ToString();
        rageQuit = other.GetComponent<movementScript>().ended.ToString();
        StartCoroutine(Upload(username + help.condition, timeSpent, distanceTraveled,rageQuit));
        Debug.Log("timer :" + timeSpent);
        Debug.Log("distance :" + distanceTraveled);
        Debug.Log("username : "+username);
        sendData = true;
    }
}
```

Figure 4.40: "OnTriggerEnter" function in "databaseSendData" script

"StartCoroutine" is a function that can be called multiple times during a frame and can be yielded, this is used to ensure that the data is being sent.

In the "IEnumerator", see figure 4.41, the webform from the networking library is used to prepare the data to be sent and received by the php file.

```
public IEnumerator Upload(string playerName, string timer,string distanceCounter,string rageQuit)
{
    WWWForm form = new WWWForm();
    form.AddField("playerName", playerName);
    form.AddField("timer", timer);
    form.AddField("distance", distanceCounter);
    form.AddField("rageQuit", rageQuit);


    UnityWebRequest www = UnityWebRequest.Post("http://switty.dk/SendData.php", form);
    yield return www.SendWebRequest();

    if (www.isNetworkError || www.isHttpError)
    {
        Debug.Log(www.error);
    }
    else
    {
        Debug.Log("Form upload complete!");
    }
}
```

Figure 4.41: IEnumerator "upload" function in "databaseSendData" script

The webform is a helper class that can generate form data to post to web servers or, in other words, translate the data to a format that can be read by HTTP. After that, the object of "UnityWebRequest" is made and is called "www". The "UnityWebRequest" then sends the webform to a web domain and stores it there.

Now that a connection is established between Unity and the web domain, it is time to look at where the data is being uploaded to in the database as illustrated below, in figure 4.42.

On the web domain, there is a file called "SendData.php" which receives the data from the game. In the beginning of the file, variables with the necessary information to connect to the database is stored. These are "$host", "$username", "$password" and "$database" and are going to be used as parameters for the "$connection" variable to make a new MySQLi connection to the database. Afterwards, the data from Unity, which was stored in the HTTP, is going to be assigned to four new variables.

Once all of this has been setup, an if statement was made to check if MySQLi can connect to the database as seen on figure 4.42.

```php
<?php
$host     =
$username =
$password =
$database =

$connection = new MySQLi($host,$username,$password,$database);

$playerName = $_POST["playerName"];
$timer = $_POST["timer"];
$distance = $_POST["distance"];
$rageQuit = $_POST["rageQuit"];

if(!$connection)
{
    die("connection failed.". mysqli_connect_error());
}

$sql = "INSERT INTO Med4_UserData ( username, time, distance,rageQuit)
        VALUES ('$playerName','$timer','$distance','$rageQuit')";


$result = mysqli_query($connection,$sql);
if(!$result)
{
    echo"there was a error";
}
else
{
    echo "everything is fine!";
}

?>
```

Figure 4.42: sendData.php on webserver

Once a connection is established, a new variable named "$sql" is made that contains a string, which resembles MySQL code. After that, another new variable named "$result" contains the "mysqli_query" function which takes "$connection" and "$sql" variables as parameters. Lastly, there is an if statement to check if the "$result" variable is true. If it is true, it will upload the data to the database.

From the database, all the data from the test participants can be exported in standard formats, such as csv and be evaluated.

4.2.3 Evaluation - Usability test

The second test that will be held is a usability test. In this test the focus will be on usability and sound design for the cues. Furthermore, participants will also be asked about the the overall concept of the game. This is being done in order to

make sure the changes that have been implemented from the first iteration did not change the perception of the game too drastically.

Procedure and setup of the usability test

This test will take place via a website called "Notion.no", here users can set up a list of activities which in this case is a test. First, the participants are greeted and basic information regarding the project group. They will be told how to download the game and try it. Afterwards, they will be asked to sign a consent form, which states that the data collected while playing the game as well as the data from the questionnaire can be used for purposes of the project report. Then the participants will fill out the questionnaire, which is comprised of a list of questions about the game itself and the sound in the game. Unfortunately, the test will be shorter than expected as there will be no test of the physical interface as it is not available. If the physical interface would have been available, a usability test would evaluate upon it in a similar way of how it is done to the gameplay.

Participants

The participants for this test will be selected by the members of the group. They will be either friends or colleagues of the group members. Test participants will be asked to try the game by clicking the link they have been send. The link comes along with a short disclaimer that they must wear a pair of headphones to properly test the game and that the test is for a semester project. They will also be told that the test and questionnaire will take approximately 10-15 minutes in total to complete.

Results of the usability test

For the usability test a total of eight participants responded. All of the participants finished the game. One of the participants had tried an earlier version of the game before, but the rest had not.

The participants were asked *"Were you alone and/or in a place without distractions?"*. Six participants responded *"yes"*, while two responded *"No, had minor distractions, still kept my focus on the game"*.

The participants were then asked *"In your own words what did you just experience?"*. Here the participants for the most part seemed to understand the concept and used keywords such as "maze", "audio cues" and "ping sound", explaining that it was meant to help guide them to the end.

The participants were also asked two "yes" or "no" questions. The questions were *"Do you think the interface was easy to navigate?"* and *"Did you understand the goal of the game while playing?"*. To both of the question all participants responded with a *"yes"*. This is valuable feedback as the group can now move forward with

this concept and design.

In the questionnaire the participants were asked to rate eight questions on a scale from one to seven where one is "completely disagree" and seven is "completely agree". The x-axis represents the participants response and the y-axis represents the number of participants that responded. All of the responses and feedback from the usability test can be seen in the appendix 2 7.2. The graphs that are shown below are some of the notable feedback that was received.

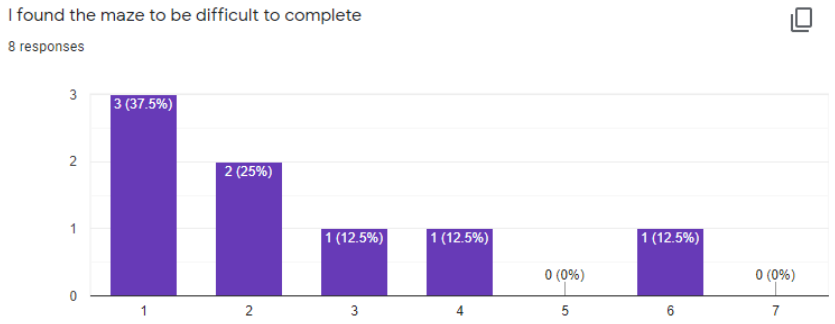


Figure 4.43: Question: I found the maze to be difficult to complete

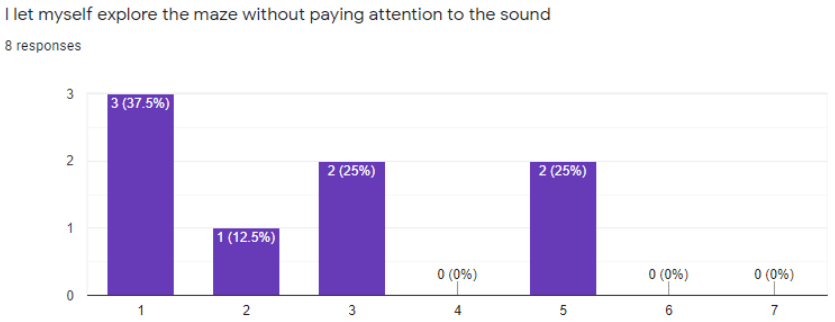


Figure 4.44: Question: I let myself explore the maze without paying attention to the sound

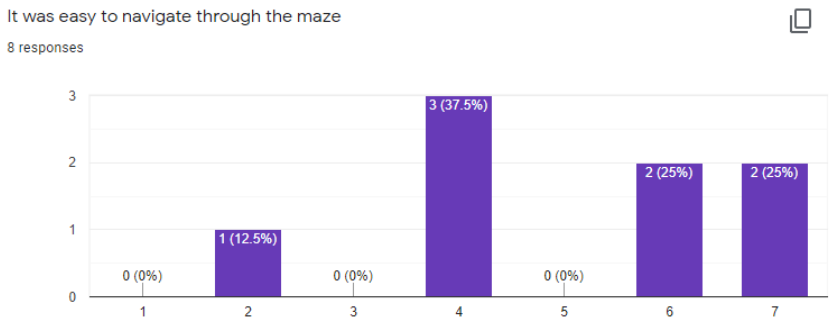


Figure 4.45: Question: It was easy to navigate through the maze

To finish this section of questions, the participants were asked to post any remarks or comments. Some participants remarked that the footsteps were a bit too loud. Others stated that the audio cues stopped working during the maze but seemingly came back later.

The sound section of the questionnaire consisted of three questions. The first question was *"Did you notice the change in sound?"*. Here seven of the eight participants responded with "yes", while one said "no".

The final question was *"If you noticed any changes, what did the change in sound mean?"*. The participants answered these questions correctly and thereby showed that they understood the purpose of the sound. The participants mentioned specifically that depending on the direction they took in the maze, a wrong or right turn, a good or bad sound would be played.

The main points to take away from this test, is that is seems to function as intended with minor flaws that needs to be fixed in the following iteration. The participants were positive about the experience and most of the participants seemed to find the game intuitive and easy to understand, with some participants still having a challenge. For the next iteration, the group will work on implementing the sound cues properly all over the map and improve the user interface.

4.3 Third Iteration

The third and last iteration is for the final A/B test. Here minor changes are made to the design and implementation. This section will end with a final evaluation based on the results of this test.

4.3.1 Design

Based on the usability test conducted in the previous iteration, multiple test participants agreed that it was not too difficult to complete the maze, while a few thought it was relatively difficult. Therefore, the project group decided to keep the design of the maze the same, as it may prove to be too complex if the audio cues were not present. The focus of this iteration will be on improving the audio design and the procedure of the online evaluation for the final implementation and evaluation.

Audio design

Based on the feedback mentioned at the end of the evaluation of iteration two, the audio cues were not present everywhere in the maze, see (4.2.3). Therefore, the positive and negative cues should be triggered by minor choices, as well as major choices, as seen in figure 4.46. These cues will guide the player back to the correct path if they have strayed away from it.

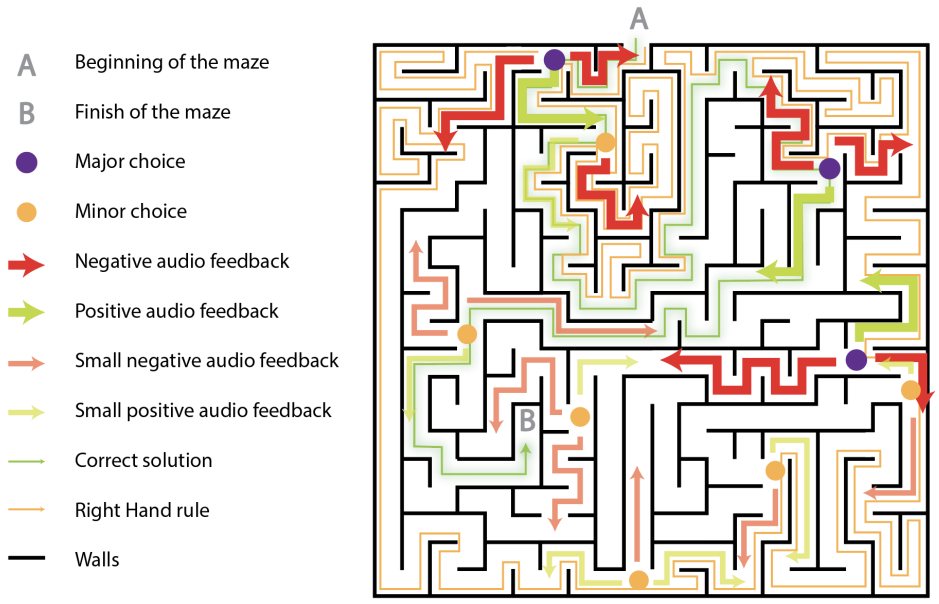


Figure 4.46: Use of audio cues and positioning of them

Further changes will be discussed in the implementation 4.3.2.

Final improvements of the In-Game User Interface

This section will outline the changes in design that were made to the user interface. In this last iteration a few important changes were made to allow for A/B testing.

Responses from test participants after the usability test were considered and it was clear that not a substantial amount needed to change. The next test, however, needs to accommodate for A/B testing. First and foremost, it was decided to have one file that the participant would download. Therefore, a system is required to govern which participants are in group A and which are in B. As mentioned in the second iteration implementation (4.2.2), a random code generator has already been created to match the player's performance data with their corresponding questionnaire responses. When this code is initially generated, Unity will check if the number is even or odd. If the number is even then the player will be placed in group A (the group with audio cues) and vice versa. Having this system reduces human error, as most things are handled by the game itself. Also, the number of participants in each group should be as equal as possible. Using a random number generator and assigning people based on whether their number is even or odd allows the process to remain as random and equal as possible.

Additionally, for convenience, the "exit" button will not exit the game immediately, but rather send the user to the appropriate questionnaire based on their group A or B.

Furthermore, when this exit button is clicked, to avoid participants forgetting their code, the game window will reduce in size and show their unique code. This is based on the *"confront nudges"*, see 2.3 - *"throttling mindless activity"* and *"reminding of the consequences"* by using a delay for the process. It will state that they may close this window when they are ready as seen in figure 4.47.

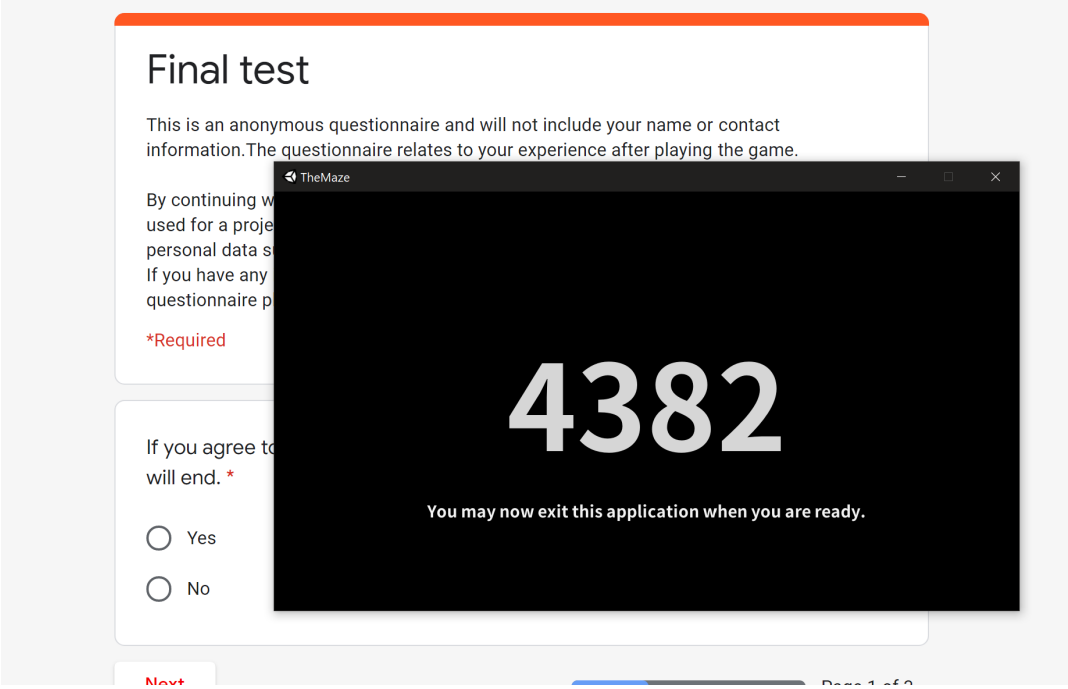


Figure 4.47: Participant is sent to questionnaire and shown their unique code.

The following section will describe how these changes will be implemented into the final iteration of the game.

4.3.2 Implementation

To prepare for the final evaluation, the database needs to be slightly altered, the user interface changes outlined in the previous section and the minor choice audio cues need to be implemented.

Final improvements of the implementation for the Game in Unity

The implementation of the game mechanics in the previous iterations has already satisfied most of the requirements. For this final iteration the maze will be polished and optimised for A/B testing.

As seen in the figure 4.46, there are also minor choices that the player can make, if they have deviated from the correct path. The function described and implemented in the second iteration, see figure 4.38, did not implement audio cues for these choices, therefore, a minor choices function was made. The minor choices were input manually. The figure 4.48 shows one minor choice check. The first if statement checks if the player is currently in one of the wrong cubes after they have left the "junction" where they must make a choice.

```
private void CheckMinorChoices(int cell) {


    if (cell == 48 || cell == 89)//Wrong cells
    {
        if (lastID == 49)
        {
            PlayBad();
        }
    }
    else if(cell == 50 && lastID == 49)//Correct cells
    {
        PlayGood();
    }
    ////////////////////////////////////////////
```

Figure 4.48: The function which governs the minor choices

The nested if-statement checks if they have come from the cube where they must make a choice, if they have the appropriate audio cue is triggered. Finally the else if statement checks if they have entered the correct cube and have come from the cube in which they must make a choice, if so, the appropriate audio cue is triggered.

Below, in figure 4.49, the function which is called when the user quits the game or completes the maze is shown. A simple if-statement checks if the username, which is a four digit integer, is even. If this returns true, the user is from group A and the appropriate link is opened. If it returns false the questionnaire for group B is opened. Below the if/else-statement, the screen resolution is set to a smaller value and full screen is disabled.

```
public void EndGame()
{
    print(usernameInt);
    if (usernameInt % 2 == 0) // A Test (Audio)
    {
        Application.OpenURL("https://forms.gle/YKvXj2NwqAsw24xT7");
        print("Option 1");
    }

    else // B Test (No Audio)
    {
        Application.OpenURL("https://forms.gle/XiNXVMwZpbaivMndA");
        print("Option 2");
    }

    print("quit");

    Screen.SetResolution(1280, 720, false);
    codePanelText.text = playerData.username;
    rageEndScreen.SetActive(false);
    codePanel.SetActive(true);
```

Figure 4.49: The function which sends the user to the appropriate questionnaire and resizes their game window.

Now that the necessary functionality for A/B testing has been implemented, a final test can be made and evaluated upon.

4.3.3 Evaluation - Final test

The goal of the final test was to find out whether or not the audio cues increased the performance of test participants. It was done by comparing the performance of the test participants who did and did not have audio cues. The independent variable in this experiment was the audio cues. The dependent variable is the performance of the player measured in time and path length. Alongside the performance data, a questionnaire was added in order to support a discussion of the results.

The final problem statement will be written as a null hypothesis and an alternative hypothesis to perform statistical analysis upon it. The final problem statement reads: *"How can audio processing effects be used to make audio cues that assist a player navigating a digital 3D maze, thereby decreasing the path length and time taken to complete it?"*. When written as null and alternative hypotheses they read:

Null hypothesis: Audio cues do not increase the performance of a player solving a digital 3D maze.

Alternative hypothesis: Audio cues do increase the performance of a player solving a digital 3D maze.

Participants

Convenience sampling was used to find participants for the final test who were sampled from online communities and directed to a website containing a guide on how to perform the test. In total 43 participants tried the game, 21 played the game without the audio cues and 22 played the game with audio cues. However, only eight of the participants that played the game without sound cues completed the game, while 20 participants completed the game with the audio clues.

Procedure of the A/B test

As the name suggests, there were two tests the participants were randomly assigned to. An A test which was the game with audio cues enabled and a B test which did not include audio cues. The game implemented a tool for measuring the time taken and the length of the path taken by the test participant 4.2.2. This data - time and path length, produced ratio data used for statistical analysis. The test was performed between groups, as the test participants only tested one version of the game. Once recruited, test participants were sent to a website built to facilitate the introduction of the test as seen in the figure 4.50. Here is the download instructions and a questionnaire to which a link was given after the participant exited the game [1].

[1] http://tiny.cc/NotionFinal



Figure 4.50: Introduction to the test

For each participant, a four digit code was randomly generated which the participant entered when they were asked to answer the questionnaire after playing the game. This code also determined whether the participant played with or without sound cues. This ensured that the grouping was random as mentioned in the section (4.3.1).

Results

In this section, the data sets from the A and B test will be reported, and the following section will analyse and conclude upon these tests. Both data sets consist of 16 values. Eight of which describe the time taken to complete the maze for the A group testing with audio cues. The other eight describe the time taken to complete the maze for the B group testing without audio cues. The data set contains 16 values in total. The data sets only include data from the participants who managed to finish the game, which means some of the responses from the A test had to be randomly left out. This was done by numbering the 20 responses and then using a random number generator to select 8 responses which would be analysed along the 8 responses from the B test. The figure below 4.51 shows the process of randomly choosing between the 20 data points.

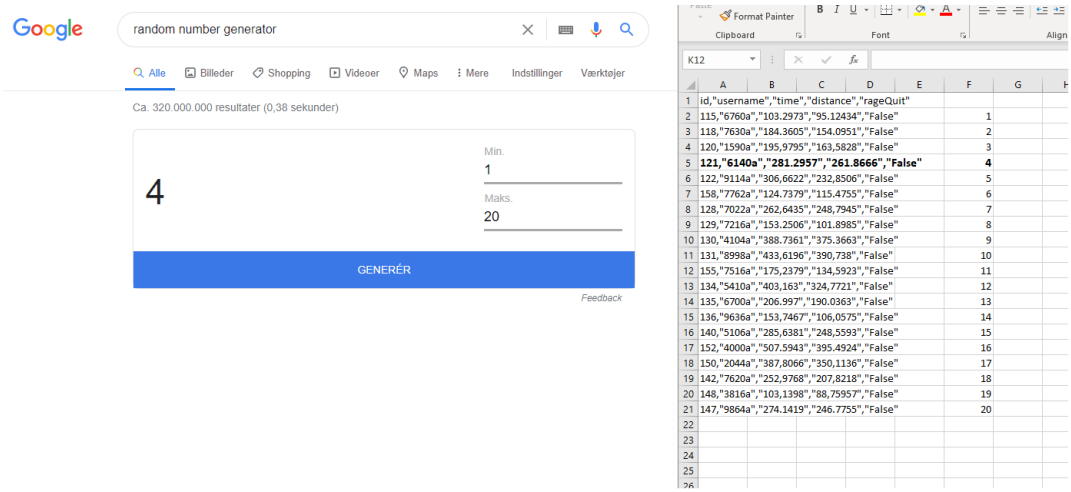


Figure 4.51: Choosing 8 responses at random for statistical analysis.

The following analysis will include descriptive statistics, various plots and graphs that visualise aspects, such as normal distribution and histograms. It will also include testing assumptions for whether or not the data is parametric and perform tests based on these sub conclusions. All of this is done in order to either reject or accept the null and alternative hypotheses.

Below are the descriptive statistics for the time taken and the path length for both groups A and B 4.52.

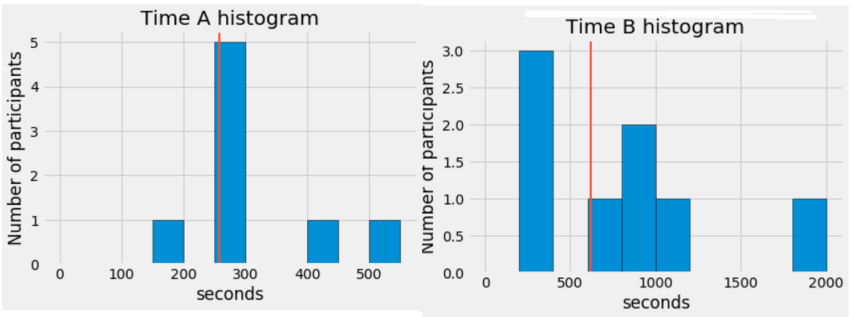
	A	B
count	8.000000	8.000000
mean	305.336400	803.389338
std	102.752969	507.909514
min	175.237900	295.963400
25%	260.226825	347.503825
50%	277.718800	808.064900
75%	315.019325	976.995900
max	507.594300	1814.815000

	A	B
count	8.000000	8.000000
mean	258.584313	617.746438
std	76.924882	295.072889
min	134.592300	233.192900
25%	237.037075	320.824475
50%	248.676900	670.640100
75%	277.592975	897.847625
max	395.492400	922.972800

Figure 4.52: Description of time data on the right, path length data on the left.

The most noteworthy takeaway from the descriptive presentation of the data above, is the significant variations in means and standard deviation between the two groups, both in regards to time and path length.

Below the time and path length values for each group and individual participants is visualised by plotting them into histograms, for both group A and B (4.53) & (4.54).



q

Figure 4.53: Group A (50 second bin width) and B (200 second bin width) histogram for time taken, red line represents the mean of each group.

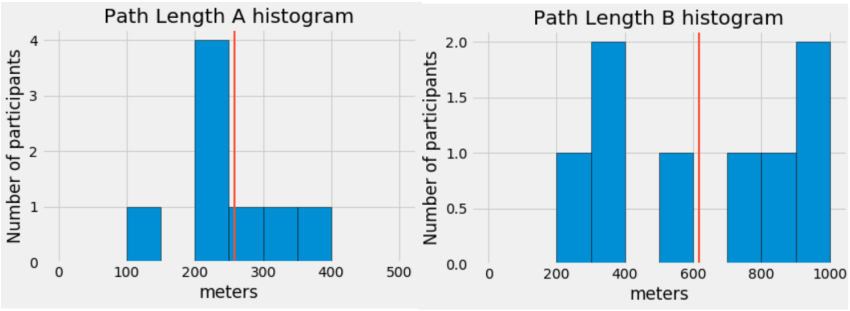


Figure 4.54: Group A (100 meter bin width) and B (100 meter bin width) histogram for path length, red line represents the mean of each group.

To be able to reject or accept the hypotheses, these will be tested using a t-test. To do this, the data gathered must be parametric, which includes confirming various assumption about the data. The level of measurement is ratio data since the dependent variables measured are time and path length. The data also needs to be both normally distributed as well as of equal variance. To test if the data is normally distributed, a probability plot as well as a Shapiro-Wilk test will be performed. Below, the probability plots for time taken and path length are shown 4.55 and 4.56. Each data point (blue dots) were compared to a theoretical normal distribution (red line).

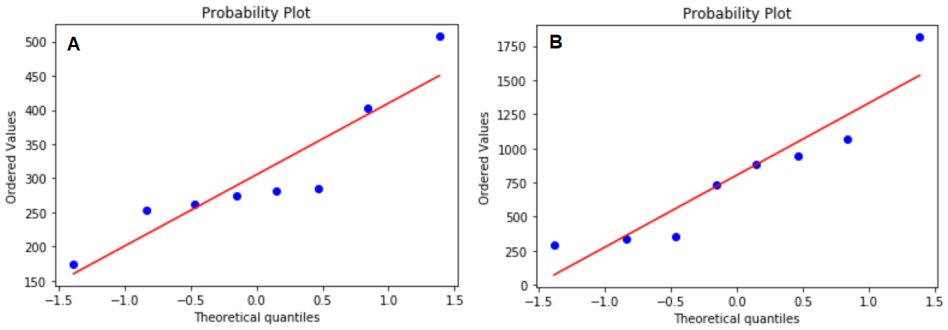


Figure 4.55: Probability plot for time taken.

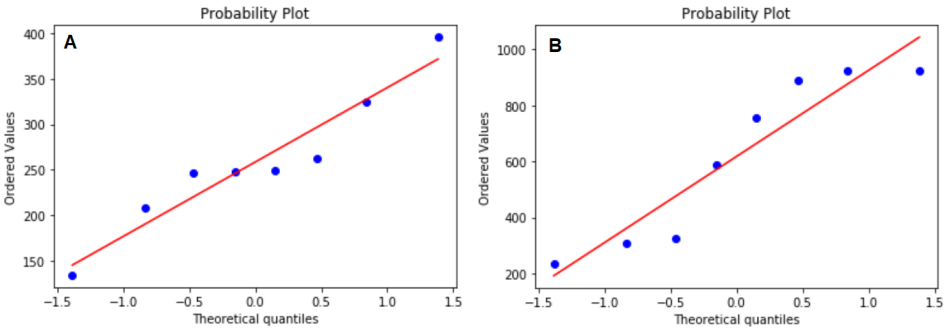


Figure 4.56: Probability plot for path length.

In the probability plots concerning time taken, it can be seen that the data somewhat follows a normal distribution for both group A and B. However, there are some outlying points in each group. In the probability plots 4.56 concerning path length, data points were also somewhat grouped around the theoretical normal distribution.

To further prove that the data was normally distributed a Shapiro-Wilk test will be performed as seen in the figure 4.57.

```
w1,p1=stats.shapiro(time.A)
print('Time A p-value =', "%.2f" %p1)
```

```
Time A p-value = 0.14
```

```
w2,p2=stats.shapiro(time.B)
print('Time B p-value =', "%.2f" %p2)
```

```
Time B p-value = 0.21
```

```
w3,p3=stats.shapiro(pathLength.A)
print('Path Length A p-value =', "%.2f" %p3)
```

```
Path Length A p-value = 0.63
```

```
w4,p4=stats.shapiro(pathLength.B)
print('Path Length B p-value =', "%.2f" %p4)
```

```
Path Length B p-value = 0.10
```

Figure 4.57: Code and results for the Shapiro-Wilk test.

This test assumed a 5% significance level and tests the null hypothesis that the data is normally distributed. The results of the Shapiro-Wilk test (time A p-value = 0.14, time B p-value = 0.21, path length A p = 0.63, path length B p = 0.10, where p - probability) accepts the null hypothesis that the data is normally distributed.

The Levene's test will be performed to test the null hypothesis that the data is of equal variance, see figure 4.58.

```
L1,psk1=stats.levene(time.A.dropna(), time.B.dropna())
print('Levenes test - Time A & B p-value =', "%.2f" %psk1)
```

```
Levenes test - Time A & B p-value = 0.02
```

```
L2,psk2=stats.levene(pathLength.A.dropna(), pathLength.B.dropna())
print('Levenes test - Path Length A & B p-value =', "%.4f" %psk2)
```

```
Levenes test - Path Length A & B p-value = 0.0010
```

Figure 4.58: Code and results for the Levene's test.

This test also assumes a significance level of 5%. The results (time taken p = 0.02, path length p = 0.001) rejects the null hypothesis that the data is of equal variance 4.58. These results mean that the assumptions needed for the data to be parametric are not met, which in turn means that the data is non-parametric.

Below, the standard error of the mean has been calculated 4.59.

```
wsem1=stats.sem(time)
print('Standard error of mean A =', "%.2f" %wsem1[0],', B =', "%.2f" %wsem1[1])
```

```
Standard error of mean A = 36.33 , B = 179.57
```

```
wsem2=stats.sem(pathLength)
print('Standard error of mean A =', "%.2f" %wsem2[0],', B =', "%.2f" %wsem2[1])
```

```
Standard error of mean A = 27.20 , B = 104.32
```

Figure 4.59: Code and result for the standard error of the mean.

The standard error of the mean (SEM) for time taken in group A was = 36.33 and for group B it was = 179.57. SEM for the path length in group A was = 27.20 and for group B it was = 104.32 4.59.

Non-parametric Welch T-test

Since the data is non-parametric, an independent Welch t-test was performed. This test does not assume equal variances in the data sets. In the Welch t-test the group assumed a significance level of 5%. It was then compared between the two sample means for both sets of data in order to test out the null and alternative hypotheses.

The results from this test does support rejecting the original null hypothesis posted above. Which states that there is no performance difference for users with or without audio cues.

The results found after performing this two tailed independent Welch t-test, the average time taken with the audio cues (M = 258.58 seconds, SEM = 36.33) were significantly lower than the time taken without the audio cues (M=617.75 seconds, SEM = 179.57), (t(7) = 2.72, p = 0.03) as seen in the figure 4.60.

```
t1, p1 = stats.ttest_ind(time.B.dropna(), time.A.dropna(), equal_var = False)
print('Independent Welch t-test (two tailed) for time t value = ', "%.2f"%t1,', p value =', "%.2f" %p1)
```

```
Independent Welch t-test (two tailed) for time t value =  2.72 , p value = 0.03
```

```
t2, p2 = stats.ttest_ind(pathLength.B.dropna(), pathLength.A.dropna(), equal_var = False)
print('Independent Welch t-test (two tailed) for path length t value = ', "%.4f"%t2,', p value =', "%.4f" %p2)
```

```
Independent Welch t-test (two tailed) for path length t value =  3.3314 , p value = 0.0105
```

Figure 4.60: Code and results for the independent Welch t-test.

Similarly with the path length of test participants, the results of the Welch t-test shows that the average path length with the audio cues (M = 305.34 meters, SEM = 27.20) were significantly lower than without audio cues (M = 803.39 meters, SEM = 104.32), (t(7) = 3.33, p = 0.0105) figure 4.60.

In the figures below 4.61 and 4.62, both data sets are visualized through box plots.

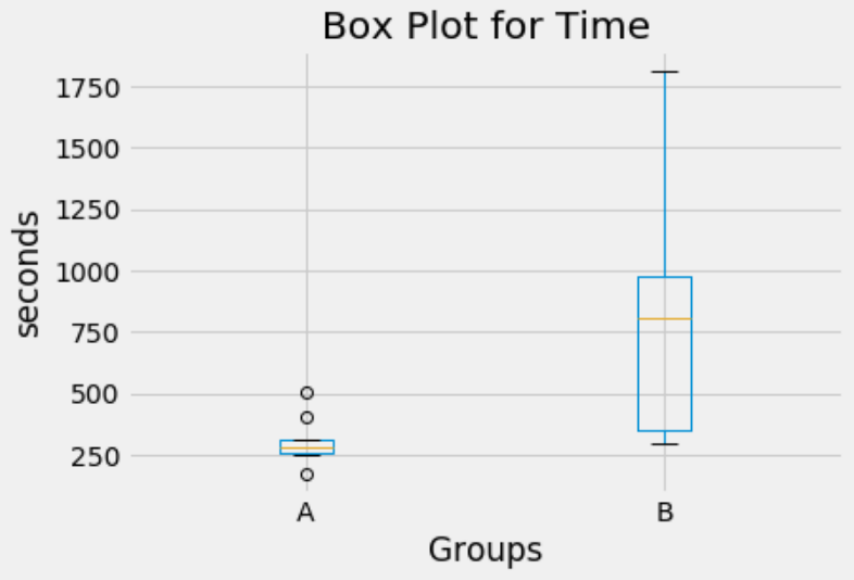


Figure 4.61: Box plot for Time.

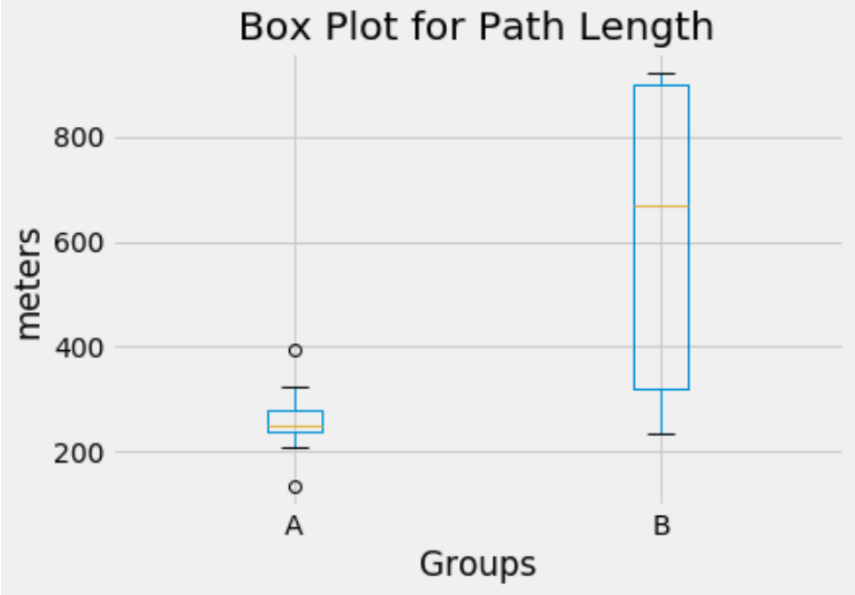


Figure 4.62: Box plot for path length.

The most noteworthy aspect of these box plots are the apparent difference in both the interquartile range as well as the minimum and maximum range of the whiskers. Group A has a much lower variability compared to that of the B group. This could help support the acceptance of the alternative hypothesis.

Next the questionnaire and its results will be reviewed to further support a rejection of acceptance of the null and alternative hypotheses.

Final Test Questionnaire Results

For the final test, two versions of a questionnaire was made. These questionnaires consisted of mostly the same questions with some differences, for example, in the questionnaire given to participants testing without sound it did not include questions about how the sound helped them navigate the maze.

The questionnaire was presented to the participants directly after they had either exited the game or completed it. Here they were met with a consent form. Both of the questionnaires would then ask the participants 13 questions about the game, their experience in the game and about the sound and its cues. The participants were also asked about their testing environment as well as whether or not they had tested previous iterations.

In the following two sections the results of the two questionnaires will be presented individually. To see the complete results, as well as the questionnaire see

the appendix 3 7.3.

Results From the Questionnaire From Participants in Group A, with audio cues

In the first questionnaire a total of 22 participants had responded. Out of these 22 participants 20 had finished the game and 2 had not, as can be seen in figure 4.63.

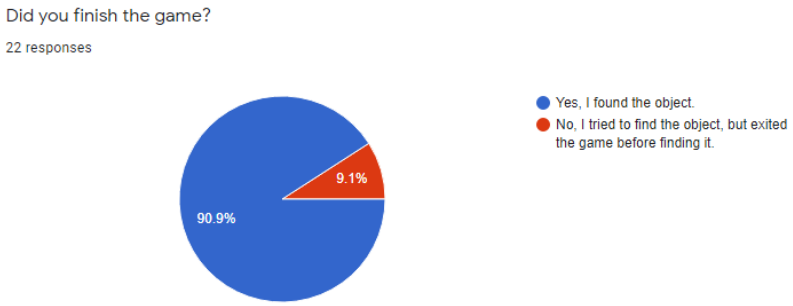


Figure 4.63: Question: Did you finish the game?

Following this, the participants were asked to respond to the following question *"Please briefly describe the environment where you did the test and whether you experienced any disturbances under way"*. Here the vast majority of the participants responded they were alone, in front of their computer and without disturbances. Some participants seemed to have misunderstood the question though, and thought it related to the in-game environment. These few participants seemingly responded correctly to the other questions, so perhaps the question was poorly formulated or too long. Following this question, the participants were asked *"Have you participated in any previous testing of this game?"*. Here just two of the participants responded "yes", those who responded "yes" did not have their time and path length used for the statistical analysis above.

The next two questions were in relation to the audio cues in the game. The first question was *"Were the audio cues necessary to guide you through the maze?"* here 21 participants responded "yes" while 1 participants responded "no". The second question was *"In your opinion, would you prefer to have more or less audio cues to help you find the object in the maze?"*. Here the participants seemed to be in somewhat of a disagreement. 9 participants said that they would like the same amount of audio cues, another 9 responded that they would have liked more audio cues and finally, 4 responded that they would have liked less audio cues.

The following 6 questions in the questionnaire were presented in the form of a likert scale. Each question asks the participants to respond to what degree they agree with the statements. 1 stands for *"completely disagree"* and 7 stands for *"completely*

agree". The x-axis represents the participants response and the y-axis represents the number of participants that responded. The graphs presented below are those which stand out in regards to the hypotheses. These graphs show the responses of the participants.

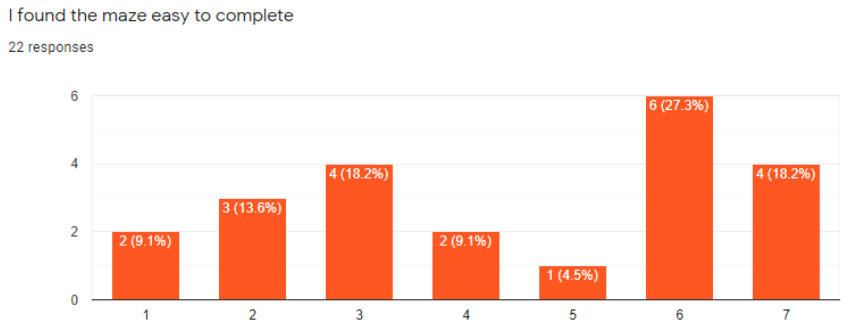


Figure 4.64: Question: I found the maze easy to complete

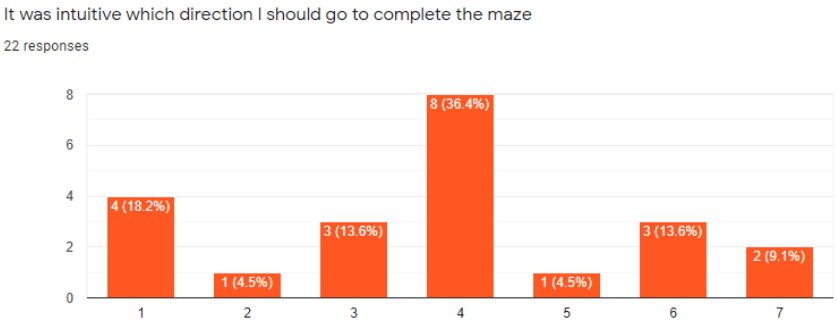


Figure 4.65: Question: It was intuitive which direction I should go to complete the maze

A final question was posed to the participants, which was *"If you have any further comments, please write them below"*. Here the participants were very helpful in pointing out ideas for improvements, remarks about how the game feels and their experience testing. Some of the more notable responses were, that the game did a great job visually of placing the participants inside the maze, but that the sound of the footsteps ruined that for some of the participants. It was remarked that the footsteps were out of sync and sounded like they were walking on wooden planks, while the game looked to be outside in a garden with soil beneath their feet. Other participants were seemingly positive about the sound in the game as well as the feel of the game and noted that as soon as they understood what the audio cues resembled, they found it easy to reach the end of the maze. This overall aligns itself with the goal for this test and the game, and could be an indication

that the game successfully assisted the players to the end of the maze guiding the players with audio cues.

Results From the Questionnaire From Participants Without audio cues

For the B test or the test without sound, 21 participants responded. Out of those 21 participants only 8 completed the maze, while 13 did not. This can be seen in figure 4.66

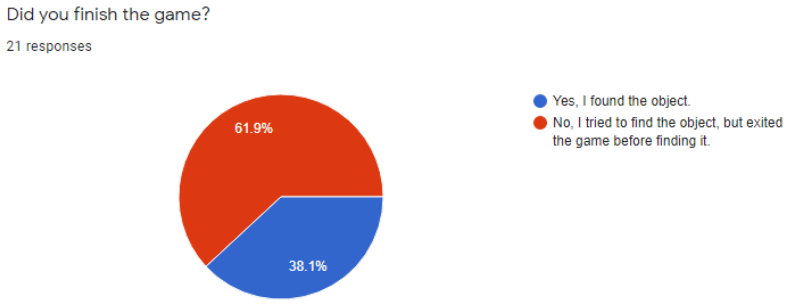


Figure 4.66: Question: Did you finish the game?

Here the participants were also asked *"Please briefly describe the environment where you did the test and whether you experienced any disturbances under way"*. Here only 1 participant misunderstood the question and answered the question in relation to in-game environment. The other participants were at home without distractions except for 1, whom was distracted a few times while trying the game.

For this test only 1 participant had tried a previous version before, out of 21 participants, again, this participant's time and path length data was not used for any statistical analysis above.

The following two questions were in relation to the choices they made in the game. The first question was *"Did you feel uncertain about making choices during the game?"*. Here 15 participants responded "yes", while 6 responded "no". The second question was *"Would you have appreciated audio cues to help you navigate the maze?"*. Here 18 of the 21 participants responded "yes" and 3 responded "no".

The following 6 questions are similar to the 6 questions asked in the "A" test, but some questions were altered to fit this version of the test. For these questions the participants also had to respond how much they agreed with a statement on a scale from 1 to 7, where 1 is *"completely disagree"* and 7 is *"completely agree"*. The x-axis represents the participants responses and the y-axis represents the number

of participants that responded. Below, the most noteworthy responses will be presented as graphs. To see all the graphs as well as the full test, see the appendix 3 7.3.

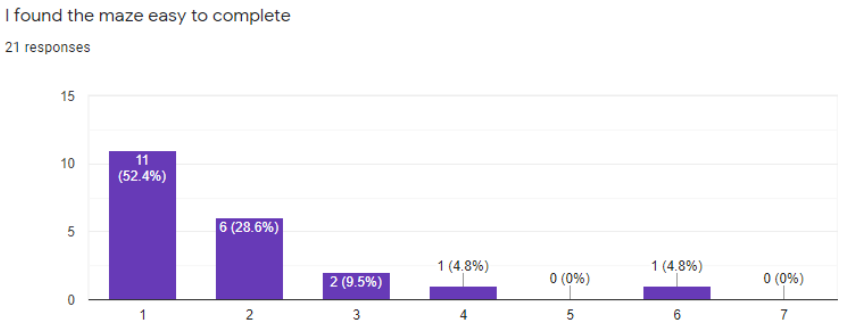


Figure 4.67: Question: I found the maze easy to complete

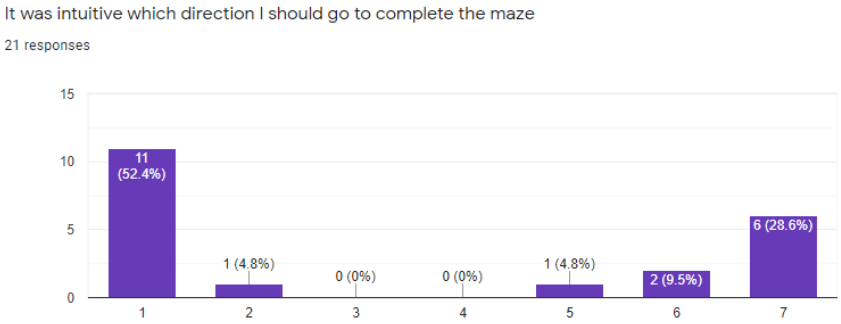


Figure 4.68: Question: It was intuitive which direction I should go to complete the maze

The final question of this questionnaire was *"If you have any further comments, please write them below."*. Here the participants had mixed responses, ranging from disliking the game and giving constructive criticism, to complimenting the graphics and wondering if the background music was meant to guide them in some way.

In the following section, some key comparisons between the two tests will be made, as well as some calculations of the responses to the likert items.

Result Comparison Between the Questionnaires

The key points for comparison between the two tests are as follows. In the question *"Did you finish the game?"* a major difference was noted between the test groups. On one side 20 out of 22 participants finished the game with sound cues, only 8 completed it without the sound cues. This data favours and acceptance of the

alternative hypothesis, in the way that the players were assisted by the audio cues.

In the final likert scale question for both of the tests, see fig. (4.65) & (4.68), the participants were asked *"It was intuitive which direction I should go to complete the maze"*. Here there is an interesting collision between the two tests, as group A answered 3.82 on average and group B answered 3.43 on average. Although, when taking a closer look at the two graphs it can be seen that the B test is split to the extremes, with the majority picking either "1" or "7". While in the A test, a majority is centered around the middle of the scale. This can be related to either a misunderstanding of the question or the personal interpretation of the word *intuitive*. Some participants might think of intuition as simply going left or right down a path, while others relate intuition to the audio cues. With that said, group A still had a higher average, above the neutral 3.5 average of the 7 point likert scale, while group B was below this average.

Final Test Conclusion

The data presented in the sections above supports a rejection of the null hypothesis in favor of the alternative hypothesis that audio cues did improve the performance of a player.

Chapter 5

Discussion

In this chapter the outcomes of the project will be discussed. It will include things like design, implementation and evaluation. Possible improvements of the project will be looked over and judgments of how it could have been done differently will be introduced.

5.1 The final test

This section will cover the discussion of the final experiment, from the procedure to the analysis and results. The goal of the experiment was to test the final problem statement by converting this into a null and alternative hypothesis. The results will be discussed below.

Results

The results showed that audio cues did indeed increase the performance of test participants testing the game on the two variables measured - time and path length. However, the sample size was very low of having twenty test participants finish the game with audio cues and only eight participants finishing the game without audio cues. To encourage users to finish the game a nudge of *"Leveraging public commitment"* mentioned in the section 2.3 could have been used. It could have been done by for example a multiplayer option. The user would then know that there is another player playing at the same time, nudging one to fulfill their commitment of finishing the game.

Due to the low number of participants finishing the maze in the smaller sample, eight participants had to be randomly selected from the larger sample. This low sample size highly affected the standard deviation as it was seen in the descriptive statistics. With more samples, it could be expected to see variables fall within a

bell curve which is commonly associated with a well designed experiment. However, through the test procedure valid statistical evidence was found which could support the final problem statement.

The reason for the low amount of samples from people testing the game without audio cues, can be explained as *the ceiling effect*. This effect occurs in experiments when the independent variable no longer affects the dependent variable. The independent variable - audio cues, which was either available or not for the test participants affected the dependent variable - their performance. However, it was observed that few participants were able to solve the game without the audio cues. In the future, to avoid the ceiling effect there should have been an implementation of a more comprehensive pilot testing. That could have shown that the difficulty level of the game without having the audio cues was too high. It could have also been tested on more groups of participants in various difficulties. One with all audio cues available, one with half of them and one without. This might have given data useful for supporting the final problem statement in a different way.

Procedure of the test

In general, there were both - pros and cons of testing online. While the procedure of the test was well planned as mentioned in the 4.3.3 there were some flaws in the way it was done. Due to the Covid-19 pandemic it was not possible to evaluate with test participants being present in the same room as the test conductors. Another flaw was the need for the participants to download the game on their own device. Not all were willing to do that and potential test participants might have decided not to participate in the test for this reason. A solution which was also considered, was to set up an online game. However, a seeming issue with implementing panning online and inconsistency across various internet browsers, it was decided not to implement the game online.

Bias

The method used for getting participants was convenience sampling. This might have introduced some bias, as participants had to be recruited through online means, and those found might not accurately represent the normal population. The test was posted on various social media sites. Many participants may have personal relationships with some of the group members, therefore introducing bias as friends and people with Medialogy related interests played the game. Volunteer bias is another concern that might have had an influence on the results. As the test participants themselves chose to participate. Furthermore, participants might have

had questions about the procedure or other aspects, but since the group was not there in person they could not ask clarifying questions. When bias is introduced into the results, it weakens the accuracy and strength of results.

Questionnaire Feedback

This section will try to identify various trends in the questionnaire related to the final experiment. Two versions of the questionnaire were created - one for group A with audio cues, another for group B, without. These responses might shed more light on the effect of the experiment on the final problem statement.

Test participants in the group A on average found the maze easier to complete while the majority in the group B did not even finish and responded accordingly, noting that it was difficult, see 4.3.3.

5.2 Physical user interface, controller

In general, it was planned to have a custom controller with the requirements that were set in the first iteration in section 2.4.1. The group believed the main concept for the controller clearly had potential as it included the concepts from Gestalt and Don Norman's design principles. However, a usability test could not be performed, to gain feedback on the controller's design. Additionally, components like a gyroscope could have been used to help the user navigate the in-game character by tilting the device, instead of pressing buttons. If a controller was built using the design sketches in figure 4.9, it could have been tested and evaluated upon with the game had Covid-19 not restricted physical contact.

A couple of the Don Norman's design principles reviewed in the section 3.9 were left out when designing the controller. Those were - constraints and feedback. To improve on the design of a controller, these could have been included. Possible considerations for the constraints would be, for instance, buttons locking themselves once one is being pressed. In that way users would not be able to press multiple buttons at the same time.

Concerning the principle of feedback, lights could have been used to indicate if the button is being pressed and the input has been registered. However, this could take away some of the attention of the user. Another solution could be the use of haptic feedback once a button is pressed, the user would feel a slight vibration in the pressed button. This would not draw the eyes of the user to the controller.

5.3 Future perspective

After discussing multiple elements of the project, future work can be looked upon.

5.3.1 Considerations regarding the audio design

It was initially planned to design earcons and use audio cues together with other audio effects implemented in pure data. However, due to the technical difficulties it was decided to use Wwise. In the future, Pure Data could be integrated with Unity and the sounds the group designed themselves could be implemented. This would require further development and understanding of audio design resulting in implementing self made audio for the game. However, Wwise fulfilled most of the functionality required to make this project.

5.3.2 Considerations for expanding the game to other platforms

In the future the game could be implemented for mobile devices. This could potentially change the whole game experience. As mentioned early in the chapter 4, a first person's view might increase the intuitiveness concerning ability to be nudged and guided though out the maze.

A first person camera view was implemented and internally tested, but there were some issues with disorientation. Another solution to ensure a more realistic feeling of solving the maze, could be through virtual reality. In that case, additional equipment besides the earphones would be needed to participate in the game. While the test would then demand a specially prepared test environment and a test being run in present, different findings of such test could potentially be found.

5.3.3 Considerations regarding the maze design

To begin with, there was only one maze design produced during this project. None other were tested as the first and second evaluations indicated a balanced difficulty. In the future a randomized maze design could be implemented to ensure replayability, however, this would make it difficult to minimise the right hand rule.

While the project is focused on audio, the visuals were purposely aimed to be kept simple. If the visuals were to be improved, additional iterations could be done. It would allow for a more interesting design including different layouts, changes in the width of the path, width of the hedges, different materials, colours, locations, etc.

5.3.4 Considerations regarding the User Interface

The UI was kept simple and only the main features were added on the screen. For the future, additional considerations would be as follows. A leader board could be implemented where the users could compare their own time and path length to the others. Perhaps, the game could be played in realtime. It could be displayed as a progression bar which could potentially use the fear nudge *"Reducing the distance"*, from section 2.3 by making it seem like the results are immediate. In this case, the user could see in the progression bar indicating how close they are to the end. That would not benefit the test for the use of audio cues.

If there was a time limit, "Make resources scarce" nudge from Fear nudges, seen in the section 2.3 would have been used. The resource of the time would be presented as scarce and would nudge the player to hurry up and complete the maze as soon as possible. Or as mentioned in one of the questionnaire responses, there could be some sort of enemy that chases the player through the maze.

By considering these factors, the users could potentially be nudged and thereby improve their personal times.

5.3.5 Final considerations

Testing could be done to evaluate on the ceiling effect, adding more or taking some elements away from the UI, randomising the maze, improving game graphics, sound quality and its diversity, designing earcons and a physical interface. The physical interface could include additional sensors to what is currently designed, which would change the game play and the overall experience of playing the game. This would require further testing in the future and the collection of data from more people to find out if these changes have a positive impact.

Chapter 6

Conclusion

An initial problem statement was formed before starting to analyse the relevant topics. It was as follows: *"How can audio cues be used to improve the performance of a player navigating through a labyrinth or maze game?"*.

The analysis was used to gain the necessary knowledge for proposing design requirements and forming the final problem statement. To start off the analysis, labyrinths and mazes were looked and it was found that mazes had the properties that were needed for the testing purposes. Then performance was looked into. It was concluded that the time taken and path length are the variables which would later be used in the testing. In the section of audio based state of the art games, three games were reviewed. It was found that tools used in these games were useful for this project. These were - binaural sound, panning and minimising the visual distractions. In the audio processing sections various audio techniques were explored which could potentially be used to manipulate sound effects in the game. The knowledge was further used to design and implement some of the audio cues and other audio effects in the game. Furthermore, nudging was looked into and it was found that multiple mechanisms of nudges were relevant for the project work. In the physical interface design section an analysis of various game controllers was looked at. Interaction with a personal computer was decided to be done via a game controller designed specially for this project. Knowledge of micro-controllers and electric circuits was gained in order to be able to build a physical interface.

After this analysis a final problem statement was formed and it reads as follows: *"How can audio processing effects be used to make audio cues that assists a player in finding the right path and decrease time taken to complete a digital 3D maze?"*

In order to reach the goal of the project and answer the final problem statement, the relevant methods to do so were described in methods chapter.

Three iterations were made in this project. Each of the iterations included a phase of design, implementation and evaluation. After the first initial test, it was discovered that the audio cues functioned as intended but they were not particularly pleasant to listen to for the test participants. Furthermore, the user interface had to be updated to include more critical information about the game. This was then implemented in the second iteration. In the second, usability test, the outcome was that the participants found the audio cues to be intuitive and the game mechanics to be easy to understand. The bugs regarding the audio was fixed, along with an update for the user interface for the next iteration, before the final A/B test. In the final test a statistical analysis, the final problem statement was written as a null and alternative hypotheses in order to accept or reject these.

The final problem statement for the project was: *"How can audio processing effects be used to make audio cues that assist a player navigating a digital 3D maze, thereby decreasing the path length and time taken to complete it?"*

Based on the results of the final experiment, mainly the Welch T-test, which tested whether or not there was a significant difference in the performance of test participants who tested with or without audio. The test concluded that there was indeed a significant difference and that audio cues increased the performance of those who had them available. Furthermore, most of the design requirements, except for those concerning the physical interface, were met, which means that the final problem statement has been answered to a satisfying extent.

Chapter 7

Appendices

In this section the appendixes of the tests will be presented. The sections will include the full responses from the participants. To see the google forms, type in the URL on to your browser manually or copy it from the text file. The google forms link will be presented at the top of each appendix.

7.1 Appendix 1

This is the link for the google forms: https://docs.google.com/forms/d/1bKXsDZ4A1zzMYm4QHIuPFIbpNT4X6u1sx6lj_PDfCLk/edit?usp=sharing

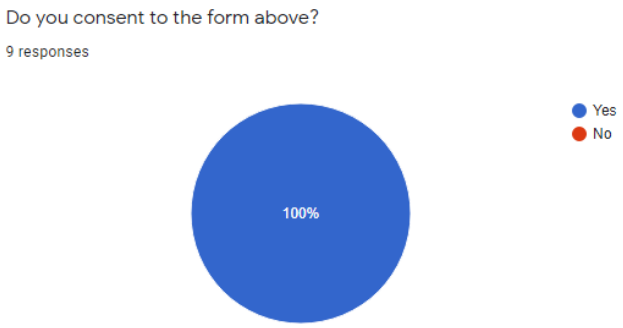


Figure 7.1: Question 1: Consent form

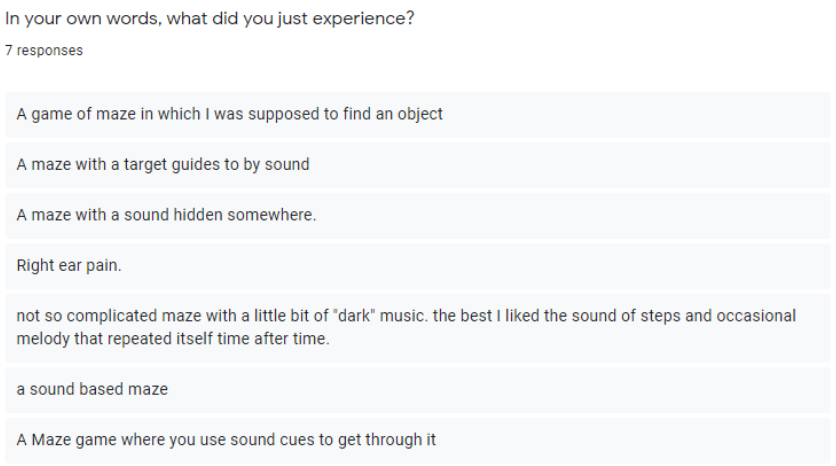


Figure 7.2: Question 2

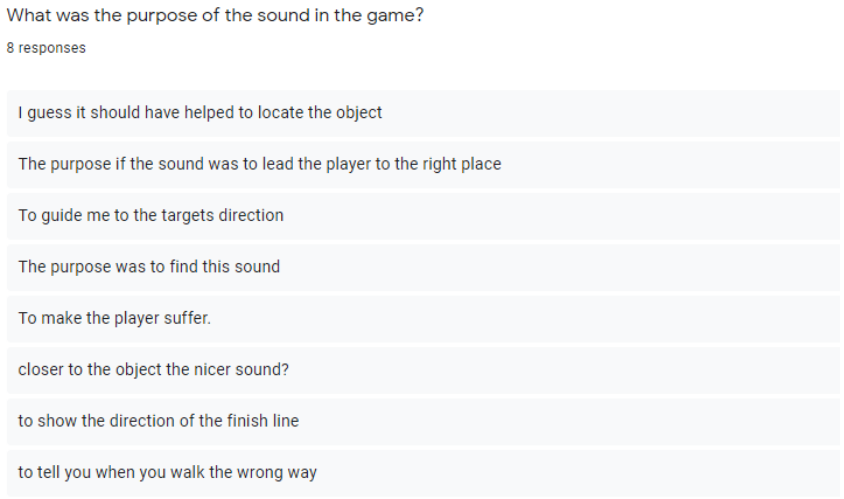


Figure 7.3: Question 3

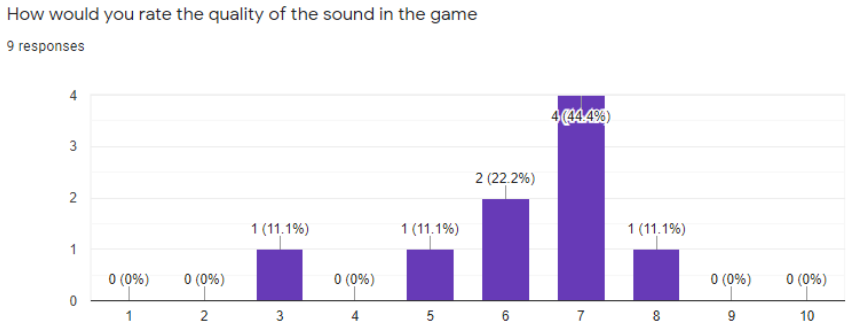


Figure 7.4: Question 4

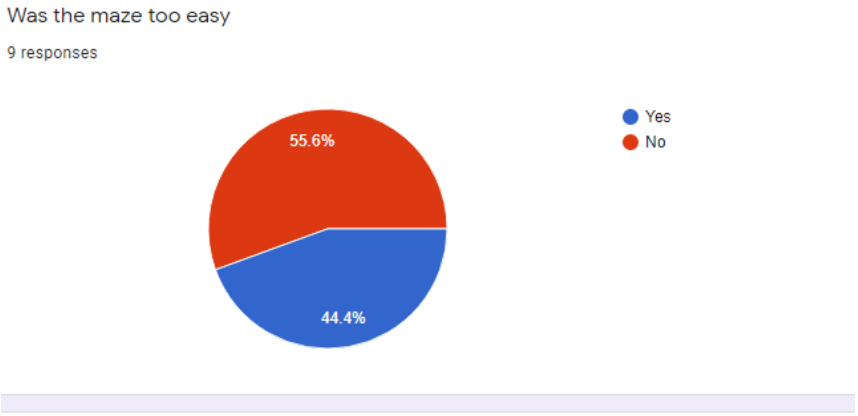


Figure 7.5: Question 5

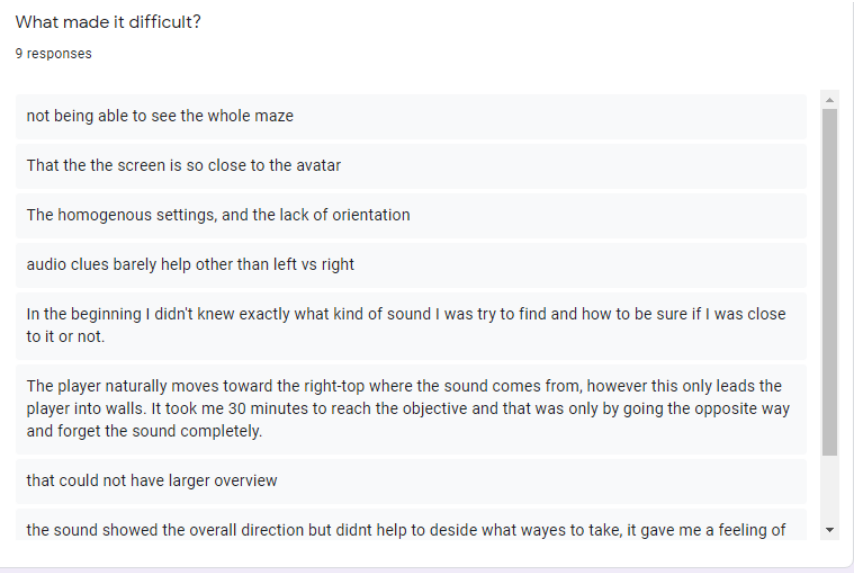


Figure 7.6: Question 6

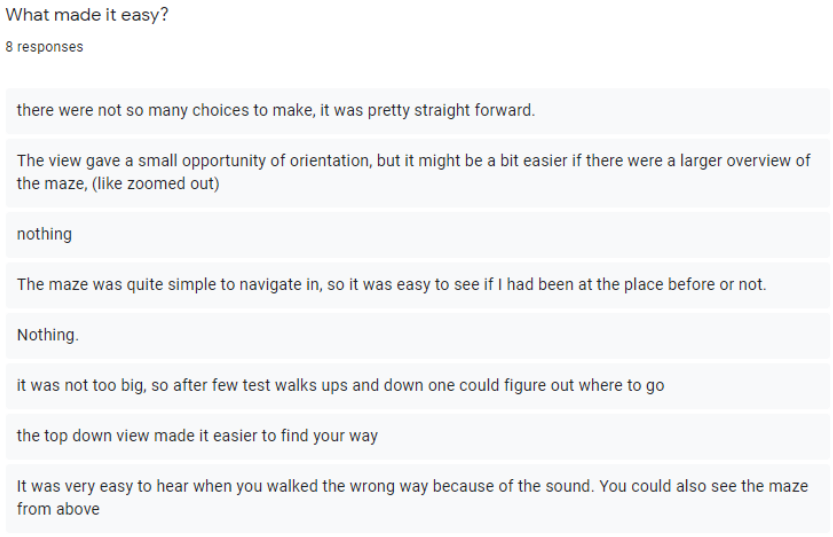


Figure 7.7: Question 7

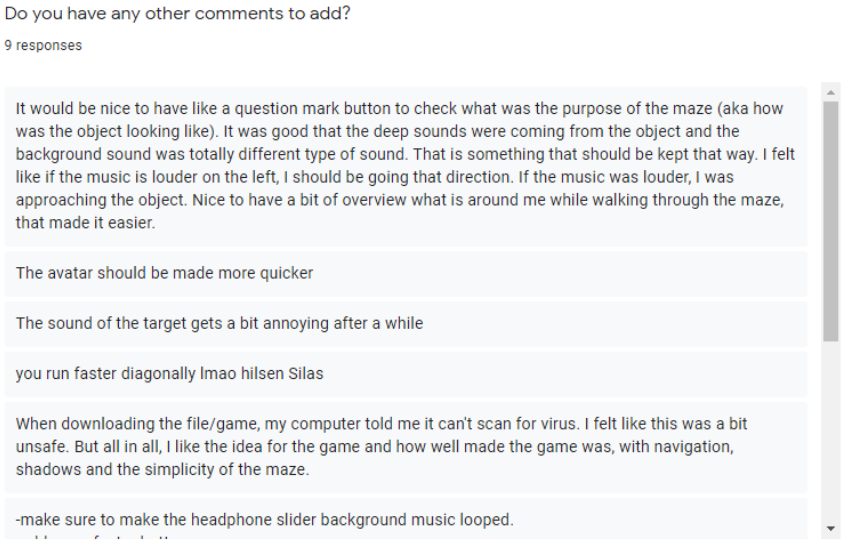


Figure 7.8: Question 8

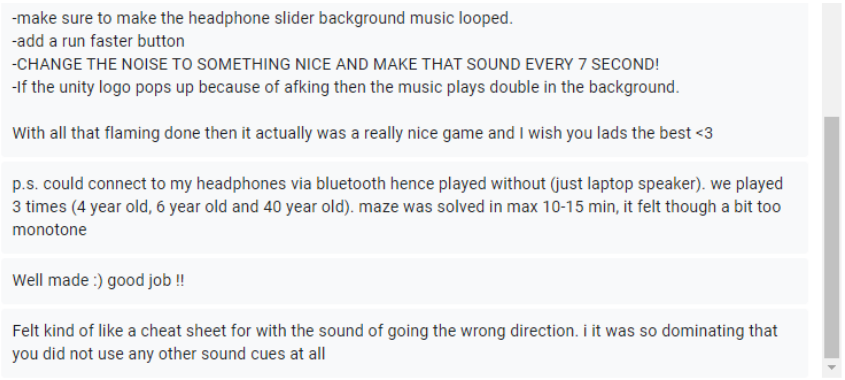


Figure 7.9: Question 8

7.2 Appendix 2

This is the link for the second test on google forms: https://docs.google.com/forms/d/1hdJ1s2ZTtzyGSNYY0tkZAqVVZr-ackGhsA5dORgQlbA/edit?usp=sharing

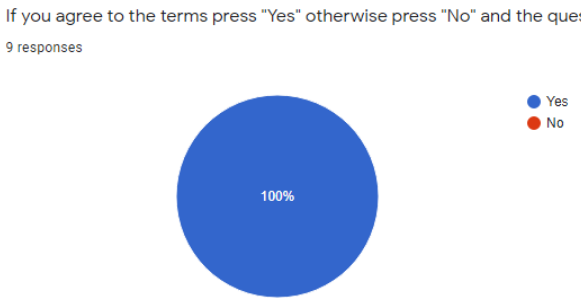


Figure 7.10: Question 1

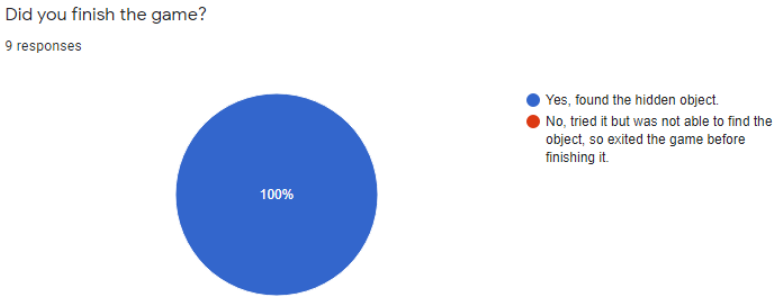


Figure 7.11: Question 2

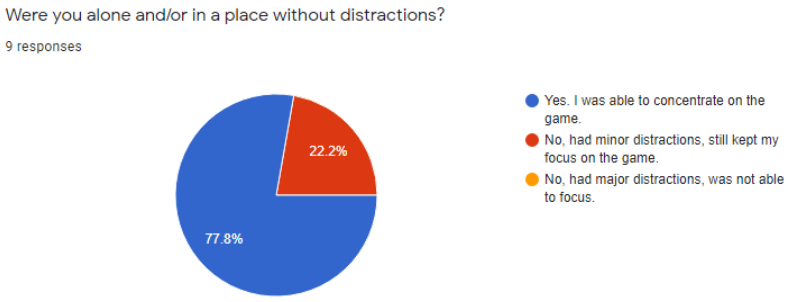


Figure 7.12: Question 3

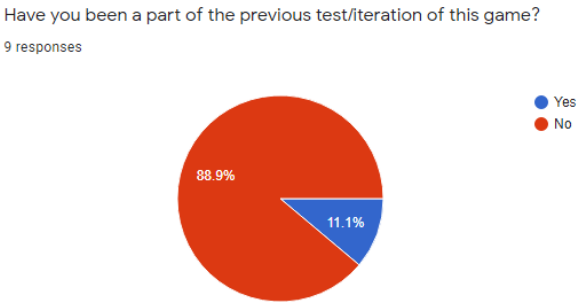


Figure 7.13: Question 4

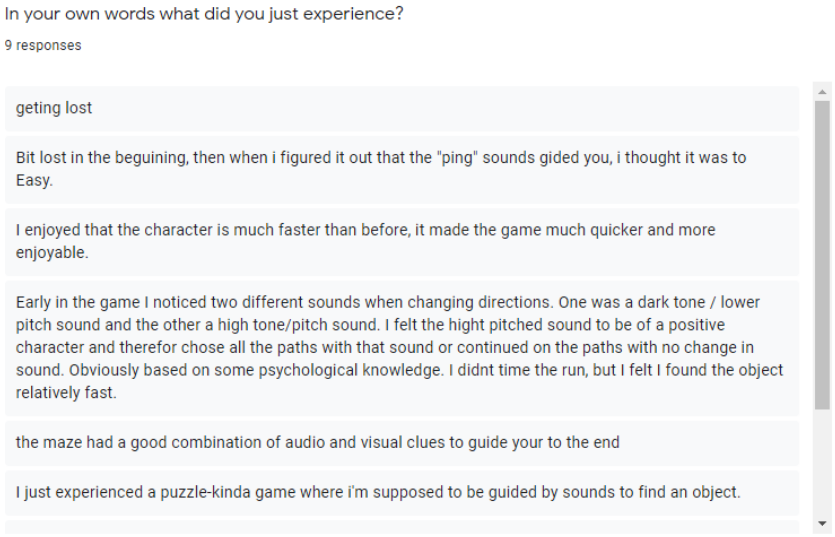


Figure 7.14: Question 5

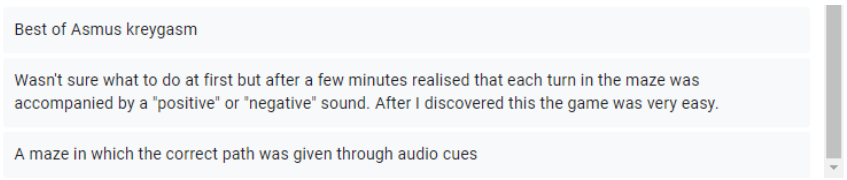


Figure 7.15: Question 5

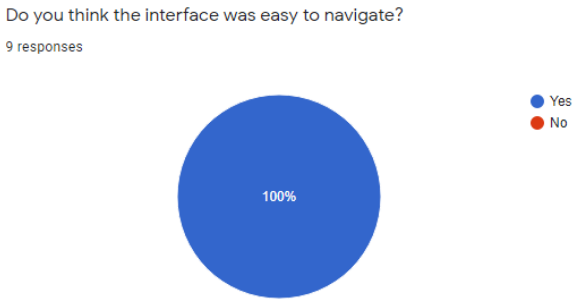


Figure 7.16: Question 6

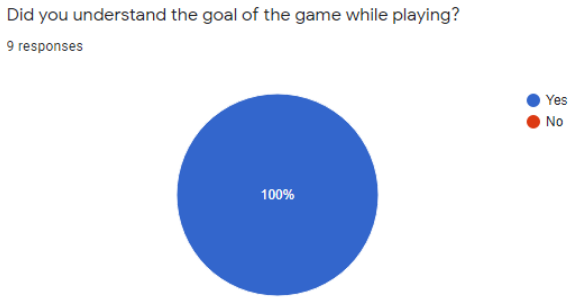


Figure 7.17: Question 7

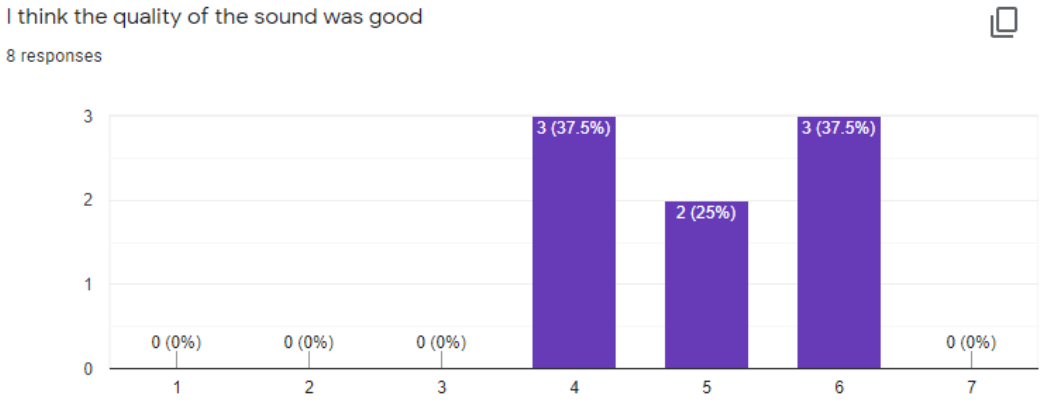


Figure 7.18: Question 8

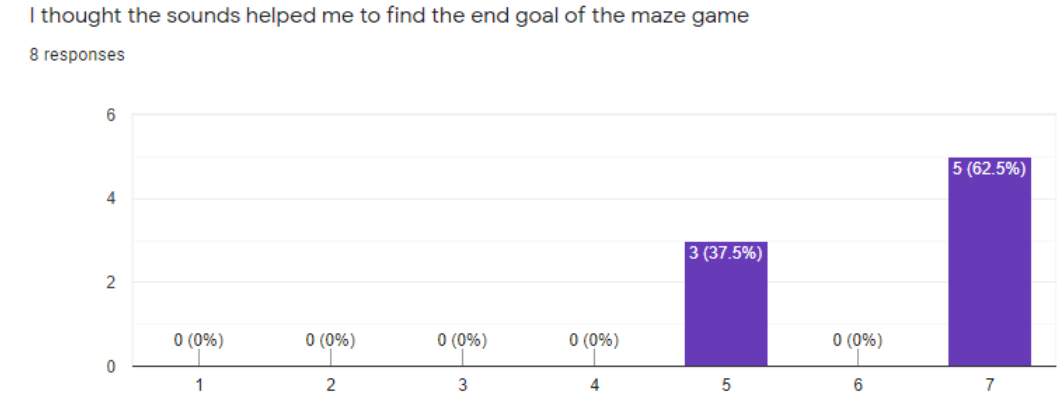


Figure 7.19: Question 9

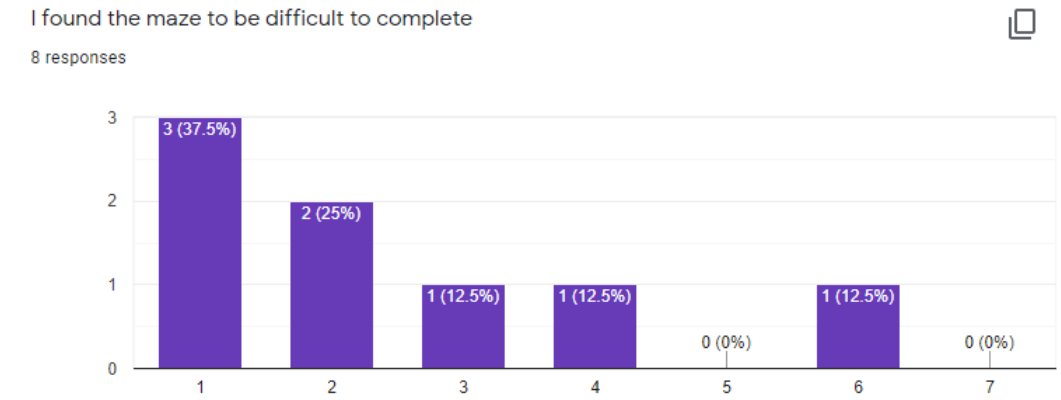


Figure 7.20: Question 10

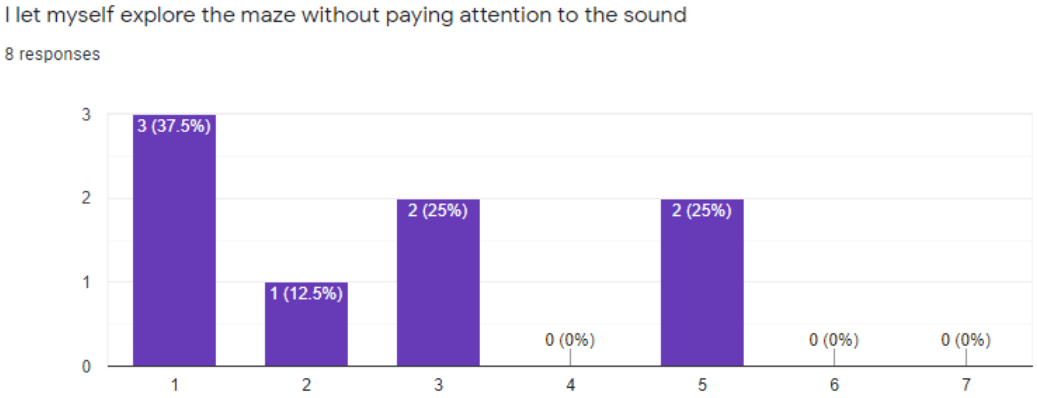


Figure 7.21: Question 11

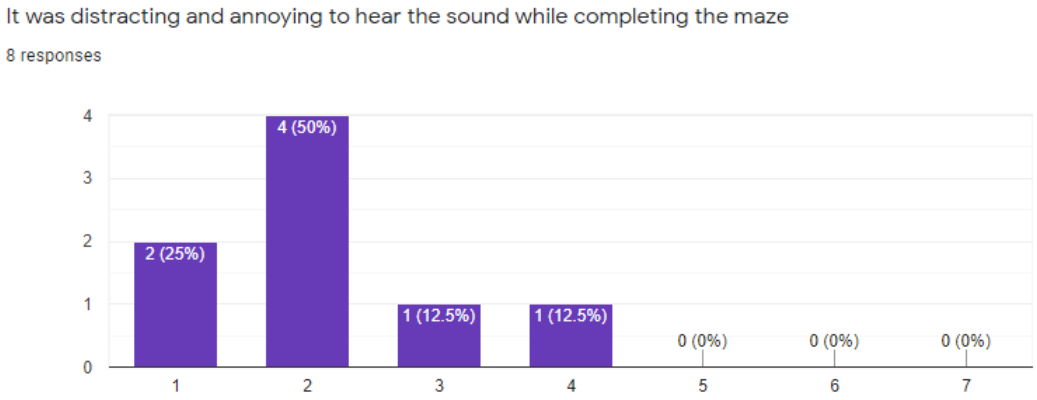


Figure 7.22: Question 12

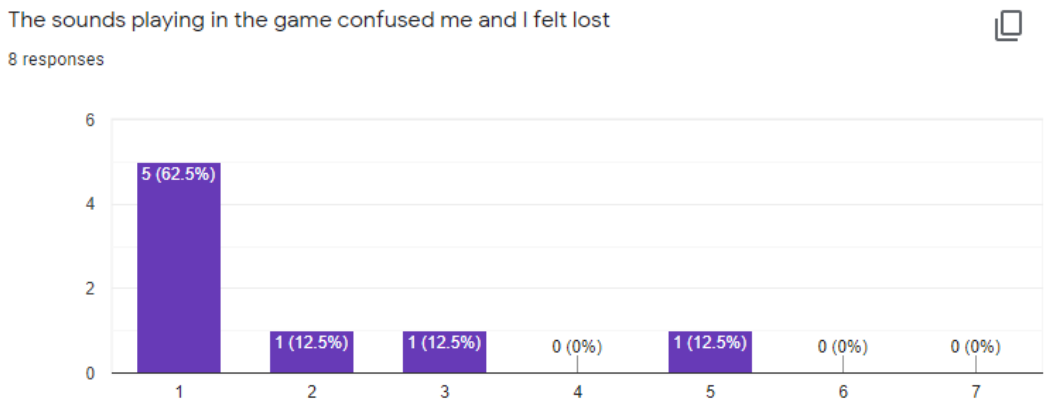


Figure 7.23: Question 13

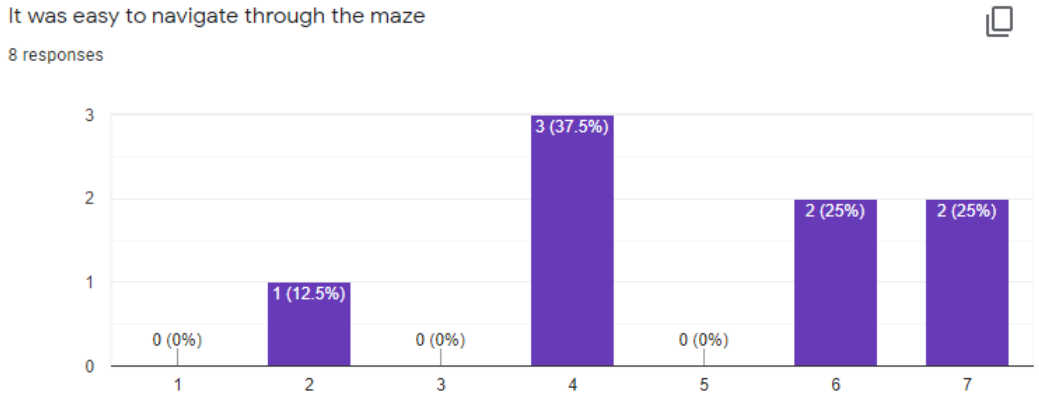


Figure 7.24: Question 14

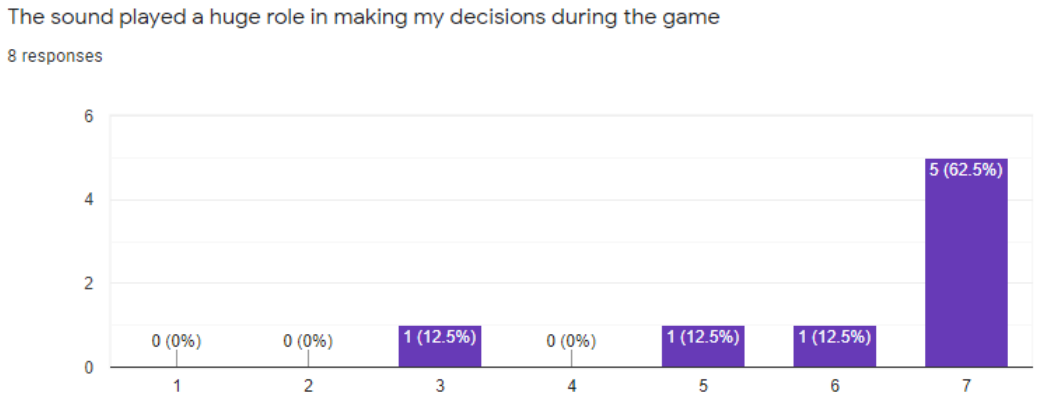


Figure 7.25: Question 15



Figure 7.26: Question 16

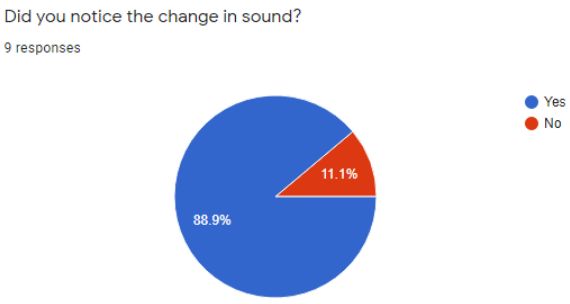


Figure 7.27: Question 17

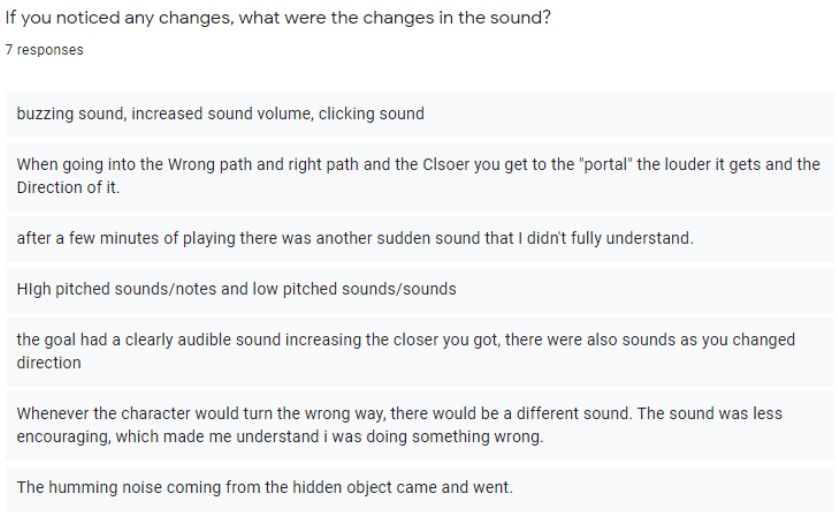


Figure 7.28: Question 18

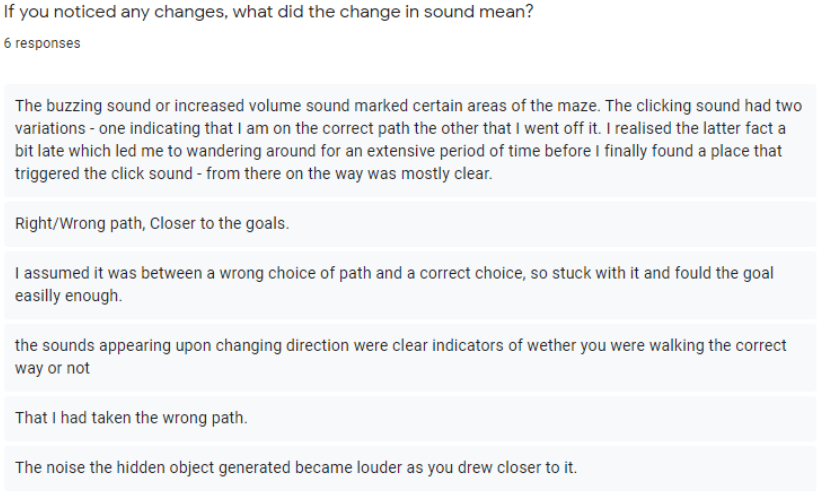


Figure 7.29: Question 19

7.3 Appendix 3

Link to the questionnaire with sound: https://docs.google.com/forms/d/1cemHPk6aiIhTOrLtdQoyG5wchE3bOu_Tu14_ODBOXtg/edit?usp=sharing

Link to the questionnaire without sound: https://docs.google.com/forms/d/1vAjl3PnsnIyQRyxdl0GEnwJfHz2XXGjuxK9qOIndWv8/edit?usp=sharing

Graphs of the responses for the test with sound

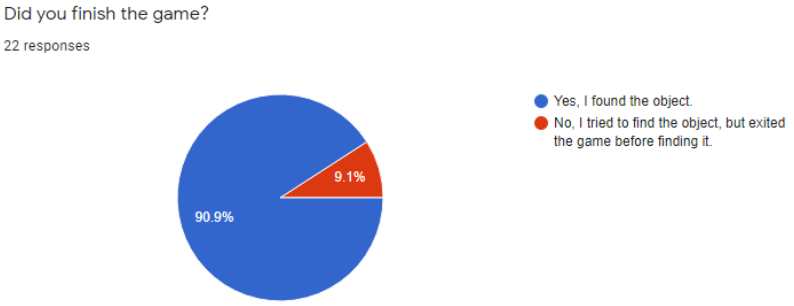


Figure 7.30: Question: Did you finish the game?

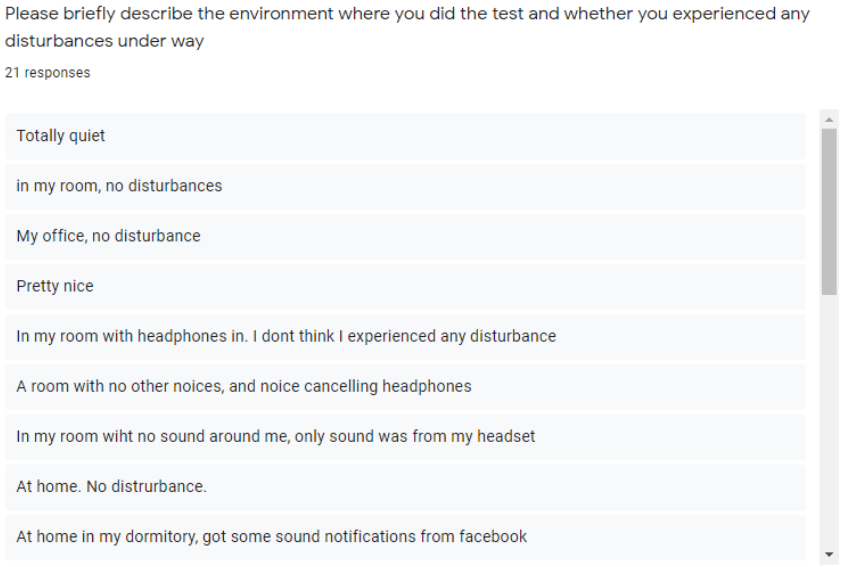


Figure 7.31: Question: Please briefly describe the environment where you did the test and whether you experienced any disturbances under way

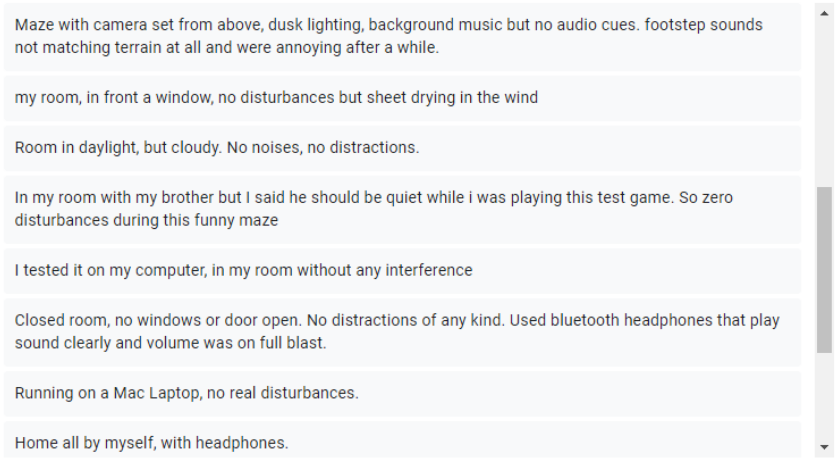


Figure 7.32: Question: Please briefly describe the environment where you did the test and whether you experienced any disturbances under way

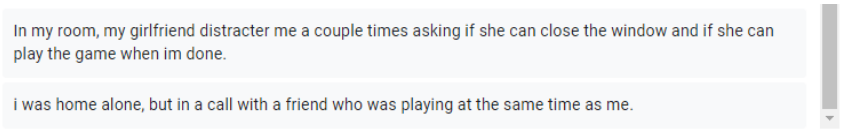


Figure 7.33: Question: Please briefly describe the environment where you did the test and whether you experienced any disturbances under way

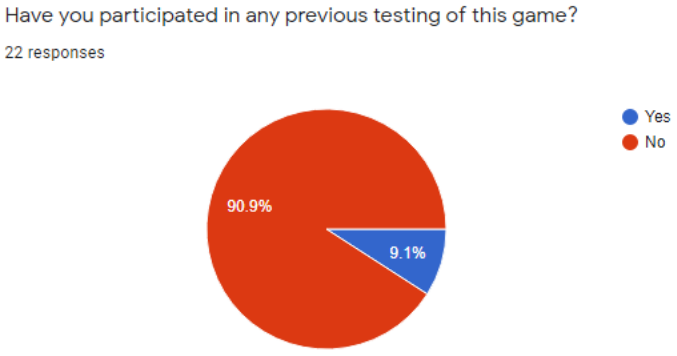


Figure 7.34: Question: Have you participated in any previous testing of this game?

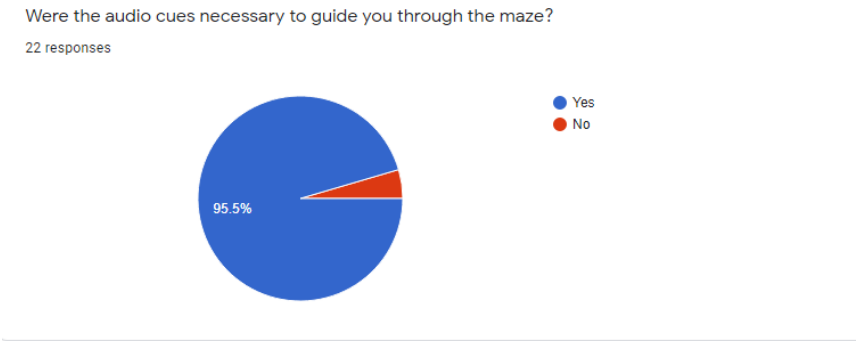


Figure 7.35: Question: It was intuitive which direction I should go to complete the maze

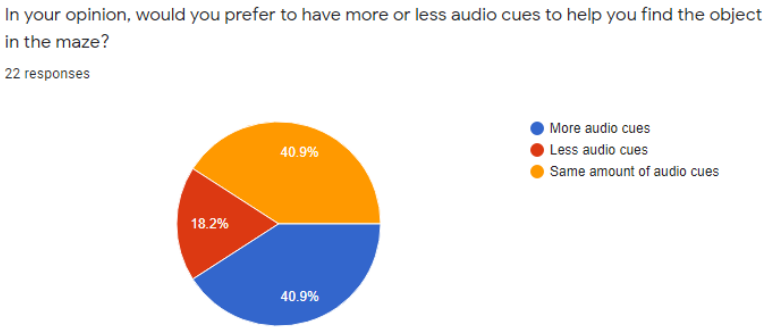


Figure 7.36: Question: In your opinion, would you prefer to have more or less audio cues to help you find the object in the maze?

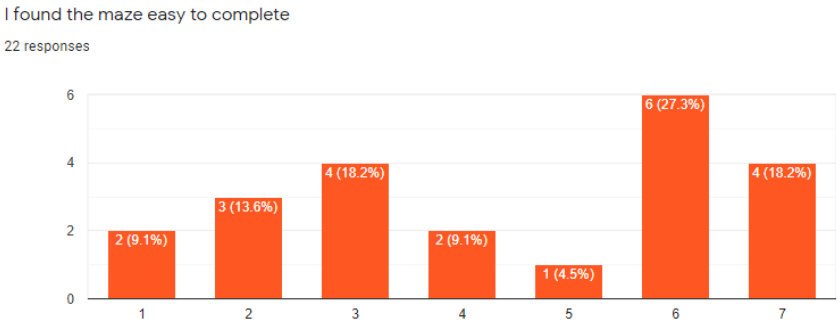


Figure 7.37: Question: I found the maze easy to complete

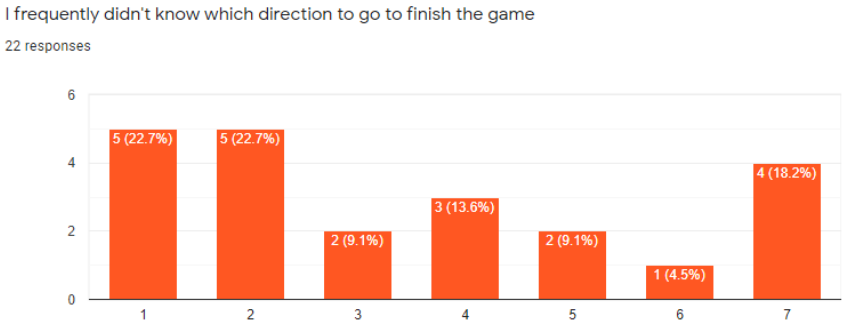


Figure 7.38: Question: I frequently didn't know which direction to go to finish the game

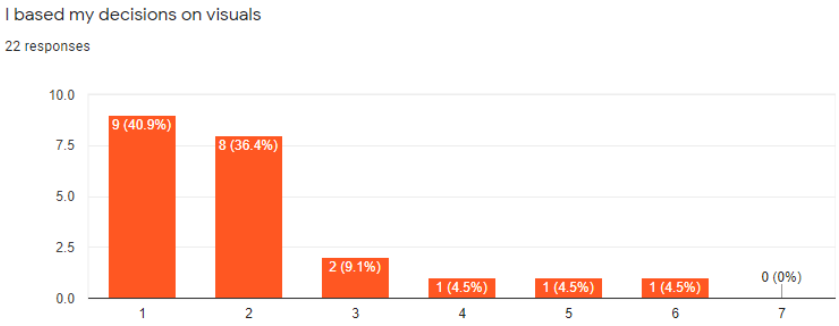


Figure 7.39: Question: I based my decisions on visuals

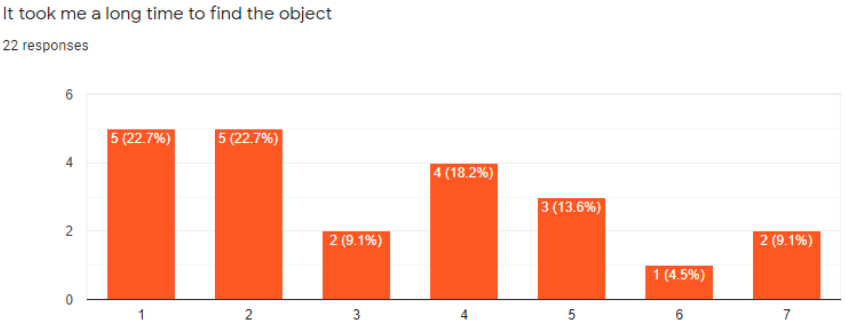


Figure 7.40: Question: It took me a long time to find the object

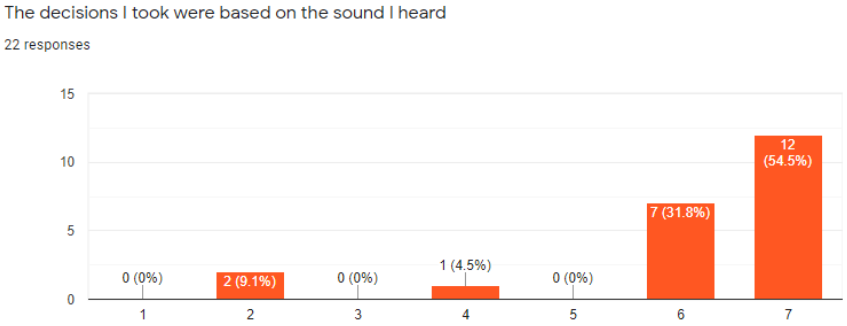


Figure 7.41: Question: The decisions I took were based on the sound I heard

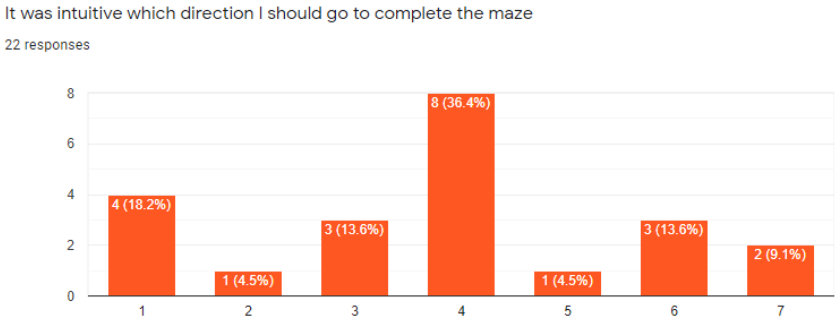


Figure 7.42: Question: It was intuitive which direction I should go to complete the maze

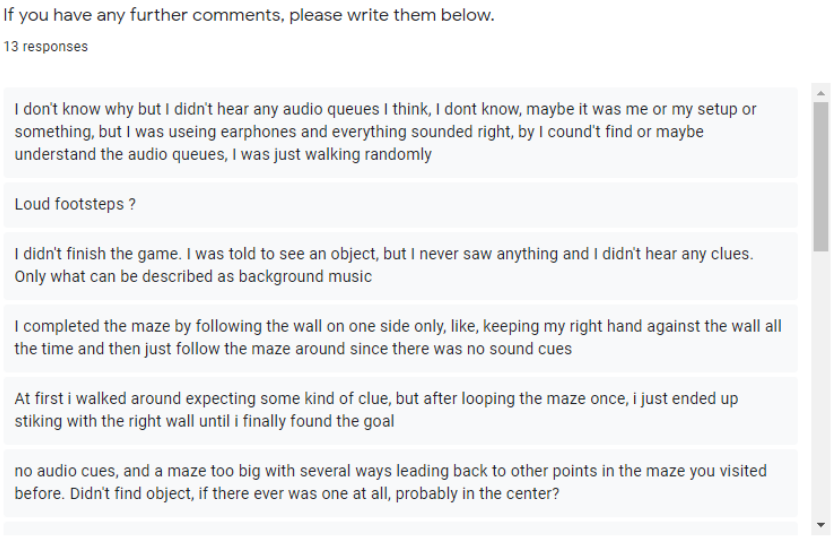


Figure 7.43: Question: If you have any further comments, please write them below.

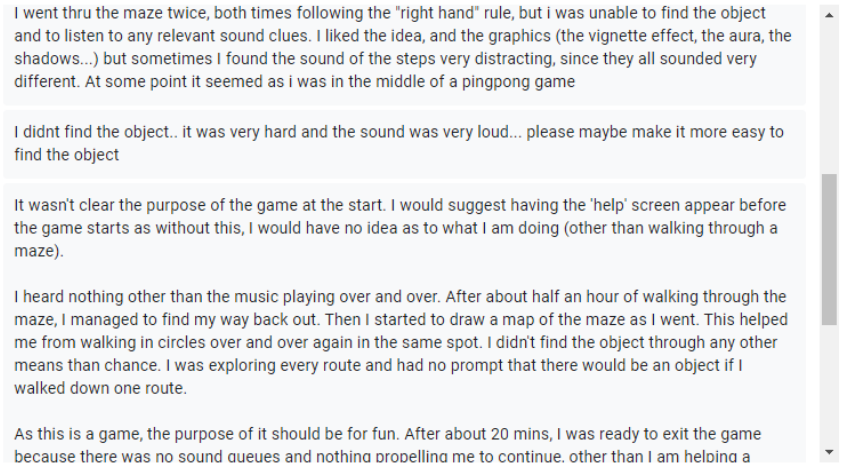


Figure 7.44: Question: If you have any further comments, please write them below.

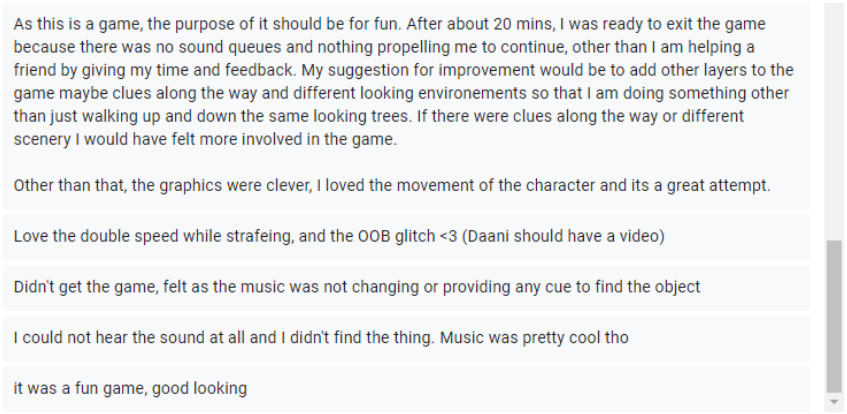


Figure 7.45: Question: If you have any further comments, please write them below.

Graphs of the responses for the test without sound

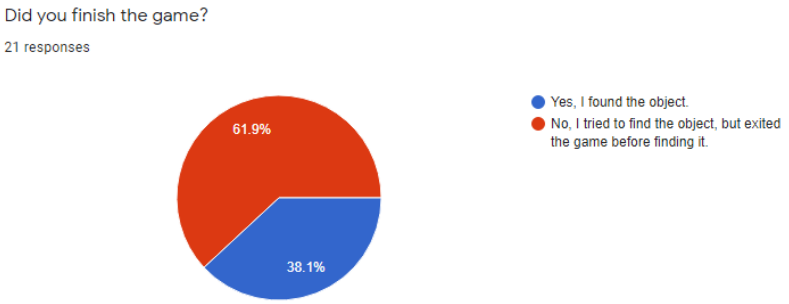


Figure 7.46: Question: Did you finish the game?

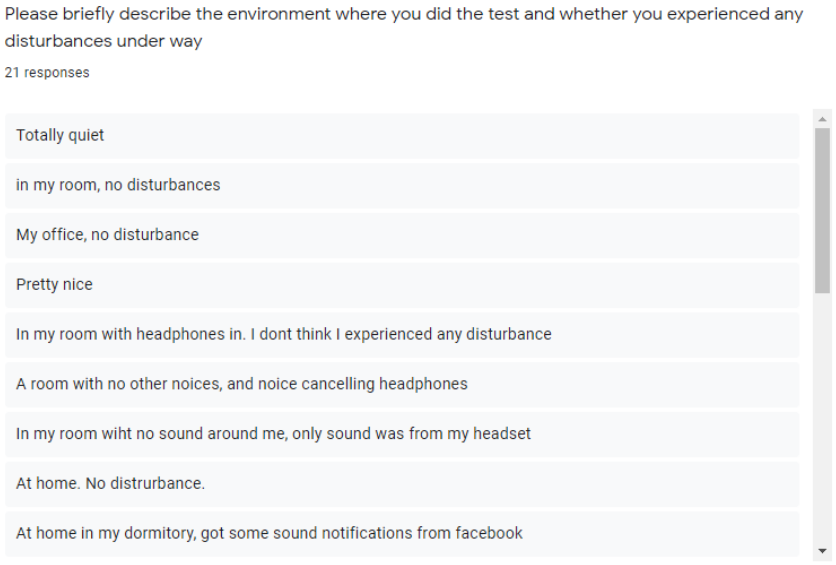


Figure 7.47: Question: Please briefly describe the environment where you did the test and whether you experienced any disturbances under way

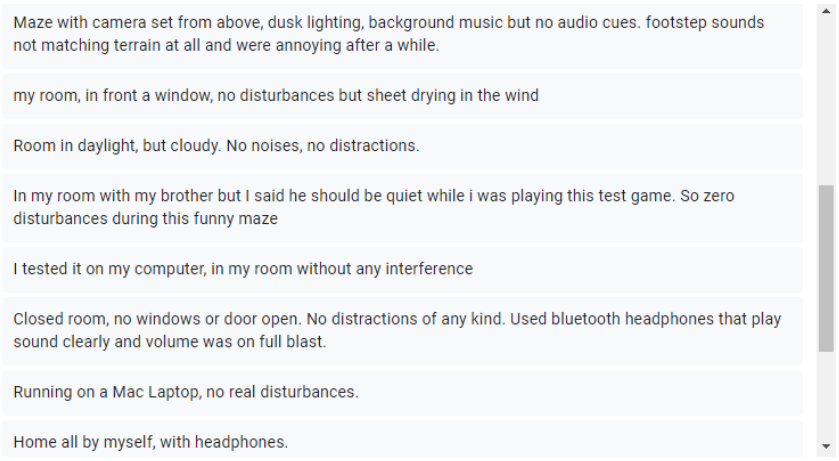


Figure 7.48: Question: Please briefly describe the environment where you did the test and whether you experienced any disturbances under way

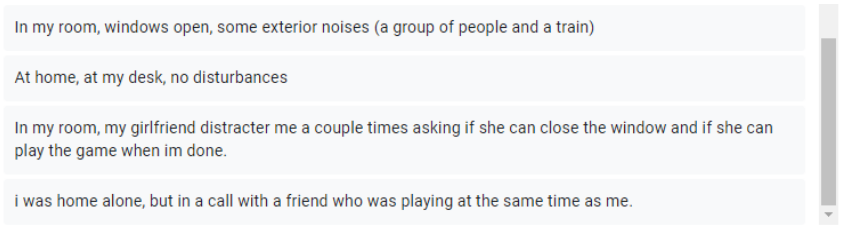


Figure 7.49: Question: Please briefly describe the environment where you did the test and whether you experienced any disturbances under way

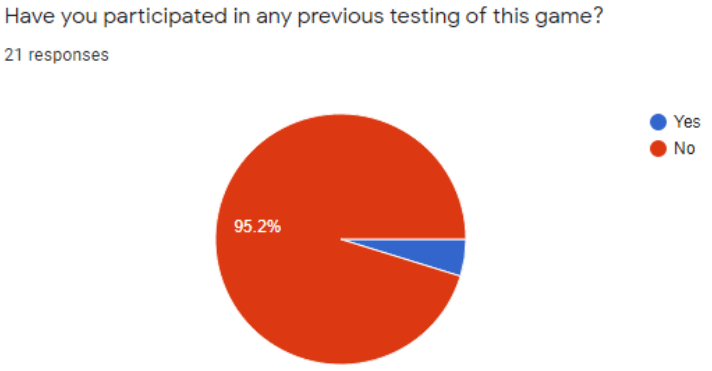


Figure 7.50: Question: Have you participated in any previous testing of this game?

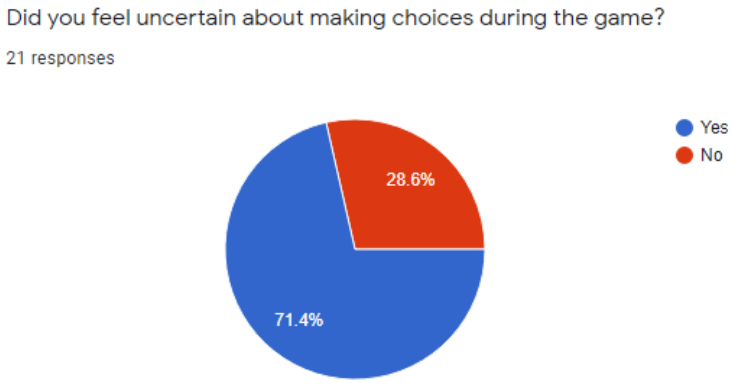


Figure 7.51: Question: Did you feel uncertain about making choices during the game?

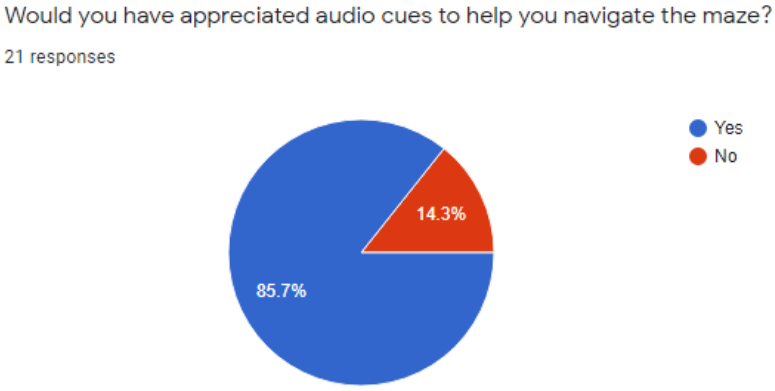


Figure 7.52: Question: Would you have appreciated audio cues to help you navigate the maze?

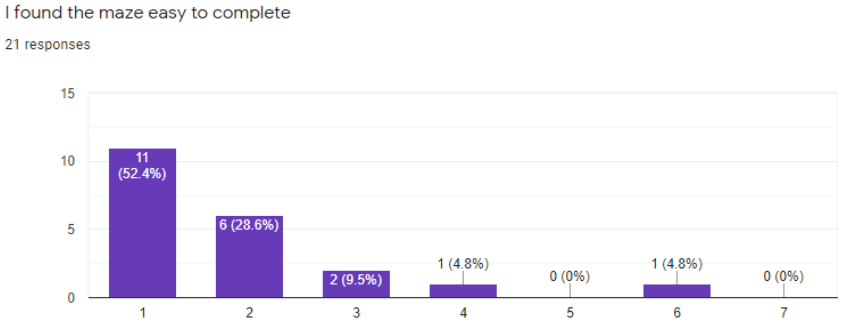


Figure 7.53: Question: I found the maze easy to complete

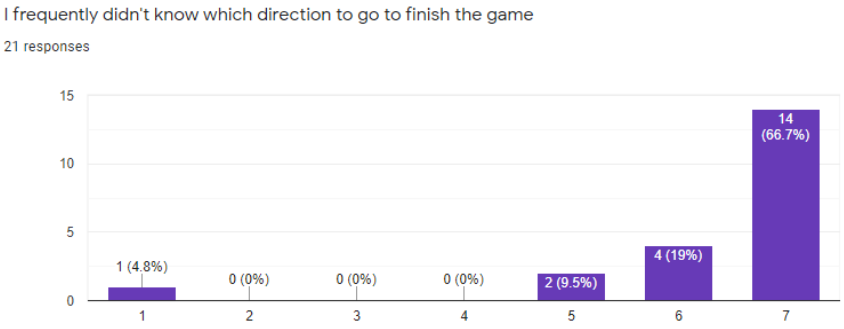


Figure 7.54: Question: I frequently didn't know which direction to go to finish the game

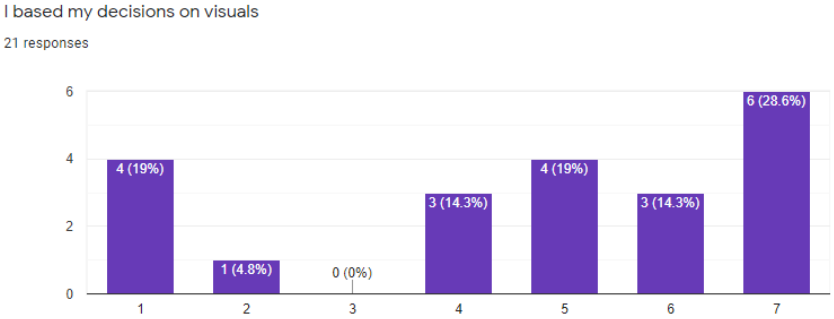


Figure 7.55: Question: I based my decisions on visuals

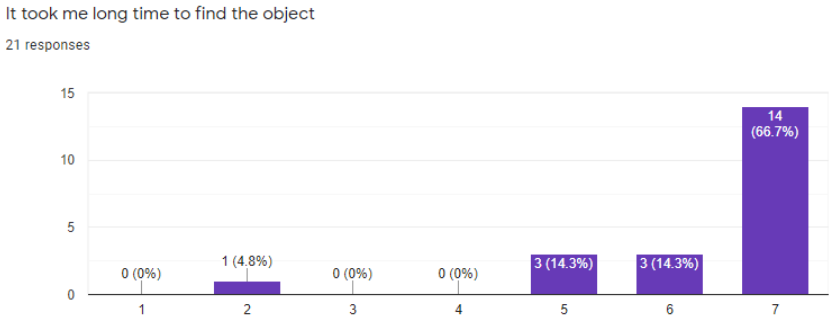


Figure 7.56: Question: It took me long time to find the object

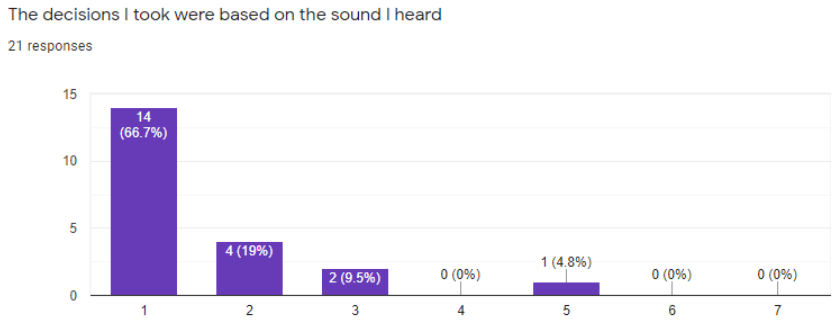


Figure 7.57: Question: The decisions I took were based on the sound I heard

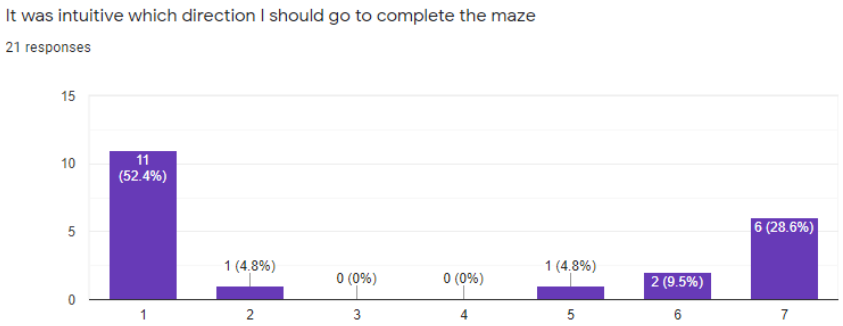


Figure 7.58: Question: It was intuitive which direction I should go to complete the maze

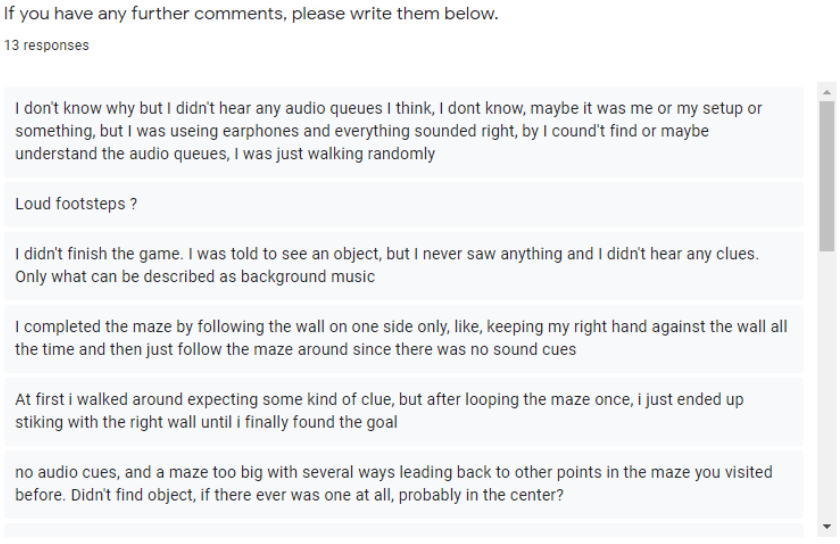


Figure 7.59: Question: If you have any further comments, please write them below.

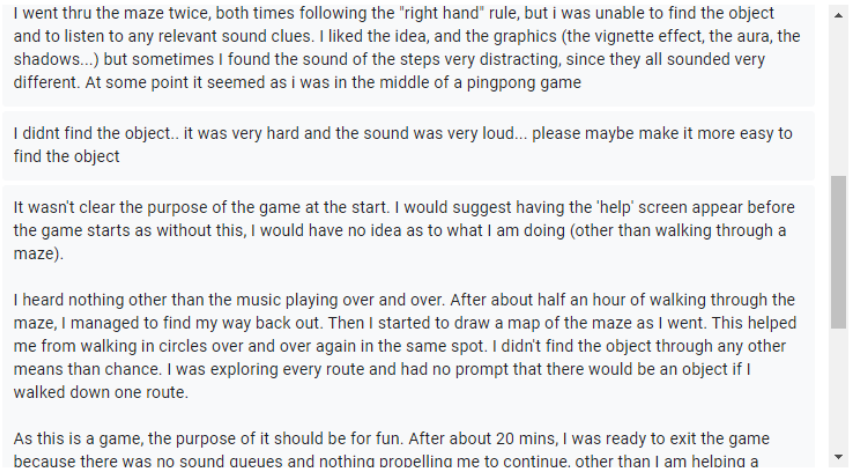


Figure 7.60: Question: If you have any further comments, please write them below.



Figure 7.61: Question: If you have any further comments, please write them below.

Bibliography

[1] *7 Gestalt principles of visual perception: cognitive psychology for UX*. https://www.usertesting.com/blog/gestalt-principles.

[2] *ALL-PASS FILTERS*. URL: http://sepwww.stanford.edu/sep/prof/fgdp/c2/paper_html/node7.html. (accessed: 08.03.2020).

[3] Asbjoern Anderson. *Why Procedural Game Sound Design is so useful*. 2016. URL: https://www.asoundeffect.com/procedural-game-sound-design/.

[4] *Arduino Uno Rev3 | Arduino Official Store*. URL: https://store.arduino.cc/arduino-uno-rev3. (accessed: 25.05.2020).

[5] *Attenuation of Sound Waves*. 2020. URL: https://www.nde-ed.org/EducationResources/CommunityCollege/Ultrasonics/Physics/attenuation.htm.

[6] Bertrof. *Game Sound Correct*. 2020. URL: https://freesound.org/people/Bertrof/sounds/131660/.

[7] Bertrof. *Game Sound Wrong*. 2020. URL: https://freesound.org/people/Bertrof/sounds/131657/.

[8] Dr Christina Beste. *SOUND LABYRINTH – NAVIGATING WITH NOISES*. https://www.audioversum.at. 2020.

[9] Meera M Blattner, Denise A Sumikawa, and Robert M Greenberg. "Earcons and icons: Their structure and common design principles". In: *Human–Computer Interaction* 4.1 (1989), pp. 11–44.

[10] Stephen A Brewster, Peter C Wright, and Alistair DN Edwards. "Experimentally derived guidelines for the creation of earcons". In: *Adjunct Proceedings of HCI*. Vol. 95. 1995, pp. 155–159.

[11] Ana Caraban et al. "23 ways to nudge: A review of technology-mediated nudging in human-computer interaction". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pp. 1–15.

[12] Mads G Christensen. *Introduction to Audio Processing*. Springer, 2019.

[13] Mads G. Christensen. *Introduction to Audio Processing*. Springer International Publishing, 2019. DOI: 10.1007/978-3-030-11781-8. URL: https://doi.org/10.1007\%2F978-3-030-11781-8.

[14] Mads G. Christensen. *Introduction to Audio Processing*. Springer Nature Switzerland, 2019.

[15] Robert B Cialdini. *Influence*. Vol. 3. A. Michel Port Harcourt, 1987.

[16] Alastair H Cummings. "The evolution of game controllers and control schemes and their effect on their games". In: *The 17th annual university of southampton multimedia systems conference*. Vol. 21. 2007.

[17] DOWiNO. *Game "A blind legend"*. http://www.ablindlegend.com/en/about-us/. 2016.

[18] Johan Egebark and Mathias Ekström. "Status quo bias in decision making". In: *Journal of risk and uncertainty* 1 (1988), pp. 7–59.

[19] *Exploring the Gestalt Principles of Design*. URL: https://www.toptal.com/designers/ui/gestalt-principles-of-design. (accessed: 26.05.2020).

[20] *Feedforward and feedback comb filters*. URL: http://www.sengpielaudio.com/calculator-combfilter.htm. (accessed: 25.03.2020).

[21] Clara Fernandez-Vara. "Labyrinth and maze". In: *Space Time Play* (2007), pp. 74–77.

[22] Andy P Field and Graham Hole. *How to design and report experiments*. Sage, 2012.

[23] Adrian Fisher. *Adrian Fisher, Maze designer*. https://www.mazemaker.com/about. 2020.

[24] Ryan Goodrich. *Accelerometer vs. Gyroscope: What's the Difference?* https://www.livescience.com/40103\protect\discretionary{\char\hyphenchar\font}{}{}accelerometer-vs-gyroscope.html. 2018.

[25] Ryan Goodrich. *Accelerometers: What They Are & How They Work*. https://www.livescience.com/40102-accelerometers.html. 2013.

[26] C. J. Goodwin. *A-mazing research. Monitor on Psychology*. http://www.apa.org/monitor/2012/02/research. 2012.

[27] *HoloLens 2*. URL: https://www.microsoft.com/da-dk/hololens/hardware. (accessed: 27.05.2020).

[28] Audiokinetic Inc. *Applying Distance-Based Attenuation*. 2020. URL: https://www.audiokinetic.com/library/edge/?source=Help&id=applying_distance_based_attenuation.

[29] Hermann Kern and Jeff Saward. *Through the labyrinth: Designs and meanings over 5,000 years*. Prestel New York, 2000.

[30] *Kirchhoffs Circuit Law*. URL: https://www.electronics-tutorials.ws/dccircuits/dcp_4.html.

[31] *Kirchhoff's Current Law*. URL: https://www.allaboutcircuits.com/textbook/direct-current/chpt-6/kirchhoffs-current-law-kcl/. (accessed: 25.05.2020).

[32] *Kirchhoff's Voltage Law*. URL: https://www.allaboutcircuits.com/textbook/direct-current/chpt-6/kirchhoffs-voltage-law-kvl/. (accessed: 25.05.2020).

[33] Andrew J Kolarik et al. "Auditory distance perception in humans: a review of cues, development, neuronal bases, and effects of sensory loss". In: *Attention, Perception, & Psychophysics* 78.2 (2016), pp. 373–395.

[34] Samantha Lile. *How to Apply Gestalt Principles to Your Designs for Maximum Impact*. https://visme.co/blog/gestalt-design-principles/.

[35] Kevin Macleod. *Leaving Home*. 2020. URL: https://incompetech.filmmusic.io/song/4708-leaving-home.

[36] *Mazes*. URL: http://datagenetics.com/blog/november22015/index.html. (accessed: 08.03.2020).

[37] *Mazes*. URL: https://tinyurl.com/y9cr5722. (accessed: 12.03.2020).

[38] *Mazes*. URL: https://en.wikipedia.org/wiki/Maze_solving_algorithm. (accessed: 12.03.2020).

[39] Microsoft. *SPECIFICATIONS*. https://images-na.ssl-images-amazon.com/images/I/91ky6OwQu7L.pdf. 2013.

[40] Nintendo. *Technical Specs*. https://www.nintendo.com/switch/tech-specs/. 2017.

[41] Don Norman. *The design of everyday things: Revised and expanded edition*. Basic books, 2013.

[42] *Ohm's Law*. May 2020. URL: https://en.wikipedia.org/wiki/Ohm's_law.

[43] Putu Adi Guna Permana. "Scrum method implementation in a software development project management". In: *International Journal of Advanced Computer Science and Applications* 6.9 (2015), pp. 198–204.

[44] Jennifer Preece, Yvonne Rogers, and Helen Sharp. *Interaction Design*. 4th ed. Wiley, 2015.

[45] Graeme D. Ruxton. "The unequal variance t-test is an underused alternative to Student's t-test and the Mann–Whitney U test". In: *Behavioral Ecology* 17.4 (May 2006), pp. 688–690.

[46] Thurman C. Scott. "Time Records in Human Maze Learning and their Comparison with Maze Records". In: *The Journal of Genetic Psychology* 87.2 (1955), pp. 265–275. DOI: 10.1080/00221325.1955.10532939. eprint: https://doi.org/10.1080/00221325.1955.10532939. URL: https://doi.org/10.1080/00221325.1955.10532939.

[47] Eberhard Sengpiel. *Sound and Distance*. 2020. URL: http://www.sengpielaudio.com/calculator-SoundAndDistance.htm.

[48] *Shepard tone*. URL: https://en.wikipedia.org/wiki/Shepard_tone. (accessed: 08.03.2020).

[49] Herbert A Simon and Allen Newell. "Human problem solving: The state of the theory in 1970." In: *American Psychologist* 26.2 (1971), p. 145.

[50] Sony. *SPECIFICATIONS*. https://images-na.ssl-images-amazon.com/images/I/71hdsQdUcfL.pdf. 2013.

[51] Barry M Staw. "The escalation of commitment to a course of action". In: *Academy of management Review* 6.4 (1981), pp. 577–587.

[52] Richard H. Thaler and Cass R. Sunstein. *NUDGE: Improving Decisions About Health, Wealth, and Happiness*. 2. ed. Yale University Press New Haven & London, 2008.

[53] *The sound illusion that makes Dunkirk so intense*. URL: https://www.youtube.com/watch?v=LVWTQcZbLgY. (accessed: 08.03.2020).

[54] Yaacov Trope and Nira Liberman. "Construal-level theory of psychological distance." In: *Psychological review* 117.2 (2010), p. 440.

[55] Amos Tversky and Daniel Kahneman. "Judgment under uncertainty: Heuristics and biases". In: *science* 185.4157 (1974), pp. 1124–1131.

[56] O. Veneri, CEDRIC Stéphane Gros, and CEDRIC Stéphane Natkin. "Procedural Audio for Game using GAF". In: 2008.

[57] Yingxu Wang and Vincent Chiew. "On the cognitive process of human problem solving". In: *Cognitive Systems Research* 11 (Mar. 2010), pp. 81–92. DOI: 10.1016/j.cogsys.2008.08.003.

[58] *What Are Electric Circuits?* URL: https://www.allaboutcircuits.com/textbook/direct-current/chpt-1/electric-circuits/. (accessed: 25.05.2020).

[59] www.123rf.com. *Vector - Modern console gamepad black and white sketch vector illustration graphic design*. 2020. URL: https://www.123rf.com/photo_109789819_stock-vector-modern-console-gamepad-black-and-white-sketch-vector-illustration-graphic-design.html.

[60] Steven Yantis and Richard A Abrams. *Sensation and perception*. Worth Publishers NY, 2014.

[61] *Z-Domain - Pole Zero Plots Relationship with System Frequency Response*. URL: https://www.youtube.com/watch?v=m5TP2uG_02M. (accessed: 25.03.2020).